Datalab Project 1: Collaborative Filtering

Groupe **Regelegorilab**

Nathan Rouillé
Rayane Dakhlaoui
Sacha Khosrowshahi

IASD
October 21, 2025

1 Introduction

This project explores collaborative filtering methods to predict user ratings on a set of items. We implemented and compared several algorithms, tuning their key hyperparameters to understand how design choices affect performance.

Our primary evaluation metric is RMSE, used to quantify prediction error. Because the course leaderboard also ranked models by accuracy, we optimized for accuracy-oriented variants when appropriate.

2 ALS and SGD

We focus on two classical matrix factorization techniques:

- **Alternating Least Squares (ALS)**
- **Stochastic Gradient Descent (SGD)**

Both approaches decompose the user–item rating matrix into latent factors representing users' and items' underlying preferences. We aim to understand the differences in formulation, optimization behavior, and sensitivity to hyperparameters.

The primary metric used to evaluate performance is the **Root Mean Squared Error (RMSE)**

2.1 Method 1: Alternating Least Squares (ALS)

2.1.1 Motivation

We first implemented **ALS** because it provides a stable and interpretable framework for matrix factorization. The main advantage with this method is that it's a deterministic approach and we're working with closed-form problems which allows fast convergence.

2.1.2 Results and Discussion

The ALS model typically converged within a few iterations due to its closed-form updates. It provided a good baseline with relatively low RMSE but limited flexibility in handling sparse gradients.

We observed that increasing the latent dimension k improved performance until a plateau, after which overfitting began to occur. The best RMSE was obtained for $k = 64$ and $\lambda = 0.01$.

2.2 Method 2: Stochastic Gradient Descent (SGD)

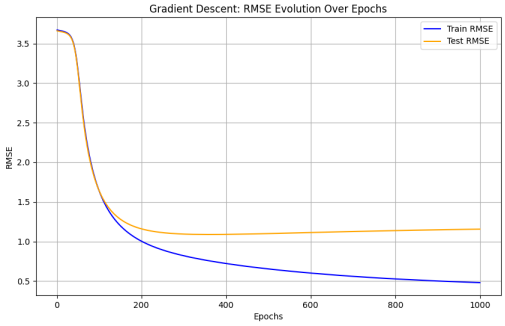
2.2.1 Motivation

After obtaining the ALS baseline, we moved to **SGD** to gain finer control over the optimization dynamics. Unlike ALS, SGD updates parameters incrementally, which can lead to faster convergence on large sparse datasets.

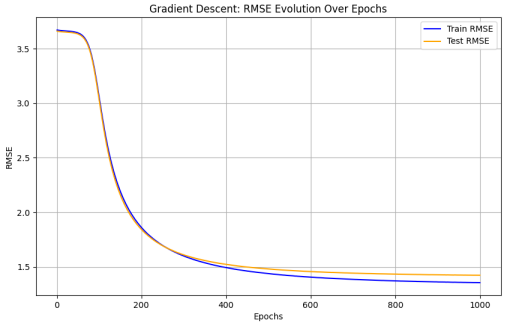
2.2.2 Hyperparameters

We did a grid search on multiple hyper parameters in order to assess the advantages of having strong or low regularization but also, the influence of the number of latent vectors k.

We first start to assess the RMSE on low regularization coefficients and we see that the test RMSE starts to rise after descending. This shows obvious overfitting on our training data. We therefore go from $\lambda_U = 0.01$ and λ_I=0.01 to λ_U=0.1 and λ_I=0.1 in order to get better results.



(a) $\lambda_U = 0.01$ and λ_I=0.01



(b) λ_U=0.1 and λ_I=0.1

Figure 1: RMSE evolution with different regularizations

Finally we refined the search by doing a grid search and conclude that the best results were obtained with λ_U=0.3 and λ_I=0.3

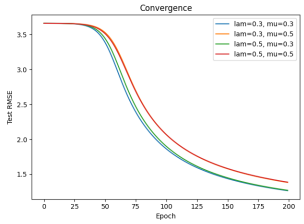


Figure 2: Enter Caption

The final hyperparameter we wanted to analyze was the number k of latent vectors. These latent vectors are important in matrix factorization because they give information on the semantics of the task. Here the latent vectors could be a representation of the correlations of genres between movies.

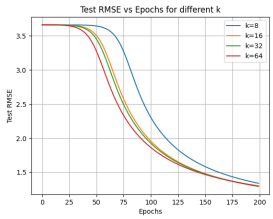


Figure 3: Influence of number of latent vectors on RMSE

2.3 Discussion and Conclusion

Both ALS and SGD proved effective for matrix factorization, each with distinct advantages. ALS offered a stable and interpretable baseline thanks to its deterministic closed-form updates and rapid convergence, but it wasn't a flexible method. Transitioning to SGD allowed finer control over the optimization through the learning rate and regularization, enabling incremental updates. When carefully tuned, SGD achieved smoother convergence and lower RMSE, demonstrating that while ALS provides a solid foundation, gradient-based methods can yield superior performance for collaborative filtering tasks.

