Collaborative Filtering

Marc Schachtsiek[*], Christian Kayo[†], Nicolas Durat[‡]

[1]*Data Science Project, Université Dauphine-PSL, 75016 Paris, France*

October 16, 2022

Abstract

The objective of this report is to use matrix factorization (MF) for collaborative filtering in a recommender system and gain familiarity with different approaches. These techniques factorize a rating matrix into user and item embeddings and can be seen as a regression-type optimization problem. Three different techniques were investigated - naive gradient descent as a baseline, Non-Negative Matrix Factorization (NMF), and Sparse Principal Component Analysis (SPCA). However, results for SPCA were not achieved due to the project time constraint and implementation problems. For NMF, the first step was a comparative investigation of different implementations, followed by a detailed look at the optional orthogonality constraint and its effects and a comparison against the baseline. It was found that NMF does not exceed the baseline performance but that, contrary to initial beliefs, the additional orthogonality constraint had no negative effect on the quality of the embeddings while providing better interpretability.

Keywords: Recommendation systems, Collaborative Filtering, Matrix Factorization

Contents

[*]E-mail: marc.schachtsiek@dauphine.eu
[†]E-mail: christian.kayo@dauphine.eu
[‡]E-mail: nicolas.durat@dauphine.eu

1 Context and Problem Statement

For collaborative filtering in a recommender system, the general principle is to represent the ratings in a Users $\times$ Items matrix. The correlations between the ratings can be exploited to express this matrix as the product of two matrices $U \in \mathbb{R}^{m\times d}$ (the user-embedding matrix) and $V \in \mathbb{R}^{n\times d}$ (the item-embedding matrix) of lower rank. The resulting matrices are made up of vectors that are called latent factors and combine in the following form $UV^\top = \hat{A} \approx A$. The objective is to minimize the cost function $C(A, \hat{A})$, which is composed of the Frobenius norm of the difference between the matrix A and the product of the matrices U and $V^\top$, to which regularisation terms or additional constraints can be added.

$$C(U, V) = \|A - UV^\top\|_F^2 + \lambda\|U\|_F^2 + \mu\|V\|_F^2 \tag{1}$$

with $U \in \mathbb{R}^{m\times d}$, $V \in \mathbb{R}^{n\times d}$ and $\|X\|_F = tr(X^\top X)$. To minimize this function, we will use several optimization algorithms, for which we will compare implementation performance, hyperparameters, possible additional constraints and quality of solutions.

The primary datasets used were the MovieLens datasets provided by GroupLens: MovieLens Dataset. The GroupLens research group has been collecting movie ratings from volunteers since 1995 and has produced datasets of various sizes. The results presented are for the 25M dataset.

2 Approaches & Implementations

2.1 Baseline (Gradient Descent)

2.1.1 Presentation

Gradient descent is an iterative optimization algorithm to find the minimum of different functions [1]. Its principle is as follows: at each iteration, we take the opposite direction of the gradient of the function C to be optimized, until convergence to either a local or global minimum depending on the convexity of the function.

2.1.2 Implementation

During training the gradient descent allows to update the coefficients of the function C such that:

$$U^{(t+1)} = U^{(t)} - \eta^{(t)} \frac{\partial C}{\partial U}(U^{(t)}, V^{(t)}) \qquad V^{(t+1)} = V^{(t)} - \xi^{(t)} \frac{\partial C}{\partial V}(U^{(t)}, V^{(t)}) \tag{2}$$

Where $U^{(t+1)}$, $U^{(t)}$ and $V^{(t+1)}$, $V^{(t)}$ are consecutive points and $\eta^{(t)}$, $\xi^{(t)}$ the learning rates, the direction of the steepest slopes.

2.2 Non-Negative Matrix Factorization (NMF)

2.2.1 Presentation

The first approach implemented was Non-Negative Matrix Factorization (NMF) [2]. It is a commonly used matrix factorization technique that requires a non-negative input matrix ($M \in \mathbb{R}_{\geq 0}^{m\times n}$) [1]. It produces a basis matrix ($W \in \mathbb{R}_{\geq 0}^{m\times r}$) and an encoding matrix ($H \in \mathbb{R}_{\geq 0}^{r\times n}$) that additively combine to a parts-based representation to recover the input matrix as closely as possible. NMF can be extended with an orthogonality constraint imposed through a penalty. It has been shown to increase interpretability by enhancing an inherent clustering property of NMF. It is referred to as Orthogonal NMF (ONMF) with the following optimization statement:

$$\min_{W\geq 0 H\geq 0} \|M - WH\|_F^2 + \mu\|HH^\top - I\|_F^2 \tag{3}$$

[1] The matrices have been labelled according to NMF literature. For Equation 1: $M = A$, $W = U$, $H = V^\top$

The problem has been shown to be non-convex, and finding an exact solution has been proven to be an NP-hard problem. No algorithm guarantees global optimality but fixing one matrix while computing the other leads to an efficiently solvable problem with a "good enough" solution.

