Prediction and Recommendation with Confidence in Collaborative Filtering Recommender Systems

A thesis submitted during 2018 to the University of Hyderabad in partial fulfillment of the award of a Ph.D. degree in the School of Computer and Information Sciences

by

TADIPARTHI V R HIMABINDU

School of Computer and Information Sciences

University of Hyderabad

P.O. Central University, Gachibowli

Hyderabad – 500046

Telangana, India

July-2018

CERTIFICATE

This is to certify that the thesis entitled **"Prediction and Recommendation with Confidence in Collaborative Filtering Recommender Systems"** submitted by **TADIPARTHI V R HIMABINDU** bearing registration number **12MCPC04** in partial fulfillment of the requirements for award of **Doctor of Philosophy** in the School of **Computer and Information Sciences** is a bonafide work carried out by her under my supervision and guidance.

The thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma.

Further, the student has the following publication(s) before submission of the thesis/monograph for adjudication and has produced evidence for the same in the form of acceptance letter or the reprint in the relevant area of her research:

1. Tadiparthi V.R. Himabindu, Vineet Padmanabhan and Arun K. Pujari. "Conformal matrix factorization based recommender system", Information Sciences, pp. 1-23, 2018. In Press.

2. Tadiparthi V. R. Himabindu, Vineet Padmanabhan, Arun K. Pujari, and Abdul Sattar. "Prediction with Confidence in Item Based Collaborative Filtering". Pacific Rim International Conference on Artificial Intelligence, pp. 125-138, Springer, 2016.

Further, the student has passed the following courses towards fulfillment of coursework requirement for Ph.D:

Course Code	Name	Credits	Pass/Fail
CS 801	Data Structure and Algorithms	4	Pass
CS 802	Operating System and Programming	4	Pass
CS 826	Data Mining	4	Pass
CS 827	Secure Computing	4	Pass

Prof. Vineet C.P. Nair
Supervisor

Prof. K. Narayana Murthy
Dean of School

DECLARATION

I, **TADIPARTHI V R HIMABINDU**, hereby declare that this thesis entitled **"Prediction and Recommendation with Confidence in Collaborative Filtering Recommender Systems"** submitted by me under the guidance and supervision of **Prof. Vineet C. P. Nair** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma.

Date : **Name: TADIPARTHI V R HIMABINDU**

Signature of the Student:

Reg. No.: 12MCPC04

// Countersigned//

Signature of the Supervisor(s):

Abstract

Now a days, recommender systems are widely being used by many websites to deal with the information overload problem by helping users in finding their items of interest. Typically, recommender systems focus on two main tasks: Prediction and Recommendation. Many algorithms have been developed to implement the recommender systems. Among them, collaborative filtering algorithms are very popular and most widely used techniques both in academic research and commercial applications. Classic approaches to collaborative filtering methods include neighborhood-based or memory-based methods and the more recent approaches focus on model-based methods specifically matrix factorization techniques.

Much research in recommender systems focuses on improving the accuracy of the recommendation algorithms. Besides accuracy, several other evaluation metrics such as coverage, novelty, unexpectedness, serendipity, diversity and confidence measure the performance of recommender systems from different perspectives. Among them confidence is very important as it enables the users to make more effective decisions by measuring uncertainty involved in predictions or recommendations. Providing a confidence display alongside a prediction can also enhance the user interaction with the system which further increases user satisfaction. Different approaches have been proposed to estimate the confidence of predictions in recommender systems. Some of these approaches are non-personalized and some of them are applicable only to specific algorithms and are not generalized. Furthermore, none of the these algorithms provide guarantees on the error rate of the predictions, where error rate is the probability of excluding the correct class label.

This dissertation focuses on estimating the confidence of predictions and recommendation sets generated by various recommendation algorithms with guaranteed error rate. In our work, we focus on collaborative filtering recommender systems, specifically, neighborhood-based methods and matrix factorization as they depend only on rating information while ignoring the specific information about users and items. The performance of neighborhood-based algorithms depend on similarity measures employed, prediction techniques used, data sets and type of feedback used in their evaluation. Therefore, before discussing the proposed confidence estimation methods for neighborhood-based algorithms, we present an empirical study on these algorithms with different similarity measures, prediction techniques and different data sets with different kinds of feedback. We use appropriate evaluation metrics to measure the performance of these algorithms and significance tests to determine whether the performance differences between different algorithms are statistically significant or not.

To estimate the confidence with guaranteed error rate, conformal prediction used in machine learning is introduced to recommender systems. Unlike the confidence estimation methods in recommender systems, conformal prediction is a generalizable framework which can be applied on top of any recommendation algorithm and can be used to produce personalized predictions with a bound on the probability of error. As the characteristics of data sets used in machine learning differs a lot from that of recommender systems, the application of conformal prediction to recommender systems is very different from its application to machine learning algorithms. As part of this thesis, we discuss the differences between machine learning algorithms and recommendation algorithms while applying the conformal prediction framework. We also show how conformal prediction can be applied to prediction task and recommendation task in recommender systems. The results are validated on movies data sets and experimental results show that the conformal prediction is suitable to binary data sets for prediction task and unary data sets for recommendation task.

To My Family and Teachers

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my family members for their true, unconditional love and continuous moral support.

I would like to thank my supervisor Dr. Vineet C. P. Nair for his invaluable guidance, endless patience and support during all these years and giving me the freedom to pursue my research in my own way. My special thanks to Dr. Arun K Pujari for introducing me to conformal prediction and for his thoughtful comments and detailed suggestions on my research work. I thank both of them for their encouragement during the tough times of my research.

My gratitude also goes to my DRC members Dr. Chakravarthy Bhagvati and Dr. S.K. Udgata for their helpful suggestions. I especially thank Dr. Chakravarthy Bhagvati for the patience he has shown and the time and effort he has spent in listening to my research work and for his constructive remarks. I wish to thank Dr. S.K. Udgata for his support and guidance.

I thank Dr. S. Durga Bhavani for being patient in listening to all my problems and helping me to find the solutions. I would also like to thank Dr. Atul Negi and Dr. Salman Abdul Moiz for their insightful comments and suggestions.

I would also like to extend my gratitude to Dr. Appa Rao, Vice Chancellor for providing necessary facilities to carry out my research work. I thank Mr. Venkateswara Rao and Mr. Vikas Kumar for many useful discussions. I sincerely thank all my teachers for their constant support, suggestions, motivation and encouragement.

TADIPARTHI V R HIMABINDU

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the increasing amount of data on the World Wide Web, the problem of finding useful information has become more and more difficult. To deal with this information overload problem, recommender systems have been developed. Recommender systems (RS) help users make more effective decisions by recommending the items of their interest based on users' past behavior. Recommender sytems are now being used by many websites to increase their revenue while maintaining the user satisfaction by producing more effective recommendations. Collaborative filtering (CF) recommendation algorithms are the most widely adopted and successful techniques used to built recommender systems both in academic research and commercial applications. Classic methods of collaborative filtering include *neighborhood-based* methods and recent methods are revolving around *model-based* approaches especially matrix factorization techniques. In neighborhood-based approaches, prediction and recommendations can be done either by computing the similarities between users (user-based collaborative filtering (UBCF) [64]) or similarities between items (item-based collaborative filtering (IBCF) [66]). User-based approach identifies users whose tastes are similar to that of the active user and recommends items they have liked. On the other hand, item-based approach uses item-item similarity to make the predictions. On the other hand, model-based approaches use mathematical models to make the predictions. Particularly, matrix factorization techniques [34] have recently gained much popularity because of their scalability, accuracy and also their successful application in the Netflix Prize competition.

Over the last two decades, the research in recommender systems has been focusing on developing new algorithms and enhancing the existing algorithms to improve the performance. In order to choose the best algorithm among these available algorithms one needs to evaluate and compare these algorithms. Several measures have been proposed to evaluate recommender systems from different perspectives. Still, evaluating recommender systems is inherently difficult due to the following reasons [25]:

- Which measure should be used to determine the quality of recommendation algorithms?

  Majority of the papers in RS literature focus on accuracy of recommender systems. But, most accurate predictions are not always the most useful ones to the users. For example, though RS predicts correct ratings for items, it may not be useful to the user if the user is not interested in ratings and expects only those items which he likes. For example, suppose our task is to recommend top 10 items which are liked by the user. In this case, it is not required to predict the correct ratings for all these 10 items as long as these items are the most relevant and useful among the available items. Similarly, the recommendations may not be useful to the user, if the recommender system recommends the most popular items. Because, the user can find those items by him/herself. Likewise, if the recommended items are very much similar to those which are already consumed by the user, the user will get bored of similar recommendations. Similarly, recommending similar items for instance, similar places in tourism recommendation will not be helpful to the user as the recommendation list should be diverse in tourism domain. Therefore, the measure that should be used to determine the quality of a recommender system depends on the users' task and their requirements and the recommendation domain.

- The performance of RS varies depending on the characteristics of the data set.

  The performance of RS is effected by many characteristics of the data set like number of users, number of items and sparsity of the data set. An algorithm which performs well on a data set with number of users greater than the number of items may not perform well on a data set with number of items greater than the number of users. Similarly, an algorithm which performs well on dense data set may perform worse on sparse data set.

- The goal for which the RS is evaluated.

  The goal for which the RS is designed will also play a significant role in evaluation. For example, improving user satisfaction and increasing revenue are two important goals of a recommender system. These goals require completely different approaches and different measures for evaluation as compared to the traditional ones.

A typical recommender system can perform two tasks: *prediction* and *recommendation*. In prediction task, the algorithm predicts how the user will rate a particular item. In recommendation task, the algorithm presents a list of items that the user likes. The recommendation can be done in two ways: 1. Prediction-based recommendation: Here, like in prediction task, the algorithm predicts rating for every unused item and recommends items with highest ratings. 2. Score-based recommendation: For every unused item, a score will be calculated which tells the likelihood of this item being recommended. The higher the score, the better to consider it for recommendation. Finally, items with high scores will be recommended. Type of feedback in the data set will also effect the performance of the algorithm. Therefore, choosing the data set with appropriate feed back for the given task is also very important in evaluation. We have mainly two types of feedback available: *explicit* and *implicit*. Again we can divide explicit feedback data set into two types: Numerical feedback where the feedback is in the form of ratings like 1, 2, 3, 4 and 5 and binary feedback which contains information about whether the user likes or dislikes the item instead of having ratings. Implicit feedback data sets which are also called unary data sets contain information about whether the user has used that item or not. But, unary feedback can be explicitly given by the user and binary feedback can be implicitly derived from user's behavior. In this case, unary feedback comes under explicit feedback and binary feedback comes under implicit feedback. Rating prediction and prediction-based recommendation will be generally performed on data sets with explicit feedback and score computation and score-based recommendation on implicit feedback data sets.

Despite having several measures for evaluating RS, accuracy measures are the most sought after evaluation measures. Several accuracy measures have been adapted from statistics and information retrieval to measure the accuracy of recommender systems. Among them, prediction accuracy measures like *MAE* and *RMSE* and recommendation

accuracy measures like *precision* and *recall* are very popular. Choosing the appropriate accuracy measure for the given task as well as the type of feedback in the data set is one of the crucial aspects involved in developing recommender systems.

Though accuracy is very important, it is not the only measure that determines the quality of a recommendation algorithm. Several other measures such as coverage, novelty, unexpectedness, serendipity, diversity and confidence were proposed to evaluate the recommender systems from different perspectives [70]. Among them confidence is very important as it acts as a reliability measure which measures the uncertainty associated with the predictions made by the recommendation algorithm. This in turn enables the users to make more effective decisions about which items to buy, which books to read, which movies to watch etc. Confidence tells us the system's trust in its predictions or recommendations. It helps users to distinguish between confident and inadequate recommendations thus by enabling them to make intelligent choices.

1.1 Motivation

Evaluation measures are used to find how well a given algorithm performs, to compare different recommendation algorithms along with their variants and to select the appropriate algorithm in a given context. There is no single measure to determine the quality of the recommendation algorithms suitable to all contexts. Several evaluation measures have been proposed in the literature to measure the recommender systems' performance. All these different measures assess the performance of recommendation algorithms from different perspectives. We need to choose a suitable evaluation measure for a given context. In addition to this, some of the measures may not be generalized and have their own limitations. Therefore, there is a need to improve the existing measures depending on the user requirements or goals.

Though accuracy measures are the most popular for evaluation in recommender systems, there are many aspects of the recommender systems which are not taken into consideration by accuracy measures. For example, accuracy measures do not measure the percentage of items for which the algorithm can make predictions, how many of the items in the recommendation list are not known to the users, how many of them are unexpected and interested items, how similar the items in the recommendation list are, how much uncertainty is involved in the predictions etc. We can not measure all

these aspects by a single evaluation measure. We need a different evaluation metric to measure each of these aspects. Therefore, besides accuracy, several other measure like coverage, novelty, unexpectedness, serendipity, diversity and confidence were proposed to judge the quality of recommendations from different perspectives [70].

Among them confidence is very important which measures the uncertainty of the predictions or recommendations made by the algorithm. There are several factors such as data, algorithmic parameters and model selection that leads to uncertainty in recommender systems.

1. Data: One of the reasons for uncertainty in recommender systems is sparsity of data. If the data is sparse, we do not have enough information to make predictions or recommendations. It is difficult to make accurate predictions or recommendations for users who have rated very few items. Similarly, the predictions made for the items with many ratings are more accurate than the items with few ratings. Therefore, the more ratings we have in the data set, the more confident the algorithm is in making its recommendations or predictions [49]. Second reason is noisy data. Users will rate same items differently at different times. Therefore user's given rating can be considered as the noisy evidence of the user's true rating [26]. Another reason is data from untrusted users.

2. Change of user preferences over time: In recommender systems, user preferences may vary dynamically. As users tend to explore more and more items, their tastes change over time accordingly [34]. User preferences will also be changed depending on his current situation like mood, location, season, weather condition etc.

3. Algorithmic parameters and Model selection: Choice of different parameter values in the algorithm, suitability of the algorithm for the given data (for example, if there are few neighbors for the given active user or the correlation between the neighbors and the active user is very low), assumptions used in the model and appropriateness of model to the given data are also the reasons which causes uncertainty.

Several approaches have been proposed to measure the uncertainty involved in predictions or recommendations made by the recommendation algorithms. One of the most common methods which can be used to estimate the confidence in machine learning algorithms is probability. For instance, posterior probabilities produced by naive bayes algorithm can be used as confidence values. As most of the recommendation algorithms use machine learning techniques, we can use probability as a confidence

measure in recommender systems. Besides probability, several other confidence estimation approaches based on number of ratings [49], belief distributions [48], rating variance [4], percentage of correctly classified predictions to compare two confidence estimation algorithms [70], binary classification problem with full probability distribution [35] and resampling [47] have been proposed in the recommender systems literature. Some of the them are non-personalized and some of them are applicable only to specific algorithms and are not generalized. Furthermore, none of the above algorithms provide guarantees on the error rate of the predictions, where error rate is the probability of excluding the correct class label and there is no possibility of controlling the erroneous predictions. Conformal prediction (CP) [8, 69, 75] is a framework used to provide confidence values to individual predictions. CP can be generalized to any algorithm and can produce personalized recommendations with guaranteed error rate. Moreover, with CP we can control the number of erroneous predictions by varying the significance level, thus making it suitable to different kinds of applications.

1.2 Thesis Contributions

This dissertation focuses on conformal prediction based confidence estimation for prediction and recommendation tasks in collaborative filtering algorithms. We compare the proposed CP algorithms with their underlying algorithms only. We do not compare our results with other state-of-the-art algorithms, as our aim is not to improve the performance of the algorithm but to associate confidence to the predictions made by the algorithm without compromising on the performance. A summary of the contributions is given below:

Confidence Estimation for Prediction Task: We apply conformal prediction to recommender systems to estimate the confidence of the predictions with an upper bound on the error rate. Conformal prediction is a framework which can be applied on top of any recommendation algorithm. Conformal prediction uses two measures to indicate the quality of predictions: validity and efficiency. Validity refers to the probability of excluding the correct label from the prediction region. Efficiency refers to the tightness of the prediction regions it produces. The narrower (small number of labels) the prediction region the more efficient the conformal predictor is. The regions

produced by any CP algorithm are automatically valid. Underlying algorithm and nonconformity measure (NCM) definition do not effect the validity of CP. They only effect the efficiency of CP. Therefore, producing the tight prediction regions by defining the efficient nonconformity measure suitable to the underlying algorithm is the primary goal of CP.

As part of this work, we define different nonconformity measures based on the underlying algorithm and we find the best nonconformity measure in terms of prediction accuracy and efficiency.

We apply CP on item-based collaborative filtering algorithm and matrix factorization. We do not apply CP to user-based collaborative filtering algorithm as it is very difficult to find the required number of neighbors belonging to each class. This problem arises in user-based algorithm for both numerical and binary data sets. This same problem arises when CP is applied on top of item-based collaborative filtering algorithm on numerical data sets. But for binary data sets, it is not a problem to find the required number of neighbors for each class in item-based algorithm in many cases. Therefore, we have used binary feedback data sets while applying CP on top of item-based collaborative filtering algorithm.

We have shown different ways of applying CP to matrix factorization algorithm and analyzed the results and found that CP algorithm is not producing efficient prediction regions at higher confidence levels on numerical or rating data sets due to very close possible rating values. Therefore, to improve the efficiency we applied CP on binary data sets and got better results compared to the numerical data sets.

Finally, the application of conformal prediction to recommender systems is very different from its application to machine learning as the characteristics of data sets used in machine learning and recommender systems are different. Therefore, we also discussed major differences between machine learning algorithms and recommendation algorithms, how to adapt conformal prediction to different recommendation algorithms and what constitutes an example, features and labels in recommender systems. Details of the proposed algorithms and experimental analysis for item-based and matrix factorization algorithms are discussed in chapters 4 and 5 respectively.

Confidence Estimation for Recommendation Task: Conformal prediction is originally designed to associate confidence values to each prediction. In addition to prediction task, recommendation task is equally important in recommender systems.

Attaching confidence values to a set of recommended items gives the relevance of the recommended items to user tastes. Chapter 6 discusses the following queries regarding the application of conformal prediction in recommender systems: How conformal prediction can be adapted to recommendation task in neighborhood-based algorithms? How error rate is redefined in recommendation scenario? How measures of machine learning can be used to measure validity and efficiency of recommendation lists produced at the given confidence levels? How to compute confidence values for the *top-n* recommendation list for each user? What is the trade-off between precision and number of recommended items at different significance levels? We have also shown the performance differences between two neighborhood-based conformal recommendation algorithms, i.e., *user-based* and *item-based*.

1.3 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 discusses basic concepts of recommender systems, their tasks and various algorithms used to implement recommender systems. It also presents different types of feedback that users provide to give their opinion on different items. It also discusses different approaches and measures used to evaluate recommender systems. In chapter 3, we present an empirical study on neighborhood-based algorithms along with different prediction techniques and similarity measures for task-based evaluation on multiple feedback types. The application of conformal prediction to prediction task in recommender systems is discussed in chapters 4 and 5. In these chapters we present the major differences between the recommendation algorithms and machine learning algorithms as well as the different nonconformity measures suitable to a given recommendation algorithm. Chapter 6 presents the application of conformal prediction to recommendation task and and the adaptation of suitable measures from machine learning to find validity and efficiency of the proposed CP algorithms for recommendation task. Chapter 7 summarizes the contributions and their results along with some directions for future research.

1.4 Publications of the Thesis

1. Tadiparthi V.R. Himabindu and Vineet Padmanabhan and Arun K. Pujari. ”Estimating Confidence for Top-N Recommendations in Neighbourhood-based Algorithms”. Yet to be submitted.

2. Tadiparthi V.R. Himabindu, Vineet Padmanabhan and Arun K. Pujari. ”Conformal matrix factorization based recommender system”, Information Sciences, pp. 1-23, 2018. In Press.

3. Tadiparthi V. R. Himabindu, Vineet Padmanabhan, Arun K. Pujari, and Abdul Sattar. ”Prediction with Confidence in Item Based Collaborative Filtering”. Pacific Rim International Conference on Artificial Intelligence, pp. 125-138. Springer, 2016.

Chapter 2

Background

This chapter introduces fundamental concepts related to recommender systems and their evaluation. We first give a brief overview of recommender systems and their tasks and different kinds of data on which they work. We then introduce the most popular recommendation algorithms in the literature, namely collaborative filtering (CF) algorithms which are solely based on the rating information to make predictions and recommendations. Finally, we discuss different approaches and measures used to evaluate collaborative filtering recommender systems along with the data sets used in our experiments throughout this thesis.

2.1 Recommender Systems

From users' perspective, the main goal of recommender systems is to deal with the information overload problem by helping users in finding the useful, relevant and interesting items based on their past preferences. Mathematically, we can formulate the recommendation problem as [5]:

Let $\mathcal{U}$ be the set of users and $\mathcal{I}$ be the set of items available in the system. Let f be a utility function such that $f(u, i)$, where $u \in U$ and $i \in I$, measures the usefulness of item i to user u, i.e., $f : \mathcal{U}\ X\ \mathcal{I} \rightarrow \mathcal{R}$. For each $u \in U$, the recommender system has to find the unused item $i'_u \in I$ that maximizes the utility function:

$$i'_u = \underset{i \in \mathcal{I}}{\operatorname{argmax}}\, f(u, i) \tag{2.1}$$

However, f values are available only for a small fraction of user-item pairs and are not available for all the user-item pairs. The goal of the recommender system is to find the utility values for all unknown user-item pairs and make recommendations based on these values. In other words, we can think of utility function as a matrix where users $\mathcal{U}$ form the rows and items $\mathcal{I}$ as columns. The utility value for the given user-item pair $f(u, i)$ represent the preference of user u for the item i. As the preference of all users for every item is not available, we need to first estimate the preference of user for every unknown item and then recommend the items based on the estimated preference values. A number of different algorithms were proposed to estimate these utility values. We review these methods in section 2.4.

2.2 Types of Feedback

From the recommendation problem discussed in the previous section, it is clear that users and items are the two basic entity sets in recommender systems. In addition to users and items, another important aspect that we should consider is the utility value which is the relationship between user and item. The recommendations are usually made based on these utility values. The utility value represents the feedback that a user gives on a specific item. The feedback given by the user can be either explicit or implicit.

- Explicit Feedback: In explicit feedback data sets, preferences are expressed by users. This can take many forms:
  - Numerical: The feedback is in the form of ratings like *1-5* or *1-10* depending on the rating scale chosen by the users. On 1-5 rating scale 1 represents strong dislike and 5 represents strong like towards the item.
  - Ordinal: The feedback is in the form of *ordinal rating scale* such as *strongly disagree, disagree, neutral, agree and strongly agree*. This kind of feedback is generally collected with questionnaires. We can convert numerical rating scale to ordinal and vice-versa (1, 2, 3, 4 and 5 on numerical scale is equivalent to strongly disagree, disagree, neutral, agree and strongly agree respectively on ordinal rating scale and vice-versa). Therefore, hereafter we use numerical and ordinal rating scales interchangeably in our thesis.

  - Binary: Users give their preferences in the form of $like$ or $dislike$, for example *up and down votes* in reditt or in Pandora Internet radio.
  - Unary: Users would give their preference only when they $like$ the item, for example $like$ in face book.

- Implicit Feedback: Feedback is generated by the recommender system, through inferences it makes about the users behavior [27]. This can be either binary or unary:
  - Binary: For example, if a user spends more time on reading a web page or he/she downloaded the web page or there are more number of views for that web page then it could be a $like$. On the other hand, if the user spent only few seconds on the web page since opening, then it could be a $dislike$. Similarly, if a user listens to a song to the end it could be a $like$ and if he/she skips the song immediately it could be a $dislike$.
  - Unary: If a user buys or consumes an item, then it would be a $like$. Examples include user watched a movie, purchased a product, clicked on some item etc.

Numerical and ordinal feedback is explicit as it is always expressed by a user whereas binary and unary feedbacks can be either explicit or implicit as they can be explicitly stated by the users themselves or inferred from the users' actions. Unary feedback is often *positive-only* as it contains only $like$ information (or usage information).

Both explicit and implicit feedback exhibit different characteristics of users' preferences [27].

- Users in general do not rate every item that they have used. Therefore, it is difficult to obtain sufficient and representative feedback from a population of users. On the other hand, implicit feedback is abundant as we can infer the information from explicit feedback given by the user as well as users' actions.
- Explicit feedback is more accurate than implicit feedback in representing the users' interests. Because in the former case, opinions are expressed by the users explicitly whereas in the latter case system has to rely on the application of tools

and methodologies for capturing and observing the user's actions and interpreting those actions to make inferences about the user's interests.

- Both suffer from noise and are sensitive to the user's context. For example, assume that a customer bought a product from some e-commerce website. Initially, he is satisfied with that product as it works well for some time. Hence, he rated it 5 (explicit feedback). After a few days, he found some problem with that product and as a result he might think not to purchase that product or not to purchase the items form that e-commerce site in future, but might forget to update that rating. Other users still think that it is a top rated product [78]. One more reason is that the user who liked an item in the past may not like it in the future due to change in his/her tastes. The ratings given by the user in the past might not reflect his/her present taste. Same kind of problem arises with implicit feedback. For example, if a user purchases an item, we may think he/she likes that item which may not always be true. The item may have been purchased for a friend, or perhaps the user was disappointed with the product.

2.3 Recommender Systems Tasks

As we mentioned in section 2.1, recommender systems are aimed at delivering useful, relevant and interesting items to the users. This can be done in two steps:

- Prediction: For every item that is not yet rated by the user, predict the preference of the user for that item. These preferences can be generated in one of the two ways depending on the type of feedback given by the user.
  - Rating Prediction: When the feedback is explicit (except unary), preference of the user for the given item is predicted as rating. In the case of numerical data sets, this predicted rating is in the form of *1-5* or *1-10* whereas in the binary data sets, the prediction just tells whether the user likes the item or not.
  - Score Prediction: Here, the score for each unknown item is calculated which tells us how much the user prefers that item. Generally used when the data is implicit, specifically when we have *positive-only* feedback (unary

ratings) in the data set. Scores for items are also computed in binary feedback data sets.

- Recommendation: In this step, items are ordered in descending order based on the predicted ratings or scores. Then, items with the highest predicted ratings or scores will be recommended to the user.

Therefore, prediction and recommendation are the two main tasks performed by the recommender systems in order to recommend useful, relevant and interesting items to the users.

Most of the papers in recommender systems literature concentrated on rating prediction. The goal here is to accurately predict the ratings that the users would give to unknown items and recommend those items with the highest predicted ratings. But as mentioned in [50], the items with the highest ratings are not always the ones which are useful to the user. For example, recommending the most popular items (items with many ratings) are not always useful to the users because users can anyway find those items on their own. Sometimes, an item with the highest predicted rating may not be useful for the user due to insufficient information available to make the prediction. This is the case, where an item has limited number of ratings (say, rated by only 2 or 3 users) but all these ratings are high. As we are making prediction based on the available ratings, in this case, only 2 or 3 ratings, this prediction is not accurate.

Though prediction is important in recommender systems, in general, users expect a fixed number of useful and relevant items in recommendation list. These items are ordered in decreasing order of their preference. That means most preferred items are preceded by less preferred items in the list. In the case of explicit ratings, specifically, numerical ratings, the items are ordered based on their predicted ratings, whereas in the case of binary and unary ratings the items are ordered based on the predicted scores. In the case of binary ratings only the items which are predicted as $like$ are recommended to the user based on the predicted scores.

2.4 Collaborative Filtering Algorithms

Users and Items are the two basic entity sets in recommender systems where predictions and recommendations will be done based on the information related to these two

entities. This information can be related to independent entities i.e., either users or items but not both or it can be related to both entities. The former is known as content information and the latter is collaborative information.

- Content information: This information is related to a user or an item. e.g. user attributes like age, gender, qualification, hobbies or item attributes like the price, quantity purchased, category etc.

- Collaborative information: This information is related to both user and item which is the feedback given by the user for that item. As discussed in section 2.2, this feedback can be either explicit or implicit. Explicit feedback can be numerical or binary whereas implicit feedback is *positive-only* feedback which does not contain negative preferences.

There are several algorithms in recommender systems literature to recommend items based on content information or collaborative information or both. In this section, we present some popular algorithms based on collaborative information as our thesis focuses on collaborative filtering algorithms.

2.4.1 Baselines

Baselines are the simple methods used to make predictions and recommendations. Though, generally these methods are not used in practical sense, they are useful to compare with new algorithms. The new algorithm should perform much better than the existing baseline method.

Prediction: Following are the baselines used to make simple predictions:

1. Global Average: Here, the predicted rating for every unknown rating is the average of all the available ratings. The disadvantage with this approach is that we cannot separate useful items from other items to make recommendations to the user as the predicted rating for every unknown item is same.

$$\hat{r_{u,i}} = \frac{1}{|D|} X \sum_{v \in D} \sum_{j \in D} r_{v,j} \tag{2.2}$$

where D is the set of available ratings in the data set.

2. User Average: The rating for every unknown item of the given user is the average of rating given by the user on known items. This method also has the same disadvantage of not being able to recommend useful items to the user as all the predicted ratings for every unknown item of the given user are same.

$$\hat{r_{u,i}} = \frac{1}{|D_u|} X \sum_{j \in D_u} r_{u,j} \tag{2.3}$$

where D_u is the set of ratings given by user u.

3. Item Average: The rating for every unknown item of the given user is the average of the ratings given to that item. Item average method is more useful compared to user average because we can distinguish preferred items from other items and recommend the preferred items to the given user as the predicted ratings of unknown items for the given user are different.

$$\hat{r_{u,i}} = \frac{1}{|D_i|} X \sum_{v \in D_i} r_{v,i} \tag{2.4}$$

where D_i is the set of ratings assigned to item i.

Recommendation: Two popular baseline methods for recommending the items are random and popularity.

1. Random: We randomly pick some unknown items for the given user and recommend those items. Alternatively, random scores are assigned to the unknown items for the given user. Recommend the items with highest scores by ordering all the unknown items in descending order based on the randomly assigned scores. This is very inaccurate method of recommending the items to users as we are not considering any information about users, items or ratings in making the recommendations.

2. Popularity: In this approach, every item is assigned a score which is based on the item's popularity. Popularity score depends on the type of feedback in the data set. For example, in implicit feedback data sets where only positive feedback is available, popularity score of the item refers to the number of users having consumed this item. The disadvantage of this approach is that it is non-personalized as the items to be recommended to every user are same unless the most popular items are already consumed by the user.

2.4.2 Neighborhood-based Algorithms

Neighborhood-based algorithms are very simple and most popular collaborative filtering algorithms. As the name suggests, these algorithms use preferences of similar neighbors to make predictions and recommendations [64, 66]. This can be done as follows: In neighborhood-based algorithms first we need to compute the similarities between the entities to find similar neighbors for the given entity. Once the neighbors are found, we use the preferences of these neighbors to predict the ratings or scores of unknown items of the given user. Then, recommend the items with highest predicted ratings or scores by ranking the unknown items of the given user in descending order. Therefore, neighborhood-based algorithms are composed of three steps:

- Similarity Computation
- Rating or Score Prediction
- Top-N Recommendation

As we mentioned in section 2.2, the two basic entities in recommender systems are users and items. Based on these two entities, two neighborhood-based collaborative algorithms are proposed: User-based collaborative Filtering and Item-based collaborative filtering.

1. User-based Collaborative Filtering [64]:

The intuition behind user-based collaborative filtering is that users like those items which are liked by the users having similar taste. Therefore, feedback given by similar users is used to predict the preferences of the unknown items of the given user. Based on these predicted preferences, items are recommended to the given user. Three steps followed in user-based collaborative filtering algorithm are:

- User Similarity Computation: In this step, we need to find the similarities between given user and all other users. In order to find the similarity between two users say u and v, we need to consider the items that are consumed by both users. We have several similarity measures to perform this task and these similarity measures are different for different kinds of feedback. Most popular among them are Pearson and cosine similarity measures.

- Rating or Score Prediction: Once the similarities are computed, k most similar users are extracted and the preference of every unknown item of the given user is found by aggregating the preferences of similar users. The preference can be a rating in case of numerical data sets or a score in the case of *positive-only* feedback data sets.

- Top-N Recommendation: Finally, rank all unknown items in descending order based on their predicted rating or score and recommend the topmost N items in the ranked list.

2. Item-based Collaborative Filtering [66]:

The general idea underlying the item-based collaborative filtering is that a user will more likely purchase items that are similar to items he/she already purchased. Therefore, feedback given to the similar items is used to predict the preferences of the given item. Based on these predicted preferences, items are recommended to the given user. The three steps in item-based collaborative filtering algorithm are:

- Item Similarity Computation: In this step, we need to find the similarities between all items available in the system. In order to find the similarity between two items say i and j, we need to consider the users who consumed both these items. Several similarity measures were proposed to perform this task and these similarity measures are different for different kinds of feedback. Most popular similarity measures used in item-based collaborative filtering are Pearson and cosine and adjusted cosine.

- Rating or Score Prediction: Once the similarities are computed, k most similar items for the given unknown item from the set of consumed items of the given user are extracted and the preference of this item is found by aggregating the preferences of similar items. Here also, the preference can be a rating in case of numerical data sets or a score in case of *positive-only* feedback data sets.

- Top-N Recommendation: Finally, rank all unknown items in descending order based on their predicted rating or score and recommend the topmost N items in the ranked list.

We give detailed explanation of different similarity measures for different neighborhood-based algorithms and different kinds of feedback data sets in chapter 3. We also explain different ways of predicting the ratings and scores for each algorithms and for different feedback data sets.

2.4.3 Model-based Algorithms

Neighborhood-based algorithms perform well when the data set is dense. Their accuracy tends to decrease with the decrease in the amount of information available in the data set because the similarity computations are not very accurate under sparsity conditions. Furthermore, neighborhood-based algorithms are not scalable with the increase in number of users and items in the data set. Most of the real world data sets used in recommender systems are very sparse and have large number of users and items. Therefore, neighborhood-based algorithms are not scalable, fast and do not produce accurate results for these data sets. To deal with these problems, model-based algorithms were proposed. Model-based algorithms first build a model based on the information available in the data set. This involves capturing hidden information in the data set and using the built model to make predictions and recommendations. Thus, model-based algorithms are scalable, fast and accurate compared to neighborhood-based algorithms when the data set is sparse with large number of users and items. However, the accuracy of these algorithms depends on how well the model fits the real data. In this section, we discuss the most popular model-based algorithm, namely, matrix factorization.

Matrix Factorization [34]:

Matrix Factorization(MF) is a low dimensional factor model which represents both users and items by a small number of latent factors. The assumption here is that these small number of factors influence the preferences of the user for an item. For example, if two users give the same rating, say 5 on the rating sale of *1-5* to the same movie then it might be the case that both users like the actor of the movie or because it is a comedy movie which is the genre preferred by both users. So if we can find these latent factors then we can predict the rating that a particular user might give to a particular item.

Formally, we define the matrix factorization as follows: Given a user-item rating matrix $Y \in \mathbb{R}^{m \times n}$ where m is the number of users and n is the number of items.

Assume that d is the number of latent factors and r is the number of possible ratings. Now our task is to find two matrices, user latent feature matrix $W \in \mathbb{R}^{m \times d}$ and item latent feature matrix $V \in \mathbb{R}^{n \times d}$ such that their product is approximately equal to Y:

$$W \times V^T = \hat{Y} \approx Y \tag{2.5}$$

Here each row of W contains weights which represents how much the user prefers each latent factor. Similarly, each row of V represents to what extent the movie is characterized by each latent factor. To get the prediction of a rating that a user will give to an item, we can calculate the dot product of the two vectors corresponding to user i and item j:

$$\hat{y}_{ij} = w_i v_j^T = \sum_{a=1}^{d} w_{ia} v_{aj} \tag{2.6}$$

This is the basic matrix factorization. Several extensions were made to improve the performance of the basic matrix factorization algorithm such as Regularized Matrix Factorization (RMF) [20, 34], Non negative Matrix Factorization (NMF) [38] and Maximum-margin Matrix Factorization (MMMF) [63, 72]. The detailed explanation of these variants is given in chapter 5.

2.4.4 Memory-based Vs. Model-based Algorithms

- Neighborhood-based algorithms are easy to understand and simple to implement while achieving reasonably accurate results. Whereas model-based algorithms are very complex which involves estimation of large number of parameters. Furthermore, they depend on the number of assumptions. If the assumptions of the model do not hold, it may lead to wrong predictions.

- Neighborhood-based techniques depend on the user similarities (user-based algorithm) and item similarities (item-based algorithm) to make predictions and recommendations . On the other hand, model-based algorithms use several machine learning algorithms to train data to build a model which can then be used to predict the rating for an unknown item of the given user.

- The neighborhood-based algorithms are instance-based learning algorithms or lazy learners. They use entire data set every time they need to form the neigh-

borhood (subset of users or items) for the given user or item. Therefore, prediction process is very slow in neighborhood-based algorithms. In contrast, model-based algorithms are eager learners which build a model to capture the hidden relationships in the data. Using the built model they make the predictions for unknown user-item pairs without having to use the entire training data which speeds up the prediction process. For example, in matrix factorization we first decompose the user-item matrix into a user feature matrix and item feature matrix. Then we use these matrices to make the predictions instead of using the ratings in the entire data set.

2.4.5 Limitations of Collaborative Filtering Algorithms

Collaborative filtering algorithms suffer from two main drawbacks:

- Collaborative filtering algorithms rely on the rating information to make accurate predictions. But, real world data sets used in recommender systems are very sparse. Due to this fact, they do not perform well for very sparse data sets. A solution for this problem is to use content information along with collaborative information.

- When information is not available for an entity, we can not make recommendations. For example, if the user is new to the system then not even a single rating is available to the user. In this case, we cannot make recommendations to the user as the preferences of that user is still unknown. Similarly when a new item is introduced in the system, we cannot recommend this item to any user as we cannot predict the preference of that item for the given user unless some ratings are available for that item. This is called cold-start problem. One solution for this problem is to include only those users and items for which minimum number of ratings are available.

2.5 Evaluating Collaborative Filtering Algorithms

In order to measure the performance of the any given collaborative filtering algorithm and to compare its performance with other algorithms, we need to consider three as-

pects:

- Data sets on which we test our algorithm
- Evaluation Approaches or Experimental settings
- Evaluation Measures to assess the performance

2.5.1 Data sets

Collaborative filtering algorithms are usually tested by conducting experiments which require users' preferences over a set of items. A data set that is suitable for collaborative filtering consists of feedback given by the real users to real items on a real system. As mentioned in section 2.2, this feedback can be either explicit or implicit. Explicit feedback data sets may contain numerical or ordinal ratings where the feedback is in the form of rating scale like 1-5 or binary in which the feedback contains only two values $\{+1(like), -1(dislike)\}$ or unary where we have $positive\text{-}only$ feedback. Implicit feedback can be either binary or unary but is inferred from users' actions instead of explicitly stated by the user. In our experiments, we use four data sets related to movies: Movielens 100K, Eachmovie, Movielens 1M and Movielens-small-latest. All these are numerical feedback data sets. To conduct some of our experiments, we need to convert these data sets into binary and unary. The conversion is done as follows:

- Conversion from numerical feedback to binary feedback: This is done in two ways [39].
  - All the ratings above the midpoint of the rating scale in the corresponding data set are considered as liked items (+1) and all other ratings are disliked items (-1). If the midpoint results in a floating point number then round it to the next integer value.
  - For every user, all ratings above the user average are treated as liked (+1) and all others are considered as disliked (-1).
- Conversion from numerical feedback to unary feedback: consider all ratings as positive (+1). In other words, we are assuming that users like all the movies they have watched.

In what follows, we describe the four data sets mentioned above in detail. Table 2.1 gives the comparison of these four data sets in terms of number of users, movies and ratings.

- MovieLens 100K [2]: The MovieLens 100k data set consists of 100,000 ratings on a rating scale of 1-5 given by 943 users to 1682 movies. It also contains simple demographic information for the users like age, gender, occupation and zip code and information about movies such as movie title, release date, video release date, IMDb URL and movie genre. The data was collected through the MovieLens web site between September 19th, 1997 through April 22nd, 1998. Each user has atleast 20 ratings.

- Eachmovie [1]: HP/Compaq Research (formerly DEC Research) ran the EachMovie recommendation service for 18 months. The original EachMovie dataset contained 2,811,983 ratings (1-6 stars) entered by 72,916 users for 1628 different movies. This sub-dataset which contains only those users who rated more than atleast 20 movies includes the 2579983 ratings from 36656 users on 1621 movies. Rating values range from 1-6.

- Movielens 1M [2]: Movielens 1M contains 1,000,209 ratings on a rating scale of 1-5 given by 3,952 users who joined MovieLens in 2000 to 6040 movies. Like movielens 100K data set, Movielens 1M is also collected from the same website and also contain users demographic information and movie information. All selected users had rated at least 20 movies.

- MovieLens-latest-small [2]: Collected from the MovieLens web site Movielens-latest-small contains 1,00,234 ratings on a rating scale of 1-5 given by 718 users to 8915 movies. The data was created between March 26, 1996 and August 05, 2015. All selected users had rated at least 20 movies.

2.5.2 Evaluation Approaches

In this section, we describe the most popular evaluation approaches mentioned in recommender systems literature in detail along with the advantages and disadvantages of each approach [70].

Table 2.1: Summary of Data sets

Dataset	#Users	#Items	#Ratings
MovieLens 100K	943	1682	100000
EachMovie	36656	1621	2579983
MovieLens 1M	6040	3952	1000209
MovieLens-latest-small	718	8915	100234

1. Offline Experiments

Offline experiments have been extensively used in the recommender systems literature to measure the performance of the algorithms and to compare different algorithms. They are very simple and inexpensive and depend only on the pre-collected data sets which contains user preferences for different items. In offline experiments, in order to make reliable recommendations, we assume that the user behaviour in the past reflects the user behaviour in future. Therefore, offline experiments try to simulate the users' behaviour based on their preferences in the data set. Further more, offline experiments provide an objective way of measuring the performance of recommendation algorithms as they are based on evaluation measures which capture quantitative dimensions instead of depending on the subjective measures which can be evaluated only using online evaluation.

Offline evaluation partitions data into two subsets: training and test sets. This partition can be done in several ways:

- Holdout: A subset of the available ratings are randomly selected for test set in the whole data set. The remaining ratings form the training set. For example 20% of the randomly selected ratings in the data set constitute test set and the remaining 80% form the training set.

- User-based holdout: For the given user, randomly select fixed percentage of ratings as the test set and remaining ratings as the training set. For example, 20% ratings given by the user is randomly selected as the test set and the remaining 80% of the user's ratings form the training set.

- Leave-k-out: Instead of selecting a fixed percentage of items for the given user, here we select a fixed number of items randomly, say k, which may be for example 5 or 10.

- Leave-one-out: Similar to Leave-k-out with $k = 1$ where for the given user only one item is held out for the test set and the remaining items form the training set.

- *k*-fold cross-validation. It divides the data into k folds which are mutually exclusive. The experiment is repeated k times where i^{th} fold, $i \in \{1, 2, ..., k\}$ is used as the test set in the i^{th} run and remaining folds $\{1, 2, ..., k\} \setminus i$ are used as the training set and finally the results of all folds are averaged.

The advantage of *k*-fold cross-validation as compared to other methods is that it is less sensitive to the way in which the data is partitioned. Every rating is there in the test set exactly once, and is part of the training set *k-1* times. Therefore, any variance in the predictions is reduced as *k* is increased. Usually $k = 5$ or $k = 10$ is a good choice. But we can also achieve more or less similar performance with other methods by repeating the experiments k times for example say $k = 5$.

Once the data is partitioned, data in the training set is used to make predictions or recommendations. Test set is used as a ground truth to check the accuracy of the predicted ratings or the relevance of the recommended items generated by the given algorithm.

Advantages:

- Offline experiments provide a cost-effective way to compare the performance of several recommendation algorithms as they require no interaction with real users.

- Offline experiments are well suited to measuring the objective dimensions of recommendation algorithms like prediction and recommendation accuracy.

Disadvantages:

- As there is no interaction with the users, offline experiments are limited to objective evaluation of recommendation algorithms like their prediction power and recommendation ability. They cannot measure subjective dimensions like how generated recommendations influence the real user, whether the users are satisfied with the recommendations etc.

- Offline experiments assume that past user behaviour reflects his/her future behaviour. Therefore, they cannot capture some important aspects like change in user's taste over time. As a result, the user may not be satisfied with the recommendations generated by the algorithm.

Therefore, offline experiments are generally used to filter out inappropriate algorithms and retain only a small set of candidate algorithms which are then tested using more costly user studies or online experiments. For example, tune the parameters of the algorithm using an offline experiment, and then the algorithm with the best tuned parameters is run using online experiment in order to make recommendations to real users.

2. User Studies

Another way of evaluating the performance of a recommendation algorithm is to recruit a set of test subjects and ask them to interact with the recommendations generated by the recommendation algorithm. Many works include the use of user studies in the evaluation of Recommender Systems [65]. For example, Pu et al. [62] and Knijnenburg et al. [32] proposed different evaluation frameworks for the evaluation with user studies. While subjects are interacting with the recommender system one can observe user's behavior and several subjective aspects such as how much time they are spending in viewing the item, whether they are interested in the generated recommendations etc. Often several qualitative questions in the form of questionnaire or surveys are collected throughout this process. Such questions address those aspects such as whether the user is really satisfied with the recommended items, whether the users feel surprised and interested in the recommendations etc. which cannot be measured with quantitative aspects.

Advantages:

- We can directly observe user behavior while he/she is interacting with the system in user studies and thus we can observe how generated recommendations influence the user behaviour.

- User studies allow us to collect qualitative data which can be used to measure the subjective dimensions of recommender system such as newness, surprise which cannot be measured with objective measures.

Disadvantages:

- User studies are very expensive to conduct as it involves recruiting sufficiently large set of test subjects and measuring their behavior while they are interacting with the recommender systems and repeating this several times in order to make reliable conclusions. This whole process is time consuming and if the test subjects are not volunteers, we should have to pay the compensation for participating in user studies.

- Test subjects should represent the population of users of the real system, otherwise there is a chance that the results can be biased toward a particular set of users. Even when the subjects represent the true population of users, the results can still be biased because they are aware that they are participating in an experiment. This is especially true in case of paid subjects who generally try to satisfy the company conducting the experiment.

3. Online Experiments

One of the problems with offline experiments is that, they can not correctly judge the relevance of a recommended item to the given user unless that item is already consumed by the user. In other words, if a recommender system recommends an item which is not present in the test set, this item is treated as an irrelevant item in offline experiments. This may be the relevant item for the user in true sense, but as it is not consumed by him/her till now because he/she is not ware of that item. And with offline experiments, there is no way to know whether user likes an item if we would recommend an unknown item to that user. As discussed above, another problem with offline experiments is that they can not capture the user's change in taste over time. To deal with these problems, online experiments are conducted where real users directly interact with the system.

