# Game Recommender System

**Nguyễn Trần Nhật Bảo Nguyễn Đăng Khoa**

**Abstract**

A recommender system aims to provide users with personalized online product or service recommendations to increase sales and profit while presenting users with the most fitting and suitable product. Most recommender systems rely on explicit feedback like ratings or reviews but it is not always certain that an user will leave an explicit feedback. In those cases, we need a system that can utilize implicit feedback, information that can be collected simply by monitoring the user's activities. One rapidly growing area where this kind of recommender system is useful is the gaming industry. This paper therefore discusses one approach to the implicit feedback recommender system and uses it to generate possible games for players.

**Keywords:** Recommender systems, game, implicit, collaborative filtering, alternating least square

# Introduction

As gaming is growing in popularity, an important challenge is providing gamers with a large variety of suitable games to their taste, therefore increasing sales for game distributors. One of the tools that address this challenge is recommender systems. These systems provide users with personalized recommendations for products or services, which hopefully suit their unique taste and needs.

There are three main types of recommender systems: collaborative filtering, content based filtering and hybrid recommender systems. The content-based filtering uses the description of the items in order to recommend items similar to what a user likes. The hybrid recommender system consists of combining the content-based and the collaborative filtering. Our algorithm in this work, collaborative filtering, is based on the principle that if two people liked the same things in the past, if one of them likes something new, the other is likely to like it too. The advantage of the collaborative filtering method is that the algorithm doesn’t need to understand or process the content of the items it recommends. The only required information is the past behavior of users, which might be their previous transactions or the way they rate products.

Recommender systems rely on different types of input. Most convenient is the high quality explicit feedback, which includes explicit input by users regarding their interest in products. However, explicit feedback is not always available. Thus, recommenders can infer user preferences from implicit feedback, such as purchase history, browsing history, search patterns, time spent playing the games, etc, with no ratings or specific actions needed.

# Related Works

Syed Muhammad Anwar. 2017 [1] proposed using collaborative filtering to create a Game recommender system. The authors used two different methods: item-based filtering and user-based filtering. Item-based filtering focuses on finding similarity between the game that the user has picked and one other game. After finding similarity of picked unrated game and all the rated games, selection of those rated games is made which are in the nearest neighbor (having maximum similarity with the unrated) of unrated picked game. Using these nearest neighbors, the system predicts the rating of the picked unrated game for active users. With user-based filtering, the model will find similarity between the active user and a set of other chosen gamers. The unrated games were sorted in descending order of prediction. The system then passes the list of unrated games having the highest prediction to the user-based recommendation part of the algorithm. Similarity between the active user and other chosen gamers is then calculated using Pearson correlation formula. If the system predicts the rating of the chosen game greater than average 3.0 (since a scale was set for rating games from 1 - 5), then the game will be recommended to user/gamer, otherwise it will not be recommended. The next priority game from the list of unrated games will be picked and similar steps will be followed to either recommend or move on to the next game. The authors assembled and analyzed 1200 ratings of 30 games in 5 different genres. The survey included 90 users. The model achieved 92% accuracy.

The first implicit feedback version of the Matrix Factorization model was introduced by Hu et al. 2008 [2]. One of their main findings is that implicit user observations should be transformed into two paired magnitudes: preferences and confidence levels. In other words, for each user-item pair, they derive from the input data an estimate to whether the user would like or dislike the item (“preference”) and couple this estimate with a confidence level.

Chen et al. [3] diagnose the baseline implementation of Alternating Least Square (ALS) , a Matrix Factorization algorithm, and observe that it is a lack of awareness of the hierarchical thread organization on modern hardware. This leads to inefficient use of hardware resources: unbalanced thread use and scattered memory access. Thus, the thread batching technique and three architecture-specific optimizations are applied. On the other hand, they implement the ALS solver in OpenCL so that it can run on various platforms (CPUs, GPUs, and MICs). Based on the architectural specifics, they select a suitable code variant for each platform to efficiently map it to the underlying hardware. The experimental results show that their implementation performs 5.5× faster on E5-2670 and 21.2× faster on K20c than the baseline implementation.

This assignment will attempt to use the ALS Matrix Factorization algorithm on the Steam game dataset to build a recommender system for games that are suitable and engaging for players, encouraging them to spend money to purchase new games, utilizing data from Steam itself.

# Data preparation

This project uses the Steam Video Games from Kaggle. Steam is the world's most popular PC Gaming hub. The collected data is a list of user behaviors, with columns: user-id, game-title, behavior, value. Each row of the dataset represents the behavior of a user towards a game, either ‘play’ or ‘purchase’. If the behavior is ‘play’, the value associated with it corresponds to the amount of hours played. If the behavior is ‘purchase’, the value associated with it is 1, meaning the user purchased the game. In the case of this user dataset, the value associated with ‘purchase’ is always 1.

