Movie Recommendation System

Darshee Machhar
Rutgers University
dm1639@scarletmail.rutgers.edu

Pankti Nanavati
Rutgers University
pn266@scarletmail.rutgers.edu

Vamsi Krishna Bulusu
Rutgers University
nb847@scarletmail.rutgers.edu

ABSTRACT

Recommendation systems have become increasingly important in recent years with the growth of online data. In this paper, we propose a movie recommendation system using stochastic gradient descent (SGD) for matrix factorization. We used the MovieLens dataset, which is widely used in the research community for movie recommendation systems. We preprocessed the data and conducted exploratory data analysis to understand the characteristics of the dataset. We then applied the Matrix Factorization to factorize the user-movie rating matrix into user and movie latent matrices. Following which we used SGD algorithm to learn the latent vectors and tune our model parameters. We evaluated our model using Precision, Recall, F-measure, NDCG metrics. Our model achieved an RMSE of 0.075 and a MSE of 0.275, which are competitive with state-of-the-art methods. Our study provides insights into the effectiveness of stochastic gradient descent for matrix factorization in the movie recommendation domain.

KEYWORDS

Recommender System, Content-based model, Latent Factor model, Collaborative Filtering Model, Cold Start, Matrix Factorisation, Stochastic Gradient Descent

1 INTRODUCTION

Summarize the background, problem setting, proposed method, and experiment results.

2 RELATED WORK

Movie recommendation systems have been extensively studied in the literature. One of the earliest methods proposed for recommendation systems is collaborative filtering (CF), which relies on the similarity between users or items to make recommendations. CF can be further classified into two categories: user-based and item-based CF. In user-based CF, the system finds users with similar tastes to the active user and recommends movies they have liked. In item-based CF, the system finds movies similar to those the active user has liked and recommends them. However, CF suffers from the sparsity problem when there are few ratings available for some users or movies. Matrix factorization (MF) is a popular method for overcoming the sparsity problem in recommendation systems.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

MF factorizes the user-movie rating matrix into two lower-dimensional matrices, representing user and movie latent factors. It then uses these factors to predict the ratings of unseen movies for a given user. The MF method has been widely used in various applications, including movie recommendation systems.

Alternating Least Squares (ALS) is a commonly used algorithm for matrix factorization in recommendation systems. ALS is an iterative algorithm that alternates between optimizing the user and item latent factors. Although ALS is a simple and effective method, it suffers from high computational complexity, especially for large datasets.

Stochastic Gradient Descent (SGD) is another popular algorithm for matrix factorization. It is an optimization method that updates the parameters of the model iteratively based on the gradient of the loss function. SGD is computationally efficient and can handle large-scale datasets. Moreover, SGD can be used for online learning, where the model is updated as new data becomes available.
Recently, deep learning-based methods have been proposed for recommendation systems, including autoencoders, convolutional neural networks, and recurrent neural networks. These methods have shown promising results, but they require large amounts of data and computational resources.

In the context of movie recommendation systems, various studies have used the MovieLens dataset, including traditional CF, MF, and deep learning-based methods. These studies have shown that MF is a simple and effective method for movie recommendation, and the use of algorithms such as ALS and SGD can further improve its performance.

3 PROBLEM FORMALIZATION

We aim to build a movie recommendation system that predicts movie ratings for users and provides personalized top-10 movie recommendations for each user. We will use the MovieLens dataset, which contains user-movie ratings.

Let R be the user-movie rating matrix, where R(i,j) denotes the rating given by user i to movie j. We split the dataset into a training set, consisting of 60% of the ratings, and a testing set, consisting of the remaining 40% of the ratings. We aim to develop a model that predicts the ratings in the testing set as accurately as possible.

After predicting the ratings, we will create a top-10 recommendation list for each user, containing movies that the user has not rated in the training set. We will evaluate the quality of the recommendation list using precision, recall, F-measure, and NDCG metrics.

We will use matrix factorization with stochastic gradient descent as our recommendation algorithm. The goal is to learn two low-dimensional latent matrices, one for users and one for movies, that can accurately predict the ratings. Our objective function is to minimize the mean of squared errors between the predicted ratings and the actual ratings.

