Recommender System for Bounce Interactive v0.1

Introduction

Bounce Interactive offers approximately 40 different online games to around 8000 regular users. Typical users play a small subset from the catalogue of games and it is believed recommendations for alternate games could result in increased user activity. This document describes a prototype collaborative filtering; matrix factorisation recommender system for the application and summarises three sets of test results as evidence of concept.

Method

The choice of method, namely Singular Value Decomposition (SVD), is inspired by its success in the Netflix Prize for movie recommender systems. Collaborative filtering more generally is used in recommendations for music, books, friends and interests and is considered scalable, accurate and flexible. Both explicit user feedback (surveys, 'likes', etc) and implicit feedback (purchase history, 'dislike by omission') can be incorporated, although its foremost use cases emphasise the former.

While it is reasonable to assume explicit feedback is a reliable linear scaling of user preference there are several reasons for caution in relying solely on implicit feedback: (1) possible non-linear relationships between user preference and the feedback measure(s) (2) conditional independencies between feedback measures (3) the inherent assumption of collaborative filtering is that people who agreed in the past will also agree in the future, but that assumption does not carry through to agreement between users' behaviour or external circumstances. These issues may be partly mitigated through feature engineering, for example to represent a statistic following a power law as log-linear, but clearly the measure of user preference or 'Rating' must be properly defined.

SVD is a 'latent factor method'. By contrast to a 'neighbourhood method', where users are connected to items by observable traits of the game/movie/song, such as whether it features Leprechauns/was directed by Terry Gilliam/was sung by Cher, 'latent factors' are only inferred from the data. It is instructive to imagine the latent factors representing characteristics of playing the game like excitement or escapism, but they also effectively encode user preference for unforeseeable/idiosyncratic qualities of the game experience. As a matrix factorisation method it is implied that 'Rating' as input to SVD is a linear combination of these latent factors.

The assumption that players who agreed about a game in the past will agree about other games in the future is expressed through players' consistent preference for a certain mix of the latent factors (i.e. player A and player B agree about game X because it has an abundance of latent factor Q; players A and B both have an affinity for Q). It is also given that each game can be characterised by a certain profile over the latent factors (i.e. the abundance of latent factor Q is a permanent characteristic of game X) although it is not necessary to

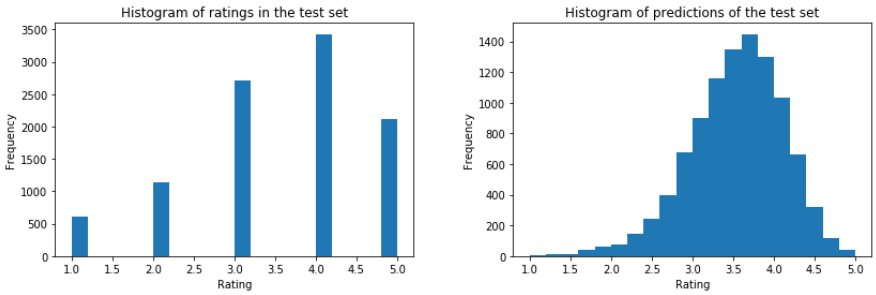

Figure 1: Real (movie) data: (Left) 'Ratings'. (Right) 'Predictions'.

assign true meaning or category to any of the latent factors. Training the SVD algorithm is a matter of profiling each user and game over the latent factors using past ratings as a target variable and where the more similar a user profile is to a game profile, the higher the 'Rating'.

A weakness of this method is the 'cold start' problem: new games and users have no data and so cannot be profiled. It is proposed to work around by profiling new users and games based on observable traits and seeding new items with ratings from similar games/users.

Input Data

List each available 4-tuple: 'User ID', 'Game ID', 'Rating', 'Time Stamp' in a CSV or text file.

Output Data

TBC (For example, 'All Unknown Ratings'/'top 5 games for each user') to be output to CSV or text file.

