International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

An Analysis of Machine Learning Methods for Ranking in Recommendation Systems

Shubham Milind Phal[1], Smriti Srivastava[2]

[1]*Student, Dept. of Computer Science & Engineering, RV College of Engineering, Karnataka, India*
[2]*Assistant Professor, Dept. of Computer Science & Engineering, RV College of Engineering, Karnataka, India*

Abstract - *Recommendation systems are an integral part of any e-commerce business. Several prominent e-commerce websites such as Amazon, Netflix etc use recommendation systems in order to enhance the quality of user experience by providing intuitive and personalized recommendations. Ranking of items is an important step in providing relevant recommendations. Primarily Machine-Learned Ranking (MLR) methods have been used in a multitude of information retrieval problems such as online-advertising, document retrieval etc. The ranking function is generally learned using either a Pointwise, Pairwise or a Listwise approach. In this work we analyze the efficacy of several prominent machine learning methods that are used to rank items. We apply these methods on the popular open-source MovieLens 100K Dataset and summarize the results.*

Key Words: **Content Based Filtering, Collaborative Filtering, Learning to rank, Normalized Discounted Cumulative Gain (NDCG), Gradient Boosting, AdaBoost, Light GBM, XGBoost, Machine Learning**

1. INTRODUCTION

Recommendation systems serve as the forefront of many e-commerce businesses. Their ability to learn user preferences by monitoring user-activity and studying user-behaviour has been exploited in marketing products to interested users as well as targeted cross-selling.

1.1 Ranking in Recommendation Systems

Content based filtering [1] and Collaborative filtering [2] are the 2 most commonly used approaches in the design of a recommendation system. A Content Based filtering approach recommends products to users which are similar to the items that they have indicated their interest in, or have bought in the past. The similarity between products can be measured using various metrics such as the Euclidean distance, cosine similarity matrix, Pearson's coefficient of correlation etc. However a major limitation of this approach of recommending is that the recommendations are limited to those products which are similar to the products the user has already interacted with, in the past. This implies that these methods rely heavily on analyzing user-activity for providing recommendations. As a result these methods are unable to recommend new products to users which is a very important aspect of cross-selling. In order to recommend new products to users it is important to understand user-behaviour in addition user-activity. Collaborative filtering seeks to solve this problem by either building a user-profile or an item-profile. In general the collaborative filtering approach starts off with a common user-item space and matrix factorization is applied to generate either a user-user matrix or an item-item matrix. For instance the user based collaborative filtering algorithm attempts to find similarity between users by looking at the pattern of their interactions across items. It then uses this information to recommend new items not bought by the user but bought by similar users. The Collaborative filtering approach suffers from the cold start problem, wherein the addition of a new user or new item to the dataset can cause the system to provide irrelevant recommendations. This is because the system does not have any history of the user and hence does not know the user behaviour, or because the new item does not have any user interaction on it which is very important in item based collaborative filtering.

In addition to providing product recommendations to users a good Recommender System also needs to rank the products based on some scoring criterion. The similarity (or correlation) score calculated either during content based filtering or collaborative filtering may not always be the best indicator of the fact that the user will buy the item.

For instance, an e-commerce website selling movies, needs to provide a ranked list of movie recommendations to its customers such that the movie having the highest buy probability (probability that the movie shown will be bought by customer) is displayed on the top. While providing the movie recommendations, the recommender system may either use a collaborative or a content based filtering approach to analyze user preferences. However when displaying the recommendations to the users, the system needs to rank these recommendations. Different features may be relevant for the task of recommending and for that of ranking. For example when considering similarity between new movies and the movies previously bought by the customer, features such as genre, production studio, ratings etc may be more relevant. Therefore movies having high degree of similarity on these features are most likely to receive a higher correlation score and therefore would be the top recommendations provided by a content based or collaborative filtering strategy. But when we consider the buy probability of these movies, other features such as price, release date etc may be more important indicators. Intuitively it can be easily understood that if a movie is new or if the

movie price is low there is a higher probability that a customer would buy the movie Therefore although the movies recommended by the system had a high correlation with the user-preferences they may not eventually be bought by the customer. Thus a naive approach of ranking movie recommendations based on their similarity score as determined by the similarity metric may not necessarily be the most optimal. Therefore there is a need to develop a ranking function which can learn to rank recommendations.

