International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Improving Stability, Smoothing and Diversifying of Recommender Systems

Sagar Sontakke[1], Pratibha Chavan[2]

[1] M.E. Department of I.T., R.M.D. Sinhgad School of Engineering, Savitribai Phule Pune University,Pune, Maharashtra,India

[2]Assistant Professor, Department of I.T., R.M.D. Sinhgad School of Engineering, Savitribai Phule Pune University, Pune, Maharashtra, India

Abstract: ***Recommender systems are used extensively now-a-days for various web-sites such as for providing products suggestions based on customers' purchase history and searched product keywords. Current recommendation system approaches lack of a high degree of stability. Diversification of prediction is also important feature of recommender system. Having displayed same set of results every time may increase lack of trust in recommendation systems. In this paper, our focus is on stability of different recommender system approaches and providing diversity in results. We will focus on different recommender system approaches, methods provided to improve the stability and approaches in increasing diversity of recommender system.***

Keywords: Recommendation Systems, Stability, Diversity, Iterative Smoothing, Collaborative Filtering, Bagging

1. Introduction

Recommender systems are type of information filtering system that predicts results based on ranking, rating or preference that a user would give to an item. Recommender systems are very common now-a-days and used in variety of applications. Various applications or websites related to movies, music, news, books, scholarly articles, search queries, social tags, online dating or matchmaking for marriages and products in retail store etc. are using recommender systems extensively. Recommender systems are great alternative to traditional search algorithm as these systems take into consideration various other factors like user history, ratings, rankings etc. apart from just indexing.

There are basically following three ways of producing a list of recommendation by system:

- Collaborative filtering
- Content-based filtering
- Hybrid filtering

Collaborative filtering uses a user's behavior history such as items previously purchased or ratings given to those items etc. and similar decisions made by other users. Content based filtering focuses on characteristics of an item to recommend additional items with similar properties. Hybrid approaches are the combination of content based and collaborative filtering.

Recommender Systems stability is defined as the ability of the system to be consistent in predictions made on the same items by the same algorithm, when any new incoming ratings are in complete agreement to system's prior estimations. Stability is a measure of the level of internal consistency among predictions made by a given recommendation algorithm. Putting in simple form, stability is a measure of consistency of recommender system's predictions. Consistency is one of the important characteristics of recommender systems. Inconsistency in recommender systems may have a negative impact on users' trust of the recommender system and can reduce users' acceptance of future suggestions by the system. This makes Stability as the most important and desired property.

Diversity is another important property that recommender system should consider. Due to overload of information, it is pretty much possible to have redundant data in system predictions. Items with similar properties are less useful in users' point of view. To reduce this redundancy, it is important to consider diversity of the system.

In this paper, we are going to survey different approaches available to improve stability of the recommender systems and we are also going to see various ways to increase diversity in the system to reduce the redundancy.

2. Literature Survey

In [1] authors provides an extensive empirical evaluation of stability for six popular recommendation algorithms on four real world datasets. This paper provides experiments, results of which suggest that stability performance of individual recommendation algorithm is consistent across a variety of datasets and settings.In this paper, it is suggested that model-based recommendation algorithm consistently demonstrate higher stability than neighborhood-based collaborative filtering technique. Analysis of important factors such as sparsity of original rating data, normalization of input data, the number of new incoming ratings, the distribution of incoming ratings, the distribution of evaluation data, etc. to report their impact on recommendation stability is performed in this paper.

In [2] a survey of neighborhood-based methods for the item recommendation problem is presented. Nearest-neighbors is most popular approach among collaborative recommendation approaches. It is the simple and efficient approach. The main advantages and their important characteristics are described by the authors.While implementing neighborhood-based systems, the required essential decisions and practical information on how to make such decisions are suggested in this document. Large recommendation systems have issue of

sparsity and limited coverage. This problem is also discussed and solutions are provided in this paper.

In [3] various efficient techniques available among collaborative filtering are discussed. A framework is suggested which will combine these techniques to obtain good predictions. The methods provided in this paper are compared against Netflix Prize dataset It is proposed that the set of predictors with addition of biases to the regularized SVD, post processing SVD with kernel ridge regression is extended using a separate linear model for each movie and using methods similar to SVD but with fewer parameters. All predictors and selected 2-way interactions between them are combined using linear regression on a holdout set.

