COMP47580 Collaborative and Content-Based Filtering Algorithms

Ravikanth Gollapalli

18361993

Abstract. A comparative analysis of 3 collaborative filtering algorithms: Embarrassingly Shallow Autoencoders (EASE), Singular Value Decomposition (SVD) & Bayesian Personalized Ranking (BPR) and an implementation of an alternative to collaborative filtering, content-based filtering.

1 Introduction

With the rise of social media platforms like Twitter, Instagram & TikTok, and the rapid development of technologies such as Netflix, Amazon and Google, recommender systems have evolved to adapt to the needs to the customers on these platforms. As the market becomes more competitive, companies make use of recommender systems to keep their users on their platforms. This is done by giving them recommendations for content they might like to consume, items they may wish to purchase and people they might know, depending on the platform that makes use of it. For example, Amazon provides recommendations in their homepage by showing you items you may be interested in purchasing based on previously searched items. But why is this required?

The amount of information available on the internet is growing day by day, making it very difficult and inefficient for consumers to sift through the data to find what they want. Recommender systems require data or inputs to process and make recommendations. This could be in the form of meta data about a product, product reviews, knowing the audience that a product targets, items rated by a user and the number of times an item has been interacted with by a given user. There are multiple recommendation problems such as being able to predict a rating for a product or user, predicting the products that a user would most want to purchase, ranking an array of products given a particular user, and providing quality recommendations given a group of people. These problems have been somewhat solved through collaborative algorithms, content based algorithms, and personalised/non-personalised algorithms. These algorithms must be outputted in a user friendly manner and evaluated based on different measures, some of which we will look at in this report.

For the purpose of this research report, I will be focussing on 2 parts; Algorithm Analysis and Advanced Analysis. The algorithm analysis section will evaluate some of the collaborative filtering algorithms implemented in this module. I will be comparing 3 algorithms, Embarrassingly Shallow Autoencoders

(EASE), SingularValueDecomposition (SVD) and Bayesian Personalized Ranking (BPR).

As part of my advanced analysis, I will be implementing a content based filtering algorithm. I will be creating a recommender system that uses genome metrics to offer recommendations. Studies argue that content based filtering is a more ethical way to propose recommendations due to the lack of user data being used and also a more efficient way for certain systems.

Collaborative filtering algorithms aim to provide recommendations for users based the choices of other users whereas content-based recommenders use the meta data of the items and bases their recommendations on that. This means the items that are given as a recommendation were similar items that were chosen before.

2 Problem Definition

The focus of the beginning part of the analysis is to identify how different collaborative filtering algorithms compare when evaluating against each other. The three collaborative filtering algorithms I will be comparing are Embarrassingly Shallow Autoencoders (EASE), Singular Value Decomposition (SVD) and Bayesian Personalized Ranking (BPR) to see which one of these performs best as a collaborative filtering algorithm. This task will be achieved by carefully tuning the algorithms and compare them using Recall as a Top N recommendation measure of success. It is worth noting that EASE and SVD are 2 linear models, while BPR is not. Understanding whether or not linear models are better is another comparison I will be able to make post evaluation of the collaborative filtering algorithms.

3 Methodology and Algorithms

In order to compare the different collaborative filtering algorithms, I first need to get an understanding of the algorithms and the parameters they contain that need to be tuned.

EASE: To begin, I will look at EASE, or Embarrassingly Shallow Autoencoders. EASE is a linear model for recommendation that is built to give a prediction of the most relevant items to the user. This is done by recreating the input as its output [1]. EASE is an asymmetric solution that requires the inversion of the Gramian matrix. Gramian depends on the amount of users who have given a rating to 2 items in binary data.

EASE contains only 1 parameter, lambda. Lambda is a regularisation parameter for EASE. By testing the parameter through incremental increase, I will be able to find the optimal configuration for the parameter.

SVD: Next, I will look at Singular Value Decomposition (SVD). SVD is another linear model. The SVD of an M x N real matrix takes a factorisation of a form where the column U is a left singular value, the column V is a right singular value and the diagonal values of the M x N matrix, that has only positive numbers on it's diagonal, is the singular values. The calculation of an SVD can be done through the eigendecomposition of a matrix.

