Recommender System: Final Project Report

Shikha Chamoli

May 7, 2016

Contents

1 Project Objective **2**

2 Approaches used **2**
- 2.0.1 User based collaborative filtering (Memory based) . . . 2
- 2.0.2 Item based collaborative filtering (Memory based) . . . 2
- 2.0.3 Simple Bayesian collaborative filtering (Model based) . . . 2
- 2.0.4 Hybrid (Memory based) . . . 2
- 2.1 Similarity Computation & Clustering . . . 2

3 Dataset: Preprocessing and preliminary experiments **2**
- 3.1 Training and test sets . . . 3
- 3.2 Visualization of ratings data . . . 3
- 3.2.1 Inferences from data visualization . . . 4

4 Experiments and Evaluations **4**
- 4.1 Evaluation Metrics . . . 4
- 4.2 Performance comparision of Similarity Measures . . . 4
- 4.2.1 Summary Table (Which similarity measure is best?) . . . 5
- 4.3 Comparison of UBCF , IBCF, Bayesian & Hybrid models . . . 5
- 4.3.1 Sample of confusion matrix . . . 7
- 4.4 Summary table on performances . . . 8

5 Challenges encountered during this project **8**

6 Conclusion **8**

References **9**

1 Project Objective

Building a movie recommender system using collaborative filtering approach. Analysis and performance comparison of User based, Item based and bayesian model base hybrid collaborative filetring techniques.

2 Approaches used

2.0.1 User based collaborative filtering (Memory based)

The can be constructed solely from a single user's behavior or from the behavior of other users who have similar traits. [6]

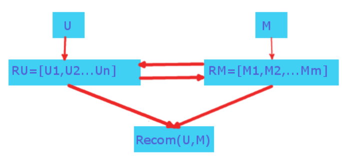
2.0.2 Item based collaborative filtering (Memory based)

This utilizes user rating data to compute the similarity between items. [6]

2.0.3 Simple Bayesian collaborative filtering (Model based)

This model is in part inspired by [2] and [1]. Given a User (U) and a Movie (M) not rated by U, we want create a simple bayesian classifier, that takes inputs from both "Related Users" RU and "Related Items" RI. "Related Users" are those users who are similar to U, in their choice of movies, so RU will be created using a distance measure as in UBCF. "Related Items" are those movies that are similar to M and RU is computed using some similarity measure as in IBCF. The model will compute conditional probability of U liking M, conditioned on the actions of RU on MI.

The model can be pictorially represented as :



2.0.4 Hybrid (Memory based)

This combines the best results of user based and item based collaborative filtering methods. [6]

2.1 Similarity Computation & Clustering

We used three of the best performing similarity measures here :

1. Correlation-Based Similarity (Pearson correlation)
2. Vector Cosine-Based Similarity
3. Jaccard Distance

3 Dataset: Preprocessing and preliminary experiments

We used MovieLens Dataset. This dataset describes 5-star rating, a movie recommendation service. All users had rated at least 20 movies. Links contain movie ids from imdb and tmdb. Movies contain movie titles and genres. Ratings contains per user, per movie ratings (not all users rate all movies). Tags contain user specific personalization tags on movies.

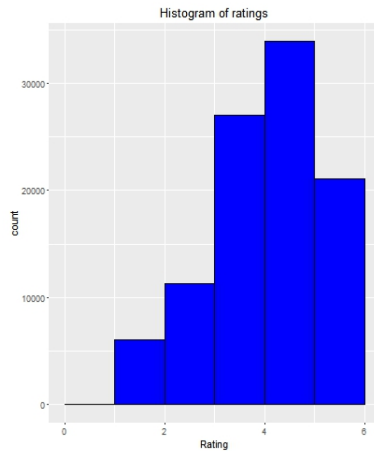
# Movies	# Users	# Ratings	Files in Dataset	Size
10329	668	105339	'links.csv', 'movies.csv', 'ratings.csv' and 'tags.csv'	144 MB

3.1 Training and test sets

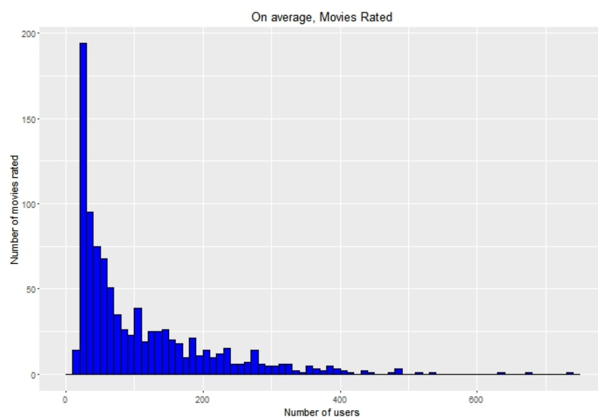
We split the ratings data into training and test sets with a ratio of 80% - 20%.

