Recommender System

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*—**

Keywords: MovieLens, GroupLens, Recommender System, Collaborative Filtering

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 nam 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 certain SVD methods.

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

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

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).

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

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.

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

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 how likely the user is to consume new items [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 1 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 an 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 (LR)
Model 02 - Users and Movies + Genres	Collaborative filtering, Linear Regression content based (LRC)
Model 03 - Users and Movies + Genres	Collaborative filtering, Single Value Decomposition (SVD)

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

The error metrics used in the present work 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.

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

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

C. Exploratory data analysis

The dataset consists of several movies across a wide range of genres. As seen in Fig. 1 there is positively skewed distribution, as is common in this type of dataset.

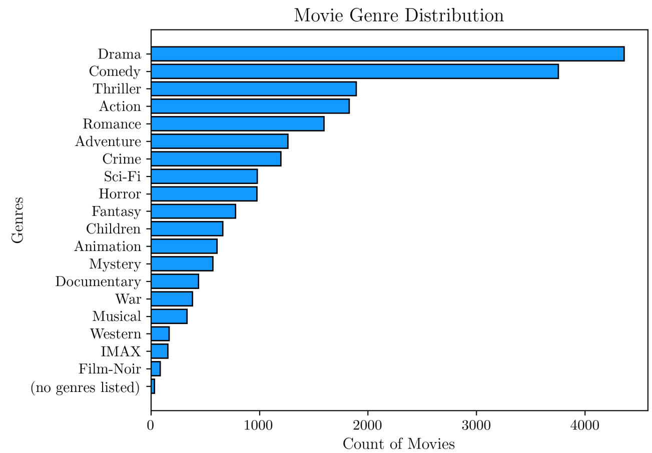

Fig. 1: Distribution of movies per movie genre.

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.

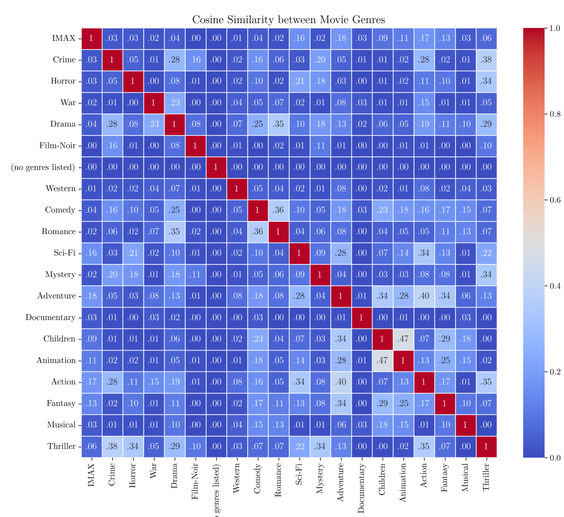

Fig. 2: Cosine similarity index between genres.

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

Genre	**Genre**	**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.

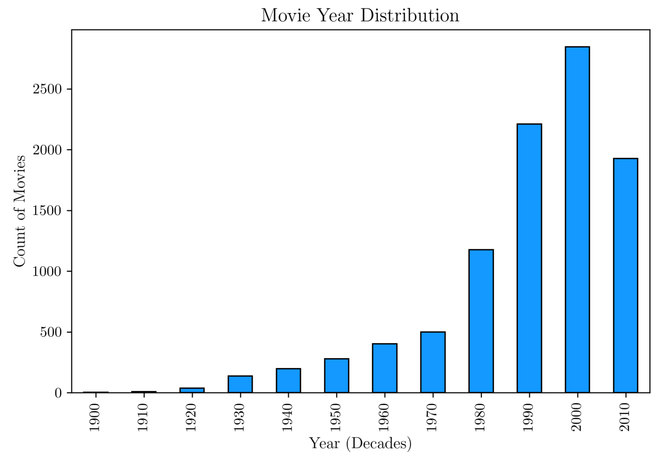

Fig. 3: Distribution of movies per year.

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).

