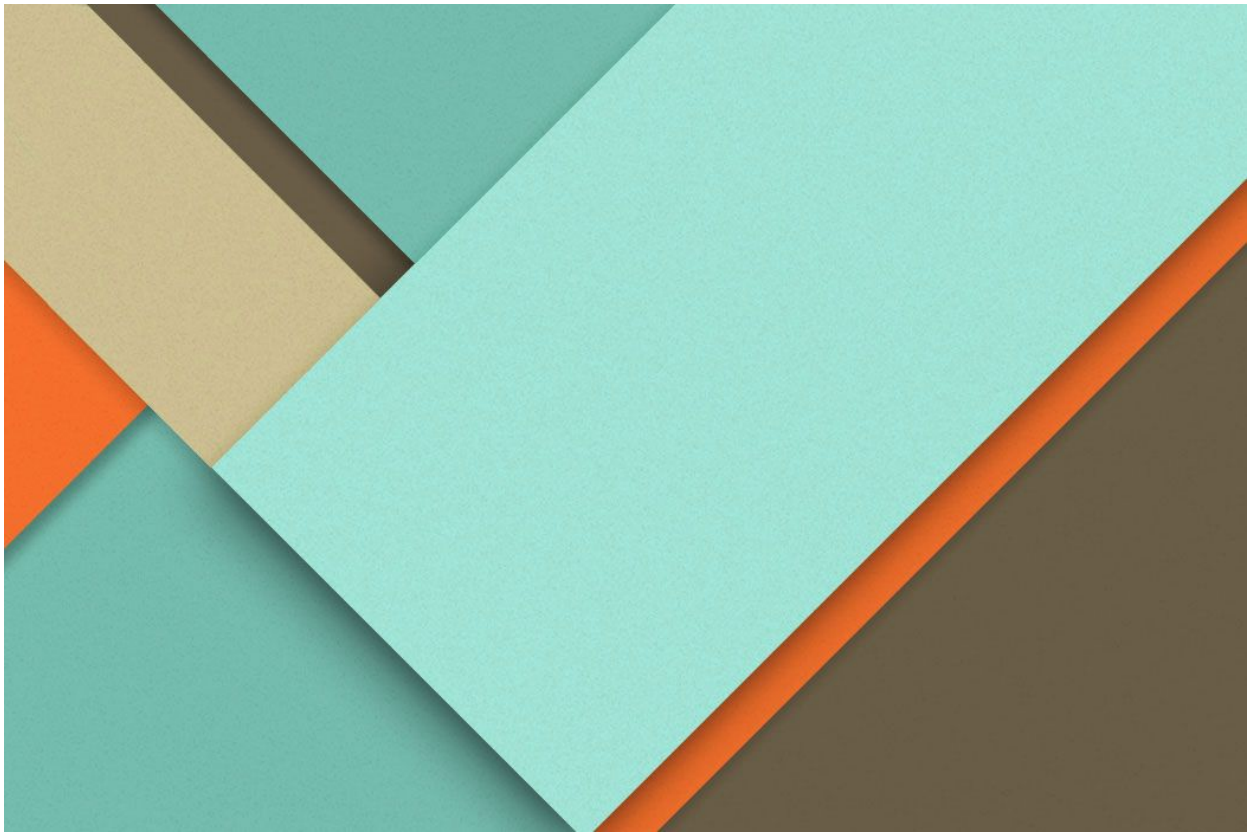

IT- 562

Recommendation Engine and Applications

Music Recommendation Engine

—

Tune In

Roshan Shah

Shrey Shah

Kajal Gosaliya

Rutul Patel

Overview

We have built a Music Recommendation engine which can be said to be a fusion of Reinforcement Learning (RL) and Collaborative Filtering (CF).

We were inspired to blend both the approaches since present major entertainment recommender engines like Netflix and Spotify use CF to a great extent as in reality CF is really practical (Much of the population can be grouped effectively). RL provides the recommender engine a personal feel which every user desires from a recommender engine.

The model also tries to the target the Exploitation vs Exploration dilemma.

Initially the dataset is empty and is built by the RL section incrementally. The only data we have initially is the song and its genre. User preferences are filled in and weighed incrementally. Initially only the RL part is actively recommending, while once sufficient data is gathered by it, the CF part comes into play, which uses data accumulated so far. So, our recommender engine doesn't suffer from cold-start, but rather combines the best of approaches so the user feels comfortable at the early stage and gets best of the results when CF kicks in.

Decision Parameters

1. Song Ratings
2. Genre

We have considered Song Ratings by the users and Genres as decision parameters for our recommendation engine.

