Steam Game Recommendations

Christopher Gomes
Steven Horn

Intro

- Goal: video game recommendation system based off of past games that users have played
- Dataset:*
  - User id
  - Game name
  - Behavior name (purchase/play)
  - Amount
- Output: List of 10 recommended games per user

*https://www.kaggle.com/tamber/steam-video-games/data

Data Cleaning

- Removed purchase behaviors
- Kept games played between 10 and 500 people
- Kept users that played at least 10 games
- Min-max normalized play times by user
- Game names were encoded to integer ids

Analysis

- Sparsity of the data is 97%
- Used leave-one-out strategy to compare recommendations to a game the user actually played
- Found percent of games in the catalog that were recommended
- For context, if we randomly chose 10 games to recommend for each user, the accuracy would be 0.96%

Implicit Models

We tested the accuracy of a variety of different models to learn about which models are most effective. We also tested these models using both normalized and non-normalized values. We used the Implicit library's implementations.

- Alternate Least Squares
- Bayesian Personalized Ranking
- Logistic Matrix Factorization

Implicit: https://implicit.readthedocs.io/en/latest/index.html

Alternate Least Squares

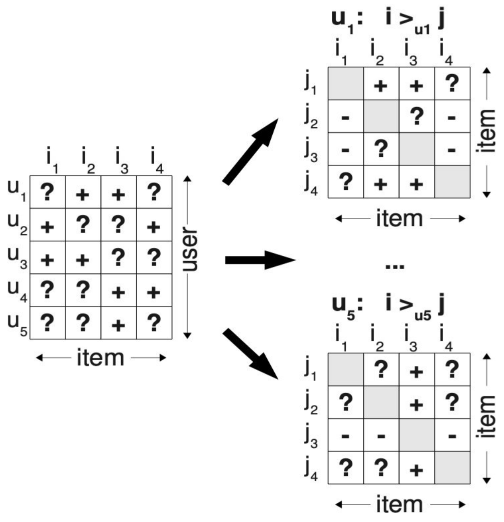
High Level Description:

- Based on paper: http://yifanhu.net/PUB/cf.pdf
- Factor out a user/item matrix R into user factor X and item factor Y, multiply to estimate r_{ui}
- Cost function is non-convex - need to make it into a quadratic function and minimize
- Alternate fixing y_i and computing x_u, and fixing x_u and computing y_i
- Recommend to user u the K items with the largest values of the estimated $r_{ui} = x^T_u * y_i$

Results:

- On average, 17.57% of users had their test game recommended to them by the model
- On average, the model recommended 73.91% of the games that are available in the data set

Bayesian Personalized Ranking



High Level Description:

- Based on paper: https://arxiv.org/pdf/1205.2618.pdf
- Creates triples (u, i, j) s.t. user u prefers item i to item j
- Use a Bayesian formulation to optimize parameters of another model
    - $p(\Theta | >_u) \propto p(>_u | \Theta)\, p(\Theta)$, where Θ = parameter vector of arbitrary model class (e.g. MF)
- Gives recommendations based on $x_{uij} := x_{ui} - x_{uj}$, which are predicted by the other model

Results:

- On average, 19.92% of users had their test game recommended to them by the model
- On average, the model recommended 97.31% of the games that are available in the data set

Logistic Matrix Factorization

High Level Description:

- Based on paper: https://web.stanford.edu/~rezab/nips2014workshop/submits/logmat.pdf
- Problem setup is similar to ALS
- Calculate $p(l_{ui} | x_u, y_i, \beta_i, \beta_j)$, where l_{ui} is the event user u likes item i, and β are biases
- Do this by tuning X, Y, β by maximizing log p(X, Y, β | R)

Results:

- On average, 5.69% of users had their test game recommended to them by the model
- On average, the model recommended 0.96% of the games that are available in the data set

Neural Collaborative Filtering (NCF)

- Combine feed-forward neural network with traditional collaborative filtering algorithms
- Based on paper: https://arxiv.org/abs/1708.05031
- Percent of users were their played game was in their recommendations: 15.69%
- Percent of games in catalog recommended: 20.81%

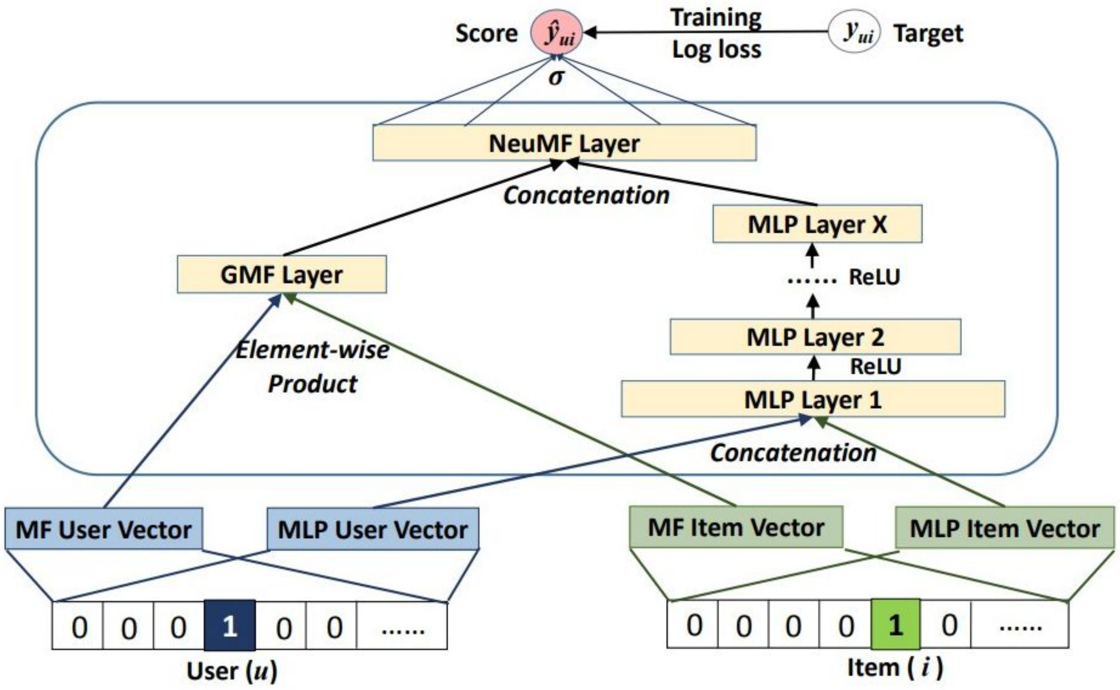


Figure 3 in https://arxiv.org/abs/1708.05031

Possible Next Steps

- More data cleaning
  - Clean up some game names
  - Fixing Thresholds for users and games to include
  - Up minimum time to be considered a play
- Tune NCF model