In [4] the comparison is made on recommenders based on a set of properties that are relevant to application. A few algorithms are compared using some evaluation metric instead of absolute benchmarking of algorithms. The experimental settings are suggested to make decisions between algorithms. Three experiments namely Offline settings where there is no interaction from user, a small group of subjects experiment and provides experience and large scale online experiment where real user populations interact with the systems are reviewed. Different questions to ask, protocols to use in experimentation etc. are suggested. A largest of properties is reviewed and systems are evaluated in the given relevant properties. A large set of evaluation metrics in the context of the property that is evaluated, is also surveyed.

In [5], the computational complexity of user-based collaborative filtering is discussed. Model-based recommendation techniques developed to address this problem analyze the user-item matrix to discover relations between the different items and use these relations to compute the list of recommendations. One of these model-based recommendation algorithm is presented in this paper. This algorithm determines the similarities between the various items and then uses them to identify the set of items to be recommended. The method used to compute the similarity between the items, and the method used to combine these similarities in order to compute the similarity between a basket of items and a candidate recommender item are two key steps in this type of algorithm. It is experimentally presented that these item-based algorithms are two times faster than traditional user-neighborhood based recommender systems. It is also proved by experiment that the algorithm provides better quality recommendations.

In [6] collaborative filtering for web service recommendation system is discussed. Missing QoS (Quality-of-Service) values of web services are considered and also performance improvements are suggested. While measuring the similarities between users and between services, existing QoS prediction methods do not consider personalized influence of users and services. Web service QoS factors like response time and throughput depends on the location of the web services and user which is also ignored by the existing web services recommendation systems. In this paper, a location-aware personalized CF method is used for web service recommendation. It utilizes both locations of users and web services when selecting similar neighbors for the target user or service. It provides enhanced measurement for experiments using real world datasets. It improves the QoS prediction accurately and efficiently compared to traditional CF-based methods.

In [7], various collaborative filtering approaches in web services selection and recommendationdo not consider the difference between web service recommendation and product recommendation used in e-commerce sites. A hybrid collaborative filtering approach, region KNN is discussed in this paper. It is designed for large scale web service recommendation. It uses the characteristics of QoS by building an efficient region model. Memory-based collaborative filtering algorithm used in this method provides a quicker recommendation. Region KNN is highly scalable, accurate algorithm compared to other traditional CF algorithms.

In [8], voting classification algorithms such as Bagging and AdaBoost are reviewed and experimented with large empirical study comparing several variants in conjunction with a decision tree inducer (three variants) and a Naive-Bayes inducer. These algorithms use perturbation, reweighting, and combination techniques. These techniques affect classification error. It is discussed that why and when these algorithms affect classification error. It is determined that Bagging reduced variance of unstable methods, while boosting methods reduced both the bias and variance of unstable methods but increased the variance for Naive-Bayes was very stable. It showcases fundamental differences between AdaBoost and Arc-x4. Some of voting variants are introduced in this paper such as pruning versus no pruning, use of probabilistic estimates, weight perturbations (Wagging), and back fitting of data. It is proved by experiment that Bagging improves when probabilistic estimates in conjunction with no-pruning are used, as well as when the data was back fit. Tree sizes are measured and it is shown that there is a positive correlation between the increase in the average tree size in AdaBoost trials and its success in reducing the error. Voting methods and non-voting methods are compared which indicated that voting methods lead to large and significant reductions in the mean-squared errors. Practical problems are also discussed.

In [9], a scalable, general-purpose iterative smoothing algorithm is proposed in conjunction with different traditional recommendation algorithms. It improves the stability. The experiments with real world rating data proves the proposed approach more stable compared to the original algorithms. By improving stability, it does not sacrifice the predictive accuracy but instead at the same time improved it.

In [10], a software agents Syskill&Webert are described. These are used to rate pages on World Wide Web. It provides the pages those might interest a user. Thus it increases the classification accuracy without seeing many rated pages. Using 'conjugate priors' from Bayesian statistics for probability revision, the user profile is revised when more training data is available. This approach is compared to learning algorithms that do not make use of such background knowledge, and it is found that a user defined profile can significantly increase the classification accuracy.

3. System Design And Algorithms

In this system, we are going to improve stability of the recommender system by using iterative smoothing and bagging algorithm and iterative smoothing algorithm provided in [11]. We are also going to improve diversity in the prediction by implementing the diversity algorithm provided in [12].

The system consists of different modules. User will search the product, the product search is queried in database which is either stored locally or available on internet. The stability algorithm will be applied on these searched queried results. The diversity ranking algorithm is used to generate diversified results. History available on product and User preferences from profile & user history will be used for improving stability and diversification.