Results

Real Data

The MovieLens 100k dataset of 100000 movie ratings. It is used to demonstrate the algorithm at moderate scale with real data, simulating in particular the likely sparsity of ratings in a real system; most people play only a small subset of games/watch a small subset of films, and explicitly rate even fewer - the movie ratings are exclusively explicit feedback and range from 1 to 5. RMSE is <0.9, meaning the algorithm is typically accurate to less than 1 whole rating grade. Figure 1 illustrates correspondence between 'Ratings' and 'Predictions'.

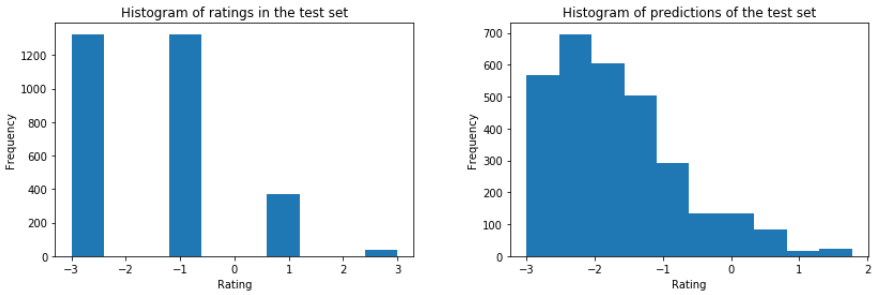

Figure 2: [I] Synthetic (deterministic) data: (Left) 'Ratings'. (Right) 'Predictions'.

Synthetic Data

Data are synthesised for 500 imaginery users and 8 imaginery games. Users are defined over 3 binary traits: Age (Young/Old); Sex (Male/Female); Frequent Player (Y/N). Games are also defined over 3 binary traits: Volatility (High/Low); High Stakes (Y/N); Travel Themed (Y/N). Ratings are synthesised by assuming exclusively young users like volatile games, exclusively male users like high stakes games and exclusively frequent users like travel themed games. Then 15% of ratings are deleted at random to simulate some arbitrary sparsity.

[I] User traits increment/decrement 'Rating' deterministically (+1/-1) to range $[-3, 3]$. RMSE is 0.9, meaning the algorithm is typically accurate to within 1 whole rating grade. Figure 2 illustrates correspondence between 'Ratings' and the 'Predictions'.

[II] User traits increment/decrement 'Rating' stochastically ($\mathcal{N}(+1, 0.3)/\mathcal{N}(-1, 0.3)$). RMSE is 1.0, meaning the algorithm is typically accurate to around 1 whole rating grade. Figure 3 illustrates correspondence between 'Ratings' and the 'Predictions'.

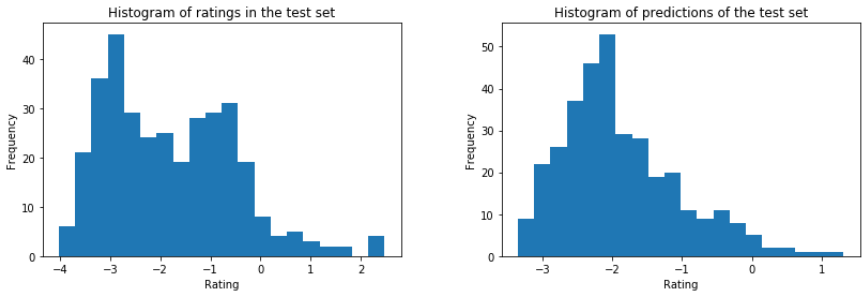

Figure 3: [II] Synthetic (stochastic) data: (Left) 'Ratings'. (Right) 'Predictions'.

Conclusion

A baseline implementation of an SVD Recommender system has been implemented and test results are given for three scenarios. There is overall good evidence of concept but with a tendency for bias in the cases of synthetic data, and with predictably noisy resolution in the stochastic case. The synthetic data sets are small but dense. The 'Rating' data in the three scenarios is either explicit or *highly* simplified implicit feedback.

Next Steps

Engineer the input data. If the specification dictated by the above methodology is not attainable then alternatives can be considered.