Our aim is to build an accurate and trustworthy recommendation system that provides personalized movie recommendations to users based on their movie preferences and history. We aim to address the challenge of building a system that is unbiased and fair in its recommendations, and does not reinforce existing biases or stereotypes.

4 THE PROPOSED MODEL

We propose to use matrix factorization with stochastic gradient descent to train our recommendation model. The goal is to learn two low-dimensional latent matrices, one for users and one for movies, that can accurately predict the ratings. Our objective function is to minimize the mean of squared errors between the predicted ratings and the actual ratings.

Let R be the user-movie rating matrix, where R(i,j) denotes the rating given by user i to movie j. We assume that the ratings are generated by a linear combination of latent factors, such that:

$$R(i, j) = U(i) * M(j)^T$$

where U(i) is the i-th row of the user latent matrix, M(j) is the j-th row of the movie latent matrix. The goal is to learn the latent matrices U and M that minimize the reconstruction error between the predicted and actual ratings.

We use stochastic gradient descent to learn the latent matrices. At each iteration, we randomly sample a batch of training ratings (i,j) from the training set, compute the prediction error, and update the corresponding rows of the latent matrices using the gradients. We also use regularization to prevent overfitting and ensure that the latent matrices are sparse.

Our model has hyperparameters such as the number of latent factors, learning rate, regularization strength, and batch size. We will perform a hyperparameter search using cross-validation on the training set to select the best set of hyperparameters that optimize the prediction accuracy.

After training the model, we will use it to predict the ratings in the testing set and create a top-10 recommendation list for each user, containing movies that the user has not rated in the training set. We will evaluate the quality of the recommendation list using precision, recall, F-measure, and NDCG metrics.

5 EXPERIMENTS

5.1 Dataset

- The Movielens dataset is a widely used dataset in the development of recommendation systems. It comprises more than 1 million movie ratings provided by around 6040 users.
- The size of the dataset is 6 MB, which is relatively small for a recommendation system dataset but still substantial enough for experimentation and testing.
- Along with the ratings, the dataset contains movie metadata like genre, year, and title.
- The dataset has been divided into training and validation sets based on ratings, with a 60

5.2 Dataset Preprocessing

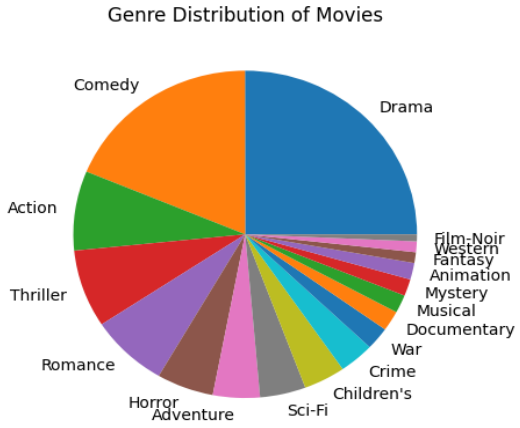
- Downloading and loading the data: The code downloads a movie rating dataset from the internet and loads two data files ('ratings.dat' and 'movies.dat') into pandas dataframes. The 'ratings.dat' file contains user ratings for various movies, while the 'movies.dat' file contains information about each movie, such as the title and genre.
- Encoding user and movie IDs: To prepare the data for training a recommendation model, the user and movie IDs in the ratings dataframe are encoded into sequential integers using a dictionary. This ensures that each user and movie has a unique ID that can be used by the model.
- Mapping movie IDs to their titles: A dictionary is created to map the encoded movie IDs to their respective movie titles. This is useful for displaying the recommended movies to the user in a more user-friendly format.
- Splitting the data into training and validation sets: Finally, the ratings data is split into a training set and a validation set using the train_test_split function from the scikit-learn library. This is necessary to evaluate the performance of the recommendation model on data that it has not seen during training.

