Collaborative Filtering for Movie Recommendations

Data Science Lab - Assignment 1

GreGre

Louis Carron, Filomène Roquefort, Jules Roques

October 2025

1 Introduction

The goal of this project is to predict user movie ratings by implementing and comparing several collaborative filtering algorithms. Given a highly sparse user-item matrix from the MovieLens dataset (about 98% missing entries in the train dataset), we aim to find the most effective matrix completion strategy.

This report is structured as follows. Section 2 details our implementation of matrix factorization via GD and ALS. Section 3 describes our work on Iterative PCA and its nonlinear extension, Kernel PCA, with a focus on the challenges posed by sparse data. Section 4 discusses the crucial role of data normalization. Section 5 presents our experimental setup and compares the performance of all models. Finally, Section **??** summarizes our findings and the key lessons learned.

2 Core Methods: Matrix Factorization

The primary approach to collaborative filtering is low-rank matrix factorization, which approximates the user-item rating matrix $R \in \mathbb{R}^{n \times d}$ as the product of two smaller latent factor matrices, $U \in \mathbb{R}^{n \times k}$ (user factors) and $I \in \mathbb{R}^{d \times k}$ (item factors), such that $R \approx UI^T$. A key advantage of these methods is their ability to work directly on the sparse data by optimizing a loss function only on the observed ratings Ω.

To minimize the regularized square loss, we implemented both *Alternating Least Squares* (ALS) and *Gradient Descent* (GD) methods. GD iteratively updates the elements of I and U taking steps in the direction of the negative gradient of the regularized squared error loss function on known data. *Implementation.* Our ALS implementation alternates between solving the least-squares problem for one factor matrix and the other. This process is repeated until convergence. *Implementation.*

3 Alternative Method: Iterative PCA and Its Extension

As an alternative, we explored methods based on Principal Component Analysis (PCA), which required adapting the algorithm to handle missing data through an iterative imputation scheme.

3.1 Iterative PCA

The standard PCA cannot run on a matrix with missing values. We therefore implemented an iterative EM-style algorithm. *Implementation.*

1. **Impute:** Fill all missing entries of R with an initial guess (e.g., item means).
2. **Reduce (SVD):** Perform SVD on the completed matrix and keep the top-k components to get a low-rank approximation $\hat{X}$.
3. **Update:** Replace the previously missing entries with new predictions from $\hat{X}$, while keeping the original known values fixed.
4. **Repeat** until convergence.

3.2 Extension: Iterative Kernel PCA

To capture potential non-linear relationships in user preferences, we extended the iterative framework to use Kernel PCA (KPCA).

The Pre-Image Problem. A significant challenge in KPCA is that the mapping to the high-dimensional feature space is a one-way function (in fact we don't even know the function, just the inner product between two of its images in the higher-dimensional space). To reconstruct the ratings from the abstract non-linear features, we must solve the "pre-image problem". Following the approach described by Honeine & Richard (2011) [2], we trained a regularized linear model (Ridge regression) to learn the approximate inverse mapping from the feature space back to the rating space. This allowed us to generate the predictions needed for the "Update" step of our iterative algorithm. *Implementation*

4 Data Pre-processing & Normalization

Data normalization is a critical pre-processing step. We experimented with several strategies for both initializing our iterative models and for preparing the data for all algorithms:

- **Item Mean:** Centering data by subtracting the mean rating for each movie. This removes item bias (e.g., universally acclaimed movies).
- **User Mean:** Centering data by subtracting each user's mean rating. This removes user bias (e.g., users who rate everything high or low).

Experimental results will demonstrate the impact of different normalization configurations in the case of the Iterative PCA, showing that this choice can influence the performance. For other models, we restricted ourselves to **user mean normalization** for the final results shown in the next sections, due to time constraints.

5 Experiments and Results

5.1 Experimental Setup

We used the provided MovieLens dataset, and mainly used cross-validation for our experiments. We merged the train and validation sets, and we split the whole into 3 folds. At each round, all models were optimized on the training set and evaluated on the validation set. The final evaluation was done by taking the mean of the evaluations for the different folds.