We enquire about ratings of the songs recommended by the engine to get a feedback and change behaviour accordingly. The ratings also help to model Collaborative filtering by finding similar items to user's preferences.

What are we trying to do ?

We broke down the problem in three parts:

1. Reinforcement Learning
2. Collaborative Filtering
3. Exploration

If we suggest certain amount of results to the user,
then the probability that the result will be from :

RL is α ,

CF is β and

Exploration is ϵ

where $0 \leq \alpha, \beta, \epsilon \leq 1$ and $\alpha + \beta + \epsilon = 1$

Exploration Vs Exploitation

Initially the recommendations would be done by Exploration and RL.
CF will only come to play after around 30 iterations of recommendations.

Since, initially, a user might not have yet frozen into his comfortable genre, we provide a high exploration. And it decreases gradually. Continuously rating it high for certain iterations will lead to exhaustion of his favourite genre and again there is a need for high exploration

Exploration is modelled , where ϵ starts at a high of 0.5 and gradually decreases over multiple iterations to 0 .1 and increase again to slightly less than 0.5 and it keeps on going so on to stabilize at some value at infinity.

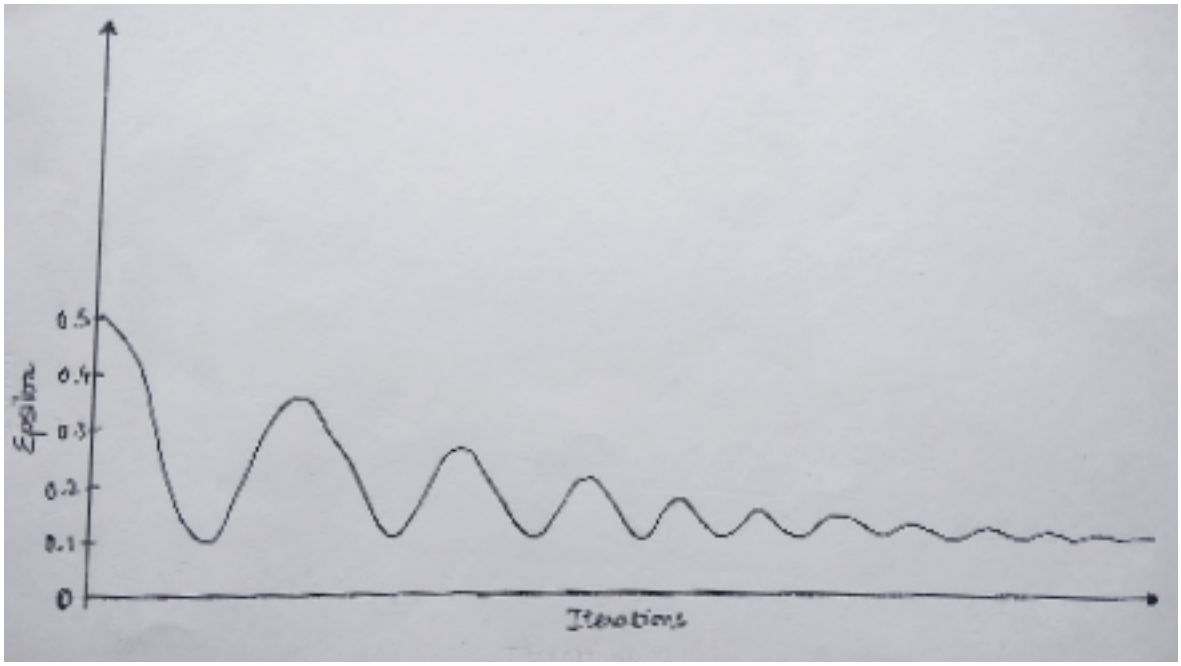

Figure : Epsilon as a function of iterations

Reinforcement Learning based on Multi-arm Bandit

Initially when a new user enters our system, we select 10 random songs and suggest it to the user. Based on the ratings given by user , we calculate a score for the genre associated with the song. Genre score is calculated for each genre for each user. Next time when a song has to be selected ,it selects a random song (which hasn't yet been suggested to the user) among top n genres (arms of the bandit), and suggest it to the user.
The rating of the song is also updated in the user-song matrix to be used later in CF.

Each user 'u' will have a genre-score for a genre to express his liking towards it, calculated as:

Genre-Score(genre 'g' , user 'u') :

$$\frac{\sum_{i \in genre} r_{ui}}{\sum_{i \in genre} (\text{i for which user has rated})}$$

where r_{ui} is the rating by user u to song i

Genre-Score is calculated for each genre for each user.

After the genre-score has been calculated, new recommendation from RL is made as :

Select $\lfloor(\alpha$ * total size of recommended list)$\rfloor$ random songs from top **n** genres
Where top **n** genres are defined as Genres who feature in first n genres when

genres are sorted through genre-score in decreasing order.

