Game Recommendations with Collaborative Filtering on Steam*

Jiayan Dong
jid001@ucsd.edu
University of California San Diego
La Jolla, California, USA

Zhuoliang Pu
zhpu@ucsd.edu
University of California San Diego
La Jolla, California, USA

Kangming Yu
kay003@ucsd.edu
University of California San Diego
La Jolla, California, USA

Figure 1. Steam Autumn Sale

Abstract

Online shopping is an indispensable part of modern people's lives and an important part of the modern economy. People can freely and quickly choose and buy from thousands of commodities. However, the wide variety of products also makes it difficult for users to quickly find their favorite products, so the product recommender system is a very important part of online shopping. In recommender systems, people often only pay attention to the interaction between the user and the item, or only pay attention to the obvious similar features between the item and the item (such as label, price, release time, etc.). However, in a realistic recommendation system, we often need to combine the two to better achieve personalized recommendations for users. In this paper, we compare and analyze the model that only focuses on the interaction between users and items, and the model that combines the two. We used a data set from the Steam video game distribution platform. The data set contains records of users purchasing games as well as information of games. We also analyzed the simple model that has been fine-tuned, as well as the Factorization Machine model. We tested the performance of these two models when new features were added. It turns out that when selecting appropriate game and user interaction features and game information features, both the simple model and the Factorization Machine model can give accurate game recommendations, but the simple model does not perform well when adding features, while the Factorization Machine model can be effective to use new features to provide the most accurate predictions.

Keywords: Recommender System, Factorization Machine, Collaborative Filtering

*CSE 158 Assignment 2

1 Exploratory Analysis

1.1 Interpretation of Datasets

For dataset analysis, we selected four relevant datasets for Steam games, including all reviews of 25,799 users of some of the games they purchased, 88,310 users and the games they own, 615 game bundles, and 32,135 Steam games[1].

- **Dataset 1** (all reviews of 25,799 users of some of the games they purchased) : This dataset includes when each user posted a review, whether the game was recommended, and the review content.
- **Dataset 2** (88,310 users and the games they own) : This dataset contains a collection of the game ID, name, and time of play of each user-owned game.
- **Dataset 3** (615 game bundles) : This dataset contains the price per bundles, the set of games per bundles, and basic game information.
- **Dataset 4** (32,135 Steam games) : This dataset contains each game's publisher, genres, game ID, name, release date, tags, price, etc.

According to the information contained in the data sets, we focus on in-depth analysis of two datasets (Dataset 2 and Dataset 4).

1.2 Exploratory Analysis for Dataset 2

After analyzing Dataset 2, we found the following interesting phenomena:

- We found that each user has a user ID and a Steam ID. But in other datasets, user ID are used to identify users. So, we use the user ID to represent each user. Although we found that two users had the same user ID, we ignored it because it could be that the information of the account was collected twice before and after the user ID was changed.
- We also found that there were 17,398 users who had not purchased the game, and we could not obtain their other information from any other datasets, so these users would be a challenge in the construction of a personalized recommendation system because it is a cold start problem.
- In addition, we found that some players might not play some of the purchased games. Excluding 17,398 users who did not purchase games, the average number of games played as a percentage of the games they purchased was 68.22%, and the median was 72.13%. Thus, the purchase of a game does not mean that the user is interested in the game, but may be caused by other inducements, such as the price of the game. Therefore, whether a user buys a game is not the same predictive question as whether a user likes a game.
- The average number of games purchased per user was 72.67. Since we don't know why people without buying games do not buy games, and the percentage of people without buying games is high, we exclude people without buying games from our calculation of the average number of games purchased per user.
- From this dataset, we can also get information about the popularity of games by counting the number of times each game is purchased. We can rank the popularity of a game by ranking the number of purchases in descending order (see Figure 2), and we can define the number of purchases of a game as popularity. However, we found that the distribution of prevalence was extremely unbalanced (see Figure 3). A total of 10,978 games were purchased, 10,407 games with a popularity between 1 and 2000, but only 299 games with a popularity between 2000 and 4000, and only 15 games with a popularity greater than 20,000. We believe that popularity is a useful feature for predicting whether a user will buy a game, although it may not be a personalized recommendation.

