Recommender Systems: Implementation of Collaborative Filtering Algorithms and Benchmark

Hugo Veríssimo
Foundations of Machine Learning 24/25
University of Aveiro
Aveiro, Portugal
hugoverissimo@ua.pt

João Cardoso
Foundations of Machine Learning 24/25
University of Aveiro
Aveiro, Portugal
joaopcardoso@ua.pt

***Abstract*—Many areas of life can be improved by suggestions from our surroundings. Recommender systems have been used reliably since their early inception the 90's, and the field exploded upon the introduction of the Netflix prize, which aimed at developing the best and most efficient recommender system for their movie platform. In this work three recommender systems based on collaborative filtering are developed and presented, based on one of the MovieLens datasets. The models are compared considering their error metrics, and the capacity to rank items accurately. The models are compared with the literature, focusing on the framework used for comparison.**

Keywords: MovieLens, GroupLens, Recommender System, Collaborative Filtering, Linear Regression, FunkSVD

I. Introduction

Since the early 90's that the production rate of multimedia content has increased dramatically (pun intended). Initially, users relied mostly on video store owners, film critics on newspapers, friends. With increasing volumes of content, recommender systems have appeared as data-driven methods to reliably and quickly recommend movies based on the users' own appreciations.

Recommender systems are information tools that provide users with recommended items (ideally) based on their list of preferences [1], [2]. These can be divided in different types, depending on the algorithm and approach to the data. Three categories can be considered: non-personalized, content-based, and collaborative filtering algorithms. Regardless of the algorithm of choice, these face crucial challenges: cold start problem (where it is difficult to tailor recommendations to a user without known preferences, or recommend an item with no reviews); data sparsity (given that most users review few items in the universe of possible items, leaving most of the user/item matrix empty); and scalability (as data grows exponentially, processing becomes evermore expensive and troublesome). These systems have been widely used in many different areas (online shopping, music, books, movie recommendation), and significant investment has gone into developing evermore personalized algorithms. A notable case for this was the Netflix prize competition in 2006, a moment where the research in the field skyrocketed. As a result, it was needed to develop better frameworks for comparison, considering not only the metrics, but data preprocessing, preparation, and routine, in order to ensure reproducibility across models and authors [3].

For that reason, the developed recommender systems is based on one of the most widely used movie databases, MovieLens dataset, for education and development [4]. The choice of this dataset allowed for a based comparison with algorithms from the literature, and facilitate the analysis and interpretation of the results here presented.

II. State of the Art

The field of recommender systems is wide and covers many different algorithms and techniques. In this paper we focus on collaborative filtering using matrix factorization, as it is the main focus of the work here developed.

The Netflix prize is often cited as one of the main drivers for research in collaborative filtering recommender systems [5]. One of the initial awarded proposals was that by Brandyn Webb, known by his alias "Simon Funk". Despite the name of the algorithm (FunkSVD), Singular Value Decomposition is not used, it uses instead gradient descent to find the latent feature values used to predict the ratings matrix. The algorithm uses only the available ratings, representing a great advantage against SVD methods that struggle with mostly sparse matrices.

Unlike SVD, the original matrix is decomposed between two matrices (for users and movies), where the diagonal matrix typically found in SVD is merged into one of the two. Since the original matrix is so sparse, the u matrix and the v matrices are initiated randomly, and estimated by minimizing the error relative to the original matrix via gradient descent.

$$r_{ij} = u_i \cdot v_j \tag{1}$$

$$u_{if}(\text{new}) = u_{if}(\text{old}) + 2\alpha(r_{ij} - \tilde{r}_{ij})v_{jf} \tag{2}$$

$$v_{jf}(\text{new}) = v_{jf}(\text{old}) + 2\alpha(r_{ij} - \tilde{r}_{ij})u_{if} \tag{3}$$

There is a caveat however, given that the model requires fine tuning of numerous parameters (number of latent features, learning rate, training iterations, regularization parameter), which can lead to overfitting [6]. In the work by Zhou et al., alternating least squares with weighted λ regularization (ALS-WR).

$$f(U,M) = \sum_{\{i,j\}|r_{i,j}\in I} \left(r_{i,j} - u_i^T m_j\right)^2 + \lambda \left(\sum_i n_{u_i} \|u_i\|^2 + \sum_j n_{m_j} \|m_j\|^2 \right) \quad (4)$$

The algorithm expresses the rating matrix as the product of two smaller matrices U (user matrix) and M (item matrix). Thanks to its simplicity, the algorithm tackles both scalability and sparseness of user profiles, with the added bonus of not overfitting [7]. A third model by Gopalan et al. uses a probabilistic approach by assuming that the observed rating is drawn from a Poisson distribution, which is parameterized by the inner product of a user weights vector and an item weights vector.

$$y_{ui} \sim Poisson(\theta_u^T \beta_i) \quad (5)$$

Where:

User weights: $\theta_u = [\theta_{u1}, \ldots, \theta_{uk}]$
Item weights: $\beta_i = [\beta_{i1}, \ldots, \beta_{ik}]$
$\theta_{uk} \sim Gamma(a, b)$
$\beta_{ik} \sim Gamma(c, d)$

With this, the model is able to compute the probability for each unconsumed item that the user might consume [8].

