Dauphine | PSL
UNIVERSITÉ PARIS

Data Science Lab

Assignment 1 : Collaborative Filtering

Auteurs

Barthélémy BREGEON
Maïssa ELAFANI
Sakula HYS

IASD - 2024
Université Paris-Dauphine

1 Introduction and approach

Recommendation Systems are among the most successful applications of Machine Learning, widely used by platforms like Netflix, YouTube, and Yelp to suggest movies, videos, or services that we might enjoy. In the context of Assignment 1 for the Data Science Lab in the IASD master's program, we applied the core principles behind these recommendation systems, with the goal of designing and implementing one or more systems utilizing collaborative filtering techniques.

To address this problem, we first decided to use the matrix factorization method, and then optimized this approach by incorporating additional techniques.

In this report, you will find an overview of how we implemented matrix factorization, including a formula that add a bias. We will also describe the preprocessing and post-processing steps we took, followed by an explanation of how we integrated feature addition into the previous methods.

2 Stochastic Gradient Descent with added bias

2.1 Principle of the algorithm

For this approach, we, first, consider a rating matrix R, where r_{ui} is the rating of user u for movie i. We approximate R by decomposing it into two matrices.

$$R \approx X.Y^T \tag{1}$$

where X (user-feature matrix) and Y (movie-feature matrix) capture latent factors and we have the following predicted rating :

$$\hat{r}_{ui} = x_u.y_i^T \tag{2}$$

By analyzing the input data in parallel, we noticed that certain trends could be "predicted," such as the tendency for users to give ratings above the average rather than below.
After some research, we choose to incorporated these trends as biases and added them to the predicted rating. Including bias, the prediction becomes :

$$\hat{r}_{ui} = x_u \cdot y_i^T + (\mu + b_u + b_i) \tag{3}$$

where μ is the global average rating, and b_u and b_i are the user and movie biases which are specific to, respectively, each row and each column.

The loss function, incorporating regularization becomes :

$$L = \sum_{ui} \left(r_{ui} - \left(\mu + b_u + b_i + x_u \cdot y_i^T\right)\right)^2 + \lambda \left(||b_u||^2 + ||b_i||^2 + ||x_u||^2 + ||y_i||^2\right) \tag{4}$$

And by applying the gradient descent to each weight, the updates for the parameters are :

$$b_u \leftarrow b_u - \eta \left(e_{ui} - \lambda b_u\right) b_i \leftarrow b_i - \eta \left(e_{ui} - \lambda b_i\right) x_u \leftarrow x_u - \eta \left(e_{ui} y_i - \lambda x_u\right) y_i \leftarrow y_i - \eta \left(e_{ui} x_u - \lambda y_i\right) \tag{5}$$

where e_{ui} is the prediction error $e_{ui} = r_{ui}$ - $\hat{r}_{ui}$

2.2 Results

The results of this model are promising. In the best case, we achieved an **RMSE of 0.844** and an **accuracy of 29%**. We conclude that our model fits reasonably well compared to the baseline model and is capable of predicting user preferences effectively. As indicated by the RMSE, our predictions are very close to the actual values, although they are not identical. Therefore, the ideal role of our model would be in movie recommendations, which aligns with our initial goal.

However, given the low accuracy achieved with this model, we cannot ascertain the exact ratings of users, which limits its optimal use for this type of task. If the initial requirements change, we will need to readjust the model accordingly.

3 Data processing for better accuracy

Upon inspection of the grade distribution, we noticed that the distribution was obviously not uniform. In particular, half integers are way less frequent than integer grades. We noticed that our model always gave distributions which had to many half integer grades when rounded to the nearest half integer or integer (Figure 1).

