A machine learning approach to TV recommendation

Fahim Md. Rafiq
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh

Mir Ishrak Maheer Dhruba
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh

Nawab Haider Ghani
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh

Tahsinur Rahman
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh

***Abstract*— During recent time's machine learning is shaping our recommendation systems to reach new heights by analyzing large sets of data quickly using extensive computational powers which weren't possible before. One such large set of data consists of TV shows and its viewers and therefore we made a TV show recommendation system using a machine learning approach known as the Collaborative Filtering model. In our paper we have described the way we implemented our algorithm and have explained the results it had generated. We furthermore determined the accuracy of our algorithm and a list of TV shows which our users might enjoy. On doing so we have learned that our algorithm works for cold-start conditions where much of the TV show's ratings are unknown. This is was a significant finding since this infers that our system will be able to work where very few training data points are present.**

Keywords—Recommender system; TV shows; Collaborative filtering

I. Introduction

In the ever-growing media industry the number of TV shows has risen significantly over the decade. Viewers welcome streaming systems which provide them with personalized contents based on their previous viewing history or ratings which in turn results in a very personalized list of TV shows which the user might enjoy. This can generate very accurate results if the user's past viewing history is already known in large quantity since most of the recommendation algorithms heavily depend on it.

Other recommendation systems depends on profiling the users such as taking a survey which finds out what genre of TV show the user likes and recommends accordingly or uses the viewer's watch history and finds TV shows of similar genre. Popular recommendation system algorithms consist of SVM (Support Vector Machine), SVD (Singular-Value Decomposition), Naive Bayes, CNN (Convolutional Neural Network) and Decision Trees.

However, if the profiling approach is not used and if a user has recently joined a service then the system won't be able to provide a personalized recommendation of TV shows due to lack of watch history of the user [1]. This cold-start problem [1][2] however can be solved using a machine learning approach known as Collaborative filtering model which uses stochastic gradient model [7][8]. This in turn with the profiling system will generate a much more accurate prediction of the TV shows which the user might enjoy more.

Therefore, keeping in mind the cold-start problem, this paper explains in detail how a latent matrix is used with user ratings and TV shows. Our dataset consists of users rating a few TV shows leaving much of the TV shows unrated. The dataset is split into a 75:25 ratio where the first 75% is used for training and the latter 25% is used for testing [4]. In the end, our system is able to provide a list of top 10 TV shows which our user might enjoy the most.

Finally, our system is created using python 3.0 which uses data manipulation and data-science libraries such as Scikit-learn, Pandas, NumPy and Matplotlib.

II. Methods

A. Dataset

The dataset for the project is acquired from the TVDB website and is made up of two separate files. The first file titled tvshows consists of a numeric tvId, title of the show, a unique tvdbID, the network the show is broadcasted on, tvdb rating of each show and the genres that the show belongs to. It has 462 instances, representing 462 different shows. The second file named ratings consists of userid, tvid and rating which represents what each user rated a particular show out of 5. It has 499997 instances of 3374 users that rated 462 different TV shows. Plotting these data on a histogram showed that most users provided a rating for only a few TV shows while majority of TV shows also received only a few ratings.

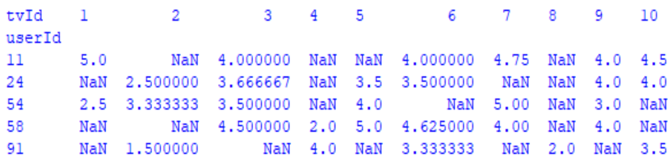
B. *Feature Selection and Extraction*

The genre feature is based on values between 0 and 1 to demonstrate which genre a TV show belongs to. Since most TV shows belong to different genres with being more inclined to one or more genres it is hard to represent it using numeric values. Hence, genre is not considered as a feature in this model since these values are arbitrary. The primary features are tvId, userId and rating where each instance represented the rating given to a particular show by a specific user. New features are extracted from the existing features. These are count review by user and count review by TV shows which is the number of ratings given by a particular user and the number of ratings received by each TV show respectively. Also two other features are extracted, which are average rating by user and average rating by TV show. Average rating by user is the mean of all ratings given by that user while average rating by TV show is the mean of all ratings received by that show. 8 features were used for the final model including the index feature.

C. *Rating Prediction and Recommendation*

The recommendation system is based on the TV show rating data using a model-based Collaborative filtering approach, Matrix factorization, to predict the rating of the TV show a user has not rated yet and to provide list of recommended TV shows based on those predicted rating [3]. The concept behind matrix factorization is to learn latent factors associated with users and TV shows. These latent factors can be thought of user's preferences and TV show characteristics respectively. These are hidden features but given the rating data, it is possible to learn from them.