	Game name	Nums of Purchase
0	Dota 2 Test	49571
1	Counter-Strike: Global Offensive	43776
2	Garry's Mod	43301
3	Unturned	38682
4	Left 4 Dead 2	37044
5	Left 4 Dead 2 Beta	37044
6	Terraria	29239
7	Warframe	25807
8	Portal 2	24465
9	Counter-Strike: Source	24220

Figure 2. The top 10 most bought games

	Popularity (Num of purchases)	Num of games
0	[1 - 2000)	10407
1	[2000 - 4000)	299
2	[4000 - 6000)	97
3	[6000 - 8000)	63
4	[8000 - 10000)	46
5	[10000 - 15000)	33
6	[15000 - 20000)	18
7	>= 20000	15

Figure 3. Distribution of game count and popularity

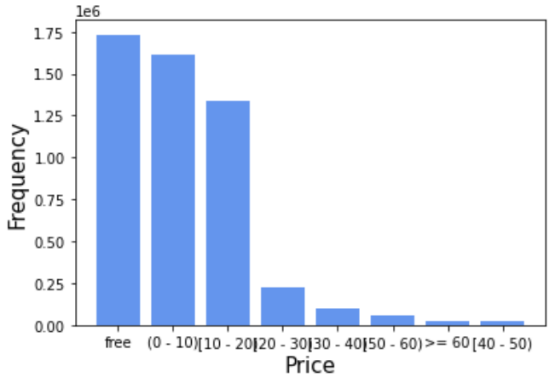

Figure 4. The Price of the Game Versus the Number of Purchases

1.3 Exploratory Analysis for Dataset 4

After analyzing Dataset 4, we found the following interesting phenomena:

- There are outliers in the dataset. In some cases, the game name based on the game ID is empty string. For example, the game name of the top 1 most popular game should be "Dota 2 Test" (queried in the Steam database), but the game name in the dataset is empty string. Therefore, we use the game ID to identify the game, not the game name.
- We found a clear correlation between the distribution of the game price and the number of games purchased. Basically, the lower the price of a game, the more times that game will be purchased (see Figure 4). For free games, there were 1,733,362 purchases, but for games over 60 dollars, there were 19,513 purchases. So, our guess is that the price of a game is related to whether a user buys the game, and that might help predict whether a user will buy the game. Moreover, the game price can either help construct a non-personalized recommendation system, or it can be a feature of a personalized recommendation system.
- As for game genres, we analyzed the relationship between game genres and users (see Figure 5). In total, there are 22 game genres in the dataset. Every user (excluding those who without buying games) had an average of 9.46 game genres, and the median was 10. Since this number is nearly half of the total number of genres, it may be difficult to construct a non-personalized recommendation system based on whether a game has some genres or not. However, the feature (game genres) may help us calculate the similarity between users and construct personalized recommendation system.

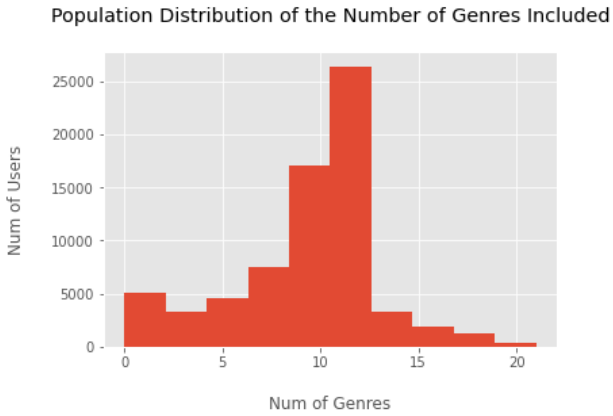

Figure 5. Population Distribution of the Number of Genres Included

- Also, we analyzed the relationship between the game tags and users (see Figure 6). As with the analysis of game genres, we calculated that every user (excluding those who without buying games) had an average of 128.38 tags, and the median was 126. Although the average of tags per user is also close to half of the total tags, we can still try to use this feature to help calculate the similarity between users to construct a personalized recommendation system.