III. Methodology

A. Data description

The dataset MovieLens for Education and Research (small) was used to test the different models. It contains 100.836 ratings, from 0.5 to 5, for 9724 movies (and its genres) and 610 users, where each user rated at least 20 movies [9].

B. Data splitting & models implemented

The dataset was split between train and test (80/20), which resulted in 80668 ratings for the training and 20168 for the testing dataset. Three models were developed, as described in the Table I.

TABLE I: Models implemented to the MovieLens dataset.

Models - features	Description
Model 01 - Users and Movies	Collaborative filtering, Linear Regression (CF-LR)
Model 02 - Users and Movies + Genres	Collaborative filtering, Linear Regression content based (CB-LR)
Model 03 - Users and Movies	Collaborative filtering, FunkSVD (CF-FunkSVD)

The models were fitted with an 8-fold cross validation to find the best hyperparameters for the considered interval. For Model 03, the package *surprise* and the tools therein [10] were used.

The primary error metrics employed in this study were the mean absolute error (MAE) and the root mean squared error (RMSE). The MAE considers all error equally, regardless of their size, whereas RMSE strongly penalizes larger errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (6)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (7)$$

C. Exploratory data analysis

The dataset comprises a diverse selection of movies spanning a wide range of genres, as illustrated in Fig. 1, with a total of 20 genres.

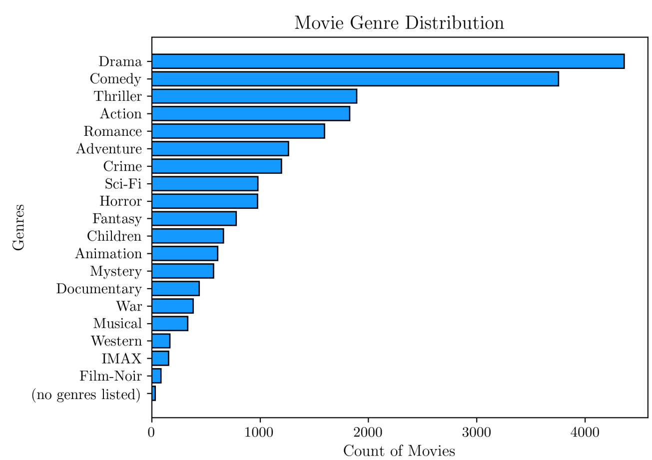

Fig. 1: Distribution of movies per movie genre.

Despite the variety in genres, it's important to retain that generally movies are more complex than representing a single genre. In that sense, it is useful to assess the similarity between genres (using the cosine similarity index), as illustrated in Fig. 2. This helps understand how the task of recommending a movie can start to grow more complex as we add more and more detail to the dataset. In Table II the five most related genres are shown.

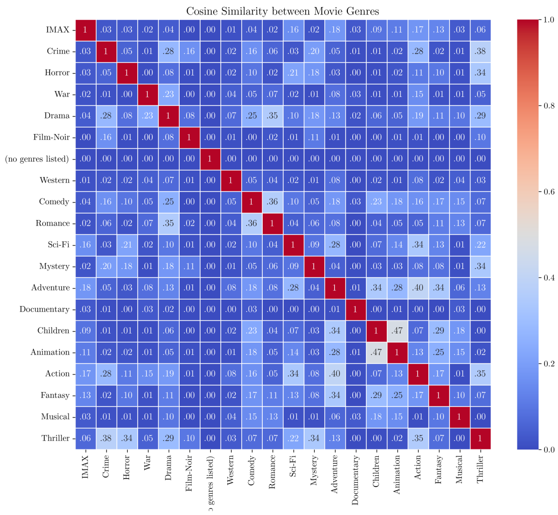

Fig. 2: Cosine similarity index between genres.

TABLE II: The five most related genres per cosine similarity index.

Genres		**Similarity Index**
Animation	Children	.47
Action	Adventure	.40
Crime	Thriller	.38
Romance	Comedy	.36
Romance	Drama	.35

The dataset is relatively recent, with movies from the 2010's, going all the way back to the early 1900's, as per Fig. 3. As expected, there are much less ratings for older movies, for two main reasons: older movies are less popular and there are less movies in general.