3 Kernelized Matrix Factorization

3.1 Setup and Goals

Building on matrix factorization (MF) with user/item biases, we implemented a kernelized variant to capture non-linear user–item interactions. The key idea is to retain the MF structure while replacing $p_u^\top q_i$ with a kernel function $\kappa(p_u, q_i)$, making the model possibly more expressive.

After training three KernelMF models on the same train/test split, each with a different kernel on the user–item pairs. We then (i) add an item-based genre kernel to produce a neighbourhood predictor, and (ii) learn a *linear combination* of all predictors by tuning nonnega-

tive weights $\mu \in \mathbb{R}^p$ on the probability simplex, $\mu \geq 0,\ \mathbf{1}^\top \mu = 1$. Predictions combine as $[\widehat{y} = \sum_{j=1}^{p} \mu_j \widehat{y}^{(j)} = P\mu]$, where $P \in \mathbb{R}^{n \times p}$ stacks the p prediction vectors on the evaluation split.

3.2 Model

Predictions are given by $[\hat{r}_{ui} = \mu + b_u + b_i + \kappa(p_u, q_i)]$, where μ is the global mean, b_u and b_i are user/item biases, and $p_u, q_i \in \mathbb{R}^k$ are latent factors. With offsets (a, c) stabilize the kernel scale; γ controls sensitivity.

3.3 Training (kernel-specific SGD)

We train with **SGD on observed triplets** (u, i, r_{ui}) and L_2 regularization.

Updates biases with learning rate η are:

$$e_{ui} = r_{ui} - \hat{r}_{ui},$$
$$b_u \leftarrow b_u + \eta\,(e_{ui} - \lambda_b b_u),$$

We compiled these per-kernel updates with `Numba` to speed up epochs and enable practical hyperparameter sweeps.

3.4 Base KernelMF models

We trained three KernelMF baselines on the same data split:

- **Linear (MF + biases)**: best single model overall (RMSE ≈ 0.912); biases explain a substantial share of signal early in training.
- **RBF**: competitive after sweeping γ (RMSE ≈ 0.989 near $\gamma = 1$); otherwise worse.
- **Sigmoid**: underperforms on this split (RMSE ≈ 1.083). Possibly too sensitive to (γ, a, c) due to saturation.

These numbers set a clear baseline: the linear kernel is strong, while the other two give small but sometimes useful signals.

3.5 Genre kernel and neighbourhood predictor

From `namesngenre.npy` we build a multi-hot matrix $G \in \mathbb{R}^{\#\text{items} \times \#\text{genres}}$. We define an **item–item genre similarity** with cosine:

$$K_{\text{genre}}(i, h) = \frac{\langle G_i, G_h \rangle}{\|G_i\|\,\|G_h\|}.$$

Using K_{genre}, we predict a user u's rating on item i with an item-based neighbourhood rule:

$$\widehat{r}_{ui} = \mu_u + \frac{\sum_{j \in \mathcal{N}_k(u)} K_{\text{genre}}(i, j)\,(r_{uj} - \mu_u)}{\sum_{j \in \mathcal{N}_k(u)} K_{\text{genre}}(i, j) + \lambda},$$

where μ_u is the user mean, $\mathcal{N}_k(u)$ are the k items rated by u (top-k most similar to i), and λ is a small shrinkage to stabilize low-weight cases.

3.6 Linear ensembling on the simplex

We learn the mixture μ with a **Dirichlet random search** on the simplex, picking the weights that minimize RMSE on the evaluation split. We then apply these weights to the corresponding test predictions.

3-model ensemble (KernelMF only). Mixing {linear, sigmoid, RBF} prefers the linear model strongly and brings a small improvement compared to uniform mixing.

We get $\mu^\star \approx [0.9394,\ 0.0005,\ 0.0305]$ for {linear, sigmoid, rbf}, and $\text{RMSE}_{\text{test}}(\mu^\star) = \mathbf{0.906}$

4-model ensemble (KernelMF + genre). Adding the genre predictor as a fourth column improves a bit the final score. On our test split, the tuned weights and the resulting error are:

We get $\mu^\star \approx [0.5713,\ 0.0005,\ 0.0352,\ 0.3931]$ for {linear, sigmoid, rbf, genre},

and $\mathrm{RMSE}_{\mathrm{test}}(\mu^\star) = \mathbf{0.8963}$, and $\mathrm{RMSE}_{\mathrm{test}}(\text{uniforme}) = 0.9395$.

3.7 Takeaways

After comparing ALS and SGD approaches and KernelMF methods, we extended the SGD-based MF by introducing user and item bias terms. This simple modification allows the model to account for systematic rating differences between users and items (for example, generous users or globally popular movies).

By learning these biases jointly with the latent factors, we obtained a clear improvement in RMSE and smoother convergence. It also keeps the optimization problem relatively simple while significantly improving prediction accuracy.

