Recommendation Systems
An Aggregation of Approaches

Arjun C
School of Information Technology and Electrical Engineering
The University of Queensland, Qld, 4072, Australia

Abstract

Recommendation systems are the engines that try to predict what a user might like based on his history of likings. The engine has to identify relationship of user's likings to others' or the relationship of items based on user's history and build recommendations. It is not very easy to build a personalized space for a user and come up with recommendation given the fact that preferences keep altering and also due to the sparsity in data available to the systems. This paper describes different approaches used to build recommendations on the movielens dataset [1]. Popularity based recommendations, Content Based recommendations, Collaborative filtering using Matrix Factorization and Auto-encoders will be discussed here with some information on evaluation metrics.

1 Introduction

The increasing use of web for product sales and digital media content access has led to the development of recommendation systems. It has become very easy for users to provide feedback on the content they use (e.g. YouTube videos). Typical ways of providing feedback are "Rating" or "Like or Dislike". The objective of recommender systems is to deduce user preferences and item similarities. Here, the entities considered are "Users" and their likeliness or rating for the "Items". The basic principle of recommender systems is that there is an interdependency on the user-item interactions, i.e. user who likes many comedy movies is more likely to prefer comedy movies over historical movies. These correlations are used by the system to learn user and item features to come up with recommendations [2].

Recommender systems are typically used to target two problem definitions. They either try to predict the user rating for unrated items or try to predict top N items as recommendations. Possible approaches, their basic idea and the results are discussed in the following sections.

2 Background

In this section, a simple introduction to the methods is given along with the related works.

The dataset used is the movielens dataset with approximately 1 million ratings (in the range 1-5) by a total of 6040 users for 3883 movies. The data would be used as a matrix of (Users X Movies) with cell values being user's rating for the movie and 0 for unrated.

Populaity based method: As put forward by Aggarwal [2], the most basic implementation of recommender systems is to suggest the top N items to every user. The assumption here is that the item rated the most is the most popular, e.g. if a movie "Star Wars" has received the most number of ratings, this will be recommended to every user.

The advantage of this method is that there is no user preference or item attributes required. The main disadvantage is that it is not personalized. This method is used to solve the cold start problem (when the system does not have any information on a new user's ratings).

Content based method: As discussed by Parivash et al. [3], this method is mainly reliant on the item description. In the case of movielens dataset, each movie is categorized as belonging to a set of genres. If Alice likes a movie "Toy Story", which belongs to "Animation" and "Children" genres, the system has to identify movies that belog to these genres. The movie similarity is calculated using distance measures like cosine similarity. This is still a good method given the categorization of data on the web.

Collaborative filtering method: This method is based on predicting ratings for unrated movies based on similarities in the ratings between users. E.g. If Alice and Bob are identified to have similar liking, if Alice has rated four movies as 5,5,2,1 and Bob has rated three movies as 5,4,*,1, then the unrated third movie's rating would be very close to 2. The major techniques used are Matrix Factorization [4, 5] and Autoencoders [6-8].

Matrix Factorization: Given a rating matrix "R" of (Users X Movies), it can be reduced to two lower rank matrices P and Q, which can then be used to reconstruct R. This can be implemented in various ways [5], but this is not a very scalable approach.

AutoEncoders: A type of neural network used for compression. This identifies latent features in the data and tries to reconstruct the input data. Output layer has the same number of nodes as Input. This has been majorly considered for recommenders.

The main advantage of Collaborative filtering is that they are personalized and proven to predict ratings very closely. The main disadvantage is that it is not capable of recommending to new users.

Evaluating Recommender systems: The systems that try to predict user ratings are evaluated using RMSE on only the rated items [5].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2}$$

"n" being the number of ratings in the test set, p_i and r_i being predicted and actual rating.

It can be argued that none of the metrics is suitable for all purposes, based on the fact that user preferences change. But, it still is necessary to evaluate with the information we have.

3 Implementation and Results

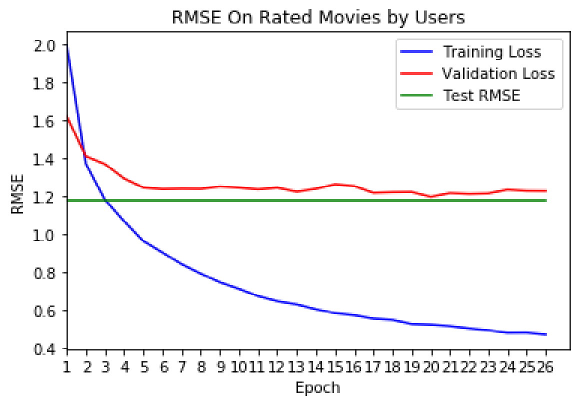
The basic train (70%), test (30%) and validation (20% of train) splits were carried out for Matrix factorization and Autoencoders.