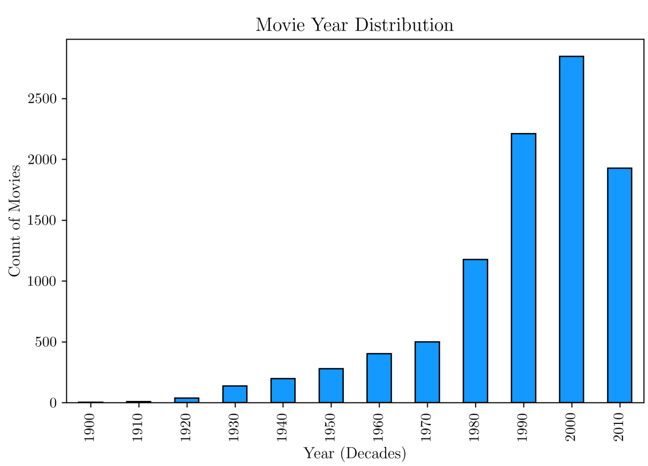

Fig. 3: Distribution of movies per year.

As expected, both distributions from Fig. 4 and 5 are positively skewed as mentioned before. In general there are more people rating a small amount of movies, and the number of movies with higher counts of ratings tends to decrease rapidly (as there are very few, very popular movies).

This type of distribution is commonly found across recommender systems, and represents well the challenges posed to these algorithms: sparsity, whereas most of the matrix is empty (98.3 % of missing values); bias, popular movies tend to dominate the analysis; and the long tail represents the challenge to make recommendations based on the large number of movies (or users) with very few ratings (or that rated few movies).

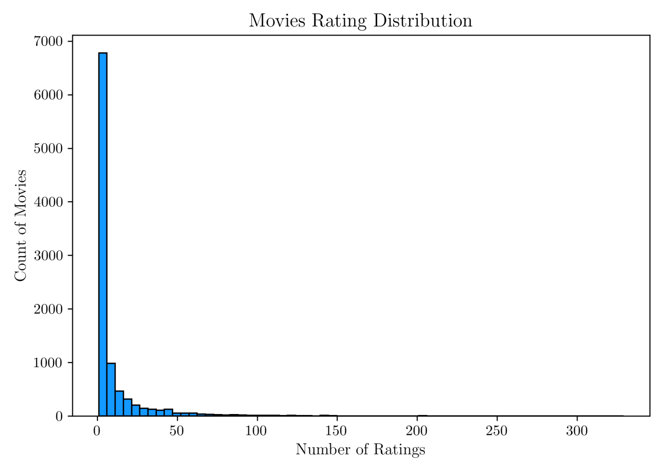

Fig. 4: Distribution of the number of movies and the amount of ratings. The first bin correspond to 1 movie rated.

To finalize, it is important to mention (again) that the minimum number of ratings per user is 20, and the maximum is 2698. The minimum rating per movie is 1, and the maximum is 329 ratings. The most rated movies (and not so coincidentally) the highest rated movies were:

- Forrest Gump (1994) - 329 ratings
- Pulp Fiction (1994) - 317 ratings
- The Shawshank Redemption (1994) - 307 ratings

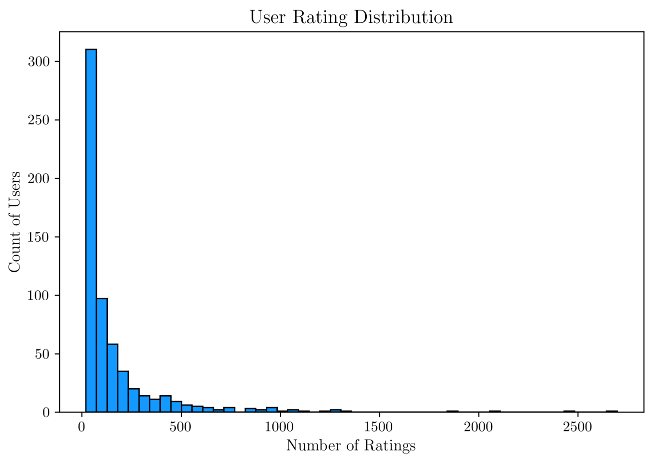

Fig. 5: Distribution of the number of users and the amount of ratings. The first bin correspond to 20 movies rated.

IV. Classification Models

The three models here presented are first and foremost linear regression problems, where by fitting a model with the known data to the users and the movies, it will be possible to estimate the ratings of movies that have not been reviewed. In this sense, besides the model itself, it is necessary to fit two hyperparameters, the learning rate α, and the cost parameter λ. For both CF-LR and CB-LR 20 features were used for users, and for movies (40 features in total). The number of features was defined empirically, based on the number of genres in the dataset (20 genres).