1.2 Literature Review

A study-of the state of the art reveals the existence of several approaches to learning the ranking function. In the Pointwise (or item-wise) learning [3] approach a single record is looked at a time in the loss function. In the Pairwise learning [4] approach a pair of records are looked at a time in the loss function and the goal is to come up with an optimal ordering for the pairs that minimizes the number of inversions in ranking. Listwise learning[5] approaches look at the entire list of records at a time and try to come up with an optimal ordering for it. As such several 'Learning to Rank' (LTR) neural network models have been developed to exploit one of the above 3 methods for learning. For instance RankNet [6] uses neural nets to minimize the number of inversions (incorrect orderings) in ranking. Specifically RankNet attempts to optimize its cost function using Stocashtic Gradient Descent (SGD). LamdaRank[7] optimizes RankNet training using only the gradients associated with the cost instead of the costs and performs better than RankNet in terms of both speed and accuracy. LamdaMART[8-9] uses gradient boosted trees for optimizing the cost function derived from LamdaRank and is found to perform better than both LamdaRank and RankNet. Several other specialized models also exist but a detailed study of each of them is out of scope of this work.

2. LEARNING TO RANK RECOMMENDATIONS

In this section we analyze popular machine learning methods that have traditionally been used for classification but can also serve as powerful alternatives to the specialized neural network based LTR methods. In this work we specifically focus on the Pointwise learning approach for learning the ranking function.

2.1 Dataset

The MovieLens 100K Dataset[10] is a popular open-source dataset that has been widely used as a benchmark dataset in the development of several recommender systems. It contains 100,000 ratings (ranging from 1 to 5) given by 1000 users on 1700 movies. Apart from this the Dataset also provides information about movie-specific features such as genre(18 classes), release date etc. This Dataset however misses some features of user-interaction necessary for learning a ranking function. Therefore we simulate these synthetic features for the purpose of our experiment and use these features to generate synthetic user-interaction data. These simulations are designed to replicate a real world scenario wherein users interact with the product recommendations provided by an e-commerce website (in our case movies).

2.2 Data Preparation

Apart from the movie-specific features provided in the dataset for each movie such as genre, release date, url etc we append 2 additional features to each movie namely 'average_rating' and 'num_of_ratings'. These are obtained by grouping the ratings data by 'movie_id' and taking the mean and count of the ratings for each group respectively. The cost of a movie is an important factor that customers consider before buying a movie. The dataset however does not provide a 'movie_price' feature. Therefore we simulate a synthetic feature 'movie_price' for each movie. Intuitively one can say that the price of a movie will be relative to the age of a movie and its 'average_rating'. Therefore we calculate the relative 'movie_price' of a movie by calculating the 'normalized_rating' (relative rating) and 'normalized_age'(relative age) as shown in equation 1

$$normalized\_age = \frac{(most\_recent\_date - movie\_release\_date)}{(most\_recent\_date - oldest\_date)}$$

$$normalized\_rating = \frac{(5 - movie\_average\_rating)}{(5 - 1)}$$

$$movie\_price = (1 - normalized\_rating) * (1 - normalized\_age) * 10$$

Eq -1: Formula to calculate 'movie_price'

Before simulating the synthetic user-interaction data we simulate another synthetic feature 'buy_prob' i.e the probability that a movie recommended will be bought by the user. Intuitively we can consider the 'buy_prob' of a movie to be linearly dependent on the 'movie_price'. We calculate the 'buy_prob' of a movie as shown in the equation 2

$$buy\_prob = 1 - movie\_price * 0.1$$

Eq -2: Formula to calculate 'buy_prob'

Thus according to the equation the movie with the least price has the highest probability of being bought by the customer. The histogram distribution of all the features is illustrated in Fig 1

Fig 1: Histogram distribution of features

We use the 'buy_prob' feature to simulate the synthetic user-interaction data. Specifically for the Pointwise learning approach we are interested in user-interaction data of the type as elucidated in Table 1.

CustomerID	MovieID	Buy	outcome
customer_1	movie_1	0	0
customer_1	movie_2	0	0
customer_1	movie_3	1	1
customer_2	movie_2	0	0
customer_2	movie_3	1	1

Table 1 User-Interaction Data (Pointwise)

This data can be interpreted as customer_1 was recommended the movies, movie_1, movie_2 and movie_3 and decided to buy movie_3. Similarly customer_2 was recommended the movies movie_2 & movie_3 and decided to buy movie_3. As this is a Pointwise learning approach the outcome is a replica of whether the customer decided to buy a recommended movie or not.