The project run steps of the proposed work is show below:

- First user have to login to get user's data such as his ratings
- User queries the movies based on search criteria
- System uses Jaccard similarity coefficient to calculate similarity.
- Based on similarity, clusters are created.
- Based on user's query, the cluster is selected and predictions are made
- User accepts the prediction of the system or give the new ratings for the movies
- System then recalculates the prediction.
- During re-prediction, iterative smoothing or bagging algorithm is used and movie predictions are displayed
- Finally number of recommended movies, their ratings is displayed.
- In a separate section, diversified movies are displayed from the different clusters.

The following figure 1 displays the system overview diagram.

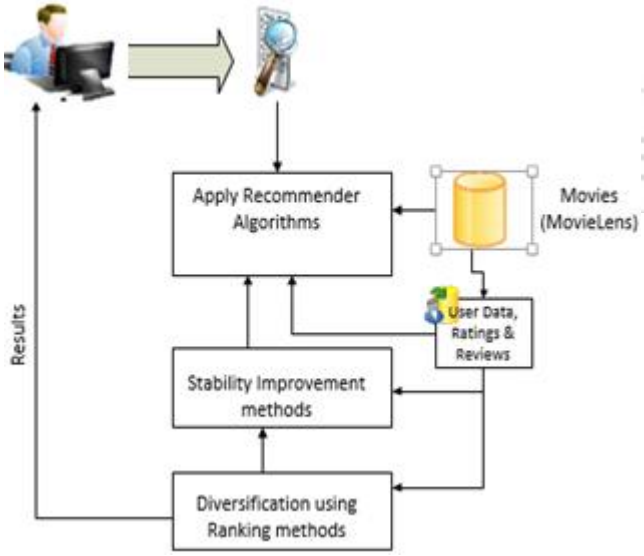

Figure 1: System Overview

Iterative Smoothing Algorithm:

Following steps are for Iterative smoothing algorithm that we are going to use for recalculation of predictions when user either accepts the system predictions or enters the new predictions.

Iterative Smoothing Algorithm:

Inputs: known ratings data D
number of iterations K
recommendation algorithm T

Process:

1. Build model f_0 on known ratings D using some standard recommendation algorithm T, i.e., $f_0 \leftarrow T(D)$
2. Apply model f_0 to compute predictions P_0 for unknown ratings $S \setminus D$, i.e., $P_0(u,i) = f_0(u,i)$ for $(u,i) \in S \setminus D$
3. For each iteration $k \in \{1, \ldots, K\}$
   For each unknown rating pair $(u, i) \in S \setminus D$
   a. Construct dataset $D_{k,u,i}$ by including all known ratings D and all predicted ratings P_{k-1} from the previous iteration, except for rating $P_{k-1}(u,i)$, i.e.,
   $$D_{k,u,i} = D \cup P_{k-1} \setminus \{P_{k-1}(u,i)\}$$
   b. Build model $f_{k,u,i}$ on dataset $D_{k,u,i}$ using T, i.e.,
   $$f_{k,u,i} \leftarrow T(D_{k,u,i})$$
   c. Make prediction on (u, i) and store in P_k, i.e.,
   $$P_k(u, i) = f_{k,u,i}(u, i)$$
4. Output predictions made in the final iteration P_K

Output: P_K

Figure 2: Iterative Smoothing Algorithm

4. Results Analysis

To get the desired results, we are using the database available to us i.e. MovieLens 100k. This database has 10000 ratings given to various movies by 980 users. To improve the stability we have first used the bagging algorithm along with the iterative smoothing algorithm to get the desired improvement to the recommendation.

As discussed in [1], we first calculated the entire User-Item Space. This entire User-Item space value is calculated as the product of set of users and set of items. The entire user item space is denoted by S = U * I. After displaying the entire user-item space values, we have then provided the recommendation of movies based on Item similarity based prediction. We then displayed the known ratings of the movies.

One of the widely used predictive accuracy metrics for the recommendation system is root mean square error (RMSE). RMSE is squares each individual error (deviation) before averaging them, thereby assigning larger penalty for bigger errors. We have used the RMSE to calculate the accuracy of our recommendation system. We plotted the graph based on the values received.