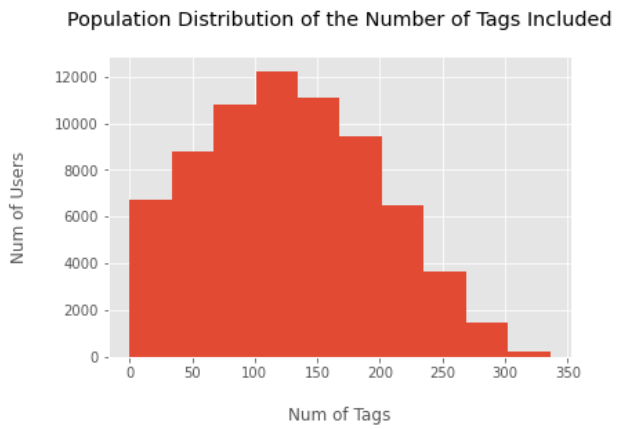

Figure 6. Population Distribution of the Number of Tags Included

- Other information such as the date the game was released will also help us build personalized recommendations.

2 Predictive Task

2.1 Task identification

After the exploratory analysis of these datasets, we recognized that users using Steam are more than just gamers. They could also be collectors who just want to collect games but barely play most of their collections. As a video game distribution platform, we believe the primary goal for Steam is to sell games. Therefore, we identify our predictive task as: given a (user, game) pair, we predict if this user would buy this game or not.

2.2 Input data construction

First, we filtered out users who had never purchased any game, and we also only considered games that were bought by at least one user. Then for each user, we randomly choose games they didn't purchase to construct our input vector, so that each user vector has the same amount of games purchased and not purchased. Although to be aware that in the whole dataset there are 4 people who purchased more than half of the games that are purchased by at least one user, so we didn't make them have the equal amount of purchase and none-purchase. Although considering that they only account for 0.01% of the overall users, we believe that it is fine to keep it that way.

2.3 Evaluation method

For our predictive task, we divided the dataset of users who have bought at least 1 game and the games they bought into training, test, validation sets with a ratio of 8:1:1, in which we perform model training on the training set, and choose the best performing approach on validation set, and test it on test set. The 2 evaluation matrices we used are prediction accuracy and AUC (Area Under the ROC curve), where we only used accuracy on our simple popularity based model (described in section 3) and used both of these matrices on the Factorization Machine model (also described in section 3).

2.4 Feature identification

After the exploratory analysis, we found many features seem useful for our predictive task:

1. Tags of games: Just like most people prefer to live in their comfort zones, we believe that gamers would prefer buying games with tags they already enjoy. Therefore, we believe that this could be very useful to predict if a user would like to buy a game or not.
2. Publisher: We also noticed that some users show loyalty to certain game publishers, therefore we believe that this would also be an important thing that could be used in our training model.
3. Specification: Specification is also a very useful game classification standard. Some players like to play Single-player games, so they have a higher probability of buying Single-player games. Others like to play online games, so they prefer to buy online games. Specification distinguishes games from the way of play, and the player's preference for how to play will also determine whether to buy a game or not.
4. Price: Budget would also be an important factor on if a user would buy a game or not. From our exploratory analysis, we discovered that the amount of games owned by users is proportional to the price of games. There could be many interpretations on this phenomenon, for instance, for game collectors they could add more collections with the same budget. And for regular games they could less financial stress to try different games. But no matter what, we believe that this is an important factor that we should utilize in our model.

