

IT 562 - Recommendation Engine and Applications

Tune In

Roshan Shah
Shrey Shah
Rutul Patel
Kajal Gosaliya



Introduction

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

We were inspired to mix both the approaches since present major entertainment recommender engines like Netflix and Spotify use CF to a great extent. RL provides the recommender engine a personal feel.

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

Initially the dataset is empty and is built by the RL section 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.



Decision Parameters

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 we are 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 20 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.

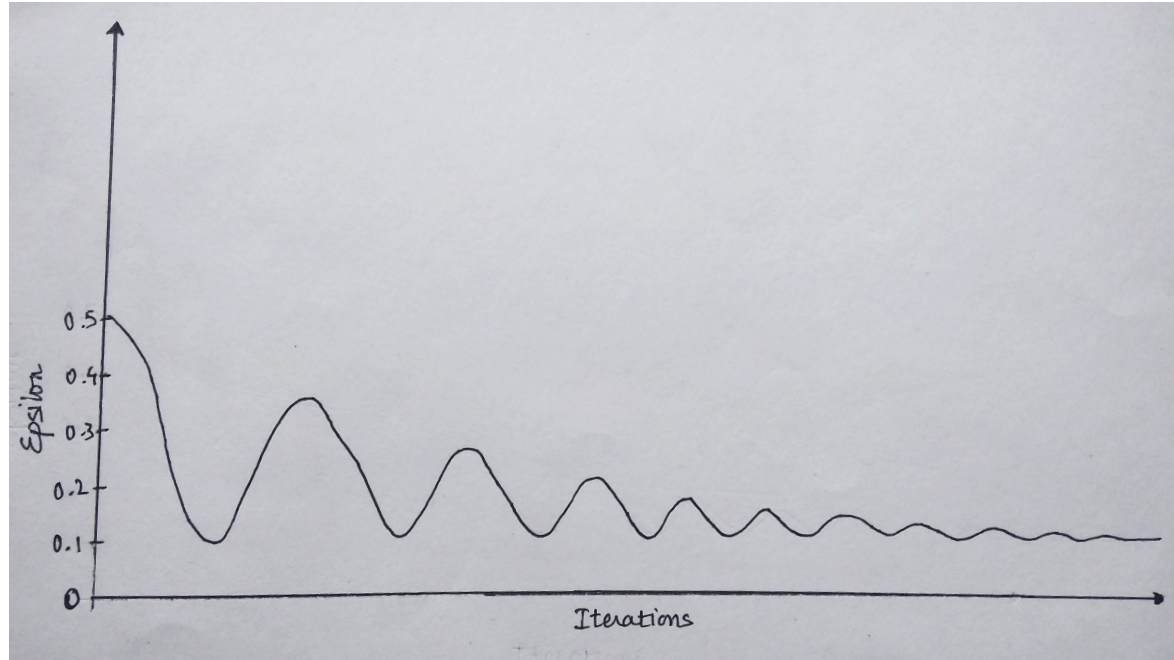


Fig : Epsilon vs Iterations



RL based on Multi-arm Bandit

Initially we select 10 random songs and suggest it to the user. Based on the ratings given by user , we calculate a rating for genre associated with the song. 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

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

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



RL based on Multi-arm Bandit (Continued...)

A low rating for a song will lead to a decreased genre-score for that user. So In the next round when new recommendation has to be made, the probability of song to be chosen from that 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.



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 :

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



Collaborative Filtering (Continued...)

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 .

A Quick walkthrough of the algorithm




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	

A Quick walkthrough of the algorithm



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	


A Quick walkthrough of the algorithm

		Users												Songs	Sim(1,m)
		1	2	3	4	5	6	7	8	9	10	11	12		
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

A Quick walkthrough of the algorithm (for 2 nearest neighbours)

		Users												Sim(1,m)
Songs		1	2	3	4	5	6	7	8	9	10	11	12	
	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

A Quick walkthrough of the algorithm (for 2 nearest neighbours)



		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

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

A Quick walkthrough of the algorithm (for 2 nearest neighbours)

		Users												Sim(1,m)
Songs		1	2	3	4	5	6	7	8	9	10	11	12	
	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

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



Datasets used

We have taken an initial dataset of songs from the [Million Song Dataset](#).

Extracted data of approx 25k songs from the dataset.

Since our recommender engine is a blend of Reinforcement Learning and Collaborative Filtering . We incrementally built our dataset by inputs from the RL section, to utilise it further it in both RL and CF parts.

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

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

“Thanks !”

