Steam Game Recommender

Capstone Project, Machine Learning Engineer Nanodegree

Rahul Phatak

Definition

Project Overview

Steam is a gaming platform where multiple games on different platforms (including PC, Mac, and Linux) are sold to players. It also has a social component including streaming and community forums. Many video game providers also hold frequent discounts on the offered games. Users can also see recommendations based on the games they have played already.

Improving the recommendations shown to a user can increase user engagement and increase revenue of the game producer (and Steam too). Given the huge variety of games available across various platforms, targeting the correct user who is likely to buy the game, conduct micro-transactions, and/or recommend the game to his friends is a very lucrative problem for game producers. Since Steam receives a cut on games sold through Steam, Valve also has an interest in properly recommending games.

Datasets

- Steam Video Games dataset
  URL : https://www.kaggle.com/tamber/steam-video-games

  This dataset provided by Tamber contains user playtime data for different Steam games, ie the hours a particular user has played a particular game.

  To quote the content description from the Kaggle page - “This dataset is a list of user behaviors, with columns: user-id, game-title, behavior-name, value. The behaviors included are 'purchase' and 'play'. The value indicates the degree to which the behavior was performed - in the case of 'purchase' the value is always 1, and in the case of 'play' the value represents the number of hours the user has played the game.”
- Internet Game Database API

  Instead of using DBPedia as planned in the original capstone proposal, the Internet Game Database API was used since it contained detailed game information and had a good API as compared to DBPedia which would have heavy SPARQL queries.

Scoring Function

A scoring function will be defined to incorporate the wide range in hours played across different users and different games. This will be an attempt to reduce skew and normalise the given data.

$$score\ =\ personal\ ratio\ *\ normalised\ hours$$

Example

If user XYZ has played game A for 20 hours and he has totally played 100 hours on Steam:

$$personal\ ratio\ =\ 20/1000\ =\ 0.2$$

But what if game A is a singleplayer campaign which only lasts around 10 hours? Then user XYZ seems to like game A a lot compared to other players. It is assumed that the players playing game A are normally distributed. If game A is distributed with a mean of 10 hours and a standard deviation of 5 hours,

$$normalised\ hours\ =\ \frac{20-10}{5} = 2$$

$$score\ =\ 0.2\ *\ 2\ =\ 0.4$$

Problem Statement

Create a recommendation system which can predict user engagement with games (the scoring function which is defined below) given playtime data about the games and the user. Top-5 games will be recommended to the user along with the predicted score (defined above). This score can be used by companies to decide revenues. Eg. If a user plays a game (eg Candy Crush) very much as compared to other players, similar long-term games with micro-transactions can be suggested to him. On the other hand, if a user plays a lot of different games but doesn’t spend a lot of time on any of them, another variety of games with one-time prices can be shown to him.

Metrics

Instead of using “Precision at N” as used by Kevin Wong, an RMSE(Root Mean Squared Error) score will be used since the predicted score is more important in this case than just predicting whether the game is played or not. Eg. Suggesting “Candy Crush” to a player who plays such micro-transaction games for a long time (and pays for the in game boosts) will generate more revenue than suggesting a game which has a small one time price.

Solution

Collaborative filtering will be used to fill in the missing user-game score matrix. The top 5 games (arranged in order of descending scores) will then be chosen and shown to the user as recommendations. Matrix factorization will be used to characterise the latent features between users and games. Since game data is also available (from IGDB) factorisation machine models will also be used to improve performance.

Since explicit data is available (in the form of the defined scores), a factorization_recommender will be used as suggested here. Note : item_similarity_recommender will not be used since it can not incorporate user and game specific information (as noted in the relevant documentation Notes section). Parameter tuning will then be done to improve performance.

Analysis

Data Exploration

The dataset provided by Tamber had 200,000 interactions out of which only 70,489 interactions contained playtime data. Out of these 70,489 interactions, there were duplicate entries for 12 player-games. The duplicate entries were removed and only the last occurrences were kept.

Unique playtime interactions	70,477
Duplicate playtime interactions	12
Purchase interactions	129,511
Total	200,000

Player Distribution
There are 11350 unique players in this dataset. A majority of players (58%) played only one game. 95% of the players had played less than 25 games. There were many (86 to be exact) players who had played more than 100 games.

```
Number of players with <=   1 games:  6559 ( 58%)
Number of players with <=   5 games:  9221 ( 81%)
Number of players with <=  10 games:  9978 ( 88%)
Number of players with <=  25 games: 10753 ( 95%)
Number of players with <=  50 games: 11069 ( 98%)
Number of players with <= 100 games: 11267 ( 99%)
Number of players with <= 250 games: 11343 (100%)
Number of players with <= 500 games: 11350 (100%)
```

Game distribution

There are 3600 unique games in this dataset. Similar to the player distribution, there was a lot skew in this case too. Only 4 games had more than 1000 players with DOTA 2 leading the pack with 4841 players. DOTA 2 in fact had more players than the 3 next most popular games combined. 1018 games had only 1 player.

```
Number of games with >=    2 players:  2582 ( 72%)
Number of games with >=    5 players:  1584 ( 44%)
Number of games with >=   10 players:  1053 ( 29%)
Number of games with >=   50 players:   292 (  8%)
Number of games with >=  100 players:   133 (  4%)
Number of games with >=  200 players:    41 (  1%)
Number of games with >=  500 players:    10 (  0%)
Number of games with >= 1000 players:     4 (  0%)
```