2.2.2 Implementation

The most common method for solving NMF-type problems is the multiplicative update (MU) rule algorithm. The final MU rules are stated below. In this case, the update of W precedes the update of H, hence the use of $W^{(t+1)}$ in the H-update step:

$$W^{(t+1)} \leftarrow W^{(t)} \times \frac{MH^{(t)^\top}}{W^{(t)}H^{(t)}H^{(t)^\top}} \qquad H^{(t+1)} \leftarrow H^{(t)} \times \frac{W^{(t+1)^\top}M}{W^{(t+1)^\top}W^{(t+1)}H^{(t)}} \tag{4}$$

They are applied by fixing W and H for the computation of H and W, respectively. This algorithm can be extended and computationally accelerated by computing multiple updates for each step. This accelerated MU (AMU) algorithm requires multiple computationally expensive matrix computations to be computed only once for several updates.

Another algorithm is the Hierarchical Alternating Least Squares or HALS. It is also based on updating one matrix while fixing the other but updates the matrices column-wise for W and row-wise for H, thereby including newly calculated columns or rows in the same update step. The same acceleration of AMU can also be applied to HALS (AHALS).

Both algorithms and their accelerated versions were implemented, and the MU algorithm was extended with the orthogonality constraint of ONMF [3]. All algorithms have been built to accept a matrix M with missing data by evaluating the error function only on observed data points. The algorithms have been implemented in Python and use the NumPy and Scipy.sparse packages:

- NumPy: Linear algebra library for matrix multiplication, norms and other linear algebra functions
- Scipy.sparse: Implementations of sparse matrices in CSR, CSC and COO formats. These are used to represent the missing data rating matrix in a space and computationally efficient format.

2.3 Sparse Principal Component Analysis (SPCA)

2.3.1 Presentation

Sparse Principal Component Analysis (SPCA) is a dimension reduction technique that constructs principal components and adds sparsity to the factorized matrices. Its objective is to maximize the variance along a vector and to reconstruct the PCA with a few inputs [4]. It can be seen as a regression optimization problem using an elastic net penalty composed of lasso and ridge regularization. The lasso (L1) regularization reduces the coefficients towards zero and selects the variables to produce a sparse model. The ridge (L2) regularization ensures the reconstruction of principal components. It erases the drawbacks of the lasso regularization by selecting all possible variables and not only variables limited to the number of observations. The optimization problem is as follows:

$$\arg\max_{U,V} \|A - AUV^\top\|_F^2 + \lambda_1 \sum_{j=1}^{k} \|v_j\|^2 + \lambda_2 \sum_{j=1}^{k} \|v_j\| \tag{5}$$

The objective is to minimize the Frobenius norm by adding two penalty terms, each with a regularization parameter. The higher the parameter, the larger the penalty and, consequently, the smaller the magnitude of the coefficients.

2.3.2 Implementation

Two SPCA versions were planned to be implemented: SPCA with Elastic Net [5], and GPower [6] (only Lasso regularization). The algorithms consist of four steps:

- Update V (PC loading) given fixed U (PC weights) by elastic net / lasso gradient descent regression

- Update U (PC weights) given fixed V (PC loading) by singular value decomposition (SVD)
- Repeat the first two steps until convergence
- Eigenvectors (PC axes) ridge normalization

Each part of the algorithm was implemented as closely as possible, as outlined in the original paper [5]. However, combining the individual parts proved more difficult than initially expected. This was compounded by two separate facts that eventually led to focusing on a more in-depth analysis of NMF and implementing GD as a baseline.

Firstly, adapting the algorithm for a matrix with missing data would have required an entirely different implementation of an SVD-like technique - essentially another special case matrix factorization issue with multiple algorithms and techniques to select, compare and implement.

Secondly, with sparse input data, enforcing sparsity in the output would have likely resulted in significantly poorer model performance. SPCA would be more appropriate to tasks where a further understanding of the principal components is the primary objective, e.g., correctly identifying root causes rather than most accurately predicting an outcome. [7]

3 Procedure

3.1 Initial Goal

The overall goal was to produce a matrix factorization to minimize the following loss $\|A - UV\|_F^2$. While this is a common goal for matrix factorization, additional requirements are usually looked for, such as interpretability of the output, computational complexity, or guarantees of a global minimum.

It was decided to use GD as a baseline model and evaluate the performance of the other methods against it. In addition to base performance, the effect of the orthogonality constraint for NMF was investigated, including the prospect of better interpretability of the output. For SPCA, the initial plan was to investigate how regularization affects performance. Since the implementation was not finalized for the aforementioned reasons, there are no comparable results for this technique.