The first model developed aims to fit both movies (x parameter) and users (θ parameter) simultaneously, as depicted in the equation below, by optimization of the cost function with regularization.

$$\min_{\substack{x^{(1)},\dots,x^{(n_m)} \\ \theta^{(1)},\dots,\theta^{(n_u)}}} \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)} \right)^2 \quad (8)$$

The second model fits only the users parameter, given that it accounts for the movies parameters by using the movie genres.

$$\min_{\theta^{(1)},\dots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)} \right)^2 \quad (9)$$

The third and final model is the famous FunkSVD. As mentioned in the state of the art, despite the many regularization parameters, we've limited the analysis to the number of latent features, overall learning rate (which could be fit for the many features possible), and the regularization parameters.

A. CF-LR Model

Different ranges of α (between 0.0001 and 0.002) and λ (between 0 and 100) were given to the model to be fit by usin 8 cross-validation in the training dataset. This allowed to progressively lower RMSE, throughout 500 iterations.

The best hyperparameters for this model were $\lambda = 6$ and $\alpha = 0.0005$, with an average RMSE from 8-CV of 1.25622.

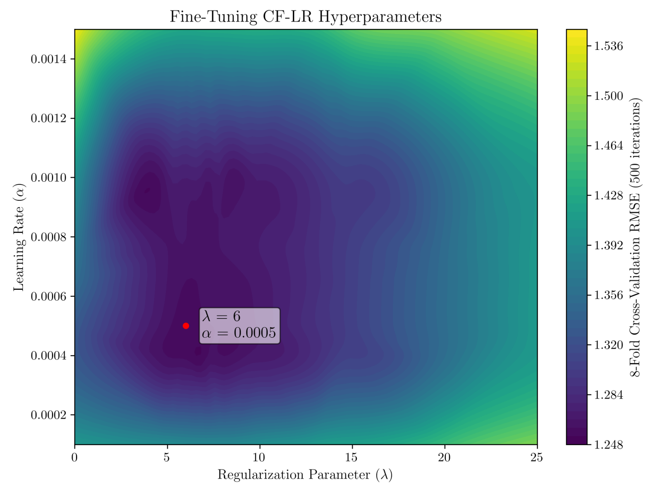

Fig. 6: Learning rate (α) and regularization parameter (λ) fitting.

Once the optimal hyperparameters were defined, the learning curve was estimated from the 8 CV on the training dataset, to assess the validity of the model and that it is not overfit. To further confirm that little to no overfitting is observed, in Fig. 7 the progression of the training score and the CV score tends to improve across the train set size, not overlapping or changing the trend significantly. It is important to bear in mind that there is a risk of slight overfitting, considering that there are 40 features to be fit.

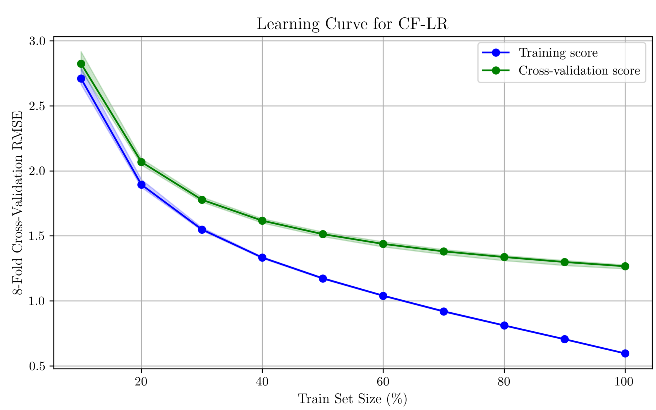

Fig. 7: Learning curve, RMSE progression throughout the training dataset.

Using the defined hyperparamters, the cost function was estimated, where we see it sharply decreasing as the iterations run, finally converging at the limit.

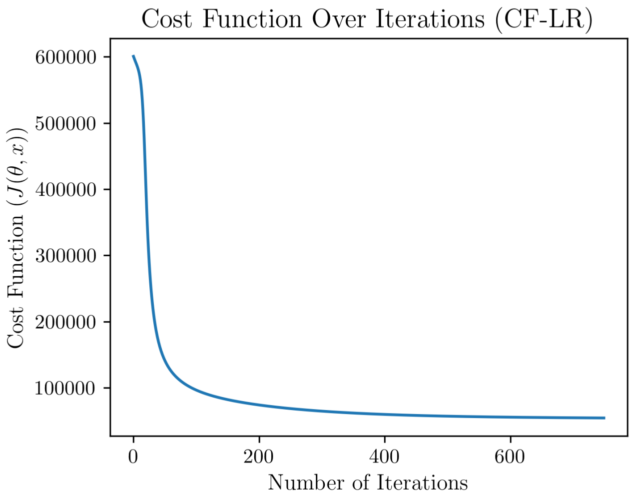

Fig. 8: Cost function evolution with the number of iterations in the training set.