SVD contains 2 regularisation parameters, K & Mu. I will first tune K to grow in steps of 5 to see what trend the growth takes. After this, Mu will also be incremented in steps of 0.1 to observe the change in values from doing so.

BPR: Bayesian Personalized Ranking (BPR) is a pairwise approach towards dealing with the Top N recommendation problem. It uses probabilistic modelling to calculate loss, which is different to other matrix factorisation algorithms. This model is known as a generative model, which is unlike the linear models we saw with the other 2 algorithms.

The parameters are inferred from the data to make predictions. In this analysis, I tuned all of the parameters given to get the best configuration for this model. These parameters included: the dimensions, the learning rate, the number of passes, weighted regularisation of P, weighted regularisation of Q, weight of the item bias and the number of reports. These parameters were all tuned one by one, in order to get the best configuration of parameters for this model.

3.1 Measure of Success

In order to evaluate these models, I will be using Recall as a measure of success. Recall can be calculated using the recommended set, and getting the proportion of all of the items that are relevant in that set.

Below in Fig. 1 is the formula for Recall.

$$\frac{tp}{tp + fn}$$

Fig. 1: Recall formula

Recall is the true positive values divided by the true positive + false negative values. I decided to go with recall because it seemed to be the most relevant for the problem I am addressing. As N becomes bigger, the value of recall gets closer to 1, meaning an algorithm will return more relevant results. This is necessary

when thinking about recommender systems and in particular those that I am researching in this paper.

4 Evaluation

4.1 Research questions addressed

- **Research Question 1:** When tuned finely, EASE is the best collaborative filtering algorithm, outperforming SVD and BPR.
- **Research Question 2:** Linear model based algorithms tend to have higher measures of success than models that are not linear.

4.2 Datasets

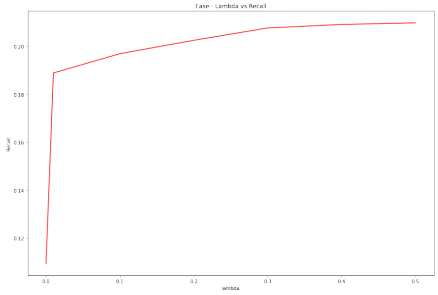
The dataset used for all of the algorithms was the "ml20m" subset from the Movie Lens dataset. The ML 20M dataset contains multiple different files. These include: ml20m_small.genscores, ml20m.genscores, ml20m.items, ml20m.tags, ml20m.test and ml20m.train.

4.3 Results

EASE Parameter Tuning: EASE has a single parameter, lambda, that needs to be tuned. Lambda is a regularisation term that is used to penalise large norms. The default for lambda was 0.01 and as we can see in the table and the graph, the lambda with the highest recall, when properly tuned, was 0.5. This can be seen in Fig. 2.

lambda	Popularity	Precision	Recall
0.00	0.17165	0.03448	0.10942
0.01	0.25072	0.05688	0.18893
0.1	0.25819	0.06021	0.19698
0.2	0.26184	0.06156	0.20258
0.3	0.26408	0.06271	0.20774
0.4	0.26549	0.06417	0.2092
0.5	0.26713	0.065	0.20987

(a) Table



(b) Graph

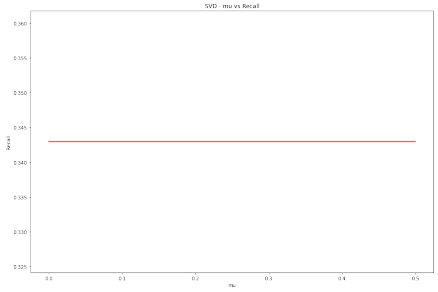
Fig. 2: EASE - Post Tuning

SVD Parameter Tuning: SVD has 2 parameters to tune, K and Mu. I first tuned K and found the best value for K to be 15. The default value for this was 5. This can be seen in Fig. 9 (Appendix).

Next I tuned the Mu value, which is displayed in Fig. 3. The Mu default was 0.0. When I began increasing the numbers, I observed little to no change in the values, leading me to believe that Mu doesn't get heavily influenced by tuning parameters within the range that I tuned them.

mu	Popularity	Precision	Recall
0.0	0.25027	0.10667	0.34295
0.1	0.25027	0.10667	0.34295
0.2	0.25027	0.10667	0.34295
0.3	0.25027	0.10667	0.34295
0.4	0.25027	0.10667	0.34295
0.5	0.25027	0.10667	0.34295

(a) Table



(b) Graph

Fig. 3: SVD - Post Tuning

Through my evaluation, I can conclude that SVD is a more effective algorithm than EASE. I came to this conclusion by comparing the recall values for both algorithms when the paramters have been tuned.

