Recommender System

April 5, 2023

Malthe Faurschou Tøttrup
201907882@post.au.dk
Department of Electrical and Computer Engineering Aarhus University

Lukas Koch Vindbjeg
201906015@post.au.dk
Department of Electrical and Computer Engineering Aarhus University

Abstract

Recommendation systems have become increasingly significant in contemporary applications, encompassing e-commerce, social networking platforms, and streaming services. This paper details our efforts in constructing a recommendation system for the ML-100K dataset, which comprises 100,000 movie ratings from 943 users spanning across 1,682 films. Our methodology entails comparing three prominent recommendation models (SVD, KNNbasic, and NMF) and identifying SVD as the superior model based on the root mean square error (RMSE) evaluation metric. Subsequently, a grid search is employed to optimize the hyperparameters of the SVD model.

The results of our experiments indicate that our recommendation system effectively predicts movie ratings and provides pertinent recommendations to users. The optimal performance is achieved by the SVD model with hyperparameters **n_factors**=40, **n_epochs**=25, **lr**=0.008, and **reg_all**=0.08, yielding an average RMSE of 0.92. We also discovered that recommending five movies strikes an optimal balance between precision and recall. This research lays the foundation for further exploration in the realm of recommendation systems. We aim to assess our system using alternative datasets and investigate distinct recommendation methodologies that can enhance our model's interpretability, thereby offering improved explanations for the recommendations presented to end users.

Table of Content

Introduction

Recommendation systems have become increasingly important in today's world as a means of assisting users in finding relevant items. These systems have been applied to a wide range of domains, including movies, books, products, and even friends on social media. One of the most widely studied recommendation system datasets is the ml-100k dataset, which contains a set of ratings provided by users of the MovieLens website.

The ml-100k dataset provides a unique opportunity to develop and evaluate recommendation systems. The dataset is relatively small, containing only 100,000 data samples, but it has been widely used as a benchmark for evaluating recommendation algorithms. In this paper, we explore the task of building a recommendation system for the ml-100k dataset.

Our goal is to develop a model that can accurately predict the ratings that users will give to movies they have not yet seen. We consider several types of recommendation systems, including content-based, collaborative filtering, and hybrid systems. We evaluate the performance of our model using standard evaluation metrics, such as RMSE and precision-recall.

The remainder of this paper is organized as follows. Section 2 provides a review of the related literature on recommendation systems, focusing on papers that are relevant to our work. Section 3 describes the methodology we used to build the recommender system. Section 4 presents our results and evaluates the performance of our model. Finally, Section 5 concludes the paper and discusses potential directions for future work.

Related Work

Recommendation systems have been studied extensively in the literature, and many different approaches have been proposed for addressing the problem of item recommendation. In this section, we review some of the most relevant papers in the field of recommendation systems, focusing on those that are related to our work on the ml-100k dataset.

Collaborative filtering is one of the most widely used approaches for recommendation systems. Collaborative filtering is based on the idea that people who have similar preferences in the past are likely to have similar preferences in the future. In the context of recommendation systems, this means that users who have rated similar movies in the past are likely to have similar ratings for movies they have not yet seen. One of the earliest and most influential papers on collaborative filtering was the work of Resnick and Varian [1], who introduced the idea of using a user-item matrix to predict missing ratings.

Content-based recommendation systems, on the other hand, rely on the similarity between items based on their attributes (e.g., genre, director, actors) rather than on user ratings. Content-based methods have the advantage of being able to make recommendations for new items, but they may not perform as well as collaborative filtering methods when there is not enough information about the user's preferences. In the context of movie recommendation, a popular approach is to use movie metadata such as genre, director, and actors to create movie profiles. Once these profiles are created, recommendations can be made based on the similarity between the profile of the movie and the user's preferences.

Hybrid recommendation systems combine multiple approaches to improve recommendation accuracy. For example, a hybrid system may combine collaborative filtering and content-based methods to take advantage of the strengths of each approach. Several papers have explored the use of hybrid systems for recommendation, including the work of Burke [2], who introduced the concept of “knowledge-based” recommendation systems that combine content-based and collaborative filtering approaches.