The quality of recommender systems depends on a variety of factors other than the predictive and recommendation ability such as user's information needs, newness and diversity in the recommendation list, items they are familiar, amount of trust they have in their system etc. Online evaluation is the only setting which can measure these aspects. With online evaluation we can compare a small set of candidate algorithms and rank them based on their performance. For example, if users follow the recommendations of one system more often than the other we can say that one system is superior to

others. For this reason, many real world systems employ an online testing system [33], where multiple algorithms can be compared. Generally, these systems measure system performance by means of user-centric metrics such as page views, click-through rate [22] etc. While conducting such experiments it is important to select the users randomly, so that the comparisons between alternatives are fair.

Advantages:

- Online experiments capture shift in user's interests with time.
- In order to improve the quality of a recommender system, it is important to know why users like the recommendations of one system over the other. In other words, user satisfaction is an important aspect that decides the quality of a recommender system. Online evaluation is the only setting where we can measure the user satisfaction as the interaction of real users with the system is direct. With online evaluation we can also measure overall system goals like long-term profit or user retention and how these goals are effected by accuracy and diversity of recommendations.

Disadvantages:

- Comparing many algorithms in online setting is very expensive.
- Online experiments are sometimes risky. For example, if a recommender system generates irrelevant recommendations to the real users, the users may not be interested to use the system in future. This may have negative impact on recommender system which is not desirable in commercial applications.

For these reasons, it is best to run an online evaluation on a set of very few candidate recommendation algorithms, after an extensive offline study provides evidence that the candidate algorithms are reasonable. Perhaps a user study that measures the users attitude towards the system can be carried out and this gradual process could reduce the risk in causing significant user dissatisfaction.

2.5.3 Evaluation Measures

The evaluation of recommender systems is not easy because we cannot judge the quality of recommendation algorithms based on a single dimension alone. While evaluating the algorithms we should consider different aspects like the task for which the recommender system is designed, type of data on which it is intended to work etc. Several evaluation measures were proposed in the literature to measure the performance of the recommender systems from different perspectives. Evaluation measures are typically used to assess the performance of the algorithms and to compare the performance of the newly proposed algorithm with the existing algorithms. These measures are broadly classified into two categories: Accuracy measures and Beyond accuracy measures.

1. Accuracy Measures:

Among the several measures proposed in recommender systems literature, accuracy is the most widely used metric. Most of the researchers in recommender systems focused on evaluating their algorithms using accuracy evaluation metrics. Two important tasks performed by recommendation algorithms are prediction and recommendation. Therefore, accuracy metrics for recommender systems are mainly classified into two categories depending on the task for which the algorithms are designed to perform: Predictive accuracy measures and Classification accuracy measures [25, 70].

- Prediction task:

  One of the basic assumptions in recommender systems is that users prefer to use a recommendation algorithm which produces accurate predictions. Based on this assumption many recommendation algorithms are designed to produce the predicted ratings as close as possible to the true ratings. Once the ratings are predicted by the recommendation algorithm we need to asses the algorithm's predictive ability using predictive accuracy measures. These predictive accuracy measures are different for numerical and binary feedback data sets.

  (i) Predictive Accuracy measures for numerical feedback data sets:

  Predictive accuracy metrics for numerical feedback data sets measure how close the predicted ratings produced by a recommendation algorithm is to the true ratings. In other words, they measure how the predicted ratings differ from true

ratings. These measures are specifically used in recommender systems which show the predicted ratings alongside the recommended items to the user.

– Mean Absolute Error (MAE): It measures the average absolute deviation between a predicted rating and the corresponding true rating.

$$MAE = \frac{\sum_{i=1}^{|t|} |T_i - P_i|}{|t|} \tag{2.7}$$

where T_i is the true rating

P_i is the predicted rating

t is the set of test ratings

Advantages:

* It is very simple to calculate and easy to understand.
* It has well studied statistical properties that provide for testing the significance of the difference between the mean absolute errors of two recommendation algorithms.

Disadvantages:

* Mean absolute error is not appropriate for *top-N* recommendation task where the recommendation algorithm recommends a list of N top ranked items. Here, users concern is only on the errors in items present in *top-N* list. For all other items which we assume as not interested items for the user, the user does not care about the errors.
* It is also not appropriate in situations where only the distinction between preferred and not preferred items is important. For example in a data set with a rating scale of 1-5, items with predicted ratings 4 and 5 are considered to be preferred items and remaining items (items with predicted ratings 1, 2 and 3) are the not preferred items. In this case, user is not generally interested in absolute errors of not preferred items.

– Root Mean Square Error (RMSE): It is related to mean absolute error where the errors are squared before summing to emphasize the large errors.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{|t|}(T_i - P_i)^2}{|t|}} \tag{2.8}$$

RMSE has the same disadvantages as MAE but when comparing two recommendation algorithms RMSE prefers the algorithm which makes small errors over many test items to the algorithm which makes large errors on few test items.

(ii) Predictive Accuracy measures for binary feedback data sets:

MAE and RMSE are not appropriate for binary feedback data sets where we have only two preferences: like *(+1)* and dislike *(-1)*. The reason is that we cannot measure absolute errors in this case. The appropriate metric to assess the performance in binary feedback data sets is to measure correctness of the predicted ratings or preferences.

* Percentage of Correctly Classified Test Items (PCCTI): Here, we compute how many of the test items' preferences are correctly classified and is given by

$$PCCTI = \frac{\sum_{i=1}^{|t|} \mathbb{1}_i}{|t|} \tag{2.9}$$

where $\mathbb{1}_i = \begin{cases} 0 & \text{if } T_i = P_i \\ 1 & \text{otherwise} \end{cases}$

- Recommendation task:

  Though prediction is an important task in recommender systems, the ultimate goal of any recommender systems is to recommend a set of items to the user that he/she is interested in. Therefore, identifying the truly relevant items is crucial in recommendation task. Two measures are popularly used for this purpose: Precision and Recall

  – Precision: Precision measures how many of the recommended items are truly relevant.

$$precision = \frac{|relevant \cap recommended|}{|recommended|} \tag{2.10}$$

– Recall: Recall measures how many of the relevant items are actually being recommended.

$$recall = \frac{|relevant \cap recommended|}{|relevant|} \tag{2.11}$$

Disadvantage:

Precision and recall are sensitive to the number of recommendations *(N)*. If the number of recommended items is much greater than the number of relevant items we will get low precision values. Similarly, if the number of relevant items are greater than the number of recommended items one can easily expect lower recall values.

2. Measures beyond Accuracy

- Coverage [23, 24, 31]: It is a measure of the percentage of items for which the recommendation algorithm can make predictions or recommendations. A recommendation algorithm with high coverage means that the algorithm is able to make predictions or recommendations for large number of items which indicates the quality of the recommendation algorithm.

$$Coverage = \frac{|I_p|}{|I|} \tag{2.12}$$

where I_p is the set of items for which the predictions are made

I is the total number of items

For example, in user-based neighborhood-based algorithm, we can make prediction to a particular item if and only if that item is used by at least one neighbour. In this case, I_p refers to the number of candidate items which are used by at least one neighbor.

- Novelty [15, 16, 31, 73, 80]: It is a measure of interestingness of a recommendation to a user and at the same time the novel recommendations are those items that the user did not know.

- Unexpectedness [3, 30]: Unexpected items are the items that the user is unaware of, but the difference between novel and unexpected item is that the user may find the novel item by him/herself but not the unexpected item. Unexpected items are not always interesting to the users.

- Serendipity [23, 31, 36]: It is a measure of interestingness and unexpectedness of a recommendation to a user. So, a serendipitous recommendation is also a novel recommendation although the reverse may not be true.

  Novelty, Unexpectedness and Serendipity are the subjective measures which require user interaction.

- Diversity [12, 15, 16, 31, 37, 73, 81]: Generally, all the above measures are applied to a single recommended item. But diversity is a measure applied to a set of recommended items which gives information about how different the items are with each other. Intra-List Diversity (ILD) measures diversity as the average pairwise distance between the items in the recommendation list R_u [81] and is given by

$$ILD = \frac{\sum_{i,j \in R_u} dist(i,j)}{|R_u|(|R_u - 1|)} \tag{2.13}$$

  $dist(i,j)$ be the distance between item i and item$j$

  R_u is the set of items recommended to the user u

- Confidence: Confidence in the recommender systems tells us how certain the system is in making the predictions or recommendations [70]. In other words, confidence gives the uncertainty in predictions made by recommendation algorithms. Several factors such as data sparsity, noisy data, choice of different parameter values, suitability of the algorithm for the given data and assumptions used in the model lead to uncertainty in recommender systems. As a result, some of the predictions made by the algorithms are inaccurate. Displaying the wrong or inaccurate predictions to the users will decrease the trust of the users in recommendation systems as the users think that the system might be wrong and possibly refrain from using that recommender system in future. Therefore, measuring the confidence of the predictions and attaching them with the predicted

value is a good idea in retaining the customer loyalty which in turn increases the revenue while improving the customer satisfaction.

A confidence measure is important as it can help users decide which movies to watch, products to buy, and also help an e-commerce site in making a decision on which recommendations should not be displayed, because an erratic recommendation can diminish the trust of users in the system [4, 5]. Mcnee et al.[6] studied how does adding a confidence display change users satisfaction and behavior and concluded that addition of a confidence display to a recommender system increases user satisfaction. It also alters users behavior. For user tasks with varying amounts of risk, users were more likely to seek out or avoid low confidence recommendations as appropriate.

In addition to the above measures, we can also judge the quality of a recommendation algorithm by measuring its run time performance. We can do this by measuring the computation time. We can measure this computation time from a theoretical perspective, looking at the computational complexity or from a more practical perspective, by measuring the run time.

2.6 Summary

In this chapter, we have introduced recommender systems and their tasks. We also explained various collaborative filtering algorithms and different kinds of data on which these algorithms work. Though, several recommendation algorithms like content-based algorithms, collaborative filtering algorithms, hybrid algorithms, knowledge-based algorithms are proposed in the literature, we focus on collaborative filtering algorithms in our research as they depend only on the rating information and do not require any other external information like user and item attributes. Therefore, we discussed only collaborative filtering algorithms in this chapter. We described baseline methods, neighborhood-based algorithms and model based algorithms for collaborative filtering. Baseline methods discussed in this chapter are based on user average, item average and global average and are generally used for comparison purposes. Neighborhood-based algorithms follow a three step process to make recommendations: similarity computation, rating or score prediction and *top-N* recommendation. Model-based algorithms

first build the model based on the training data and use that model to make predictions or recommendations. For example, in matrix factorization, we assume that there are some hidden factors in items which influence the users to prefer the items. Based on this assumption, we build a model which contains two components: User feature matrix and Item feature matrix and we use these matrices for prediction or recommendation.

Finally, we described the evaluation of recommender systems and different aspects involved in it, i.e., the data sets used in our experiments throughout the thesis and different evaluation approaches and evaluation measures proposed in the literature. In our experiments, we use four data sets related to movies: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small. Three evaluation approaches proposed in the literature are also discussed each with its advantages and disadvantages: Offline experiments, User studies and Online experiments. Then, different evaluation measures used to measure the performance of recommendation algorithms are briefly explained.

Chapter 3

Task-based Evaluation of Neighbourhood-Based CF Algorithms with Multiple Feedback Types

In the previous chapter, we discussed different collaborative filtering recommendation algorithms, specifically neighbourhood-based algorithms and matrix factorization in detail, and their evaluation. The performance of the neighbourhood-based algorithms vary depending on the similarity measures used to find the similarity between two entities, prediction techniques used to predict the ratings or scores, data set used in evaluation and type of feedback in the data set. Therefore, in this chapter, we present an empirical analysis on neighbourhood-based algorithms with different similarity measures, prediction techniques and data sets with different types of feedback.

In this chapter, we evaluate neighbourhood-based recommendation algorithms for different feedback data sets with suitable offline evaluation measures. For each of these algorithms and for each type of feedback we discuss the following aspects:

- Similarity measures
- Rating or Score Prediction
- Item Recommendation

We also evaluate these algorithms using appropriate significance tests in order to measure the statistical significance of the performance differences among the algorithms

with different similarity measures and between different algorithms for each of the prediction and recommendation tasks. We describe different significance tests used in our experiments to determine whether the observed differences in results are due to luck or really significant. In summary, we discuss the following questions in this chapter:

- How does the different neighbourhood-based algorithms proposed in the literature work for different feedback data sets?
- What similarity measures are available for each of these algorithms and for different feedback data sets?
- How to predict the rating or score for different feedback data sets?
- How the prediction and recommendation ability of the neighborhood-based algorithms are measured for different kinds of feedback?
- What Significance tests are used to determine whether the performance differences of different algorithms are statistically significant?
- Which algorithm best suits for prediction and recommendation tasks and why?
- Why unary feedback data sets perform better than numerical and binary for recommendation task?
- Analyze the performance differences between the different prediction techniques and similarity measures of different algorithms for different tasks, different kinds of feedback and different data sets.

Finally, we discuss our experimental results along with some important observations.

3.1 Neighborhood-based CF Algorithms

Neighborhood-based CF algorithms perform prediction and recommendation by computing the similarities between the entities in an entity set. We have two kinds of entities in recommender systems: Users and Items. If neighborhood-based CF algorithm predicts the ratings or recommends the items based on the similarities between

the users we call it as the user-based CF algorithm. On the other hand, if it is based on similarities between items then we call it as the item-based CF algorithm. In what follows, we discuss different similarity measures used in each of these algorithms for different feedback data sets and how these similarity values can be used to make predictions and recommendations to users.

3.1.1 User-based Collaborative Filtering

The intuition behind user-based collaborative filtering algorithm [64] is that user likes the items that are liked by other similar users. In order to do this, first we need to find the subset of users similar to the target user or for the given item of the target user. Then predict the ratings or scores for every unused item of the target user and based on these predicted ratings/scores, recommend the items to the target user. Here, we discuss, how this can be done for different feedback data sets.

1. Numerical feedback data sets:

In numerical feedback data sets, user feedback is in the form of ratings like 1-5 or 1-10, where 1 represents user's strong dislike and 5 or 10 represent strong like towards the item.

- Similarity Computation: In user-based collaborative filtering, in order to find the similarity between two users u and v, we need to find the items that are rated by both users. Pearson and Cosine are the two most widely used similarity measures to compute the similarity values in user-based collaborative filtering for numerical feedback data sets.

  (i) Pearson Similarity: Pearson correlation coefficient measures linear correlation between two users u and v by computing the covariance between these two users.

$$similarity(u,v) = \frac{\sum_{i \in I_u \cap I_v}(r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{ui} - \overline{r_u})^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{vi} - \overline{r_v})^2}} \tag{3.1}$$

  where I_u is the set of items consumed by user u

  I_v is the set of items consumed by user v

r_{ui} is the rating given by user u to item i

$\overline{r_u}$ is the user u's average rating

r_{vi} is the rating given by user v to item i

$\overline{r_v}$ is the user v's average rating

The equation given above will give the value between [-1,1].

(ii) Cosine Similarity: Here, the ratings given by each user is viewed as a vector and the similarity between two users is measured by computing the Cosine of the angle between them.

$$similarity(u, v) = \frac{\vec{u}.\vec{v}}{||\vec{u}||_2 * ||\vec{v}||_2} \tag{3.2}$$

where $\vec{u}$ is the ratings vector of user u

$\vec{v}$ is the ratings vector of user v

. is the dot product between two vectors

The values produced by this function lies between [0,1].

- Rating Prediction:

  Once the similarities are computed using any one of the above similarity measures, a set of k nearest neighbours for the target user is extracted and the similarity values of these nearest neighbours along with their ratings are used to predict the ratings for the items to be recommended to the target user. In other words, we are recommending those items to the target user which are liked by his/her neighbours. The problem with this approach is that small neighbourhood sizes lead to poor coverage. Coverage here refers to percentage of items for which the algorithm can produce predictions. To deal with this problem we use the following approach in selecting the neighbours for unused items: instead of selecting the neighbours for the target user, here we are assuming that all users are neighbours for the target user. For each item predicted, the highest ranking neighbours (they form the user's neighbourhood for that item) that have rated the item are used to compute a prediction. This means that a user may have a

different neighbourhood for each item [24]. Once the neighbourhood is formed , prediction can be done in two ways:

- Simple Weighted Average (SWA): This is just a simple weighted average of the ratings given by the neighbours of the target user a to item i and is given by

$$prediction(a,i) = \frac{\sum_{u=1}^{k} similarity(u,a) * r_{ui}}{k} \tag{3.3}$$

- Average deviation from mean (ADM): Instead of directly using the ratings given by the neighbours, if we use average deviation from mean, we can produce much more accurate predictions.

$$prediction(a,i) = \overline{r_a} + \frac{\sum_{u=1}^{k} similarity(u,a) * (r_{ui} - \overline{r_u})}{k} \tag{3.4}$$

- Item Recommendation: After predicting the ratings for every unused item of the target user a using either simple weighted average or average deviation from mean, we recommend *top-N* items with highest ratings.

2. Binary feedback data sets:

In binary feedback data sets we have only two ratings: $+1$ represents $like$ and -1 represents $dislike$.

- Similarity Computation: Most popular similarity measures used to compute the similarity between two users for binary feedback data sets are Phi coefficient, Cosine and Jaccard.

  (i) Phi coefficient (Pearson): This is the binary version of the Pearson coefficient which is used to measure the association between two binary variables and is defined as follows:

$$similarity(u,v) = \frac{n_{u+1,v+1}n_{u-1,v-1} - n_{u+1,v-1}n_{u-1,v+1}}{\sqrt{n_{u+1}n_{u-1}n_{v+1}n_{v-1}}} \tag{3.5}$$

  where $n_{u+1,v+1}$ is the number of items liked by both users u and v

  $n_{u-1,v-1}$ is the number of items disliked by both users u and v

$n_{u+1,v-1}$ is the number of items liked by user u and and disliked by user v

$n_{u-1,v+1}$ is the number of items disliked by user u and and liked by user v

n_{u+1} is the number of items liked by user u

n_{u-1} is the number of items disliked by user u

n_{v+1} is the number of items liked by user v

n_{v-1} is the number of items disliked by user v

The above function generates a value between [-1,+1]

(ii) Cosine similarity: Two users are considered to be similar if they both liked the same movies and disliked the same movies and is defined as

$$similarity(u,v) = \frac{n_{u+1,v+1} + n_{u-1,v-1} - n_{u+1,v-1} - n_{u-1,v+1}}{\sqrt{n_{u+1} + n_{u-1}}\sqrt{n_{v+1} + n_{v-1}}} \tag{3.6}$$

This is same as the Cosine similarity defined for numerical data sets. In other words, this equation is equivalent to Equation (3.2)

The value lies between [-1,+1]

(iii) Jaccard similarity: Jaccard similarity is defined in the same way as Cosine. The only exception is that in the denominator we consider only those items that are consumed by at least one user and is given by

$$similarity(u,v) = \frac{n_{u+1,v+1} + n_{u-1,v-1} - n_{u+1,v-1} - n_{u-1,v+1}}{|I_u \cup I_v|} \tag{3.7}$$

where I_u is the set of items consumed by user u

I_v is the set of items consumed by user v

This function also produces the values that are between [-1,+1].

- Preference prediction:

  Similar to numerical data sets, here also k nearest neighbors are selected based on the similarity values computed using any of the above similarity measures. Based on the preferences of the nearest neighbors for the given item i and their

similarity values to the target user, we can predict preference of the target user a for item i as follows:

- Based on Majority (BoM): Here we count the number of nearest neighbors who liked (+1) the item i and the number of nearest neighbors who disliked (-1) the item i. If the number of users who gave $+1$ are greater than those of -1, then we predict that the target user a likes the item i otherwise he/she dislikes the item i.

$$preference(a,i) = \begin{cases} +1 & \text{if } n_{+1} > n_{-1} \\ -1 & \text{otherwise} \end{cases} \tag{3.8}$$

  where n_{+1} is the number of nearest neighbors who liked (+1) item i
  n_{-1} is the number of nearest neighbors who disliked (-1) item i

- Based on Sum similarities (BoSS): Here we sum up the similarity values of all the nearest neighbors who rated the item i as +1 and sum up the similarity values of all the nearest neighbors who rated the item i as -1. Then we compare these two similarity values and assign the label with the highest similarity score.

$$preference(a,i) = \begin{cases} +1 & \text{if } SS_{+1} > SS_{-1} \\ -1 & \text{otherwise} \end{cases} \tag{3.9}$$

  where SS_{+1} is the sum of similarity values of all the nearest neighbors who liked (+1) item i
  SS_{-1} is the sum of similarity values of all the nearest neighbors who disliked (-1) item i

- Item recommendation: Generally, in binary settings we recommend all items which are predicted as +1 to the target user a. But in *top-N* setting, we have to recommend only N items. In this case, in order to distinguish the items which are predicted as +1, we also associate a score which is the sum of similarity values of all the neighbors who liked the item i. Based on these scores we can recommend *top-N* items with the highest scores among the items which are predicted as liked (+1).

3. Unary feedback data sets:

Unary data sets contain information about whether the user liked the item. Generally, if a user has consumed an item or purchased an item we assume that he liked that item, though this might not be true in all cases. In unary feedback, we do not have any information about items that a user has disliked. Therefore, in unary data sets '1' indicates user liked that item.

- Similarity Computation: Several similarity measures can be used to compute the similarities between two users in unary feedback data sets. Among them Phi coefficient, Cosine, Jaccard and Dice are the popular similarity measures used for unary feedback data sets.

  (i) Phi coefficient (Pearson): This is similar to phi coefficient defined for binary data sets and is defined as follows:

$$similarity(u,v) = \frac{n_{u1,v1}n_{u0,v0} - n_{u1,v0}n_{u0,v1}}{\sqrt{n_{u1}n_{u0}n_{v1}n_{v0}}} \tag{3.10}$$

  where $n_{u1,v1}$ is the number of items consumed by both users u and v

  $n_{u0,v0}$ is the number of items not consumed by both users u and v

  $n_{u1,v0}$ is the number of items consumed by user u and not consumed by user v

  $n_{u0,v1}$ is the number of items not consumed by user u and consumedby user v

  n_{u1} is the number of items consumed by user u

  n_{u0} is the number of items not consumed by user u

  n_{v1} is the number of items consumed by user v

  n_{v0} is the number of items not consumed by user v

  The above function generates a values between [-1,+1]

  (ii) Cosine Similarity: This similarity measure is based on the number of common items consumed by both users u and v.

$$similarity(u,v) = \frac{|I_u \cap I_v|}{\sqrt{|I_u|}\sqrt{|I_v|}} \tag{3.11}$$

where I_u is the set of items consumed by user u

I_v is the set of items consumed by user v

This equation is similar to the Cosine similarity function defined for ordinal and binary feedback data sets i.e., Equation (3.2)

This function produces the values that are between [0,1].

(iii) Jaccard Similarity: This is similar to Cosine similarity measure except that in the denominator we consider all items consumed by at least one user and is given by

$$similarity(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{3.12}$$

The generated similarity values lies between [0,1].

(iv) Dice coefficient: This is a variant of Jaccard and is defined as

$$similarity(u, v) = \frac{2 * |I_u \cap I_v|}{|I_u| + |I_v|} \tag{3.13}$$

This function also produces the values that are between [0,1].

- Score computation: Once the similarities are computed, we extract the k most similar users for the given item i of the target user a and score for the item i is computed by summing up all the similarity values of k nearest neighbors with that of the target user.

$$score(a, i) = \sum_{u=1}^{k} similarity(a, u) \tag{3.14}$$

- Item Recommendation:

  After computing the score for every unused item of the target user we recommend *top-N* items with the highest scores to the target user a.

3.1.2 Item-based Collaborative Filtering

The underlying principle behind item-based collaborative filtering algorithm [66] is that users like to purchase those items that are similar to the items they have consumed in the past. In order to do this, first we need to find the subset of items from the set of items consumed by the target user for the given item. Then predict the ratings or scores for every unused item of the target user using the ratings and similarity values of the neighboring items and based on these predicted ratings or scores, recommend the items to the target user. Here, we discuss, how this can be done for different feedback data sets.

1. Numerical feedback data sets:

- Similarity Computation: In item-based collaborative filtering, in order to find the similarity between two items i and j, we need to find the users who consumed both items. Most popularly used similarity measures to compute the similarity between the two items in item-based collaborative filtering are Pearson, Cosine and Adjusted Cosine.

  (i) Pearson Similarity: Similar to Pearson correlation coefficient used for user-based CF except that it measures linear correlation between two items i and j by computing the covariance between these two items.

$$similarity(i,j) = \frac{\sum_{u \in U_i \cap U_j}(r_{ui} - \overline{r_i})(r_{uj} - \overline{r_j})}{\sqrt{\sum_{u \in U_i \cap U_j}(r_{ui} - \overline{r_i})^2}\sqrt{\sum_{u \in U_i \cap U_j}(r_{uj} - \overline{r_j})^2}} \tag{3.15}$$

  where U_i is the set of users who consumed item i

  U_j is the set of users who consumed item j

  r_{ui} is the rating given by user u to item i

  r_{uj} is the rating given by user u to item j

  $\overline{r_i}$ is the item i's average rating

  $\overline{r_j}$ is the item j's average rating

  The above equation will give the value between [-1,1].

(ii) Cosine Similarity: Here, the ratings assigned to each item is viewed as a vector and the similarity between two items is measured by computing the Cosine of the angle between them.

$$similarity(i,j) = \frac{\vec{i}.\vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2} \tag{3.16}$$

where $\vec{i}$ is the ratings vector of item i

$\vec{j}$ is the ratings vector of item j

. is the dot product between two vectors

The values produced by this function lies between [0,1].

(iii) Adjusted Cosine Similarity: Cosine similarity measure when used in item-based collaborative filtering does not consider the differences in rating scale used by different users. This will not be a problem in user-based collaborative filtering because there the similarity is computed between two users where each user follows the same rating scale. But in item-based algorithm we are computing the similarity between the items. Rating information of an item is composed of ratings given by different users. Therefore, this will be a problem in item-based algorithm. To deal with this problem, we use Adjusted Cosine similarity in which we subtract the user's average rating from the corresponding user rating for each co-rated pair. This can be defined as follows:

$$similarity(i,j) = \frac{\sum_{u \in U_i \cap U_j}(r_{ui} - \overline{r_u})(r_{uj} - \overline{r_u})}{\sqrt{\sum_{u \in U_i \cap U_j}(r_{ui} - \overline{r_u})^2}\sqrt{\sum_{u \in U_i \cap U_j}(r_{uj} - \overline{r_u})^2}} \tag{3.17}$$

$\overline{r_u}$ is the user u's average rating.

- Rating Prediction:

  Once the similarities are computed using any one of the above similarity measures, a set of k nearest neighbors for the given item of the target user a is extracted and the similarity values of these nearest neighbors along with their ratings are used to predict the ratings for the items to be recommended to the

target user. Once the neighborhood is formed, prediction can be done in two ways:

- Simple Weighted Average (SWA): This is a just a simple weighted average of the ratings that the target user a has assigned to the nearest neighbors of item i and is given by

$$prediction(a,i) = \frac{\sum_{j=1}^{k} similarity(j,i) * r_{aj}}{k} \tag{3.18}$$

- Average Deviation from Mean (ADM): Similar to the user-based collaborative filtering, instead of directly using the ratings given to the nearest neighbors, here we use average deviation from mean.

$$prediction(a,i) = \overline{r_i} + \frac{\sum_{j=1}^{k} similarity(j,i) * (r_{aj} - \overline{r_j})}{k} \tag{3.19}$$

- Item Recommendation: After predicting the ratings for every unused item of the target user a using either simple weighted average or average deviation from mean, we recommend *top-N* items with highest ratings.

2. Binary feedback data sets:

- Similarity Computation: The similarity measures used in item-based CF are similar to those used in user-based CF. But, here, we use them to compute the similarity between two items.

  (i) Phi Coefficient (Pearson): This is the binary version of the Pearson coefficient which is used to measure the association between two binary variables and is defined as follows:

$$similarity(i,j) = \frac{n_{i+1,j+1}n_{i-1,j-1} - n_{i+1,j-1}n_{i-1,j+1}}{\sqrt{n_{i+1}n_{i-1}n_{j+1}n_{j-1}}} \tag{3.20}$$

  where $n_{i+1,j+1}$ is the number of users who liked both items i and j

  $n_{i-1,j-1}$ is the number of users who disliked both items i and j

  $n_{i+1,j-1}$ is the number of users who liked item i and disliked item j

$n_{i-1,j+1}$ is the number of users who disliked item i and liked item j

n_{i+1} is the number of users who liked item i

n_{i-1} is the number of users who disliked item i

n_{j+1} is the number of users who liked item j

n_{j-1} is the number of users who disliked item i

The above function generates a value between [-1,+1]

(ii) Cosine similarity: Two items are considered to be similar if users who liked the item i and users who liked the item j are same and similarly users who disliked the item i and users who disliked the item j are same and is defined as

$$similarity(i,j) = \frac{n_{i+1,j+1} + n_{i-1,j-1} - n_{i+1,j-1} - n_{i-1,j+1}}{\sqrt{n_{i+1} + n_{i-1}}\sqrt{n_{j+1}n_{j-1}}} \tag{3.21}$$

This is same as the Cosine similarity defined for numerical data sets. In other words, this equation is equivalent to Equation (3.16)

The value lies between [-1,+1]

(iii) Jaccard similarity: Jaccard similarity is defined in the same way as Cosine. The only exception is that in the denominator we consider only the users who consumed items either of i and j or both.

$$similarity(i,j) = \frac{n_{i+1,j+1} + n_{i-1,j-1} - n_{i+1,j-1} - n_{i-1,j+1}}{|U_i \cup U_j|} \tag{3.22}$$

where U_i is the set of users who consumed item i

U_j is the set of users who consumed item j

This function also produces the values that are in between [-1,+1].

- Preference prediction:

  Similar to ordinal data sets, here also k nearest neighbors are selected based on the similarity values computed using any of the above similarity measures. Based on the preferences of the nearest neighbors for the given item i and their similarity values, we can predict preference of the target user a for item i as follows:

- Based on Majority (BoM): Here we count the number of nearest neighbors of item i for which target user a has assigned +1 and the number of nearest neighbors of item i for which target user a has assigned -1. If the number of items with +1 are greater than those of -1, then we predict that the target user a likes the item i otherwise he/she dislikes the item i.

$$preference(a,i) = \begin{cases} +1 & \text{if } n_{+1} > n_{-1} \\ -1 & \text{otherwise} \end{cases} \tag{3.23}$$

where n_{+1} is the number of nearest neighbors liked (+1) by the target user a

n_{-1} is the number of nearest neighbors disliked (-1) by the target user a

- Based on Sum Similarities (BoSS): Here we sum up the similarity values of all the nearest neighbors of item i liked (+1) by the target user a and sum up the similarity values of all the nearest neighbors of item i disliked (-1) by the target user a. Then we compare these two similarity values and assign the label with the highest similarity score.

$$preference(a,i) = \begin{cases} +1 & \text{if } SS_{+1} > SS_{-1} \\ -1 & \text{otherwise} \end{cases} \tag{3.24}$$

where SS_{+1} is the sum of similarity values of all the nearest neighbors liked (+1) by the target user a

SS_{-1} is the sum of similarity values of all the nearest neighbors disliked (-1) by the target user a

- Item recommendation: Generally, in binary settings we recommend all items which are predicted as +1 to the target user a. But in *top-N* setting we have to recommend only N items. In this case, in order to distinguish the items which are predicted as +1, we also associate a score which is the sum of similarity values of all the neighbors liked *(+1)* by the target user a. Based on these scores we can recommend *top-N* items with the highest scores among the items which are predicted as liked *(+1)*.

3. Unary feedback data sets:

- Similarity Computation: Similar to user-based collaborative filtering here also we use Phi Coefficient, Cosine, Jaccard and Dice to compute the similarity between two items in unary feedback data sets.

  (i) Phi coefficient (Pearson): This is similar to phi coefficient defined for binary data sets and is defined as follows:

$$similarity(i,j) = \frac{n_{i1,j1}n_{i0,j0} - n_{i1,j0}n_{i0,j1}}{\sqrt{n_{i1}n_{i0}n_{j1}n_{j0}}} \tag{3.25}$$

  where $n_{i1,j1}$ is the number of users who consumes both items i and j

  $n_{i0,j0}$ is the number of users who did not consume both items i and j

  $n_{i1,j0}$ is the number of users who consumed item i and who did not consume item j

  $n_{i0,j1}$ is the number of users who did not consume item i and who consumed item j

  n_{i1} is the number of users who consumed item i

  n_{i0} is the number of users who did not consume item i

  n_{j1} is the number of users who consumed item j

  n_{j0} is the number of users who did not consume item j

  The above function generates a value in between [-1,+1]

  (ii) Cosine Similarity: This similarity measure is based on the number of common users who consumed both items i and j.

$$similarity(i,j) = \frac{|U_i \cap U_j|}{\sqrt{|U_i|}\sqrt{|U_j|}} \tag{3.26}$$

  where U_i is the set of users who consumed item i

  U_j is the set of users who consumed item j

  This equation is similar to the Cosine similarity function defined for ordinal and binary feedback data sets. In other words, this is equivalent to Equation (3.16).

  This function produces the values that are in between [0,1].

(iii) Jaccard Similarity: This is similar to Cosine similarity measure except that in the denominator we consider all users who consumed atleast one item in $\{i, j\}$

$$similarity(i, j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \tag{3.27}$$

The generated similarity values lies in between [0,1].

(iv) Dice coefficient: This is a variant of Jaccard and is defined as

$$similarity(i, j) = \frac{2 * |U_i \cap U_j|}{|U_i| + |U_j|} \tag{3.28}$$

This function also produces the values that are in between [0,1].

- Score computation: Once the similarities are computed, we extract the k most similar users for the given item i of the target user a and score for the item i is computed by summing up all the similarity values of k nearest neighbors of the given item i of target user a.

$$score(a, i) = \sum_{j=1}^{k} similarity(j, i) \tag{3.29}$$

- Item Recommendation:

  After computing the score for every unused item of the target user we recommend *top-N* items with the highest scores to the target user a.

3.2 Significance Testing

Evaluation measures are important to assess the performance of any recommendation algorithm as well as to compare different algorithms to find performance differences between them. But, in order to determine whether these observed differences are due to luck or really significant we need to perform significance tests. While doing significance tests we generally claim something about the the algorithms to be compared which we call it as the *null hypothesis*. We assume that what we claim about the algorithm is true but has not been proven yet. Alternative hypothesis is the statement

that we accept when null hypothesis is rejected. For example, when comparing two algorithms the null hypothesis is that the performance of these two algorithms is same. In other words, if there is any performance difference between these two algorithms, then this difference is due to luck or by chance. Alternative hypothesis is a statement that believes to be true when the null hypothesis is rejected. That means alternative hypothesis states that these two algorithms do not perform equally, i.e., if one algorithm is performing better than the other then this difference is due to the superiority of the best performing algorithm. A statistical significance test usually generates a value between 0 and 1 and we call it as the p-value. The p-value gives the probability of getting the results (or more extreme results) given that the null hypothesis is true. In order to calculate p-value we use test statistic which is a numerical summary of a data set that reduces the data to one value that can be used to decide whether to reject the null hypothesis or not.

In our experiments in order to determine whether there is any statistical significance among the algorithms, we use Neyman-Pearson approach. In this approach we first specify a significance level α. If the resultant p-value is less than α then our null hypothesis is rejected. Otherwise, we cannot reject the null hypothesis.

We broadly classify significance tests into two categories: parametric and non-parametric. Parametric significance tests make assumptions about the distribution of the measurements. Non-parametric tests make no such assumptions. Significance tests should be chosen based on the samples we are working on. The samples may be independent or dependent. Independent samples are collected from two different sets of items, whereas, dependent samples are paired measurements for one set of items. For example, consider a scenario where we want to test the effectiveness of a new drug on a patient's health condition:

- Independent samples test: Select two groups each with say 100 patients. The first group of patients are given the new drug and give the conventional treatment to other group and check whether the patients' health condition is improved in the first group compared to the second group.

- Dependent samples test: Here, we have only one group of patients. We check the patients' health condition before and after using the drug and check whether the health condition is improved after taking the drug.

In our experiments, we compare the variants of a single algorithm with different prediction techniques and with different similarity measures. We also compare different algorithms which are designed to perform the same task. In all cases, the test set of items for each target user is same for all algorithms and their variants. Therefore, dependent sample significance tests are the appropriate choice for our experiments. In this section, we discuss different significance tests that we use to determine the statistical significance in the performance differences of the algorithms that we are going to compare.

3.2.1 Tests to Compare Means of Dependent Samples

- Comparing two means using paired t-test [43]

  Here, we are testing whether the difference in mean errors produced by two algorithms on the same test set is significant or not. The most popular significance test used for this purpose is paired t-test. The two means are produced by the same algorithm with different prediction techniques or with different similarity measures or by two different algorithms on the same test set of items. The Paired t-test is a parametric test. We can also call this test as Dependent t-Test or Correlated t-test. The Paired t-test can only compare the means for two related groups on a data that is normally distributed. The Paired t-test is not appropriate for comparing unpaired data, comparing more than two groups, data that is not normally distributed and an ordinal/ranked data.

  In paired t-test, the null hypothesis is that the mean errors produced by two algorithms is equal. The alternative hypothesis is that the mean errors produced by two algorithms is not equal. The test statistic for the paired t-test is given by

  $$t = \frac{\overline{x}_{diff} - 0}{s_{\overline{x}}} \tag{3.30}$$

  where

  $$s_{\overline{x}} = \frac{s_{diff}}{\sqrt{n}} \tag{3.31}$$

  where $\overline{x}_{diff}$ is the mean of the differences

  $s_{\overline{x}}$ is the estimated standard error of the mean ($s/\sqrt{n}$)

s_{diff} is the standard deviation of the differences in the test set

n is the number of items in the test set

Under the null hypothesis, this statistic follows a t-distribution with $n-1$ degrees of freedom. Compare this calculated t value with the critical t value with $n-1$ degrees of freedom from the t distribution table for a chosen significance level. If the calculated t value is greater than the critical t value, then we reject the null hypothesis and conclude that the mean errors are significantly different.

- Comparing more than two means using One-Way Repeated Measures ANOVA [44, 46]

  One-Way Repeated Measures ANOVA is an extension of the paired t-test which is used to determine whether the performance differences of three or more algorithms is significant. This is a parametric test. We can also call this test as one-way ANOVA for Dependent samples or one-way ANOVA for Correlated samples or Within subjects ANOVA. In One-Way Repeated Measures ANOVA, the null hypothesis is that the mean errors produced by all algorithms is equal.The alternative hypothesis is that at least one of the mean errors produced by these algorithms is not equal to the others. The test statistic for the One-Way Repeated Measures ANOVA is given by

$$F = \frac{MS_{bg}}{MS_{error}} \tag{3.32}$$

where

$$MS_{bg} = \frac{SS_{bg}}{df_{bg}}$$
$$MS_{error} = \frac{SS_{error}}{df_{error}}$$
$$SS_{bg} = SS_T - SS_{wg}$$
$$SS_T = n * \sum_{i=1}^{g} (\overline{x_{i.}} - \overline{\overline{x}})^2$$
$$\overline{x_{i.}} = \frac{\sum_{j=1}^{n} x_{ij}}{n}$$

$$
\begin{aligned}
\overline{\overline{x}} &= \frac{\sum_{i=1}^{g}\sum_{j=1}^{n} x_{ij}}{g*n} \\
SS_{wg} &= \sum_{i=1}^{k} SS_i \\
SS_i &= \sum_{j=1}^{n}(x_{ij} - \overline{x_{i.}})^2 \\
df_{bg} &= g - 1 \\
SS_{error} &= SS_{wg} - SS_{subjects} \\
SS_{subjects} &= \frac{\sum_{j=1}^{n}(\sum_{i=1}^{k} x_{ji})^2}{g} - \frac{(\sum_{i=1}^{g}\sum_{j=1}^{n} x_{ij})^2}{g*n} \\
df_{error} &= df_{wg} - df_{subjects} \\
df_{wg} &= (g*n) - g \\
df_{subjects} &= n - 1
\end{aligned}
\tag{3.33}
$$

where g is the number of groups

n is the number of subjects

where MS_{bg} is the mean sum of squares between the groups

MS_{error} mean square of the error

SS_{bg} is the sum of squares for the differences between the groups

df_{bg} is the degrees of freedom for the group variable

SS_{error} is the sum of squares for the error

df_{error} is the degrees of freedom for error

SS_T is the total sum of squares

$\overline{x_{i.}}$ mean for group i

$\overline{\overline{x}}$ mean for total

SS_{wg} is the sum of squares for the differences within groups

SS_i is the sum of squares for group i

$SS_{subjects}$ is the sum of squares for the differences among n subjects

df_{wg} is the degrees of freedom for all the groups

$df_{subects}$ is the degrees of freedom for subjects within a group

This F-statistic is a ratio of the variability between groups compared to the variability within the groups. F value will be close to 1 if these two variances are equal. In repeated measures ANOVA we need not consider the variability that exists inside the *g* groups which reflects pre-existing individual differences among the subjects. A significant F-ratio tells us only that the aggregate difference among the means of the several groups is significantly greater than zero. It does not tell us which group mean is significantly different from which other group mean. In order to find which group means are different from others we use Tukey HSD Test.

Tukey HSD (honestly significant difference) Test:

The Tukey test revolves around a measure known as the Studentized range statistic, which we will abbreviate as Q. For any particular pair of means among the *g* groups, let us designate the larger and smaller as M_L and M_S respectively. The Studentized range statistic can then be calculated for any particular pair as

$$Q = \frac{M_L - M_S}{\sqrt{\frac{MS_{error}}{n}}} \tag{3.34}$$

Then find the critical Q value at the given significance level. This value gives how large the difference between the means of any two particular groups must be in order to be regarded as significant. In order to be considered significant at or beyond the 0.05 level, the difference between any two particular group means (larger - smaller) must be equal to or greater than $HSD_{0.05}$.

$$HSD_{0.05} = Q_{0.05} * \sqrt{\frac{MS_{error}}{n}} \tag{3.35}$$

3.2.2 Tests to Compare Proportions of Dependent Samples

- Comparing two proportions using Mcnemar's test [45, 77]

	Algorithm 2 positive	Algorithm 2 negative	Row total
Algorithm 1 positive	a	b	a+b
Algorithm 1 negative	c	d	c+d
Column total	a+c	b+d	n

Table 3.1: Contingency Table

McNemar's test is a statistical test used to determine the significance of the difference between two correlated proportions. It is a non-parametric test used on paired nominal data. McNemar's test is applied to $2X2$ contingency tables where our data is represented as cell frequencies. Each cell frequency value gives the number of test items satisfying the conditions in the the corresponding row and column. The $2X2$ contingency table, which tabulates the outcomes of two algorithms on a total of n test items is shown in 3.1.

In McNemar's test we check whether the row and column marginal frequencies are equal i.e., we are checking whether there is marginal homogeneity. The null hypothesis of marginal homogeneity states that the two marginal probabilities for each outcome are the same, i.e. $p_a + p_b = p_a + p_c$ and $p_c + p_d = p_b + p_d$.

Thus the null and alternative hypotheses are

$p_b = p_c$ and

$p_b \neq p_c$

The McNemar test statistic is:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{3.36}$$

Under the null hypothesis, with a sufficiently large number of discordants (cells b and c), χ^2 has a chi-squared distribution with 1 degree of freedom. If the χ^2 result is significant, this provides sufficient evidence to reject the null hypothesis, in favour of the alternative hypothesis that $p_b \neq p_c$. This would mean that the marginal proportions are significantly different from each other.

- Comparing more than two means using Cochran's Q test [7, 76]

  Cochran's Q test is an extension to the McNemar's test for correlated samples which is used to test whether the differences between three or more propor-

tions is significant. Cochran's Q test is a non-parametric statistical test to verify whether *g* groups have identical effects. In Cochran's Q test we have only binary response like *success/failure* or *1/0*. The test assesses whether the proportion of successes is the same between groups. Thus, the null hypothesis is that the number of successes in all groups are same and the alternative hypothesis is that there is a difference in the number of successes between groups.

The test statistic for Cochran's Q test is

$$Q = g(g-1)\frac{\sum_{i=1}^{g}(X_{.i} - \frac{X_{..}}{g})^2}{\sum_{j=1}^{n} X_{j.}(g - X_{j.})} \tag{3.37}$$

where g is the number of groups

n is the number of subjects

$X_{.i}$ column total for the i^{th} group

$X_{..}$ grand total

$X_{j.}$ row total for the j^{th} subject

For large samples, the test statistic, Q is distributed as chi-square with $g-1$ degrees of freedom. As in the McNemar's test, only subjects who do not have the same response in all categories contribute to the overall Q statistic.

For significance level α, the critical region is

$Q > \chi^2_{1-\alpha,g-1}$

$\chi^2_{1-\alpha,g-1}$ is the $(1-\alpha)$-quantile of the chi-squared distribution with $g-1$ degree of freedom. The null hypothesis is rejected if the test statistic is in the critical region. If the Cochran's Q test rejects the null hypothesis, pairwise multiple comparisons can be made by applying Mcnemar's test on the two groups of interest.

3.3 Empirical Evaluation of Neighborhood-based Algorithms for Prediction and Recommendation Tasks

3.3.1 Experimental Setup

We tested our algorithms on four data sets: MovieLens 100K, EachMovie, MovieLens 1M and MovieLens-latest-small. We randomly selected 50 users and for each user 60% of the data is considered as training set and remaining 40% is taken as the test set. The first, third and fourth data sets contain ratings from 1-5 and the second data set consists of ratings from 1-6. We tested our algorithms for neighbourhood size = 20. In other words, maximum number of neighbours we considered in our experiments is 20. We also perform the significance tests to determine the statistical significance between the performance differences of different algorithms at 0.05 significance level.

The problem with similarity measures in neighbourhood-based algorithms is that they produce high correlation between two users or items even when they share only a few number of items or users. As a result such users or items become part of the neighborhood for the candidate user or item and thus can be used to predict the ratings or scores for the given item of the target user. But, such predictions are going to be inaccurate as these predictions are based not on the truly correlated neighbors. In [24], authors have used significance weighing to improve the accuracy by reducing the similarity weights that were based on a small number of co-rated items or users. Similar to [24], we apply significance weight $n/50$, if two users or items share less than 50 items or users, where n is the number of corated items or users. A significance weight of 1 is used if the number of items or users between two users or items is greater than or equal to 50. In addition to this, we also consider only those users or items as neighbors whose similarity score is greater than zero.