5. Discount Price: Steam is famous in its frequent discount events. Most people wait until the game is on sale before buying, so the discount price of a game is as important as its price.
6. Popularity: We found that for the top 5 most popular games, half of the users in our dataset had bought them. So we believe that this would definitely be a feature that could have a huge impact on users' game selection. Publishing year: We discovered that among the most bought games, some of them are fairly new, and some of them are classics that have been published a long time ago. Thus, we believe that the publish date would also play certain roles in users' game selection.

3 Appropriate Model

3.1 Simple popularity based predictor

From our exploratory analysis we quickly discovered that the popularity of the games may play an important role on users' decision on purchasing a game, therefore, we started with building a simple popularity based model, which would predict the user would buy if the given game's popularity is above some threshold. For evaluation, we used our validation set, and computed the accuracy between the prediction and the label in the validation set. On our initial attemptation, we simply used a threshold of the 50th percentile of popularity, and got an accuracy of 75% as our baseline. Then to maximize the potential of this model, we tried different thresholds and eventually got a max accuracy of 90% with the threshold of 1325 (see Figure 7).

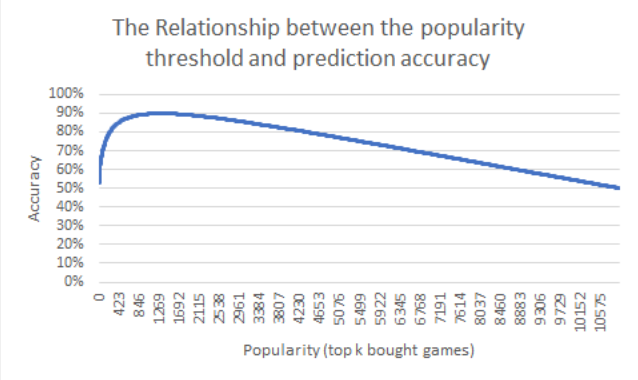

Figure 7. The Relationship between the popularity threshold and prediction accuracy

We were honestly surprised by the performance this model could achieve. Thus, we decided to incorporate more features into this model. As of now the simple popularity based model doesn't have any features that are useful to provide personalized recommendations, so we immediately thought the similarity of games' tags would be a useful tool to start with. Specifically, given a (user, game) pair, we would compute the Jaccard similarity between the tags of the given game and the tags of all the other games this user has bought. If the max of these similarities exceeds a certain threshold, we predict it as would buy. This feature alone would give a maximum of 65% accuracy (see Figure 8), so we believe that it could potentially have some effect on improving the accuracy of our popularity based prediction.

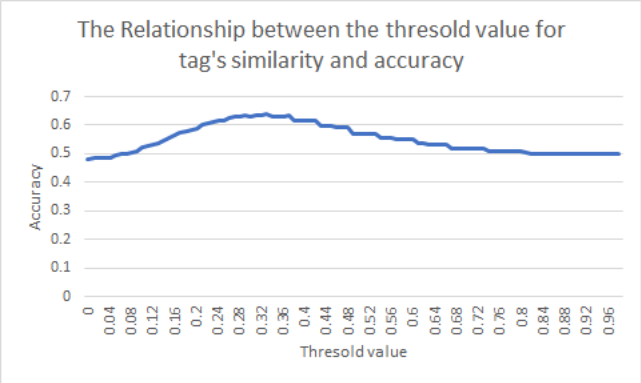

Figure 8. The Relationship between the threshold value for tag's similarity and accuracy

However, when we incorporated this feature with our best popularity based model, we surprisingly found that the similarity of tags had completely no effect on changing the accuracy. We also recognized that the price and publishers of games could be useful features since budget is undoubtedly important when making decisions, and it is also common that some gamers are loyal to certain publishers. We tried to include both features to our best popularity based model in various forms:

- Comparing the price with the average price the user spent
- Comparing the price with the most price range the user has spent on
- Calculating how many games with the same publisher of the given game the user has bought
- Check if the given game share the same publisher with this user's top K most played games

Besides this, we also tried to utilize the similarity between the given user and the users who had bought the given game, but unfortunately, all of these attempts

lead to a worse accuracy on our validation set. Therefore, even though the popularity based model seems to provide a promising prediction accuracy on our validation set, since we were not able to add any useful features for personalized prediction, we decided to use a more complex model.