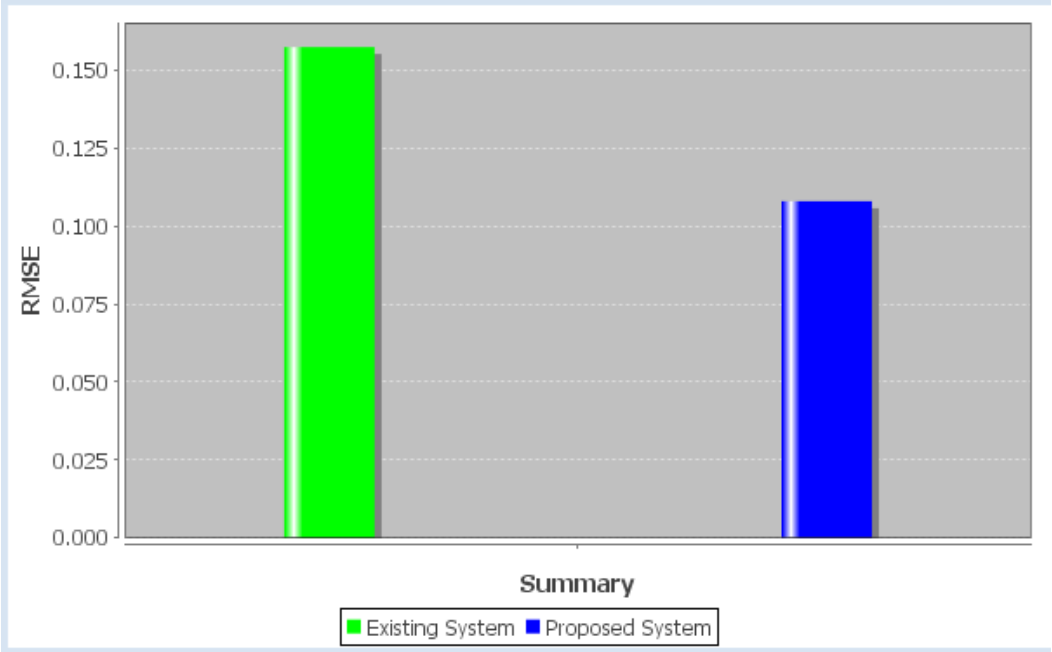

Figure 3: RMSE Comparison

Recommendation stability measures the inherent consistency among the different predictions made by the system. Specifically, the stability of a recommender system is defined as the extent to which the system's predictions for the same items stay the same/similar (i.e., are stable), when any and all new incoming ratings submitted to the recommender system are in complete agreement with system's prior predictions. Intuitively, (in)stability of a recommendation algorithm represents the degree of (in)consistency between the different predictions made by the algorithm. That is, two predictions of the same recommender system can be viewed as inconsistent with each other, if adding one of them to the training data for the system changes the other prediction. Stability is then measured by comparing the two sets of predictions to compute their root mean squared difference, which is called root mean square shift (RMSS). For calculating RMSS, we considered the results that we received after applying Iterative smoothing algorithm and the known ratings available in the MovieLens dataset and the formula given in [1]. The total number of users considered are around 5000 with number of items considered are 10, the value of the noise is 40 and then we calculated the unknown ratings using Iterative smoothing algorithm. Based on these results we plotted the graph.