5.2 The Impact of Normalization

This section propose to demonstrate the impact of normalization on the **Iterative PCA**'s goodness-of-fit. We experimented different configurations and validation RMSE results are shown in the following Figure 1 :

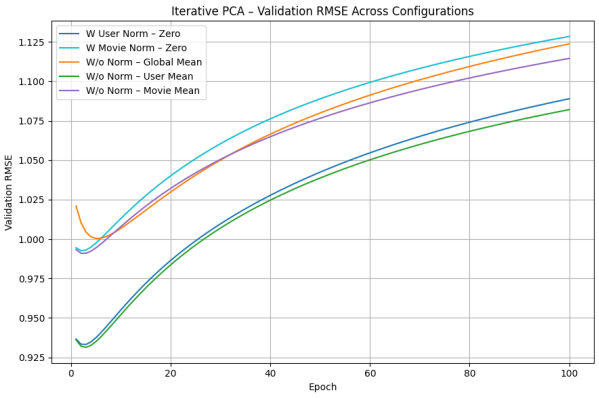


Figure 1: Validation RMSE across normalization strategies for the Iterative PCA model. (W: with, W/o: without)

As expected, the evolution of the validation RMSE is similar between the configuration using user mean normalization with zero initialization and the one without normalization but initialized with user mean truncation, with the latter showing a slight performance advantage. The plot demonstrates that the bias introduced by user ratings is more significant than that introduced by movie ratings. Consequently, the Iterative PCA models initialized or normalized using user debiasing outperform those using movie debiasing by approximately 0.05 RMSE.

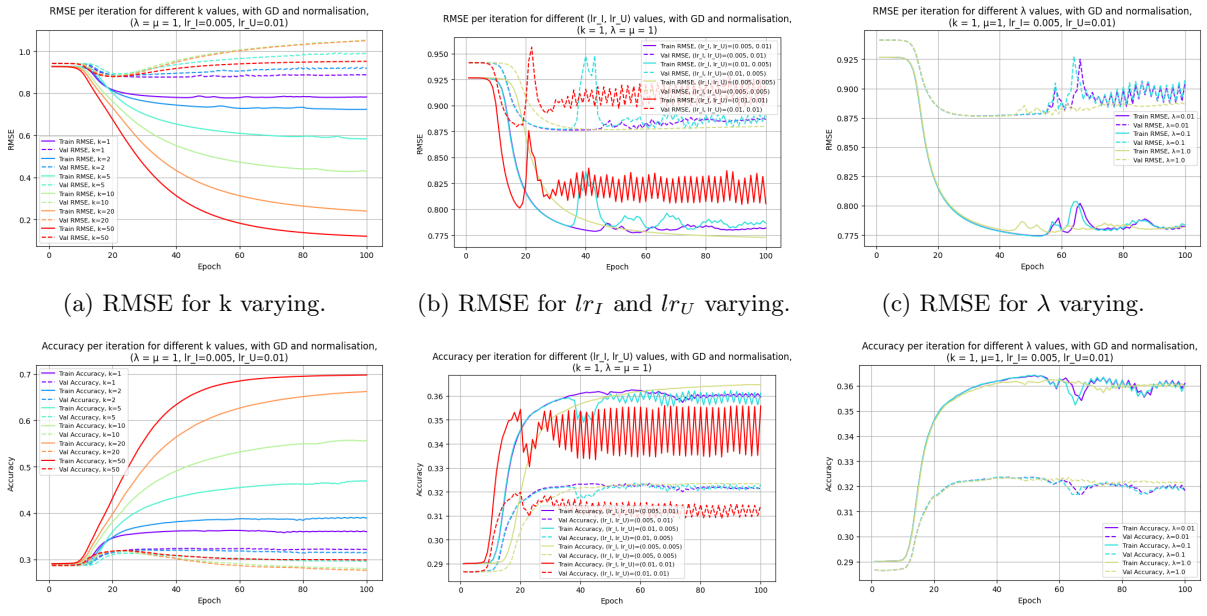
5.3 Hyperparameter Tuning

We performed a hyperparameter search using a grid search methodology. For each algorithm, we defined a grid of plausible parameter values and trained the model on the training set for every combination. The combination yielding the lowest RMSE on the validation set was selected for the final model comparison.