3.2 Baseline

To set the baseline of this report, gradient descent was run with several parameters for r and a learning rate that is halved every 20 iterations for fast but continuous improvement. The learning rate was determined empirically. The graph in Figure 1 shows the experimental results. The optimal rank r for Gradient Descent was around 30, and the final test error was about 1.25. For $r = 15$ a slightly lower learning rate had to be used as it initially stopped after just a few iterations.

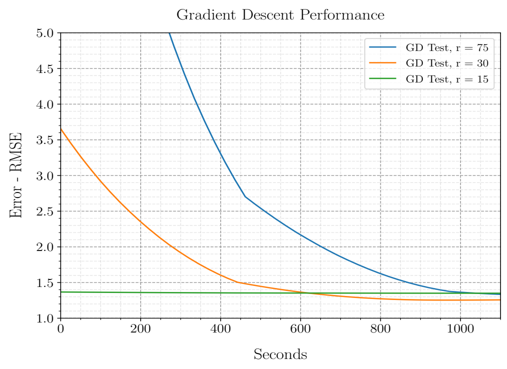

Figure 1: Gradient Descent performance

3.3 NMF Experiments

For the first experiment, the implementations of NMF were compared. Specifically, the relative convergence rates, the computational complexity and the overall final performance were of interest. Figure 2 displays the results of the four algorithms with the same initialization for W and H. As expected, HALS outperforms MU and the accelerated versions outperform their counterparts. After enough training, the performance of AMU compared to either HALS implementation becomes negligible. Additionally, this first result shows that NMF does not outperform the baseline model with a test RMSE more than 2x as high. Likely, the constraint of non-negativity limits the possible output space too much.

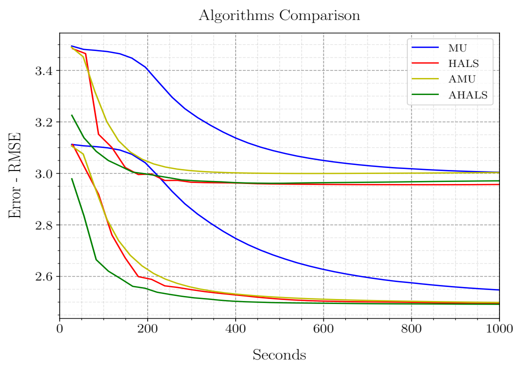

Figure 2: NMF relative algorithm performance

The graphs shown in Figure 3 display the results of the rank estimation for the dataset. It shows that the optimal rank is $r \approx 80$. Since NMF provides only a local minimum, the optimal rank depends on the initialization of W and H. 4 shows an narrowed average over 5 different initializations. The lowest average was observed at $r = 75$. However, almost every other rank r has a model that outperforms that average. This and the range of RMSE demonstrate that choosing the exact rank is not imperative for this problem and dataset - any rank around 75 or 80 will represent the performance of NMF on this task. For any other NMF experiments, the rank was fixed on 75.

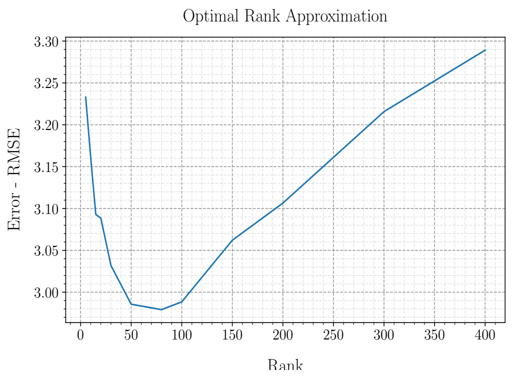

Figure 3: Overall rank approximation

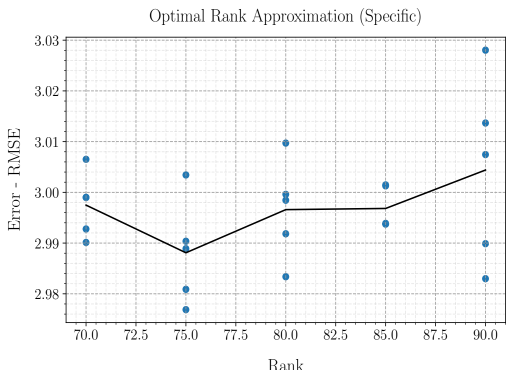

Figure 4: Narrowed rank approximation

The graph shown in Figure 5 displays the prediction performance of ONMF against NMF. The performance of the ONMF algorithm is essentially identically to NMF. This result was surprising, as the expectation was that an additional constraint would lead to a higher error.