We tested our algorithms at different *top-N* values $\{10, 20, 30, 40, 50\}$ while recommending a fixed number of items to users and the maximum number of neighbors is 20. One problem with recommendation task in numerical data sets is selecting the appropriate test set. We cannot select the test items randomly for every user for recommendation task as we do in the case of prediction task. Because in recommendation task we generally recommend items with highly predicted ratings like 4 and 5 in a data set with a rating scale of 1-5. Therefore, our test set in Movielens 100K, Movielens

Dataset	Performance Measure	User-based Nearest Neighbors Algorithm					
		Pearson			Cosine		
		SWA	ADM	Significant?	SWA	ADM	Significant?
Movielens 100K	MAE	0.7627	**0.6918**	**YES**	0.7933	**0.7174**	**YES**
	RMSE	0.9595	**0.8926**		0.9908	**0.9261**	
	Coverage	0.997	0.997		0.9976	0.9976	
Eachmovie	MAE	1.0613	**0.9579**	**YES**	1.1116	**0.9726**	**YES**
	RMSE	1.3584	**1.2419**		1.3958	**1.2344**	
	Coverage	1	1		1	1	
Movielens 1M	MAE	0.7289	**0.6516**	**YES**	0.7435	**0.6666**	**YES**
	RMSE	0.9269	**0.8768**		0.9477	**0.8956**	
	Coverage	1	1		1	1	
Movielens-latest-small	MAE	0.6915	**0.6432**	**YES**	0.6986	**0.6513**	**YES**
	RMSE	0.8869	**0.8294**		0.8948	**0.8382**	
	Coverage	0.9789	0.9789		0.9817	0.9817	

Table 3.2: Comparison of prediction techniques

1M and Movielens-small-latest should consists of only those items with ratings 4 or 5. In case of Eachmovie where the rating scale is in the range of 1-6, test set contains the items with ratings 5 and 6. Otherwise we might end up with very low precision and recall values. Similarly, for binary data sets the test set for a user should consists of only those items that are liked by the user. To be compatible with the test sets in numerical data sets we use the following approach to convert our numerical data sets into binary: All the ratings above the midpoint of the rating scale in the corresponding data set are considered as liked items (+1) and all other ratings are disliked items (-1). If the midpoint results in a floating point number then round it to the next integer value.

As a result items with ratings 4 and 5 are considered as liked (+1) items and items with ratings 1,2 and 3 are treated as disliked (-1) in Movielens 100K, Movielens 1M and Movielens-latest-small. Where as, in Eachmovie data set we consider the items with ratings 5 and 6 as liked (+1) and the items with 1, 2, 3 and 4 as disliked (-1).

3.3.2 Experimental Results

1. Prediction Task:

- **Numerical Feedback**

  User-based Algorithm

  Table 3.2 compares the predictive performance of different prediction techniques of user-based algorithm with different similarity measures for numerical data in

Dataset	Performance Measure	User-based Nearest Neighbors Algorithm		
		Pearson	Cosine	Significant?
Movielens 100K	MAE	**0.6918**	0.7174	**YES**
	RMSE	**0.8926**	0.9262	
	Coverage	0.997	0.997	
Eachmovie	MAE	**0.9579**	0.9726	NO
	RMSE	1.2419	**1.2344**	
	Coverage	1	1	
Movielens 1M	MAE	**0.6516**	0.6666	**YES**
	RMSE	**0.8768**	0.8956	
	Coverage	1	1	
Movielens-latest-small	MAE	**0.6432**	0.6505	NO
	RMSE	**0.8294**	0.8369	
	Coverage	0.9789	0.9789	

Table 3.3: Comparison of similarity measures

terms of MAE and RMSE for all four data sets. We also perform significance testing to measure the statistical significance between two prediction techniques i.e., SWA and ADM for each of the similarity measures. For this, we use paired t-test. From Table 3.2 we can observe that ADM produces more accurate predictions compared to SWA for all similarity measures and this performance difference is statistically significant at 0.05 significance level. This is true for all data sets. Therefore, ADM is the best prediction technique for user-based algorithm for all data sets.

In Table 3.3 we compare the performance of different similarity measures (Pearson and Cosine with ADM as the prediction technique) of user-based algorithm. Here also, we use paired t-test to perform significance testing. From Table 3.3 we can observe that Pearson clearly outperforms Cosine for all data sets. This difference is significant only for Movielens 100K and Movielens 1M data sets. For Eachmovie and Movielens-latest-small the difference between Pearson and Cosine is not statistically significant. From this discussion, it is clear that the performance of user-based algorithm with Pearson and Cosine is not same for all data sets and Pearson is preferable to Cosine for user-based algorithm.

Item-based Algorithm

Table 3.4 compares the performance of SWA and ADM for item-based algorithm with different similarity measures for all data sets and used paired t-test for significance testing. Like in user-based algorithm, here too ADM outperforms SWA

Dataset	Performance Measure	Item-based Nearest Neighbors Algorithm								
		Pearson			Cosine			Adjusted Cosine		
		SWA	ADM	Significant?	SWA	ADM	Significant?	SWA	ADM	Significant?
Movielens 100K	MAE	0.7373	**0.6939**	**YES**	0.75	**0.7158**	**YES**	0.6958	**0.6927**	NO
	RMSE	0.9337	**0.8851**		0.9498	**0.9006**		0.89	**0.885**	
	Coverage	0.9953	0.9953		0.9976	0.9976		0.9976	0.9976	
Eachmovie	MAE	0.991	**0.9365**	**YES**	1.038	**0.9596**	**YES**	0.9436	**0.9247**	**YES**
	RMSE	1.2575	**1.2042**		1.3259	**1.2367**		1.2218	**1.1977**	
	Coverage	1	1		1	1		1	1	
Movielens 1M	MAE	0.7036	**0.6407**	**YES**	0.6914	**0.6428**	**YES**	0.6604	**0.6441**	**YES**
	RMSE	0.8968	**0.8414**		0.8999	**0.8388**		0.8631	**0.8381**	
	Coverage	1	1		1	1		0.9988	0.9988	
Movielens-latest-small	MAE	0.6663	**0.6305**	**YES**	0.6622	**0.6345**	**YES**	**0.6295**	0.6346	NO
	RMSE	0.8542	**0.81**		0.853	**0.8173**		**0.8092**	0.8149	
	Coverage	0.9552	0.9552		0.9817	0.9817		0.9817	0.9817	

Table 3.4: Comparison of Prediction Techniques

Dataset	Performance Measure	Item-based Nearest Neighbors Algorithm			
		Pearson	Cosine	Adjusted Cosine	Significant?
Movielens 100K	MAE	0.6939	0.7149	**0.6922**	**YES, M1 vs M2, M2 vs M3**
	RMSE	0.8851	0.8999	**0.8846**	
	Coverage	0.9953	0.9953	0.9953	
Eachmovie	MAE	0.9365	0.9596	**0.9247**	**YES, M1 vs M2, M2 vs M3**
	RMSE	1.2042	1.2367	**1.1977**	
	Coverage	1	1	1	
Movielens 1M	MAE	0.6403	0.6424	**0.6441**	NO
	RMSE	0.8412	0.8385	**0.8381**	
	Coverage	0.9988	0.9988	0.9988	
Movielens-latest-small	MAE	0.6305	0.63	**0.6282**	NO
	RMSE	0.81	0.8096	**0.8048**	
	Coverage	0.9552	0.9552	0.9552	

Table 3.5: Comparison of similarity measures

and this difference is significant for all data sets and for all similarity measures except Adjusted Cosine. For Adjusted Cosine, ADM shows better performance than SWA for all data sets except movielens-latest-small for which SWA performs better than ADM. The difference between ADM and SWA is small for Movielens 100K and Movielens-latest-small data sets and this difference is not significant. The difference is significant for Eachmovie and Movielens 1M data sets.

In Table 3.5 we compare the performance of three similarity measures Pearson, Cosine and Adjusted Cosine (with ADM as the prediction technique) for item-based algorithm. We use one-way repeated measures ANOVA to determine the significance between the performance differences of different similarity mea-

Dataset	Performance Measure	Algorithm		
		User-based Pearson	Item-based Adjusted Cosine	Significant?
Movielens 100K	MAE	0.6918	**0.693**	NO
	RMSE	0.8926	**0.8852**	
	Coverage	0.997	0.997	
Eachmovie	MAE	0.9579	**0.9247**	**YES**
	RMSE	1.2419	**1.1977**	
	Coverage	1	1	
Movielens 1M	MAE	0.6506	**0.6441**	NO
	RMSE	0.8756	**0.8381**	
	Coverage	0.9988	0.9988	
Movielens-latest-small	MAE	0.6432	**0.6334**	NO
	RMSE	0.8294	**0.8126**	
	Coverage	0.9789	0.9789	

Table 3.6: Comparison of User-based and Item-based Algorithms

sures. The following observations are made:

- For all data sets, though, Adjusted Cosine outperforms Pearson and Cosine similarity measures, the performance difference between Pearson and Adjusted Cosine is very small and is not statistically significant.
- For Movielens 100K and Eachmovie data sets, both Pearson and Adjusted Cosine outperform Cosine and this difference is significant.
- For Movielens 1M and Movielens-latest-small data sets, the performance difference between all three similarity measures is very small and is not statistically significant.

From this discussion, Pearson and Adjusted Cosine are the appropriate similarity measures for item-based algorithm. The behaviour of these measures is not same for all data sets. They are showing similar behavior for Movielens 100K and Eachmovie where they both are significantly outperforming Cosine and for Movielens 1M and Movielens-latest-small where their performance is very close to Cosine with no statistical significance.

User-based Algorithm vs Item-based Algorithm

From Tables 3.3 and 3.5 we can see that Pearson is the best similarity measure for user-based algorithm and both Pearson and Adjusted Cosine are the best similarity measures for item-based algorithm. In Table 3.6, we compare

the performance of user-based algorithm with Pearson and item-based algorithm with Adjusted Cosine (we can also take Pearson). As can be seen from the table, it is clear that the performance of both algorithms is almost similar and the difference between them is not significant for all data sets except Eachmovie. For Eachmovie data set, item-based algorithm is clearly outperforming user-based and this difference is statistically significant. That means for numerical data sets both user-based and item-based algorithms show almost similar performance with no statistical significance for Movielens 100K, Movielens 1M and Movielens-latest-small. But for Eachmovie, item-based algorithm gives significantly better performance than user-based algorithm and therefore is more preferable.

In summary, for prediction task in numerical data sets

- Prediction techniques: ADM performs well compared to SWA for both user-based and item-based algorithms (Tables 3.2 and 3.4).
- Similarity measures: We use ADM to produce predictions in both algorithms as it shows better performance compared to SWA. Pearson gives better performance compared to Cosine in user-based algorithm (Table 3.3) and both Pearson and Adjusted Cosine perform well compared to Cosine in item-based algorithm (Table 3.5).
- User-based vs Item-based: Item-based algorithm with Adjusted Cosine performs well compared to user-based algorithm with Pearson (Table 3.6).

- **Binary Feedback**

  User-based Algorithm

  Table 3.7 compares the predictive ability of the two prediction techniques used for binary data sets (BoM and BoSS) of user-based algorithm with different similarity measures in terms of percentage of correctly classified test items (PCCTI) for all data sets. To determine the statistical significance between the two techniques we use Mcnemar's test. From the table, we can see that the performance difference between the two techniques is very small and is also not statistically significant for all data sets and for all similarity measures except the Cosine similarity measure of Movielens 1M for which this difference is significant.

Dataset	Performance Measure	User-based Nearest Neighbors Algorithm								
		Pearson			Cosine			Jaccard		
		BoM	BoSS	Significant?	BoM	BoSS	Significant?	BoM	BoSS	Significant?
Movielens 100K	PCCTI	0.6522	**0.6582**	NO	0.6682	**0.6742**	NO	0.6706	**0.6783**	NO
	Coverage	0.9959	0.9959		0.9959	0.9959		0.9959	0.9959	
Eachmovie	PCCTI	0.6701	**0.6753**	NO	0.6798	**0.682**	NO	**0.6857**	0.685	NO
	Coverage	1	1		1	1		1	1	
Movielens 1M	PCCTI	0.6677	**0.6689**	NO	**0.6859**	0.6786	**YES**	**0.6911**	0.6855	NO
	Coverage	1	1		1	1		1	1	
Movielens-latest-small	PCCTI	0.6516	**0.6562**	NO	0.6596	**0.6608**	NO	0.6602	**0.6634**	NO
	Coverage	0.9758	0.9758		0.9766	0.9766		0.9766	0.9766	

Table 3.7: Comparison of Prediction techniques

Dataset	Performance Measure	User-based Nearest Neighbors Algorithm			
		Peason	Cosine	Jaccard	Significant?
Movielens 100K	PCCTI	0.6578	0.6744	**0.6786**	**YES, CC1 vs CC2, CC1 vs CC3**
	Coverage	0.9947	0.9947	0.9947	
Eachmovie	PCCTI	0.6753	0.682	**0.685**	NO
	Coverage	1	0.9947	0.9947	
Movielens 1M	PCCTI	0.6689	0.6786	**0.6855**	**YES, CC1 vs CC3**
	Coverage	1	0.9947	0.9947	
Movielens-latest-small	PCCTI	0.6589	0.661	**0.6636**	NO
	Coverage	0.9746	0.9947	0.9947	

Table 3.8: Comparison of Similarity Measures

In Table 3.8, we show the performance difference between various similarity measures (with BoSS as the prediction technique (we can also use BoM)) of user-based algorithm and we use Cochran's Q test to determine the statistical significance and the following points are observed:

- Though, the performance difference between Cosine and Jaccard is small-for all data sets, this difference is significant for Movielens 100K. Both Cosine and Jaccard outperform Pearson for all data sets. But the difference between Pearson and Cosine is significant only for Movielens 100K and the difference between Pearson and Jaccard is significant for both Movielens 100K and Movielens 1M.
- For Eachmovie and Movielens-latest-small, the difference between all three similarity measures is not significant.

That means, here also, all three measures are not performing in the same way for all data sets. Cosine and Jaccard are good for Movielens 100K and Movielens

Dataset	Performance Measure	Item-based Nearest Neighbors Algorithm								
		Pearson			Cosine			Jaccard		
		BoM	BoSS	Significant?	BoM	BoSS	Significant?	BoM	BoSS	Significant?
Movielens 100K	PCCTI	0.6547	**0.6649**	NO	0.6736	**0.6819**	NO	0.6765	**0.6813**	NO
	Coverage	0.9894	0.9894		0.9976	0.9976		0.9976	0.9976	
Eachmovie	PCCTI	0.6597	**0.6761**	NO	0.6805	**0.682**	NO	0.6761	**0.6812**	NO
	Coverage	1	1		1	1		1	1	
Movielens 1M	PCCTI	0.6708	**0.6821**	NO	**0.6865**	0.6857	**NO**	0.6865	**0.689**	NO
	Coverage	0.9996	0.9996		0.9996	0.9996		0.9996	0.9996	
Movielens-latest-small	PCCTI	**0.652**	0.6499	NO	0.6702	**0.6728**	NO	0.6762	**0.6803**	NO
	Coverage	0.9228	0.9228		0.9814	0.9814		0.9814	0.9814	

Table 3.9: Comparison of Prediction Techniques

Dataset	Performance Measure	Item-based Nearest Neighbors Algorithm			
		Peason	Cosine	Jaccard	Significant?
Movielens 100K	PCCTI	0.6649	0.6816	**0.6822**	**YES**
	Coverage	0.9894	0.9894	0.9894	
Eachmovie	PCCTI	0.6761	**0.682**	0.6812	NO
	Coverage	1	1	1	
Movielens 1M	PCCTI	0.6824	0.6856	**0.6888**	NO
	Coverage	0.9992	0.9992	0.9992	
Movielens-latest-small	PCCTI	0.6498	0.6708	**0.6773**	**YES, CC1 vs CC2, CC1 vs CC3**
	Coverage	0.9225	0.9225	0.9225	

Table 3.10: Comparison of Similarity Measures

1M, whereas, all three similarity measures perform in the similar fashion for Eachmovie and Movielens-latest-small.

Item-based Algorithm

Table 3.9 compares the predictive ability of BoM and BoSS of item-based algorithm with different similarity measures for all data sets. To determine the statistical significance between the two techniques we use Mcnemar's test. Like in user-based algorithm, here also, the performance difference between the two prediction techniques is very small and is also not statistically significant.

In Table 3.10, we show the performance difference between various similarity measures (with BoSS as the prediction technique (we can also use BoM)) of item-based algorithm and we use Cochran's Q test to determine the statistical significance and we observe the following:

– The performance difference between Cosine and Jaccard is small and is also not statistically significant. Though, both these measures outperform

Dataset	Performance Measure	Algorithm		
		User-based Jaccard	Item-based Jaccard	Significant?
Movielens 100K	PCCTI	0.6783	**0.6825**	NO
	Coverage	0.9959	0.9959	
Eachmovie	PCCTI	**0.685**	0.6812	NO
	Coverage	1	1	
Movielens 1M	PCCTI	0.6853	**0.689**	NO
	Coverage	0.9996	0.9996	
Movielens-latest-small	PCCTI	0.6633	**0.6803**	**YES**
	Coverage	0.9763	0.9763	

Table 3.11: Comparison of User-based and Item-based Algorithms

Pearson for all data sets, this difference is significant only for Movielens-latest-small.

Therefore, Cosine and Jaccard are more preferable than Pearson for item-based algorithm.

User-based Algorithm vs Item-based Algorithm

As we have seen from the Tables 3.8 and 3.10, Cosine and Jaccard are best measures compared to Pearson for both user-based and item-based algorithms. Table 3.11 compares the performance of user-based algorithm and item-based algorithms with Jaccard similarity measure (we can also use Cosine). For all data sets except Movielens-latest-small, the difference between user-based and item-based algorithm is very small and this difference is not significant. For Movielens-latest-small, item-based algorithm is showing better performance than user-based algorithm and this difference is statistically significant.

In summary, for prediction task in binary data sets

- Prediction techniques: Both BoM and BoSS perform equally for both user-based and item-based algorithms (Tables 3.7 and 3.9).
- Similarity measures: We use BoSS to produce predictions in both algorithms. Cosine and Jaccard show better performance compared to Pearson in both user-based algorithm (Table 3.8) and item-based algorithm (Table 3.10).
- User-based vs Item-based: Item-based algorithm with Jaccard performs well compared to user-based algorithm with Jaccard (Table 3.11).

2. Recommendation Task:

- **Numerical Feedback**

  In order to recommend items to users in numerical data sets, we first need to predict the ratings for each unknown item of the user. Based on the predicted ratings we recommend *top-N* items with highest predicted ratings to the user. As discussed above, two prediction techniques used in neighborhood-based algorithms are SWA and ADM.

  User-based Algorithm

  Table 3.12 compares the recommendation accuracy of SWA and ADM for user-based algorithm with different similarity measures in terms of precision and recall. We use paired t-test to determine whether the performance differences between these two techniques is significant.

  – For Movielens 100K, though SWA outperforms ADM in most of the cases, the performance difference between ADM and SWA is small but is significant in nearly half of the cases.

  – For Eachmovie data set ADM is giving better performance than SWA but this difference is not significant.

  – For Movielens 1M data set ADM is significantly outperforming SWA.

  – For Movielens-latest-small, though ADM is performing better than SWA, the difference between these two prediction techniques is small and is also not significant.

  From this discussion, it is clear that ADM is preferable to SWA.

  Table 3.13 compares the performance of various similarity measures (with ADM as the prediction technique) of user-based algorithm. Here also, we use paired t-test for significance testing and the following observations are made for different data sets:

  – For Movielens 100K and Movielens 1M, Cosine shows good performance compared to Pearson. But this difference is not significant for Movielens 100K data set, whereas this difference is significant for Movielens 1M data set.

Dataset	Performance Measure	TOP-N	User-based Nearest Neighbors Algorithm					
			Pearson			Cosine		
			SWA	ADM	Significant?	SWA	ADM	Significant?
Movielens 100K	Precision	10	0	**0.006**	NO	0.002	**0.014**	**YES**
		20	0.026	**0.028**	NO	**0.043**	0.036	NO
		30	**0.0553**	0.0393	**YES**	**0.074**	0.0487	**YES**
		40	**0.073**	0.0565	**YES**	**0.0855**	0.0605	**YES**
		50	**0.078**	0.0616	**YES**	**0.0932**	0.0648	**YES**
	Recall	10	0	**0.0016**	NO	0.0008	**0.0045**	NO
		20	0.0101	**0.0125**	NO	**0.0272**	0.0167	NO
		30	**0.0419**	0.0308	NO	**0.0569**	0.0433	NO
		40	**0.0776**	0.0642	**YES**	**0.0948**	0.0704	**YES**
		50	**0.1066**	0.089	**YES**	**0.127**	0.0929	**YES**
	Coverage		0.962	0.962		0.9976	0.9976	
Eachmovie	Precision	10	0.112	**0.142**	NO	0.136	**0.162**	NO
		20	0.107	**0.124**	NO	0.124	**0.125**	NO
		30	0.0973	**0.1073**	NO	0.1073	**0.1107**	NO
		40	0.0915	**0.0965**	NO	0.0945	**0.095**	NO
		50	0.084	**0.0904**	NO	0.0848	**0.0868**	NO
	Recall	10	0.0468	**0.0604**	NO	0.0589	**0.0622**	NO
		20	0.0897	**0.1085**	NO	**0.0992**	0.0889	NO
		30	0.1219	**0.1384**	NO	**0.1262**	0.1256	NO
		40	0.1507	**0.1649**	NO	**0.15**	0.1412	NO
		50	0.1771	**0.1884**	NO	**0.1672**	0.159	NO
	Coverage		0.9985	0.9985		0.9996	0.9996	
Movielens 1M	Precision	10	0.008	**0.078**	**YES**	0.002	**0.148**	**YES**
		20	0.039	**0.091**	**YES**	0.067	**0.141**	**YES**
		30	0.0673	**0.1007**	**YES**	0.1027	**0.1393**	**YES**
		40	0.075	**0.0985**	**YES**	0.1135	**0.133**	**YES**
		50	0.0788	**0.1004**	**YES**	0.1124	**0.1284**	**YES**
	Recall	10	0.0015	**0.0151**	**YES**	0.0009	**0.0285**	**YES**
		20	0.0168	**0.0409**	**YES**	0.0347	**0.0562**	**YES**
		30	0.0441	**0.0646**	**YES**	0.0702	**0.0874**	**YES**
		40	0.0676	**0.0846**	**YES**	0.1052	**0.1132**	NO
		50	0.0878	**0.1083**	**YES**	0.1286	**0.1379**	NO
	Coverage		0.9214	0.9214		0.9346	0.9346	
Movielens-latest-small	Precision	10	0.002	**0.002**	NO	**0.002**	**0.002**	NO
		20	0.001	**0.003**	NO	0.001	**0.003**	NO
		30	0.0027	**0.0033**	NO	0.0013	**0.004**	NO
		40	0.0025	**0.004**	NO	0.0015	**0.0035**	NO
		50	0.002	**0.0036**	NO	0.0012	**0.0036**	NO
	Recall	10	0.0009	**0.0013**	NO	0.0009	**0.0013**	NO
		20	0.0009	**0.0029**	NO	0.0009	**0.0045**	NO
		30	0.0047	**0.0051**	NO	0.0017	**0.0071**	NO
		40	0.0051	**0.0077**	NO	0.004	**0.0073**	NO
		50	0.0051	**0.008**	NO	0.004	**0.0077**	NO
	Coverage		0.8792	0.8792		0.9773	0.9773	

Table 3.12: Comparison of Prediction Techniques

Dataset	Performance Measure	TOP-N	User-based Nearest Neighbors Algorithm		
			Pearson	Cosine	Significant?
Movielens 100K	Precision	10	0.006	**0.014**	NO
		20	0.028	**0.036**	NO
		30	0.0393	**0.0487**	NO
		40	0.0565	**0.0605**	NO
		50	0.0616	**0.0648**	NO
	Recall	10	0.0016	**0.0045**	NO
		20	0.0125	**0.0167**	NO
		30	0.0308	**0.0433**	NO
		40	0.0642	**0.0704**	NO
		50	0.089	**0.0929**	NO
	Coverage		0.962	0.9976	
Eachmovie	Precision	10	0.142	**0.162**	NO
		20	0.124	**0.125**	NO
		30	0.1073	**0.1107**	NO
		40	**0.0965**	0.095	NO
		50	**0.0904**	0.0868	NO
	Recall	10	0.0604	**0.0622**	NO
		20	**0.1085**	0.0889	NO
		30	**0.1384**	0.1256	NO
		40	**0.1649**	0.1412	NO
		50	**0.1884**	0.159	NO
	Coverage		0.9985	0.9996	
Movielens 1M	Precision	10	0.078	**0.148**	**YES**
		20	0.091	**0.141**	**YES**
		30	0.1007	**0.1393**	**YES**
		40	0.0985	**0.133**	**YES**
		50	0.1004	**0.1284**	**YES**
	Recall	10	0.0151	**0.0285**	**YES**
		20	0.0409	**0.0562**	NO
		30	0.0646	**0.0874**	**YES**
		40	0.0846	**0.1132**	**YES**
		50	0.1083	**0.1379**	**YES**
	Coverage		0.9214	0.9346	
Movielens-latest-small	Precision	10	**0.002**	**0.002**	NO
		20	**0.003**	**0.003**	NO
		30	0.0033	**0.004**	NO
		40	**0.004**	0.0035	NO
		50	**0.0036**	**0.0036**	NO
	Recall	10	**0.0013**	**0.0013**	NO
		20	0.0029	**0.0045**	NO
		30	0.0051	**0.0071**	NO
		40	**0.0077**	0.0073	NO
		50	**0.008**	0.0077	NO
	Coverage		0.8792	0.9773	

Table 3.13: Comparison of Similarity Measures

– For Eachmovie and Movielens-latest-small, the performance difference between these two similarity measures is small in many cases and is also not significant in all cases.

Therefore, Cosine is preferable to Pearson for user-based algorithm for recommendation task.

Item-based Algorithm

In Table 3.14, we compare the performance of SWA and ADM for item-based algorithm with different similarity measures. For all data sets, ADM outperforms SWA and this difference is statistically significant in most of the cases for Movielens 100K and Movielens 1M and in all cases for Eachmovie. It is significant in very few cases (Pearson) for Movielens-latest-small.

Table 3.15 shows the comparison of performance of different similarity measures (with ADM as the prediction technique) of item-based algorithm. Here, we use one way ANOVA for significance testing and we observe the following:

– For Movielens 100K, Pearson is outperforming Cosine and Adjusted Cosine and the difference between Pearson and Adjusted Cosine as well as Pearson and Cosine is significant in most of the cases. But the difference between Cosine and Adjusted Cosine is not significant.

– For Eachmovie, though Pearson shows good performance compared to Cosine and Adjusted Cosine in nearly all cases, this performance difference is very small and is not significant. In other words, all three similarity measures are giving similar performance.

– For Movielens 1M, though the performance difference between Pearson and Cosine is small, this difference is significant in half of the cases. But the performance difference of Adjusted Cosine with Pearson is significant in most of the cases and with Cosine is not significant in all cases.

– For Movielens-latest-small, though Pearson is outperforming Cosine and Adjusted Cosine in all cases, this difference is significant only in very few cases.

Dataset	Performance Measure	TOP-N	Item-based Nearest Neighbors Algorithm								
			Pearson			Cosine			Adjusted Cosine		
			SWA	ADM	Significant?	SWA	ADM	Significant?	SWA	ADM	Significant?
Movielens 100K	Precision	10	0.006	**0.058**	**YES**	0.002	**0.006**	NO	**0.006**	**0.006**	NO
		20	0.008	**0.063**	**YES**	0.001	**0.031**	**YES**	0.013	**0.027**	NO
		30	0.0107	**0.0813**	**YES**	0.0007	**0.052**	**YES**	0.0133	**0.038**	**YES**
		40	0.0095	**0.082**	**YES**	0.0015	**0.0635**	**YES**	0.0185	**0.0495**	**YES**
		50	0.0112	**0.082**	**YES**	0.002	**0.0704**	**YES**	0.02	**0.056**	**YES**
	Recall	10	0.0005	**0.016**	**YES**	0.0002	**0.0007**	NO	0.0006	**0.0011**	NO
		20	0.0047	**0.0302**	**YES**	0.0002	**0.0131**	**YES**	0.0055	**0.0123**	NO
		30	0.0076	**0.0641**	**YES**	0.0002	**0.0433**	**YES**	0.0072	**0.0248**	**YES**
		40	0.0098	**0.0879**	**YES**	0.0008	**0.0708**	**YES**	0.015	**0.0478**	**YES**
		50	0.0139	**0.106**	**YES**	0.0012	**0.0979**	**YES**	0.0204	**0.0688**	**YES**
	Coverage		0.8924	0.8924		0.992	0.992		0.9725	0.9725	
Eachmovie	Precision	10	0.01	**0.082**	**YES**	0.006	**0.068**	**YES**	0.016	**0.072**	**YES**
		20	0.027	**0.082**	**YES**	0.017	**0.067**	**YES**	0.022	**0.076**	**YES**
		30	0.0273	**0.0727**	**YES**	0.0167	**0.0613**	**YES**	0.0247	**0.0633**	**YES**
		40	0.025	**0.068**	**YES**	0.015	**0.055**	**YES**	0.0265	**0.0595**	**YES**
		50	0.0276	**0.0624**	**YES**	0.014	**0.0528**	**YES**	0.0252	**0.0568**	**YES**
	Recall	10	0.0024	**0.037**	**YES**	0.0035	**0.0303**	**YES**	0.0044	**0.0323**	**YES**
		20	0.0201	**0.0658**	**YES**	0.0202	**0.0581**	**YES**	0.0137	**0.0658**	**YES**
		30	0.0311	**0.0892**	**YES**	0.0294	**0.0831**	**YES**	0.0276	**0.0799**	**YES**
		40	0.0374	**0.1102**	**YES**	0.0334	**0.0958**	**YES**	0.0382	**0.1015**	**YES**
		50	0.0502	**0.1272**	**YES**	0.0393	**0.1085**	**YES**	0.045	**0.1172**	**YES**
	Coverage		0.9972	0.9972		0.9995	0.9995		0.9847	0.9847	
Movielens 1M	Precision	10	0.006	**0.03**	**YES**	0	**0.016**	NO	0.006	**0.022**	**YES**
		20	0.004	**0.049**	**YES**	0	**0.041**	**YES**	0.007	**0.024**	**YES**
		30	0.0047	**0.0587**	**YES**	0	**0.0547**	**YES**	0.0093	**0.0273**	**YES**
		40	0.004	**0.057**	**YES**	0.0005	**0.0605**	**YES**	0.0115	**0.0405**	**YES**
		50	0.0036	**0.062**	**YES**	0.0004	**0.0604**	**YES**	0.0116	**0.0392**	**YES**
	Recall	10	0.0011	**0.0075**	**YES**	0	**0.002**	NO	0.0009	**0.0037**	**YES**
		20	0.0012	**0.0209**	**YES**	0	**0.0184**	**YES**	0.0018	**0.0073**	**YES**
		30	0.002	**0.0351**	**YES**	0	**0.0358**	**YES**	0.0033	**0.0139**	**YES**
		40	0.0022	**0.0479**	**YES**	0. 0005	**0.0527**	**YES**	0.0051	**0.0302**	**YES**
		50	0.0026	**0.0632**	**YES**	0. 0005	**0.0674**	**YES**	0.0059	**0.0379**	**YES**
	Coverage		0.897	0.897		0.9318	0.9318		0.9171	0.9171	
Movielens-latest-small	Precision	10	0	**0.014**	**YES**	0	**0.002**	NO	0	**0.002**	NO
		20	0	**0.008**	**YES**	0	**0.003**	NO	0.001	**0.003**	NO
		30	0.0013	**0.0053**	NO	0.0006	**0.002**	NO	0.002	**0.0027**	NO
		40	0.002	**0.004**	NO	0.0005	**0.002**	NO	0.002	**0.003**	NO
		50	0.0024	**0.0044**	NO	0.0004	**0.002**	NO	0.0028	**0.0032**	NO
	Recall	10	0	**0.0051**	**YES**	0	**0.0013**	NO	0	**0.0002**	NO
		20	0	**0.0053**	**YES**	0	**0.0026**	NO	0.0001	**0.0015**	NO
		30	0.0026	**0.0053**	NO	0.0001	**0.0026**	NO	0.0013	**0.0017**	NO
		40	0.0031	**0.0053**	NO	0.0001	**0.0048**	NO	0.0014	**0.0021**	NO
		50	0.0034	**0.0066**	NO	0.0001	**0.005**	NO	0.0019	**0.0024**	NO
	Coverage		0.6276	0.6276		0.9672	0.9672		0.9324	0.9324	

Table 3.14: Comparison of Prediction Techniques

Dataset	Performance Measure	TOP-N	Item-based Nearest Neighbors Algorithm			
			Pearson	Cosine	Adjusted Cosine	Significant?
Movielens 100K	Precision	10	**0.058**	0.006	0.006	**YES, P1 vs P2, P1 vs P3**
		20	**0.063**	0.031	0.027	**YES, P1 vs P2, P1 vs P3**
		30	**0.0813**	0.052	0.038	**YES, P1 vs P3**
		40	**0.082**	0.0635	0.0495	**YES, P1 vs P3**
		50	**0.082**	0.0704	0.056	**YES, P1 vs P3**
	Recall	10	**0.016**	0.0007	0.0011	**YES, R1 vs R2, R1 vs R3**
		20	**0.0302**	0.0131	0.0123	**YES, R1 vs R2, R1 vs R3**
		30	**0.0641**	0.0433	0.0248	**YES, R1 vs R2, R1 vs R3**
		40	**0.0879**	0.0708	0.0478	**YES, R1 vs R3**
		50	**0.106**	0.0979	0.0688	**YES, R1 vs R2, R1 vs R3**
	Coverage		0.8924	0.992	0.9725	
Eachmovie	Precision	10	**0.082**	0.068	0.072	NO
		20	**0.082**	0.067	0.076	NO
		30	**0.0727**	0.0613	0.0633	NO
		40	**0.068**	0.055	0.0595	NO
		50	**0.0624**	0.0528	0.0568	NO
	Recall	10	**0.037**	0.0303	0.0323	NO
		20	**0.0658**	0.0581	**0.0658**	NO
		30	**0.0892**	0.0831	0.0799	NO
		40	**0.1102**	0.0958	0.1015	NO
		50	**0.1272**	0.1085	0.1172	NO
	Coverage		0.9972	0.9995	0.9847	
Movielens 1M	Precision	10	**0.03**	0.016	0.022	**NO**
		20	**0.049**	0.041	0.024	**YES, P1 vs P3**
		30	**0.0587**	0.0547	0.0273	**YES, P1 vs P2, P1 vs P3**
		40	0.057	**0.0605**	0.0405	**NO**
		50	**0.062**	0.0604	0.0392	**YES, P1 vs P2, P1 vs P3**
	Recall	10	**0.0075**	0.002	0.0037	**NO**
		20	**0.0209**	0.0184	0.0073	**YES, R1 vs R2, R1 vs R3**
		30	0.0351	**0.0358**	0.0139	**YES, R1 vs R2, R1 vs R3**
		40	0.0479	**0.0527**	0.0302	**YES**
		50	0.0632	**0.0674**	0.0379	**YES, R1 vs R2, R1 vs R3**
	Coverage		0.897	0.9318	0.9171	
Movielens-latest-small	Precision	10	**0.014**	0.002	0.002	**YES, P1 vs P2, P1 vs P3**
		20	**0.008**	0.003	0.003	NO
		30	**0.0053**	0.002	0.0027	NO
		40	**0.004**	0.002	0.003	NO
		50	**0.0044**	0.002	0.0032	NO
	Recall	10	**0.0051**	0.0013	0.0002	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		20	**0.0053**	0.0026	0.0015	NO
		30	**0.0053**	0.0026	0.0017	NO
		40	**0.0053**	0.0048	0.0021	NO
		50	**0.0066**	0.005	0.0024	NO
	Coverage		0.6276	0.9672	0.9324	

Table 3.15: Comparison of Similarity Measures

Therefore, Pearson is the best similarity measure compared to all other measures for item-based algorithm for recommendation task.

User-based Algorithm vs Item-based Algorithm

From the Tables 3.13 and 3.15, we concluded that Cosine and Pearson are the best similarity measures for user-based and item-based algorithms respectively. Table 3.16 compares the user-based algorithm with Cosine and item-based algorithm with Pearson.

- For Movielens 100K, item-based algorithm is showing better performance than user-based algorithm and this difference is significant in more than half of the cases.
- For Eachmovie and Movielens 1M, user-based algorithm outperforms item-based algorithm and this difference is significant in most of the cases in Eachmovie and in all cases in Movielens 1M.
- For Movielens-latest-small, the difference between these two algorithms is very small and is not statistically significant in nearly all cases.

Therefore, user-based algorithm is best compared to item-based algorithm.

In summary, for recommendation task in numerical data sets

- Prediction techniques: ADM is showing good performance compared to SWA for both user-based and item-based algorithms (Tables 3.12 and 3.14).
- Similarity measures: We use ADM to produce predictions in both algorithms. Cosine gives better performance compared to Pearson in user-based algorithm (Table 3.13) and Pearson outperforms Cosine and Adjusted Cosine in item-based algorithm (Table 3.15).
- User-based vs Item-based: User-based algorithm with Cosine performs well compared to item-based algorithm with Pearson (Table 3.16).

- **Binary Feedback**

  In order to recommend items to users in binary data sets, we first need to predict the ratings for each unknown item of the user and their corresponding scores. Based on the predicted ratings and their scores we recommend $top - N$ items

Dataset	Performance Measure	TOP-N	Algorithm		
			User-based Cosine	Item-based Pearson	Significant?
Movielens 100K	Precision	10	0.014	**0.058**	**YES**
		20	0.036	**0.063**	**YES**
		30	0.0487	**0.0813**	**YES**
		40	0.0605	**0.082**	**YES**
		50	0.0648	**0.082**	NO
	Recall	10	0.0045	**0.016**	**YES**
		20	0.0167	**0.0302**	NO
		30	0.0433	**0.0641**	NO
		40	0.0704	**0.0879**	NO
		50	0.0929	**0.106**	**YES**
	Coverage		0.9976	0.8924	
Eachmovie	Precision	10	**0.162**	0.082	**YES**
		20	**0.125**	0.082	**YES**
		30	**0.1107**	0.0727	**YES**
		40	**0.095**	0.068	**YES**
		50	**0.0868**	0.0624	**YES**
	Recall	10	**0.0622**	0.037	**YES**
		20	**0.0889**	0.0658	NO
		30	**0.1256**	0.0892	**YES**
		40	**0.1412**	0.1102	NO
		50	**0.159**	0.1272	**YES**
	Coverage		0.9996	0.9972	
Movielens 1M	Precision	10	**0.148**	0.03	**YES**
		20	**0.141**	0.049	**YES**
		30	**0.1393**	0.0587	**YES**
		40	**0.133**	0.057	**YES**
		50	**0.1284**	0.062	**YES**
	Recall	10	**0.0285**	0.0075	**YES**
		20	**0.0562**	0.0209	**YES**
		30	**0.133**	0.0351	**YES**
		40	**0.1132**	0.0479	**YES**
		50	**0.1379**	0.0632	**YES**
	Coverage		0.9346	0.897	
Movielens-latest-small	Precision	10	0.002	**0.014**	**YES**
		20	0.003	**0.008**	NO
		30	0.004	**0.0053**	NO
		40	0.0035	**0.004**	NO
		50	0.0036	**0.0044**	NO
	Recall	10	0.0013	**0.0051**	NO
		20	0.0045	**0.0053**	NO
		30	**0.0071**	0.0053	NO
		40	**0.0073**	0.0053	NO
		50	**0.0077**	0.0066	NO
	Coverage		0.9773	0.6276	

Table 3.16: Comparison of User-based and Item-based Algorithms

(items which are predicted as like with highest predicted scores) to the user. As discussed above, two prediction techniques used in neighborhood-based algorithms are BoM and BoSS.

User-based Algorithm

In Table 3.17 we compare the recommendation accuracy based on BoM and BoSS for user-based algorithm with different similarity measures. Though, Boss is outperforming BoM for all data sets and all similarity measures, this difference is very small and is not significant in most of the cases for Movielens 100K and in some cases for Eachmovie and in all cases for Movielens 1M and Movielens-latest-small. Therefore, BoSS is preferable to BOM.

In Table 3.18, we compare the performance difference between similarity measures (with BoSS as the prediction technique) of user-based algorithm and we observe the following:

- For both Movielens 100K and Eachmovie, Pearson outperforms Cosine and Jaccard and this difference is significant in all cases between Pearson and Cosine and in less than half of the cases and in nearly all cases between Pearson and Jaccard for Movielens 100K and Eachmovie respectively. Jaccard is outperforming Cosine and this difference is significant in less than half of the cases for Movielens 100K and is not sgnificant in nearly all cases for Eachmovie.
- For Movielens 1M, Jaccard is outperforming Pearson in many cases and Cosine in all cases and the difference between Jaccard and Pearson is not significant in nearly all cases, but the difference between Jaccard and Cosine is significant in nearly all cases.
- For Movielens-latest-small, the performance difference between all similarity measures is very small in most of the cases and is not significant in all cases.

Therefore, Pearson is preferable to other similarity measures for user-based algorithm.