Popularity Based Method: Top 10 movies were identified by ranking movies on the basis of number of ratings. The returned results are ["American Beauty", "Star Wars 4", "Star Wars 5", "Star Wars 6", ……..]. These still are good results for new users.

Content Based Method: This system was implemented using nearest neighbours method. The algorithm used was out of the box python implementation. The mapping of user's profile was not implemented as the main goal was to identify the similar movies based on Genre. The results for movies similar to "Toy Story" returned ["Toy Story", "Toy Story 2", "Chicken Run", "Bug's life"…..].

Collaborative Filtering: This method was tried out using Matrix Factorization on the idea suggested by Gábor et al. [5]. The algorithm performs really well for matrices of very less order. However, with increase in the matrix order and the sparsity, the time taken for each iteration is extensive. An execution of this on a very small matrix (not movielens-1m) gave an average RMSE of 0.5. The issue is that it is computationally expensive. Hence, not scalable. When python implementation of Singular Value Decomposition (matrix factorization method) was used out of the box (does not optimize on the loss only on rated movies) it still gave good recommendations. One of the major disadvantages is that when there is a new user, with ratings, the whole of the algorithm has to be run including the new user.

Autoencoders were implemented using Keras with Tensorflow backend. Two hidden layers of size 200 each (different number of nodes were tried out but this gave reasonable results in very less time), with custom loss as RMSE was built. This gave a very good result in less that 30 epochs on use of early stopping (to avoid overfitting). The RMSE achieved was 1.17, which is comparatively good for the computation time. Some of the results are reported to have an RMSE of 0.96 to 1.1[6]. The resultant graph of the experimentation is as below.



4 Conclusion

From the results it can be said that all methods have their pros and cons. It is up to the implementer to select the right one or an ensemble of these. End of the day, businesses are all about profit and as long as something suits their needs with least cost, it is implementable. It is safe to say that no recommender system can be perfect, given the fact that users' preferences keep changing. It is very much agreeable that recommendation systems have changed the way we use web and there is major scope for research and improvement in this area.

Acknowledgment

I extend my gratitude to Prof. Neil Bergmann and my tutor Ms. Saskia Van Ryt for their guidance.

References

[1] F. Harper and J. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 1-19, 2016.
[2] C. C. Aggarwal, Recommender Systems The Textbook. Cham: Cham : Springer International Publishing : Imprint: Springer, 2016.
[3] P. Pirasteh, J. J. Jung, and D. Hwang, "Item-based collaborative filtering with attribute correlation: A case study on movie recommendation," vol. 8398, ed, 2014, pp. 245-252.
[4] E. Kharitonov, "Empirical Study of Matrix Factorization Methods for Collaborative Filtering," in Pattern Recognition and Machine Intelligence: 4th

International Conference, PReMI 2011, Moscow, Russia, June 27 - July 1, 2011. Proceedings, S. O. Kuznetsov, D. P. Mandal, M. K. Kundu, and S. K. Pal, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 358-363.
[5] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," ed, 2008, pp. 267-274.
[6] Y. Ouyang, W. Liu, W. Rong, and Z. Xiong, "Autoencoder-Based Collaborative Filtering," in Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part III, C. K. Loo, K. S. Yap, K. W. Wong, A. T. Beng Jin, and K. Huang, Eds. Cham: Springer International Publishing, 2014, pp. 284-291.
[7] S. Sedhain, A. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders Meet Collaborative Filtering," ed, 2015, pp. 111-112.
[8] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," Expert Systems With Applications, vol. 69, pp. 29-39, 2017.

Biography

Arjun C, currently pursuing Master of Computer Science at The University of Queensland, also an Entrepreneur, who successfuly ran Lush pillows for 2 years as a student. I received my Bachelor of Engineering from The National Institute of Engineering, India, after which I worked with Unisys for 18 months as a software developer. My major implementation being Speech Recognition using Deep learning and a journal publication on the same. I have also worked on building commercial websites. My biased areas of interest include data mining and machine learning.