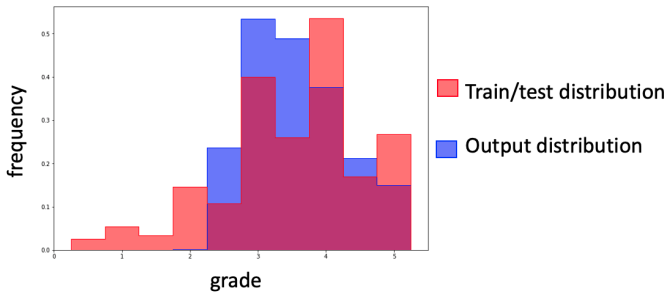

Figure 1: Original distribution

To compensate this effect we had the idea to allocate more space along the grade axis for grades that are more frequent in the distribution. To do this, we calculate the intervals proportional to the frequency of each grade, modify the input matrix accordingly, each grade is modified to be at the center of the interval allocated to it along the grade axis (see Figure 2). In short we normalize the grade distribution to be constant. After training the model we round the grades with respect to the intervals previously allocated. This gave an output distribution that was closer to the input (Figure 3) We also tried normalizing the input distribution with more complicated functions to get an output distribution even closer to the input. The one that got us the colsest is a gaussian function (Figure 2).

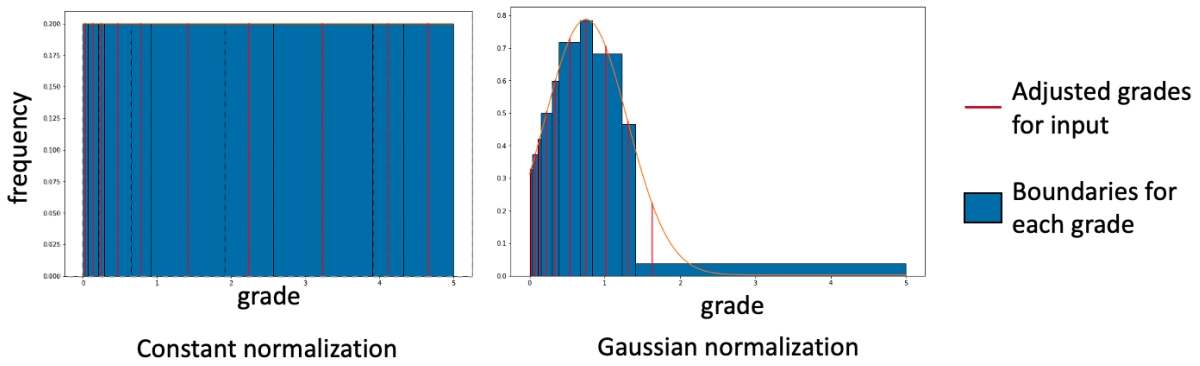

Figure 2: Normalizing functions

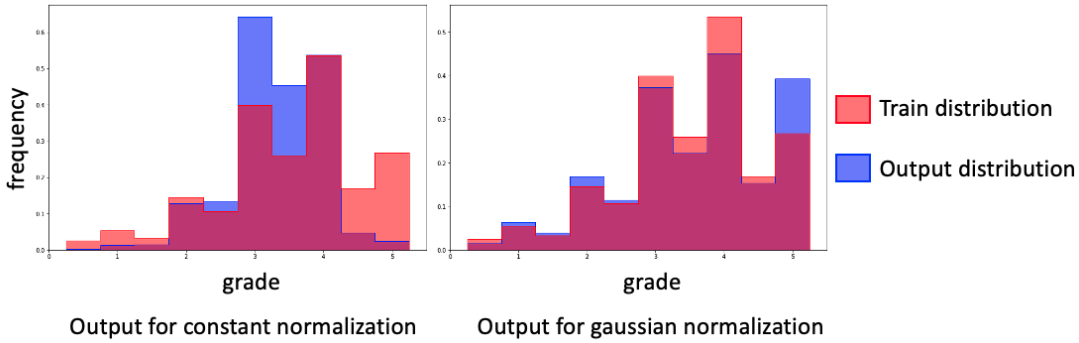

Figure 3: Distributions after input normalization

In terms of accuracy, the constant normalization was a significant improvement over rounding up to the closest half integer or integer but the gaussian normalization gave bad results. We think that this is due to the fact that the training data is too distorted in this case, in particular the lower grades are very close to each other. The output distribution we get is artificially close to the training distribution but the model didn't really learn anything.

Distribution	Accuracy
initial	25.34%
constant	28.66%
gaussian	20%

4 Additional features

Collaborative filtering is all based on recommending an item to an user according to their similarities with other users. But the item itself has characteristics that might influence its choice over another one. Intuitively, it then seems appropriate to consider these characteristics in predicting the ratings.

