Report of assignment 1

Group name: RiMaYa
Rithy Sochet, Marc Kaspar, Yannis Fontaine

October 8, 2024

1 Introduction

For this assignment we were tasked to implement a recommendation system based on collaborative filtering. By using users input on different object, we try to predict the rating that an user will to an object. By doing this for every users for every object we seek to fill the database which was sparse to a dense one. In the following of the report, we will refer to the dataset of ratings as a 2-D matrix, denoted by R, where rows represent an object and columns represent an item. The matrix contains ratings going from 0 to $5(0, 0.5, 1, 1.5...5)$. We will consider that there are m users and n items so $R \in \mathbb{R}^{m\times n}$

2 Organisation of the implementation

For this project we mainly used the libraries numpy, pytorch, tensorflow to implement different models, we also used matplotlib to plot the result of our experiments.

We implemented three differents models:

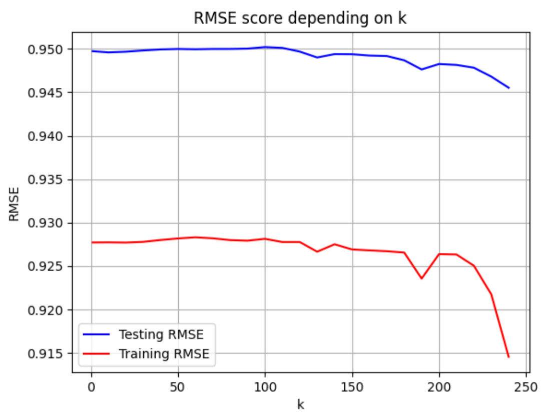
- Matrix Factorisation in the file: *matrix_factorisation.py*
- Deep Matrix Factorisation in the file: *Deep_Matrix_Factorization.py*
- Neural Collaborative Filtering in the file: *ncf.py*

We also implemented different metrics used during the project in the file: *metric.py*. Finally, we used the files *main.py* and *main2.py* to test our models and plot the results of our experiment.

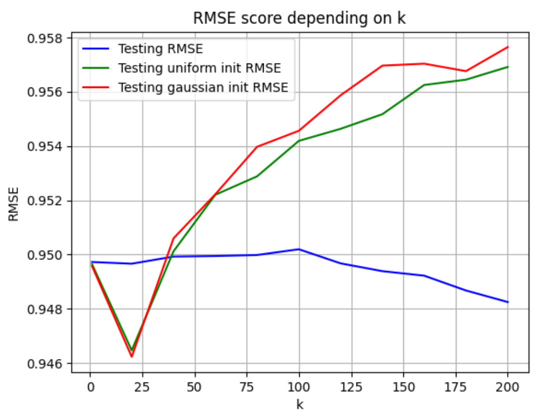
3 Matrix factorisation

Our first approach consists in estimating the rating matrix R by the product of two matrix $I \in \mathbb{R}^{m\times k}$ and $U \in \mathbb{R}^{n\times k}$, with k a hyperparameter to fix. We used Alternated Least Square (ALS) to train the model.
One try on the testing platform with the parameters: $k = 4$, $\lambda = \mu = 1$ and initialisation of U and I with only 1 gave: $RMSE = 0.943$, $accuracy = 21.81$, with $time = 1.72s$.
We experimented on multiple things: the initialisation of the matrices U and I, the regulariser λ and μ and the value of k.

- For the choice of k, our experiments showed that with $k \leq 120$ there isn't much difference. A small k around 10 might be enough to get a good score. Note that taking a big value for k increases the training time, so this is another reason not to take a bigger value if not necessary.
- For the regularisation, we tested different value for λ and μ. The tests showed that having λ and μ be different gives the best results. In particular, when $\lambda > \mu$. When both are equal and small $(= 0.1)$ we have an overfitting on the training dataset (the RMSE went below 0.4 with k around 200).