Item-based Algorithm

Dataset	Performance Measure	TOP-N	User-based Nearest Neighbors Algorithm								
			Pearson			Cosine			Jaccard		
			BoM	BoSS	Significant?	BoM	BoSS	Significant?	BoM	BoSS	Significant?
Movielens 100K	Precision	10	0.46	**0.464**	NO	0.41	**0.414**	NO	0.456	**0.46**	NO
		20	**0.376**	**0.376**	NO	0.325	**0.327**	NO	0.359	**0.362**	NO
		30	0.32	**0.3207**	NO	**0.276**	0.2747	NO	0.2967	**0.2993**	NO
		40	0.2845	**0.285**	NO	0.242	**0.2425**	NO	0.2565	**0.2605**	NO
		50	0.2544	**0.2556**	NO	0.2192	**0.2208**	NO	0.23	**0.2336**	NO
	Recall	10	0.1894	**0.1933**	NO	0.1574	**0.1621**	NO	0.1821	**0.1869**	NO
		20	0.2889	**0.2896**	NO	0.2339	**0.2386**	NO	0.2692	**0.2762**	NO
		30	0.3446	**0.347**	NO	**0.2927**	0.2914	NO	0.3162	**0.3207**	NO
		40	0.3941	**0.3972**	NO	0.3308	**0.3344**	NO	0.3479	**0.3605**	**YES**
		50	0.4322	**0.4358**	NO	0.3584	**0.3634**	NO	0.3809	**0.3928**	**YES**
	Coverage		0.8998	0.8998		0.9454	0.9454		0.9454	0.9454	
Eachmovie	Precision	10	0.416	**0.418**	NO	0.294	**0.3**	NO	0.352	**0.358**	NO
		20	0.31	**0.311**	NO	0.207	**0.213**	**YES**	0.237	**0.242**	**YES**
		30	0.25	**0.2527**	NO	0.1653	**0.1727**	**YES**	0.1867	**0.1927**	**YES**
		40	0.211	**0.214**	NO	0.139	**0.1445**	**YES**	0.1535	**0.1595**	**YES**
		50	0.1796	**0.184**	**YES**	0.1192	**0.1248**	**YES**	0.1324	**0.1388**	**YES**
	Recall	10	0.2293	**0.2315**	NO	0.15	**0.1545**	NO	0.1878	**0.1932**	NO
		20	0.3197	**0.3221**	NO	0.2081	**0.2177**	**YES**	0.2472	**0.2559**	**YES**
		30	0.3663	**0.3741**	NO	0.2476	**0.2655**	**YES**	0.2864	**0.2969**	**YES**
		40	0.4039	**0.4128**	NO	0.2666	**0.2845**	**YES**	0.3053	**0.3182**	**YES**
		50	0.4182	**0.4316**	**YES**	0.2832	**0.3048**	**YES**	0.3232	**0.3394**	**YES**
	Coverage		0.9534	0.9534		0.9987	0.9987		0.9987	0.9987	
Movielens 1M	Precision	10	0.416	**0.418**	NO	**0.34**	**0.34**	NO	**0.432**	**0.432**	NO
		20	0.335	**0.336**	NO	**0.273**	**0.273**	NO	0.36	**0.361**	NO
		30	0.2993	**0.3**	NO	**0.2447**	**0.2447**	NO	0.3033	**0.304**	NO
		40	0.274	**0.275**	NO	**0.221**	**0.221**	NO	**0.267**	**0.267**	NO
		50	0.2496	**0.2504**	NO	**0.2072**	**0.2072**	NO	**0.2452**	**0.2452**	NO
	Recall	10	0.1105	**0.1119**	NO	**0.0882**	**0.0882**	NO	**0.124**	**0.124**	NO
		20	0.1718	**0.1732**	NO	**0.1327**	**0.1327**	NO	0.1943	**0.1953**	NO
		30	0.2245	**0.2259**	NO	**0.1829**	**0.1829**	NO	0.2396	**0.2406**	NO
		40	0.2726	**0.275**	NO	**0.2154**	**0.2154**	NO	**0.2795**	**0.2795**	NO
		50	0.3102	**0.3126**	NO	**0.2491**	**0.2491**	NO	0.3147	**0.3153**	NO
	Coverage		0.9113	0.9113		0.9128	0.9128		0.9128	0.9128	
Movielens-latest-small	Precision	10	0.346	**0.348**	NO	0.348	**0.354**	NO	0.352	**0.354**	NO
		20	0.277	**0.281**	NO	0.272	**0.276**	NO	0.27	**0.272**	NO
		30	0.23	**0.2313**	NO	0.228	**0.2313**	NO	0.2287	**0.2313**	NO
		40	0.196	**0.1965**	NO	0.1965	**0.199**	NO	0.2035	**0.205**	NO
		50	0.18	**0.1804**	NO	0.176	**0.1788**	NO	0.1804	**0.1812**	NO
	Recall	10	0.1334	**0.1341**	NO	0.1493	**0.1524**	NO	**0.1494**	0.1492	NO
		20	0.1875	**0.1915**	NO	0.2177	**0.2219**	NO	0.2046	**0.2048**	NO
		30	0.2239	**0.2258**	NO	0.2505	**0.256**	NO	0.2469	**0.2491**	NO
		40	0.2441	**0.2455**	NO	0.269	**0.2747**	NO	0.2752	**0.2771**	NO
		50	0.2645	**0.266**	NO	0.288	**0.2958**	NO	0.296	**0.2984**	NO
	Coverage		0.6192	0.6192		0.8141	0.8141		0.8141	0.8141	

Table 3.17: Comparison of Prediction Techniques

Dataset	Performance Measure	TOP-N	User-based Nearest Neighbors Algorithm			
			Pearson	Cosine	Jaccard	Significant?
Movielens 100K	Precision	10	**0.464**	0.414	0.46	**YES, P1 vs P2**
		20	**0.376**	0.327	0.362	**YES, P1 vs P2, P2 vs P3**
		30	**0.3207**	0.2747	0.2993	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	**0.285**	0.2425	0.2605	**YES, P1 vs P2, P1 vs P3**
		50	**0.2556**	0.2208	0.2336	**YES, P1 vs P2, P1 vs P3**
	Recall	10	**0.1933**	0.1621	0.1869	**YES, R1 vs R2, R2 vs R3**
		20	**0.2896**	0.2386	0.2762	**YES, R1 vs R2, R2 vs R3**
		30	**0.347**	0.2914	0.3207	**YES, R1 vs R2**
		40	**0.3972**	0.3344	0.3605	**YES, R1 vs R2**
		50	**0.4358**	0.3634	0.3928	**YES, R1 vs R2, R1 vs R3**
	Coverage		0.8998	0.9454	0.9454	
Eachmovie	Precision	10	**0.418**	0.3	0.358	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		20	**0.311**	0.213	0.242	**YES, P1 vs P2, P1 vs P3**
		30	**0.2527**	0.1727	0.1927	**YES, P1 vs P2, P1 vs P3**
		40	**0.214**	0.1445	0.1595	**YES, P1 vs P2, P1 vs P3**
		50	**0.184**	0.1248	0.1388	**YES, P1 vs P2, P1 vs P3**
	Recall	10	**0.2315**	0.1545	0.1932	**YES, R1 vs R2**
		20	**0.3221**	0.2177	0.2559	**YES, R1 vs R2, R1 vs R3**
		30	**0.3741**	0.2655	0.2969	**YES, R1 vs R2, R1 vs R3**
		40	**0.4128**	0.2845	0.3182	**YES, R1 vs R2, R1 vs R3**
		50	**0.4316**	0.3048	0.3394	**YES, R1 vs R2, R1 vs R3**
	Coverage		0.9534	0.9987	0.9987	
Movielens 1M	Precision	10	0.418	0.34	**0.432**	**YES, P1 vs P2, P2 vs P3**
		20	0.336	0.273	**0.361**	**YES, P1 vs P2, P2 vs P3**
		30	0.3	0.2447	**0.304**	**YES, P1 vs P2, P2 vs P3**
		40	**0.275**	0.221	0.267	**YES, P1 vs P2, P2 vs P3**
		50	**0.2504**	0.2072	0.2452	**YES, P1 vs P2, P2 vs P3**
	Recall	10	0.1119	0.0882	**0.124**	**YES, R1 vs R2, R2 vs R3**
		20	0.1732	0.1327	**0.1953**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.2259	0.1829	**0.2406**	**YES, R1 vs R2, R2 vs R3**
		40	0.275	0.2154	**0.2795**	**YES, R1 vs R2, R2 vs R3**
		50	0.3126	0.2491	**0.3153**	**YES**
	Coverage		0.9113	0.9128	0.9128	
Movielens-latest-small	Precision	10	0.348	**0.354**	**0.354**	NO
		20	**0.281**	0.276	0.272	NO
		30	**0.2313**	**0.2313**	**0.2313**	NO
		40	0.1965	0.199	**0.205**	NO
		50	0.1804	0.1788	**0.1812**	NO
	Recall	10	0.1341	**0.1524**	0.1492	NO
		20	0.1915	**0.2219**	0.2048	NO
		30	0.2258	**0.256**	0.2491	NO
		40	0.2455	0.2747	**0.2771**	NO
		50	0.266	0.2958	**0.2984**	NO
	Coverage		0.6192	0.8141	0.8141	

Table 3.18: Comparison of Similarity Measures

Dataset	Performance Measure	TOP-N	Item-based Nearest Neighbors Algorithm								
			Pearson			Cosine			Jaccard		
			BoM	BoSS	Significant?	BoM	BoSS	Significant?	BoM	BoSS	Significant?
Movielens 100K	Precision	10	0.156	**0.22**	**YES**	0.16	**0.248**	**YES**	0.162	**0.264**	**YES**
		20	0.121	**0.175**	**YES**	0.14	**0.204**	**YES**	0.14	**0.208**	**YES**
		30	0.1102	**0.1587**	**YES**	0.1267	**0.1773**	**YES**	0.128	**0.1847**	**YES**
		40	0.1029	**0.142**	**YES**	0.1175	**0.159**	**YES**	0.1185	**0.166**	**YES**
		50	0.0929	**0.1272**	**YES**	0.1112	**0.1492**	**YES**	0.1108	**0.1513**	**YES**
	Recall	10	0.0571	**0.0802**	**YES**	0.0535	**0.0877**	**YES**	0.0535	**0.0957**	**YES**
		20	0.0831	**0.1195**	**YES**	0.0936	**0.1426**	**YES**	0.0901	**0.1434**	**YES**
		30	0.1095	**0.1569**	**YES**	0.1246	**0.1832**	**YES**	0.12	**0.1883**	**YES**
		40	0.1325	**0.1864**	**YES**	0.1481	**0.211**	**YES**	0.1446	**0.222**	**YES**
		50	0.1488	**0.2076**	**YES**	0.1714	**0.2426**	**YES**	0.1686	**0.2466**	**YES**
	Coverage		0.7879	0.7879		0.9774	0.9774		0.9774	0.9774	
Eachmovie	Precision	10	0.126	**0.188**	**YES**	0.166	**0.2313**	**YES**	0.172	**0.247**	**YES**
		20	0.086	**0.138**	**YES**	0.127	**0.1753**	**YES**	0.13	**0.1903**	**YES**
		30	0.0673	**0.1107**	**YES**	0.098	**0.1377**	**YES**	0.102	**0.1523**	**YES**
		40	0.056	**0.0915**	**YES**	0.0855	**0.1163**	**YES**	0.0885	**0.1258**	**YES**
		50	0.0484	**0.078**	**YES**	0.0721	**0.1021**	**YES**	0.0765	**0.1076**	**YES**
	Recall	10	0.0702	**0.1083**	**YES**	0.0933	**0.1425**	**YES**	0.0987	**0.1541**	**YES**
		20	0.0979	**0.1481**	**YES**	0.1333	**0.1949**	**YES**	0.1405	**0.2174**	**YES**
		30	0.1171	**0.1743**	**YES**	0.1463	**0.2217**	**YES**	0.1587	**0.2454**	**YES**
		40	0.1273	**0.1869**	**YES**	0.1665	**0.2426**	**YES**	0.1774	**0.2598**	**YES**
		50	0.1357	**0.1932**	**YES**	0.1741	**0.2582**	**YES**	0.1887	**0.2706**	**YES**
	Coverage		0.9733	0.9733		0.9982	0.9982		0.9982	0.9982	
Movielens 1M	Precision	10	0.174	**0.25**	**YES**	0.236	**0.27**	**YES**	0.236	**0.278**	**YES**
		20	0.141	**0.201**	**YES**	0.19	**0.218**	**YES**	0.194	**0.23**	**YES**
		30	0.1253	**0.1673**	**YES**	0.1687	**0.1933**	**YES**	0.168	**0.1973**	**YES**
		40	0.113	**0.148**	**YES**	0.15	**0.1695**	**YES**	0.152	**0.1805**	**YES**
		50	0.1056	**0.1356**	**YES**	0.1376	**0.1564**	**YES**	0.1412	**0.1668**	**YES**
	Recall	10	0.049	**0.0713**	**YES**	0.0589	**0.0663**	**YES**	0.061	**0.0748**	**YES**
		20	0.0801	**0.1143**	**YES**	0.0903	**0.102**	**YES**	0.0931	**0.1125**	**YES**
		30	0.0987	**0.136**	**YES**	0.1154	**0.1308**	**YES**	0.1145	**0.1377**	**YES**
		40	0.1136	**0.1554**	**YES**	0.1365	**0.1502**	**YES**	0.1379	**0.1645**	**YES**
		50	0.1315	**0.1727**	**YES**	0.1504	**0.1685**	**YES**	0.1551	**0.1856**	**YES**
	Coverage		0.8576	0.8576		0.9244	0.9244		0.9244	0.9244	
Movielens-latest-small	Precision	10	0.12	**0.182**	**YES**	0.138	**0.182**	**YES**	0.122	**0.156**	**YES**
		20	0.104	**0.139**	**YES**	0.108	**0.144**	**YES**	0.108	**0.131**	**YES**
		30	0.0933	**0.1193**	**YES**	0.0993	**0.13**	**YES**	0.094	**0.122**	**YES**
		40	0.0835	**0.1065**	**YES**	0.088	**0.117**	**YES**	0.085	**0.1105**	**YES**
		50	0.076	**0.096**	**YES**	0.0824	**0.108**	**YES**	0.0796	**0.1016**	**YES**
	Recall	10	0.0515	**0.0791**	**YES**	0.0407	**0.0589**	**YES**	0.0303	**0.0468**	**YES**
		20	0.0834	**0.1154**	**YES**	0.0657	**0.0899**	**YES**	0.0633	**0.0797**	**YES**
		30	0.103	**0.1379**	**YES**	0.086	**0.1188**	**YES**	0.0793	**0.1101**	**YES**
		40	0.1228	**0.1577**	**YES**	0.0988	**0.1382**	**YES**	0.0926	**0.1262**	**YES**
		50	0.1333	**0.171**	**YES**	0.1163	**0.1577**	**YES**	0.1069	**0.1421**	**YES**
	Coverage		0.3208	0.3208		0.9378	0.9378		0.9378	0.9378	

Table 3.19: Comparison of Prediction Techniques

In Table 3.19, we compare the performance of BoM and BoSS for item-based algorithm with different similarity measures. For all data sets, BoSS is clearly outperforming BoM and this difference is significant for all data sets and for all similarity measures.

Table 3.20 compares the performance of different similarity measures of (with BoSS as the prediction technique) item-based algorithm.

- For Movielens 100K, Eachmovie and Movielens 1M, though Jaccard is outperforming Pearson and Cosine, the performance difference between Cosine and Jaccard is not significant. They both outperform Pearson. But the performance difference between Pearson and Cosine is significant in few cases for Movielens 100K and Movielens 1M and in all cases for Eachmovie. Pearson differs from Jaccard significantly in many cases for Movielens 100K, in all cases for Eachmovie and in very few cases for Movielens 1M.
- For Movielens-latest-small, the performance differences between all three algorithms are small and are not significant.

Therefore, Jaccard and Cosine are preferable to Pearson for item-based algorithm.

User-based Algorithm vs Item-based Algorithm

As seen from Tables 3.18 and 3.20 Pearson and Jaccard (also Cosine) are the best similarity measures for user-based and item-based algorithms. Table 3.21 compares the performance of user-based and item-based algorithms with Pearson and Jaccard (we can also consider Cosine) similarity measures respectively. For all data sets, user-based algorithm outperforms item-based algorithm and this difference is significant in all cases for all data sets.

In summary, for recommendation task in binary data sets

- Prediction techniques: BoSS outperforms BoM for both user-based and item-based algorithms (Tables 3.17 and 3.19).

Dataset	Performance Measure	TOP-N	Item-based Nearest Neighbors Algorithm			
			Pearson	Cosine	Jaccard	Significant?
Movielens 100K	Precision	10	0.22	0.248	**0.264**	**YES, P1 vs P3**
		20	0.175	0.204	**0.208**	**YES, P1 vs P3**
		30	0.1587	0.1773	**0.1847**	**YES, P1 vs P3**
		40	0.142	0.159	**0.166**	**YES, P1 vs P3**
		50	0.1272	0.1492	**0.1513**	**YES, P1 vs P2, P1 vs P3**
	Recall	10	0.0802	0.0877	**0.0957**	NO
		20	0.1195	0.1426	**0.1434**	NO
		30	0.1569	0.1832	**0.1883**	**YES, R1 vs R3**
		40	0.1864	0.211	**0.222**	**YES, R1 vs R3**
		50	0.2076	0.2426	**0.2466**	**YES, R1 vs R2, R1 vs R3**
	Coverage		0.7879	0.9774	0.9774	
Eachmovie	Precision	10	0.188	0.2313	**0.247**	**YES, P1 vs P2, P1 vs P3**
		20	0.138	0.1753	**0.1903**	**YES, P1 vs P2, P1 vs P3**
		30	0.1107	0.1377	**0.1523**	**YES, P1 vs P2, P1 vs P3**
		40	0.0915	0.1163	**0.1258**	**YES, P1 vs P2, P1 vs P3**
		50	0.078	0.1021	**0.1076**	**YES, P1 vs P2, P1 vs P3**
	Recall	10	0.1083	0.1425	**0.1541**	**YES, R1 vs R2, R1 vs R3**
		20	0.1481	0.1949	**0.2174**	**YES, R1 vs R2, R1 vs R3**
		30	0.1743	0.2217	**0.2454**	**YES, R1 vs R2, R1 vs R3**
		40	0.1869	0.2426	**0.2598**	**YES, R1 vs R2, R1 vs R3**
		50	0.1932	0.2582	**0.2706**	**YES, R1 vs R2, R1 vs R3**
	Coverage		0.9733	0.9982	0.9982	
Movielens 1M	Precision	10	0.25	0.27	**0.278**	NO
		20	0.201	0.218	**0.23**	NO
		30	0.1673	0.1933	**0.1973**	**YES**
		40	0.148	0.1695	**0.1805**	**YES, P1 vs P2, P1 vs P3**
		50	0.1356	0.1564	**0.1668**	**YES, P1 vs P2, P1 vs P3**
	Recall	10	0.0713	0.0663	**0.0748**	NO
		20	0.1143	0.102	**0.1125**	NO
		30	0.136	0.1308	**0.1377**	NO
		40	0.1554	0.1502	**0.1645**	NO
		50	0.1727	0.1685	**0.1856**	NO
	Coverage		0.8576	0.9244	0.9244	
Movielens-latest-small	Precision	10	**0.182**	**0.182**	0.156	NO
		20	0.139	**0.144**	0.131	NO
		30	0.1193	**0.13**	0.122	NO
		40	0.1065	**0.117**	0.1105	NO
		50	0.096	**0.108**	0.1016	NO
	Recall	10	**0.0791**	0.0589	0.0468	NO
		20	**0.1154**	0.0899	0.0797	NO
		30	**0.1379**	0.1188	0.1101	NO
		40	**0.1577**	0.1382	0.1262	NO
		50	**0.171**	0.1577	0.1421	NO
	Coverage		0.3208	0.9378	0.9378	

Table 3.20: Comparison of Similarity Measures

Dataset	Performance Measure	TOP-N	Algorithm		
			User-based Pearson	Item-based Jaccard	Significant?
Movielens 100K	Precision	10	**0.464**	0.264	**YES**
		20	**0.376**	0.208	**YES**
		30	**0.3207**	0.1847	**YES**
		40	**0.285**	0.166	**YES**
		50	**0.2556**	0.1513	**YES**
	Recall	10	**0.1933**	0.0957	**YES**
		20	**0.2896**	0.1434	**YES**
		30	**0.347**	0.1883	**YES**
		40	**0.3972**	0.222	**YES**
		50	**0.4358**	0.2466	**YES**
	Coverage		0.8998	0.9774	
Eachmovie	Precision	10	**0.418**	0.247	**YES**
		20	**0.311**	0.1903	**YES**
		30	**0.2527**	0.1523	**YES**
		40	**0.214**	0.1258	**YES**
		50	**0.184**	0.1076	**YES**
	Recall	10	**0.2315**	0.1541	**YES**
		20	**0.3221**	0.2174	**YES**
		30	**0.3741**	0.2454	**YES**
		40	**0.4128**	0.2598	**YES**
		50	**0.4316**	0.2706	**YES**
	Coverage		0.9534	0.9982	
Movielens 1M	Precision	10	**0.418**	0.278	**YES**
		20	**0.336**	0.23	**YES**
		30	**0.3**	0.1973	**YES**
		40	**0.275**	0.1805	**YES**
		50	**0.2504**	0.1668	**YES**
	Recall	10	**0.1119**	0.0748	**YES**
		20	**0.1732**	0.1125	**YES**
		30	**0.2259**	0.1377	**YES**
		40	**0.275**	0.1645	**YES**
		50	**0.3126**	0.1856	**YES**
	Coverage		0.9113	0.9244	
Movielens-latest-small	Precision	10	**0.348**	0.156	**YES**
		20	**0.281**	0.131	**YES**
		30	**0.2313**	0.122	**YES**
		40	**0.1965**	0.1105	**YES**
		50	**0.1804**	0.1016	**YES**
	Recall	10	**0.1341**	0.0468	**YES**
		20	**0.1915**	0.0797	**YES**
		30	**0.2258**	0.1101	**YES**
		40	**0.2455**	0.1262	**YES**
		50	**0.266**	0.1421	**YES**
	Coverage		0.6192	0.9378	

Table 3.21: Comparison of User-based and Item-based Algorithms

- Similarity measures: We use BoSS to produce predictions in both algorithms. Pearson gives better performance compared to Cosine and Jaccard in user-based algorithm (Table 3.18) and Cosine and Jaccard outperform Pearson in item-based algorithm (Table 3.20).
- User-based vs Item-based: User-based algorithm with Pearson performs well compared to item-based algorithm with Jaccard (Table 3.21).

- **Unary Feedback**

  User-based Algorithm

  Table 3.22 compares the performance difference of different similarity measures for user-based algorithm in unary data sets.

  - For Movielens 100K and Movielens 1M, the performance difference between Jaccrd and Dice is very small and is not statistically significant. They both outperform Pearson and Cosine. The performance of Pearson and Cosine is very close and is not significant. But both Pearson and Cosine differ from Jaccard and Dice significantly in very few cases for Movielens 100K and in many cases for Movielens 1M.
  - For Eachmovie and Movielens-latest-small, the performances of all similarity measures are very close to each other and are also not significant in nearly all cases for Eachmovie and in all cases for Movielens-latest-small.

  Therefore, Jaccard and Dice are preferable to other similarity measures for user-based algorithm.

  Item-based Algorithm

  Table 3.23 compares the performance of different similarity measures for item-based algorithm in unary data sets.

  - For Movielens 100K, Cosine outperforms all others and significantly differs from other similarity measures in very few cases. But for all other similarity measures, their performance differences are very small and also not significant.

Dataset	Performance Measure	TOP-N	User-based Nearest Neighbors Algorithm				
			Pearson	Cosine	Jaccard	Dice	Significant?
Movielens 100K	Precision	10	0.484	0.5	**0.52**	**0.52**	**YES**
		20	0.415	0.419	**0.434**	**0.434**	NO
		30	0.3687	0.3693	**0.3813**	0.378	NO
		40	0.3285	0.3295	**0.338**	0.335	NO
		50	0.2992	0.3	**0.3064**	0.3048	NO
	Recall	10	0.2123	0.2165	0.2331	**0.2338**	**YES, R1 vs R3, R1 vs R4**
		20	0.3402	0.3336	**0.3551**	0.3547	**YES, R2 vs R3, R2 vs R4**
		30	0.4265	0.4237	**0.4473**	0.4459	**YES, R1 vs R3, R2 vs R3, R2 vs R4**
		40	0.4906	0.4907	**0.5075**	0.5024	**YES**
		50	0.5429	0.5458	**0.5592**	0.5566	NO
	Coverage		0.9764	0.9976	0.9976	0.9976	
Eachmovie	Precision	10	0.51	0.48	**0.53**	0.528	**YES**
		20	0.426	0.414	**0.439**	0.436	**YES, P2 vs P3, P2 vs P4**
		30	0.3507	0.3513	0.36	**0.3607**	NO
		40	0.307	0.3035	0.3095	**0.312**	NO
		50	0.2748	0.272	**0.2752**	0.2748	NO
	Recall	10	0.3134	0.3042	**0.3131**	0.3096	NO
		20	0.4932	0.4815	**0.5039**	0.5026	NO
		30	0.5764	0.5789	0.5898	**0.5902**	NO
		40	0.6503	0.6432	0.6562	**0.6597**	NO
		50	0.7105	0.698	0.7045	**0.705**	NO
	Coverage		0.9991	0.9996	0.9996	0.9996	
Movielens 1M	Precision	10	0.506	0.506	**0.55**	0.542	YES
		20	0.417	0.42	0.468	**0.47**	**YES, P1 vs P3, P1 vs P4, P2 vs P3, P2 vs P4**
		30	0.3793	0.3693	**0.4133**	**0.4133**	**YES, P1 vs P3, P1 vs P4, P2 vs P3, P2 vs P4**
		40	0.3425	0.3305	0.363	**0.365**	**YES, P1 vs P3, P1 vs P4, P2 vs P3, P2 vs P4**
		50	0.3172	0.3092	0.3268	**0.33**	**YES, P1 vs P4, P2 vs P3, P2 vs P4**
	Recall	10	0.1494	0.1432	**0.1596**	0.1589	NO
		20	0.2386	0.2298	0.2585	**0.259**	**YES, R1 vs R4, R2 vs R3, R2 vs R4**
		30	0.3023	0.2898	**0.3359**	0.3342	**YES, R1 vs R3, R1 vs R4, R2 vs R3, R2 vs R4**
		40	0.3531	0.3356	0.378	**0.3781**	**YES, R1 vs R3, R1 vs R4, R2 vs R3, R2 vs R4**
		50	0.4035	0.3859	**0.4192**	0.4188	**YES, R2 vs R3, R2 vs R4**
	Coverage		0.9333	0.9346	0.9346	0.9346	
Movielens-latest-small	Precision	10	0.41	0.414	0.422	**0.424**	NO
		20	0.341	0.337	**0.344**	0.34	NO
		30	0.2947	0.2907	0.2973	**0.2987**	NO
		40	0.261	0.261	0.26	**0.2635**	NO
		50	0.2376	0.234	0.2344	**0.2356**	NO
	Recall	10	0.1674	0.1688	0.1779	**0.1795**	NO
		20	0.2596	0.2603	**0.2754**	0.2701	NO
		30	0.3241	0.3265	0.3315	**0.3351**	NO
		40	0.3722	0.3681	0.3752	**0.3797**	NO
		50	0.4152	0.4	0.408	**0.4103**	NO
	Coverage		0.966	0.9773	0.9773	0.9773	

Table 3.22: Comparison of Similarity Measures

Dataset	Performance Measure	TOP-N	Item-based Nearest Neighbors Algorithm				
			Pearson	Cosine	Jaccard	Dice	Significant?
Movielens 100K	Precision	10	0.422	**0.472**	0.412	0.404	**YES, P1 vs P2, P2 vs P3, P2 vs P4**
		20	0.341	**0.37**	0.35	0.345	NO
		30	0.2887	**0.312**	0.2973	0.3007	NO
		40	0.2605	**0.272**	0.2685	0.268	NO
		50	0.23	0.2436	0.2464	**0.248**	NO
	Recall	10	0.1835	**0.2147**	0.1763	0.1699	**YES, R1 vs R2, R2 vs R3, R2 vs R4**
		20	0.2803	**0.3031**	0.2815	0.2749	**NO**
		30	0.3411	**0.3612**	0.3407	0.342	**NO**
		40	0.4038	**0.4103**	0.396	0.3952	**NO**
		50	0.4446	**0.4483**	0.435	0.434	NO
	Coverage		0.9886	0.992	0.992	0.992	
Eachmovie	Precision	10	0.414	0.506	**0.514**	**0.514**	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
		20	0.334	**0.415**	0.406	0.403	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
		30	0.2787	**0.3407**	0.3293	0.3307	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
		40	0.242	0.28	**0.2855**	0.284	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
		50	0.216	0.2432	**0.2488**	0.248	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
	Recall	10	0.2712	0.2986	0.3003	**0.3007**	NO
		20	0.4289	**0.4745**	0.4576	0.4514	**YES, R1 vs R2**
		30	0.5109	**0.5683**	0.5395	0.5401	**YES, R1 vs R2**
		40	0.5692	**0.6099**	0.6038	0.599	**YES, R1 vs R2, R1 vs R3**
		50	0.6182	0.6438	**0.6505**	0.6423	NO
	Coverage		0.9812	0.9995	0.9995	0.9995	
Movielens 1M	Precision	10	0.362	**0.444**	0.406	0.398	**YES, P1 vs P2**
		20	0.266	**0.351**	0.34	0.331	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
		30	0.2373	**0.3007**	0.2967	0.2967	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
		40	0.2085	**0.272**	0.2705	0.27	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
		50	0.19	**0.252**	0.2472	0.248	**YES, P1 vs P2, P1 vs P3, P1 vs P4**
	Recall	10	0.1081	**0.121**	0.1094	0.1056	NO
		20	0.1543	**0.1865**	0.1786	0.1714	NO
		30	0.204	**0.2285**	0.2207	0.2179	NO
		40	0.2364	**0.269**	0.2633	0.2614	NO
		50	0.2596	**0.3025**	0.2994	0.3014	**YES, R1 vs R2, R1 vs R4**
	Coverage		0.9318	0.9318	0.9318	0.9318	
Movielens-latest-small	Precision	10	0.318	0.336	0.342	**0.346**	NO
		20	0.262	0.277	**0.299**	0.296	**YES, P1 vs P3, P1 vs P4**
		30	0.226	0.24	0.2447	**0.2487**	NO
		40	0.1975	0.2065	0.213	**0.215**	YES
		50	0.1812	0.184	0.19	**0.1912**	NO
	Recall	10	0.1345	0.1436	**0.1486**	0.1466	NO
		20	0.2126	0.2291	**0.2367**	0.2311	NO
		30	0.2681	**0.287**	0.2776	0.2794	NO
		40	0.3041	**0.3154**	0.3149	0.3181	NO
		50	0.3429	0.3383	**0.3431**	0.3412	NO
	Coverage		0.967	0.9672	0.9672	0.9672	

Table 3.23: Comparison of Similarity Measures

- For Eachmovie, Movielens 1M and Movielens-latest-small, the performance differences between Cosine, Jaccard and Dice are very small and are not significant. All of them outperform Pearson and their differences are significant in nearly half of the cases (especially in case of precision) for Eachmovie and Movielens 1M and in very few cases for Movielens-latest-small.

Therefore, Cosine is preferable to other similarity measures for item-based algorithm.

User-based Algorithm vs Item-based Algorithm

From Tables 3.22 and 3.23, we can see that Jaccard and Dice are the similarity measures which performed well for user-based algorithm and Cosine is the best similarity measure for item-based algorithm. Table 3.24 compares the user-based algorithm and item-based algorithm with Jaccard (we can also use Dice) and Cosine similarity measures respectively. For all data sets, user-based algorithm outperforms item-based algorithm and this difference is significant in nearly all cases for Movielens 100K and in nearly half cases for Eachmovie and in all cases for Movielens 1M and Movielens-latest-small data sets.

In summary, for recommendation task in unary data sets

- Similarity measures: Jaccard and Dice give better performance compared to Pearson and Cosine in user-based algorithm (Table 3.22) and Cosine outperforms Pearson, Jaccard and Dice in item-based algorithm (Table 3.23).
- User-based vs Item-based: User-based algorithm with Jaccard performs well compared to item-based algorithm with Cosine (Table 3.24).

- **Numerical vs binary vs Unary Feedback**

  User-based Algorithm

  Table 3.25 compares the recommendation accuracy of user-based algorithm for different types of feedback i.e., Numerical (Cosine(Table 3.13)), binary (Pearson(Table 3.18)) and Unary (Jaccard(Table 3.22)). The performance is better in unary feedback compared to Numerical and binary for all data sets. Compared to Numerical feedback, user-based algorithm performs well for binary feedback data. The performance difference between unary and binary data is significant in

Dataset	Performance Measure	TOP-N	Algorithm		
			User-based Jaccard	Item-based Cosine	Significant?
Movielens 100K	Precision	10	**0.52**	0.472	**YES**
		20	**0.434**	0.37	**YES**
		30	**0.3813**	0.312	**YES**
		40	**0.338**	0.272	**YES**
		50	**0.3064**	0.2436	**YES**
	Recall	10	**0.2331**	0.2147	NO
		20	**0.3551**	0.3031	**YES**
		30	**0.4473**	0.3612	**YES**
		40	**0.5075**	0.4103	**YES**
		50	**0.5592**	0.4483	**YES**
	Coverage		0.9976	0.992	
Eachmovie	Precision	10	**0.53**	0.506	NO
		20	**0.439**	0.415	NO
		30	**0.36**	0.3407	NO
		40	**0.3095**	0.28	**YES**
		50	**0.2752**	0.2432	**YES**
	Recall	10	**0.3131**	0.2986	NO
		20	**0.5039**	0.4745	NO
		30	**0.5898**	0.5683	NO
		40	**0.6562**	0.6099	**YES**
		50	**0.7045**	0.6438	**YES**
	Coverage		0.9996	0.9995	
Movielens 1M	Precision	10	**0.55**	0.444	**YES**
		20	**0.468**	0.351	**YES**
		30	**0.4133**	0.3007	**YES**
		40	**0.363**	0.272	**YES**
		50	**0.3268**	0.252	**YES**
	Recall	10	**0.1596**	0.121	**YES**
		20	**0.2585**	0.1865	**YES**
		30	**0.3359**	0.2285	**YES**
		40	**0.378**	0.269	**YES**
		50	**0.4192**	0.3025	**YES**
	Coverage		0.9346	0.9318	
Movielens-latest-small	Precision	10	**0.422**	0.336	**YES**
		20	**0.344**	0.277	**YES**
		30	**0.2973**	0.24	**YES**
		40	**0.26**	0.2065	**YES**
		50	**0.2344**	0.184	**YES**
	Recall	10	**0.1779**	0.1436	**YES**
		20	**0.2754**	0.2291	**YES**
		30	**0.3315**	0.287	**YES**
		40	**0.3752**	0.3154	**YES**
		50	**0.408**	0.3383	**YES**
	Coverage		0.9773	0.9672	

Table 3.24: Comparison of User-based vs Item-based Algorithms

Dataset	Performance Measure	TOP-N	User-based Nearest Neighbors Algorithm			
			Ordinal Cosine	Binary Pearson	Unary Jaccard	Significant?
Movielens 100K	Precision	10	0.014	0.464	**0.52**	**YES, P1 vs P2, P1 vs P3**
		20	0.036	0.376	**0.434**	**YES, P1 vs P2, P1 vs P3**
		30	0.0487	0.3207	**0.3813**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.0605	0.285	**0.338**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.0648	0.2556	**0.3064**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
	Recall	10	0.0045	0.1933	**0.2331**	**YES, R1 vs R2, R1 vs R3**
		20	0.0167	0.2896	**0.3551**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.0433	0.347	**0.4473**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.0704	0.3972	**0.5075**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.0929	0.4358	**0.5592**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.9976	0.8998	0.9976	
Eachmovie	Precision	10	0.162	0.418	**0.53**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		20	0.125	0.311	**0.439**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		30	0.1107	0.2527	**0.36**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.095	0.214	**0.3095**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.0868	0.184	**0.2752**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
	Recall	10	0.0622	0.2315	**0.3131**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		20	0.0889	0.3221	**0.5039**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.1256	0.3741	**0.5898**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.1412	0.4128	**0.6562**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.159	0.4316	**0.7045**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.9996	0.9534	0.9996	
Movielens 1M	Precision	10	0.148	0.418	**0.55**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		20	0.141	0.336	**0.468**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		30	0.1393	0.3	**0.4133**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.133	0.275	**0.363**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.1284	0.2504	**0.3268**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
	Recall	10	0.0285	0.1119	**0.1596**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		20	0.0562	0.1732	**0.2585**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.0874	0.2259	**0.3359**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.1132	0.275	**0.378**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.1379	0.3126	**0.4192**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.9346	0.9113	0.9346	
Movielens-latest-small	Precision	10	0.002	0.348	**0.422**	**YES, P1 vs P2, P1 vs P3**
		20	0.003	0.281	**0.344**	**YES, P1 vs P2, P1 vs P3**
		30	0.004	0.2313	**0.2973**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.0035	0.1965	**0.26**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.0036	0.1804	**0.2344**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
	Recall	10	0.0013	0.1341	**0.1779**	**YES, R1 vs R2, R1 vs R3**
		20	0.0045	0.1915	**0.2754**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.0071	0.2258	**0.3315**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.0073	0.2455	**0.3752**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.0077	0.266	**0.408**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.9773	0.6192	0.9773	

Table 3.25: Performance of User-based Algorithm for different types of Feedback

many cases for Movielens 100K and Movielens-latest-small data sets and in all cases for Eachmovie and Movielens 1M data sets. The performance differences of numerical feedback with binary and unary is significant in all cases for all data sets.

Item-based Algorithm

In Table 3.26 we compare the the recommendation accuracy of item-based algorithm for different types of feedback i.e., Numerical (Cosine(Table 3.15)), binary (Pearson(Table 3.20)) and Unary (Jaccard(Table 3.23)). The performance of the algorithm for unary data is better compared to binary data which in turn is showing good performance compared to Numerical data. The performance differences of all three are significant in all cases.

In summary, unary feedback is more appropriate for recommendation task compared to numerical and binary data. This is because in numerical data sets, in order to recommend items to the users, we first predict the ratings for all unknown items of the user. It may be possible that many items might share the same predicted rating value. But, in *top-N* recommendation task, we are limited to recommending only the topmost *N* items to the user. As a result, some good items may not be recommended to the user, though their predicted ratings are high which leads to poor recommendation accuracy. Furthermore, in order to distinguish the items with same predicted ratings we need to extract more information from the data set. In case of binary data sets, we recommend to the user only those items with predicted rating $like$. It is always the case that many items may be predicted as $like$ in binary data sets as we have only two possible ratings, $like$ and $dislike$, and all unknown items have to take one of these two ratings. In our experiments while recommending the liked items to users, we distinguished the liked items with the scores based on the neighbors' similarity values and recommended top-N items with highest scores. Though we get better recommendation accuracy than numerical data sets, this accuracy is not comparable with unary data sets. One reason for this may be the following:

We do not exactly know what items are liked and disliked by the users because our data sets originally contain numerical data values. We converted them into binary by taking mid point on the rating scale and considered all ratings greater

Dataset	Performance Measure	TOP-N	Item-based Nearest Neighbors Algorithm			
			Ordinal Pearson	Binary Jaccard	Unary Cosine	Significant?
Movielens 100K	Precision	10	0.058	0.264	**0.472**	**YES, P1 vs P2, P1vs P3, P2 vs P3**
		20	0.063	0.208	**0.37**	**YES, P1 vs P2, P1vs P3, P2 vs P3**
		30	0.0813	0.1847	**0.312**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.082	0.166	**0.272**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.082	0.1513	**0.2436**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
	Recall	10	0.016	0.0957	**0.2147**	**YES, R1 vs R2, R1vs R3, R2 vs R3**
		20	0.0302	0.1434	**0.3031**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.0641	0.1883	**0.3612**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.0879	0.222	**0.4103**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.106	0.2466	**0.4483**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.8924	0.9774	0.992	
Eachmovie	Precision	10	0.082	0.247	**0.506**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		20	0.082	0.1903	**0.415**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		30	0.0727	0.1523	**0.3407**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.068	0.1258	**0.28**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.0624	0.1076	**0.2432**	**YES, P1 vs P3, P2 vs P3**
	Recall	10	0.037	0.1541	**0.2986**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		20	0.0658	0.2174	**0.4745**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.0892	0.2454	**0.5683**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.1102	0.2598	**0.6099**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.1272	0.2706	**0.6438**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.9972	0.9982	0.9995	
Movielens 1M	Precision	10	0.03	0.278	**0.444**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		20	0.049	0.23	**0.351**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		30	0.0587	0.1973	**0.3007**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.057	0.1805	**0.272**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.062	0.1668	**0.252**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
	Recall	10	0.0075	0.0748	**0.121**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		20	0.0209	0.1125	**0.1865**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.0351	0.1377	**0.2285**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.0479	0.1645	**0.269**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.0632	0.1856	**0.3025**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.897	0.9244	0.9318	
Movielens-latest-small	Precision	10	0.014	0.156	**0.336**	**YES, P1 vs P2, P1vs P3, P2 vs P3**
		20	0.008	0.131	**0.277**	**YES, P1 vs P2, P1vs P3, P2 vs P3**
		30	0.0053	0.122	**0.24**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		40	0.004	0.1105	**0.2065**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
		50	0.0044	0.1016	**0.184**	**YES, P1 vs P2, P1 vs P3, P2 vs P3**
	Recall	10	0.0051	0.0468	**0.1436**	**YES, R1 vs R3, R2 vs R3**
		20	0.0053	0.0797	**0.2291**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		30	0.0053	0.1101	**0.287**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		40	0.0053	0.1262	**0.3154**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
		50	0.0066	0.1421	**0.3383**	**YES, R1 vs R2, R1 vs R3, R2 vs R3**
	Coverage		0.6276	0.9378	0.9672	

Table 3.26: Performance of Item-based Algorithm for different types of Feedback

than the mid point as *like* and other as *dislike*. This actually led to imbalanced data sets as the distribution of ratings in the data sets used in our experiments is uneven. Probably this approach might be the reason for not getting the results with binary data sets that are comparable to unary. To deal with imbalanced data sets, we tried another approach to convert numerical ratings to binary in which we compute the average rating for every user and all ratings greater than the average ratings are considered as like and others as dislike. But even with this approach also we could not get the results comparable to that of unary. Because, though our resultant data sets are balanced with this approach, the attached similarity values are not properly distinguishing the items predicted as like (+1).

If we observe the real-world data sets, specifically data sets related to movies and books, most of them are biased towards the high ratings. That means, in many cases, users are interested to rate the item only if they like it. As a result, we do not have sufficient number of low ratings in numerical data sets and sufficient number of dislikes in binary data sets which leads to inaccurate ratings which in turn effect the recommendations made by the algorithm. Whereas, in unary data sets we only have usage information which tells whether the user has consumed this item or not and here we assume that if the user purchased some item means the user has liked that information. In other words, unary data sets contain only one rating $like$. Based on this information, we compute the scores which can then be used to make predictions and recommendations.

3.4 Important Observations

In our work, we evaluated different neighbourhood-based methods with different similarity measures and prediction techniques using appropriate evaluation measures for prediction and recommendation task. We also performed significance testing to determine the performance differences between the algorithms, their similarity measures and prediction techniques. We evaluated these algorithms for different feedback data sets for each task: Numerical, Binary and Unary. Through our experiments, we showed that the performance of the different prediction techniques and similarity measures is not same for different algorithms, different data sets, different kinds of feedback and

different tasks. In other words, the prediction technique or similarity measure which performs best for one algorithm or for one data set may not perform better for other algorithm or other data set. In the same way, the algorithms with different prediction techniques and similarity measures behave differently for prediction and recommendation tasks.

Here, we summarize the performance of various prediction techniques, similarity measures for both user-based algorithm and item-based algorithms and the performance of these algorithms on different data sets with numeric, binary and unary feedback.

1. Prediction task

- Prediction techniques:
  - Numerical feedback:

    (i) User-based Algorithm: ADM performs better than SWA with significant performance difference for all similarity measures and for all data sets.

    (ii) Item-based Algorithm: ADM performs better than SWA with significant performance difference for all similarity measures except for Adjusted Cosine similarity for Movielens-latest-small where the performance difference between both prediction techniques is small with no statistical significance.
  - Binary Feedback:

    (i) User-based Algorithm: Both prediction techniques BoM and BoSS are giving similar performance with no statistical significance for all data sets.

    (ii) Item-based Algorithm: The performance difference between BOM and BOSS is very small and this difference is also not significant for all data sets.

  From this, we can conclude that, ADM is the best prediction technique to work on numerical data for both user-based and item-based algorithms for all data sets. For binary feedback, we can use either BoM or BoSS for both user-based and item-based algorithms for all data sets as they both give similar performance.

- Similarity Measures:

- Numerical Feedback:

  (i) User-based Algorithm: Pearson is performing better than Cosine with significant difference for Movielens 100K and Movielens 1M data sets and with no statistical significance for other two data sets Eachmovie and Movielens-latest-small.

  (ii) Item-based Algorithm: Pearson and Adjusted Cosine perform similarly with no significant statistical difference between them and they both outperform Cosine. This difference is significant for Movielens 100K and Eachmovie, whereas, for Movielens 1M and Movielens-latest-small the performance of all three similarity measures is almost the same and is not significant.

- Binary Feedback:

  (i) User-based Algorithm: Cosine and Jaccard are performing better than Pearson with significant performance differences for Movielens 100K and Movielens 1M. For Eachmovie and Movielens-latest-small, the performance differences between all three similarity measures is small and are not significant.

  (ii) Item-based Algorithm: Cosine and Jaccard outperform Pearson with significant difference for Movielens 100K and Movielens-latest-small. For Eachmovie and Movielens-latest-small, there is a little performance difference between all three similarity measures with no statistical significant difference.

From the above discussion, for numerical data sets, we can say that Pearson is the best similarity measure for user-based algorithm. For item-based algorithm, Pearson and Adjusted Cosine are best similarity measures. For binary feedback, both Cosine and Jaccard are the best similarity measures for both user-based and item-based algorithms

- User-based vs Item-based:

  - Numerical Feedback: Item-based algorithm with Adjusted Cosine outperforms user-based algorithm with Pearson with significant difference for

Eachmovie and with no significant difference for Movielens 1M and Movielens-latest-small. For Movielens 100K, user-based and item-based are showing similar performance with no statistical difference.

– Binary Feedback: Item-based algorithm with Jaccard is outperforming user-based algorithm with Jaccard with significant statistical difference in Movielens-latest-small. For Movielens 100K, Eachmovie and Movielens 1M data sets, they both perform similarly with no statistical difference between them.

Hence, for numerical feedback data, item-based algorithm is best compared to user-based algorithm. Similarly, item-based algorithm performs better than user-based algorithm for binary feedback data sets. Therefore, for prediction task, item-based algorithm is the best suitable algorithm to work on both numerical feedback and binary feedback data sets. The reason is, in item-based algorithm, ratings for unused items for the given target user are predicted based on the ratings given by the same user for the previously consumed items, whereas, in user-based algorithm, ratings are predicted based on ratings given by the neighbouring users. Therefore, the ratings predicted by item-based algorithm are more accurate compared to user-based algorithm.

2. Recommendation task

- Prediction techniques:

  – Numerical feedback:

  (i) User-based Algorithm: ADM outperforms SWA for all similarity measures and for all data sets except for Movielens 100K where SWA significantly gives better performance than ADM in many cases. For Eachmovie and Movielens-latest-small data sets the difference between ADM and SWA is not significant, whereas for Movielens 1M data set, this difference is significant.

  (ii) Item-based Algorithm: ADM outperforms SWA with significant performance difference for all similarity measures and for all data sets except Movielens-latest-small where the difference is not significant in most of the cases.

- Binary Feedback:

  (i) User-based Algorithm: Both prediction techniques BoM and BoSS give similar performance with no statistical significance for all similarity measures and for all data sets except Eachmovie where this difference is significant in most of the cases.

  (ii) Item-based Algorithm: BoSS significantly outperforms BoM for all similarity measures and for all data sets.

From this, we can say that ADM is the best prediction technique to work on numerical data sets for both user-based and item-based algorithms. For binary feedback, BoSS is the better prediction technique compared to BoM.

- Similarity Measures:

  - Numerical Feedback:

    (i) User-based Algorithm: Both Pearson and Cosine are showing similar performance for user-based algorithm with no statistical significance for all data sets except Movielens 1M where Cosine outperforms Pearson with statistical significance.

    (ii) Item-based Algorithm: Pearson outperforms other two measures and their differences are statistically significant for Movielens 100K. For Eachmovie and Movelens-latest-small, performance of all three similarity measures is almost similar and is not significant. For Movielens 1M Pearson and Cosine outperform Adjusted Cosine with no significant difference between Pearson and Cosine. But the performance differences between Pearson and Cosine with Adjusted Cosine are significant in many cases.

  - Binary Feedback:

    (i) User-based Algorithm: For Movielens 100K and Eachmovie, Pearson significantly outperforms Cosine and Jaccard. For Movielens 1M, Jaccard outperforms all others but this difference with Pearson is not significant and is significant with Cosine. For Movielens-latest-small, the performance difference between all similarity measures is very small and is not significant.

(ii) Item-based Algorithm: For item-based algorithm both Cosine and Jaccard outperform Pearson for all data sets except Movielens-latest-small and the difference between Cosine and Jaccard is not significant although they significantly differ from Pearson in some cases. In Movielens-latest-small all measures are performing similarly and their differences are also not statistically significant.

- Unary Feedback:

  (i) User-based Algorithm: For Movielens 100K and Movielens 1M, Jaccard and Dice are showing similar performance with no statistical significance between them. They both significantly outperform Pearson and Cosine. For Eachmovie and Movielens-latest-small, the performances of all similarity measures are showing similar performance and their performance differences are also not significant.

  (ii) Item-based Algorithm: For Movielens 100K, though Cosine outperforms all other similarity measures, it significantly differs from other similarity measures in very few cases. Cosine, Jaccard and Dice are showing similar performance for Eachmovie, Movielens 1M and Movielens-latest-small data sets. They all significantly outperform Pearson for Eachmovie and Movielens 1M data sets. But for Movielens-latest-small data set their differences with Pearson are not significant.

From this discussion, we conclude that Cosine and Pearson are the best similarity measures to work on numerical data sets for user-based and item-based algorithms respectively. For binary data sets, Pearson is the best performing similarity measure for user-based algorithm and both Jaccard and Cosine outperform Pearson in item-based algorithm. For unary data sets, Jaccard and Dice are the best similarity measures for user-based algorithm and Cosine is best for item-based algorithm.

- User-based vs Item-based:

  - Numerical Feedback: For Movielens 100K, item-based algorithm is the best suitable algorithm compared to user-based algorithm. For Eachmovie

and Movielens 1M data sets, user-based algorithm is showing better performance than item-based algorithm. For Movielens-latest-small, both user-based and item-based algorithms are giving similar performance.

- Binary Feedback: User-based algorithm significantly outperforms item-based algorithm for all data sets.
- Unary Feedback: User-based algorithm always outperforms item-based algorithm and this difference is significant for all data sets. Therefore, user-based algorithm is more preferable than item-based algorithm.