The dataset includes a table with the different genres for a movie, its title, and year of release. We decided to take into account the genres and year, since basic statistics didn't show interesting contribution of the titles in the ratings.

4.1 Approach

FeatureMF [1] is an approach that takes this kind of features into account Let N the number of features in total, n_t the number of all possible values for feature of index $t \in \{1, ..., N\}$. For each item i, we can represent its features in a table :

Features F_t	**Attributes** α_t
year	2000
genres	Comedy, Drama, Action, Children

Table 1: Features for item i

FeatureMF considers adding the information on the features for each item i in the latent space of Matrix Factorization. In practice, it consists in 'replacing' the item factor Y_i in the Matrix Factorization by adding a feature term $\mathcal{F}(i)$ to it. For each feature, each attribute $\alpha_t(a)$ is projected onto the latent space as a vector, where $a \in \{1, ..., n_t(i)\}$ the number of possible values for feature t. The matrix in which each row corresponds to a projected attribute is noted y_t. The feature term for each item i is calculated as follows :

$$\mathcal{F}(i) = \sum_{t=1}^{N} \frac{1}{n_t(i)} \sum_{a} y_t(a)$$

This term depends solely on the item, and represents the vector sum of the averages for every feature in the latent space. The objective function and learning are modified, with a regularization term on the features :

$$\tilde{r}_{ui} = X_u (Y_i + \mathcal{F}(i))^\top$$

$$f = \sum_{(u,i)} \left((r_{ui} - \tilde{r}_{ui})^2 + \lambda(\|X_u\|^2 + \|Y_i\|^2 + b_u^2 + b_i^2 + \sum_{t=1}^{N} \sum_{a} \|y_t(a)\|^2) \right)$$

4.2 Interest

FeatureMF caught our attention as an efficient way to include additional features to matrix factorization. This approach is interesting for several reasons :

- scalability : learning the features in the latent space rather than including them in the similarity function helps to handle large data. However, our implementation was suboptimal and resulted in high computation time.
- sparsity : Zhang et al. [1] state that FeatureMF performs better than classic Matrix Factorization when the data is even sparser. This was not verified due to lack of time.

Algorithm 1: Learning the FeatureMF Model

Input : $R, \lambda, \alpha, \gamma$ (*learning rate*), d,c, $iter \leftarrow 0$
Output: rating predictions $\tilde{r}_{ui}$

Initialize the low-rank matrices for users (P), items (Q) and features ($Y = \{y_t | t = 1..N\}$), the bias vectors for users (BU) and items (BI)
while *iter* $<$ *maxIter or error on validation set decrease* **do**
  while $(u, i) \in K$ **do**
    $\tilde{r}_{ui} \leftarrow P_u^T (Q_i + \sum_{t=1}^{N} |F_t(i)|^{-1} \sum_{a \in F_t(i)} y_t(a)) + b_{ui}$
    $e_{ui} \leftarrow \tilde{r}_{ui} - r_{ui}$
    $BU_u \leftarrow BU_u - \gamma(e_{ui} + \lambda \cdot BU_u)$
    $BI_i \leftarrow BI_i - \gamma(e_{ui} + \lambda \cdot BU_i)$
    $P_u \leftarrow P_u - \gamma(e_{ui}(Q_i + \sum_{t=1}^{N} |F_t(i)|^{-\alpha} \sum_{a \in F_t(i)} y_t(a)) + \lambda \cdot P_u)$
    $Q_i \leftarrow Q_i - \gamma(e_{ui} P_u + \lambda \cdot Q_i)$
    foreach $y_t \in Y$ **do**
      foreach $a \in F_t(i)$ **do**
        $y_t(a) \leftarrow y_t(a) - \gamma(e_{ui} |F_t(i)|^{-1} P_u + \lambda \cdot y_t(a))$
      end
    end
  end
  $iter \leftarrow iter + 1$
end

- cold-start problem : the feature term added to the learning process only depends on each item i, therefore, even when a user has rated few items, ratings can be predicted based on the items' characteristics

4.3 Results

The results for this method could not be tested on the platform (path problem for the features), but the metrics computed locally gave similar yet higher RMSE and lower accuracy compared to our approach without the features.