The key hyperparameters tuned for each model were:

- **Matrix Factorization with ALS & GD:**
  - Number of latent factors (k): We tested values in $\{1, 2, 5, 10, 20, 50\}$.
  - Regularization parameters (λ, μ): For each, we searched over $\{0.01, 0.1, 1, 10\}$, depending on whether the input data were normalized for training or not.
  - For GD we also tuned the learning rate for the U matrix and the learning rate for the I matrix. We took the values in $\{0.001, 0.005, 0.01\}$

  The best combination we found for GD was $k = 50$, $\lambda = \mu = 1.0$, $lr_U = 0.01$ $lr_I = 0.005$, and it reached minimum validation RMSE for 35 iterations. We obtained a RMSE value of 0.876 and accuracy of 0.322. Results are shown in the following Figure **??**.

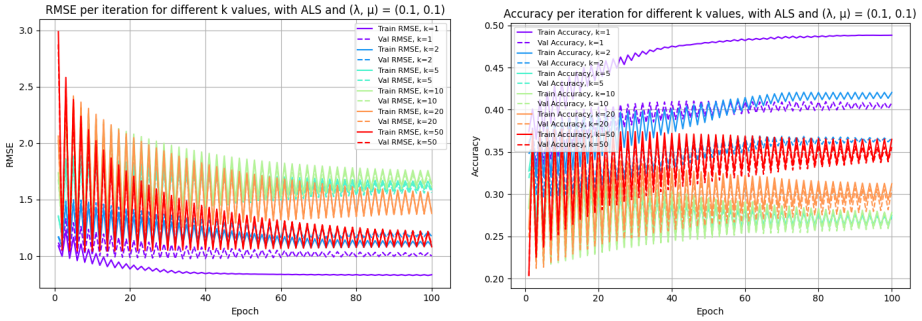


(a) RMSE for k varying. (b) RMSE for lr_I and lr_U varying. (c) RMSE for λ varying.

(d) Accuracy for k varying. (e) Accuracy for lr_I and lr_U varying. (f) Accuracy for λ et varying.

Figure 2: Cross Validated Average Metrics over splits for Matrix Factorization with Gradient Descent. Parameters for each figure are those for the best combination for MF with GD, varying one parameter at a time. In 2a and 2d, k varies in $\{1, 2, 5, 10, 20, 50\}$, in 2b and 2e, lr_I and lr_I vary in $\{0.005, 0.01\}$, in 2b and 2e, λ varies in $\{0.01, 0.1, 1.0\}$.

The best combination we found for Matrix Factorization with ALS was $k = 1$, $\lambda = \mu = 0.1$, and it reached minimum validation RMSE for 20 iterations. We obtained a RMSE value of 0.886, and an accuracy of 0.315. Results are shown in the following Figure 3.



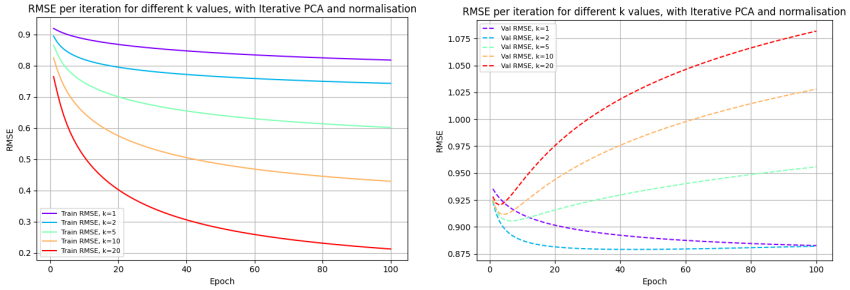
(a) Training and Validation RMSE. (b) Training and Validation accuracy.

Figure 3: Cross Validated Average Metrics over splits for Matrix Factorization with Alternative Least Squares.

- **Iterative PCA:**
  - Number of principal components (k): We tested values in $\{1, 2, 5, 10, 20\}$.

  The best combination we found for the Iterative PCA algorithm was $k = 1$, and it reached minimum validation RMSE for 50 iterations.

  Notice, on Figure 4, that the training error is not reported since, by construction, it is zero. Indeed, the algorithm only updates missing values and leaves the observed entries unchanged.