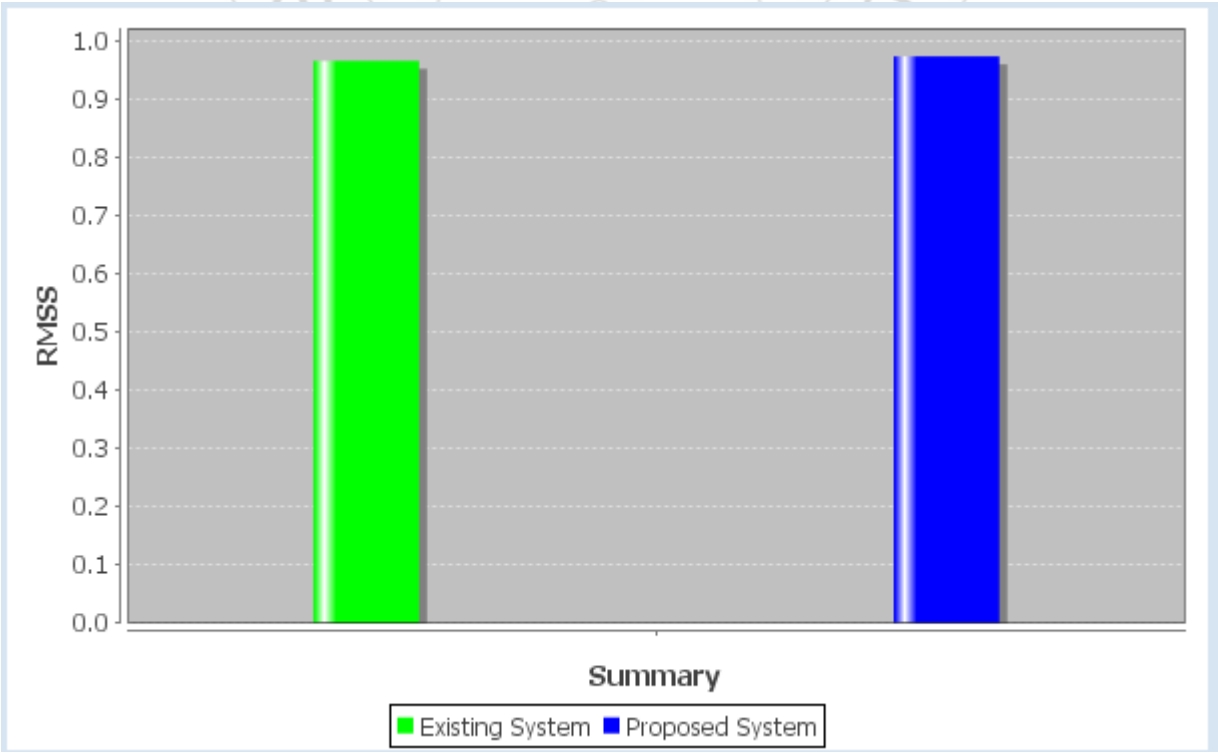

Figure 4: RMSS Comparison after Iterative Smoothing

In pattern recognition and information retrieval with binary classification, precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance.

In an information retrieval scenario, the instances are documents and the task is to return a set of relevant documents given a search term; or equivalently, to assign each document to one of two categories, "relevant" and "not relevant". In this case, the "relevant" documents are simply those that belong to the "relevant" category. Recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents, while precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search.

In a classification task, the precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been).

In information retrieval, a perfect precision score of 1.0 means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved) whereas a perfect recall score of 1.0 means that all relevant documents were retrieved by the search (but says nothing about how many irrelevant documents were also retrieved).

For our system, we have also calculated the precision and recall of our system and plotted the graph with the results. Similarly, we have also finding and displaying diversified results. The following table 1 shows the results found using our approach.

Table 1: Result Table

Parameters	Values
RMSE	0.96220919
RMSS	1.08469
Precision	0.00665
Recall	150.2941

5. Conclusion

Stability and Diversity are important features of recommender systems. It is an important property of recommendation algorithms.Our proposed system will help in improving stability of the recommender systems by applying iterative smoothing algorithm. The diversity of the system is also improved by taking into consideration of User's preferences & quality requirements to generate the results. Our proposed system will thus help in improving stability of the system.

References

[1] G. Adomavicius and J. Zhang, "On the stability of recommendation algorithms," in Proc. ACM Conf. Recommender Syst., 2010
[2] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in Recommender Systems Handbook, F. Ricci et al., Eds. New York, NY, USA: Springer,2011, pp. 107–144
[3] A. Paterek, "Improving regularized singular value decompositionfor collaborative filtering," in Proc. KDD Cup, 2007, pp. 39–42.
[4] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in Recommender Systems Handbook, F. Ricci, et al. Eds. New York, NY, USA: Springer, 2011., pp. 257–294
[5] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," ACM Trans. Inform. Syst., vol. 22, no. 1, pp. 143–177, 2004
[6] Liu, J., Location-Aware and Personalized Recommendation Collaborative Filtering for Web Service, IEEE transactions, May 2015
[7] Region KNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation, IEEE conference 2010
[8] An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants by ERIC BAUER,1998 Kluwer Academic Publishers, Boston
[9] Iterative Smoothing Technique for Improving Stability of Recommender Systems by GediminasAdomavicius
[10] Revising User Profiles The Search for Interesting Web Sites by D Billsus, AAAI.org 1996
[11] Improving Stability of Recommender Systems: A Meta-Algorithmic Approach by GediminasAdomaviciusand Jingjing Zhang, IEEE transactions, 2015
[12] Diversifying Web Service Recommendation Results via Exploring Service Usage History by Guosheng Kang, IEEE transactions, 2015
[13] Using of Jaccard Coefficient for Keywords Similarity by SuphakitNiwattanakul, JatsadaSingthongchai, EkkachaiNaenudorn and SupachanunWanapu , Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013
[14] A Survey on Stability and Diversity of Recommender Systems by Sagar Sontakke & Pratibha Chavan, IJIRCCE, Volume3, Issue 12, December 2015

Author Profile

Sagar Sontakke is a M.E. Student in the Department of Information Technology, R.M.D Sinhgad School of Engineering, Savitribai Phule Pune University, Pune, MH, India.

Prof. Pratibha Chavan is an Asst. Professor in the Department of Information Technology, R.M.D Sinhgad School of Engineering, Savitribai Phule Pune University, Pune, MH, India.