3.2 Factorization Machine model

In the selection of complex models, we used the Factorization Machine model here, and focused on 1) using Alternating Least Square (ALS) solver, 2) using Markov chain Monte Carlo (MCMC) solver[12]. Factorization Machine has proven its versatility in recommender systems in many studies, that is, it can effectively capture the information in the history of interaction between users and items, and decompose users and items into latent factors. Then it's parameters are determined by machine learning methods, user bias, item bias and model parameters, and finally an efficient prediction model is generated.

$$f(x) = w_0 + \sum_{i=1}^{F} w_i x_i + \sum_{i=1}^{F} \sum_{j=i+1}^{F} \langle \gamma_i, \gamma_j \rangle x_i x_j \qquad (1)$$

In the Steam data set, this Factorization Machine model combines the record of the user's purchase of the game with the features of the game itself, and then generates a set of latent factors to represent user, game, and explicit features of game. This allows us to capture the user's preference for the game's explicit features, such as whether the user prefers a certain publisher, a certain type of game, or a certain price range while capturing the history of the user's purchase of the game.

Both Factorization Machine models with ALS solver and MCMC solver can efficiently and accurately predict whether a user has purchased a certain game. However, the model using the ALS solver requires us to give the initial parameters of the L2 regularization value, which means that we need to perform tuning hyperparameters many times to obtain the best model. On the other hand, although the model using the MCMC solver integrated regularization parameters and let the model decide by itself, its accurate prediction requires us to predict the results while training the model, which slows down the prediction process and increases the amount of calculation. When using the Factorization Machine model, the most important parameter is the size of the latent factor (Rank of the model). When the Rank is very small, and the size of our latent factor is small, that is, the latent factor contains less information. At this time, the model will not be able to fit the training data, so it is easy to underfit. When the Rank is very large, the size of our latent factor is large, that is, the latent factor contains more information. At this time, the model will overfit the training data.

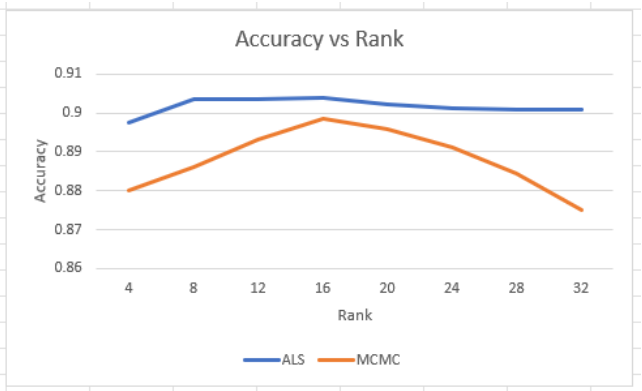

Figure 9. Accuracy vs Rank for Factorization Machine

We can see that when the Rank is 16, our Factorization Machine model has the best prediction (see Figure 9). At this point, we can go further and adjust the initial parameters of the L2 regularization value of the model using the ALS solver.

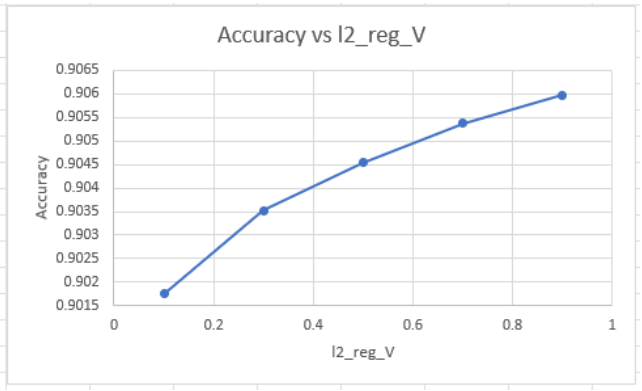

Figure 10. Accuracy vs l2_reg_V for Factorization Machine

We can see that when l2_reg_V is 0.9, the Factorization Machine model using the ALS solver has the best prediction (see Figure 10).