BPR Parameter Tuning: BPR has multiple parameters to tune, and with this in mind, I began by tuning the dimensions. For the default parameters, I found 10 dimensions to yield the highest recall value, so I kept this as a constant as can be seen in Fig. 10 (Appendix).

I then tried to work out the best learning rate for this algorithm, in Fig. 11 (Appendix). The learning rate is the gradient to descent proportion. As we can see, the learning rate of 0.2 gives the best recall value.

I then explored the number of passes needed to reach a more optimal solution. As we can see, 300 passes yields the highest recall value, which is in Fig. 12.

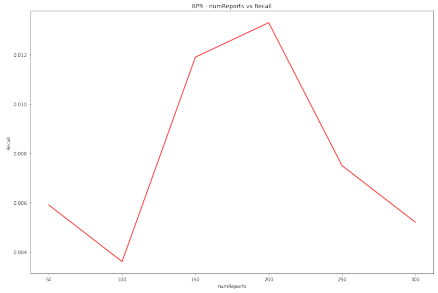
The variable regWeightP and regWeightQ both have the same value of 0.5 that give us the highest recall value, as can be seen in Fig. 13 (Appendix) and Fig. 14 (Appendix) respectively.

The bias of the item gives us the highest recall value when it's set to 0.1, in Fig. 15 (Appendix).

Lastly, we inspect the number of reports in Fig. 4. We see a peak at 200, which gives us the highest recall value.

numReports	Precision	Recall
50	0.0047	0.0059
100	0.0036	0.0036
150	0.0088	0.0119
200	0.0073	0.0133
250	0.0088	0.0075
300	0.0098	0.0052

(a) Table



(b) Graph

Fig. 4: BPR - Post Tuning

4.4 Discussion

From the tuning of the various different parameters in my analysis, I am able to address the research questions I set out at the beginning of the evaluation.

Research Question 1: When tuned finely, EASE is the best collaborative filtering algorithm, outperforming SVD and BPR.

I made this assumption due to the fact that Netflix has an implementation of this algorithm, and considering the dataset is around movies, I presumed this would excel. EASE performed well when finely tuned, however, SVD was actually the better algorithm in this case. I have been proven to be incorrect with my research question in this experiment.

Research Question 2: Linear model based algorithms tend to have higher measures of success than models that are not linear.

Due to the fact that linear models use linear algebra to come up with solutions and are proven to be highly effective when evaluated with Top-N recommendation accuracy metrics, I formulated this research question. As we can see, this research question was proven to be right, as the SVD and EASE algorithms significantly outperformed BPR.

5 Advanced Analysis

5.1 Preface

Now that the collaborative filtering algorithms have been evaluated and compared against, I can carry forward with the advanced analysis section. For my advanced analysis, I implemented a content based recommender system. Content based recommdender systems are used to recommend items that close to items that were selected in the past. Content based recommendations are used in multiple different forms of media such as recommending news articles, websites and videos.

5.2 Background

Recommendations in content based filtering algorithms are created using document-document similarity measures. A target item/items are used to calculate similarity of other items based on the target. A major advantage for content based collaborative filtering algorithms is that a recommendation can be made as soon as a user interacts with an item, without the requirement to collect vast amount of user data.

A paper around a movie genome recommender highlights the advantages of using content based filtering algorithms over collaborative filtering. It argues that content is consumed based on it's overall 'style' over just the general movie descriptors such as genre, plot and cast. By making use of movie genome, or tags and genres, a content based filtering model is implemented. The paper also mentions how content based recommender systems are more "privacy preserving". This means, far less sensitive user data is used and exploited, as is done by many other recommendation models that are implemented in systems across the internet. The implementation in the paper uses Audio, Visual and Metadata features. The Metadata features are of interest to me because they made use of tag features from the Movielens 20M dataset, which is what I am also working with. This research was the inspiration for the implementation that I did [2].