In the context of the ml-100k dataset, several papers have explored different approaches to recommendation. For example, Sarwar et al. [3] used a neighborhood-based collaborative filtering approach to predict movie ratings, achieving an RMSE of 0.95. Breese et al. [4] used a variant of the k-nearest-neighbor algorithm to predict movie ratings, achieving an RMSE of 0.98. Herlocker et al. [5] explored the use of a content-based approach, using movie metadata to make recommendations, achieving an RMSE of 1.08. The current state-of-the-art method is the work by Darban and Valipour [6], which uses a hybrid deep-learning autoencoder and k-means-clustering approach, and achieves an RMSE of 0.887.

In this paper, we build a recommendation system based on the work of these previous studies by exploring the use of different recommendation systems and evaluating their performance on the ml-100k dataset. We focus on developing a model that can accurately predict movie ratings.

Methodology and Materials

In this section, we describe the process we used to select and train the best recommendation model for the ml-100k dataset. To build our models we use the Surprise python library [7] which is a collection of models and functions for building and analyzing recommender systems. We decided to opt for building a recommender system based on the collaborative filtering approach, as this is regarded as a simple yet effective method. Collaborative filtering is based on the idea that if two users have similar preferences for a set of items, then they are likely to have similar preferences for other items as well. Therefore, the system can recommend items to a user based on the items that similar users have liked or purchased in the past. An illustration of how collaborative filtering works is shown in Figure 1.

There are two main types of collaborative filtering techniques: memory-based and model-based.

Memory-based collaborative filtering algorithms, such as K-Nearest Neighbors (KNN) and cosine similarity, work by calculating the similarity between users based on their ratings or preferences for items. For example, KNN identifies the k nearest neighbors of a target user based on their similarity in terms of item ratings. Once the most similar users have been identified, the system can recommend items to the target user that were liked or purchased by those similar users.

Model-based collaborative filtering algorithms, such as Singular Value Decomposition (SVD) and Non-Negative Matrix Factorization (NMF), work by creating a mathematical model of the user-item preferences based on the ratings or preferences of all users. This model can then be used to predict how likely a user is to like or purchase a particular item.

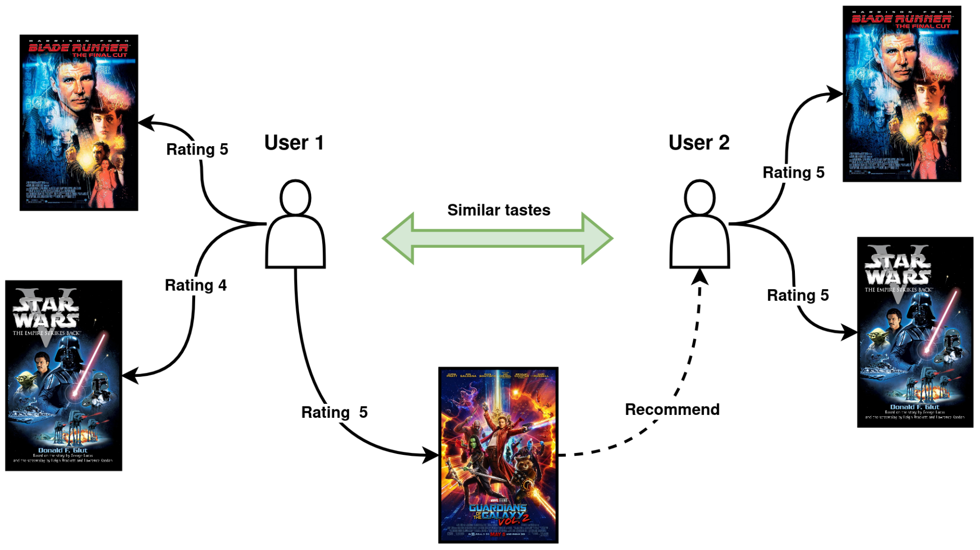

Figure 1: Illustration of the collaborative-filtering approach. In this case users one and two have similar tastes, i.e. they distribute high ratings to the same movies. User one has rated an item high, which user two has not yet rated. Therefore, this item is recommended to user two.

Finding the Best Model

In the first step, we compare three popular collaborative-filtering models: SVD, KNN with a basic cosine similarity measure (KNNBasic), and NMF. Each model has the default hyperparameters from the Suprise library. We use K-fold cross-validation with 5 folds to train and measure the performance of each model. The performance is measured using the RMSE and the results are shown in Table 1. Our results show that SVD outperforms the other models, as it has the lowest RMSE score.