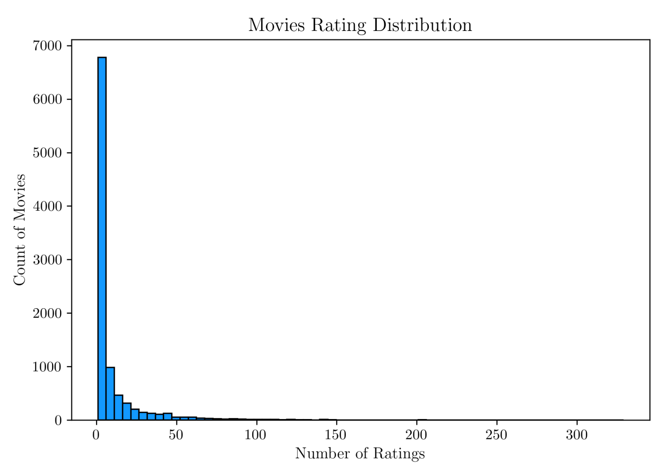

Fig. 4: Distribution of the number of movies and the amount of ratings. The first bin correspond to 1 movie rated.

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

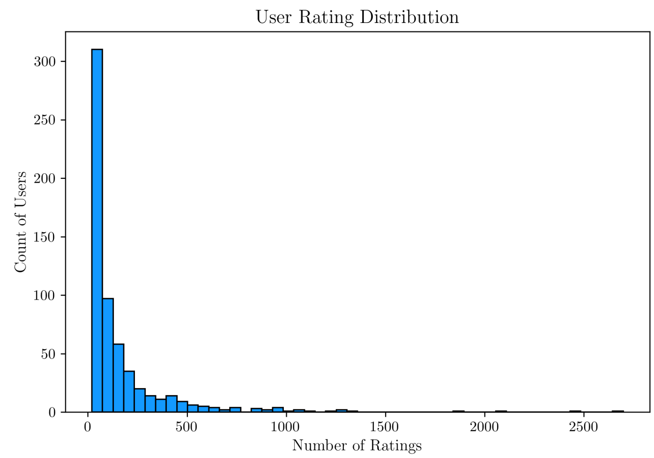

Fig. 5: Distribution of the number of users and the amount of ratings. The first bin correspond to 20 movies rated.

D. *MEDIDAS DE ERRO OU AVALIACAO*

IV. Classification Models

V. Model01

o modelo ? colaborative filtering? consiste na otimizacao simultanea do x (paremetros do movies) e do theta (param dos users?) confirmar com slides.

meter formula e parametros e isso tp slides ?

é um problema de regressao linear, pelo que exige a otimizacao dos hyperparametros lambda (parametro de custo) e alpha (learning rate)

foi dado ao modelo varios valores de alpha entre 0.0001 e 0.002 e para lambda entre 0 e 100, e atraves do 8 fold cross validation no conjunto de treino, minimizando a medida de erro RMSE, com 500 iteracoes, foram os hyperparametros otimizados, como se pode ver na figura seguinte

os melhores hyperparametros obtidos foram entao $\lambda = 6$ e $\alpha = 0.0005$, com o rmse medio do 8cv de 1.25622

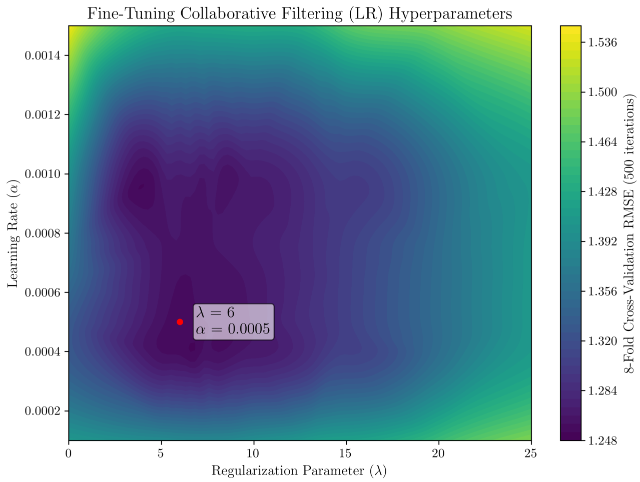

Fig. 6: CAPTIO CAPTION CAPTION

obtidos os hyperparametros otimos, foi entao aplicado 8cv learning curve nos dados de treino para verificar a validacao do modelo nos dados de treino em termos de overffting