3.2 Visualization of ratings data

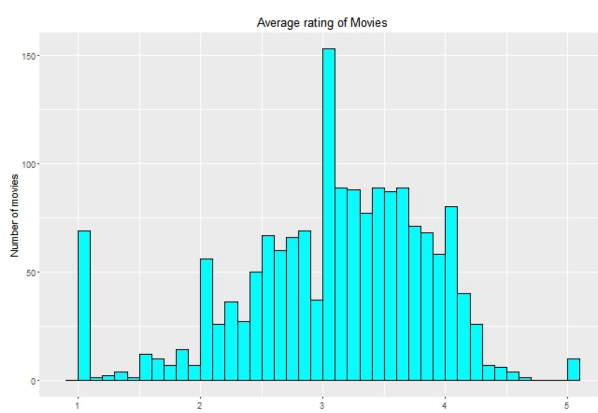
We visualized the ratings data to get a basic understanding of users rating behaviours. Following are the plots.



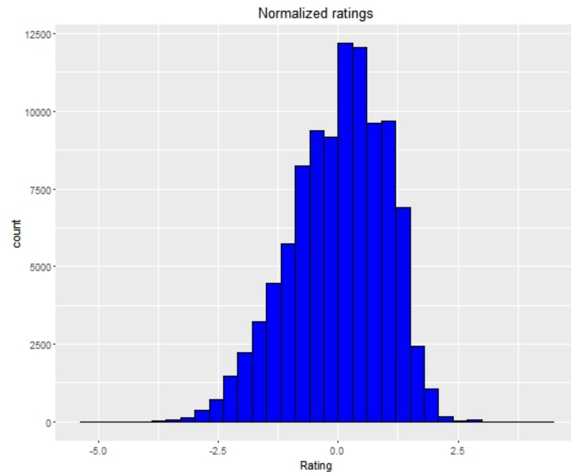
Plot 1- Ratings & their counts



Plot 2- Movies rated on average



Plot 3- Average rating/movie



Plot 4- Normalized ratings(Z-score)

3.2.1 Inferences from data visualization

Plot 1	Plot 2	Plot 3	Plot 4
Major ratings range from 3 - 4 (on a scale of 0 to 5)	Right skewed curve seen. **Ineference**: with time, the # ratings given by a user depreciates.	Bell shaped curve. **Ineference**: data is normally distributed and highest movie rating are mostly around 3.	Normalized data using Z- score to reduce individual rating bias.

4 Experiments and Evaluations

Programming Language Used	Packages	OS	My Major contribution
R 3.2.4	recommenderlab [3]	Windows 10	Eextended Recommenderlab library added Bayesian model module.

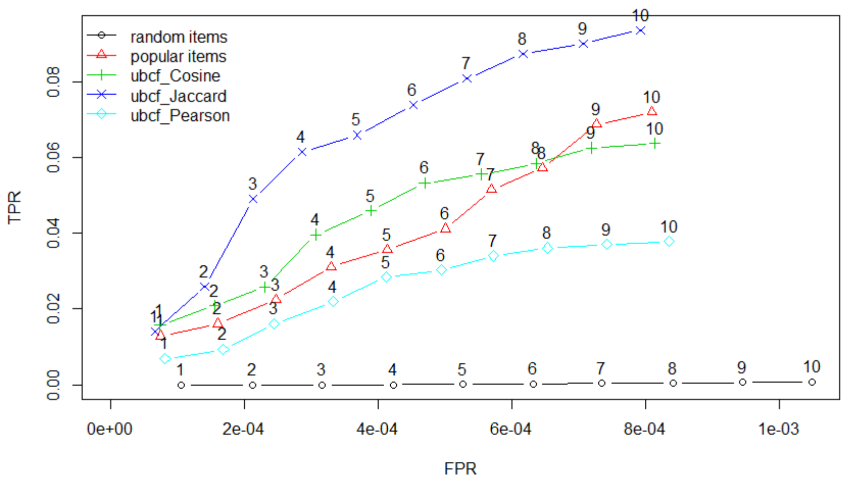
I extended the Recommenderlab library by wiriting my own bayseian model to reccomend movies and tested it against UBCF, IBCF. Both item based and user based methods require a measure of similarity (User-User or Item-Item). The choice of similarity measure is subjective and some measures may be more suited to our data, than others. Therefore we have spend some time, exploring various methods and in this section we present our findings.