With the model defined, the train and test sets were compared for the errors (RMSE and MAE), as despicted in Table III, showing (as expected) the increase of errors in the test set in regards to the training set. The fact that errors increased significantly from the training to the testing set further supports the likelihood of overfitting the data.

TABLE III: Error metrics for the train and test set, along with the number of samples for each.

Dataset	**RMSE**	**MAE**	**Support**
Train Set	0.58842	0.44604	80668
Test Set	1.24037	0.90651	20168

B. CB-LR Model

The CB-LR model is less demanding as it is only estimating the features related to the users, considering that the movie features are estimated from the genres given per movie in the dataset. Each movie is weighted in terms of how much it is represented by a given genre (e.g., Pearl Harbor being for example 33/33/33: war, drama and romance), which is represented by vector x. The movie features don't need to be optimized, and as the name states, the model is a linear regression *content based*, leaving the vector θ, user features, to be optimized.

As seen in Fig. 9, α was optimized between 0.00005 and 0.007, whereas λ was varied between 0 and 10. The optimized hyperparameters (as a result of 8-fold CV) are pinpointed in the graph, corresponding to (0.05, 0.005) with 8-fold CV RMSE of 0.98617.

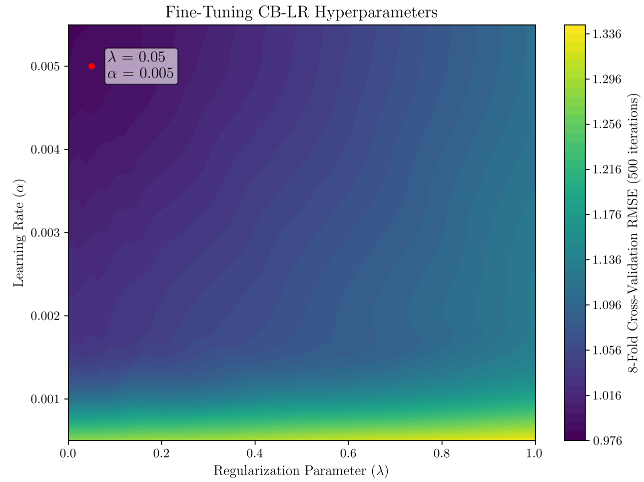

Fig. 9: Learning rate (α) and regularization parameter (λ) fitting.

As in CF-LR the learning curve was estimated to ensure no overfitting occurred. In this case the trends vary in an identical fashion for the whole training set, which is a good indicator of an adequate fit.

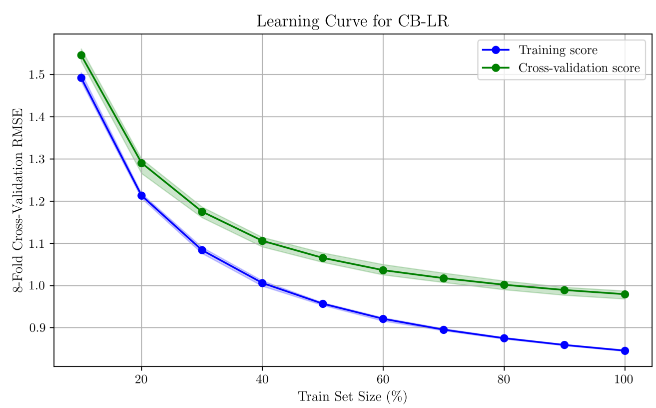

Fig. 10: Learning curve, RMSE progression throughout the training dataset.

In Fig. 11 the cost function is minimized as in CF-LR, indicating proper convergence of the model.

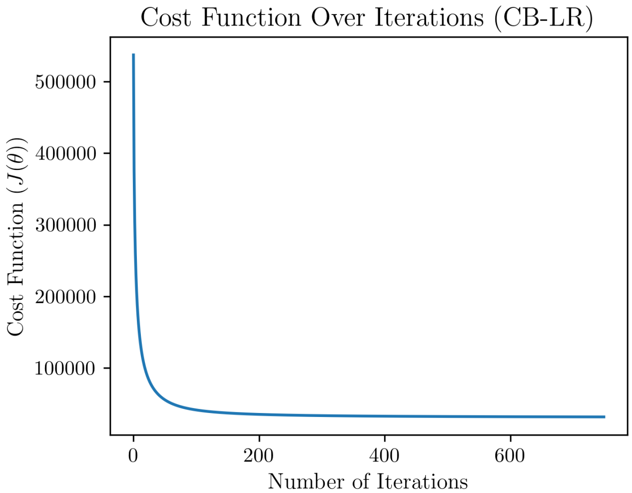

Fig. 11: Cost function evolution with the number of iterations in the training set.

The error metrics in Table VI further contribute to the model's proper fit, that do not vary significantly from the train to the test set.