Model	**RMSE**
SVD	0.94
KNNBasic	1.02
NMF	0.96

Table 1: Model comparison based on average RMSE score after 5 folds cross-validation process.

Based on these results, we select SVD as our primary recommendation model. SVD is a matrix factorization technique that can be used to reduce the dimensionality of the user-item interaction matrix, which represents the user-item ratings or preferences. The idea is to represent the original matrix as the product of three matrices: a user matrix, a diagonal matrix of singular values, and an item matrix. The singular values capture the most important features or factors that explain the variability in the original matrix, and the user and item matrices represent the relationships between users and items in this reduced feature space. The pseudo-code for the algorithm is shown in Table 2. However, the implementation in the Pythons Surprise library may differ slightly.

```
SVD(user-item-ratings, factors, learning rate, epochs, lambda):
  1 Initialize user factors matrix ($P$) and item factors matrix ($Q$) with random values
  2 Set the number of factors, learning rate, regularization term ($\lambda$), and number of epochs
  3 For each epoch:
    3.1 Shuffle the list of user-item-rating triplets
    3.2 For each user-item-rating triplet $(u, i, r_{\text{ui}})$ in the list:
      3.2.1 Calculate the predicted rating: $\hat{r}_{\text{ui}} = P[u] \cdot Q[i]$
      3.2.2 Calculate the prediction error: $e_{\text{ui}} = r_{\text{ui}} - \hat{r}_{\text{ui}}$
      3.2.3 Update user factors: $P[u] = P[u] + \text{learning\_rate} \cdot (e_{\text{ui}} Q[i] - \lambda P[u])$
      3.2.4 Update item factors: $Q[i] = Q[i] + \text{learning\_rate} \cdot (e_{\text{ui}} P[u] - \lambda Q[i])$
  4 Return the learned factors $P$ and $Q$
```

Table 2: Pseduo code for the SVD algorithm. The algorithm uses stochastic gradient descent (SGD) to learn the factors iteratively by minimizing the prediction error between the true ratings and the predicted ratings, which are calculated as the dot product of the user and item factors. The final learned matrices can then be used to make predictions for unseen user-item pairs.

Hyperparameter space Search

After deciding to use SVD to build the model we use a grid search to find the optimal combination of hyperparameters. The hyperparameters for SVD are the following:

- **n_factors**: This hyperparameter controls the number of latent factors to use when factorizing the user-item matrix. Latent factors are underlying characteristics or features of the data that the model tries to learn. Setting this hyperparameter too low may result in underfitting, while setting it too high may result in overfitting.
- **n_epochs**: This hyperparameter controls the number of iterations or epochs the model goes through during training. Setting this hyperparameter too low may result in underfitting, while setting it too high may result in overfitting or slower training times.
- **lr_all**: This hyperparameter controls the learning rate for all parameters in the model during training. Setting this hyperparameter too high may result in unstable training, while setting it too low may result in slow convergence or suboptimal performance.
- **reg_all**: This hyperparameter controls the regularization term for all parameters in the model during training. Regularization is a technique used to prevent overfitting by adding a penalty term to the objective function. Setting this hyperparameter too low may result in overfitting, while setting it too high may result in underfitting.

For the grid search, we manually define 3-4 options for each of the hyperparameters. Then we use 5-fold cross-validation to train and evaluate each combination of the hyperparameters. Our grid search results suggest that the best combination of hyperparameters for the SVD model are **n_factors**=40, **n_epochs**=25, **lr**=0.008, and **reg_all**=0.08. With these hyperparameters, we get an average RMSE score of 0.92 for the 5-fold cross-validation.

Experiments, Results, and Discussion

In this section, we present the experiments we conducted to evaluate the performance of our recommendation system on the ml-100k dataset. First, we compare our results to other works and state-of-the-art methods. Then we analyze the precision-recall curve for our recommendation model to determine the number of relevant items to recommend. Finally, we investigate some of the movie recommendations generated by our model.

Method Comparison

Since the ml-100k dataset is a relatively small dataset is common to average the RMSE score for 5-fold cross-validation. During our hyperparameter search, we presented our best achieved average RMSE result of 0.92. Table 3 shows how our method compares to some of the other methods on the dataset. From the results we can see that our method performs fairly well compared to the other methods, however, it is still outperformed by the state-of-the-art.