The tradeoff between RMSE and accuracy suggests a value of k of 7 in this range of k tested. The more we increase k, the closer the model gets to the baseline. We cannot say that the model performs better, and as stated previously, computation on the feature term could be optimized, and hyperparameters tuned further.

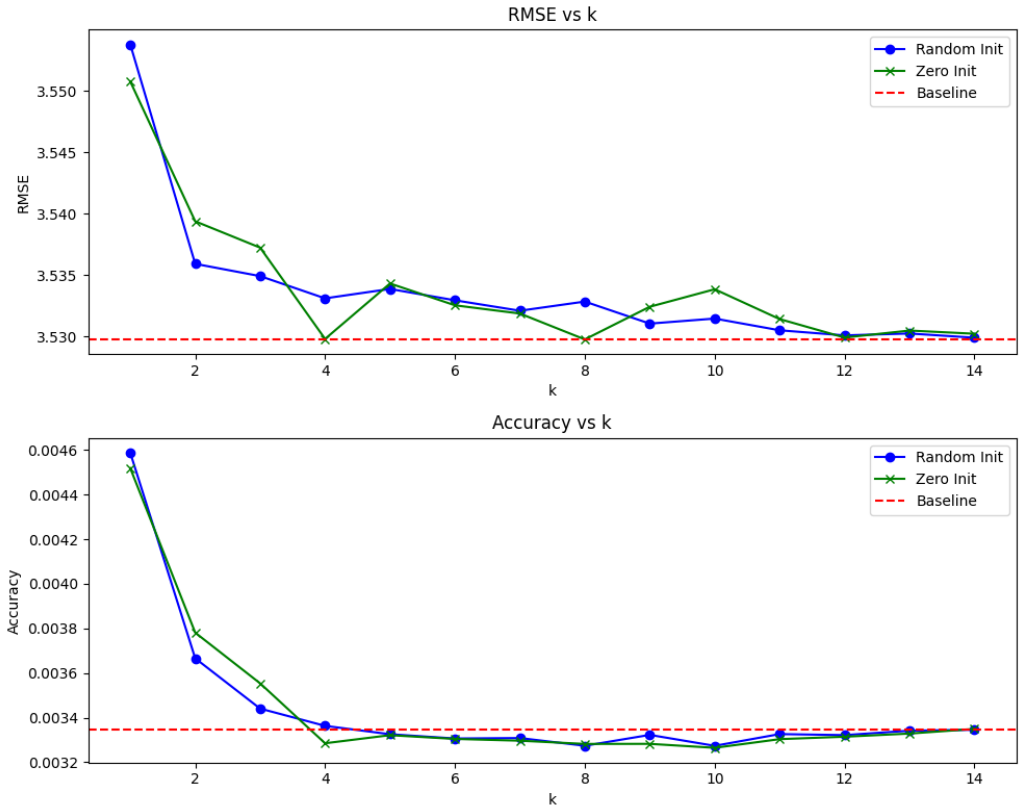

Figure 4: Comparison of the RMSE and accuracy of the model for different values of k the latent space dimension and different initialization of the features. The baseline corresponds to our approach without including the features

5 Conclusion

In conclusion, matrix factorization offers numerous advantages. It is relatively straightforward to implement and executes quickly. Its applications extend beyond recommendation systems, as we have seen and experienced and with a coherent selection of features, it allows for a reduction in the overall dimensionality of the data. However, this approach also has its limitations, such as cold start issues.

To address these challenges, alternative methods, such as content-based models, could be explored. Experimenting with these models would provide valuable insights into their differences and potential effectiveness. Moreover, as the field of machine learning continues to evolve, integrating hybrid approaches that combine various techniques could lead to even more robust solutions, enhancing user experiences and improving predictive accuracy across diverse applications.

References

[1] Haiyang Zhang, Ivan Ganchev, Nikola S. Nikolov, Zhanlin Ji, and Máirtín O'Droma. Featuremf: An item feature enriched matrix factorization model for item recommendation. *IEEE Access*, 9:65266–65276, 2021.