Compared with the abnormal phenomenon that the accuracy decreased in the simple model when add new features, the Factorization Machine model is more normal, that is, adding related features can improve the accuracy(see Figure 11):

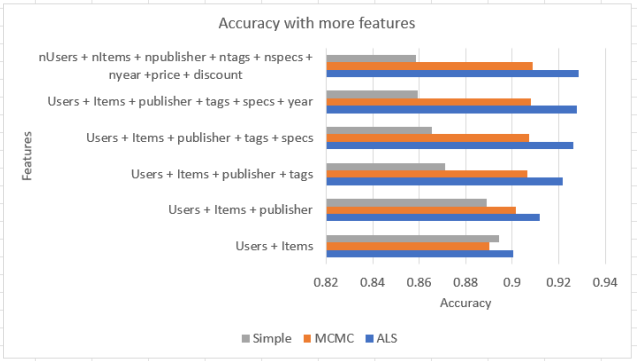

Figure 11. Accuracy with more features for 3 models

The last is the runtime analysis. The runtime of the simple model is better than that of the Factorization Machine model (see Figure 12). The Factorization Machine model involves a lot of machine learning content and matrix calculations, so a larger amount of calculation is required, and the calculation of the simple model is relatively simple.

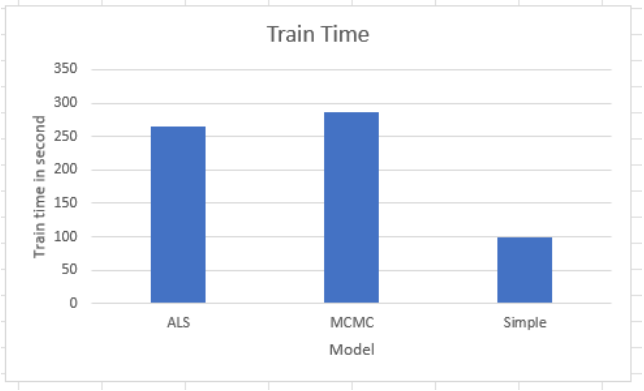

Figure 12. Train time for 3 models

In summary, the characteristics of the Steam data set enable both the simple model and the Factorization Machine model to provide good prediction accuracy when only using the user's purchase history. The simple model has the advantage of short training time, but the simple model cannot use more item features to provide personalized recommendations. Although the Factorization Machine model takes a long time to train, it can provide more accurate personalized recommendations by adding more item features.

4 Related Literature

The traditional item recommender system usually only focuses on the use of user and item interaction records to achieve collaborative filtering. These models often try different matrix factorization methods, or use neural networks to embed users and items as vectors. Then use machine learning to obtain the user's bias, the item's bias, and the weighted inner product of the user and the item vector to predict the relationship or rating between the user and the item. However, in actual recommender model applications, we often find that the relationship between users and items is often also affected by the explicit features of users or items. Several papers have tried to incorporate the explicit features of users or items into the recommender system model [5, 6, 8–10]. There is also a study on the music recommender model and found that there is a positive correlation between the play counts (item features) and the ratings (explicit feedback), so we can predict the ratings by introducing the play counts as the item feature.

The Steam data set in this paper comes from a study on Item Recommendation in Monotonic Behavior Chains [1]. This data set is used to analyze the prediction of different interactions between users and items in this study, that is, first predict whether the user will buy the game, then predict whether the user will play the game, then predict whether the user will evaluate the game, and finally whether the user recommends in the evaluation. Our experiment focuses on the research by adding the explicit features of the game, plus the user's game purchase history, to predict whether the user has purchased the game, so as to recommend the game that the user is most likely to buy. In that experiment, many similar data sets were also studied, such as clicks and purchases in the YooChoose data set, reviews and recommendations in the Yelp data set, and reviews and recommendations in the GoogleLocal data set[1]. They are also used to verify the accuracy of the recommendation system's interaction predictions of users and items.