Method	**RMSE**
Ours (SVD)	0.92
Sarwar et al. [3]	0.95
Breese et al. [4]	0.98
Herlocker et al. [5]	1.08
Darban and Valipour [6]	**0.887**

Table 3: Model comparison based on average RMSE score after 5 folds cross-validation process on the ml-100k dataset.

Precision and Recall

To determine the number of items to recommend, we use the precision-recall curve. Precision and recall are metrics commonly used to evaluate the effectiveness of recommendation systems. Precision measures the proportion of recommended items that are relevant to the user, while recall measures the proportion of relevant items that are recommended. A higher precision and recall indicate a more effective recommendation system. We plot the precision-recall curve in Figure 2 for our SVD model using different numbers of recommended items, to find a balance between the two. From the plot, we can see that the two lines intersect at the point of recommending 5, and thus achieves a good balance between precision and recall.

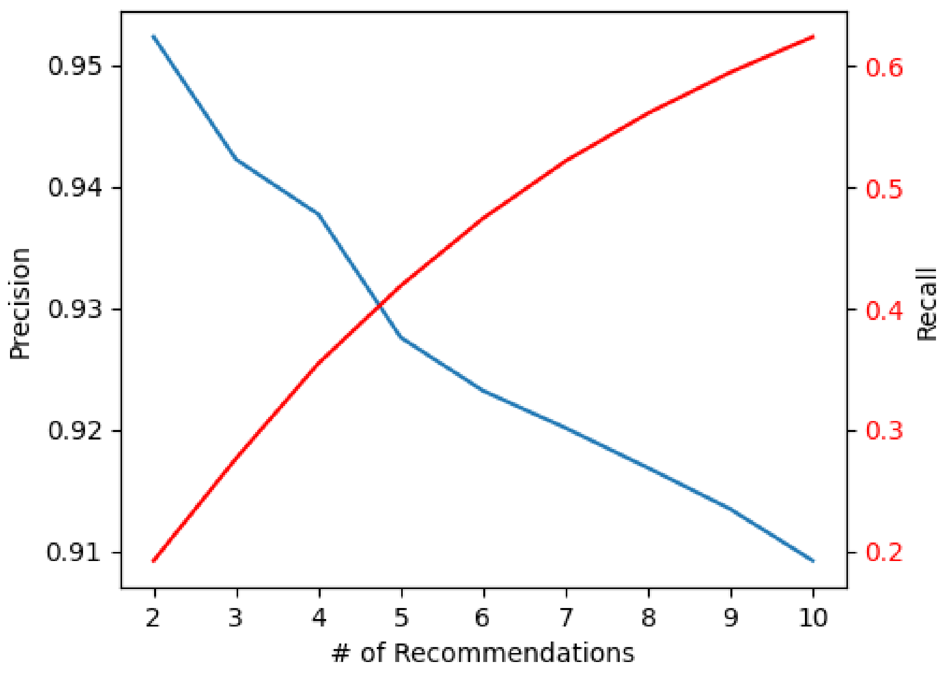

Figure 2: Precision-recall curve. The curve shows that recall increases as we recommend more items while the precision drops. The best balance between the two is at the point where the two lines intersect at 5 items.

Model Recommendations

Following an evaluation of the RMSE and precision-recall metrics of our model, we sought to provide concrete recommendations using the aforementioned model. To achieve this, we constructed an anti-test set consisting of all user-item pairs lacking a rating in the original dataset, enabling us to utilize the entire dataset for training. Once we had retrained the model with the same hyperparameters, we could then employ it to suggest movies to users that they have yet to view. It is important to note, that due to the absence of target rating values used in the computation of the RMSE and precision-recall metrics, the precision of our model cannot be accurately evaluated. This situation mirrors the deployment of real-world recommender systems, where the precision of the model is evaluated primarily based off of customer feedback.

In the present study, we have recommended five movies to user 10, based on the model's prediction of their highest ratings. Table 4 displays the five movies with the highest predicted ratings for user 10, even though the decimal point ratings may not necessarily align with the integer-based 1-5 rating scale. Nonetheless, these ratings prove useful in determining the most relevant movies for recommendation purposes.

Movie ID	Title	Predicted rating
1449	Pather Panchali (1955)	4.95
318	Schindler's List (1993)	4.89
408	A Close Shave (1995)	4.77
169	The Wrong Trousers (1993)	4.75
114	Wallace & Gromit: The Best of Aardman Animation (1996)	4.74

Table 4: Our model's highest predictions for the user with ID 10. Based on these predictions we recommend these 5 movies to the user.