For the final model only part of the dataset is used, consisting of 8 features and 255110 instances of users who have rated more than 300 shows and shows which have been rated more than 600 times. Cross-validation [4] is used to split the data into 0.75:0.25 ratio of which 75% was kept for training and 25% is used for testing the data. Then the rating data of both training and test set is organized into n x m matrix, where n is the number of users and m is the number of TV shows. After that the missing values in the dataset or the NaN values are filled with 0.



```
tvId    1         2         3    4    5         6     7    8    9   10
userId
11      5.0       NaN  4.000000  NaN  NaN  4.000000  4.75  NaN  4.0  4.5
24      NaN  2.500000  3.666667  NaN  3.5  3.500000   NaN  NaN  4.0  4.0
54      2.5  3.333333  3.500000  NaN  4.0       NaN  5.00  NaN  3.0  NaN
58      NaN       NaN  4.500000  2.0  5.0  4.625000  4.00  NaN  4.0  NaN
91      NaN  1.500000       NaN  4.0  NaN  3.333333   NaN  2.0  NaN  3.5
```

Fig. 1. Matrix representing ratings given by each user to a particular TV show

The latent factors of users and TV shows are learned by modeling the problem as a linear regression one through the minimization of the cost function [5]. The cost function is minimized by:

$$\min_{q,\, p} \sum_{(u,\, i) \epsilon K} \left(r_{ui} - p_u^T q_i\right)^2 + \lambda\left(\parallel p_u \parallel^2 + \parallel q_i \parallel^2\right) \qquad (1)$$

Here P is the vector representing the latent factors for the user, Q is the vector representing the latent factors for the TV show and λ is the regularization term (L2 penalty) to prevent overfitting [6]. In this model λ is initialized to a value of 0.1. Then the algorithm of the stochastic gradient descent [7][8] is implemented by first initializing the vectors P and Q with random number of latent factors. For this model the number of latent factors is 20. After that, until convergence occurs or the maximum number of iterations is reached, the gradient is calculated for each data point. Additionally, the P and Q vectors are also updated simultaneously. The gradient descent algorithm is written as:

$$Q_{(i\,+\,1)} = Q_{(i)} + \gamma(e_{ui} \cdot P_u - \lambda \cdot Q_i) \qquad (2)$$

Here γ is the learning rate or step size to update gradient [7] and eui is the error term. In this model the learning rate is initialized to 0.01 and the maximum number of iterations or epochs to 100. Finally, the prediction of each user on a particular TV show is determined through the dot product of the vectors P and Q:

$$\hat{r}_{ui} = P_u^T Q_i \qquad (3)$$

D. *Analytical Methods*

For this system the python language is used as the main programming language. Specifically, python version 3 is used since it is the latest version and has more advanced functionalities than its predecessors. A few python packages are used in the system. Numpy and pandas are used to process the data. Matplotlib is used to visualize the data and results. The package Scikit-learn is used to train the machine learning model that makes the prediction.

III. RESULTS

We implemented the Collaborative Filtering system and Stochastic Gradient Descent and measured the prediction accuracy with test set from TVDB datasets. After measuring the accuracy of the prediction, the algorithm is used to suggest the TV shows to be watched by the user. No specific patterns could be found between the recommended TV shows, which show that the shows are based on the preferences of the individual users. These preferences were hiding as latent factors which were found by the system. The system also predicted possible rating of a user on a TV show they have not rated yet. The first experiment finds the accuracy of the prediction by comparing the Root-Mean-Squared Error (RMSE) value of the training and testing dataset. As the number of Epochs increases, the difference between the RMSE value of training and testing data becomes stable. The accuracy of the prediction is around 85%. The formula used for finding RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (X_{obs,i} - X_{model,i})^2}{n}} \qquad (4)$$

Here, n is the total number of test data

Xobs is the test item of the matrix

Xmodel is the predicted item of the matrix

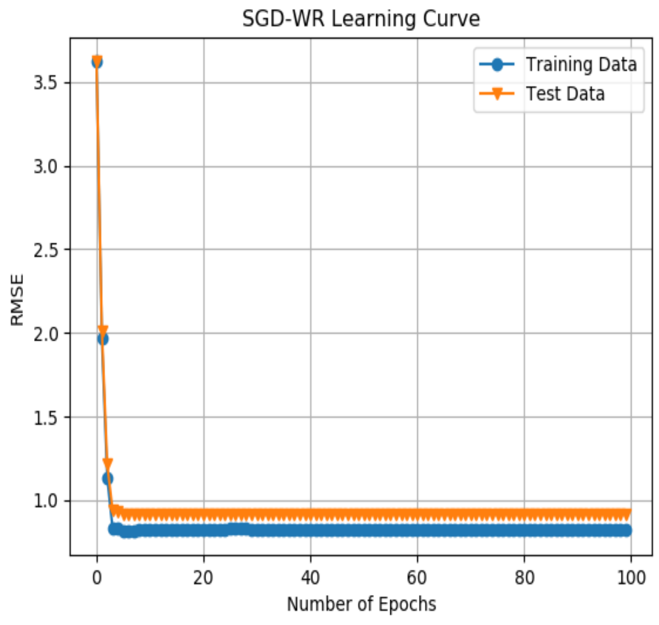
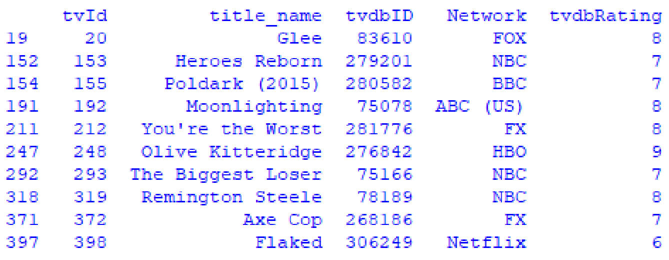