TABLE IV: Error metrics for the train and test set, along with the number of samples for each.

Dataset	RMSE	MAE	Support
Train Set	0.84859	0.65997	80668
Test Set	0.97254	0.74605	20168

C. CF-FunkSVD Model

The CF-FunkSVD model was designed with a minimal set of optimized hyperparameters to reduce the risk of overfitting and to evaluate the impact of these parameters on the model's performance. The optimization process was conducted using a *GridSearchCV* with an 8-fold cross-validation. In Table V the range of values for each hyperparameter is given, where *n_factors* corresponds to the number of factors, *lr_all* is the overall learning rate, and *reg_all* the regularization parameter. The remaining parameters are kept as default, indicated in *surprise* Python package [10]. The best fit values were 50, 0.01, and 0.1, respectively.

TABLE V: FunkSVD model hyperparameters search space.

Hyperparameter	Possible Values
$n\_factors$	$\{5, 15, 30, 40, 50\}$
lr_all	$\{0.005, 0.01, 0.05, 0.1, 1\}$
reg_all	$\{0.02, 0.1, 1, 5, 10\}$

The learning curve (Fig. 12) shows a different behavior considering what was observed in the other two models, where both training and testing are converging, with the former evidencing increasing RMSE, while the latter is decreasing.

With this sort of approach, as the training set increases the model becomes more generalistic, while compromising it's accuracy, resulting in increased error. The fact that the testing set shows decreasing RMSE is a sign that the model is able to be more generic.

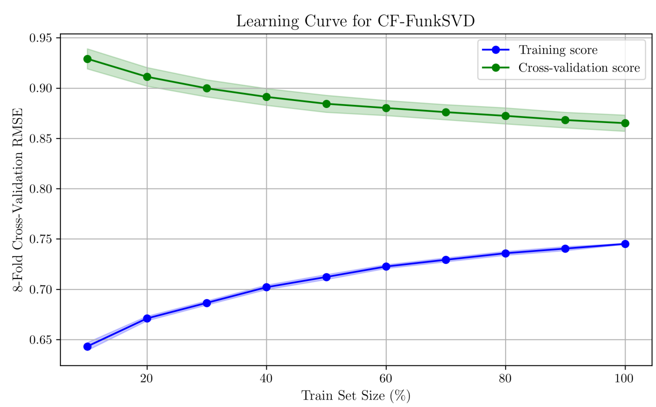

Fig. 12: Learning curve, RMSE progression throughout the training dataset.

As before, the error metrics evidence an adequate fit between the training and testing set, considering that there is not a significant increase between the training and the testing sets.

TABLE VI: Error metrics for the train and test set, along with the number of samples for each.

Dataset	RMSE	MAE	Support
Train Set	0.75035	0.58313	80668
Test Set	0.87412	0.66957	20168

V. Discussion

A. Performance Metrics

In order to assess the three models, the several error metrics are presented in the form of boxplot (Fig. 13, obtained by applying 8 CV to the whole dataset (7 folds for fitting and 1 fold for test), while using the optimized models. The recall, F1 score, and precision are known throughout the literature and commonly used for many machine learning algorithms. Their definitions can be found in the authors' previous work [11].

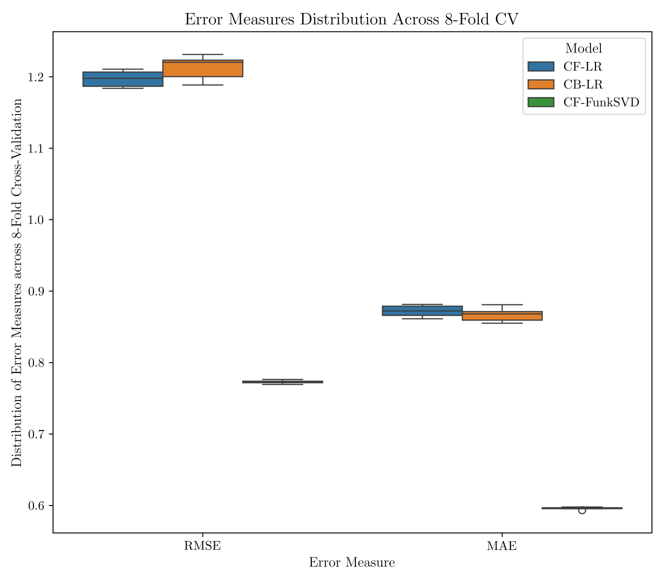

Fig. 13: Boxplot of RMSE and MAE error metrics determined by 8 CV across whole dataset

Despite the narrow range of errors, showing a high precision and repeatability for each model, CF-FunkSVD clearly outperforms both models for both error metrics.