(a) RMSE with different k



(b) RMSE with different initialisation

- For the initialisation of the matrices U and I, we tried testing with matrices filled with 1 (blue curve in figure (b)), filled with random ratings from 0.5 to 5 chosen from a uniform distribution (green curve) and another one with a gaussian distribution centered at 2.5 and standard deviation at 1 (red curve). For each k we trained the model 5 times, thus giving 5 scores, the score is then the average. Overall it seems that an initialisation with only 1 gives the better results.

4 Deep matrix factorisation

For our second method, we decided to use deep matrix factorization (DMF). Our algorithm was based on the search paper [XDZ+17]. This model will use matrix factorization and neural networks to improve recommended systems. The deep neural network takes as input a user-item matrix containing ratings. The model projects users and items into a low-dimensional representation space, where interactions are computed.

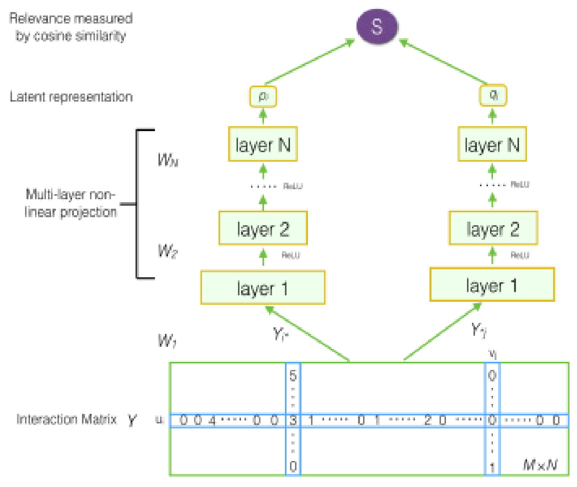


Figure 1: The architecture of Deep Matrix Factorization Models

To predict a rating for user i with regard to item j, we take the vector representing all the opinions of user i, and the vector representing all the ratings of item j.

Our 2 vectors are passed through a deep neural network using ReLU activation functions. Unlike generalized matrix factorization (GMF), which captures linear interactions between users and items in a similar way to traditional matrix factorization, our model offers a more flexible approach. The neural network captures non-linear interactions between users and items, enabling more complex patterns to be modeled.

In our architecture, the vectors resulting from the intermediate layers are projected into a latent space where the similarity between a user and an item is calculated using the cosine similarity function. As the ratings of our model vary between 0 and 5, we adjusted the output by multiplying by 5.

$$\hat{Y}_{ij} = F^{DMF}(u_i, v_j|\Theta) = cosine(p_i, q_j) * 5 = \frac{p_i^T q_j}{\|p_i\| \|q_j\|} * 5$$

By optimizing the hyperparameters we got : embedding vector dim = 64, epochs = 10, batch size = 32, learning rate = 0.001, number of layer = 2.