From the above discussion, we can say that user-based algorithm is preferable to item-based algorithm for recommendation task for all feedback data sets. The reason is that in item-based algorithm, we recommend the items to the target user which are similar to the items that he has consumed in the past. But, user-based algorithm recommends the items to the target user which are liked by his neighbours. As a result, the items recommended by item-based algorithm might be similar to the items consumed by the user and the items recommended by the user-based algorithm are diverse compared to the item-based algorithm. Users generally get bored of similar recommendations and prefer some diversity in the recommendation list. Therefore, user-based algorithm is preferable to item-based algorithm for recommendation task.

- Numerical vs Binary vs Unary

  The performance of both user-based and item-based algorithms are better for unary feedback compared to Numerical and binary for all data sets and this performance difference is significant. This difference is mainly due to the ambiguity (because several items may share the same predicted rating) in recommending the items for both numerical and binary data sets. Therefore, unary feedback is more suitable for recommendation task compared to numerical and binary feedback.

Chapter 4

Prediction with Confidence in Item-based Collaborative Filtering

In the previous chapter, we presented an empirical analysis on evaluation of user-based and item-based algorithms with different similarity measures, prediction techniques and data sets with different types of feedback using appropriate evaluation measures. In this chapter, we are going to propose an algorithm based on conformal prediction to associate confidence values to the predictions made by the neighborhood-based algorithms. We do not apply CP to user-based collaborative filtering algorithm as it is very difficult to find the required number of neighbours belonging to each class. This problem arises in user-based algorithm for both numerical and binary data sets. This same problem arises when CP is applied on top of item-based collaborative filtering (IBCF) algorithm on numerical data sets. But for binary data sets, it is not a problem to find the required number of neighbours for each class in item-based algorithm in many cases. Therefore, we have used binary feedback data sets while applying CP on top of item-based collaborative filtering algorithm. Major contributions of this chapter are:

- Adaptation of conformal prediction to Item-based collaborative filtering.
- Define NCMs based on similarity measure used in IBCF and empirically determine the best NCM.
- Empirically demonstrate validity and efficiency of our conformal prediction algorithm.

The rest of the chapter is organized as follows: First, we explain the importance of confidence values in recommender systems and how these confidence values will be practically shown to the users and how the users' decisions will be affected by that. We also review different methods proposed in the literature to estimate the confidence of collaborative filtering algorithms. Then, we introduce conformal prediction which is a generalized framework used in machine learning to estimate the confidence values of individual predictions. We also discuss variants of conformal prediction along with the validity and efficiency measures used to determine the quality of predictions made by the conformal prediction algorithms. Next, we discuss item-based collaborative filtering which we have chosen as the underlying algorithm for conformal prediction in order to associate the confidence values produced by the item-based collaborative filtering. Later, we describe how to apply conformal prediction to item-based collaborative filtering. Finally, we demonstrate via experiments the validity and efficiency of the proposed conformal prediction algorithm.

4.1 Importance of Confidence Values in Recommender Systems

The primary goal of a recommender system is to predict the preferences of the users for unseen items and recommend the items with highest predicted ratings or scores. But it is often hard to make the reliable predictions for some users or items due to uncertainty. Several factors such as sparse data, noisy data, change of user preferences over time, algorithm parameters and model selection lead to uncertainty in recommender systems.

As a result, some of the predictions made by the algorithms are inaccurate. Displaying the wrong or inaccurate predictions to the users will decrease the trust of the users in recommendation systems as the users think that the system might be wrong and possibly refrain from using that recommender system in the future. Therefore, measuring the confidence of the predictions and attaching them with the predicted value is a good idea in retaining the customer loyalty which in turn increases the revenue while improving the customer satisfaction.

Next question that should be answered is how these confidence values will be shown to the users?

In rating prediction task, we show confidence of each item along with its predicted rating. For example, i_1, i_7 and i_9 as given below are the items for which we need to predict the ratings for a particular target user. We show confidence values of these items along with their predicted values as follows:

item	**predicted rating**	**confidence**
i_1	9	90%
i_7	5	33%
i_9	2	75%

Similarly, if our task is to recommend top 5 items to users u_2, u_3 and u_6, then we show the confidence values alongside the generated recommendation list of each user.

item	**Top-5 recommendation List**	**confidence**
u_2	$i_2, i_3, i_7, i_{14}, i_{18}$	95%
u_3	$i_6, i_9, i_{12}, i_{15}, i_{18}$	77%
u_6	$i_3, i_6, i_7, i_8, i_{11}$	27%

How the users' decisions will be affected by the associated confidence values?

- In many cases, the user can benefit from observing these confidence values [49]. When the system reports a low confidence for an item, the user may tend to further research the item before making a decision. For example, suppose that a system predicts a rating of '5' for a movie with very high confidence, say 99%, and another movie with the same rating but a lower confidence, say 30%. Then the user may add the first movie immediately to the watching queue, but may further read the plot synopsis for the second movie, and perhaps a few movie reviews before deciding to watch it [70].

- Providing a display of confidence alongside a prediction can also help the user to decide on his/her personalized level of comfort with the prediction's confidence. This also allows for context-based decisions. For example, when the user truly cannot find items of interest, he/she may be willing to consider less confident predictions, whereas when the user has already found some interesting items, he/she may not be willing to consider such predictions [71].

- A confidence measure is important as it can help the users decide which movies to watch, products to buy, and also help an e-commerce site in making a decision on which recommendations should not be displayed, because an erratic recommendation can diminish the trust of users in the system [4, 25].

- Mcnee et al. [49] studied how adding a confidence display can change users' satisfaction and behaviour and concluded that addition of a confidence display to a recommender system increases user satisfaction. It also alters users' behaviour. For user tasks with varying amounts of risk, users were more likely to seek out or avoid low confidence recommendations as appropriate.

4.2 Confidence Estimation Methods in Collaborative Filtering

Very few methods have been proposed in the recommender systems literature to estimate the confidence of collaborative filtering algorithms.

1. McNee et al. [49] proposed a simple confidence metric to be used in recommender systems. They conducted an online study to determine the influence of confidence values on users' decisions and addressed the following questions:

- How confidence values change users satisfaction with a recommender system?

- How confidence values change users behavior in a recommender system?

- Is there any difference in the perception of novice and experienced users towards the confidence values?

- How does providing training on confidence display in a recommender system affect user satisfaction and behaviour?

We have several CF algorithms in recommender systems literature. The intuition behind each algorithm is different. Therefore separate confidence metrics can be proposed for each of these algorithms. For example, in user-based algorithm, the strength of the user's neighbourhood can be used as a base for the confidence metric. Similarly, in item-based algorithm, item similarity values can be used to define a confidence

metric. But, in [49] authors proposed a metric which can be applied to all the CF algorithms instead of defining a separate confidence metric for each of these algorithms.

All collaborative filtering algorithms make their predictions and recommendations based on the rating information. The more number of items the user consumes, the better the system knows about the user's interests. As a result, it can make more accurate recommendations. Similarly, if an item is consumed by many users, the system can make accurate predictions. Therefore, there is a possibility that the recommender system can make inaccurate predictions when the number of available ratings for an item or user is very less. Therefore, authors propose a confidence metric for an item as the number of ratings available for that item. They tested it on Movielens data set and observed that the predictions for movies with many ratings are more accurate than those with few ratings.

As a confidence display, the authors have chosen to use dice for risky movies. For example, movies with fewer than 40 ratings were considered very risky and were marked with two dice and movies with 41 to 80 ratings were marked with one die. Clicking on the dice takes users to a page which briefly explains how predictions were made, why the prediction was risky, and the number of ratings available for that movie.

Finally, the authors concluded the following:

- Associating confidence values to predictions increases user satisfaction.
- Associating confidence values to predictions changes users' behaviour. For user tasks with varying amounts of risk, users were more likely to seek out or avoid low confidence recommendations as appropriate.
- Novice and experienced users perceived the addition of confidence values differently. Novice users are less likely to notice the attached confidence display, but will make use of it if they notice it. Experienced users will notice such a display, but already had opinions of the system, which affected their acceptance and use of the display.
- Training has a considerable impact on user satisfaction in a recommender system. Providing training to new users increased user satisfaction over just adding the confidence display to the system. Providing training to experienced users

increased their usage of the system, but decreased their overall satisfaction with the recommender.

Advantages:

- This is a generalized algorithm and can be applied to any collaborative filtering algorithm.
- The proposed confidence metric is very simple to calculate and can be easily implemented in recommender systems.

Disadvantage:

- The produced confidence values are non-personalized.

2. Mclaughlin and Herlocker [48] proposed a Belief Distribution algorithm for user-based collaborative filtering, which generates belief distributions for each prediction. In their paper, authors observed that neighbourhood-based algorithms failed to produce useful and interesting items while generating *top-N* recommendation list. Two reasons why this flaw has gone unnoticed in neighbourhood-based algorithms by many researchers are the following:

- Most of the researchers used mean absolute error to determine the accuracy of neighbourhood-based algorithms and found that these algorithms are good at predicting the correct ratings for the items. But, good prediction algorithms are always not necessarily good at recommending the useful items.
- Most of the researchers conducted their experiments using offline data sets. As there is no interaction with real users in offline experiments, researchers have no way to detect this kind of problem.

In order to detect this flaw, authors suggested the use of classification accuracy metrics such as precision and recall along with MAE to evaluate these algorithms.

In particular, the authors identified two cases where user-based collaborative algorithms failed to produce good recommendations: active user having too few neighbours who has rated an item and movies have been rated by neighbours with very low correlation to the active user. These are the two main sources of error which causes uncertainty

in predictions and recommendations produced by user-based collaborative algorithm. Therefore, they proposed a belief distribution algorithm based on user-based collaborative algorithm which not only predicts rating values but also the uncertainty involved in each prediction.

In [26], authors observed that, users will give different ratings for an item when asked to rate the same item at different times. This change can be attributed to mood, change in taste, or a variety of other reasons. There is also the possibility that the user entered the wrong rating accidentally. Thus in [48], authors consider the user's rating as noisy evidence of the user's true rating. To capture this fact, they represent the current belief of each user's rating as a belief distribution across all possible discrete rating values. For example, if the user rating is 4 on a 1 to 5 rating scale, we might choose to represent our belief of the true ratings as [.02, .06, .11, .70, 11] with .02 being the our belief that the user's actual rating is 1 and .7 being our belief that the user's actual rating is 4.

Let Y be an m by n rating matrix where m is the number of users and n is the number of items. $r_{u,i}$ is the rating given by user u to item i which is in the range $[1, r_{max}]$, where r_{max} is the maximum rating on a rating scale, for example 5, on a rating scale of 1-5.
The belief distribution algorithm generates the *top-N* recommendations to the target user a as follows:

For every user u, find the similarity with the target user a using Pearson correlation coefficient which is represented with $w_{(a,u)}$ and identify the N most similar users and use their ratings to make predictions for the target user. But, instead of using the neighbours' true ratings, normalized ratings are used to make the predictions in order to improve the accuracy. This normalization is done by subtracting the neighbours' ratings from their respective average ratings which are rounded to the nearest integer. If the rating scale is from 1 to r_{max}, then this normalization results in an integer in the range from $-(r_{max} - 1)$ to $(r_{max} - 1)$. Next, map each rating to a discrete belief distribution, representing our belief of user u's rating on item i. Thus, the difference distribution $d_{(u,i)}$ is a vector of size $2(r_{max} - 1)$ representing the belief that the user rated an item certain distances from their average rating. For each item i that is unrated by the target user a and has been rated by at least one of the user's neighbours, the difference in distribution $d'_{(a,i)}$ is calculated by summing over the difference distributions

of all the neighbours, weighting by the similarity with each neighbour. Then, $d'_{(a,i)}$ is transformed into a vector of size r_{max} as follows:

- The target user's average rating is rounded to the closest integer
- Remap the indices of the predicted belief vector by adding the rounded average rating to each index.
- Since we cannot have ratings less than 1, sum the belief values for all indexes less than or equal to 1, to get the belief for rating 1.
- Since we cannot have ratings greater than r_{max}, sum all belief values for ratings greater than or equal to r_{max} and this becomes belief for r_{max}.

These belief distributions can then be used to make decisions.

Advantage:

- Confidence values are personalized.

Disadvantages:

- This algorithm is not generalized as it is applicable only to user-based algorithm.
- Although their algorithm achieves good classification accuracy in terms of precision by making sure that more popular items are recommended, their algorithm is not as accurate as original user-based CF as far as the prediction accuracy (in terms of MAE) is concerned. Though, they mentioned that with some additional tuning the performance of their algorithm in terms of MAE is close to the user-based CF, they did not demonstrate this empirically.

3. Adomavicius et al. [4] observed that the recommendation accuracy monotonically decreases as the rating variance of the user/item increases. Based on this observation the authors proposed several approaches to improve the accuracy of recommender systems by using rating variance to measure the confidence of recommendations. Recommender systems typically recommend the most relevant *N* items to each user. Therefore, highly-ranked items should be distinguished from others. In [4], authors have done this as follows: On a rating scale of 1-5, all ratings above the threshold 3.5 (i.e., ratings 4 and 5) are considered as highly ranked and ratings less than 3.5

(i.e., ratings 1, 2, and 3) as non-highly-ranked. Thus, the rating values of all recommended items should be greater than the threshold of 3.5. The proposed approaches are discussed below:

- Simple Filtering Approach:

  Typically, *top-N* items for each user are obtained in two steps: selecting the items with predicted ratings greater than the predefined threshold H, say, 3.5, on a rating scale of 1-5 and then choosing *top-N* items with the highest predicted ratings. In simple filtering approach, while selecting the recommended items we also consider some user-specified ratings' standard deviation threshold T in addition to the predefined threshold. That means, for every item for which the rating is to be predicted, a variance score is assigned in addition to the predicted rating. The variance score of an item is the standard deviation of all known ratings for that item. Thereafter, recommend to users only items with variance less than T and predicted rating greater than H.

- Smart and Safe Approaches:

  In smart approach, rating for an item for the given user is predicted by subtracting one standard deviation of known ratings of that item from the predicted rating generated by any traditional recommender system. Using this, one can model the worst case scenario of how low the actual rating could be, if the prediction by the traditional recommender system is not very accurate. Then, this item is considered for recommendation to the user only when the newly computed predicted rating is greater than the predefined threshold. In other words, we are checking whether this adjusted value can still be considered as highly-ranked. If so, we recommend *N* items, according to the order of the original predicted ratings. The safe approach is a slight variation of smart, where *top-N* items are recommended based on the newly adjusted ratings instead of the original predicted ones.

Finally, authors concluded that the prediction accuracy can be significantly improved using simple filtering approach. However, there was also a corresponding decrease in the coverage of recommendations. But, smart and safe approaches generate

valuable recommendations by providing a good balance of prediction accuracy and coverage.

Advantage:

- This is a generalized algorithm and can be applied to any collaborative filtering algorithm.

Disadvantage:

- The produced confidence values are non-personalized.

4. Shani and Gunawardana [70] defines confidence as the system's trust in its predictions or recommendations. Confidence values allow to select an algorithm amongst the several candidate algorithms. For example, given two recommendation algorithms which perform similarly on other properties such as accuracy, it can be desirable to select the one that can provide valid confidence estimates. This is explained in detail with the following example:

There are two recommendation algorithms A and B and both report confidence intervals over possible movie ratings. We train A and B over a confidence threshold, ranging of 95%. For each trained model, we conduct offline experiments by hiding a part of the available ratings to test the predictive ability of both algorithms A and B. Each algorithm produces a confidence interval along with the predicted rating for all missing ratings . Let A_+ and A_- denote the number of times that the predicted rating of algorithm A was within and outside the confidence interval respectively, and similarly, B_+ and B_- denote the number of times that the predicted rating of algorithm B was within and outside the confidence interval respectively. Then, true confidence of the A and B are calculated as $\frac{A_+}{A_+ + A_-}$ and $\frac{B_+}{B_+ + B_-}$ respectively. Assume that confidence values of A and B are 0.97 and 0.94 respectively. From this, we know that A is over conservative, and computes intervals that are too large, while B is liberal and computes intervals that are too small. As we do not require the intervals to be conservative, we prefer B because its estimated intervals are closer to the required 95% confidence.

Disadvantage:

- They did not propose an algorithm to estimate the confidence values to predictions. Instead their algorithm is used to compare two confidence estimation

algorithms and select the best one. No experiments are provided as it is more like a survey paper on evaluation measures of recommender systems.

5. In [35] Koren and Sill observed that user-dependent metrics for confidence like number of ratings given by the user and user's rating variance cannot be used to rank items for the given user and hence cannot distinguish relevant items from irrelevant items which is the primary goal of any recommendation algorithm. Similarly, item-dependent metrics such as number of ratings assigned to an item, item's rating variance are not personalized, and generated confidence values for different items are same for all users. Furthermore, assessing confidence without regards to the inner workings of the prediction algorithm is likely to overlook some valuable information. Certainly, such assessments would not be helpful for combining different recommendation algorithms based on differing confidence values. To deal with these problems, authors proposed a method called *OrdRec* to predict full probability distribution over ratings which in turn helps in estimating the confidence level in the predictions. They formulate the problem of confidence estimation as a binary classification problem and find whether the predicted rating is within one rating level of the true rating. Methods like *OrdRec*, which predict a full distribution of ratings, allow a more principled approach to confidence estimation. For each user-item pair, confidence level is associated with the amount of concentration of the rating probabilities. Metrics like the standard deviation, entropy, or Gini impurity associated with the predicted rating distribution of the actual user-item pair are employed to measure the confidence. The resulting confidence metric is personal, which depends on both user and item, together with a dependency on the inner workings of the algorithm used.

Advantage:

- The algorithm can produce personalized confidence values.

Disadvantage:

- This is not a generalized algorithm and can be applicable only to their proposed CF algorithm.

6. Mazurowski [47] introduced three confidence estimation algorithms (based on IBCF) which are discussed below:

- Resmaple:

  This algorithm is based on statistical concept of resampling where the prediction process of CF algorithm is repeated N times. Every time a subset of the available ratings is selected randomly without replacement. By repeating the CF prediction process N times, we get N predictions for every rating to be predicted. The variability in those predictions will reflect how using minimally different subsets of previous ratings will affect the prediction and therefore it reflects the uncertainty of the predictions. The rationale behind this approach is that, if a small random change in the composition of the set of collected ratings could affect the prediction drastically, it is expected that the confidence of the prediction is low. If the prediction is stable regardless of changing the training data, the confidence is expected to be high. The variability is measured simply by the standard deviation of the predictions. Since a higher standard deviation is expected to correspond to lower confidence, confidence is simply calculated as one over the standard deviation.

- Resample Fast:

  This algorithm is very similar to the RESAMPLE algorithm. A disadvantage of the RESAMPLE algorithm is that the entire CF algorithm has to be repeated N times. This includes calculating similarities between the items as well as biases and it results in high time complexity of the algorithm. The RESAMPLE FAST algorithm accounts for this difficulty and repeats only part of the process. This could be understood as an approximation to the RESAMPLE algorithm which allows for a dramatic reduction of the computational cost while still keeping some if its benefits.

- Inject Noise:

  The Inject Noise algorithm, similar to the Resmaple and Resample Fast algorithms, repeats the CF prediction N times with a slightly modified training set. Instead of resampling the ratings, however, the Inject Noise algorithm changes each rating by adding a number randomly sampled from a Gaussian distribution with zero mean and standard deviation σ.

Advantage:

- This algorithm can produce personalized predictions.
- The proposed confidence estimation algorithms are generalized and can be applied to any collaborative filtering algorithm.

As we have seen, confidence values produced by some of the algorithms are non-personalized[4, 49], and some of them are applicable only to specific algorithms and are not generalized [35, 48]. *Moreover, none of the above algorithms provide guarantees on the error rate of the predictions, where error rate is the probability of excluding the correct class label.* On the other hand, confidence values generated by conformal prediction are personalized. Conformal prediction is a generalized framework which can be applied on top of any algorithm. In addition to that, CP algorithms produce prediction regions with a bound on the probability of error which means that the probability of excluding the correct class label is always guaranteed to be smaller than the specified significance level. When forced to produce point predictions, the confidence of a prediction is 1- the second largest p-value and this second largest p-value becomes the upper bound on the probability of error. Moreover, with conformal prediction we can control the number of erroneous predictions by varying the significance level, thus making it suitable for different kinds of applications.

4.3 Conformal Prediction in Machine Learning

4.3.1 Conformal Prediction and its Variants

In this section, we will introduce the general idea behind conformal prediction(CP) [8, 69, 75]. We are given a training set of examples $Z = \{z_1, z_2, ..., z_l\}$. Each $z_i \in Z$ is a pair $(x_i, y_i); x_i \in \mathbb{R}^d$ is the set of attributes for i^{th} example and y_i is the class label for that example. Our only assumption in conformal prediction is that all $z_i's$ are independently and identically distributed (i.i.d. assumption but generally a weaker assumption of exchangeability is sufficient). Our task is, given a new object x_{l+1} for which the class label is not known we have to predict the class label y_{l+1} for this object. According to our i.i.d. assumption we need to show that how typical the sequence

$$(z_1, z_2, ..., z_{l+1}) = ((x_1, y_1), (x_2, y_2), ..., (x_{l+1}, y_{l+1})) \tag{4.1}$$

is w.r.t. i.i.d. The idea is to look at all possible class labels y_j for the label y_{l+1} and for each label we have to estimate how typical the sequence (4.1) is under the i.i.d assumption. The typicalness of a sequence is obtained by using p-value function. Our prediction for y_{l+1} is the set of y_j for which Equation (4.1) is typical. One way of obtaining p-value function is by considering how strange each example in our sequence is from all other examples. To measure these strangeness values we use nonconformity measure. Nonconformity measure (NCM) $\mathcal{A}$ is a family of functions which assigns a numerical score to each example z_i indicating how different it is from the examples in the multiset $\{z_1, ..., z_{i-1}, z_{i+1}, ..., z_n\}$. We can also compute the nonconformity scores by including the example z_i in the multiset i.e., $\{z_1, ..., z_{i-1}, z_i, z_{i+1}, ..., z_n\}$.

Nonconformity measure has to satisfy the following properties [69]:

1. Nonconformity score of an example is invariant w.r.t. permutations. i.e., for any permutation π of $1, 2..., n$

$$\begin{aligned} \mathcal{A}(z_1, z_2, ..., z_n) = (\alpha_1, \alpha_2, ..., \alpha_n) \implies \\ \mathcal{A}(z_{\pi(1)}, z_{\pi(2)}, ..., z_{\pi(n)}) = (\alpha_{\pi(1)}, \alpha_{\pi(2)}, ..., \alpha_{\pi(n)}) \end{aligned} \tag{4.2}$$

2. $\mathcal{A}$ is chosen such that larger the value of α_i stranger is z_i to other examples.

The nonconformity score α_{l+1} on its own does not really give us any information as it is just a numeric value. However, we can find out how untypical the sequence (4.1) is according to $\mathcal{A}$ when given the label y_j for y_{l+1} by comparing α_{l+1} with all other nonconformity scores. This can be done using p-value function:

$$p(y_j) = \frac{\#\{i = 1, 2, ..., l+1 : \alpha_i \geq \alpha_{l+1}\}}{l+1} \tag{4.3}$$

The output of this function lies between $\dfrac{1}{l+1}$ and 1. An important property of p-value is that $\forall \epsilon \in [0, 1]$, where ϵ is the significance level, and for all probability distributions P on Z,

$$P\{\{z_1, z_2, ..., z_{l+1}\} : p(y_{l+1}) \leq \epsilon\} \leq \epsilon. \tag{4.4}$$

Proof was given in [52]. This tells that if the p-value of a given label is under some very low threshold, say 0.05, then this label is highly unlikely, as such sequences will only be generated at most 5% of the time by any i.i.d. process. Then, we exclude all

labels whose p-values are under some very low threshold ϵ. As a result, a conformal predictor outputs the set

$$\{y_j : p(y_j) > \epsilon\} \tag{4.5}$$

i.e., the set of all labels that have a p-value greater than ϵ with confidence $1 - \epsilon$. This original approach to conformal prediction is called Transductive Conformal Prediction (TCP). The formal procedure was given in [53] and is given below:

- Consider all possible classifications $Y_1, Y_2, ..., Y_c$ for the new object x_{l+1} for which we want to predict the label, and apply the underlying algorithm to every one of the possible completions (where c is the number of possible labels):

  $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l), (x_{l+1}, Y_1)$
  $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l), (x_{l+1}, Y_2)$
  .
  .
  .
  .
  $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l), (x_{l+1}, Y_c)$

- For every possible label y_i, compute a nonconformity score to each example in the set $(x_1, y_1), (x_2, y_2), ..., (x_{l+1}, y_i)$. As a result, we get the following sequences of nonconformity scores for each possible label:

  $\alpha_1^{(Y_2)}, \alpha_2^{Y_2)}, ..., \alpha_l^{(Y_2)}, \alpha_{l+1}^{(Y_1)}$
  $\alpha_1^{(Y_2)}, \alpha_2^{(Y_2)}, ..., \alpha_l^{(Y_2)}, \alpha_{l+1}^{(Y_2)}$
  .
  .
  .
  .
  $\alpha_1^{(Y_c)}, \alpha_2^{(Y_c)}, ..., \alpha_l^{(Y_c)}, \alpha_{l+1}^{(Y_c)}$

- Compute the p-value for x_{l+1} being classified as each possible label Y_i by using the following equation:

$$p(Y_i) = \frac{\#\{i = 1, 2, ..., l+1 : \alpha_i^{(Y_i)} \geq \alpha_{l+1}^{(Y_i)}\}}{l+1} \tag{4.6}$$

- Predict the classification with the largest p-value.
- Output as confidence to this prediction one minus the second largest p-value, and as credibility the p-value of the output prediction, i.e. the largest p-value.

Note that for every example in the test set we need to apply the underlying algorithm c times. As a result, the computational complexity of TCP with respect to the underlying algorithm $\mathcal{U}$ with l training examples and t test examples will be

$$\Theta(|t|c(\mathcal{U}_{train}(l+1) + (l+1)\mathcal{U}_{apply})) \tag{4.7}$$

$\mathcal{U}_{train}(l+1)$ is the time required by $\mathcal{U}$ to generate its general rule. $\mathcal{U}_{apply}$ is the time needed to apply this general rule to a new example. Because of this huge time complexity, TCP cannot be used with an underlying algorithm that requires long training time, or when the data set is large.

On the other hand, Inductive conformal predictors (ICP) splits the training set Z into two smaller sets, proper training set with $s < l$ examples and the calibration set with $q = l - s$ examples. It then applies the underlying algorithm to the proper training set to generate the prediction rule $P_{[\![z_1,z_2,....,z_s]\!]}$ and it calculates the p-value for each possible classification of the new example using only the examples in the calibration set. As a result, it only needs to apply the underlying algorithm once. Formally, the nonconformity score α_{s+i} of each example z_{s+i} in the calibration set is calculated by applying the prediction rule $P_{[\![z_1,z_2,....,z_s]\!]}$ to z_{s+i}. Similarly, the nonconformity score α_{l+g} for each possible classification Y_j of the new test example x_{l+g} is calculated by applying the prediction rule $P_{[\![z_1,z_2,....,z_s]\!]}$ to z_{l+g}. Note that nonconformity scores of the examples in the calibration set need to be computed only once. Then, compute the p-value of each possible label Y_j of x_{l+g} by comparing the nonconformity score of z_{l+g} with the nonconformity scores of all examples in the calibration set as follows:

$$p(y_j) = \frac{\#\{i = s+1, ..., s+q, l+g : \alpha_i \geq \alpha_{l+g}\}}{q+1} \tag{4.8}$$

The computational complexity of ICP will be

$$\Theta(\mathcal{U}_{train}(l-q) + (q+r)\mathcal{U}_{apply}) \tag{4.9}$$

Formally, this procedure is explained with the following steps:

- Split the training set into two smaller sets, the proper training set with s examples and the calibration set with $q = l - s$ examples.
- Use the proper training set $(z_1, z_2, ..., z_s)$ to generate a general rule $P_{[\![z_1,z_2,....,z_s]\!]}$ for classifying new examples; this general rule is created by the underlying algorithm.
- Assign a nonconformity score to each one of the examples in the calibration set. This will result in the sequence $\alpha_{s+1}, \alpha_{s+2}, ..., \alpha_{s+q}$.
- For each object $x_{l+g}, g = 1, 2, ..., |t|$
  - Consider each possible classification Y_i, i = 1,2,....,c and compute the nonconformity score $\alpha_{l+g}^{(Y_i)}$.
  - Compute the p-value for x_{l+g} being classified as each possible label Y_i using nonconformity scores of the calibration examples and $\alpha_{l+g}^{(Y_i)}$ using the following equation:

$$p(Y_i) = \frac{\#\{i = s+1, ..., s+q, l+g : \alpha_i \geq \alpha_{l+g}^{(Y_i)}\}}{q+1} \tag{4.10}$$

- Predict the classification with the largest p-value.
- Output as confidence to this prediction one minus the second largest p-value, and as credibility the p-value of the output prediction, i.e. the largest p-value.

Calibration examples should only take up a small portion of the training set, so that their removal will not dramatically reduce the predictive ability of the underlying algorithm.

The p-values obtained by both TCP and ICP for each possible classification can be used in two different modes:

- Point prediction: For each test example, predict the classification with the highest p-value. The confidence of this prediction is 1 - the second largest p-value and credibility is the p-value of this prediction for that classification. Confidence

gives us a measure of how likely our prediction is compared to all other possible classifications according to the training set. Credibility tells how well the new item with the assumed label conforms to the training set of items.

- Region prediction: Given the $\epsilon > 0$, output the prediction as the set of all classifications whose p-value $> \epsilon$ with $1 - \epsilon$ confidence that the true label will be in this set.

4.3.2 Validity and Efficiency

Two measures are used in order to indicate the quality of prediction regions produced by CP:

- Validity: Validity at specified error probability (ϵ) is satisfied, if at least $(1 - \epsilon)$ of the prediction regions of test example at that ϵ contain true label. Note that the prediction region for a test example at a given ϵ contains all possible ratings whose p-values are greater than ϵ. For example, validity is satisfied at $\epsilon = 0.1$ if at least 90% of the prediction regions of the test examples contain their true labels. To check the validity we generally use *error percentage* which is the percentage of test examples for which the true label is not inside their prediction regions and is given by

$$error\ percentage = \frac{\sum_{i=1}^{|t|} ERROR_i}{|t|} \tag{4.11}$$

where

$ERROR_i$ = $\begin{cases} 1 & \text{if } T_i \text{ is not a member of } \Gamma_i^\epsilon \\ 0 & \text{otherwise} \end{cases}$

Γ_i^ϵ is the prediction region for the i^{th} test example at significance level ϵ

T_i is the true rating of the i^{th} test example

t is the set of test examples

In other words, error percentage refers to the probability of excluding the true label from the prediction region.

Validity in terms of error percentage is given by

$$validity = 1 - error\ percentage \tag{4.12}$$

- Efficiency of CP is the tightness of prediction regions it produces. The narrower (small number of labels) the prediction region or prediction set, the more efficient the conformal predictor is.

  Criteria to determine the efficiency of conformal predictors [74]:

  1. The S (sum) criterion measures efficiency by the average sum of the p-values; small values are preferable for this criterion. It is ϵ-free.

  $$\frac{\sum_{i=1}^{|t|}\sum_y p_i^y}{|t|}$$

  2. The N (number) criterion uses the average size of a prediction set. Small values are preferable. It is a function of the significance level ϵ.

  $$\frac{\sum_{i=1}^{|t|}|\Gamma_i^\epsilon|}{|t|}$$

  3. The U (Unconfidence) criterion uses the average unconfidence over the test sequence, where the unconfidence for a test object is the second largest p-value. small values are preferable. It is ϵ-free.

  $$min_y(max_{y' \neq y} p_i^{y'})$$

  4. The F (Fuzziness) criterion uses the average fuzziness, where the fuzziness for a test object is the sum of all p-values apart from the largest one. Smaller values are preferable. It is ϵ-free.

  $$\sum_y p_i^y - max_y p_i^y$$

  5. The M (Multiple) criterion uses the percentage of objects in the test sequence for which the prediction set Γ_i^ϵ at level ϵ is multiple, i.e., contains more than one label. Smaller values are preferable. It is ϵ-dependent.

  6. The E (Excess) criterion uses the average amount the size of the prediction set exceeds 1. It is ϵ-dependent.

7. The OU (Observed Unconfidence) criterion uses the average observed unconfidence over the test sequence, where the observed unconfidence for a test example is the largest p-value for the false labels. Smaller values are preferable.

$$\frac{\sum_{i=1}^{|t|} max_{y \neq y_i} p_i^y}{|t|}$$

8. The OF (Observed Fuzziness) criterion uses the average sum of the p-values for the false labels. Smaller values are preferable.

$$\frac{\sum_{i=1}^{|t|} \sum_{y \neq y_i} p_i^y}{|t|}$$

9. The OM (Observed Multiple) criterion uses the percentage of observed multiple predictions in the test sequence, where an observed multiple prediction is defined to be a prediction set including a false label. Smaller values are preferable.

10. The OE (Observed Excess) criterion uses the average number of false labels included in the prediction set at level ϵ. Smaller values are preferable.

The regions produced by any CP algorithm are automatically valid. But efficiency in terms of tightness and usefulness of prediction regions depends on the nonconformity measure used by the CP algorithm [53]. We can define many different nonconformity measures for a given underlying algorithm and each of these measures defines a different CP. Therefore, determining an efficient nonconformity measure based on the underlying algorithm is one important step in CP. In this work, we define different nonconformity measures based on the underlying algorithm and empirically demonstrate the best nonconformity measure.

CP is increasingly becoming popular due to the fact that it can be built on top of any conventional point prediction algorithms like k-NN [57, 61], SVM [67, 68], decision trees [28, 29], random forests [11, 18], neural networks [53, 58], ridge regression [51, 56] etc. Furthermore, CP has been successfully applied to life-critical applications such as diagnosis of acute abdominal pain [54], early detection of ovarian cancer [21], the recognition of hypoxia electroencephalograms (EEGs) [79] and

classification of leukemia subtypes [9]. CP has also been applied in various other applications like the prediction of network traffic demand [17] and estimation of effort for software projects [55]. The confidence measures produced by CPs are not only useful in practice, but also their accuracy is comparable to, and sometimes even better than that of their underlying algorithms. We use conformal prediction to associate a confidence measure for each individual prediction made by our chosen underlying algorithm.

4.4 Item-based Collaborative Filtering

In IBCF, prediction and recommendation are based on item to item similarity. The key motivation behind this scheme is that a user will more likely purchase the items that are similar to items he already has purchased. This can be done as follows: Assume that I is the set of all available items. For every target user u_t, first the algorithm looks into the set of items that he has rated (training set C_t) and computes how similar they are to the target item $i_t \in S_t$ (S_t is the set of test items for the target user u_t) using a similarity measure, and then selects k most similar items $\{i_1, i_2, ..., i_k\}$. Once the most similar items are found, the prediction is then computed by taking the weighted average of the target user's ratings on these similar items. So, two main tasks in IBCF are: computing item similarities and rating prediction.

In order to apply CP to IBCF, it is appropriate to convert all ratings to binary i.e., $like$ and $dislike$. The reason for this is twofold: first, uneven distribution of ratings in the data sets: For instance, more than 80% of all ratings in MovieLens 100K are greater than 2 and nearly 70% of all ratings in Eachmovie are greater than 3. As a result, it becomes very difficult to identify k most similar items consumed by the target user which are rated as 1 when the target item rating is assumed as 1. Second, in [26], authors have shown that user's rating as the noisy evidence of user's true rating. Therefore identifying k most similar items which are rated as 1 when the new item rating is assumed as 1 does not make sense.

The simple way to convert all ratings to binary is as follows: take the middle of the rating scale as the threshold (for instance, 3 in the rating scale of 1-6) and assume all ratings greater than the threshold as liked and all other ratings as disliked. But this approach works fine when the distribution of all ratings is even which is not the

case in most of the data sets. Other approach to convert the ratings into binary is to compute the average rating for every user and consider all ratings whose rating is greater than the average as liked and the all ratings below this average as disliked. This is the best approach to deal with all types of users including pessimistic, optimistic and strict users. For example, for pessimistic (optimistic) users who usually give low (high) rating to every item they consume, we assume that they like items rated above their average rating and dislike items rated below the average. Similarly, in the case of strict users who rate every item correctly according to whether they like that item or not (gives high rating when they like and low rating when they do not like) in which case we assume all ratings above the average (approximately equal to the middle of the rating scale) as like and other ratings as dislike [39]. This ensures that our CP algorithm has reasonable number of liked and disliked items in the data set which makes it easier to find k similar items when the target item is assumed as like or dislike. Item similarity computation and prediction are done as follows:

- Item Similarity Computation: In our IBCF, we have chosen cosine based similarity measure to compute similarities among the items. Since, we have only two rating values (+1, -1), our algorithm uses a variant of the binary cosine similarity measure defined in Equation (3.21). The similarity measure defined in Equation (3.21) produces values in between [+1, -1]. But the similarity measures producing the negative values are not suitable to the proposed nonconformity measures defined in section 4.5. Therefore, we modified the similarity measure defined in Equation (3.21) so as to produce the similarity values in between [0, 1] and is defined as follows:

$$similarity(i,j) = \frac{n_{i+1,j+1} + n_{i-1,j-1}}{\sqrt{n_{i+1} + n_{i-1}}\sqrt{n_{j+1} + n_{j-1}}} \tag{4.13}$$

  Equation (4.13) is same as Equation (3.21) except that the last two terms $n_{i+1,j-1}$ (number of users who liked item i and disliked item j) and $n_{i-1,j+1}$ (number of users who disliked item i and liked item j) are removed from the numerator. In other words, we are using only the common items liked and disliked by both users for computing the similarity values. $Similarity(i,j) = 1$ when these two items are rated by exactly same set of users with same preferences. We chose

cosine similarity measure in our work because of its popularity. But we can use any similarity measure which is suitable to binary data and produces the similarity values in between [0, 1]. Our aim in this paper is not to improve the accuracy of the algorithm, but to provide confidence to the predictions generated by the algorithm.

- Predicting the label (like or dislike): Once the similarities are computed, find the k most similar items (k nearest neighbors) of the target item among the consumed items of the target user. Then predict the label for the target item as the most common label among its k nearest neighbors.

Algorithm 1 describes the item-based collaborative filtering algorithm.

Algorithm 1: : Item-based Collaborative Filtering

Input: k,u_t,C_t,S_t,I
Output: $label_i$
for all $i \in I$ **do**
  for all $j \in I$ **do**
    Compute the similarity between i and j using Equation (4.13)
  end for
end for
for all $i \in S_t$ **do**
  Find k most similar items from C_t
  Find the most common label, $label_i$ (= *like* or *dislike*) among these k most similar items
  Predict $label_i$ as the label for the item i
end for

4.5 Prediction with Confidence in IBCF

We propose a conformal prediction algorithm with item-based collaborative filtering as the underlying algorithm which is a simple and widely used algorithm in commercial applications. We propose different nonconformity measures and empirically determine the best nonconformity measure. We empirically prove validity and efficiency of the proposed algorithm. Experimental results demonstrate that the predictive performance

of conformal prediction algorithm is very close to its underlying algorithm with little uncertainty along with the measures of confidence and credibility.

4.5.1 Problem Formulation

In order to apply CP, we need a training set of examples $\{z_1, z_2, ..., z_l\}$ and a test example z_{l+1} for which we want to make the prediction. Here we discuss how this is formulated in the context of IBCF. For every target user u_t, there is a set of items W_t rated by this user and we consider a part of W_t as the training set C_t. For every item $i \in C_t$, user has assigned a label which tells whether the user liked *(+1)* or disliked *(-1)* the item i. Therefore, the possible labels are $\{$*+1, -1*$\}$. We also have a test set of items S_t ($W_t - C_t$) for which we hide the actual labels assigned by the user. Now, our task is to assign a label (which makes the current test item conforms to the training set) to each of the test set items with an associated confidence measure which is valid according to Equation (4.4).

4.5.2 Nonconformity Measures (NCMs)

We propose different NCMs based on IBCF. First we introduce the terminology to define NCMs. Since we are having only two labels, (= $\{$*+1,-1*$\}$), we are assuming that if $y = 1$ then $\bar{y} = -1$ and vice-versa. Assume that $similarity_i^y$ is vector which is a sorted sequence (in descending order) of similarity of an item i with items in C_t with the same label y. In the same manner, $similarity_i^{\bar{y}}$, is a sorted sequence (in descending order) of similarity of an item i with items in C_t with the label $\bar{y}$. The weight for the item i with label y, w_i^y is defined as the sum of similarity values of k most similar items with the label y among the set of rated items C_t of the target user u_t. Similarly $w_i^{\bar{y}}$ is defined as the sum of similarity values of k most similar items with the label $\bar{y}$ among the items in C_t of the target user u_t.

$$w_i^y = \sum_{j=1}^{k} similarity(i,j)^y. \tag{4.14}$$

$$w_i^{\bar{y}} = \sum_{j=1}^{k} similarity(i,j)^{\bar{y}}. \tag{4.15}$$

where k is the number of most similar items. $similarity(i,j)^y$ is jth most similar item in $similarity_i^y$, y is the label of the item i, $similarity(i,j)^{\bar{y}}$ is jth most similar item in $similarity_i^{\bar{y}}$ and $\bar{y}$ is the label other than the label of item i.

In what follows we define NCMs based on IBCF:

$$NCM1 = k - w_i^y. \tag{4.16}$$

$$NCM2 = \frac{1}{w_i^y}. \tag{4.17}$$

$$NCM3 = \frac{w_i^{\bar{y}}}{w_i^y}. \tag{4.18}$$

1. NCM1 & NCM2: The simple NCMs for an item i are as follows: The maximum value of the similarity function defined in Equation (4.13) is 1. As a result, the maximum value of w_i^y becomes k. The higher the value of w_i^y, the more conforming the item i is with respect to the other items. NCM1 will be high when w_i^y is small and smaller value of w_i^y indicates that the item with label y is nonconforming with other items of the same label. As a consequence, the higher the value of NCM1, the stranger the item i is with respect to the other examples with the same label according to the second property of NCM. Similarly, smaller the value of w_i^y the higher the value of NCM2. The higher the value of NCM2, the more nonconforming the item i is with respect to the other examples.

2. NCM3: A more efficient and a variant of NCM proposed in [61] can be defined by taking into consideration $w_i^{\bar{y}}$ along with w_i^y in computing the nonconformity score. According to NCM3, example i with label y is nonconforming when it is very similar to the items with label $\bar{y}$ (high value of $w_i^{\bar{y}}$) and dissimilar to the items with label y (low value of w_i^y).

4.5.3 Proposed Algorithm

Algorithm 2 (IBCFTCP) is a variant of the algorithm proposed in [61] and describes the application of TCP to IBCF in detail. For every item i in S_t of the target user u_t, try all possible labels in Y. For each label y in Y,

- Assume that y is the label for the current test item i.
- Append the test item i with the assumed label y to the set of consumed items C_t and call this set E.
- Compute the nonconformity scores of all items in E using any of the NCMs discussed in the previous subsection.
- Compute the typicalness of the sequence E using p-value function.
- For region predictions, output the prediction as the set of all labels whose p-values are $> \epsilon$ with confidence $1 - \epsilon$. In the case of point predictions, output the label with the highest p-value with confidence $1-$ the second highest p-value and credibility as the highest p-value.

4.5.4 Time Complexity

In this section, we compare the time complexity of the proposed CP algorithm (IBCFTCP) with the underlying algorithm (IBCF). The time complexity of IBCF can be divided into two parts: (i) time required to compute the similarities between all items (similarity computation) and (ii) time required to make predictions for test items (rating prediction). Let m be the number of users and n be the number of items.

- Similarity computation: We need to compute $n(n-1)$ similarities to compute the similarity between all n items. Each similarity computation requires m operations. Therefore, the time complexity for similarity computation is $n(n-1)m$ operations which is $\Theta(n^2m)$.
- Rating prediction: For each test item of the target user u_t, we need to find k similar items from C_t which requires $|C_t|k$ time and then find the most common label among these k neighbors which needs k operations. Therefore, the time complexity of predicting the rating for each test item is $|C_t|k + k$ which is $\Theta(|C_t|k)$. The time it takes for predicting the ratings for all test items in $|S_t|$ is $\Theta(|S_t||C_t|k)$. If the number of target users are n_t, then the time complexity to predict the ratings for test items of all target users is $\Theta(n_t|S_t||C_t|k)$.

Algorithm 2: : Item-based Collaborative Filtering with TCP (IBCFTCP)

Input: k,u_t,C_t,S_t,I,Y,ϵ
Output: Prediction Region, $label_i$, Confidence, Credibility
for all $i \in I$ **do**
  for all $j \in I$ **do**
    Compute the similarity between i and j using Equation (4.13)
  end for
end for
for all $i \in C_t$ **do**
  Compute w_i^y and $w_i^{\bar{y}}$
  Compute nonconformity scores using any of the NCMs discussed above.
end for
for all $i \in S_t$ **do**
  for all $y \in Y$ **do**
    Update the similarity values of item i with other items in C_t
    Compute w_i^y and $w_i^{\bar{y}}$
    Compute nonconformity score of i using any of the NCMs discussed above.
    for all $j \in C_t$ **do**
      if $label_j = y$ **then**
        if $similarity(i,j)^y > similarity(j,k)^y$ **then**
          Recompute the nonconformity score of item j
        end if
      else if $label_j = \bar{y}$ **then**
        if $similarity(i,j)^{\bar{y}} > similarity(j,k)^{\bar{y}}$ **then**
          Recompute the nonconformity score of item j
        end if
      end if
    end for
    Compute the p-value(p(y)) of item i with label y using Equation (4.3)
  end for
  Prediction Region = $\{y|p(y) > \epsilon\}$ with confidence 1-ϵ
  OR
  $label_i$ = y such that *max*$\{$*p(+1),p(-1)*$\}$ = *p(y)*
  Confidence = *1 - second highest p-value*
  Credibility = *highest p-value*
end for

The time complexity for IBCF is $\Theta(n^2m + n_t|S_t||C_t|k)$

For IBCFTCP, the time required or similarity computation is the same as that of IBCF which is $\Theta(n^2m)$. Time complexity for rating prediction is computed as follows:

- Compute the nonconformity scores for all consumed items C_t of the target user u_t: In order to compute the nonconformity score of for each consumed item i in C_t, we need to find the most similar items from C_t other than i which requires $k(|C_t|-1)$ time. We need to do this for both labels *+1* and *-1*. Using these values, computing the nonconformity score of each item takes 1 operation. Therefore the time complexity of this step is $|C_t|(2 * k(|C_t| - 1) + 1)$ which is $\Theta(|C_t|^2k)$.