In Fig. 14 to 17, the evolution of each of the metrics is calculated and displayed for each model, where k represents the "Top k" movies. One thing to consider right from the beginning, the larger the list of ranked items the harder it is for any of the algorithms to accurately identify it. In most of the metrics, CF-FunkSVD performs better than the remaining, but only slightly better for the top 100 items.

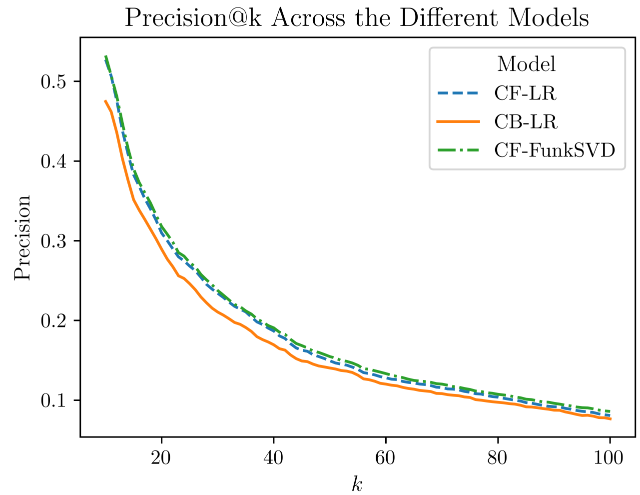

Fig. 14: Precision@k up to k = 100 for the three models.

Recall is considerably better for CF-FunkSVD across the k range, while F1@k is consistently better than the other two models. The CB-LR model performs well, considering its simplicity, suffering mostly due to the large error evidenced. The MRR metric (mean reciprocal rank) estimates how well a system ranks items for a population of users.

$$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{rank}_i} \tag{10}$$

In this case, it is notable how CB-LR performs considerably worse than the other models (for any k), and how for the top 100 CF-LR surfaces as the best performing model, although the performance over the whole range is comparable to that of CF-FunkSVD.

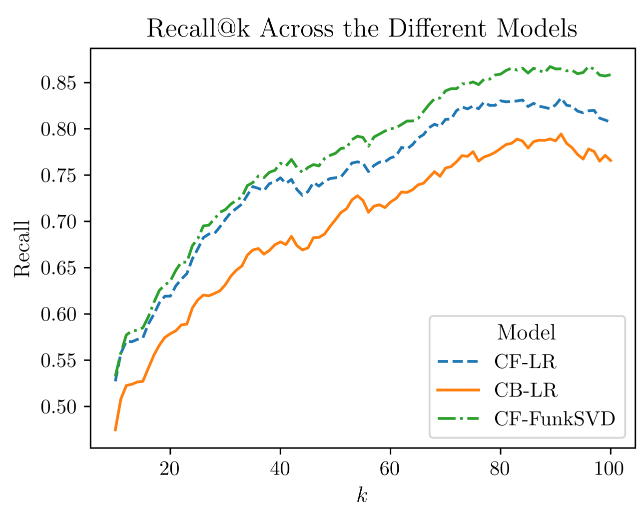

Fig. 15: Recall@k up to k = 100 for the three models.

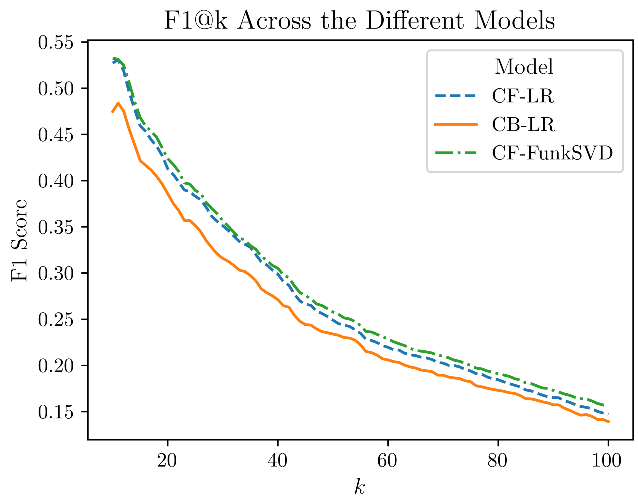

Fig. 16: F1@k up to k = 100 for the three models.

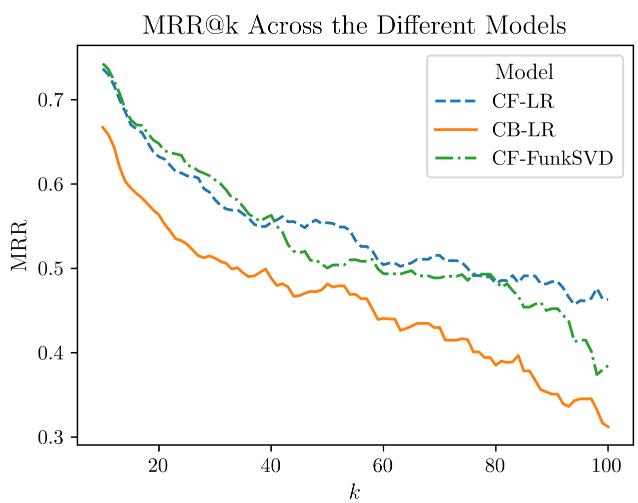

Fig. 17: MRR@k up to k = 100 for the three models.