Game	Players
Dota 2	4841
Team Fortress 2	2323
Counter-Strike Global Offensive	1377
Unturned	1069
Left 4 Dead 2	801

Exploratory Visualization

NA

Algorithms and Techniques

Collaborative Filtering

Suppose player A has high scores for ‘Sid Meier's Civilization V’, ‘Empire Total War’. Player B has high scores for these 2 games AND ‘Europa Universalis IV’ but he has low scores for ‘Grand Theft Auto V’. In that case, we can assume that player A and player B are players with similar tastes (ie. strategy games) and so we can suggest the new game ‘Europa Universalis IV’ to player A. We’ll also avoid recommending ‘Grand Theft Auto V’ to him.

This is called collaborative filtering where the system finds users with similar tastes as the target user and then recommends items which those users have used (but the target user hasn't)[1]

It is likely that a player who plays strategy games like Sid Meier's Civilization V will also be interested in playing Empire Total War. Similarly, a player who plays multiplayer first-person shooters like Counter-Strike Global Offensive a lot may also want to play Call of Duty Modern Warfare 2. So the playtime data available is supplemented by adding genre information from IGDB[2].

Benchmark

There are 40 million player-game interactions possible in this dataset.[3] But only 70,000 interactions are present in the entire dataset (even before splitting into train-CV-test). Due to this sparsity (0.17% filled entries), it is likely that a simpler model may perform better than an advanced model. As a benchmark, a simple popularity recommender will be used which will just use the average of the scores as the output. It is not tailored to individuals and won't use the genre data.

Methodology

Data Preprocessing

Removing duplicate rows

As pointed out in the "Data Exploration" section, the 12 duplicate player-game entries were removed from the dataset.

Adding genre information

Genre and developer information for the 3600 games was obtained from the IGDB website. There were 20 distinct genres and 1858 distinct developers. Due to the sparsity of the data, the developer information was discarded and only genre information was used. One-hot encoding was performed on the genres.

Removing games

Information for only 3581 games was obtained from IGDB. The interactions relating to the remaining 19 games was removed from the dataset. The 19 games were 'Dethroned!',

[1] https://en.wikipedia.org/wiki/Collaborative_filtering#Introduction
[2] https://www.igdb.com
[3] 3,500 games * 11,350 players = 40,860,000 combinations

'Voxelized', 'Aberoth', 'Motorbike', 'Deepworld', 'Immune', 'Breezeblox', 'Cosmophony', 'CRYENGINE', 'Unium', 'Rexaura', 'Squirreltopia', 'KWAAN', 'Burstfire', 'Batla', 'ROCKETSROCKETSROCKETS', 'Rotieer', 'Alganon' and 'Jumpdrive'.

Creating scoring function

Since a score was the target variable, it has to be created for the whole dataset together (since we needed scores for the test interactions too). But to avoid this test data from leaking into the training data, the scores for the training data were also calculated.
The scores for the training data created from training data ONLY were used to TRAIN the models.
The scores for the test data created from BOTH training and test data combined were used to EVALUATE the models.

Implementation

Creating benchmark model

A simple popularity recommender was created using the training data. It had an RMSE of 0.479 on the cross validation data.

Creating factorisation recommender

A basic factorisation recommender with the default parameters was created using the training data. It had an RMSE of 0.672 on the cross validation data.

Refinement

A grid search was performed to find the best set of hyperparameters for the factorisation recommender.

Hyperparameter	Values tried
linear_regularization	1e-05,1e-07,1e-09
num_factors	8,16,32
regularization	1e-06,1e-07,1e-08

The best values found were:

llinear_regularization : 1e-05
num_factors : 16
regularization : 1e-06

This improved model had an RMSE of 0.614 on the CV dataset.

Results

Model Evaluation and Validation

	Popularity Recommender	Factorisation Recommender	Tuned Factorisation Recommender
CV RMSE	0.479	0.692	0.615
Test RMSE	0.257	0.459	0.478

It is observed that the basic popularity recommender has the minimum RMSE for both the CV set and the test dataset.

Justification

Due to the sparsity of the data (only 0.17% of the possible entries), a simple model had better performance on this small dataset as compared to a more complicated model.

Conclusion

Free Form Visualisation

N/A

Reflection

In this project, it turns out that adding the genre implementation wasn’t necessary since it wasn’t used in the final popularity recommender. Defining the scoring function (and implementing it correctly) was the main issue due to the various joins and merges of the dataset. The 12 duplicate rows were one of the biggest roadblocks in this project. I only realised this issue when I was implementing the scoring function and the rows weren’t matching (as detailed in the appendix of the Jupyter notebook)

Improvements

- Define the scoring function to be a weighted product of both the scores. This could be used by companies to decide what factor they want to focus more on. For example, people who have a high normalised score ie. they are heavy players could be targeted with micro-transaction type games.
- More data! This small dataset had limited interactions. Steam could create a better recommendation engine by using their own data. With enough data, a factorisation recommender could also