4 Principal Component Analysis

4.1 PCA for Collaborative Filtering

After experimenting with several matrix factorization methods and regularization schemes, we considered Principal Component Analysis as a natural next step. PCA can be interpreted as a constrained matrix factorization in which the latent space directions are required to be orthogonal, this constraint acts as an implicit regularizer by eliminating redundant directions and limiting model capacity.

4.1.1 Problem setting and preprocessing

Let $X \in \mathbb{R}^{n \times p}$ denote the user–item rating matrix, where missing entries are represented by NaN. We first center the matrix by subtracting the per-item mean μ computed only over observed ratings. Centering is essential because PCA is sensitive to the mean (the sample mean can dominate variance directions). We do not apply per-item scaling. Empirically, scaling did not really affect predictive performance in our experiments, and it complicates NaN handling (because the per-item variance estimate becomes ill-defined for items with few observations). Thus, for robustness and simplicity of missing-value handling we use centering only.

4.1.2 Partial covariance estimation and eigen-decomposition

Because many entries are missing, the empirical covariance between C two items i and h is computed only over users who rated both items. The symmetric matrix $C \in \mathbb{R}^{p \times p}$ is then diagonalized via eigen-decomposition:

$$C = V \Lambda V^\top,$$

where $V = [v_1, \ldots, v_p]$ contains orthonormal eigenvectors and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ the eigenvalues ordered decreasingly. Retaining the first k eigenvectors yields $V_k \in \mathbb{R}^{p \times k}$ and $\Lambda_k \in \mathbb{R}^{k \times k}$.

4.1.3 Projection, reconstruction and discrete rounding

User scores (projections) onto the k-dimensional latent space are computed by projecting each (centered) user vector onto the retained item directions. Stacking these gives the score matrix $S \in \mathbb{R}^{n \times k}$.

The reconstructed rating matrix is

$$\widehat{X} = S V_k^\top + \mathbf{1}\mu^\top.$$

Missing entries in X are then imputed by corresponding entries of $\widehat{X}$. To better match the original rating scale (half-integers between 0.5 and 5.0), we finally round predictions to the nearest half-integer and clip to the valid interval $[0, 5]$.

4.1.4 Hyperparameter selection

We selected the latent dimensionality k by grid search over $k \in \{2, \ldots, 100\}$, optimizing test

RMSE. The optimal value found was $k^\star = 8$. Figure 4 shows the RMSE on the test set as a function of k (grid search).

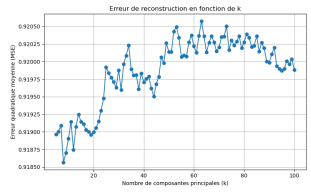


Figure 4: Top three genres most correlated with each principal component (correlation values shown).

4.2 Results and Interpretation

4.2.1 Quantitative results

With $k = 8$ components, the InDaPCA-based imputation and recommendation pipeline attains:

$$\text{RMSE}_{\text{test}} = 0.92, \qquad \text{Accuracy}_{\text{test}} = 0.24.$$

These metrics compare favorably to alternative methods we evaluated: the accuracy of 0.24 is the best observed in our experiments (we did not impose the rounding to the nearest half integer in the previous methods as it decreased performance on RMSE), and the RMSE of 0.92 is close to the best one of ≈ 0.9.

4.2.2 Qualitative interpretation of latent dimensions

To interpret the retained components (rows of V_k), we correlated them with binary genre indicators. For each component we list the three genres with highest Pearson correlation.

Two important observations follow from these numbers:

1. **Correlations are small in magnitude.** Most reported correlations are below 0.05. This indicates that no single component is dominated by a single genre, rather components reflect mixtures of genre attributes.
2. **Components capture blended user preferences.** For instance, Comp_1 shows weak association with *Comedy*, *Mystery* and *Romance* simultaneously, which we interpret as an axis capturing preference for narrative-driven, lighter-toned films rather than "comedy-only" taste. Comp_8 association with *Crime*, *Horror* and *Thriller* suggests an axis tuned toward suspenseful/intense viewing preferences.

4.3 Attempts of Improvement

4.3.1 Iterative PCA via EM

We implemented an iterative variant of PCA inspired by the EM algorithm of *Roweis (1998)* for handling missing data. Each iteration alternates between: **E-step**: imputing missing entries using the current PCA reconstruction and **M-step**: recomputing PCA on the completed matrix.

Because predictions must lie in $\{0, 0.5, 1, \ldots, 5\}$, we first used a hard rounding $\text{round}(2x)/2$ at every iteration. This caused stagnation after the first pass. We then tried a smooth continuation scheme:

$$X^{(t)} = (1 - \alpha_t)\, X^{(t)}_{\text{recon}} + \alpha_t \,\text{round}\Big(2X^{(t)}_{\text{recon}}\Big)/2,$$

with α_t increasing linearly or sigmoidal towards 1. None of these schedules yielded improvements: performance remained essentially identical to normal PCA (**RMSE** ≈ 0.93, **Acc** ≈ 0.24).

5 Conclusion

We submitted the MF SGD model without biais thanks to kernel methods and obtained 0.877 RMSE and 25.27 accuracy on the website.