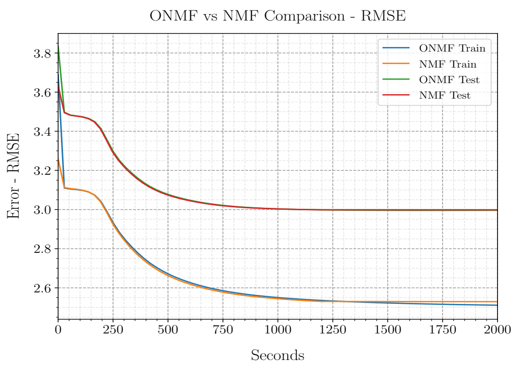

Figure 5: Performance of ONMF vs NMF

For the orthogonality evaluation, the following error was calculated for both H matrices, with I being the correctly sized identity matrix. $Orth(H) = \|HH^\top - I\|_F$. The results are $Orth_{ONMF}(H) = 8.66$ and $Orth_{NMF}(H) = 38200417$. This result clearly indicates much-improved orthogonality without any loss in performance.

Due to the project's time limit, further interpretability could not be fully investigated. There are several sources listed here that provide literature support of the conclusion that orthogonality leads to better interpretability and steps can be followed to analyze and visualize this more in depth ([8], [9]).

4 Conclusion

Overall, the experiments showed that Non-Negative Matrix Factorization does not outperform naive Gradient Descent on this matrix factorization problem. Gradient descent achieved lower train and test RMSE without any significant performance differences. One possible next step could be implementing projected gradient descent to solve the NMF optimization problem for a direct comparison.

The most surprising result of this set of experiments was the equal performance of NMF and ONMF. Generally, the orthogonality of the encoding matrix reinforces the clustering property of NMF, which leads to better interpretability of the latent factors of the H matrix. The additional constraint was expected to lead to lower performance, but this is not the case here. One explanation of this behaviour could be that many different solutions W and H are equally good factorizations of the given matrix. An additional in-depth investigation into ONMF vs NMF on different datasets and extreme cases could be performed to get more insight into the effects of the orthogonality constraint.

4.1 Takeaways

In addition to learning about different methods of solving matrix factorization, the benefits, disadvantages and difficulties, the concept of alternatingly minimizing different variables were strongly reinforced. NMF was investigated from the ground up, and while SPCA was not fully implemented, a lot was taken away from learning about the technique and attempting to solve any issues. Implementing algorithms from scratch that have otherwise only been used through libraries like $scikit-learn$ was decidedly beneficial.

Lastly, one particular challenge of this project led to a much better understanding of all of the concept and implementation parts of these algorithms - the missing data and the size of the dataset. Any implementation details must be verifiably working with the sparse matrix formats or manually adapted to accept them. Being unable to circumvent these problems by putting a representative value in for any missing ones forced a much deeper understanding of all parts of this project.

References

[1] Sebastian Ruder. An overview of gradient descent optimization algorithms. 2016. doi: https://doi.org/10.48550/arXiv.1609.04747.

[2] Daniel Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–91, 11 1999. doi: 10.1038/44565.

[3] Andri Mirzal. A convergent algorithm for orthogonal nonnegative matrix factorization. *Journal of Computational and Applied Mathematics*, 260:149–166, 2014. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2013.09.022.

[4] Juan C. Vera Rosember Guerra-Urzola, Katrijn Van Deun and Klaas Sijtsma. A guide for sparse pca: Model comparison and applications. *Psychometrika*, 86(7):893–919, 2021. doi: https://doi.org/10.1007/s11336-021-09773-2.

[5] Trevor Hastie Hui Zou and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. doi: https://doi.org/10.1198/106186006X113430.

[6] Michel Journee Yurii Nesterov, Peter Richtarik and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(15):517–553, 2010. doi: https://doi.org/10.48550/arXiv.0811.4724.

[7] Shriram Gajjar, Murat Kulahci, and Ahmet Palazoglu. Use of sparse principal component analysis (spca) for fault detection. *IFAC-PapersOnLine*, 49(7):693–698, 2016. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2016.07.259.

[8] Paul Fogel, Yann Gaston-Mathé, Douglas Hawkins, Fajwel Fogel, George Luta, and S. Stanley Young. Applications of a novel clustering approach using non-negative matrix factorization to environmental research in public health. *International Journal of Environmental Research and Public Health*, 13(5), 2016. doi: 10.3390/ijerph13050509. URL `https://www.mdpi.com/1660-4601/13/5/509`.

[9] Keigo Kimura, Yuzuru Tanaka, and Mineichi Kudo. A fast hierarchical alternating least squares algorithm for orthogonal nonnegative matrix factorization. In *Proceedings of the Sixth Asian Conference on Machine Learning*, volume 39 of *Proceedings of Machine Learning Research*, pages 129–141. PMLR, 26–28 Nov 2015.