Most of these studies have focused on how to use better models to make user and item embedding vectors to better represent the characteristics of users and items. The state-of-the-art methods currently employed to study this type of data are bprMF[11], WRMF[4, 7] and logMF[14], and these studies only use latent factors to represent users and items Features. In this paper, we directly use item features to independently generate latent factors. This paper and these studies have the same conclusion, that is, by using an appropriate model and incorporating the explicit characteristics of users or items related to explicit feedback, we can all obtain an accurate prediction model of the recommender system.

5 Results and Conclusions

We first check the results of the Simple popularity based model. Based on the results on our validation set, this model seems to work well on simple yes or no predictions, but it is hard to extend it for more personalized predictions since we were not able to incorporate any user related features. Also we believe this model may not fit very well on other datasets since we discovered that the best threshold we found for this model (1325) seems to be very close to the turning point (1725) (see Figure 13) of the value for (avg Top N games users bought / avg number of games users bought). Therefore it is highly possible that the high accuracy provided by this model is caused by the overfitting to the dataset.

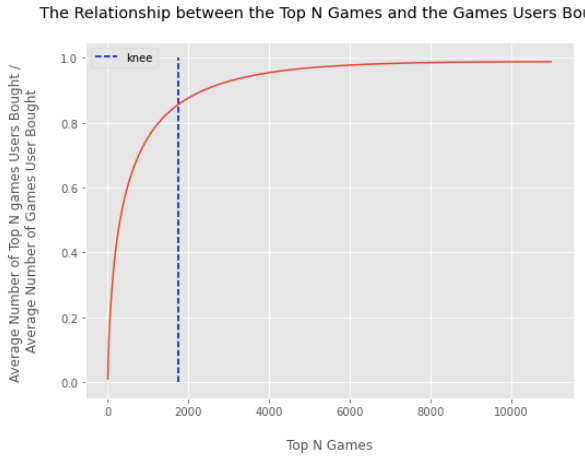

Figure 13. The Relationship between the Top N Games and the Games Users Bought

Then for the Factorization Machine model, we report the experimental results when n_iter=10, init_stdev=0.1, rank=i, l2_reg_w=0.1, l2_reg_V=0.9. We use accuracy and Area Under the ROC Curve (AUC) as our ranking metric. For the Factorization Machine with ALS solver and MCMC solver, we also report the accuracy and AUC when using different features. The basic feature is the user's historical purchase records, and then the game features such as publisher, tags, specs, and release year are added one by one by one.

We also compare SVD and SVD++[13] as baseline models.

We noticed that all models except simple baseline have similar prediction accuracy and AUC under the basic features (see Figure 14, 15), but when the game features are added one by one, the accuracy of the simple model decreases. This is largely due to the simple