Fig. 2. Learning Curve using RMSE

The second experiment finds the recommended TV shows for the user to watch. It sorts the TV shows according to the ratings provided by other users and gives a list of shows recommended for the user.



```
     tvId          title_name  tvdbID     Network  tvdbRating
19     20                Glee   83610         FOX           8
152   153      Heroes Reborn  279201         NBC           7
154   155     Poldark (2015)  280582         BBC           7
191   192        Moonlighting   75078   ABC (US)           8
211   212   You're the Worst  281776          FX           8
247   248   Olive Kitteridge  276842         HBO           9
292   293   The Biggest Loser   75166         NBC           7
318   319    Remington Steele   78189         NBC           8
371   372             Axe Cop  268186          FX           7
397   398              Flaked  306249     Netflix           6
```

Fig. 3. TV shows recommended to the user

It can be observed that the shows have no major similarities between them. They do not all belong to the same network. One of them like Flaked has a low average TVDB rating. However, it is still recommended since the system predicts that the user can maybe like this show.

$\alpha + \beta = \chi$ (1) (1)

IV. Discussion

Our system does not have a major pattern in the observation because, unlike other related systems, our system suggests top 10 shows to a user based on the relationship that the other users that have watched and rated similar shows as the user, have rated these shows highly.

This system does not categorize shows according to genres because it is difficult to establish the exact genre that a show belongs to. Thus an exception of suggesting a show of different genre than the user's choice may exist.

A major disagreement to previous works would be the profiling system that the systems use such as surveying the users to know their preferred genre of shows. Also the incapability of the previous systems to work without the ratings of each show by each user is another motivation to develop such a system as this.

The results of our system reveal that a user who has rated a show similarly to another user is actually interested in shows that the other user have already watched and rated highly even though the rating in TVDB may be low for that particular show. We can justify this finding from the result of our system given as screenshot in the results section.

Therefore we can say that our system has solved the problem of recommendation systems' incapability to work without full dataset of ratings and users and also eliminating the difficulty of assigning a specific genre to a show resulting in a recommendation show that has very few or no limitations.

V. Conclusion

To conclude, such recommender systems are becoming very important nowadays since a lot of viewers are shifting from movies to TV shows. Even though movie recommendation systems do exist but TV show recommendation systems are rare. This is due to the lack of proper dataset of TV shows. However, this system tackles this problem by making efficient use of fewer amounts of data.

The main observations from the system is that with even a small number of users rating only a few shows, it is possible to recommend TV shows to an user by finding latent factors to find similarities between different users and different TV shows.

From the results it can be seen that the TV shows recommended to the user have no discernible pattern but rather recommended solely on the preferences of the particular user. These preferences come from the hidden latent factors that were found in the system. That is what makes this system unique and efficient.

Companies like Netflix, Hulu, and Amazon use these kinds of algorithms to recommend TV shows. This shows the importance of such a system. With better computational power and larger datasets it is possible to increase the accuracy of the system even further and make the system even better.

References

[1] Hyeong-Joon Kwon, Student Member, IEEE and Kwang-Seok Hong, Member, IEEE, “Personalized Smart TV Program Recommender Based on Collaborative Filtering and a Novel Similarity Method”, IEEE Transactions on Consumer Electronics, Vol. 57, No. 3, August 2011.

[2] Lam, Xuan Nhat, et al. "Addressing cold-start problem in recommendation systems." Proceedings of the 2nd international conference on Ubiquitous information management and communication. ACM, 2008.

[3] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009).

[4] Browne, Michael W. "Cross-validation methods." Journal of mathematical psychology 44.1 (2000): 108-132.

[5] Govan, Anjela. "Introduction to optimization." North Carolina State University, SAMSI NDHS, Undergraduate workshop. 2006.

[6] Ng, Andrew Y. "Feature selection, L 1 vs. L 2 regularization, and rotational invariance." Proceedings of the twenty-first international conference on Machine learning. ACM, 2004.

[7] Gemulla, Rainer, et al. "Large-scale matrix factorization with distributed stochastic gradient descent." Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011.

[8] Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010. 177-186.

[9] Buczak, Anna, John Zimmerman, and Kaushal Kurapati. "Personalization: Improving ease-of-use, trust and accuracy of a tv show recommender." (2002).

[10] Oh, Jinoh, et al. "When to recommend: A new issue on TV show recommendation." Information Sciences 280 (2014): 261-274.