- Predict the rating for the test item: For each test item i in S_t, we assume that the test item is consumed by the user. Therefore, we need to update the similarity values of the test item i with all the consumed items of the target user which takes $|C_t|m$ time. Then in order to compute the nonconformity score of the test item i, we have to find the k similar items from the set C_t for both positive and negative ratings. This takes $2 * |C_t|k$. Computing the nonconformity score takes 1 operation. Then we need to update the nonconformity scores of all items in C_t, if required, which takes $|C_t|$ time. Finally computing the p-value takes $|C_t|$ time. This step should be repeated for both positive and negative ratings (that is 2 times as we have 2 labels $\{$+1, -1$\}$).

Therefore, the number of operations required to make the prediction for all test items in $|S_t|$ of one target user is $|C_t|(2 * k(|C_t| - 1) + 1) + |S_t| * 2 * (|C_t|m + 2 * |C_t|k + 1 + |C_t| + |C_t|)$, which is $\Theta(|C_t|^2k + S_t * 2 * (|C_t|m + |C_t|k))$. For n_t target users the time complexity is $\Theta(n_t(|C_t|^2k + S_t * 2 * (|C_t|m + |C_t|k)))$.

Finally, the time complexity of IBCFTCP is $\Theta(n^2m+n_t(|C_t|^2k+|S_t|*2*(|C_t|m+|C_t|k)))$. Generally, the time complexity for the TCP is based on the number of test examples *times* number of labels *times* the underlying algorithm time complexity. But, in our case, it is very less compared to original TCP. This is because, we can pre-compute the nonconformity scores of the consumed items for all target users and recompute them if and only if the similarity value of newly appended test item is greater than the similarity value of k^{th} most similar item of the consumed item. Furthermore, in our case, the number of possible ratings are only *2*. Therefore, even with large data sets

like Eachmovie and Movielens 1M, it is practical to implement TCP on item-based collaborative filtering.

4.6 Experiments and Results

4.6.1 Experimental Setup

We tested our algorithm on four data sets: MovieLens 100K, MovieLens 1M, MovieLens-latest-small and EachMovie. We randomly selected 50 users and for each user 60% of the data is considered as training set and remaining 40% is taken as the test set. We have chosen the users in such away that each user has at least 21 items with +1 and 21 items with -1 (after removing the test items) so that we get required number of neighbors (5-20) for each class to compute the NCMs.

4.6.2 Results and Discussion

Here we compare performance of CP with the underlying algorithm (IBCF) in terms of percentage of correct classifications (%CC) for different data sets and for different k values. In our experiments, we considered 4 different k values: 5, 10, 15 and 20. In order to do this comparison, we have to make single point predictions, since the underlying algorithm makes only single point predictions. In single point predictions, we output a label with the highest p-value. In case, if both labels share this p-value then we can take any one of these labels randomly. In our experiments we take both labels: one that is same as the true label (conforming one) and other one which is other than the true label (nonconforming one) and compute the performance of CP algorithm separately for both cases. In this way we can measure the certainty in predictions. In Table 4.1 we compare the performance in terms of %CC of IBCFTCP with that of IBCF with both conforming labels (CL) and nonconforming labels (NL). We got same results for both NCMs1&2 in terms of %CC, validity and efficiency. Therefore, we do not show their results separately. We also calculate the uncertainty (this uncertainty is w.r.t. the ability of the algorithm in producing a single label while making point predictions) in the predictions as the percentage of items having more than one label (in our case both labels) shares the highest p-value. As the percentage of uncertainty is increased the

Data set	Number of Nearest Neighbors	Algorithm								
		IBCF			IBCFTCP					
					NCMs1&2			NCM3		
		%CC		Uncer-tainty	%CC		Uncer-tainty	%CC		Uncer-tainty
		CL	NL		CL	NL		CL	NL	
Movieens 100K	5	0.6917	0.6917	0	0.7164	0.684	0.0324	**0.7053**	**0.6985**	**0.0068**
	10	0.7492	0.6408	0.1085	0.7185	0.6823	0.0362	**0.7046**	**0.6955**	**0.0092**
	15	0.6919	0.6919	0	0.7126	0.6797	0.0329	**0.703**	**0.6915**	**0.0115**
	20	0.7131	0.6593	0.0538	0.7084	0.6783	0.0301	**0.6988**	**0.6889**	**0.0099**
Eachmovie	5	0.7268	0.7268	0	0.7495	0.6913	0.0581	**0.7258**	**0.7162**	**0.0096**
	10	0.774	0.6702	0.1038	0.7457	0.6994	0.0463	**0.7293**	**0.7224**	**0.0068**
	15	0.7202	0.7202	0	0.7451	0.7041	0.041	**0.7317**	**0.7249**	**0.0068**
	20	0.7442	0.6901	0.0541	0.7439	0.6991	0.0448	**0.7296**	**0.723**	**0.0065**
Movielens 1M	5	0.7132	0.7132	0	0.7409	0.6938	0.047	**0.7224**	**0.7147**	**0.0077**
	10	0.7717	0.6613	0.1105	0.7465	0.7025	0.044	**0.7313**	**0.7239**	**0.0073**
	15	0.7234	0.7234	0	0.746	0.7044	0.0416	**0.7317**	**0.7256**	**0.006**
	20	0.7508	0.6912	0.0597	0.7405	0.7044	0.0361	**0.7279**	**0.7234**	**0.0045**
Movielens-latest-small	5	0.6327	0.6327	0	0.6545	0.6133	0.0413	**0.652**	**0.6205**	**0.0314**
	10	0.7114	0.5666	0.1448	0.6593	0.622	0.0372	**0.6627**	**0.6289**	**0.0338**
	15	0.6426	0.6426	0	0.659	0.6293	0.0297	**0.6655**	**0.6334**	**0.0321**
	20	0.6806	0.5884	0.0922	0.6565	0.6295	0.027	**0.6648**	**0.6327**	**0.0321**

Table 4.1: Performance comparison of IBCF with IBCFTCP with CLs and NLs

deviation between performance of CP with conforming label and nonconforming label will also be increased as shown in Table 4.1.

From Table 4.1 when CLs are considered, the CP algorithm (for all NCMs) is outperforming IBCF for odd k values (5 & 15) due to its slightly higher uncertainty values compared to IBCF, whereas IBCF is performing better for even k values (10 & 20) because of its significantly higher uncertainty values. In case of NLs, CP with NCMs1&2 is showing performance improvement for even k values and lower performance for odd k values (5 & 15) compared to IBCF. On the other hand, CP with NCM3 is outperforming IBCF for all k values when NLs are taken into account except for odd values of K for movielens-latest-small data set. From Table 4.1, we can say that NCM3 is the best NCM (shown in bold numbers) as it is showing good performance with less uncertainty compared to NCMs1&2. Although NCMs1&2 are outperforming NCM3 in terms of %CC when CLs are taken into consideration, these are not good NCMs, as this improvement is due to uncertain predictions produced by these NCMs. Though IBCF with CLs is performing well for even values of k compared to CP with NCM3, this performance improvement is also because of high uncertainty involved in predictions made by IBCF. Uncertainty of IBCF for odd k values is 0 (because in this case there

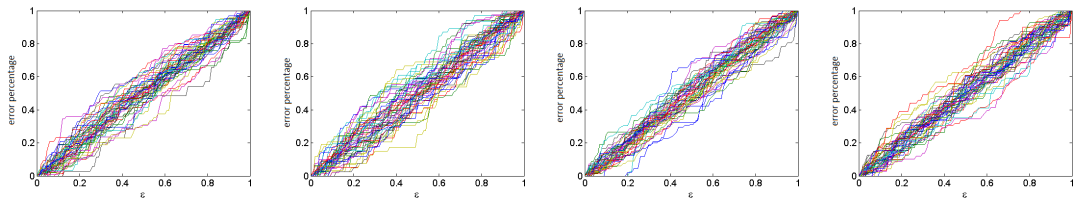


Figure 4.1: Validity of IBCFTCP for different data sets (left to right: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

is no possibility of obtaining equal number of $+1$s and -1s) and for even k values its uncertainty is greater than IBCFTCP with NCMs1&2.

All CPs are automatically valid. Notice that in IBCFTCP the training set and test set of items differ from user to user in contrast to CP in ML where the training set and test sets are fixed for the whole algorithm. So, it is necessary to show that the validity is satisfied for each user. The validity of CP with NCM3 for each user for different data sets and for $k = 20$ is shown in Figure 4.1 (We got similar results for validity and efficiency for other k values). Although validity for each user is also satisfied for NCMs1&2 we did not show the results here because NCM3 is the best NCM in terms of prediction accuracy as shown in Table 4.1 and efficiency (which will be discussed in the following paragraphs). From Fig. 4.1 we can see that the error values for most of the users are within the bounds of ϵ.

Fig. 4.2 shows average validity of IBCFTCP for different data sets and for different NCMs for $k = 20$. From Fig. 4.2, it is clear that prediction regions produced by CP with all NCMs are valid and follows a straight line as required.

Usefulness of CPs depend on its efficiency. We use three criteria to compute the efficiency of CP [58]:

1. % of test elements having prediction regions with single label.
2. % of test elements having prediction regions with more than one label.
3. % of test elements having empty prediction region.

A CP is said to be efficient when the % of second and third criteria are relatively small whereas the % of first should be high especially at higher confidence levels (60-99% (Though these are not the high confidence levels in true sense, these are the acceptable confidence levels as far as the movies domain is concerned)). Our algorithm

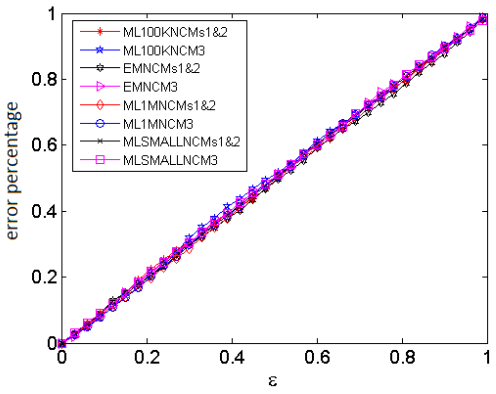


Figure 4.2: Validity

is optimal in the sense that it produces 0% empty prediction regions from 70-99% confidence levels and $< 20\%$ from 60-70% confidence levels, while showing reasonably acceptable performance (in movies domain) in minimizing the second one and maximizing the first one at these confidence levels. The % of test elements having prediction regions producing single labels at different confidence levels for different data sets, different NCMs and for $k = 20$ is shown in Figure 4.3. From Figure 4.3, we can observe that NCM3 is outperforming NCMs1&2, in producing the higher number of prediction regions with single labels. The % of single labels produced by NCMs1&2 never exceed 30% at any confidence level, whereas NCM3 is producing sufficiently large % of single labels especially from 60%-90% confidence levels. Figure 4.4 shows the % of correct predictions among the % of single labels produced by NCM3 at each confidence level. From Figure 4.4 we can see that the % of correct predictions at any confidence level is $> 65\%$.

The % of test elements having prediction regions with more than one rating at different confidence levels is shown in Figure 4.5. In this case also NCM3 is performing better than NCM1&2 as NCM3 is producing less number of prediction regions with multiple labels compared to NCMs1&2 at all confidence levels and in case of NCM3 this number is zero from 0-60% confidence levels. The % of test elements having empty prediction regions at different confidence levels is shown in Figure 4.6. In this case also NCM3 is giving good results compared to NCMs1&2 by producing less number of empty prediction regions compared to NCMs1&2 and this number is zero from 80-99% confidence levels.

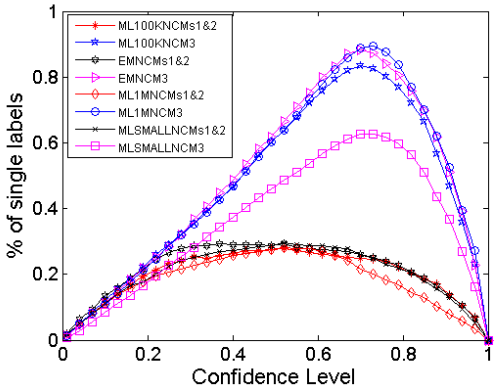


Figure 4.3: % of single labels

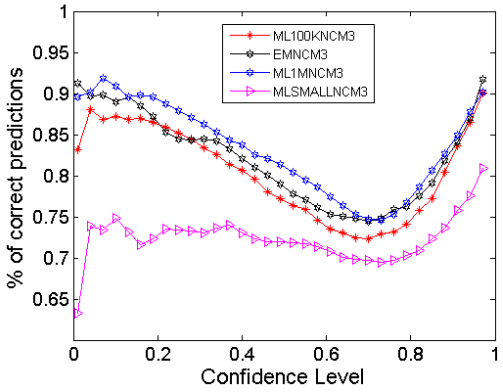


Figure 4.4: % of correct predictions

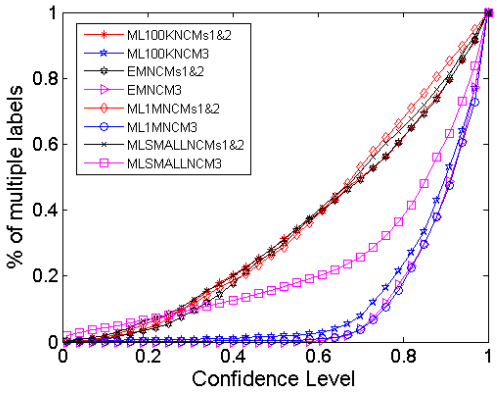


Figure 4.5: % of multiple labels

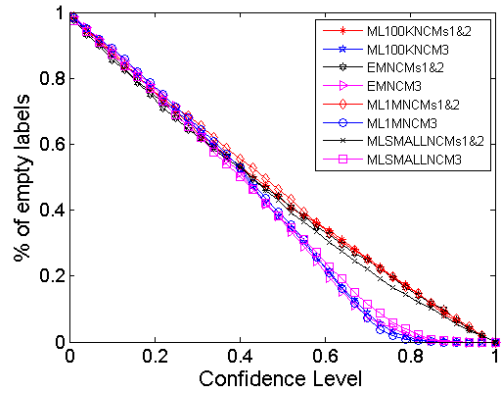


Figure 4.6: % of empty labels

The mean confidence and mean credibility of single point predictions produced by IBCFTCP is shown in Table 4.2. We also calculated mean difference between highest p-value and lowest p-value for all test ratings. We will get confidence predictions when this difference is high. From Table 4.2, we can observe that this difference for NCM3 is around 50% (except for movielens-latest-smallfor which it is around 35%), whereas it is only around 20% with NCMs1&2. Also, the mean confidence and mean credibility values produced by CP with NCM3 (shown in bold numbers) are better than that of NCMs1&2.

From these results, we can observe that the performance of the underlying algorithm (IBCF) and CP algorithm in terms of producing the percentage of correct classifications (%CC) and uncertain predictions when making the point predictions

Number of Nearest Neighbors	Performance Measure	Dataset							
		Movielens 100K		Eachmovie		Movielens 1M		Movielens-latest-small	
		NCMs 1&2	NCM3	NCMs 1&2	NCM3	NCMs 1&2	NCM3	NCMs 1&2	NCM3
5	mean confidence	0.6327	**0.8652**	0.63	**0.8874**	0.6228	**0.8842**	0.6231	**0.7567**
	mean p1 - p2	0.1721	**0.4751**	0.1693	**0.4934**	0.1535	**0.4874**	0.1816	**0.3625**
	mean credibility	0.5397	**0.61**	0.5399	**0.6061**	0.5312	**0.6032**	0.5589	**0.606**
10	mean confidence	0.6407	**0.8684**	0.6455	**0.891**	0.6304	**0.8889**	0.6293	**0.7572**
	mean p1 - p2	0.1796	**0.4725**	0.1884	**0.4929**	0.1624	**0.4877**	0.1867	**0.3574**
	mean credibility	0.5392	**0.6042**	0.5434	**0.602**	0.5324	**0.5989**	0.5578	**0.6005**
15	mean confidence	0.6495	**0.8688**	0.6558	**0.8908**	0.6376	**0.89**	0.6375	**0.7585**
	mean p1 - p2	0.19	**0.4707**	0.2029	**0.4921**	0.1719	**0.4881**	0.1949	**0.3574**
	mean credibility	0.5408	**0.602**	0.5475	**0.6014**	0.5347	**0.5981**	0.5577	**0.5991**
20	mean confidence	0.6579	**0.8688**	0.6653	**0.8895**	0.6447	**0.8903**	0.6456	**0.7598**
	mean p1 - p2	0.2011	**0.4711**	0.2164	**0.4911**	0.1814	**0.4879**	0.2043	**0.359**
	mean credibility	0.5435	**0.6024**	0.5515	**0.6017**	0.537	**0.5976**	0.559	**0.5996**

Table 4.2: Mean confidence and Mean credibility of IBCFTCP

for movielens-latest-small is less compared to all other data sets as shown in Table 4.1. Similarly, CP algorithm produces less efficient prediction regions for movielens-latest-small data set compared to all data sets as shown in Figures 4.3 to 4.5. From these figures, we can see that CP algorithm for movielens-latest-small data set is producing less number of single labels and correct predictions compared to all other data sets while producing large number of multiple labels. In addition to this, mean confidence and credibility values produced by CP algorithm for movielens-latest-small data set is small compared to all other data sets. The reason for this is the sparsity of the movielens-latest-small data set compared to other data sets. We can calculate the sparsity of the data sets using the information provided in the Table 2.1. The sparsity of Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small are 93.7%, 95.66%, 95.81% and 98.43% respectively. One reason for this sparsity is as follows: Users generally rate only a small percentage of items as it is not possible for a user to explore all items especially when there are large number of items. In our Movielens-latest-small data set, the number of items (8915) are very much larger than number of users (718). According to Equation (4.13) similarity value between two items is greater than 0 when the two items are consumed by at least one user with same preference (like or dislike). But due to the sparsity of the movielens-latest-small data set, the similarity values between many pairs of items are zero compared to other data sets which leads to poor performance (as both the underlying algorithm and CP algorithm (specifically

nonconformity measures) depend on similarity values to make the predictions).

In summary, NCM3 is the best NCM compared to NCMs1&2 in terms of prediction accuracy and efficiency. Moreover, when restricted to make single point predictions, mean confidence and credibility values produced by CP with NCM3 are higher than that of NCMs1&2. Both the underlying algorithm and CP show less performance for Movielens-latest-small data set compared to other data sets due to sparsity.

4.7 Summary

In this work, we show the adaptation of conformal prediction to item-based collaborative filtering and proposed different nonconformity measures for conformal prediction based on the underlying algorithm. Using our conformal prediction algorithm we are associating confidence values to each prediction along with the guaranteed error rate unlike the underlying algorithm which produces only bare predictions. Our algorithm is tested on different data sets and we experimentally proved that NCM3 is the best NCM compared to NCMs1&2 in achieving the prediction accuracy as good as the underlying algorithm with little uncertainty and also in producing efficient prediction regions. When making single point predictions, the mean confidence and credibility values produced by the proposed algorithm is reasonably high. Although our algorithm failed in producing large percentage of single labels at 90-99% confidence levels (desirable confidence levels in medical applications) we can use this algorithm to make predictions in certain recommendation domains such as movies, books, news articles, restaurants, music and tourism where the confidence level of 60%-90% is acceptable.

Chapter 5

Prediction with Confidence in Matrix Factorization Algorithm

In the previous chapter, we discussed the application of conformal prediction to Neighborhood-based algorithm specifically, item-based collaborative filtering algorithm. In this chapter, we apply conformal prediction to matrix factorization which is a model-based algorithm. Major contributions of this chapter are:

- We describe application of conformal prediction to matrix factorization on numerical or ordinal data sets to provide confidence values to each of the predictions made by the matrix factorization algorithm.
- We propose four different ways of applying conformal prediction to matrix factorization.
- We define different nonconformity measures suitable to matrix factorization.
- We experimentally prove the best nonconformity measure among the proposed nonconformity measures.
- We empirically demonstrate validity and efficiency of our CP algorithm.
- We experimentally demonstrate the best approach between TCP and ICP for matrix factorization problem.

- We show experimentally why the proposed CP algorithms for numerical data sets failed in producing the efficient prediction regions at high confidence levels and how can we improve the efficiency by converting the data sets to binary.

We first discuss matrix factorization in recommender systems and its variants. Thereafter, we discuss different ways of applying conformal prediction to matrix factorization algorithm. We also mention the important differences between machine learning algorithms and recommender systems, specifically, matrix factorization while applying conformal prediction. Our experimental results also demonstrate the validity and efficiency of the proposed conformal prediction algorithm. Finally, we discuss why the proposed conformal prediction algorithm is not efficient in terms of producing efficient prediction regions with numerical feedback data sets and how converting the data set into binary can improve the efficiency.

5.1 Matrix Factorization

The idea behind the matrix factorization and how can we predict the ratings for the unknown items of the user is described in detail in Chapter 2 (Section 2.4.3). Therefore, in this section, we discuss the variants of basic matrix factorization such as Regularized Matrix Factorization (RMF) [20, 34], Non negative Matrix Factorization (NMF) [38] and Maximum-margin Matrix Factorization (MMMF) [63, 72].

- RMF: In RMF [20], the objective is to find out latent feature matrices W and V such that the sum-squared error between the observed ratings in the original ratings matrix *Y* and the corresponding entries in the predicted matrix $\hat{Y}(\hat{Y} = WV^T)$ is minimized. In order to deal with the problem of overfitting *regularization* is introduced in the objective function. So the optimization problem of RMF is the following:

$$\mathcal{J} = \sum_{(i,j)\in\mathcal{O}} (y_{ij} - w_i v_j^T)^2 + \frac{\lambda}{2}(||W||_F^2 + ||V||_F^2) \tag{5.1}$$

  where $\mathcal{O}$ is the set of observed ratings, λ is a regularization parameter. The regularization term $\frac{\lambda}{2}(||W||_F^2 + ||V||_F^2)$ controls the magnitudes of W and V which makes the model generalize well to test data.

- NMF: In RMF and MMMF (which will be discussed in the next subsection) we do not have any constraints on the values contained in W and V. These matrices can contain either positive or negative values. But in NMF [38], we restrict the values in W and V such that all the values should be positive. Therefore, the objective function becomes

$$\mathcal{J} = \sum_{(i,j)\in\mathcal{O}} (y_{ij} - w_i v_j^T)^2 + \frac{\lambda}{2}(||W||_F^2 + ||V||_F^2) \text{ such that } W, V \geq 0 \qquad (5.2)$$

- MMMF: In both RMF and NMF, our aim is to minimize the sum-squared error between observed ratings of original matrix Y and their corresponding entries in the predicted matrix $\hat{Y}$. When predicting discrete values such as ratings in recommender systems, using a loss function other than the sum-squared error is more appropriate. In MMMF [63, 72] sum-squared error function is replaced with *hinge loss* function. MMMF constraints the norms of W and V (trace norm) instead of their dimensionality. Unlike RMF and NMF, the predicted matrix $\hat{Y}$ in MMMF contains only discrete values in the set R (the set of possible ratings) where $R = \{1, 2, ...r\}$. In order to output only the discrete values in MMMF we have to learn $r - 1$ thresholds $\{\theta_{i1}, \theta_{i2}...\theta_{i(r-1)}\}$ for every user i in addition to the latent feature matrices W and V. If we wish to predict how user i will rate an item j, first we need to compute $w_i v_j^T$ and compare this value against $r - 1$ threshold values of user i which will give us a discrete rating value in $\{1, 2, ...r\}$. In MMMF, we need to minimize the following objective function:

$$\mathcal{J}(W, V, \theta) = \sum_{(i,j)\in\mathcal{O}} \sum_{a=1}^{r-1} h(\mathcal{T}_{ij}^a(\theta_{ia} - w_i v_j^T)) + \frac{\lambda}{2}(||W||_F^2 + ||V||_F^2) \qquad (5.3)$$

where

$$\mathcal{T}_{ij}^a = \begin{cases} +1 & \text{if } a \geq y_{ij} \\ -1 & \text{if } a < y_{ij} \end{cases}$$

and $h(.)$ is a smoothed hinge loss function:

$$h(z) = \begin{cases} 0.5 - z & \text{if } z \leq 0 \\ 0.5(1-z)^2 & \text{if } 0 < z < 1 \\ 0 & \text{if } z > 1 \end{cases} \tag{5.4}$$

In our paper, we use MMMF as the underlying algorithm for CP because of its ability to produce accurate predictions. Furthermore, unlike RMF and NMF where the predicted ratings are floating point values (We need to round these values while coomputing the nonconformity scores), MMMF produces discrete rating values as output which makes it suitable to conformal prediction algorithm.

5.2 Conformal Matrix Factorization

In this section, we propose Conformal Matrix Factorization (CMF) by examining the applicability of conformal prediction to matrix factorization framework. Application of conformal prediction to matrix factorization is thoughtful and non-trivial due to the fact that matrix factorization problems are different from the general conformal prediction problems. The challenge is to analyze and design a matrix factorization problem suitable to the conformal prediction framework. We formulate the problem in the next subsection. Then we propose different nonconformity measures for matrix factorization. Finally, we describe the proposed algorithms for Conformal Matrix Factorization.

5.2.1 Algorithm Setup

As discussed in Chapter 2, Users and Items are the two basic entity sets in recommender systems and any prediction or recommendation is done based on the information related to these two entities. Classic recommendation approaches, as mentioned in the Chapter 1 are based on user-to-user (user-based) and item-to-item (item-based) similarities. In user-based approach we need to find a set of similar users for the target user to make predictions whereas in item-based approach we have to find a set of items similar to the given item for the target user. The only one assumption in conformal prediction is that all examples are independently and identically distributed. When conformal prediction is applied to machine learning problems, each of these examples

constitute a set of features and a label. In the case of recommender systems most of the data sets contain only rating values. Therefore, before applying conformal prediction to any recommendation algorithm, specifically for matrix factorization, we need to formulate our data suitable to CP. In other words, we need to decide what constitutes an example, features and label in matrix factorization. In this work we treat each user and item as independent and identically distributed (i.i.d.). Therefore, there is an equal probability of considering user/item as an example in our proposed conformal prediction algorithms. When conformal prediction is applied with users as examples, we are finding the conformity of the test user with training set of users. Similarly, when items are treated as examples, we find the conformity of the test item with training set of items. For each possibility, the number of training examples, the calculated nonconformity scores and the resultant p-values are different and as a consequence we get different results for user-based and item-based approaches in terms of validity and efficiency along with confidence and credibility values.

Given a partially filled user-item rating matrix $Y \in \mathbb{R}^{m \times n}$, where m is the number of users and n is the number of items, the task is to predict the missing ratings in the matrix with a confidence link to it using matrix factorization. Conformal prediction when applied to machine learning problems, each example in the training set constitutes a feature vector and a label. In the case of recommender systems, most of the data sets encompass only rating information. Moreover, matrix factorization problem assume that features are not given. Hence, we need to formulate our data suitable to conformal prediction before applying conformal prediction to matrix factorization. As mentioned earlier, in a matrix factorization setting, we are given a matrix Y in which rows represent users, columns represent items and entry Y_{ij} denote the rating given by user i for an item j. Each row in a matrix corresponds to one user and each column corresponds to one item. We propose two alternatives to represent an example.

- Each row (user) in the matrix as one example. We refer this as *user-based representation* in the subsequent discussion.
- Each column (item) in the matrix as one example. We refer this as *item-based representation* in the following discussion.

In both cases, target entry (where we need to make a prediction) is considered as a label and the remaining observed entries of a row/column are taken as a feature vector.

For example, let there be 10 users and 15 items and let $Y_{6,7}$ be the entry to be predicted. As specified above, in both cases, an entry $Y_{6,7}$ becomes a label for the test example. In the case of user-based representation rest of the observed entries in row 6 forms a feature vector (ratings given by user 6) i.e. $\{Y_{6,1}, Y_{6,2}, \ldots, Y_{6,6}, Y_{6,8}, \ldots, Y_{6,15}\}$. Similarly, in item-based representation the feature vector would be (ratings given to item 7) $\{Y_{1,7}, Y_{2,7}, \ldots, Y_{5,7}, Y_{7,7}, \ldots, Y_{10,7}\}$. Let us generalize this setting. Let Y_{ij} be the entry to be predicted. *User-based representation* forms a set of m examples $Z = \{z_1, z_2, \ldots, z_m\}$, where each z_k is a pair (x_k, y_k). It defines y_k as Y_{kj} and x_k as $\{Y_{k1}, Y_{k2}, \ldots, Y_{kn}\} \setminus \{Y_{kj}\}$. In this case, training set $Z^t = Z \setminus \{z_i\}$ and z_i is the test example. *Item-based representation* forms a set of n examples $Z = \{z_1, z_2, \ldots, z_n\}$, where each z_k is a pair (x_k, y_k). It defines y_k as Y_{ik} and x_k as $\{Y_{1k}, Y_{2k}, \ldots, Y_{mk}\} \setminus \{Y_{ik}\}$. It is to be recorded that the set $Z \setminus \{z_j\} = Z^t$ is the training set and z_j is the test example.

In this work, we propose to investigate two variants of conformal framework, namely *Transductive Conformal Prediction (TCP)* and *Inductive Conformal Prediction (ICP)* , in the context of matrix factorization using the above two representations. The proposal of investigating the two variants (TCP/ICP) in the context of two different representations (User-based/Item-based) gives rise to four novel conformal matrix factorization methods as highlighted below.

1. User-based Transductive Conformal Matrix Factorization (UBTCMF).
2. Item-based Transductive Conformal Matrix Factorization (IBTCMF).
3. User-based Inductive Conformal Matrix Factorization (UBICMF).
4. Item-based Inductive Conformal Matrix Factorization (IBICMF).

The difference between the User-based and the Item-based variants is only in the way they represent the examples. The basic algorithmic steps are common for both.

5.2.2 Machine Learning Vs Matrix Factorization

- In machine learning algorithms there is a distinction between features and labels. But in the matrix factorization problem formulated above, there is no such distinction. Features and labels both represent ratings. Therefore, rating which

is to be predicted can be treated as label for the given user or item and remaining observed ratings for that user or item act as features.

- In traditional machine learning algorithms, one cannot use the features of a test example to predict the label for another test example. This is because, unless the label is available, we cannot use that example in training. On the other hand, in matrix factorization even if some of the y_a's (labels of the training examples) are not available we can use them to train the model.

Defining (non)conformity measure is a crucial part of the conformal prediction framework. It has to be defined by carefully examining the inner workings of the underlying algorithm. In the next subsection, we propose different nonconformity measures for the matrix factorization problem.

5.2.3 Nonconformity Measures

As mentioned in the previous chapter, nonconformity measure is a measurable function $\mathcal{A}$ that assigns to every set $\{z_1, \ldots, z_n\}$ of n examples, a set $\{\alpha_1, \ldots, \alpha_n\}$ of n real numbers that is invariant with respect to permutations [69]. We define different nonconformity measures and show that they are invariant to permutation of examples.

Nonconformity Measure 1 (NCM1): The most apparent nonconformity measure is to quantify the mismatch between the original and predicted labels. Since in our case every observed entry of an example is a meaningful rating given by the user for an item, we compute the sum of all mismatches (between the original and predicted rating) of all the observed entries of an example z_k as the nonconforming measure of z_k.

$$NCM1(z_k) = \sum_{Y_a \in \Omega(z_k)} I(Y_a) \tag{5.5}$$

where $I(Y_a) = 0$ if $Y_a = \hat{Y}_a$, 1 otherwise. $\Omega(z_k)$ indicates the set of observed ratings in the example z_k and $\hat{Y}_a$ is the predicted rating. This first nonconformity measure gives the number of the observed ratings in the example that are incorrectly predicted. A problem that could arise with *NCM1* is as follows: Suppose there are two examples wherein the number of observed ratings are 15 and 20 respectively. Assume that for

the first example, *NCM1* is 10 and for the second example, it is 11. In this case, the second example is more nonconforming compared to the first one according to the second property of nonconformity measure (i.e., larger the value of α_i stranger is z_i to other examples). In fact, the first one is more nonconforming because the percentage of incorrect classifications in the first example is 66.67 and for the second one it is 55. This same problem arises when both nonconformity scores are equal, for instance, 10. This is because for the first example the number of correctly classified examples remains 66.67 whereas for the second one it is 50.

Nonconformity Measure 2 (NCM2): In order to deal with the above problem, we normalize the above nonconformity scores with the number of observed ratings.

$$NCM2(z_k) = \frac{\sum_{Y_a \in \Omega(z_k)} I(Y_a)}{|\Omega(z_k)|} \tag{5.6}$$

The Zero-one error will be a good choice if there is no relation among the possible labels because any misclassification is a misclassification. But in our case prediction of 4 is better than predicting 1 if the true rating is 5. Therefore, considering the absolute error between true rating and the predicted one instead of zero-one error is a good option here.

Nonconformity Measure 3 (NCM3): It is the sum of absolute differences between true rating and the predicted rating for all the observed ratings in the example.

$$NCM3(z_k) = \sum_{Y_a \in \Omega(z_k)} |Y_a - \hat{Y}_a| \tag{5.7}$$

But *NCM3* suffers from the same problem as *NCM1*. Assume that, here also we have two examples with the number of observed ratings as 15 and 20 respectively. Also suppose that all the predicted ratings corresponding to the observed ratings are deviating by 2 from their true ratings. Then their nonconformity scores are 30 and 40 respectively. But in fact, their predicted ratings are deviating from their true ones in the same way and therefore their nonconformity scores should be the same.

Nonconformity Measure 4 (NCM4): To deal with the above problem we normalize NCM3 with the number of observed ratings. In other words, it is just a mean absolute

error of the example.

$$NCM4(z_k) = \frac{\sum_{Y_a \in \Omega(z_k)} |Y_a - \hat{Y}_a|}{|\Omega(z_k)|} \tag{5.8}$$

However, this may not be a good nonconformity measure because it will give equal weight to all observed ratings (features) and labels. In the case of *Transductive Conformal Matrix Factorization (TCMF)*, we repeat the matrix factorization process for every possible label and in our case we also have very close possible ratings ($1, 2, \ldots, 5$ or $1, 2, \ldots, 6$). Due to the close vicinity of ratings, the resultant predicted matrices do not differ significantly from each other. As a result, the nonconformity scores of all examples for the possible ratings are also more or less the same which finally leads to very close p values. In the case of Inductive Conformal Matrix Factorization (ICMF), the nonconformity scores of the new example for all possible ratings will differ by $1/|\Omega(z_k)|$ (nonconformity scores of calibration examples will not be changed) which again leads to very close p-values. Ideally, the p-value of the correct label should be 1 and for all other labels it should be very low (in practice it is enough that if we get higher p-values for the correct label and sufficiently lower p values for all other labels). Moreover, in the case of TCMF, if we change a single rating in a matrix and if it is conforming with the other observed entries, there is a possibility that the corresponding entry in the predicted matrix will be changed. This is because we are trying to approximate the predicted matrix with the original matrix with respect to the observed ratings. As a result, if we take the difference between the true rating and the predicted rating as the nonconformity score, we can get better results compared to the previous nonconformity measures. This nonconformity measure is also a good choice for ICMF because now the nonconformity scores of new example for all the possible ratings will differ by 1 instead of $1/|\Omega(z_k)|$. This is the simple nonconformity score used in regression [75] and we use this as our next nonconformity measure.

Nonconformity Measure 5 (NCM5): Nonconformity score here is the difference between the true rating and the corresponding predicted rating of the label entry. Suppose if Y_{ij} be the entry to be predicted then Nonconformity of an example z_k is defined as follows in the case of user-based representation.

$$NCM5(z_k) = |Y_{kj} - \hat{Y_{kj}}|, 1 \leq k \leq m. \tag{5.9}$$

Similarly, in the case of item-based representation it is defined as

$$NCM5(z_k) = |Y_{ik} - \hat{Y_{ik}}|, 1 \leq k \leq n. \tag{5.10}$$

A problem that could crop up here is while calculating the nonconformity scores for those examples for which the true rating is not known (because not all ratings are available due to the the sparsity of data sets). One option here is to assign the number of ratings (r) which is greater than the highest nonconformity score as the nonconformity score to all those examples. Because the highest nonconformity score that we get is $|r - p|$, where p is the the predicted rating. Obviously, p should be greater than zero and so the highest nonconformity score is less than r. That means we are assuming that all these examples are highly nonconforming. But this assumption might lead to over conservative prediction regions when we have more number of such examples as all these examples are having the same nonconformity scores.

Nonconformity Measure 6 (NCM6): To deal with the above problem we need to distinguish the nonconformity scores of all examples more precisely. We achieve this by adding the average deviation of the predicted ratings from their true ratings for all the observed ratings in that example (other than the label) to NCM5. Again, we take Y_{ij} as the target entry. We define nonconformity measure for user-based representation as

$$NCM6(z_k) = |Y_{kj} - \hat{Y_{kj}}| + \frac{1}{|\Omega(z_k)| - 1} \sum_{Y_a \in \Omega(z_k) \setminus \{Y_{kj}\}} |Y_a - \hat{Y}_a|, 1 \leq k \leq m. \tag{5.11}$$

Nonconformity measure for item-based representation is defined as follows.

$$NCM6(z_k) = |Y_{ik} - \hat{Y_{ik}}| + \frac{1}{|\Omega(z_k)| - 1} \sum_{Y_a \in \Omega(z_k) \setminus \{Y_{ik}\}} |Y_a - \hat{Y}_a|, 1 \leq k \leq n. \tag{5.12}$$

If two examples are having the same nonconformity scores and if the average deviation of the predicted ratings from their true ratings for all observed ratings in the first example is less than that of the second example then the second example is more nonconforming compared to the first one.

Lemma 1. *The proposed nonconformity measures are invariant of the permutation of the training set. For any permutation π of $\{1, \ldots, n\}$ it should satisfy the Equation* (4.2)

Proof. In matrix factorization, permutation of rows (users) in Y will not change item feature matrix V as only the corresponding user feature vectors in W will be permuted accordingly. Similarly, permuting the columns (items) in Y will not cause any changes in the user feature matrix W. Only the corresponding item feature vectors in V will be permuted accordingly. Hence, it is evident that examples are distributed in i.i.d. fashion and that the computed nonconformity scores are invariant with respect to permutations. □

5.2.4 Proposed Algorithms

In this subsection, we describe the proposed algorithms. As we mentioned earlier user-based and item based transductive CMF (or, inductive CMF) differ among themselves only in the way the examples are represented. Hence, we provide a common algorithm for both user-based and item-based representations. As mentioned in chapter 4, we can compute nonconformity scores either by excluding the user in user-based approach (item in item-based approach) for which we are computing the nonconformity score or by including the item (user) for which we are computing the nonconformity score. We have chosen the second option over the first one in our proposed CP algorithms as the first option is more time consuming than the second one.

5.2.4.1 Transductive Conformal Matrix Factorization

The only assumption in conformal prediction is that all examples are independently and identically distributed (i.i.d. assumption but generally a weaker assumption of exchangeability is sufficient). If we want to predict the rating for a test rating (u_i, i_j), then user u_i (item i_j) becomes the test example and remaining users (items) form the training set in UBTCMF (IBTCMF) . In order to predict the rating, we have to try all possible ratings, for example, 1, 2, 3, 4 and 5, for (u_i, i_j) to see how well each of these ratings conforms to the training set. This can be done as follows:

- Append the test example with the assumed rating to the training set and call it as the appended training set and check how typical the appended sequence is with respect to i.i.d.
- Apply matrix factorization to the appended training set.

- Compute the nonconformity score for every example in the extended training set using any of the nonconformity measures discussed in the previous subsection. This Nonconformity measure tells us how different the test example with the assumed rating is from the other examples in the appended training set.

- The nonconformity score on its own does not really give us any information, it is just a numeric value. However, we can find out how untypical the appended training set is by comparing the nonconformity score of the test example with that of those examples in the training set. This can be done by using the p-value function. We compute p-value for each possible rating.

The p-values obtained for each possible rating can be used in two different modes:

- Point prediction: predict the rating with the highest p-value. The confidence of this prediction is 1 - the second largest p-value and credibility is the p-value of this prediction for that rating.

- Region prediction: Given the ϵ (error probability), output the prediction as the set of all ratings whose p-value greater than ϵ with $1 - \epsilon$ confidence.

Repeat this procedure for every test rating (u_i, i_j) in the test set t.

Algorithm 3 gives a more formal description on transductive conformal matrix factorization.

5.2.4.2 Inductive Conformal Matrix Factorization

In addition to test set, we have a set of calibration ratings which we call calibration set q in ICP. We remove both calibration and test examples from the data set and the remaining examples constitute the proper training set. In other words, all users (items) in UBICMF (IBICMF) except calibration and test examples form the proper training set in ICP. But, in fact, we can use the calibration and test examples without the calibration and test ratings for training along with the examples in proper training set. In other words, all ratings in the data set except the calibration and test ratings can be used to build matrix factorization model. Now, we apply matrix factorization on the resultant training set. Then, we compute nonconformity scores for all calibration ratings in the calibration set using any of the nonconformity measures discussed in the previous

Algorithm 3: Transductive Conformal Matrix Factorization (UBTCMF and IBTCMF).

Input: t, Y, R
Output: Prediction Region, (u_i, i_j), Confidence, Credibility
Set all $(u_i, i_j) \in t$ to unknown
for all $(u_i, i_j) \in t$ **do**
  training set $Z = \{\{U_1, U_2, ..., U_m\} \setminus U_i\}$ in UBTCMF or
  $Z = \{\{I_1, I_2, ..., I_n\} \setminus I_j\}$ in IBTCMF
  for all $a \in R$ **do**
    $(u_i, i_j) = a$
    $E_Z = Z \cup U_i$ in UBTCMF or $E_Z = Z \cup I_j$ in IBTCMF
    Apply MF to E_Z and Calculate $\hat{Y}$.
    Compute nonconformity score for I_j using any of the proposed NCMS.
    for all $z \in Z$ **do**
      Calculate nonconformity scores using any of the NCMs discussed above.
    end for
    Compute p-value *p(a)* using Equation (4.3)
  end for
  Prediction Region = $\{a|p(a) > \epsilon\}$ with confidence 1-ϵ
  OR
  $(u_i, i_j) = a$ such that *max{p(1),p(2),...,p(r)} = p(a)*
  Confidence = *1 - second highest p-value*
  Credibility = *highest p-value*
end for

subsection. Similarly, for each possible rating of the test rating (u_i, i_j), compute its nonconformity score and compare it with the nonconformity scores of calibration ratings using p-value function. As discussed above, these p-values can be used to make point predictions and to produce region predictions. Detailed description can be found in Algorithm 4.

5.2.5 Time Complexity

In this section, we compare the time complexity of the proposed CP algorithms UBTCMF, IBTCMF, UBICMF and IBICMF with the underlying algorithm MMMF. The time required for MMMF (both UBMMMF (i.e. MMMF with the test set same as UBTCMF and UBICMF) and IBMMMF (i.e. MMMF with the test set same as IBTCMF and

Algorithm 4: Inductive Conformal Matrix Factorization (IBICMF and UBICMF).

Input: q, t, Y, R
Output: Prediction Region, (u_i, i_j), Confidence, Credibility
proper training set $Z = \{U_1, U_2, ..., U_m\}$ in UBICMF or $Z = \{I_1, I_2, ..., I_n\}$ in IBICMF
Apply MF to Z and Calculate $\hat{Y}$.
for all each $(u_i, i_j) \in q$ **do**
  Compute nonconformity score using any of the NCMs discussed above.
end for
for all each $(u_i, i_j) \in t$ **do**
  for all each $a \in R$ **do**
    Calculate nonconformity score using any of the NCMs discussed above.
  end for
  Compute p-value *p(a)* using Equation (4.10).
  Prediction Region = $\{a|p(a) > \epsilon\}$ with confidence 1-ϵ
  OR
  $(u_i, i_j) = a$ such that *max{p(1),p(2),...,p(r)} = p(a)*
  Confidence = *1 - second highest p-value*
  Credibility = *highest p-value*
end for

IBICMF)) is $z|\mathcal{O}|(r-1)d$ operations, where z is the number of iterations, because optimizing the equation (5.3) involves calculating the objective function and partial derivatives of the variables involved in it. Therefore, the time complexity of MMMF is $\Theta(z|\mathcal{O}|rd)$.

For UBTCMF and IBTCMF, we need to repeat MMMF $|t|r$ times, where t is the set of test ratings. Because, for every test rating we need to find the p-value (conformity level) of all possible rating values to find the conforming label. In order to compute p-values, we have to compute nonconformity score of all examples (users in UBTCMF and items in IBTCMF). This will require m operations for UBTCMF and n operations for IBTCMF. From this, the time required for UBTCMF is $|t|r(z|\mathcal{O}|(r-1)d + m)$, whereas the time required for IBTCMF is $|t|r(z|\mathcal{O}|(r-1)d + n)$. Therefore the time complexities of UBTCMF and IBTCMF are $\Theta(|t|rz|\mathcal{O}|rd)$. In other words, the time complexity of TCP is $|t|r$ times the time required for MMMF. This is a computationally expensive process especially, for matrix factorization which has long training times.

For UBICMF and IBICMF, we need to execute MMMF only one time. Computing the nonconformity scores of calibration ratings take $|q|$ time. And computing the nonconformity scores for all test items and for all possible labels require $|t|r$ time. The time required for UBICMF and IBICMF is $z|\mathcal{O}|(r-1)d + |q| + |t|r$ and the time complexity is $\Theta(z|\mathcal{O}|rd)$ which is the same as the underlying algorithm MMMF.

5.3 Experiments and Results

We tested our algorithms on four data sets: Movielens100K, Eachmovie, Movielens 1M and Movielens-small-latest. First, third and fourth data set contains ratings from 1 to 5, second data set from 1 to 6. From each data set, we took a subset consisting of 100 users and 100 items which had the highest number of ratings. This is because of the computational complexity of TCP. In our case, we have r ratings and number of ratings to be predicted is $|t|$. So, we need to apply matrix factorization $r|t|$ times. This is computationally very expensive. For instance, the smallest data set among the bench mark data sets in recommender systems is Movielens100K. For this data set if we consider 20% of ratings (20000) as the test ratings then we need to perform matrix factorization 20000X5 =100000 times (where 5 is the number of possible ratings). If we assume that time required to do the matrix factorization once is 60 seconds then TCP algorithm will take approximately 70 days to predict the ratings. This is even higher for large data sets such as Eachmovie and Movielens 1M, where the number of available ratings or the number of possible ratings *(r)* is more compared to Movielens100K. Therefore, TCP is not a practical approach for matrix factorization problem. But we applied this to a small data set just to demonstrate the applicability of TCP and compare its performance with ICP. We performed our experiments using MMMF as the underlying algorithm. The main reason for selection of MMMF is its ability to produce discrete rating values as output. So measuring the nonconformity becomes an easy task. As mentioned above, we do not have any bound on d (number of latent features (rank)) in MMMF. But there is no need to consider the the $rank > max(m, n)$. Here we use a value of $d = 100$, $\lambda = 4.5$ and $z = 100$.

