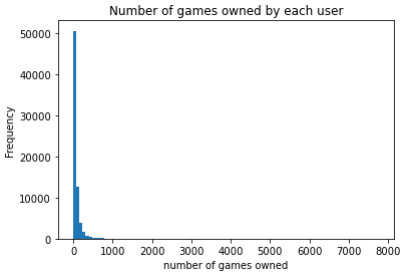
Video Game Recommendations Using Collaborative Filtering

Authors: Luke Lloyd and Alex Makhratchev

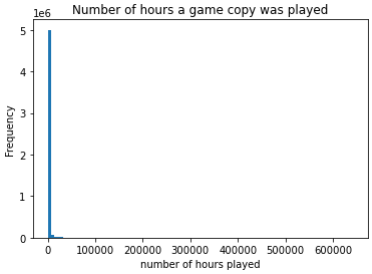
1 December 2021

1 Dataset

The datasets we will be using is the Steam Version 1: User and Item Data which can be found here. [1, 2, 3] The User/Item data is formatted in a list of json elements that include a unique identifier for each user and a list of games that the user owns. The list of games has the game's unique identifier along with the number of hours the user has played the game for. There are $68,403$ unique users that have played games and $10,050$ unique games that people have played. The total amount of game copies that have been played is $3,285,246$. The total amount of game copies owned is $5,153,209$. This means that there are $1,867,963$ game copies that have not been played. The mean number of games a user owns is 72.67. The median number of games a user owns is 40. This would indicate that the distribution of the number of games a users own is skewed positive. This makes sense as there is no upper bound on the amount of games you can own. The following histogram confirms this:



The mean hours played for a game copy is 991.50. The median hours played for a game copy is 34. This means the distribution of hours played per game copy is skewed positive. The following histogram confirms this:



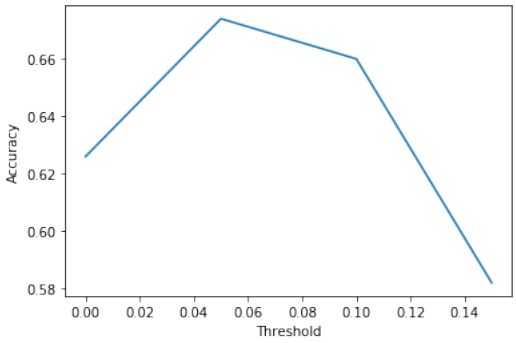
Out of the $5,153,209$ user item interactions, $1,970,693$ have less than 2 hours of play time. That is approximately 38.24% of the data.

2 Predictive Task

While performing EDA, we discovered that there are a large portion of games copies that have little to no play time. For our predictive task, we want to predict whether a user will play a game that they have purchased for more than 2 hours. We chose 2 hours here because on Steam if you have less than 2 hours of play time on a game, you can return it within 2 weeks of you buying it. Because we specifically want to study purchased games, we will construct our training, validation, and test sets by first formatting the data into tuples of *user_id, item_id, has_played*. We did this by looping through all users in the dataset, then for each user, looping through all games, filling in user_id, item_id, and set has_played = 0 if the total play time was 2 hours or less and has_played = 1 if the total play time was more than 2 hours. Then after shuffling these tuples we will split into training, validation, and test sets containing 70%, 20%, and 10% of the data respectively. To measure the success of our model, we will simply use the accuracy because the data is relatively balanced at approximately 60% positive and 40% negative. For this type of predictive task we think a game popularity threshold will be a good baseline. After trying multiple thresholds for the popularity value, 0.8 worked the best and was able to be 63% accurate on the validation set, which is 2% better than best guess. The actual models we want to implement are a Jaccard-similarty based model, and a latent factor model. Both of these models are meant for user-item pairs so we think that they will work decently well with the data.

3 Model

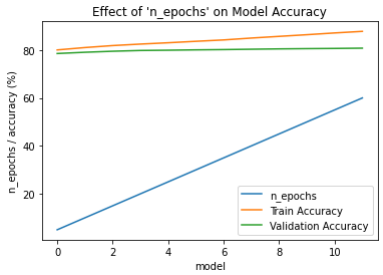
For the next iteration of the model, we used the Jaccard similarity to predict whether a user will play a game. This was done by creating two dictionaries using the training data, the first of which contained all the users per game, and the other was all the games per user. Then, for each data point in the validation set, we find all of the games the user has played and compute the Jaccard similarity between the game's users for each game and the the user in the data point. If the Jaccard similarity is higher than a given threshold for a given game, then the classifier predict that the specific user will play the game, however if none of the similarities are above the threshold, then the prediction for that user, game combination will be false. After experimenting with different threshold values for the Jaccard similarity, the performance of this model was surprisingly well on the validation data, with an almost 68% accuracy, which is 7% higher than the class balance.



The greatest drawback to this model is the run time. I could only get it to run on a couple thousand examples from the validation data set, because each example was computationally expensive for Jaccard similarity.