5.3 Data Analysis

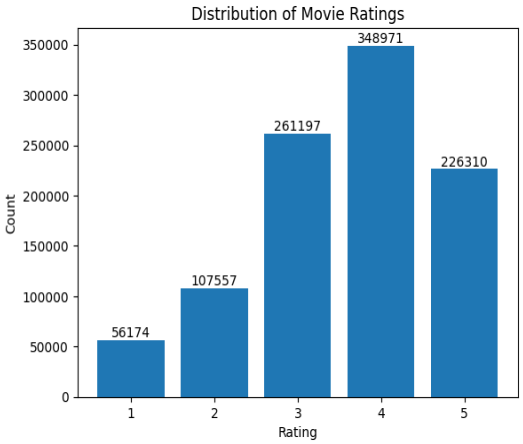
- Statistics for number of movies, users, ratings and average ratings per (user/movie):

	Statistic	Value
0	Number of Movies	3,883
1	Number of Users	6,040
2	Number of Ratings	1,000,209
3	Average Ratings per User	165.60
4	Average Ratings per Movie	269.89

- Genre Distribution of movies:



- Distribution of movie ratings:



5.4 Latent Factor Model

The traditional SVD algorithm can help to find hidden common elements and concepts among movies, which can be used to predict missing values of ratings. Since this approach is computationally intensive and can generate sparse matrices that leads to inaccuracies, we decided to use Latent Factor models instead.

Latent factor models are a class of recommendation algorithms that use matrix factorization techniques to uncover underlying patterns or "latent factors" in user-item interaction data. The basic idea behind latent factor models is that there are certain unobserved or hidden factors that influence user preferences and item characteristics, and by identifying these factors, the algorithm can make more accurate recommendations.

In matrix factorization, the user-item interaction data is represented using a matrix, where each row is a representation of a user and each column that of an item(in this case, a movie). The goal of matrix factorization is to factorize this matrix into two lower-dimensional matrices, one for users and one for items, such that the dot product or scalar product of the corresponding user and item vectors approximates the observed ratings or interactions in the original matrix.

The Latent factor models we have implemented uses a small number of latent factors, such as movie genres, etc, to represent the preferences and characteristics of users and items. These factors are learned by optimizing a loss function that minimizes the difference between the predicted ratings and the actual ratings in the training data.

One of the advantages of latent factor models is that they can handle sparse data, where users have rated only a small fraction of the available items. They can also be used for cold-start recommendations, where there is no prior information about a user or item.

5.5 Collaborative Filtering

Collaborative filtering is a popular recommendation technique that uses past interactions or preferences of users to predict their preferences for new items. The basic idea behind collaborative filtering is that users who have similar preferences in the past are likely to have similar preferences in the future. In other words, the recommendations are based on the preferences of similar users, rather than on the characteristics of the items themselves.

Collaborative filtering can be done using two main approaches: user-based and item-based. In user-based collaborative filtering, the algorithm finds users who have similar preferences to the target user and recommends items that these similar users have liked. In item-based collaborative filtering, the algorithm finds items that are similar to the ones the target user has liked in the past and recommends these similar items.

Collaborative filtering algorithms typically use a similarity metric to measure the similarity between users or items. Common similarity metrics include cosine similarity, Pearson correlation, and Jaccard similarity. In our project we apply cosine similarity using the dot product of the user and movie vectors.

One of the advantages of collaborative filtering is that it does not require explicit knowledge of the characteristics of the items, which can be useful in domains where the item features are difficult to define or measure. Collaborative filtering can also capture dynamic user preferences and adapt to changes in user behavior over time.

5.6 Fairness and Unbiases

Bias is a pervasive issue in recommendation systems that can lead to unfair or discriminatory recommendations. Biases can arise due to various factors such as historical data, user preferences, and algorithmic design. Therefore, it is crucial to address biases in recommendation systems to ensure fairness, transparency, and user trust. Here are some ways we handled biases in recommendation

systems:

- Data pre-processing: One way to address biases in recommendation systems is by pre-processing the data to remove any biases or noise. This can be done by removing outliers, normalizing the data, and balancing the distribution of user-item interactions.
- Regularization: Regularization techniques can be used to avoid overfitting and reduce the impact of biases in the recommendation algorithm. Regularization methods such as L1 and L2 regularization can help prevent the algorithm from focusing too much on individual users or items and instead promote a more balanced and diverse set of recommendations.
- Reduce over-generalization: Adding biases to a recommendation system can be a way to reduce over-generalization and improve the accuracy of the recommendations. Biases can be introduced into the recommendation algorithm to capture specific characteristics or preferences of the users or items. By incorporating these biases, the algorithm can better capture the nuances of the data and make more accurate recommendations. For example, in a movie recommendation system, a user may have a strong preference for action movies. By introducing a bias towards action movies, the recommendation algorithm can give more weight to action movies in the recommendations for that user.