4.1 Evaluation Metrics

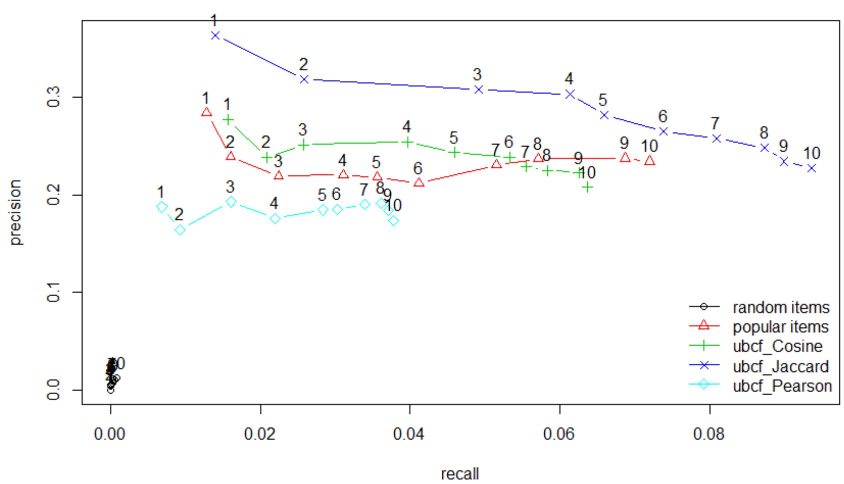
Compared performnces of UBCF, IBCF, Hybrid, Bayesian model using ratings data. The classifier uses confusion matrix and we show performances of the four techniques using ROC curve and a Precision-Recall curve for this[7]. ROC measures the fraction of negative examples that are misclassified as positive. Precision measures that fraction of examples classified as positive that are truly positive.

4.2 Performance comparision of Similarity Measures

We ran the UBCF recommendation engine for various similarity measures (most of which are implemented in [4]). I've also plotted a benchmark recommendation "Popular" engine in [3], which recommends the most popular movies to all users, without any conditioning on any other information. So all good algorithms must perform better than simplistic polular method. Following were our results :



Plot 5: ROC curve of ubcf with 3 similarity measures, popular, random benchmark



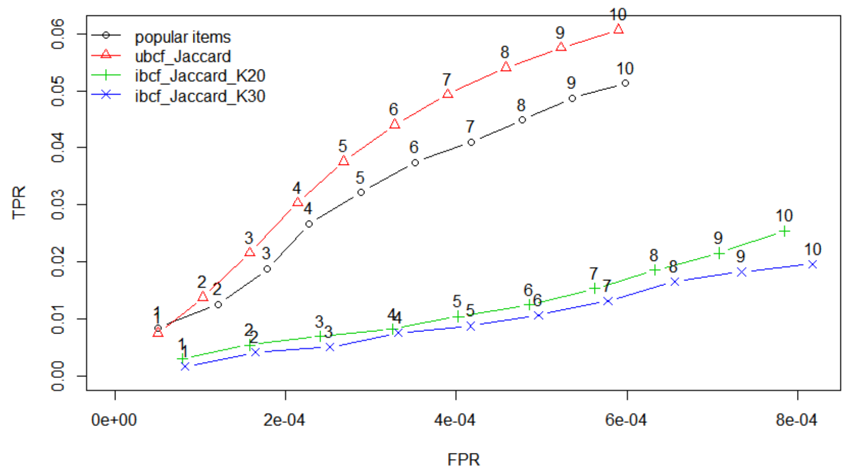
Plot 6: Precision Recall curve of ubcf with 3 similarity measures, popular, random benchmark