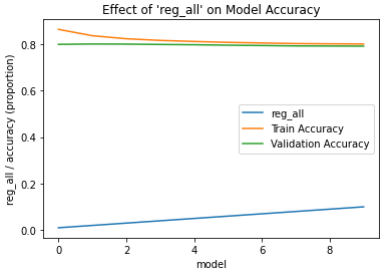
For our primary model, we used a latent factor model. We think this model is a good choice because the dataset does not include any features about the user or game so we are hoping that it will "learn" the users' preferences and the games' attributes. Initially, we intended to use a fastFM.als.FMClassification object from the FastFM library, however we ran into complications with the package that prevented us from doing so. Instead, we used a surprise.prediction_algorithms.matrix_factorization.SVD object from surprise. At first, we were concerned that this wouldn't work well though because this object doesn't do classification, it makes continuous predictions. However, after taking a look at the fastFM source code we found that fastFM simply converts continuous predictions into binary classification, so we felt this method was sufficient. In order to use this object, we first had to format the data correctly. This required first taking our shuffled train, validation, and test tuples and making them into pandas DataFrames. Then, using surprise Reader object, we turned the DataFrames into surprise DatasetAutoFolds objects which could then finally be turned into surprise Trainset objects using the build_full_trainset() method. With our data now in the right format, we could start tuning the model.

The first model parameter we chose to tune was n_epochs, which corresponds to the number of gradient descent steps the model takes. We didn't find any over-fitting as we increased the parameter, but we did observe diminishing returns.



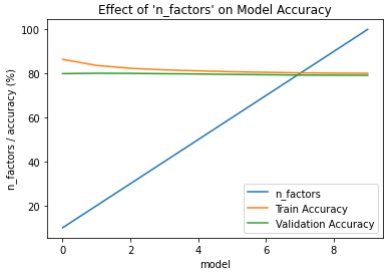
For our final model parameter, we chose n_epochs = 30 as we felt it was a good balance between accuracy, and train time.

The next model parameter we tuned was reg_all, which corresponds to the regularization term for all parameters.



For our final model we decided to use reg_all = 0.02 as that produced the highest validation accuracy.

The last and most important model parameter we tuned was n_factors, which corresponds to the size of the γ_u and γ_i vectors.



We observed a very minimal amount of over-fitting which is expected when increasing the number latent factors. For our final model parameter we used n_factors = 20 as that maximized ac-

curacy on the validation set.

4 Literature

The dataset we are using comes from a larger Steam database that has many more databases such as meta data, and user reviews of games and bundle[1][2][3]. In those studies their goal was to try different models on giving a personalized recommendation. Our goal was to simply recommend a user a game based on their other game purchases so we only required the user and game interaction data. This task has been thoroughly researched and been applied in major companies such as Amazon [5] and Netflix [4] to predict the user's preference towards a certain book or movie. The method we used is collaborative filtering, because we are not using any features from the game or the user. Originally, we we're planning on including content based filtering approaches into our model, by including the tags and genres for each game. However, during testing we realized that the amount of data to go through is too computationally expensive and would take us weeks to run a model. For one of the baseline models, we decided to try the Jaccard similarity which has proven fairly effective for a simple lightweight model [6]. Another collaborative filtering model is a Factorization Machine, which gained its popularity during the Netflix Challenge. We chose this to be our primary model, because it can easily scale with dataset size and works well with extremely sparse datasets [7]. After lots of testing with different hyper parameters, this proved to be the most promising model for the current task.

5 Results

Overall, our final model performed well. On the training set we achieved an accuracy of 83.63%; on the validation set an accuracy of 80.08%; and on the test set an accuracy of 80.03%. This is a very significant improvement over the popularity baseline, which had a validation accuracy of 63%, and our Jaccard-similarity based model, which had a validation accuracy of 68%. Because the data did not really include features other than user and item interaction, the number of possible models that would work started out small. Based off of our previous knowledge, we figured that a popularity based baseline would be decent because most people who play video games play popular games, or at least have tried them for more than 2 hours. If a game is popular then it is more likely that someone you know has played it, which is a factor that, more often than not, encourages you to play it. The Jaccard-similarity based model also aligns with our prior experience. Not only do people with similar preferences play similar games, but the community around certain games and genres reinforces this phenomena. However, there are a lot of things that the Jaccard-similarity model cannot capture. This is where a latent factor model is able to beat out a Jaccard-similarity based model. With interactive experiences like video games, there are some things that some people like spending their time on, and some things they don't. Even if all your friends are playing a certain game, if you know you're going to hate it, you probably won't play it. Because a latent factor model can capture the essence of a user's compatibility with

an item. It can do a much better job in predicting whether a user will play a game.

6 References

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

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

[3] Generating and personalizing bundle recommendations on Steam
Apurva Pathak, Kshitiz Gupta, Julian McAuley
SIGIR, 2017

[4] Netflix: www.netflixprize.com, h., http:// www.netflixprize.com. Accessed on 30th November 2021.

[5] E. Brynjolfsson, Y.J.H., M.D. Smith,, Consumer surplus in the digital economy: estimating he value of increased product variety at online booksellers. Manage. Sci, 2003. 49(11): p. 1580-1596.

[6] Ayub M, Ghazanfar MA, Maqsood M, Saleem A, editors. A Jaccard base similarity measure to improveperformance of CF based recommender systems. 2018 International Conference on Information Net-working (ICOIN), Chiang Mai, Thailand; 2018

[7] Germán Cheuque, José Guzmán, and Denis Parra. 2019. Recommender systems for Online video game platforms: The case of STEAM. ,763–771 pages.