In order to simulate these user interactions we use the Pointwise event generator pseudo code as shown in Fig 2. We simulate these events for 1000 users considering that each user interacted with 20 random movies(or recommended via collaborative/content based filtering) on an average.

```
set customers to contain the list of customerIds
for all customerIds x in customers:
        movie_recommended <- getRecommendedMovies(customerId)
    for all movieIds y in movies_recommended:
        if (numpy.random.binomial(1, buy_prob[movieId.index]) == 1):
                user_interaction_data(x, y) <- 1
        else
                user_interaction_data(x, y) <- 0
```

Fig -2: Pseudo code for Pointwise user event generation

The user interaction event distribution as a function of 'movie_price' that is obtained as a result of using the above pseudo code for simulation is illustrated in Fig 3. From the figure it is clear that the number of positive events (buy) are more when the price is less and the number of negative events (not buy) are more when the price is higher.

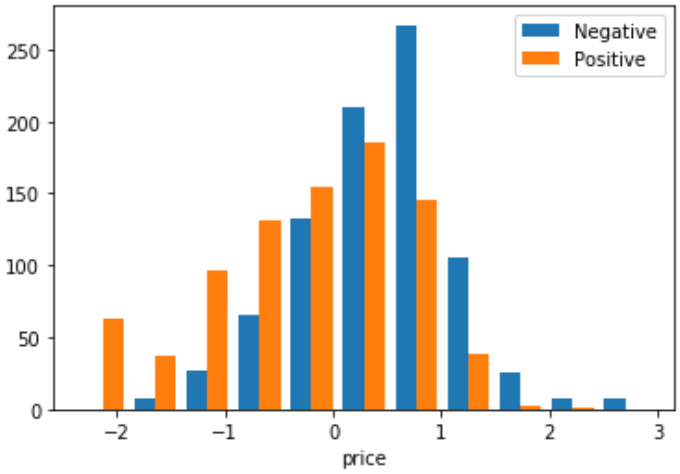

Fig -3: User-interaction event distribution simulation

The getRecommendedMovies function(Fig 2) returns a list of recommended movies for a customer using either one of collaborative or content based filtering strategies. Due to the availability of abundance of literature covering these approaches on MovieLens 100K we choose to omit the implementation of this function in our work.

2.3 Ranking using Machine Learning

The prepared data is split into training and testing splits. 80% of the data is used for training and 20% of the data is used for testing. We feed the input features and the outcome variable into various machine learning models and evaluate them via the precision, recall and accuracy metric. We obtain the optimal hyper-parameters for each model using the grid search technique [11]. In order to evaluate the ranking performed by the machine learning models we use the popular Normalized Discounted Cumulative Gain (NDCG)

score [12] metric by taking the 'buy_prob' as the ground truth measure of relevance.

2.4 Results and Analysis

Illustrated in Table 2 is the performance of various machine learning models using the pointwise approach for learning the ranking function. The enhanced Gradient Boosting methods namely Adaptive Boosting (AdaBoost)[13], Light Gradient Boosting Machine (Light GBM)[14] and eXtreme Gradient Boosting (XGBoost) [15] perform considerably better than other methods in ranking as indicated by their high NDCG scores. This can be attributed to their ability to perform optimization in the function space rather than the parameter space. That is the boosted trees are obtained by optimizing a custom objective function (such as NDCG) rather than a loss function which offers less control. It is also observed that logistic regression is a better candidate algorithm for ranking than an ensemble learning method such as random forest or a naive neural network such as the Multi-layer Perceptron.

Model Name	Train Precision	Train Recall	Train Accuracy	Test Precision	Test Recall	Test Accuracy	NDCG score
Naïve Bayes	0.84845	0.17635	0.52615	0.82758	0.18570	0.53103	0.93080
Support Vector Classifier	0.65799	0.75702	0.64733	0.65365	0.76194	0.64714	0.96391
Decision Tree Classifier	0.73643	0.70193	0.69563	0.65995	0.62630	0.61711	0.97300
Random Forest Classifier	0.71728	0.73275	0.69187	0.64734	0.67091	0.61811	0.97323
Multi-layer Perceptron Classifier	0.67849	0.71672	0.65484	0.66866	0.71096	0.64739	0.98642
Logistic Regression	0.66597	0.726656	0.646584	0.66625	0.73782	0.65265	0.98777
Gradient Boosting Classifier	0.67542	0.72880	0.65565	0.66223	0.72553	0.64564	0.98795
AdaBoost Classifier	0.65828	0.75714	0.64764	0.64916	0.75967	0.64214	0.98797
Light GBM Classifier	**0.67050**	**0.74302**	**0.65528**	**0.66056**	**0.73873**	**0.64764**	**0.98801**
XGBoost Classifier	**0.70985**	**0.74415**	**0.68968**	**0.65704**	**0.69412**	**0.63263**	**0.98803**