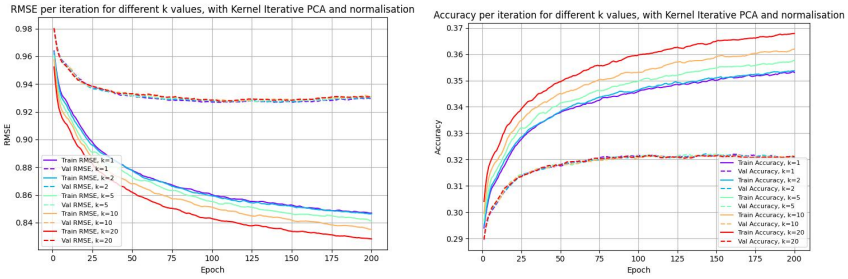
(a) RMSE Train (b) RMSE Test

Figure 4: Cross Validated Train and Test Average RMSE over splits for Iterative PCA.

The above Figure 4 shows that the Iterative PCA converges very quickly to a suboptimal solution on the test set within fewer than ten iterations when k is large, which leads to overfitting. In contrast, when k is small, for example equal to one or two, the convergence is slower but the resulting solution is more optimal, achieving the best validation metrics. Given the high sparsity of the matrix, this behavior suggests that only a small number of latent factors is required to capture its underlying structure effectively. On the training set, we observe that larger values of k yield the lowest error, indicating a better fit to the training data but clear overfitting on the validation set, which confirms our previous observations.

- **Iterative Kernel PCA:**
  - Number of principal components (k): We tested values in $\{1, 2, 5, 10, 20\}$.
  - RBF Kernel gamma (γ): fixed at 0.1.
  - Ridge Regularization alpha (α): fixed at 0.1.

  The best combination we found for the Iterative Kernel PCA algorithm was $k = 1$, and it reached minimum validation RMSE for 106 iterations.



(a) RMSE (b) Accuracy

Figure 5: Cross Validated Average Metrics over splits for Kernel Iterative PCA.

Figure 5 shows the convergence behavior of the Kernel Iterative PCA model for different values of k. The RMSE curves (a) decrease steadily across epochs, indicating stable optimization. The accuracy curves (b) exhibit a complementary trend. Reguarding the training curves, we observe the same behavior as for the iterative PCA.

This systematic process ensures that our final comparison is based on the best possible configuration for each implemented method.

5.4 Performance Comparison

The table below shows the performance of each implemented method after hyperparameter tuning. The score is given by taking the mean of the scores obtained during the cross-validation.

Table 1: Performance comparison of all implemented models after cross-validation.

Model	**Best RMSE**	**Exact Accuracy (%)**	**Exec. Time (s)**
Matrix Factorization (GD)	0.876	0.322	0.53
Matrix Factorization (ALS)	0.886	0.315	1.43
Iterative PCA	0.879	0.318	29.16
Iterative Kernel PCA (RBF)	0.927	0.322	3.86

5.5 Analysis

The results indicate that Matrix Factorization with GD performs slightly better than PCA-based approaches in terms of RMSE, accuracy, and efficiency. PCA can be viewed as a constrained form of matrix factorization, where the orthogonality imposed on the latent space reduces model flexibility and expressiveness. In addition, the iterative truncation of singular components introduces extra bias, which further limits its performance compared to unconstrained matrix factorization methods.

6 Conclusion and Lessons Learned

This project provided an in-depth comparison of several collaborative filtering approaches for movie recommendation under extreme data sparsity.

Our experiments show that matrix factorization methods, particularly Gradient Descent, remain the most reliable and computationally efficient solutions. Their direct optimization on observed entries makes them naturally suited for sparse data.

Overall, the study highlights the importance of model simplicity, data normalization, and the choice of inductive bias when designing recommender systems for incomplete datasets.

References

[1] R. Vidal, Yi Ma, and S. S. Sastry. 2016. "Generalized Principal Component Analysis". Springer Publishing Company, Incorporated, Section *Robustness Issues for PCA*.

[2] P. Honeine and C. Richard, "Preimage Problem in Kernel-Based Machine Learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 77-88, March 2011.

[3] H. I. Alshbanat, H. Benhidour, and S. Kerrache, "A Survey of Latent Factor Models in Recommender Systems," *Information Fusion*, vol. 117, p. 102905, May 2025. doi:10.1016/j.inffus.2024.102905.

[4] D. Kim and B.-J. Yum, "Collaborative Filtering Based on Iterative Principal Component Analysis," *Department of Industrial Engineering, Korea Advanced Institute of Science and Technology (KAIST)*, Taejon, Korea.