We first discuss experimental setup and then we compare the performance of MMMF with CP in terms of MAE and RMSE. Next, we show the validity and efficiency of our proposed conformal prediction algorithms. Finally, we present the mean confidence

and mean credibility values produced by our proposed CP algorithms when forced to make point predictions.

1. Experimental Setup

For TCP, each data set is divided into 5 equal parts wherein each part consists of 20 examples. We repeat our algorithm 5 times, each time using 20% of the ratings in each example in one of the 5 parts (the examples in this part becomes the test examples) as the test set and the remaining parts as the training set. Note that here we consider only 20% of ratings in each example in one of the parts as the test set. This is because if we set all entries in the example to zero then that example becomes new user in UBTCMF or new item in IBTCMF (cold-start problem). Then we cannot predict a rating without using any information (previous ratings) about that user or item, especially in collaborative filtering, without using any method to deal with this cold-start problem. To make our TCP to be compatible with the original TCP where the training and test examples are distinct, we restricted test ratings to some examples only. But, we can also select the test ratings randomly.

Similarly, for ICP, 20% of the ratings in each example in one of the 5 parts is considered as the calibration set (the examples in this part become the calibration examples). 20% of the ratings in each example in one of the other 4 parts is considered as the test set (the examples in this part becomes the test examples). The ratings in the examples of the remaining 3 parts is considered as the proper training set and the process is repeated 5 times. Here also, in order to be compatible with the standard ICP, we took calibration and test ratings from some examples only and we call those examples as calibration and test examples. We also restricted that the calibration and test examples are distinct. But, in general, we can take calibration and test ratings from any example. For example, we can take 5% of ratings from all examples as calibration ratings and another 5% as test ratings. In this case, all examples are considered as calibration and test examples and proper training set constitutes all examples without calibration and test ratings. Here, notice the difference between calibration examples and calibration ratings and between test examples and test ratings. In case of ICP, the calibration examples and test examples (unlike CP in machine learning) may overlap but not calibration ratings and test ratings.

2. Comparative Analysis

In order to compare conformal matrix factorization (CMF) with Maximum Margin Matrix Factorization (MMMF), we have to make single point predictions. In single point predictions, we output a rating with the highest p-value. If more than one rating shares this p-value then we can take any one of those ratings. In our experiments, we take two ratings: one that is closer to the true rating (most conforming rating(MCR)) and another one which is far away from the true rating (most nonconforming rating (MNR)). In this way, we can measure the certainty in predictions. If the performance of the algorithm with MCR is close to the one with MNR, then we can say that the certainty of the algorithm is high. To make things clear suppose that the p-values for the possible rating values of 1, 2, 3, 4, 5 are as given below:

Possible rating:	1	2	3	4	5
p-value:	0.2	0.56	0.85	0.85	0.44

Here, we do not refer the item for which we want to predict the rating as the test item. Test items (or test examples) in our case refers to the set of ratings given to an item from which we are choosing some percentage of ratings as the test set (in UBTCMF/UBICMF). We refer each such rating in the test set as the test rating. In other words, test rating is some entry in the test item for which we want to predict the rating. In the example given above, both ratings '3' and '4' share the highest p-value and therefore, both 3 and 4 are the predicted labels for the test rating. Suppose that the true rating for the test rating is '5'. Then '4' is the most conforming rating (MCR) (or closest rating) for the test rating and '3' is the most nonconforming rating (MNR) (rating with the maximum deviation from true rating). The absolute error value with MCR is '1' (abs(5-4)) and that with MNR is '2' (abs(5-3)). The absolute error with MCR is the minimum error and the absolute error with MNR is the maximum error. In the example given above, MCR and MNR are very close to each other and as a result the error value we get with MCR is also close to what we get with MNR. The performance of the algorithm is measured with MAE in our paper and it can be seen that the performance of the algorithm with MCR (closest rating) is very close to the performance with MNR (rating with maximum deviation). On the other hand suppose that the p-values for the possible rating values are taken as given below: Here, '1' and

Possible rating:	1	2	3	4	5
p-value:	0.85	0.56	0.2	0.85	0.44

'4' share the highest p-value. Therefore, '1' and '4' are the predicted ratings for the given test rating and '4' becomes the MCR and '1' becomes the MNR. In this case, absolute error values with MCR and MNR are '1' and '4' respectively. Therefore, the performance of the algorithm with MCR (closest rating) is very far from MNR (rating with maximum deviation). Consider a third scenario as given below: In this

Possible rating:	1	2	3	4	5
p-value:	0.66	0.56	0.2	0.85	0.44

case, only one rating ('4') shares the highest p-value and MCR/MNR are the same as well as their absolute error values are also the same. Therefore, the performance of the algorithm with MCR (closest rating) is the same as with MNR (rating with maximum deviation). In other words, the performance difference is zero. Among these three cases, the certainty of the algorithm in the third case is high (In other words, there is no ambiguity in choosing a single rating when making point predictions and therefore, there is no uncertainty in the prediction) and is therefore more preferable.

In Table 5.1 we compare the performance of UBTCMF and IBTCMF algorithms with that of MMMF in terms of MAE and RMSE with most conforming ratings (MCR) and most nonconforming ratings (MNR). Table 5.2 compares the prediction accuracy of MMMF with that of UBICMF and IBICMF. In Tables 5.1 and 5.2 UBMMMF and IBMMMF are user-based MMMF and Item-based MMMF respectively. They are similar to the original MMMF where the test ratings of UBMMMF are the same as the test ratings used in UBTCMF or UBICMF and test ratings of IBMMMF are same as the test ratings used in IBTCMF or IBICMF. As discussed above, if only one rating is having the highest p-value then, there will be no uncertainty in that prediction. Instead, if more than one rating shares the highest p-value, all such rating values becomes the candidate predicted ratings for the test rating. This is not acceptable while making point predictions. Therefore, we calculate the uncertainty in the predictions as the percentage of test ratings having more than one rating sharing the highest p-value. As the percentage of uncertainty is increased the deviation between minimum and maximum

errors will also be increased as shown in Tables 5.1 and 5.2. Therefore, along with the performance comparison, we also show the uncertainty involved in the predictions for all NCMs for each data set in Table Tables 5.1c and 5.2c.

Dataset	Performance Measure	Algorithm												
		UBMMMF	Odinal UBTCMF											
			NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
			MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	MAE	**0.672**	0.932	1.4486	0.9595	1.3931	0.7964	1.1672	0.7975	1.1484	0.5267	0.871	**0.6454**	**0.7344**
	RMSE	**0.9705**	1.3672	1.8592	1.3858	1.7942	1.1833	1.5167	1.1572	1.5021	0.8403	1.1561	**0.9456**	**1.04**
Eachmovie	MAE	**0.951**	1.5803	2.2017	1.5854	2.1033	1.4249	1.8977	1.486	1.8812	0.5055	1.6221	**0.9733**	**1.2006**
	RMSE	**1.3572**	2.1389	2.6777	2.1336	2.5957	1.971	2.3827	2.0283	2.3752	0.9366	2.059	**1.4764**	**1.7043**
Movielens 1M	MAE	**0.644**	0.9724	1.5898	1.0341	1.5328	0.8294	1.2392	0.8248	1.2366	0.5155	0.8226	**0.6403**	**0.7078**
	RMSE	**0.9515**	1.4389	1.9995	1.5027	1.9529	1.2271	1.6005	1.2254	1.5988	0.8462	1.1184	**0.9642**	**1.0219**
Movielens-small-latest	MAE	**0.6155**	0.8641	1.3958	0.8911	1.3484	0.7087	1.0691	0.7372	1.0112	0.4891	0.682	**0.5638**	**0.6072**
	RMSE	**0.8727**	1.3462	1.868	1.3709	1.8355	1.1177	1.4647	1.1384	1.4012	0.8027	0.9759	**0.8687**	**0.9117**

(a) UBMMMF with UBTCMF

Dataset	Performance Measure	Algorithm												
		IBMMMF	Odinal IBTCMF											
			NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
			MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	MAE	**0.6682**	0.9174	1.5432	0.9619	1.4605	0.7275	1.2084	0.7454	1.1284	0.5378	0.8275	**0.6539**	**0.7274**
	RMSE	**0.9616**	1.3919	1.9624	1.4459	1.9047	1.1451	1.5777	1.1433	1.5214	0.8543	1.0948	**0.9571**	**1.0222**
Eachmovie	MAE	**0.8416**	1.5781	2.0938	1.6719	2.1025	1.2671	1.6858	1.2672	1.6389	0.4848	1.3169	**0.8494**	**0.9744**
	RMSE	**1.1653**	2.172	2.6117	2.2539	2.6172	1.8085	2.1956	1.7802	2.1502	0.8532	1.6684	**1.2579**	**1.3705**
Movielens 1M	MAE	**0.5916**	0.9755	1.5617	1.0174	1.4737	0.7163	1.1242	0.7063	1.0945	0.4602	0.7379	**0.566**	**0.621**
	RMSE	**0.893**	1.4997	2.0388	1.5322	1.9521	1.128	1.5219	1.1205	1.503	0.7811	1.0256	**0.8902**	**0.9377**
Movielens-small-latest	MAE	**0.6716**	0.854	1.4549	0.9084	1.3601	0.629	1.0129	0.6264	1.0066	0.5398	0.7317	**0.6098**	**0.6631**
	RMSE	**0.9444**	1.3319	1.909	1.3815	1.8195	0.984	1.3552	0.9819	1.3648	0.8461	1.0137	**0.918**	**0.9635**

(b) IBMMMF with IBTCMF

Dataset	Algorithm											
	Odinal UBTCMF						Ordinal IBTCMF					
	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	0.297	0.2475	0.2551	0.2268	0.3177	**0.0720**	0.3367	0.2713	0.3033	0.2444	0.2851	**0.0729**
Eachmovie	0.2732	0.2393	0.241	0.1995	0.6583	**0.0434**	0.2515	0.214	0.2249	0.1982	0.6263	**0.1124**
Movielens 1M	0.3153	0.259	0.2701	0.2667	0.2878	**0.1246**	0.292	0.2476	0.2549	0.2353	0.27	**0.0523**
Movielens-small-latest	0.3217	0.2425	0.2208	0.1836	0.1907	**0.0664**	0.3675	0.2508	0.272	0.2622	0.1878	**0.0516**

(c) Uncertainty in predictions of UBTCMF & IBTCMF

Table 5.1: Performance comparison of MMMF with UBTCMF and IBTCMF with MCR and MNR and Uncertainty in UBTCMF & IBTCMF

It is clear from Tables 5.1 and 5.2 that except with NCM5 and NCM6, conformal prediction algorithm is having the worst performance when compared to that of MMMF with all other nonconformity measures. This performance deterioration is expected because of the reasons mentioned above. Although our TCP algorithm with NCM5 when most conforming ratings are taken into account is outperforming the original MMMF, we cannot say that this is a good nonconformity measure. Because this performance improvement is due to the high uncertainty compared to NCM6 involved

Dataset	Performance Measure	Algorithm												
		UBMMMF	Odinal UBICMF											
			NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
			MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	MAE	**0.672**	0.5118	1.2755	0.5385	1.1526	0.5737	0.8893	0.5646	0.989	**0.6718**	**0.6718**	**0.6718**	**0.6718**
	RMSE	**0.9705**	0.8364	1.7463	0.8657	1.6335	0.8884	1.2757	0.885	1.3765	**0.9708**	**0.9708**	**0.9708**	**0.9708**
Eachmovie	MAE	**0.951**	0.8434	1.4314	0.7926	1.5077	0.9097	1.0619	0.8867	1.0941	**0.9591**	**0.9591**	**0.9591**	**0.9591**
	RMSE	**1.3572**	1.289	1.9908	1.2362	2.1043	1.3168	1.5096	1.3044	1.5447	**1.3699**	**1.3699**	**1.3699**	**1.3699**
Movielens 1M	MAE	**0.644**	0.4889	1.2118	0.4751	1.2891	0.5259	1.0091	0.5145	1.0325	**0.6458**	**0.6458**	**0.6458**	**0.6458**
	RMSE	**0.9515**	0.8317	1.7075	0.8212	1.8012	0.8553	1.4234	0.8478	1.4501	**0.9551**	**0.9551**	**0.9551**	**0.9551**
Movielens-small-latest	MAE	**0.6155**	0.4551	1.1391	0.4529	1.1086	0.4516	1.0127	0.426	1.0425	**0.5705**	**0.5705**	**0.5705**	**0.5705**
	RMSE	**0.8727**	0.7863	1.6987	0.7724	1.6381	0.7773	1.477	0.7524	1.4665	**0.8732**	**0.8732**	**0.8732**	**0.8732**

(a) UBMMMF with UBICMF

Dataset	Performance Measure	Algorithm												
		IBMMMF	Odinal IBICMF											
			NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
			MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	MAE	**0.6682**	0.607	0.8982	0.5587	1.0583	0.6241	0.7818	0.5835	0.8615	**0.6772**	**0.6772**	**0.6772**	**0.6772**
	RMSE	**0.9616**	0.9192	1.2863	0.8759	1.5267	0.928	1.148	0.8983	1.2408	**0.9743**	**0.9743**	**0.9743**	**0.9743**
Eachmovie	MAE	**0.8416**	0.7386	1.2731	0.747	1.2406	0.7895	0.992	0.8235	0.898	**0.8432**	**0.8432**	**0.8432**	**0.8432**
	RMSE	**1.1653**	1.093	1.8326	1.0966	1.7633	1.1345	1.3954	1.1558	1.2582	**1.1696**	**1.1696**	**1.1696**	**1.1696**
Movielens 1M	MAE	**0.5916**	0.5226	0.9055	0.4591	1.197	0.5355	0.8267	0.5121	0.8736	**0.6**	**0.6**	**0.6**	**0.6**
	RMSE	**0.893**	0.8467	1.3942	0.7881	1.7342	0.8588	1.238	0.8289	1.3063	**0.9078**	**0.9078**	**0.9078**	**0.9078**
Movielens-small-latest	MAE	**0.6716**	0.5489	0.9779	0.5287	1.1084	0.5975	0.7478	0.5282	0.9546	**0.6454**	**0.6454**	**0.6454**	**0.6454**
	RMSE	**0.9444**	0.8702	1.4311	0.8452	1.597	0.8986	1.0841	0.8416	1.3708	**0.937**	**0.937**	**0.937**	**0.937**

(b) IBMMMF with IBICMF

Dataset	Algorithm											
	Odinal UBICMF						Ordinal IBICMF					
	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	0.2493	0.2038	0.1408	0.1774	**0**	**0**	0.0951	0.1641	0.0595	0.12	**0**	**0**
Eachmovie	0.1494	0.179	0.0603	0.0909	**0**	**0**	0.1335	0.1276	0.0754	0.0266	**0**	**0**
Movielens 1M	0.2408	0.2721	0.1991	0.2293	**0**	**0**	0.1208	0.2287	0.1184	0.1378	**0**	**0**
Movielens-small-latest	0.2127	0.2094	0.2303	0.2741	**0**	**0**	0.223	0.1845	0.0718	0.1794	**0**	**0**

(c) Uncertainty in predictions of UBICMF & IBICMF

Table 5.2: Performance comparison of MMMF with UBICMF and IBICMF with MCR and MNR and Uncertainty in UBICMF & IBICMF

in predictions and this fact is clear from Table 5.1c. The reason for this is the unavailability of true ratings for some of the examples in the training set. What we actually need is a nonconformity measure which gives the performance as close as possible to MMMF and is able to produce the predictions with utmost certainty. Table 5.1 shows that the performance of TCP algorithm with NCM6 is very close to MMMF algorithm and it is producing certain predictions as shown in Table 5.1c compared to the other NCMs. This reduction in uncertainty for NCM6 is obtained by distinguishing nonconformity scores of all training examples more precisely by adding the average deviation of the predicted ratings from their true ratings for all other observed ratings in that example. In case of ICP the uncertainty is zero for both NCM5 and NCM6 as shown in Table 5.2c, because the true rating is always available for calibration examples. As

a result, these two NCMs are giving the same predictive accuracy for both MCR and MNR as shown in 5.2.

3. Validity and Efficiency of CP Algorithms

Validity at specified error probability (ϵ) is satisfied, if at least $(1 - \epsilon)$ of the prediction regions of test ratings at that ϵ contain true ratings. For example, validity of prediction regions produced at 95% confidence level (where $\epsilon = 0.05$) are said to be valid if at least 95% of the prediction regions contain true label. In other words, prediction regions are said to be valid, if error percentage is always bounded by ϵ. All conformal predictors are automatically valid. Figure 5.1 shows validity (in terms of error percentage) of TCP and ICP for different data sets respectively (for NCM5 and NCM6). From Figures 5.1a and 5.1b it is clear that prediction regions produced by TCP are somewhat conservative at higher confidence levels i.e., prediction regions at these confidence levels include almost all possible ratings and NCM5 is over conservative compared to NCM6. In case of ICP, validity with respect to NCM6 follows a straight line as required, whereas with NCM5 it is like a step graph as shown in Figures 5.1c and 5.1d.

As discussed in the previous chapter, usefulness of conformal predictors depend on its efficiency. Vovk et al.[74] proposed ten different ways of measuring the efficiency of CPs. In our experiments we use Observed Excess criterion which gives the average number of false labels (ANFL) included in the prediction set at significance level ϵ. The formula to calculate ANFL is:

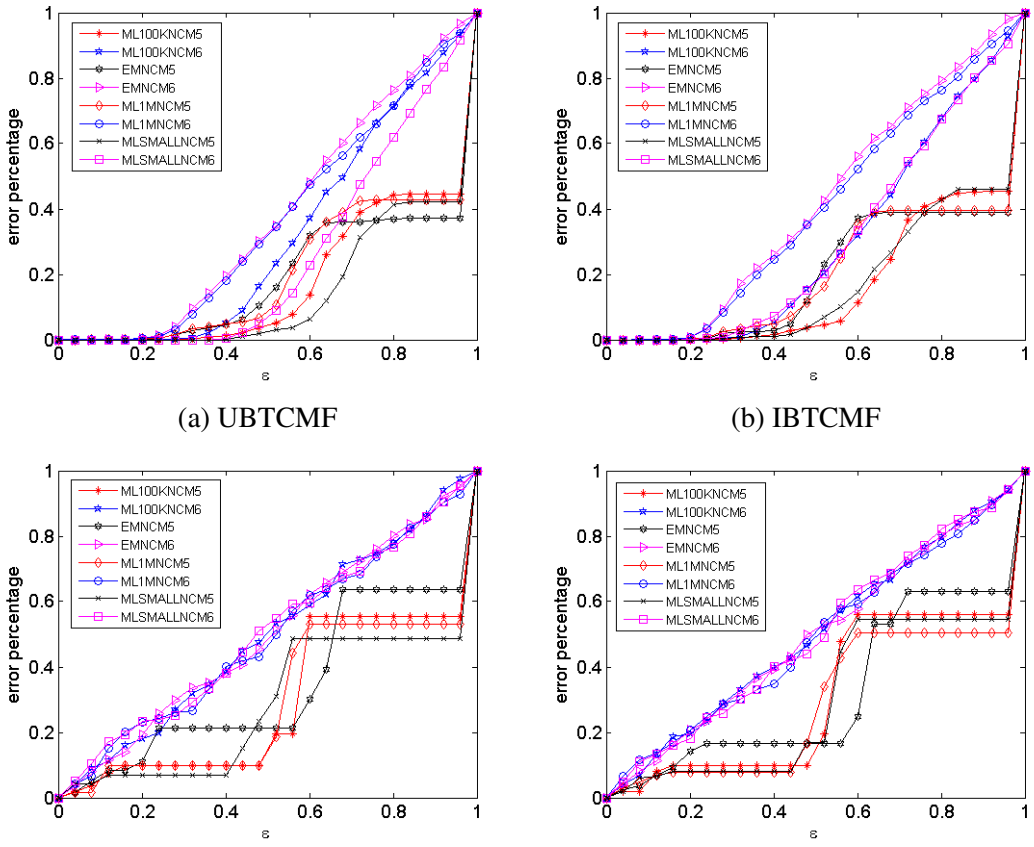
$$ANFL = \frac{\sum_{i=1}^{|t|} |\Gamma_i^\epsilon \setminus T_i|}{|t|}. \tag{5.13}$$

where Γ_i^ϵ is the prediction region for the i^{th} test item at significance level ϵ

T_i is the true label of the i^{th} test element

$|t|$ is the total number of test items.

The efficiency of our algorithm in terms of ANFL for different data sets is shown in 5.2 for TCP and ICP (NCM5 and NCM6). From these graphs, it is clear that prediction regions produced by our algorithm are too wide at higher confidence levels i.e., prediction regions at these confidence levels include almost all possible ratings. This is also the reason why our prediction regions are too conservative (Figure 5.1). From



(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

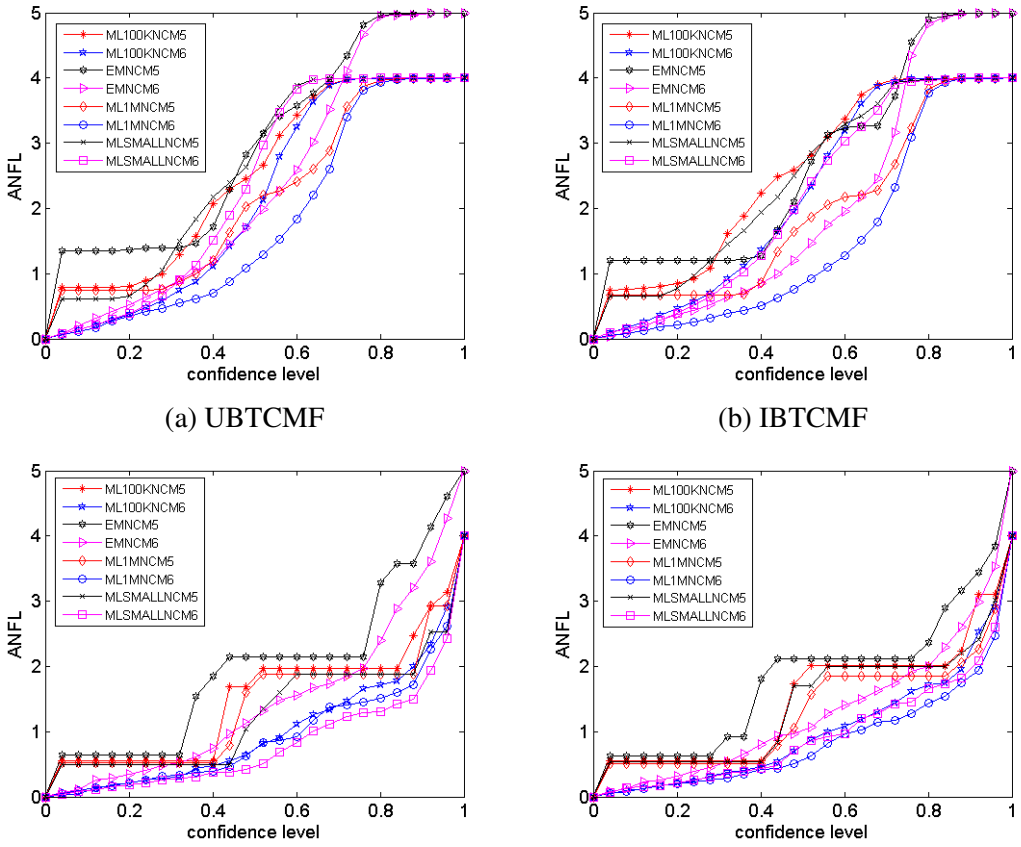
Figure 5.1: Validity (in terms of error percentage) of UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

Figure 5.2 we can observe that compared to NCM5, NCM6 is producing less number of false labels for both TCP and ICP.

In addition to ANFL, we use three other criteria to compute the efficiency of conformal predictors [58, 74]:

1. Percentage of test elements having prediction regions with single rating.
2. Percentage of test elements having prediction regions with more than one rating.
3. Percentage of test elements having empty prediction region.

A conformal predictor is said to be efficient when the percentage of second and third criteria are relatively small whereas the percentage of first should be high especially at $99\%, 95\%$ and 90% confidence levels. Although our algorithm is able to



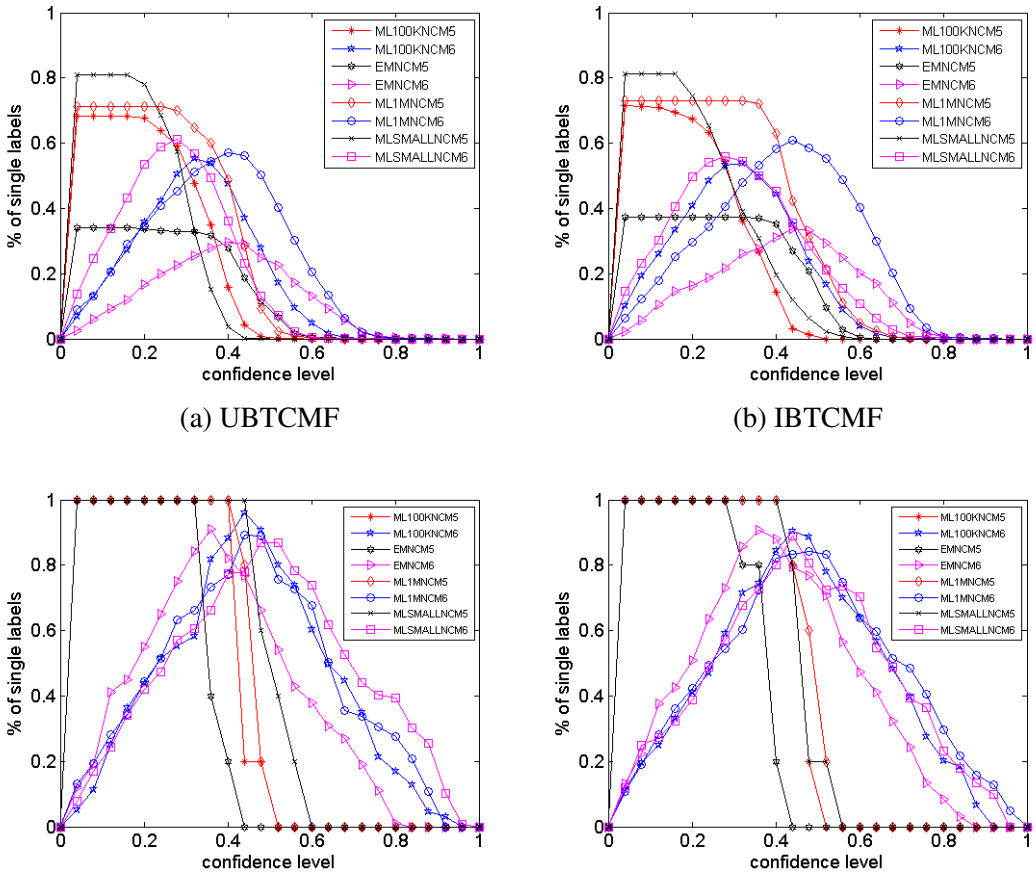
(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.2: ANFL of UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

generate empty prediction sets at these confidence levels, it failed in minimizing the second one and maximizing the first one at these confidence levels. The percentage of test ratings having prediction regions producing single rating at different confidence levels for different data sets is shown in Figure 5.3 for both TCP and ICP (NCM5 and NCM6). From Figure 5.3, it is clear that NCM6 is outperforming NCM5 in producing the single labels at high confidence levels, as NCM6 starts producing larger number of single labels at higher confidence levels compared to NCM5 if we go from higher to lower confidence levels.

Figure 5.4 shows the percentage of correct predictions among the percentage of single labels produced by NCM6 at each confidence level. We can see that the per-



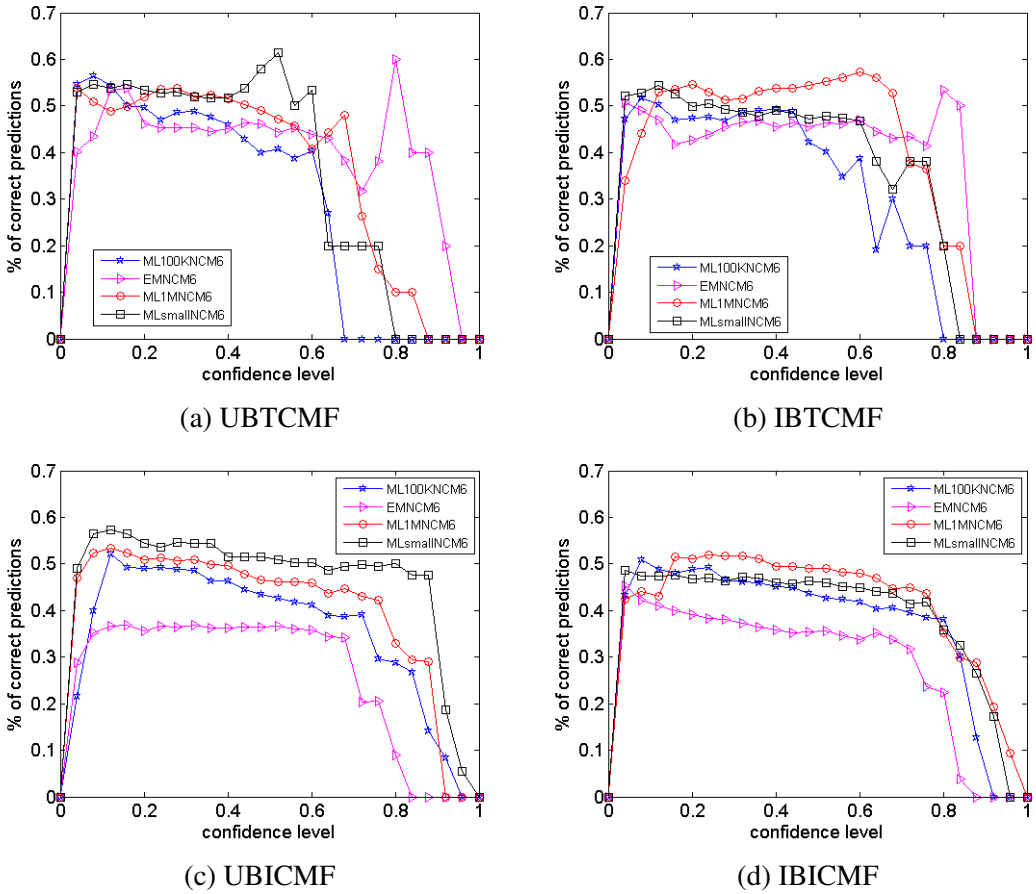
(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.3: %of single labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

centage of correct predictions at any confidence level is not exceeding 60% at all confidence levels. This percentage of correct predictions is not high, as in [61] the authors showed that their algorithm produced 97.9% - 100% correct predictions even at very high confidence levels. Their algorithm was also able to produce large number of single prediction regions at very high confidence levels 99% and 95%. But in our results, we are not able to produce good number of single prediction regions at high confidence levels due to very close possible rating values like 1-5 or 1-10 and the percentage of correct predictions is also moderate.

The percentage of test ratings having prediction regions with more than one rating at different confidence levels for different data sets is shown in Figure 5.5 for both TCP



(a) UBTCMF (b) IBTCMF

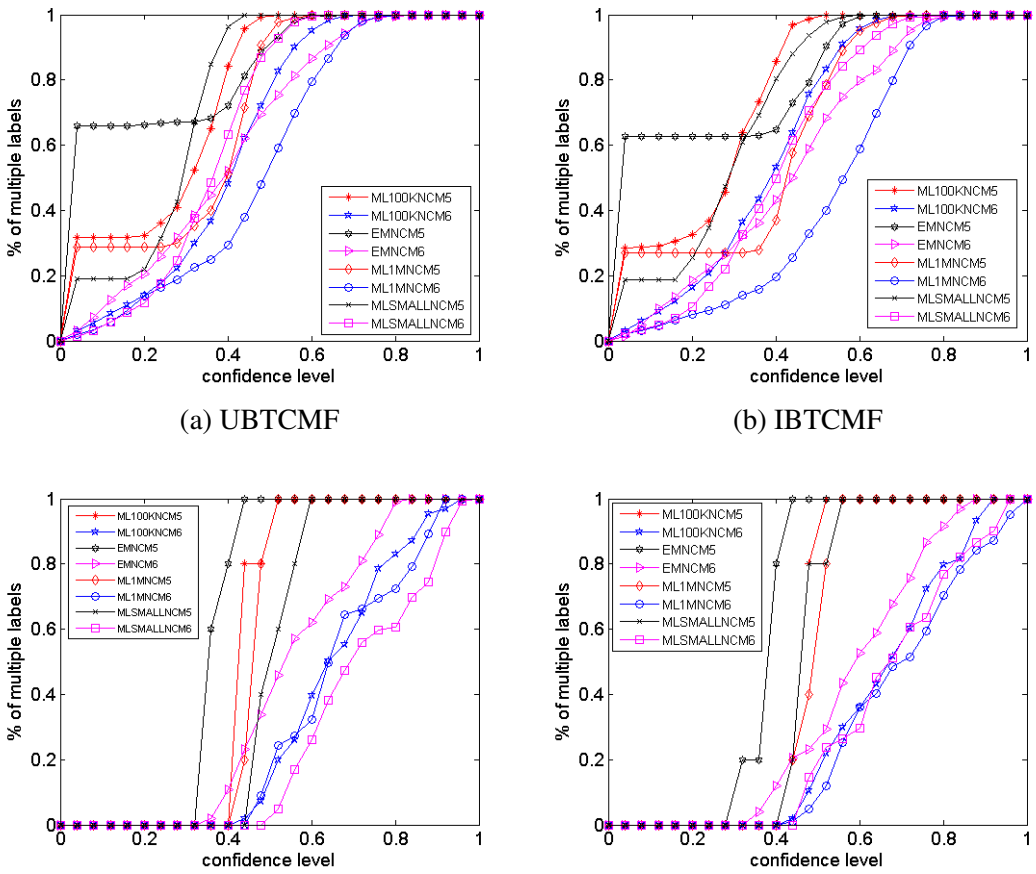
(c) UBICMF (d) IBICMF

Figure 5.4: %of correct predictions made by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

and ICP (NCM5 and NCM6). In this case also NCM6 is performing better than NCM5 as the reduction in number of labels starts earlier in NCM6 compared to NCM5 when we consider confidence levels from higher to lower.

The percentage of test ratings having empty prediction regions at different confidence levels for different data sets is shown in Figure 5.6 for both TCP and ICP (NCM5 and NCM6). Here, in contrast to earlier results, NCM5 is outperforming NCM6, as the percentage of empty labels is zero at all confidence levels in case of NCM5, whereas NCM6 is producing empty labels at lower confidence levels.

From the above figures, it is clear that ICP produces more efficient predictions than TCP. This might not be true in all cases. For instance, when we apply CP to



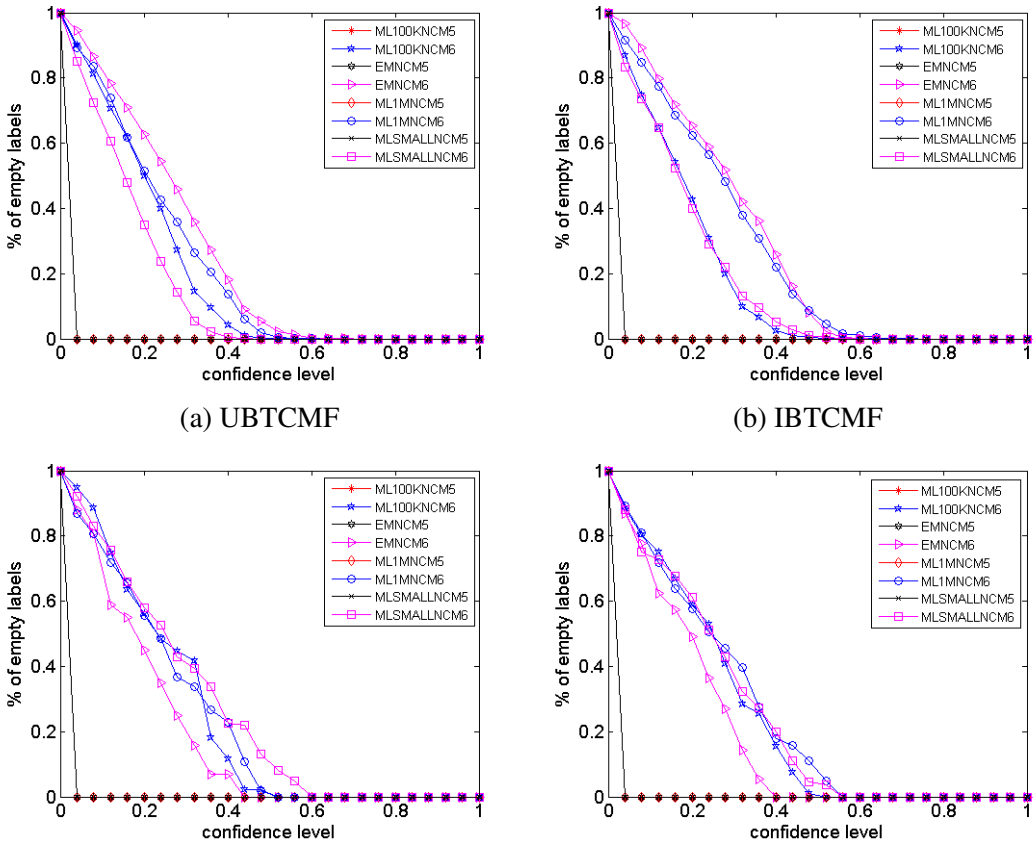
(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.5: %of multiple labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

machine learning algorithm, TCP usually outperforms ICP. Because the number of training examples is more in TCP compared to ICP. This performance gap becomes negligible as the training set size increases. In addition to number of training examples, the performance of CP also depends on the nonconformity measure defined for the algorithm. As an example, in [59], the authors proposed six nonconformity measures. Among them, TCP is limited to using only the first two nonconformity measures. They compared the widths of the regions produced by the TCP and ICP approaches for different nonconformity measures (for regression task but not for classification task). From their experiments, they concluded the following:

- For the standard regression nonconformity measure TCP produces tight predic-



(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.6: %of empty labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

tion regions compared to ICP due to the much richer set of examples it uses for calculating its predictive regions. The difference in predictive region tightness is much bigger in the results where the number of calibration examples are less.

- The region widths of the two approaches with the first two nonconformity measures are in most cases very similar in terms of distribution. This shows that these measures are not so dependent on the composition of the examples used for producing the predictive regions.

- Furthermore, when the region widths of the TCP with the first two nonconformity measures are compared with those of the ICP with the last two nonconformity measures, in many cases the regions of the ICP are somewhat tighter,

despite the much smaller set of examples it uses to compute them. This is due to the superiority of the last two nonconformity measures.

From this, we can see that the efficiency of the prediction regions not only depends on the number of examples used for producing the prediction regions but also on the efficiency of nonconformity measures.

But in our case, there are three reasons why our ICP produces more efficient prediction regions than TCP:

- Uncertainty is one of the reasons for not getting the tight prediction regions in TCP. As we see from 5.2c, uncertainty in predictions made by ICP for both NCM5 and NCM6 is zero in matrix factorization. For example, the predicted value for a test rating is 2. Then nonconformity scores of NCM5 for possible ratings 1, 2, 3, 4 and 5 are 1, 0, 1, 2 and 3. Therefore, only one rating will get highest p-value (in this example '2'). But, in case of TCP, we have to apply matrix factorization for every possible rating of a test rating by substituting that corresponding rating in place of the test rating. As we know, matrix factorization tries to approximate predicted matrix with the original matrix with respect to the observed ratings. So there is a possibility that if we put '1' for a test rating and if that is conforming with other observed entries in the matrix, it will be predicted as '1'. Same is the case for other possible ratings. Because of this, it is possible that two or more ratings can share the highest possible ratings as these ratings share same nonconformity scores. Due to this reason, in contrast to ICP, uncertainty in TCP is not zero. Clearly, this uncertainty is one of the main reasons for getting wider prediction regions in TCP. For example, two ratings 3 and 4 share the highest p-value which is, say, '0.98', then we cannot get the prediction region of single rating even at 5% confidence level as this prediction region contains both values 3 and 4.

- In case of TCP, the problem with NCMs 5 and 6 is that calculating nonconformity scores for those examples for which the true rating is not known. In our algorithm, we assigned the number of ratings as the nonconformity score to all such examples. That means we are assuming that all these examples are highly nonconforming. But this might lead to moderately high p-values even for the

wrongly predicted ratings which in turn leads to over conservative prediction regions, especially, when we have more number of such examples, because all these examples are having the same nonconformity scores in NCM5. Though this problem to some extent will be reduced for point predictions in case of NCM6 as shown in 5.1c where uncertainty is less than that of NCM5, it will still be a problem for region predictions.

- To compute the p-values:
  - In TCP, we compare nonconformity score of the test example with that of nonconformity scores of other training examples. In our case, we have only 99 training examples (in all data sets, we have only 100 users and 100 items).
  - In ICP, we compare nonconformity score of the test example with that of nonconformity scores of calibration ratings. The number of calibration ratings are very high compared to the number of training examples.

  This is entirely different from standard TCP and ICP where in standard TCP more number of examples are available for computing the predictive regions compared to ICP.

4. Confidence and Credibility

Tables 5.3 and 5.4 give mean confidence and mean credibility values for TCP and ICP respectively. We also calculated mean difference between highest p-value and the second highest p value for all test ratings. We will get confidence predictions when this difference is high. From Table 5.3, we can observe that this difference is small in TCP which results in less confidence predictions which in turn will end up in producing multiple labels at higher confidence levels. In this aspect, ICP is showing better results in producing large percentage of single labels at high confidence levels compared to TCP as shown in Table 5.4. As mentioned above, the reasons might be zero uncertainty in predictions and unlike TCP, the true ratings are always available to compute nonconformity scores in case of ICP. One more reason is that in our ICP, we are comparing the nonconformity score of the test example with more number of

Dataset	Performance Measure	Algorithm											
		Ordinal UBTCMF						Ordinal IBTCMF					
		NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	mean confidence	0.3523	0.5139	0.3491	0.5226	0.3449	**0.3939**	0.2747	0.5051	0.2908	0.5117	0.3238	**0.3814**
	mean credibility	0.6812	0.5327	0.6841	0.5259	1	0.8037	0.7581	0.5397	0.7436	0.5322	1	0.827
	mean p1-p2	0.0264	0.0382	0.0264	0.0407	0.2391	0.1804	0.0239	0.0354	0.0258	0.0347	0.2317	0.1894
Eachmovie	mean confidence	0.2905	0.5056	0.3115	0.5186	**0.4473**	0.398	0.2777	0.5124	0.2985	0.5429	**0.4737**	0.4457
	mean credibility	0.7422	0.5388	0.7221	0.5248	1	0.747	0.7554	0.5341	0.7363	0.5005	1	0.7265
	mean p1-p2	0.0264	0.0379	0.0282	0.0375	0.1525	0.1184	0.0266	0.0386	0.0284	0.0372	0.1785	0.1403
Movielens 1M	mean confidence	0.2829	0.4901	0.2945	0.5016	0.4172	**0.4685**	0.2763	0.5175	0.3064	0.5336	0.4704	**0.5389**
	mean credibility	0.7502	0.5513	0.7404	0.5448	1	0.7821	0.7555	0.5241	0.7276	0.5073	1	0.7408
	mean p1-p2	0.0257	0.0338	0.0273	0.0373	0.2966	0.2303	0.0249	0.0341	0.0264	0.0338	0.3455	0.2616
Movielens-small-latest	mean confidence	0.3528	0.4812	0.3586	0.4821	0.3024	**0.3555**	0.3083	0.4921	0.3222	0.5	0.3285	**0.4048**
	mean credibility	0.6835	0.5655	0.6773	0.5611	1	0.8459	0.7277	0.5517	0.7142	0.5401	1	0.8257
	mean p1-p2	0.0295	0.0393	0.03	0.037	0.2452	0.1919	0.0276	0.0346	0.027	0.0312	0.2641	0.2168

Table 5.3: Mean confidence and Mean credibility of UBTCMF and IBTCMF

Dataset	Performance Measure	Algorithm											
		Ordinal UBICMF						Ordinal IBICMF					
		NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	mean confidence	0.4899	0.5154	0.4904	0.5175	0.4307	**0.6529**	0.5043	0.5328	0.5035	0.522	0.4498	**0.6682**
	mean credibility	0.5511	0.5402	0.5566	0.5353	1	0.7633	0.5489	0.5297	0.5495	0.5372	1	0.7636
	mean p1-p2	0.0407	0.0554	0.0439	0.0502	0.4307	0.4162	0.0531	0.0623	0.0523	0.0562	0.4498	0.4317
Eachmovie	mean confidence	0.4908	0.5144	0.4767	0.4935	0.3668	**0.558**	0.4895	0.5085	0.4817	0.4984	0.3714	**0.5954**
	mean credibility	0.5517	0.533	0.5662	0.5481	1	0.8147	0.5701	0.5531	0.5685	0.5535	1	0.8117
	mean p1-p2	0.0423	0.0472	0.0418	0.04	0.3668	0.3726	0.0595	0.0615	0.049	0.0514	0.3714	0.4071
Movielens 1M	mean confidence	0.4994	0.5158	0.4961	0.5109	0.4644	**0.6704**	0.5042	0.5143	0.5001	0.5164	0.4837	**0.7023**
	mean credibility	0.5469	0.5307	0.5548	0.5365	1	0.763	0.5538	0.5386	0.5515	0.5357	1	0.7522
	mean p1-p2	0.0461	0.0462	0.0482	0.0448	0.4644	0.4334	0.0579	0.0527	0.0501	0.0511	0.4837	0.4545
Movielens-small-latest	mean confidence	0.5018	0.5214	0.5064	0.5262	0.5095	**0.7201**	0.4965	0.5316	0.4869	0.5171	0.4601	**0.6796**
	mean credibility	0.5423	0.5217	0.532	0.5185	1	0.7344	0.5615	0.5299	0.5645	0.5347	1	0.7596
	mean p1-p2	0.0438	0.0429	0.0347	0.039	0.5095	0.4546	0.0578	0.0613	0.0503	0.0459	0.4601	0.4392

Table 5.4: Mean confidence and Mean credibility of UBICMF and IBICMF

calibration ratings to produce prediction regions as compared to the number of training examples in ICP.

We did not show the validity and efficiency of CP with NCMs 1 to 4, as the efficiency of CP with these NCMs is very less compared to NCM5 and NCM6, although their validity property is satisfied.