Discussion

Our study is not without limitations, which warrant discussion. Firstly, the selection of SVD as our model was based on a comparison with KNNBasic and NMF, using the default hyperparameters. The optimal hyperparameters were subsequently determined through a time-intensive hyperparameter grid search, conducted after the model comparison. However, it is plausible that one of the alternative methods would have yielded superior results on the dataset with a different hyperparameter configuration. Consequently, it would have been more appropriate to conduct a grid search for each of the methods prior to the model comparison.

Secondly, our study exclusively compared collaborative filtering methods, which rely solely on user-item ratings for model construction. A more comprehensive examination incorporating content-based filtering, hybrid, and deep learning methods would have been ideal. The efficacy of these techniques varies widely depending on the dataset and available data features.

Lastly, as SVD was employed to develop our model, the provision of a compelling rationale for the recommendation to the end user is challenging. The model relies on the training data to identify users with similar preferences and employs this information to generate the prediction. Nevertheless, the methodology utilized by SVD makes it arduous to extract the metadata necessary for delivering a lucid explanation for the recommendation. Consequently, the sole basis for our recommendation to the user is the predicted output value of our model.

Conclusion and Future Work

In this paper, we presented our work on building a recommendation system for the ml-100k dataset. We started by reviewing the related work in the field of recommendation systems and highlighting the strengths and weaknesses of various approaches. We then described our approach to model selection and training, which involved comparing three popular models (SVD, KNNbasic, and NMF) and selecting SVD as the best-performing model based on RMSE. We also used a grid search to optimize the hyperparameters of the SVD model.

Our results suggest that our recommendation system is effective at predicting movie ratings and providing relevant recommendations to users. The SVD model with hyperparameters **n_-factors**=40, **n_epochs**=25, **lr**=0.008, and **reg_all**=0.08 performed best among the models we considered, indicating that the model is able to capture the underlying patterns in the data and make accurate predictions. The precision-recall curve also shows that our system is able to provide recommendations that are relevant to the user, with a reasonable trade-off between precision and recall when we recommend 5 items.

In terms of future work, we plan to evaluate our system on other datasets as well as compare it with other recommendation methods and approaches. Particularly, we would like to explore ways to incorporate contextual information, such as user demographics and movie genres, into our recommendation system to further improve its performance. We believe this will also enable us to make a more interpretable system which can give us a better basis for explaining the recommendations to the end users.

Bibliography

[1] P. Resnick, and H. R. Varian, “Recommender systems,” *Commun. Acm*, vol. 40, no. 3, pp. 56–58, Mar. 1997, doi: 10.1145/245108.245121. Accessed: Apr. 19, 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/245108.245121

[2] R. Burke, “Hybrid Recommender Systems: Survey and Experiments,” *User Model. User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002, doi: 10.1023/A:1021240730564. Accessed: Apr. 19, 2023. [Online]. Available: http://link.springer.com/10.1023/A:1021240730564

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proc. 10th Int. Conf. World Wide Web*, Hong Kong Hong Kong, Apr. 2001, pp. 285–295, doi: 10.1145/371920.372071. Accessed: Apr. 19, 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/371920.372071

[4] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering,” arXiv, 2013. Accessed: Apr. 19, 2023. [Online]. Available: http://arxiv.org/abs/1301.7363 (Comment: Appears in Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI1998))

[5] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, “Evaluating collaborative filtering recommender systems,” *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004, doi: 10.1145/963770.963772. Accessed: Apr. 19, 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/963770.963772

[6] Z. Z. Darban, and M. H. Valipour, “GHRS: Graph-based Hybrid Recommendation System with Application to Movie Recommendation,” *Expert Syst. Appl.*, vol. 200, p. 116850, Aug. 2022, doi: 10.1016/j.eswa.2022.116850. Accessed: Apr. 20, 2023. [Online]. Available: http://arxiv.org/abs/2111.11293 (Comment: 14 pages, 13 figures, under review in an Elsevier journal)

[7] N. Hug, “Surprise: a python library for recommender systems,” *J. Open Source Softw.*, vol. 5, no. 52, p. 2174, 2020, doi: 10.21105/joss.02174. [Online]. Available: https://doi.org/10.21105/joss.02174