The original dataset doesn’t have headers and contains a non-meaning 0 column, and those shown in the table below are added for convenience based on the data description.

| **user\_id** | **game\_title** | **behavior** | **value** | **0** |
| --- | --- | --- | --- | --- |
| 151603712 | The Elder Scrolls V Skyrim | purchase | 1 | 0 |
| 151603712 | The Elder Scrolls V Skyrim | play | 273 | 0 |
| 151603712 | Fallout 4 | purchase | 1 | 0 |
| 151603712 | Fallout 4 | play | 87 | 0 |
| 151603712 | Spore | purchase | 1 | 0 |
| 151603712 | Spore | play | 14.9 | 0 |

Table 1: Original dataset

The dataset contains a total of 200,000 rows, including 5,155 unique games and 12,393 unique users.

The dataset structure is reformatted by combining 2 rows of interaction between an user and a game into 1 row. For each row, only user\_id, game\_name and hours values are kept as they are the main features of the problem.

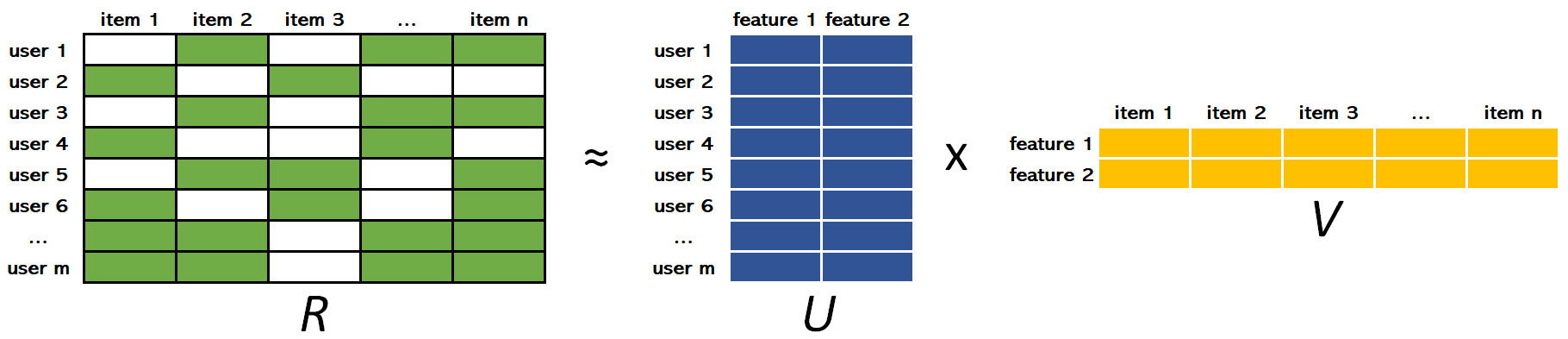
| **user\_id** | **game\_name** | **hours** |
| --- | --- | --- |
| 151603712 | The Elder Scrolls V Skyrim | 273 |
| 151603712 | Fallout 4 | 87 |
| 151603712 | Spore | 14.9 |
| 151603712 | Fallout New Vegas | 12.1 |
| 151603712 | Left 4 Dead 2 | 8.9 |

Table 2: Dataset after cleaning

# Method

This section describes an implementation of a collaborative filtering recommendation algorithm using matrix factorization with implicit data. Implicit data is implicitly expressed by the amount of hours users play a game, contrary to explicit data like rating or preference of users towards a game.

The Alternating Least Squares (ALS) is the model used to fit the data and to generate recommendations. The ALS algorithm uses matrix factorization, which is basically taking a large matrix and factoring it into smaller matrices whose product equals the original one. For our case of collaborative recommender system with implicit data, the matrix factorization mathematically reduces the original matrix “all users vs all items” into smaller matrices “all users vs some features” and “all items vs some features”. The mentioned “latent” features are learnt from the data and don’t necessarily represent any real metadata.



In the figure above, R is the original matrix of user-items containing some kind of implicit data (hours played) within it. U and V have weights measuring how each user-item relates to each feature. The goal is to compute the weights of U and V such that R ≈ U x V. The ALS (Alternating Least Squares) algorithm iteratively alternates (hence its name) between optimizing U and fixing V and vice versa until a convergence that approximates R the best it can.

Hu et al. analyze the implicit feedback measure through the insertion of two new measures that represent the preference and confidence that exist behind the uptake of an item [2]. Preference tells us if an item is consumed or not.

Where is some measure of implicit feedback. The second measure stands for the confidence of that preference.

Where α is a linear scale factor that sets more importance to relevant items, above never played ones. The value α = 40 is usually used as a result of the original paper [2] but the value 15 in our code is also suggested as a suitable value. Under these definitions, the search of latent factors for users and items is made by the optimization of the following loss function.

For our project we use the ALS model implemented in the implicit python library. The library utilizes Conjugate Gradient Method, multi-threaded training routines, also Cython and OpenMP to fit the models in parallel among all available CPU cores. In addition, the ALS model has custom CUDA kernels, speeding up the algorithm and reducing training time significantly.