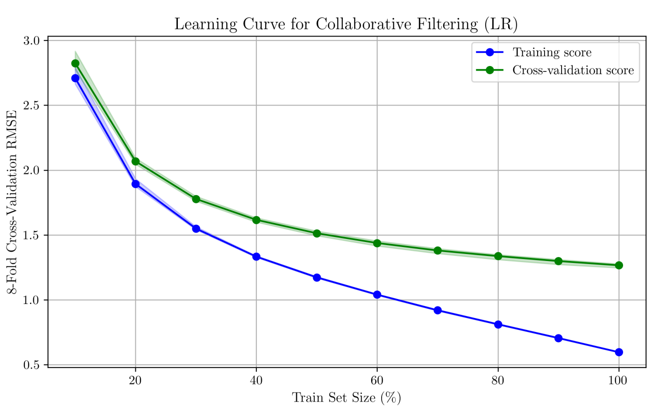

Fig. 7: CAPTIO CAPTION CAPTION

dps foi entao aplicada a descida do gradiente? regressao linear? para obter a estimativa dos parametros x e theta (movies e users) do problema.

a figura seguinte mostra a funcao custo ao longo das iteracoes, revelanco a convergencia da mesma

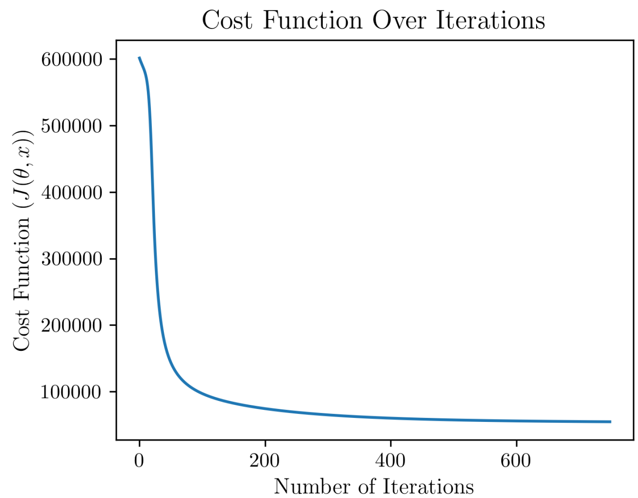

Fig. 8: CAPTIO CAPTION CAPTION

apos a estimativa dos ratings para todas as combinacoes de movies e users, compararam-se estas estimativas com as presentes na train e test data

TABLE III: CAPTIO CAPTION CAPTION

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

VI. MODEL02

aqui é igual ao primeiro modelo mas ao inves de termos o x? (movies features) vamos ter os generos dos filmes, por exemplo se um filme é romance/drama é [0,0,....5,.5] tipo

VII. MODEL03

TABLE IV: SVD 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	$\{0.02, 0.1, 1, 5, 10\}$

Number of Singular Values or Components: n_factors

Learning rate: lr_all

Regularization Parameters: reg_all

Best Hyperparameters: {'n_factors': 50, 'lr_all': 0.01, 'reg_all': 0.1}

lallla learning curve

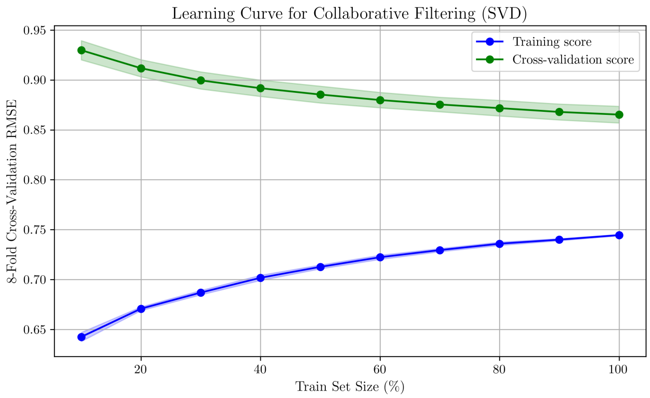

Fig. 9: CAPTIO CAPTION CAPTION

lalala results meauseres

TABLE V: CAPTIO CAPTION CAPTION

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

VIII. Discussion

A. *Performance Metrics*

B. *Decision Boundaries*

C. *Literature Benchmark*

IX. Conclusion

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/