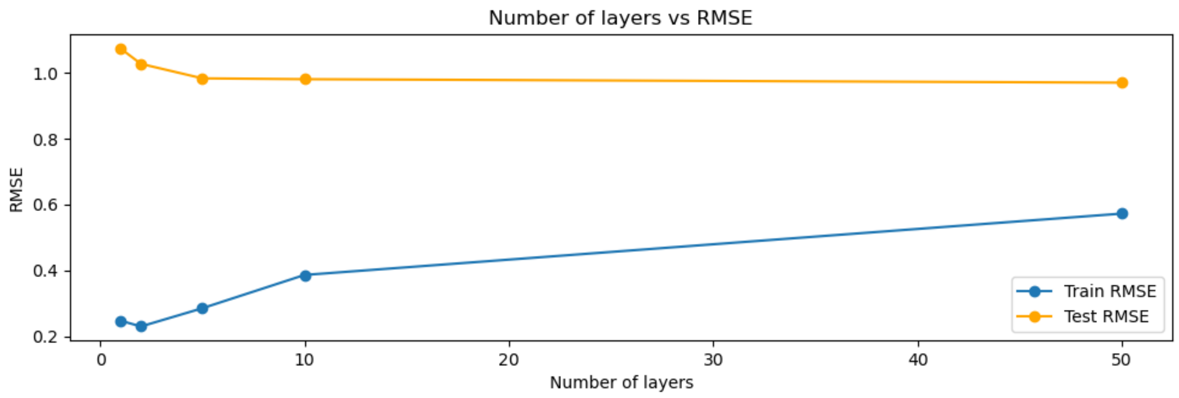


Figure 2: RMSE-train and RMSE-test depending on the number of layers for DMF

This figure is an example of the optimization we've made. We obtain an RMSE-test of 1.053 and RMSE-train of 0.230 For the evaluation, we got an RMSE-eval of 0.9 and an accuracy of 0 in a time of 520 sec. The accuracy can be explain because we didn't round off to the nearest 0.5.

Model enhancement

To improve our model, we have added genre information to our films. This provides additional information when training our model, which can then be used to predict the rating for a new film based on the person's tastes. We also rounded the score prediction to the nearest 0.5 for accuracy. Because results before this had many digits after the comma, the result can only be [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]

Before training our new model with gender information, we need to preprocess it. We tokenize our genders datas so that each gender is represented by a number. After that, we optimize again our hyperparameters :

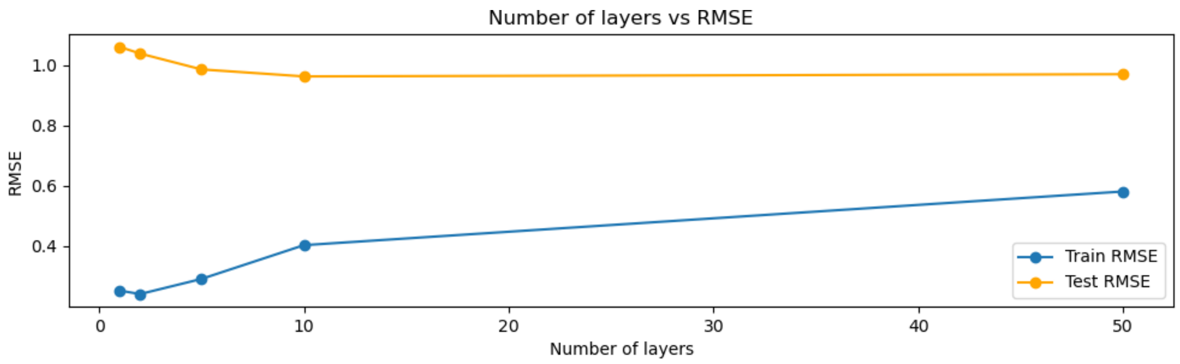


Figure 3: RMSE-train and RMSE-test depending on the number of layers for DMF with rounding and genre

This figure is an example of the optimization we've made. Note that, as in [XDZ+17], the number of optimal layers is 2 Our optimal parameters that we will use are: number of layers = 2, embedding vector dim = 64, learning rate = 0.001, epoch = 30, batch size = 32 We obtain an RMSE-test of 1.045 and RMSE-train of 0.242 For the evaluation we didn't have time to get any results on the platform due to several bugs in a row.

Method	RMSE-test
DMF	1.053
DMF with rounding and genre	1.045

Table 1: Results for DMF method

To conclude with DMF, we're not getting very satisfactory results, which can be explained by the fact that we don't have enough training data. Another thing is that the addition of genders hasn't really changed our model, which is strange, it should have improved it, there may be mistakes made on it.

5 Neural Collaborative Filtering

For our third method, we decided to use Neural Collaborative Filtering (NCF). We based our algorithm on this research article. [HLZ+17]. It is a model used for recommendation systems. It models the user and items iteractions in the latent space effectively with a neural network which allows it to capture more complex relationships between users and items. Moreover, it is a generalization of Matrix Factorization.