# Result and discussion

After training the model, we use Mean Average Precision at K (mAP@k) score to evaluate the model. Mean Average Precision at K is the mean of the average precision at K (APK) metric across all instances in the dataset. APK is a metric commonly used for information retrieval. APK is a measure of the average relevance scores of a set of the top-K documents presented in response to a query. For each query instance, the set of top-K results will be compared with the set of actual relevant documents, that is, a ground truth set of relevant documents for the query.

The AP@k and mAP@k formulas and computations are as follows:



In which TP stands for True Positives, whereas N(k) and TP seen can be calculated from the following formulas.





After calculated AP@k for every user, the mAP@k then will be equal to the average of overall AP@k



The result is shown in Table 1

| **Metric** | **Score** |
| --- | --- |
| mAP@k | 0.1025355 |

Table 3: Result of the ALS model

The top precision score comes out pretty low. However, we think that this score is not enough to tell whether the model is working or not. More domain knowledge based should also be used to evaluate the result. For example, here are the games that our model recommend for a user:

| **User previous game** | **Games recommended by the model** |
| --- | --- |
| 0 Alien Swarm  1 Cities Skylines  2 Counter-Strike  3 Counter-Strike Source  4 Day of Defeat  5 Deathmatch Classic  6 Deus Ex Human Revolution  7 Dota 2  8 Half-Life  9 Half-Life 2  10 Half-Life 2 Deathmatch  11 Half-Life 2 Episode One  12 Half-Life 2 Episode Two  13 Half-Life 2 Lost Coast  14 Half-Life Blue Shift  15 Half-Life Opposing Force  16 Portal  17 Portal 2  18 Ricochet  19 Team Fortress 2  20 Team Fortress Classic | 1. Left 4 Dead 2  2. Left 4 Dead  3. XCOM  4. Aliens vs. Predator  5. Fallout 3 - Game of the Year Edition  6. Amnesia The Dark Descent  7. Audiosurf  8. BioShock  9. Guns of Icarus Online  10. Call of Duty 4 Modern Warfare |

Table 4: Top 10 game recommendation for an user

As can be seen, this user mainly plays first-person shooter games such as Half-life, Counter-Strike or Team Fortress 2. Our model is able to suggest similar games such as Left 4 Dead, Call of Duty 4, Aliens vs Predator… for this user.

# Conclusion and perspective

We think our model performs quite well but there is still room for improvement. Our proposed user rating based on playtime still has many flaws. It cannot perfectly reflect a user’s view of the game. There are many cases where a game can have very little play time but still the user enjoys playing it. For example, “Journey” is a 2015 indie game that players are able to finish within 90 minutes. On the other hand, “Assassin’s Creed Valhalla” takes about 91 hours to finish. That doesn’t mean that the user is not enjoying “Journey” as much as “Assassin’s Creed Valhalla”. In the future we would create a better rating system to better the recommender.

Another problem our recommender is facing is the cold-start problem. This issue occurs when the model can’t recommend new users and new items. The cold-start problem has been a traditional issue for the Collaborative-Filtering model. In the future when we have more time and more computational resources, we would like to tackle this problem using more powerful deep learning, neural network models such as AutoRec, DeepRec, Collaborative Denoising Auto-encoder, etc.

# References

[1] Syed Muhammad Anwar. 2017. A game recommender system using collaborative filtering (GAMBIT).

[2] Yifan Hu, Yehuda Koren and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets.

[3] Jing Chen, Jianbin Fang, Weifeng Liu, Tao Tang, Xuhao Chen and Canqun Yang. 2017. Efficient and Portable ALS Matrix Factorization for Recommender Systems.

# 

# Appendix A. Project plan management

| **Task Name** | **Priority** | **Owner** | **Start date** | **End date** | **Status** |
| --- | --- | --- | --- | --- | --- |
| Find documents | High | Khoa | 14/05/2022 | 20/05/2022 | Finished |
| Review related papers | Medium | Bảo, Khoa | 14/05/2022 | 24/05/2022 | Finished |
| Review and analyze public dataset | Low | Bảo | 14/05/2022 | 20/05/2022 | Finished |
| Collect and label data | High | Khoa | 20/05/2022 | 01/06/2022 | Finished |
| Evaluate potential method | Medium | Khoa | 20/05/2022 | 01/06/2022 | Finished |
| Experiment | Low | Bảo, Khoa | 01/06/2022 | 10/07/2022 | Finished |
| Compare results | Medium | Bảo, Khoa | 10/07/2022 | 20/07/2022 | Finished |
| Writing appendix | Low | Bảo, Khoa | 10/07/2022 | 20/07/2022 | Finished |
| Future works | High | Khoa | 10/07/2022 | 20/07/2022 | Finished |

# Appendix B. Source code and data

| Item | Link | Description |
| --- | --- | --- |
| Data | <https://drive.google.com/file/d/1fGw3uC9WOpaBi4QbrRxuqr32tl6Y_Y3d/view?usp=sharing> |  |
| Source code | <https://drive.google.com/file/d/1ZOfMGuw-mMhFNqupayfFTTWa3TvvnMKA/view?usp=sharing> |  |