Table 2 Performance of models using Pointwise learning Approach

3. CONCLUSIONS

A good recommender system must be capable of not only providing recommendations but also ranking them. The relevant features to be considered while recommending and ranking may be different. In this work, the user-interactions on Movie-Lens 100K Dataset were simulated in order to replicate the real-world interactions of users with an e-commerce website. A pointwise approach was used for learning the ranking function. The performance of several prominent machine learning models was documented. It was found that Gradient Boosting methods perform considerably better than the other methods in the learning to rank task. The Light GBM classifier model with an NDCG score of 0.98801 and XGBoost Classifier model with an NDCG score of 0.98803 were found to outperform all other models in the ranking task. In the future we seek to extend this work by adopting Pairwise and Listwise approaches to learning which are generally considered to be more optimal.

ACKNOWLEDGEMENT

We would like to thank RV College of Engineering for actively supporting this work

REFERENCES

[1] V.M. Robin, and V.S. Maarten. 'Using Content-Based Filtering for Recommendation', NetlinQ Group, Gerard Brandtstraat, Amsterdam, The Netherlands, pp. 26-28. 1999.

[2] Zhang, R., Liu, Q., Chun-Gui, Wei, J.-X., & Huiyi-Ma. (2014). 'Collaborative Filtering for Recommender Systems'. 2014 Second International Conference on Advanced Cloud and Big Data. doi:10.1109/cbd.2014.47

[3] Chen Wen-Hao, Hsu Chin-Chi, Lai Yi-An, Liu Vincent, Yeh Mi-Yen, Lin Shou-De(2019),'Attribute-Aware Recommender System Based on Collaborative Filtering: Survey and Classification' ,Frontiers in Big Data Vol 2, doi:10.3389/fdata.2019.00049

[4] Sharma, Amit & Yan, Baoshi. (2013). Pairwise learning in recommendation: Experiments with community recommendation on linkedin. RecSys 2013 - Proceedings of the 7th ACM Conference on Recommender Systems. 193-200. 10.1145/2507157.2507175

[5] Shi, Yue & Larson, Martha & Hanjalic, Alan. (2010). List-wise learning to rank with matrix factorization for collaborative filtering. 269-272. 10.1145/1864708.1864764.

[6] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. Proceedings of the 22nd International Conference on Machine Learning - ICML '05. doi:10.1145/1102351.1102363

[7] Christopher J.C. Burges, Robert Ragno, and Quoc Viet Le. 'Learning to rank with nonsmooth cost functions'. Proceedings of the NIPS, 2006

[8] Christopher J C Burges, Krysta M Svore, Paul N. Bennett, Andrzej Pastusiak, and Qiang Wu. Learning to rank using an ensemble of lambda-gradient models. Journal of Machine Learning Research (JMLR) 14:25–35, 2011.

[9] Christopher J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. 2010

[10] Harper F. Maxwell and Konstan Joseph A. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TIIS). 2015 Dec; 5(4). Available from http://dx.doi.org/10.1145/2827872

[11] B. H. Shekar and G. Dagnew, "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data," 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), Gangtok, India, 2019, pp. 1-8, doi: 10.1109/ICACCP.2019.8882943.

[12] Wang, Yining & Wang, Liwei & Li, Yuanzhi & He, Di & Liu, Tie-Yan & Chen, Wei. (2013). A Theoretical Analysis of NDCG Type Ranking Measures. Journal of Machine Learning Research. 30.

[13] Tharwat, Alaa. (2018). AdaBoost classifier: an overview. 10.13140/RG.2.2.19929.01122.

[14] Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye and Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." NIPS (2017).

[15] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACMSIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '16. New York,New York, USA: ACM Press; 2016. pp. 785–794

BIOGRAPHIES

Shubham phal is a student pursuing his bachelors in Computer Science and Engineering at R.V College of Engineering (RVCE), Bangalore, India. His research interests include Big Data Analytics and Machine learning. He has worked on various projects in the Machine Learning and Deep learning space and has published several research papers in international conferences of ACM and IEEE. He is currently an intern at Morgan Stanley India

Prof. Smriti Srivastava is Masters in Computer Science and Engineering with a wide experience of 10 years in teaching. Working currently at R.V.C.E as Assistant Professor. Areas of interest includes wireless networks and machine learning.