5.3 Implementation

For the purpose of this analysis, I implemented the non numeric feature similarity function, Jaccard index. The features that I took into consideration for this was the tags used in the Movie Lens genome files.

Jaccard Index: Below, in Fig. 5, you can see the formula for the Jaccard Index. The Jaccard Index takes 2 sets, A and B, and gets the intersection between the sets A and B, and divides it with the union of A and B.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Fig. 5: Jaccard Index Formula

Code & Algorithm: I implemented this code in Python, due to my proficiency of the language. I took 3 files from the datasets: genome scores, tags and items.

The genome scores dataset contains the IDs of the movie, IDs of the tags associated with the movie, and the relevance of the tags to the movie. I then created a combined dataset which contained the names of the movies and names of the tags by mapping other datasets onto the genome datasets.

The combined dataset, sorted by relevance of tags, can be seen in Fig. 6:

	movieId	name	tagId	tag	relevance
1810801	30749	Hotel Rwanda (2004)	362	ethnic conflict	1.00000
1559164	55805	Before the Devil Knows You're Dead (2007)	269	crime gone awry	1.00000
1105441	3639	"Man with the Golden Gun, The (1974)"	2	007 (series)	1.00000
1869111	48982	Flushed Away (2006)	16	aardman studios	1.00000
1103185	3633	On Her Majesty's Secret Service (1969)	2	007 (series)	1.00000
...	...	...	...	...	...
798497	2424	You've Got Mail (1998)	1002	swedish	0.00075
799970	2432	Stepmom (1998)	219	claymation	0.00075
261914	748	"Arrival, The (1996)"	219	claymation	0.00075
1352345	4823	Serendipity (2001)	1002	swedish	0.00075
217577	544	Striking Distance (1993)	1002	swedish	0.00050

Fig. 6: Combined Dataset

After this, I grouped the movieIDs using the "groupby" feature in Python. I got the top 100 most relevant tags for each movieID and stored them all in a new dataframe. I then was able to append all of the tags for a given target movie into a list. I then created a function in Python which can be seen in Algorithm 1, to apply the Jaccard formula over the given list.

Algorithm 1: Jaccard in Python

```
function jaccard(list1, list2)
    nominator ← list(set(list1).intersection(list2)));
    denominator ← list1.union(list2);
    return len(nominator)/len(denominator);
```

By setting a target movie, I was now able to receive the top 10 recommendations for this movie. I then modified this function to give me the top 10 recommendations for a user, based on the last 3 movies they saw. Algorithm 2 contains the pseudocode for this:

The function takes in 3 arguments, movies a, b and c. These are the last 3 movies a user has seen. Based off this, it gets all of the tags associated with these movies and puts them to a list. This list is then merged and turned into the

Algorithm 2: Top 10 Movies Recommender

```
function recommend(a, b, c)
    movieA ← genomescore[genomescore['movieId'] == a];
    movieA ← list(movieA[tag]);
    movieB ← genomescore[genomescore['movieId'] == b];
    movieB ← list(movieB[tag]);
    movieC ← genomescore[genomescore['movieId'] == c];
    movieC ← list(movieC[tag]);
    targetList ← merge(movieA, movieB, movieC);
    tagList[jaccard] ← jaccard(targetList, tags);
    tagList ← tagList[jaccard].sort(descending);
    tagList ←tagList.head(10);
    return tagList
```

target list. The target list then has the jaccard similarity function mapped over it, returning a similarity score for each row. These values are then sorted in descending order, and the top 10 are returned, giving the top 10 recommendations for a user based on the last 3 movies they watched.

5.4 Results

I ran this over 2 types of scenarios. Scenario 1 is where the movies are very similar in content and we observe the results received for this in Fig. 7.

	movieId	name	jaccard
1250	6377	Finding Nemo (2003)	0.626016
1595	76093	How to Train Your Dragon (2010)	0.534351
1575	68954	Up (2009)	0.511278
1600	79091	Despicable Me (2010)	0.509091
1441	45517	Cars (2006)	0.415254
365	1270	Back to the Future (1985)	0.395833
127	364	"Lion King, The (1994)"	0.394958
1551	63859	Bolt (2008)	0.376147
17	34	Babe (1995)	0.373832
731	2716	Ghostbusters (a.k.a. Ghost Busters) (1984)	0.364286

Fig. 7: Results for the movies Toy Story, Monsters, Inc. and A Bug's Life

I ran the code over the movies Toy Story, Monsters, Inc. and A Bug's Life. As we can see, the movie recommendations yield very similar movies in content.