6 RESULTS

We evaluated our model using Precision, Recall, F-measure, NDCG metrics.

Recall = $\frac{TruePostive}{TruePositive + FalseNegative}$

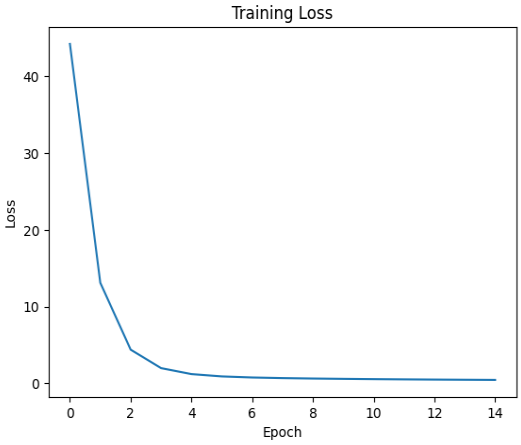
Precision = $\frac{TruePostive}{TruePositive + FalsePostive}$

F1 Score= $\frac{2 * Precison * Recall}{Precision + Recall}$

NDCG = $\frac{DCG}{IDCG}$ where,

DCG (Discounted Cumulative Gain) = $\sum \frac{2^{rel(i)} - 1}{log_2(i+1)}$, where rel(i) is the relevance score of the i-th ranked item.
IDCG (Ideal Discounted Cumulative Gain) = DCG of the ideal ranking, where items are sorted in descending order of relevance.

MSE (Mean Squared Error) = $\frac{\sum(y_{true} - y_{pred})^2}{n}$, where n is the number of samples, y_{true} is the true target value, and y_{pred} is the predicted target value.



	Loss Function
MSE	0.0757224190643579
RMSE	0.27517706856560176

	Evaluation Metrics
Recall	0.5784641032208768
Precision	0.7038179551649094
F_Score	0.635013832725297
NDCG	0.8273042336595495

Here is a sample unwatched movie recommendation for a user.

	Movie Title	Genre
0	Ghostbusters (1984)	ComedylHorror
1	Crime and Punishment in Suburbia (2000)	ComedylDrama
2	Friday the 13th (1980)	Horror
3	Sixteen Candles (1984)	Comedy
4	Bio-Dome (1996)	Comedy
5	Gone Fishin' (1997)	Comedy
6	Frankenstein (1931)	Horror
7	Frances (1982)	Drama
8	Out to Sea (1997)	Comedy
9	Tampopo (1986)	Comedy

7 CONCLUSIONS AND FUTURE WORK

In conclusion, collaborative filtering using matrix factorization is a popular technique for recommendation systems that involves decomposing a user-item interaction matrix into lower-dimensional latent factors. This approach has been shown to effectively handle sparse and high-dimensional data, and can provide accurate recommendations even for new or unseen items.

One future direction for collaborative filtering using matrix factorization is to explore more advanced models that incorporate additional features such as temporal dynamics or user demographics. Another potential area for improvement is to develop more efficient algorithms for large-scale matrix factorization, as current methods can become computationally expensive for very large datasets.

Furthermore, recent research has also focused on addressing issues related to fairness, diversity, and serendipity in recommendation systems. In particular, incorporating diversity and serendipity into matrix factorization models has shown promising results in increasing user satisfaction and engagement.

Overall, collaborative filtering using matrix factorization is a versatile and powerful technique for recommendation systems, and ongoing research will likely continue to advance its effectiveness and applicability in various domains.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor, Prof. Yongfeng Zhang, for his valuable guidance and support throughout the course of this project. His insightful comments and suggestions have greatly contributed to the improvement of this work.
I would also like to thank my colleagues and classmates who provided useful discussions and feedback during the development of this project. Their insights and constructive criticisms have helped me to refine my ideas and improve the quality of this work.

REFERENCES

(1) https://towardsdatascience.com/movie-recommendation-system-based-on-movielens-ef0df580cd0e
(2) https://www.scaler.com/topics/machine-learning/recommender-system-using-movielens/
(3) https://en.wikipedia.org/wiki/Recommender_system
(4) https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
(5) https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b