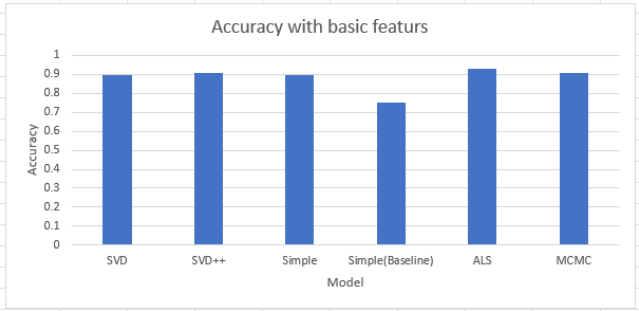

Figure 14. Accuracy for all models

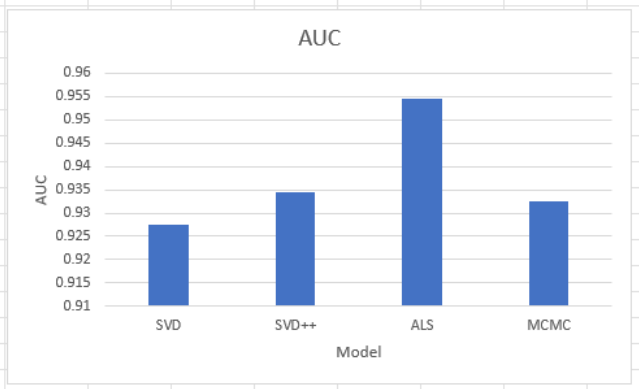

Figure 15. AUC for all models

model having complex features. and then the model is overfitting. And we also noticed that when the Factorization Machine with ALS solver adds game features one by one, it can capture the correlation between the features of the game and the user's preference towards the game, so we can see that the accuracy and AUC increase with the addition of game features. Then we can conclude that when using basic features, all models have similar predictive capabilities, but the Factorization Machine with ALS solver is more able to capture the user's preference for the features of the game, so it has the best predictive capabilities.

At the same time, we know that user purchase game history and game features work well in the Factorization Machine model, while the game features do not work in the simple model, and cause overfitting. The parameter adjustment of the Factorization Machine with ALS solver model is analyzed in Section 3, Rank = 16 and l2_reg_w=0.1, l2_reg_V=0.9 to ensure that the model can fit the training set well without underfitting or overfitting. This is also the advantage of the Factorization

Machine with ALS solver, that is, the ability to manually adjust the degree of model fit to achieve the best prediction effect.

To sum up, the Factorization Machine with ALS solver plus user purchase game history and game explicit features can produce the best model, because it can simultaneously capture user purchase game history and users' preferences of game explicit features, so as to better predict whether users bought games. However, the simple model and SVD and SVD++ cannot capture the user's preference for the explicit x features of the game, so the prediction results are relatively poor.

References

[1] Wang-Cheng Kang, Julian McAuley. *Self-attentive sequential recommendation.* In *ICDM*, 2018.

[2] Mengting Wan, Julian McAuley. *Item recommendation on monotonic behavior chains.*In *RecSys*, 2018.

[3] Apurva Pathak, Kshitiz Gupta, Julian McAuley. *Generating and personalizing bundle recommendations on Steam.* In *SIGIR*, 2018.

[4] Y. Hu, Y. Koren, and C. Volinsky. *Collaborative filtering for implicit feedback datasets.* In *ICDM*, 2008.

[5] G. Jawaheer, M. Szomszor, and P. Kostkova. *Comparison of implicit and explicit feedback from an online music recommendation service.* In *HetRec*, 2010.

[6] N. N. Liu, E. W. Xiang, M. Zhao, and Q. Yang. *Unifying explicit and implicit feedback for collaborative filtering.* In *CIKM*, 2010.

[7] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. *One-class collaborative filtering.* In *ICDM*, 2008.

[8] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang. *Transfer learning to predict missing ratings via heterogeneous user feedbacks.* In *IJCAI*, 2011.

[9] D. Parra and X. Amatriain. *Walk the talk: Analyzing the relation between implicit and explicit feedback for preference elicitation.* In *UMAP*, 2011.

[10] D. Parra, A. Karatzoglou, X. Amatriain, and I. Yavuz. *Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping.* In *CARS*, 2011.

[11] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. *Bpr: Bayesian personalized ranking from implicit feedback.* In *UAI*, 2009.

[12] Immanuel Bayer. *fastFM: A Library for Factorization Machines.* Journal of Machine Learning Research 17, pp. 1-5, 2016.

[13] Nicolas Hug. *Surprise: A Python library for recommender systems.* Journal of Open Source Software, 2020.

[14] C. C. Johnson. *Logistic matrix factorization for implicit feedback data. NIPS*, 2014.