When I run the code over 3 different movies in style, we observe the following results in Fig. 8.

	movieId	name	jaccard
1624	89745	"Avengers, The (2012)"	0.357759
1527	58559	"Dark Knight, The (2008)"	0.346154
1595	76093	How to Train Your Dragon (2010)	0.340426
1250	6377	Finding Nemo (2003)	0.336170
1132	4886	"Monsters, Inc. (2001)"	0.327354
1615	87232	X-Men: First Class (2011)	0.301653
1505	55765	American Gangster (2007)	0.301653
365	1270	Back to the Future (1985)	0.296296
1181	5349	Spider-Man (2002)	0.296296
1575	68954	Up (2009)	0.290984

Fig. 8: Results for the movies Toy Story, The Wolf of Wall Street and Guardians of the Galaxy

As we can see, we observe smaller Jaccard values for Toy Story, The Wolf of Wall Street and Guardians of the Galaxy, when the movies are different in styles, but still obtain a list of movies that are similar in content to all of the above listed movies.

6 Conclusions

This study of collaborative and content based filtering algorithms has been very insightful. There were a number of conclusions that I was able to derive from this study. To begin, it was quite clear that linear models outperformed non linear models in collaborative filtering. As well as this, Singular Value Decomposition (SVD) came out on top as the best performing algorithm for this dataset. It beat BPR, and EASE in the algorithm analysis.

On top of this, we saw a study that proposed content based recommender systems are more effective and ethical than collaborative filtering algorithms. I implemented the genome metric algorithm, as a result. This algorithm uses the Jaccard index. I created 2 forms of implementation, one that took 1 movie, and another that took 3. The basic version of my implementation took 1 movie as a target and gave recommendations based on the tags of that movie. The advanced version of the implementation allowed a user to input the last 3 movies they saw, and provided recommendations based off that. There are definitely

improvements to be made in this implementation to make it more accurate. For example, a survey asking for demographic information in a non intrusive manner could be implemented to improve the recommendations, as was done in the paper I studied [2].

I learned a lot about the various different algorithms I looked at in this analysis, and going forward, I believe more companies such as Netflix should look at adopting a less invasive and more effective system of recommendation. Content based algorithms have huge score to increase overall user experience.

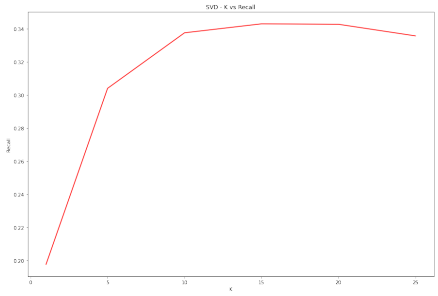
References

1. Steck, H. (2019, May). Embarrassingly shallow autoencoders for sparse data. In The World Wide Web Conference (pp. 3251-3257).
2. Deldjoo, Y., Schedl, M. and Elahi, M., 2019. Movie Genome Recommender: A Novel Recommender System Based on Multimedia Content. 2019 International Conference on Content-Based Multimedia Indexing (CBMI),.

7 Appendix

K	Popularity	Precision	Recall
1	0.3118	0.06969	0.19766
5	0.26065	0.09667	0.30399
10	0.25105	0.10521	0.33754
15	0.25027	0.10667	0.34295
20	0.25159	0.105	0.34268
25	0.25232	0.10375	0.33567

(a) Table

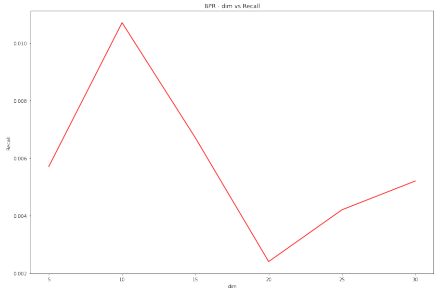


(b) Graph

Fig. 9: SVD - K vs Recall

dim	Precision	Recall
5	0.0078	0.0057
10	0.0109	0.0107
15	0.0057	0.0067
20	0.0031	0.0024
25	0.0041	0.0042
30	0.0073	0.0052