A low rating for a song will lead to a decreased genre-score for that genre for the user. So In the next round when new recommendation has to be made, the probability of song to be chosen from that genre decreases, since it's probability of occurring in the top 'n' genre decreases.

Similarly, the probability of a song of a particular genre being recommended increases if song of the same genre was rated high previously by other user.

This course of action progresses, where previously unrated genres come into picture via exploration, which has been modelled already.

Collaborative Filtering

We have implemented item-item collaborative filtering based on Pearson's correlation coefficient.
We find items that have been liked by user and try to find similar items based on ratings.

Similarity by Pearson's correlation for items i and j is defined as :

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R_u})^2}}.$$

The rating for target item i for active user a can be predicted by using a simple weighted average as:

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

where K is the neighborhood of most similar items rated by active user a, and w(i,j) is the similarity between items i and j.

There are many similarity coefficients like Jaccard, Cosine, Pearson etc, We have sticked to Pearson's correlation.

Reason for using Pearson's correlation to calculate similarity is that it adjusts for the user bias. Some user tend to give 3 to a song even if they like it, while some users tend to give it a 5. By subtracting mean rating of the user, it adjusts for the user bias. Then the positive ratings mean, he likes the song, while a negative rating means he dislikes the song.

In theory, Item-Item CF and User-User CF are dual approaches and thus, have similar performance. But in practice, Item -Item CF clearly outperforms User-User CF.

Reason for using Item-Item Collaborative Filtering as compared to User-User Collaborative Filtering is that Items are inherently simpler than users. There is less variation among items as compared to users. For Example, a user can like both death metal and soothing music, while an item is less likely to belong to both. Notion of Item similarity is thus more meaningful than User similarity.

A Quick walkthrough of the CF algorithm:

Let the matrix below be rating matring, where user rates a song in the range (1-5). The cells highlighted in yellow are the respective ids of songs and users.

Users

Songs	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

We want to predict, what would be the rating given by User-5 to Song-1 .
So we calculate Pearson Coefficient, for each song , with respect to every other song. We have listed the similarity value in the column beside the rating matrix.

Users

Songs	1	2	3	4	5	6	7	8	9	10	11	12	Sim(1,m)
1	1		3		?	5			5		4		1
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

For the purpose of this example we have taken the 2 nearest neighbours to predict the rating. For the purpose of our demo-code we took the 5 nearest neighbours. Nearest neighbours in this case would be the songs with the maximum similarity value, which the user has rated. I.e. Song-3 and Song-6 in this case. The rating is calculated as the ratio of (sum of (similarity x rating)) to the sum of (absolute values of similarity)

Users

Songs	1	2	3	4	5	6	7	8	9	10	11	12	Sim(1,m)
1	1		3		**2.59**	5			5		4		1
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

$$\text{Prediction of Rating} = \frac{2 * (0.41) + 3 * (0.59)}{0.41+0.59} = 2.59$$

Thus , a rating of 2.59 is suggested for song-1 for the user-5.

Datasets

We used 2 datasets for our implementation - one to build incrementally and the other to momentarily test our CF code.

I. Million Song Dataset

The Million Song Dataset is a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

A subset of Million Song Dataset was used which contains fields such as:

Track id
Title
Artist id
Artist name
Genre
Listen Count
Loudness
Energy
Tempo
Album
Release Year
Duration

We extracted Track id, Title and Genre name of approx 25k songs from the dataset. We allotted genre id to each of the genres manually.

II. Movielens-100k

Since, we incrementally built our dataset by inputs from the RL section, to utilise it further it in both RL and CF parts.

Therefore, To test our collaborative filtering section, we picked up a different dataset to test accuracy results .

The dataset used was: movielens-100k

It consisted of:
100,000 ratings (1-5) from 943 users on 1682 movies.
Each user has rated at least 20 movies.

Accuracy

We couldn't test the accuracy of our RL part, since if we tried to test how many items we liked from the suggested list, it would be subject to bias.

Also since RL isn;t a supervised learning technique there is no precise way to get accuracy.

But for our, CF part we could measure accuracy as

Surprise library in Python has implemented the exact same algorithm. To test accuracy our algorithm, we used surprise lib.

By running the CF code on a sparse movie-lens dataset ,
we get mean RMSE of 0.9912 with Standard Deviation of 0.0016.

	Fold 1	Fold 2	Mean	Std
RMSE (testset)	0.9896	0.9928	0.9912	0.0016
Fit time	2.77	2.72	2.75	0.02
Test time	11.72	11.93	11.83	0.10