B. Literature Benchmark

The models in question are simpler in application compared to CF-FunkSVD, however, it is worthwhile to reflect on their performance comparing with the literature. First and foremost, the differences at this level are not so significant, as the CF-FunkSVD depends largely on fine tuning of the multitude of hyperparameters available. However, it still outperforms both types of linear regression.

TABLE VII: CAPTIO mesaures@100 across the models for the test data only

Measure	**CF-LR**	**CB-LR**	**CF-FunkSVD**
Precision@100	0.08073	0.07659	0.08585
Recall@100	0.80732	0.76585	0.85854
F1@100	0.14679	0.13925	0.15610
MRR@100	0.46304	0.31212	0.38477
RMSE	1.23691	0.97240	0.87268
MAE	0.90417	0.74607	0.66804

THE TABLE ALLOWS FOR AN ALMOST DIRECT COMPARATIONS TO THE WORK OF.... so apra fazer referencia a tabela tlvz

When considering the work of Paullier et al. [12], where seven different models were tested under the same conditions, to highlight the comparison framework and approach to this type of problem, the models developed and presented in this paper fall well in line with the performance achieved in the literature. The values are not worth detailing here, as the datasets are not the same, Paullier used the smaller MovieLens 100k dataset, and comparing the FunkSVD models, the one presented in this work shows better metrics, largely due to the dimension of the dataset to better fit the algorithm.

VI. Conclusion

The purpose of this project was to learn, assess and employ different recommendation systems. The literature review allowed to understand how different models emerged throughout the years, and one of the biggest motivations for the expansion of the field, the Netflix prize. The dataset used is a benchmark across the field, decently sized, which allowed for proper fitting of the models and comparison with the literature.

The developed models offer opportunities for further enhancement. Both collaborative filtering (CF-LR) and content-based (CB-LR) models could benefit from a more systematic approach to determining the optimal number of features, rather than relying on empirical estimates. Additionally, the third model presents numerous hyperparameters that warrant fine-tuning, which could serve as a dedicated research endeavor. There are several promising directions for future exploration, including incorporating the FunkSVD model into an ensemble algorithm.

Work Load

Both authors contributed equally to the project.

References

[1] J. A. Konstan and J. Riedl, "Recommender systems: From algorithms to user experience," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 101–123, 2012. [Online]. Available: https://doi.org/10.1007/s11257-011-9112-x

[2] R. Katarya and O. P. Verma, "An effective collaborative movie recommender system with cuckoo search," *Egyptian Informatics Journal*, vol. 18, no. 2, pp. 105–112, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110866516300470

[3] A. Said and A. Bellogín, "Comparative recommender system evaluation: benchmarking recommendation frameworks," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ser. RecSys '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 129–136. [Online]. Available: https://doi.org/10.1145/2645710.2645746

[4] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 19:1–19:19, 2015. [Online]. Available: https://doi.org/10.1145/2827872
[5] Netflix Prize Community, “Netflix prize forum: Topic 1537,” 2009, archived on the Wayback Machine. [Online]. Available: https://web.archive.org/web/20090924184639/http://www.netflixprize.com/community/viewtopic.php?id=1537
[6] S. Funk, “Netflix update: Try this at home,” 2006, accessed: 2025-01-09. [Online]. Available: https://sifter.org/~simon/journal/20061211.html
[7] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, “Large-scale parallel collaborative filtering for the netflix prize,” in *Algorithmic Aspects in Information and Management*, R. Fleischer and J. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–348.
[8] P. Gopalan, J. M. Hofman, and D. M. Blei, “Scalable recommendation with poisson factorization,” 2014. [Online]. Available: https://arxiv.org/abs/1311.1704
[9] G. Research, “The movielens datasets,” 2025, accessed: 2025-01-09. [Online]. Available: https://grouplens.org/datasets/movielens/
[10] N. Hug, “Surprise: A python library for recommender systems,” 2017, accessed: 2025-01-14. [Online]. Available: https://surpriselib.com/
[11] H. Veríssimo and J. Cardoso, “Applying machine learning models to assess fatigue and prevent injury in high performance swimming athletes,” *Foundations of Machine Learning*, 2024.
[12] A. Paullier and R. Sotelo, “A recommender systems’ algorithm evaluation using the lenskit library and movielens databases,” in *2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2020, pp. 1–7.