(a) Table

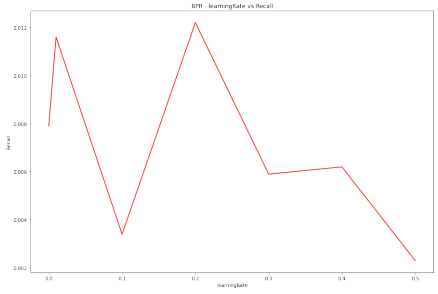


(b) Graph

Fig. 10: BPR - dim vs Recall

learningRate	Precision	Recall
0.00	0.0088	0.0079
0.01 (default)	0.0083	0.0116
0.1	0.0067	0.0034
0.2	0.0104	0.0122
0.3	0.0078	0.0059
0.4	0.0057	0.0062
0.5	0.0026	0.0023

(a) Table

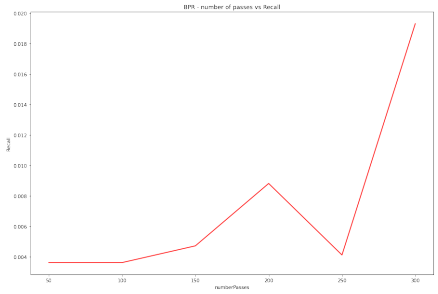


(b) Graph

Fig. 11: BPR - learningRate vs Recall

numberPasses	Precision	Recall
50	0.0041	0.0036
100	0.0036	0.0036
150	0.0047	0.0047
200	0.0093	0.0088
250	0.0047	0.0041
300	0.0140	0.0193

(a) Table

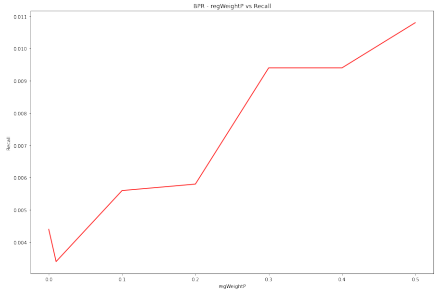


(b) Graph

Fig. 12: BPR - number of passes vs Recall

regWeightP	Precision	Recall
0.00	0.0073	0.0044
0.01 (default)	0.0052	0.0034
0.1	0.0057	0.0056
0.2	0.0041	0.0058
0.3	0.0067	0.0094
0.4	0.0083	0.0094
0.5	0.0098	0.0108

(a) Table

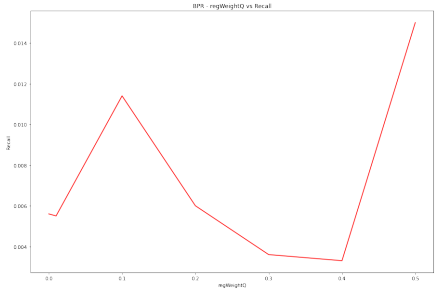


(b) Graph

Fig. 13: BPR - regWeightP vs Recall

regWeightQ	Precision	Recall
0.00	0.0073	0.0056
0.01	0.0047	0.0055
0.1	0.0073	0.0114
0.2	0.0083	0.0060
0.3	0.0021	0.0036
0.4	0.0031	0.0033
0.5	0.0088	0.0150

(a) Table

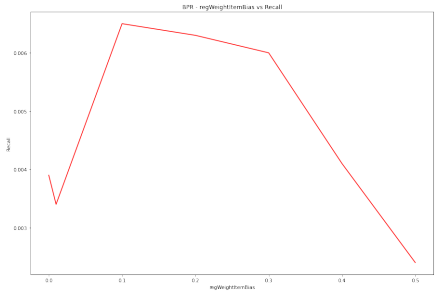


(b) Graph

Fig. 14: BPR - regWeightQ vs Recall

regWeightItemBias	Precision	Recall
0.00	0.0057	0.0039
0.01	0.0057	0.0034
0.1	0.0073	0.0065
0.2	0.0062	0.0063
0.3	0.0057	0.0060
0.4	0.0031	0.0041
0.5	0.0026	0.0024

(a) Table



(b) Graph

Fig. 15: BPR - regWeightItemBias vs Recall