Figure 4 illustrates the architecture of NCF:

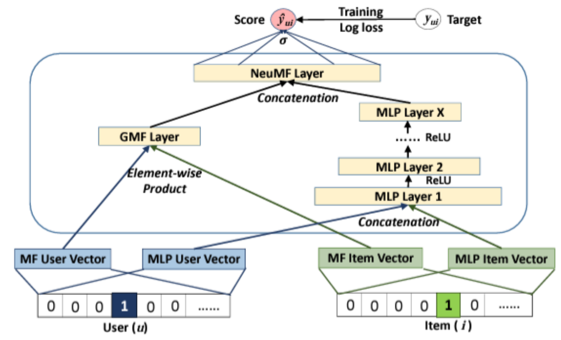


Figure 4: Basic NCF model

Users and items are represented as embeddings in a lower-dimensional space. These embeddings capture the latent features of users and items.

The information goes through a multi-layer perceptron using the ReLU activation functions and Generalized Matrix factorization layer. GMF captures linear user-item interactions, similar to traditional matrix factorization. MLP captures non-linear user-item interactions, providing flexibility to model complex patterns. The last MLP layer and the GMF layer are finally concatenated and go through a last layer called NeuMF (Neural Matrix Factorization) with a sigmoid activation function. In our model, since the ratings were between 0 and 5 we had to multiply the output by 5 to be able to get a good model.

We used Stochastic Gradient Descent with a learning rate of 0.001 to train our model. We used a batch size of 128 and 30 epochs. We minimized the RMSE loss. These parameters gave us the best results overall.

Figure 5 shows the evolution of the RMSE with a Neural Collaborative Filtering algorithm with the previous parameters.

Finally, we tried rounding the predictions to get more accurate results, table 2 displays these results.

We can see that rounding gives a lower RMSE but a higher accuracy.

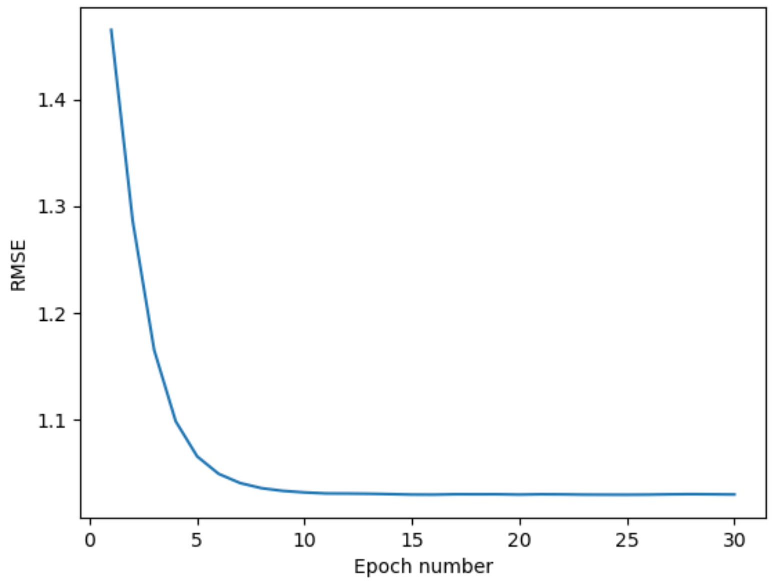


Figure 5: Evolution of the RMSE with Neural Collaborative Filtering

Method	Time (s)	RMSE	Accuracy (%)
NCF	17.22	1.0379	0.00
NCF with rounding	17.38	1.1473	26.31

Table 2: Results for NCF method

6 Conclusion and improvement

We worked on recommendation systems with collaborative filtering method such as matrix factorisation and methods involving neural networks. It allowed us to have a first try at implementing some architecture of neural network with library like pytorch and tensorflow to solve a difficult problem. However, with three different methods, it was difficult to get a lot of experiment done and thus giving not as many results to discuss as we would have liked. In further project, it might be better to try less method and focus on trying different kinds of tests to improve the implemented models.

Some interesting things that we discussed during the project were the efficiency of linear model like matrix factorisation against some neural network. Indeed, a simple model fine tuned gives better result with less resources (in time and computational means) than a neural network not well tuned. This really showed the importance of spending time experimenting with simple model and possible improvements rather than immediately exploring the larger and more complex models.

References

[HLZ$^+$17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[XDZ$^+$17] Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th international conference on world wide web*, pages 3203–3209, 2017.