4.2.1 Summary Table (Which similarity measure is best?)

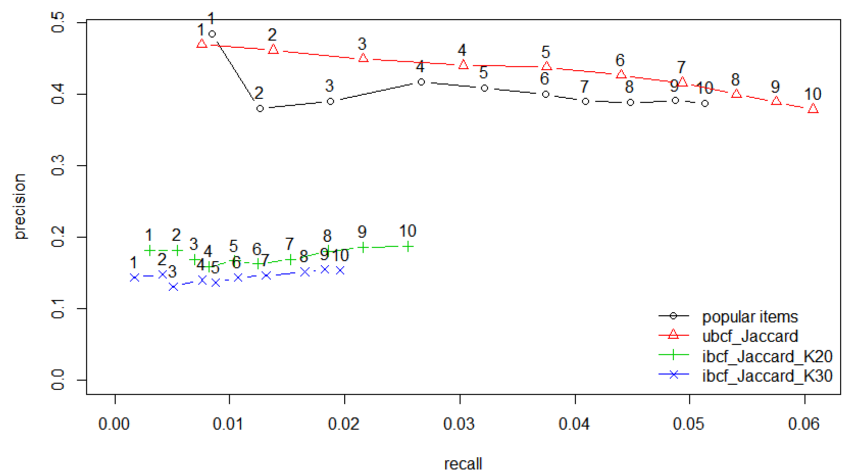
Similarity Measure	Performance	Interesting Results
Jaccard	**Best for both UBCF, IBCF**	-
Cosine	Good	-
Pearson	Average	-
Popular (recommends popular movies to all users- Benchmark)	Good	Popular forecasts compares well with advanced method like “UBCF using pearson”

4.3 Comparison of UBCF , IBCF, Bayesian & Hybrid models

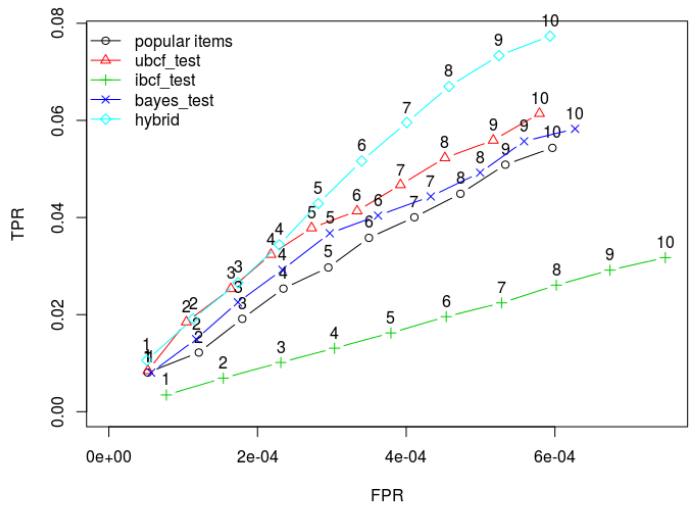
Comparision graphs of ibcf and ubcf with popular benchmark (Plot 7, Plot 8)



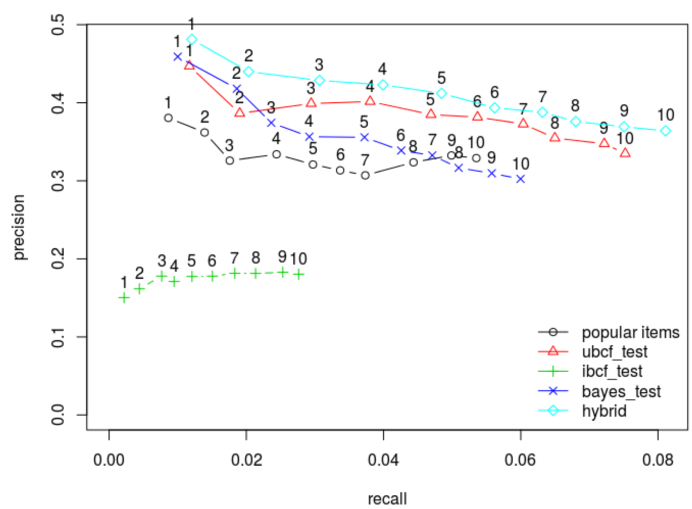
Plot 7 : ROC curve (ubcf, ibcf, popular)



Plot 8 : Precision Recall curve (Popular, UBCF with Jaccard distance, IBCF with cluster sizes 20,30)



Plot 9 : ROC curve (Popular, UBCF with Jaccard distance, IBCF with cluster sizes 20,Bayes, Hybrid)



Plot 10 : Precision Recall curve (Popular, UBCF with Jaccard distance, IBCF with cluster sizes 20,Bayes, Hybrid)

4.3.1 Sample of confusion matrix

Bayesian confusion matrix sample:

	TP	FP	FN	TN	precision	recall	TPR	FPR
1	0.440299	0.477612	111.1567	10202.93	0.479675	0.007883	0.007883	4.6512E-05
2	0.768657	1.059701	110.8284	10202.34	0.422764	0.014412	0.014412	0.00010331
3	1.104478	1.634328	110.4925	10201.77	0.406504	0.021917	0.021917	0.00015936
4	1.380597	2.268657	110.2164	10201.13	0.382114	0.026679	0.026679	0.00022131
5	1.597015	2.962687	110	10200.44	0.354472	0.029607	0.029607	0.00028929
6	1.828358	3.641791	109.7687	10199.76	0.338753	0.034167	0.034167	0.00035571
7	2.141791	4.231343	109.4552	10199.17	0.340302	0.038932	0.038932	0.00041329
8	2.373134	4.902985	109.2239	10198.5	0.330285	0.043554	0.043554	0.00047896
9	2.61194	5.567164	108.9851	10197.84	0.323397	0.04935	0.04935	0.00054389
10	2.858209	6.223881	108.7388	10197.18	0.318699	0.053041	0.053041	0.00060809

Hybrid confusion matrix sample:

	TP	FP	FN	TN	precision	recall	TPR	FPR
1	0.462687	0.537313	111.1343	10202.87	0.462687	0.010615	0.010615	5.26E-05
2	0.873134	1.119403	110.7239	10202.28	0.436567	0.021335	0.021335	0.00011
3	1.30597	1.679104	110.291	10201.72	0.435323	0.031216	0.031216	0.000164
4	1.671642	2.30597	109.9254	10201.1	0.41791	0.039662	0.039662	0.000226
5	2.037313	2.932836	109.5597	10200.47	0.407463	0.04666	0.04666	0.000287
6	2.358209	3.597015	109.2388	10199.81	0.393035	0.052745	0.052745	0.000352
7	2.731343	4.208955	108.8657	10199.19	0.390192	0.061628	0.061628	0.000412
8	3.044776	4.880597	108.5522	10198.52	0.380597	0.069147	0.069147	0.000478
9	3.358209	5.552239	108.2388	10197.85	0.373134	0.075732	0.075732	0.000543
10	3.656716	6.238806	107.9403	10197.16	0.365672	0.081394	0.081394	0.00061

4.4 Summary table on performances

	Popular	**IBCF**	**UBCF**	**Hybrid**	**Bayesian**
Training time	NA	Very slow	Fast	Average	Slow
Prediction time	Fast	Fast	Slow	Fast	Fast
Performance	Average	Bad	Good	**Best**	Good
Accuracy of predictions	Bad	Bad	Good	**Best**	Good
Model Complexity	Least	Average	Average	High	High
Interesting observation	Performs comaprable to UBCF with pearson	-	-	-	-

5 Challenges encountered during this project

1. In extending "recommenderlab" I came across multiple signature for some in-built functions so it took me a lot of time to figure out how to use it in my implementations of Bayesian and Hybrid models.
2. Not enough documentation /literature of recommenderlab that could be used while extending this library, It took me a lot of internet serach and readings to finally fix this.
3. Solution of above problems: I figured, I'll need to write these new models in S4 to extend this recommenderlab library for new Bayesian and Hybrid models.
4. I couldn't perform the tests with larger datasets available on movielens , R kept crashing, so I picked the small dataset available there.
5. Took a lot of time to run specially for training IBCF models.

6 Conclusion

1. Both User Based, and Item Based collaborative filtering techniques work based similarity measures. We tried out a range of similarity measures, and concluded that the Jaccard Distance performs the best.
2. We chose simplistic benchmark- Popular which recommends top most popular movies to all users, without any conditions involved. So, all other models that are considered good must perform better than this.
3. IBCF took a lot of time in training data. In our MovieLens data, the number of Items is much larger than the number of movies, hence the difference.

4. Bayesian classifier is better than naive benchmark and it doesn't use any extra information as compared to UBCF.

5. UBCF in general performs better than IBCF. We realized through our experiments, that standalone IBCF method can perform worse than a naive benchmark of "Popularity" engine (which recommends popular movies to all users).

6. I extended the recommenderlab library by implementing Bayesian(Bayes_test) and Hybrid models.

7. Hybrid model performs the best in this dataset.

References

[1] Luis M. de Campos, Juan M. FernÃ¡ndez-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785 – 799, 2010.

[2] Shengbo Guo. *Bayesian Recommender Systems: Models and Algorithms*. PhD thesis, The Australian National University, 10 2011.

[3] Michael Hahsler. recommenderlab: A framework for developing and testing recommendation algorithms, 2011.

[4] David Meyer and Christian Buchta. *proxy: Distance and Similarity Measures*, 2015. R package version 0.4-15.

[5] Toby Segaran. *Programming Collective Intelligence*. O'Reilly, first edition, 2007.

[6] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[7] Michele Usuelli Suresh K. Gorakala. *Building a Recommendation System With R*. PACKT Publishing, 1 edition, 2015.