In summary,

- NCM6 is the best nonconformity measure compared to all other nonconformity measures in terms of prediction accuracy and efficiency for both TCP and ICP.
- As far as the prediction accuracy is concerned, TCP is outperforming ICP (with MCR), because of the more number of ratings used for training and the higher uncertainty values in TCP compared to ICP. But from efficiency perspective, ICP is showing better results compared to TCP as there is no uncertainty involved in

predictions produced by ICP and also the mean confidence and credibility values are higher for predictions made by ICP than that of TCP.

- Though, validity is satisfied for both TCP and ICP, prediction regions produced by TCP are too wide at higher confidence levels i.e., prediction regions at these confidence levels include almost all possible ratings and NCM5 is over conservative compared to NCM6. In case of ICP, validity with respect to NCM6 follows a straight line as required, whereas with NCM5 it is like a step graph.
- The time required for TCP is many times greater than ICP. As mentioned above, TCP cannot be used in this context as the matrix factorization requires long training time and most of the data sets in recommender systems contain more number of ratings.
- Although our algorithms failed in producing single labels at higher confidence levels due to very close possible rating values, we can use this algorithm to make single point predictions with associated confidence and credibility.

5.4 Summary

In this work, we have given different ways of applying conformal prediction to Matrix Factorization and proposed different nonconformity measures based on matrix factorization. Unlike MMMF which produces only bare predictions, using our proposed conformal prediction algorithm we are associating confidence to each prediction along with the guaranteed error rate. Our algorithm is tested on different data sets and we experimentally proved that NCM6 is the best nonconformity measure among the proposed nonconformity measures in achieving the prediction accuracy as good as the underlying algorithm with little uncertainty in TCP and zero uncertainty in ICP. From efficiency perspective also, it is the good measure compared to other Nonconformity measures. ICP is the better option for matrix factorization problem as it is giving good results from efficiency perspective and also the mean confidence and credibility values of ICP are higher than that of TCP when restricted to make point predictions. Furthermore, the time required for ICP is very less compared to TCP. Though our conformal prediction algorithms failed in producing large percentage of single labels at higher

confidence levels and good percentage of correct predictions due to very close possible ratings, we can use this algorithm to make single point predictions with associated confidence and credibility values. We can improve efficiency of our conformal predictors by converting all ratings to binary which tells whether the user likes or dislikes the item, instead of using the rating scale of 1-5.

5.5 Conformal Matrix Factorization with Binary Feedback

The proposed matrix factorization based conformal prediction algorithms with numerical feedback failed in producing efficient prediction regions due to very close possible ratings such as 1, 2, 3, 4 and 5. To improve the efficiency of conformal predictors we convert numerical rating data sets which contain ratings such as 1-5 or 1-6 or 1-10 into binary feedback data sets which contain only two ratings $\{$+1,-1$\}$ which represent like and dislike respectively. Binary ratings are the most natural form of the feedback which is desirable in recommender systems because users generally expect the recommender system to distinguish liked items from unliked items instead of predicting the ratings for every item. This was done using the same procedure that we explained in the previous chapter which involves calculating the average rating for every user and considering all ratings whose rating is greater than the average as *liked* and all the ratings below this average as disliked. This is the best approach to deal with all types of users including pessimistic, optimistic and strict users. This procedure also ensures that the resultant binary data sets are balanced.

5.5.1 Binary MMMF and its Nonconformity Measures

- Binary MMMF:

  This is a variant of the objective function (5.3) and is defined as

$$\mathcal{J}(W, V, \theta) = \sum_{(i,j)\in\mathcal{O}} h(Y_{ij}X_{ij})) + \frac{\lambda}{2}(||W||_F^2 + ||V||_F^2) \tag{5.14}$$

where X is the predicted matrix
$h(.)$ is hinge loss function:

$$h(z) = max(0, 1 - z) = \begin{cases} 0 & \text{if } z >= 1 \\ 1 - z & \text{if } z < 1 \end{cases} \tag{5.15}$$

- Nonconformity Measures: NCM1, NCM2, NCM5 and NCM6 in binary feedback data sets are similar to NCM1, NCM2, NCM5 and NCM6 defined for numerical feedback data sets. We define two new nonconformity measures NCM3 and NCM4 suitable to binary feedback:

  NCM3: It is defined as follows:

  $$NCM3 = -(T_e * P_e) \tag{5.16}$$

  If the true rating and predicted rating are the same, the resultant nonconformity score is -1 otherwise it is 1.

  NCM4: It is just a modified version of NCM4 and is defined as follows:

  $$NCM4 = 1 + NCM5 \tag{5.17}$$

  If the true rating and predicted rating are the same, the resultant nonconformity score is 0 otherwise it is 2.

5.5.2 Time Complexity

In this section, we compare the time complexity of proposed CP algorithms UBTCMF, IBTCMF, UBICMF and IBICMF with the underlying algorithm MMMF. The time complexity of MMMF (both UBMMMF and IBMMMF) is $z|\mathcal{O}|d$ operations which is $\Theta(z|\mathcal{O}|d)$ because optimizing equation (5.14) involves calculating the objective function and partial derivatives of the variables involved in it.

For UBTCMF and IBTCMF, we need to repeat MMMF $|t|r$ times because for every test rating we need to find the p-value of all possible rating values to find the conforming label. In order to compute p-values, we have to compute nonconformity score of all

examples which requires m operations for UBTCMF and n operations for IBTCMF. From this, the time required for UBTCMF is $|t|r(z|\mathcal{O}|(r-1)d+m)$, whereas the time required for IBTCMF is $|t|r(z|\mathcal{O}|d+N)$. But, in binary MMMF $r=2$. Therefore the time complexities of UBTCMF and IBTCMF are $\Theta(2*|t|z|\mathcal{O}|d)$. In other words, the time complexity of TCP is $2*|t|$ times the time required for MMMF. This is somewhat computationally expensive process but better than the ordinal MMMF.

For UBICMF and IBICMF, we need to execute MMMF only one time. Computing the nonconformity scores of calibration ratings and test ratings for all possible labels take $|q|$ and $|t|r$ operations. Therefore, the time required for UBICMF and IBICMF is $z|\mathcal{O}|d+|q|+|t|r$ and the time complexity is $\Theta(z|\mathcal{O}|d)$ which is the same as the underlying algorithm MMMF.

5.5.3 Experiments and Results

The experimental setup for binary feedback data sets is the same as that of numerical feedback data sets.

1. Comparative Analysis

In Table 5.5 we compare the performance in terms of percentage of correctly classified test items of UBTCMF and IBTCMF algorithms with that of MMMF with most conforming ratings (MCR) and most nonconforming ratings (MNR). Similarly, Table 5.6 compares the prediction accuracy of MMMF with that of UBICMF and IBICMF. In Tables 5.5 and 5.6 UBMMMF and IBMMMF are User-based MMMF and Item-based MMMF respectively. They are similar to the original MMMF where the test ratings of UBMMMF are same as the test ratings used in UBTCMF or UBICMF and test ratings of IBMMMF are same as the test ratings used in IBTCMF or IBICMF. We also calculate the uncertainty in the predictions as the percentage of items having more than one rating sharing the highest p-value. As the percentage of uncertainty is increased the deviation between minimum and maximum errors will also be increased as shown in Tables 5.5 and 5.6. Therefore, along with the performance comparison, we also show uncertainty involved in predictions for all NCMs for each data set in Tables 5.5c and 5.6c.

Dataset		Algorithm											
	Binary UBMMMF	Binary UBTCMF											
		NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
		MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	**0.6854**	0.6571	0.5377	0.6471	0.537	0.7549	0.6094	0.7549	0.6094	0.7549	0.6094	**0.703**	**0.6748**
Eachmovie	**0.6844**	0.6899	0.5364	0.6633	0.5539	0.7454	0.6099	0.7454	0.6099	0.7454	0.6099	**0.6915**	**0.6686**
Movielens 1M	**0.6968**	0.6872	0.5277	0.6734	0.5492	0.756	0.6289	0.756	0.6289	0.756	0.6289	**0.709**	**0.6836**
Movielens-latest-small	**0.5918**	0.6285	0.4902	0.6224	0.4998	0.6672	0.5324	0.6672	0.5324	0.6672	0.5324	**0.6165**	**0.583**

(a) UBMMMF with UBTCMF

Dataset		Algorithm											
	Binary IBMMMF	Binary IBTCMF											
		NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
		MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	**0.6671**	0.6936	0.544	0.6929	0.5824	0.7244	0.5861	0.7244	0.5861	0.7244	0.5861	**0.6755**	**0.6497**
Eachmovie	**0.6817**	0.6882	0.5211	0.6729	0.5617	0.7402	0.624	0.7402	0.624	0.7402	0.624	**0.6962**	**0.6747**
Movielens 1M	**0.7004**	0.6727	0.5445	0.6755	0.551	0.7662	0.6283	0.7662	0.6283	0.7662	0.6283	**0.7069**	**0.6872**
Movielens-latest-small	**0.6653**	0.6836	0.544	0.6628	0.5909	0.7192	0.614	0.7192	0.614	0.7192	0.614	**0.6848**	**0.6652**

(b) IBMMMF with IBTCMF

Dataset	Algorithm											
	Binary UBTCMF						Binary IBTCMF					
	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	0.1195	0.1101	0.1455	0.1455	0.1455	**0.0281**	0.1496	0.1105	0.1383	0.1383	0.1383	**0.0258**
Eachmovie	0.1535	0.1093	0.1355	0.1355	0.1355	**0.0228**	0.1671	0.1112	0.1162	0.1162	0.1162	**0.0214**
Movielens 1M	0.1595	0.1242	0.1271	0.1271	0.1271	**0.0254**	0.1282	0.1245	0.1379	0.1379	0.1379	**0.0197**
Movielens-small-latest	0.1383	0.1226	0.1349	0.1349	0.1349	**0.0336**	0.1396	0.072	0.1052	0.1052	0.1052	**0.0197**

(c) Uncertainty in predictions of UBTCMF & IBTCMF

Table 5.5: Performance comparison of MMMF with UBTCMF and IBTCMF with MCR and MNR and Uncertainty in UBTCMF & IBTCMF

It is clear from Tables 5.5 and 5.6 that the CP algorithm with NCM6 is achieving good performance compared to all other NCMs in case of TCP. The performance of CP algorithm with NCM3, NCM4 and NCM5 is same in all cases. Results of NCM1 and NCM2 are approximately similar with that of NCM3, NCM4 and NCM5 as far as the TCP is concerned. But in case of ICP, NCMs 3-6 are performing equally well with zero uncertainty.

2. Validity and Efficiency

Figure 5.7 shows validity (in terms of error percentage) of TCP and ICP for different data sets respectively (for NCM5 and NCM6). The validity results of NCM3 and NCM4 are same as that of NCM5. From Figures 5.7a and 5.7b it is clear that the prediction regions produced by NCM5 are over conservative compared to NCM6. In

Dataset		Algorithm											
	Binary UBMMMF	Binary UBICMF											
		NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
		MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	**0.6854**	0.7252	0.5744	0.7044	0.6215	**0.6798**	**0.6798**	**0.6798**	**0.6798**	**0.6798**	**0.6798**	**0.6798**	**0.6798**
Eachmovie	**0.6844**	0.7024	0.6225	0.6996	0.6296	**0.6758**	**0.6758**	**0.6758**	**0.6758**	**0.6758**	**0.6758**	**0.6758**	**0.6758**
Movielens 1M	**0.6968**	0.734	0.6011	0.7317	0.5987	0.6924	**0.6924**	**0.6924**	**0.6924**	**0.6924**	**0.6924**	**0.6924**	**0.6924**
Movielens-latest-small	**0.5918**	0.6645	0.5027	0.6275	0.5265	**0.5906**	**0.5906**	**0.5906**	**0.5906**	**0.5906**	**0.5906**	**0.5906**	**0.5906**

(a) UBMMMF with UBICMF

Dataset		Algorithm											
	Binary IBMMMF	Binary IBICMF											
		NCM1		NCM2		NCM3		NCM4		NCM5		NCM6	
		MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR	MCR	MNR
Movielens 100K	**0.6671**	0.6963	0.584	0.6829	0.6024	**0.6534**	**0.6534**	**0.6534**	**0.6534**	**0.6534**	**0.6534**	**0.6534**	**0.6534**
Eachmovie	**0.6817**	0.7161	0.5875	0.7169	0.5961	**0.6834**	**0.6834**	**0.6834**	**0.6834**	**0.6834**	**0.6834**	**0.6834**	**0.6834**
Movielens 1M	**0.7004**	0.7337	0.5835	0.7405	0.5704	**0.6921**	**0.6921**	**0.6921**	**0.6921**	**0.6921**	**0.6921**	**0.6921**	**0.6921**
Movielens-latest-small	**0.6653**	0.6839	0.5927	0.6971	0.581	**0.6543**	**0.6543**	**0.6543**	**0.6543**	**0.6543**	**0.6543**	**0.6543**	**0.6543**

(b) IBMMMF with IBICMF

Dataset	Algorithm											
	Binary UBTCMF						Binary IBTCMF					
	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	0.1508	0.0828	**0**	**0**	**0**	**0**	0.1124	0.0805	**0**	**0**	**0**	**0**
Eachmovie	0.0799	0.07	**0**	**0**	**0**	**0**	0.1286	0.1208	**0**	**0**	**0**	**0**
Movielens 1M	0.1328	0.133	**0**	**0**	**0**	**0**	0.1502	0.1701	**0**	**0**	**0**	**0**
Movielens-small-latest	0.1618	0.101	**0**	**0**	**0**	**0**	0.0912	0.1161	**0**	**0**	**0**	**0**

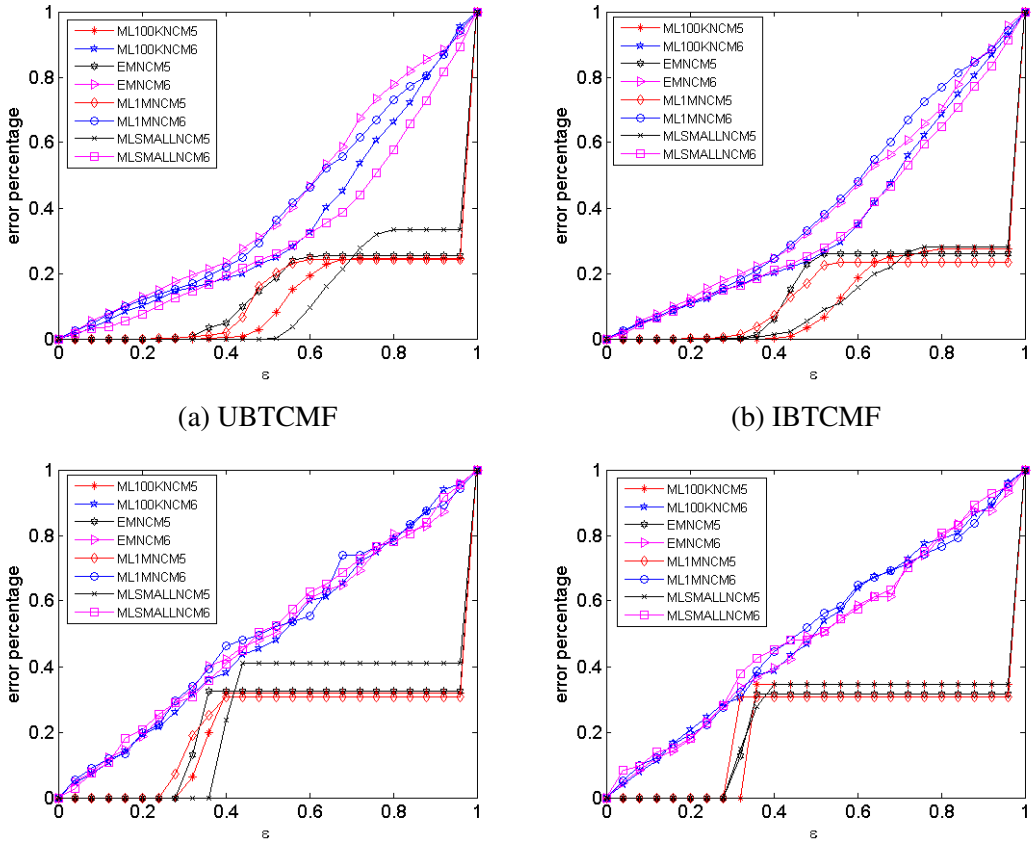
(c) Uncertainty in predictions of UBICMF & IBICMF

Table 5.6: Performance comparison of MMMF with UBICMF and IBICMF with MCR and MNR and Uncertainty in UBICMF & IBICMF

case of ICP, validity with respect to NCM6 follows a straight line as required, whereas with NCM5 it is like a step graph as shown in Figures 5.7c and 5.7d.

Like in numerical data sets, the efficiency of prediction regions produced by the CP algorithm for binary feedback is measured using percentage of single labels, percentage of correct predictions among the percentage of single labels, percentage of multiple labels and percentage of empty labels.

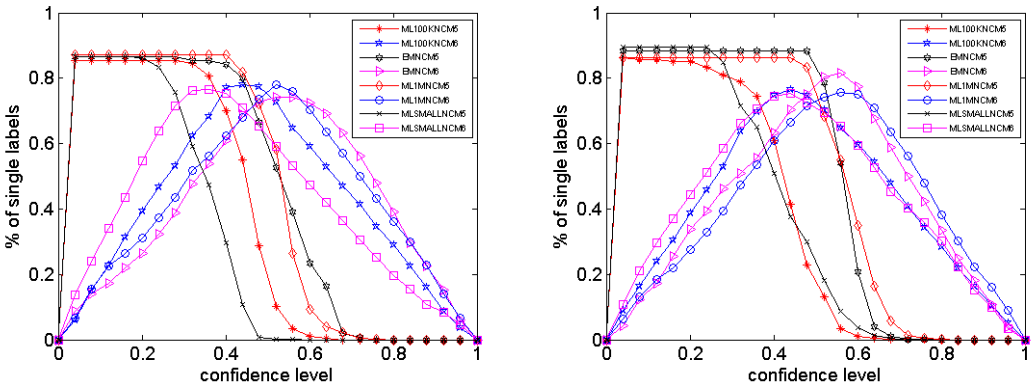
Figure 5.8 shows the efficiency of the prediction regions in terms of percentage of single labels produced by TCP and ICP. As can be seen from the Figure 5.8, NCM6 is the best nonconformity measure compared to NCM5, as it produces more number of single labels compared to NCM5 especially at high confidence levels. Like in numerical data sets, ICP is preferable to TCP, because it produces large percentage of single labels than TCP as we go from lower to high confidence levels.



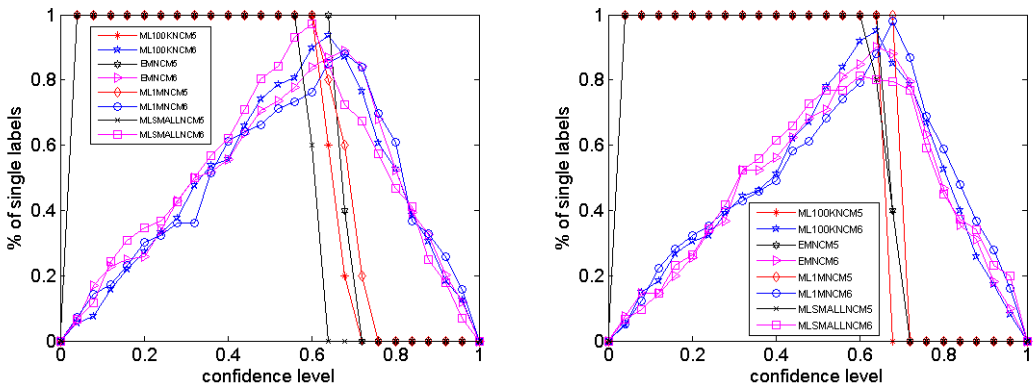
(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.7: Validity (in terms of error percentage) of UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets



(a) UBTCMF (b) IBTCMF



(c) UBICMF (d) IBICMF

Figure 5.8: %of single labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

Figure 5.9 shows the percentage of correct predictions produced by TCP and ICP with NCM6. From the Figure 5.9 we can observe that, the percentage of correct predictions produced by TCP and ICP at all confidence levels is greater than or equal to 60%.

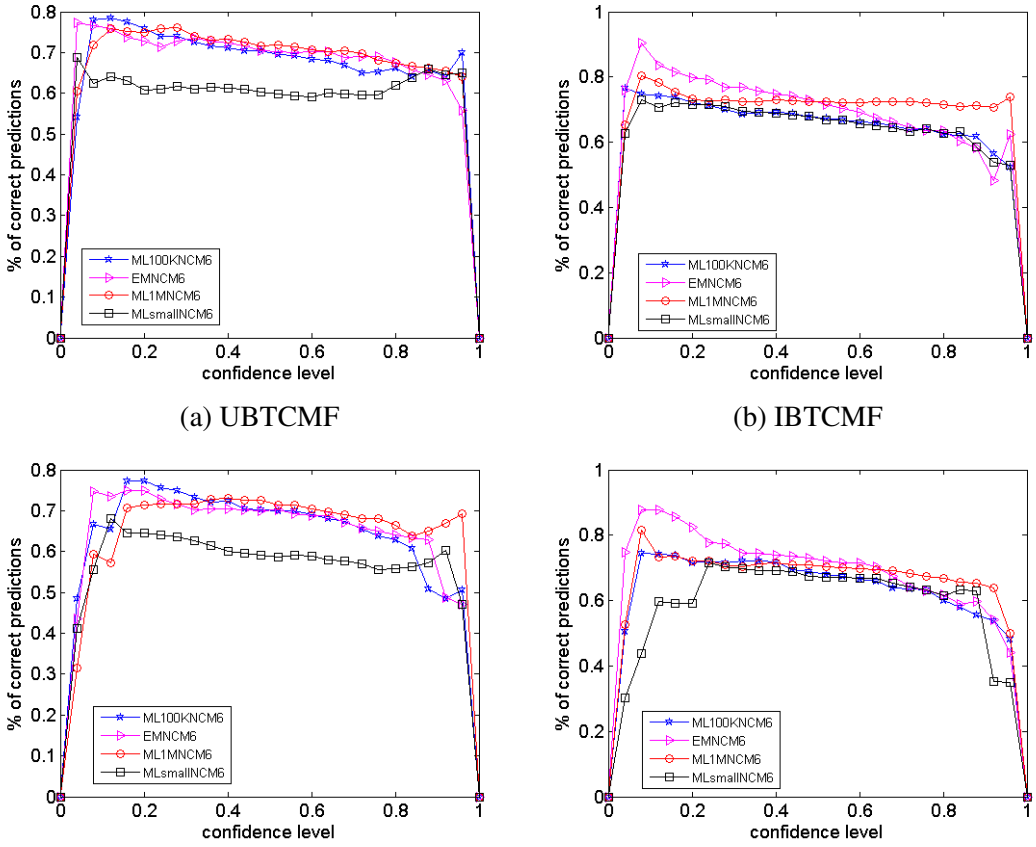
Figure 5.10 shows the percentage of multiple labels produced by TCP and ICP with NCM5 and NCM6. From the Figure 5.10, it is clear that NCM6 produces less number of multiple labels compared to NCM5 at all confidence levels. Therefore, here also, NCM6 is more preferable compared to NCM5.

Figure 5.11 shows the percentage of empty labels produced by TCP and ICP with NCM5 and NCM6. In contrast to Figures 5.8 and 5.10 where NCM6 outperforms NCM5, in Figure 5.11, NCM5 is showing better performance than NCM6 as the number of empty labels produced by NCM5 is zero at all confidence levels.

From these figures it is clear that NCM6 is performing better than NCM5. The results of NCM3 and NCM4 are the same as that of NCM5. Therefore we are not showing their results separately.

3. Confidence and Credibility

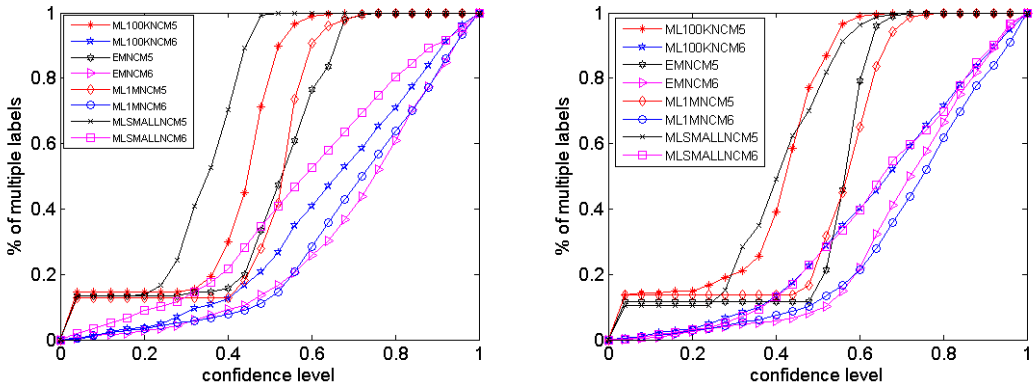
Tables 5.7 and 5.8 give mean confidence and mean credibility values along with mean difference between highest p-value(p1) and the second highest p-value(p2) for TCP and ICP respectively. Here also, NCM6 is outperforming all other NCMs in



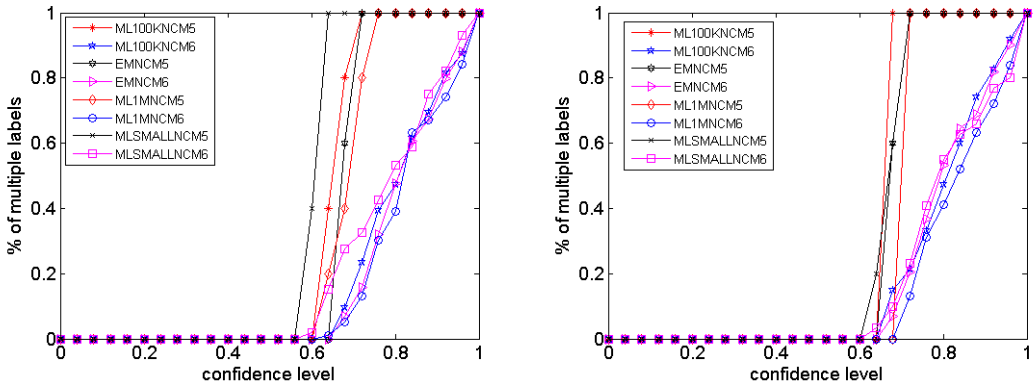
(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.9: %of correct predictions made by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets



(a) UBTCMF (b) IBTCMF



(c) UBICMF (d) IBICMF

Figure 5.10: %of multiple labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

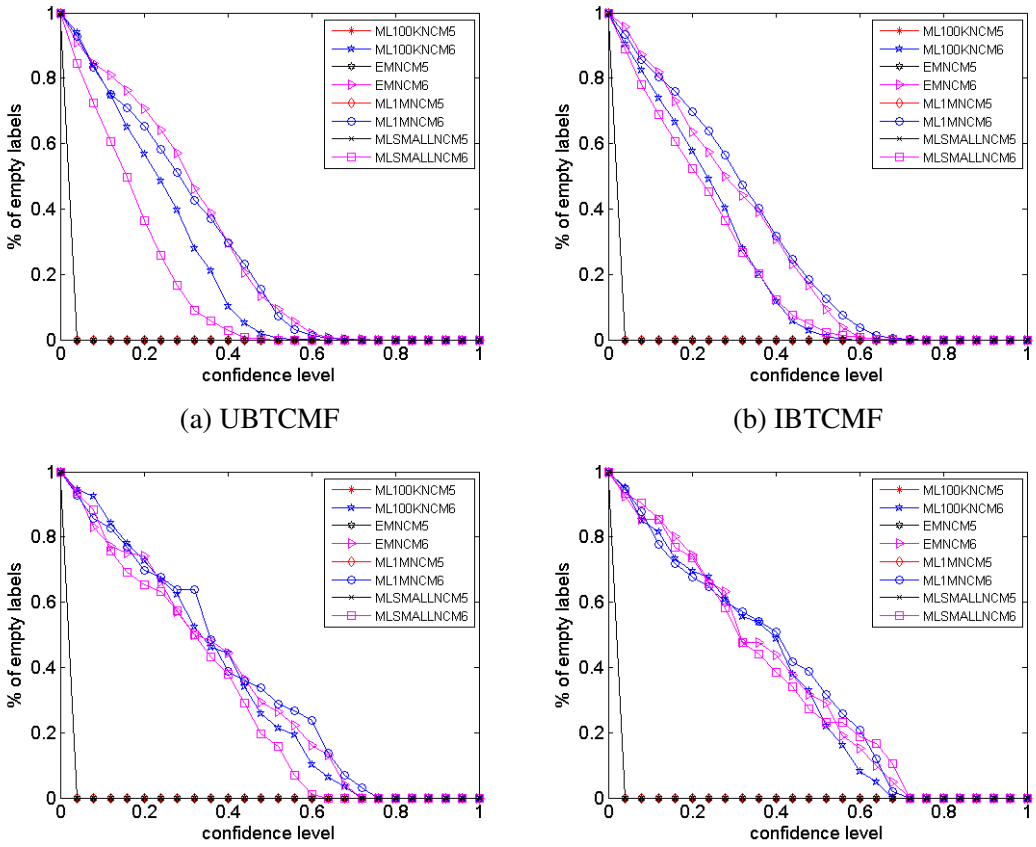
producing high confidence values. Furthermore, compared to the confidence values produced by CP algorithm for numerical feedback, CP algorithm for binary feedback produces better confidence values for both TCP and ICP.

Dataset	Performance Measure	Algorithm											
		Binary UBICMF						Binary IBTCMF					
		NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	mean confidence	0.3763	0.5474	0.3878	0.3878	0.3878	**0.6398**	0.3835	0.5616	0.3701	0.3701	0.3701	**0.6405**
	mean credibility	0.6927	0.5403	1	1	1	0.7725	0.6837	0.5153	1	1	1	0.7722
	mean p1-p2	0.0678	0.0866	0.3864	0.3864	0.3864	0.412	0.0657	0.0758	0.3687	0.3687	0.3687	0.4125
Eachmovie	mean confidence	0.3418	0.5772	0.4698	0.4698	0.4698	**0.7051**	0.341	0.5193	0.5015	0.5015	0.5015	**0.7033**
	mean credibility	0.725	0.505	1	1	1	0.7078	0.7093	0.5437	1	1	1	0.7139
	mean p1-p2	0.0653	0.0812	0.4685	0.4685	0.4685	0.4127	0.0487	0.0619	0.5003	0.5003	0.5003	0.417
Movielens 1M	mean confidence	0.3434	0.5547	0.4657	0.4657	0.4657	**0.6956**	0.3429	0.5594	0.4986	0.4986	0.4986	**0.7162**
	mean credibility	0.7191	0.5212	1	1	1	0.7215	0.7242	0.5224	1	1	1	0.6993
	mean p1-p2	0.0609	0.0747	0.4644	0.4644	0.4644	0.4168	0.0659	0.0805	0.4972	0.4972	0.4972	0.4153
Movielens-lateset-small	mean confidence	0.3524	0.4905	0.3116	0.3116	0.3116	**0.5706**	0.3925	0.5443	0.3791	0.3791	0.3791	**0.6428**
	mean credibility	0.7063	0.5868	1	1	1	0.839	0.6748	0.5391	1	1	1	0.7818
	mean p1-p2	0.0572	0.0761	0.3102	0.3102	0.3102	0.4093	0.0659	0.0827	0.378	0.378	0.378	0.4244

Table 5.7: Mean confidence and Mean credibility of UBTCMF and IBTCMF

We did not show the validity and efficiency of CP with NCMs 1 to 2, as the efficiency of CP with these NCMs is very less compared to other NCMs, although their validity property is satisfied.

From Figures 5.7 to 5.11, it is clear that NCM6 is preferable to NCM5. Because, the prediction regions produced by NCM5 are more conservative compared to NCM6. Furthermore, though NCM5 outperforms NCM6 by producing 0% of empty labels at all confidence levels, its performance in producing the percentage of multiple labels



(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.11: %of empty labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

Dataset	Performance Measure	Algorithm											
		Binary UBICMF						Binary IBTCMF					
		NCM1	NCM2	NCM3	NCM4	NCM5	NCM6	NCM1	NCM2	NCM3	NCM4	NCM5	NCM6
Movielens 100K	mean confidence	0.5183	0.548	0.663	0.663	0.663	0.8126	0.511	0.5418	0.6537	0.6537	0.6537	**0.8097**
	mean credibility	0.5543	0.5279	1	1	1	0.656	0.5498	0.5244	1	1	1	0.656
	mean p1-p2	0.0711	0.0751	0.663	0.663	0.663	0.4686	0.0597	0.0654	0.6537	0.6537	0.6537	0.4657
Eachmovie	mean confidence	0.5154	0.5522	0.6766	0.6766	0.6766	0.821	0.5038	0.5249	0.6831	0.6831	0.6831	**0.8109**
	mean credibility	0.5566	0.5375	1	1	1	0.6467	0.5608	0.5336	1	1	1	0.6526
	mean p1-p2	0.0754	0.0851	0.6766	0.6766	0.6766	0.4736	0.0591	0.0611	0.6831	0.6831	0.6831	0.4576
Movielens 1M	mean confidence	0.5225	0.546	0.6866	0.6866	0.6866	0.8307	0.5208	0.5546	0.6933	0.6933	0.6933	**0.8368**
	mean credibility	0.5371	0.5124	1	1	1	0.6361	0.5408	0.5149	1	1	1	0.6385
	mean p1-p2	0.0583	0.0571	0.6866	0.6866	0.6866	0.4668	0.0601	0.0678	0.6933	0.6933	0.6933	0.4753
Movielens-lateset-small	mean confidence	0.5068	0.5389	0.5984	0.5984	0.5984	0.7891	0.513	0.5445	0.6658	0.6658	0.6658	**0.8151**
	mean credibility	0.5561	0.5349	1	1	1	0.6994	0.5398	0.5211	1	1	1	0.6513
	mean p1-p2	0.0613	0.0728	0.5984	0.5984	0.5984	0.4885	0.0518	0.0644	0.6658	0.6658	0.6658	0.4663

Table 5.8: Mean confidence and Mean credibility of UBICMF and IBICMF

and single labels is not better than that of NCM6 for both TCP and ICP. In the same way, the confidence values produced by NCM6 are better than NCM5. Between TCP and ICP, ICP outperforms TCP in producing the efficient and more confident prediction regions.

5.6 Conformal Matrix Factorization with Ordinal vs Binary Feedback Data

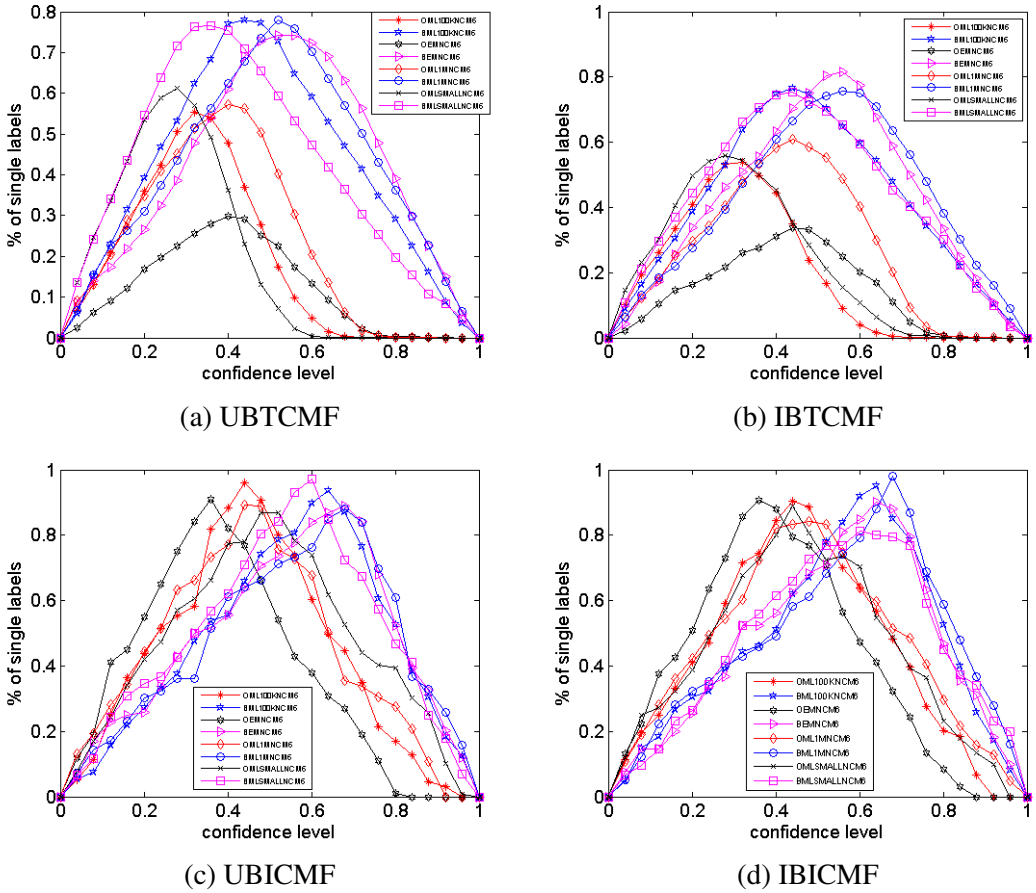
As we discussed above, due to close possible rating values like 1, 2, 3, 4 and 5 in numerical data sets, the proposed conformal prediction algorithms were not able to produce efficient prediction regions. In order to improve the efficiency of the prediction regions we converted numerical feedback to binary feedback. In this section we compare the efficiency of proposed conformal prediction algorithms for both numerical and binary feedback data sets and analyze the results.

Figure 5.12 shows the comparison of percentage of single labels produced by the proposed CP algorithms for both numerical and binary data sets. As can be seen from the figure, CP with binary feedback is producing large percentage of single labels compared to CP with numerical feedback.

Figure 5.13 compares the percentage of correct predictions among the percentage of single labels produced by CP algorithm with numerical and binary feedback. From the figure, we can observe that CP algorithm with binary feedback is making more number of correct predictions compared to CP with numerical feedback. For numerical feedback this percentage is around 60%, whereas for binary feedback it is around 80%.

Figure 5.14 compares the percentage of multiple labels produced by CP algorithm with numerical and binary feedback. From the figure, it is clear that CP algorithm with binary feedback is producing less number of multiple labels compared to CP with numerical feedback.

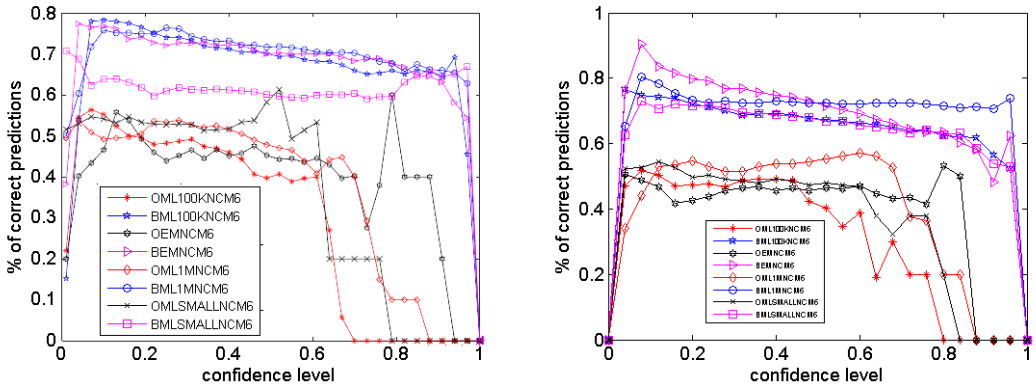
Figure 5.15 compares the percentage of empty labels produced by CP algorithm with numerical and binary feedback. From the figure, we can observe that CP with numerical feedback is producing less number of empty labels compared to CP with numerical feedback in contrast to Figures 5.12 to 5.14, where CP with binary feedback is outperforming CP with numerical feedback.



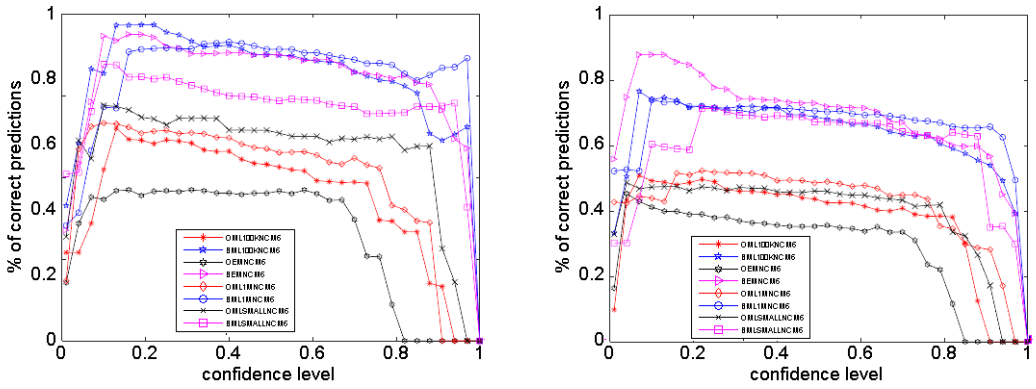
(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.12: %of single labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets



(a) UBTCMF (b) IBTCMF



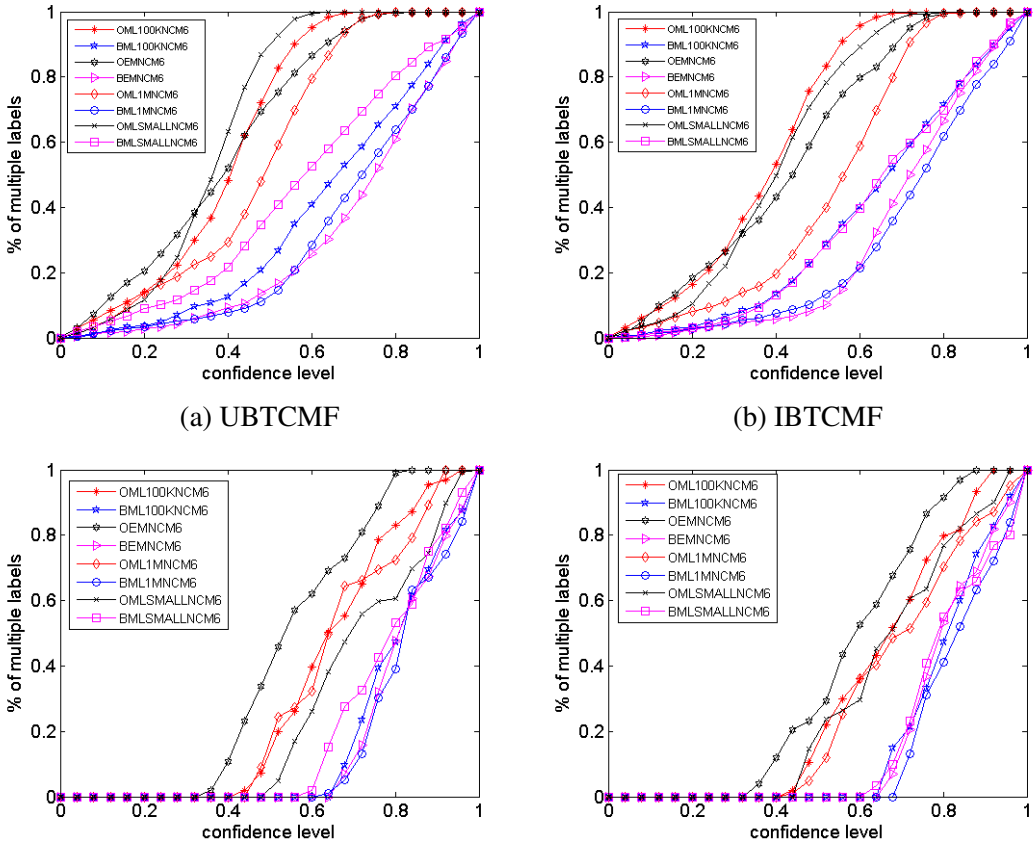
(c) UBICMF (d) IBICMF

Figure 5.13: %of correct predictions made by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

From these figures it is clear that CP algorithms for binary feedback data sets are outperforming their counterparts in numerical data sets. Only in the case of percentage of empty labels, numerical data sets are showing slightly better performance compared to binary data sets.

5.7 Summary

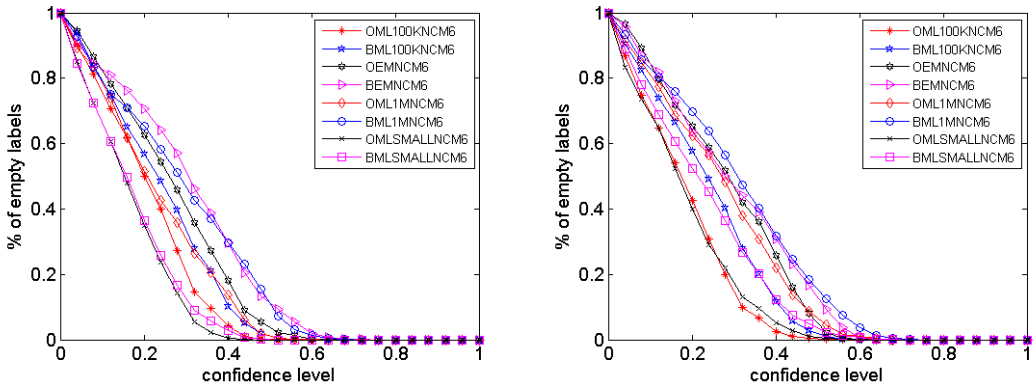
The proposed conformal prediction algorithms failed in producing large percentage of single labels and good percentage of correct predictions at higher confidence levels for numerical feedback data sets due to very close possible ratings. To improve the efficiency of the proposed conformal prediction algorithms, we first converted the numeric feedback data to binary, and the algorithms are applied on the resultant data sets. Similar to numerical feedback data we proposed six nonconformity measures suitable to binary data and compared the accuracy of the CP algorithm with the proposed NCMs with the binary MMMF. Like in numeric feedback data, here also, NCM6 shows better accuracy with less uncertainty compared to all other NCMs in case of TCP. But for ICP, NCMs 3-6 are performing equally well with zero certainty. Then we compared the validity and efficiency of NCMs 3-5 with NCM6. As the performance of all three NCMs i.e., NCM3, NCM4 and NCM6 is same for validity and efficiency we have shown only NCM5 result when comparing the validity and efficiency with NCM6. Though validity



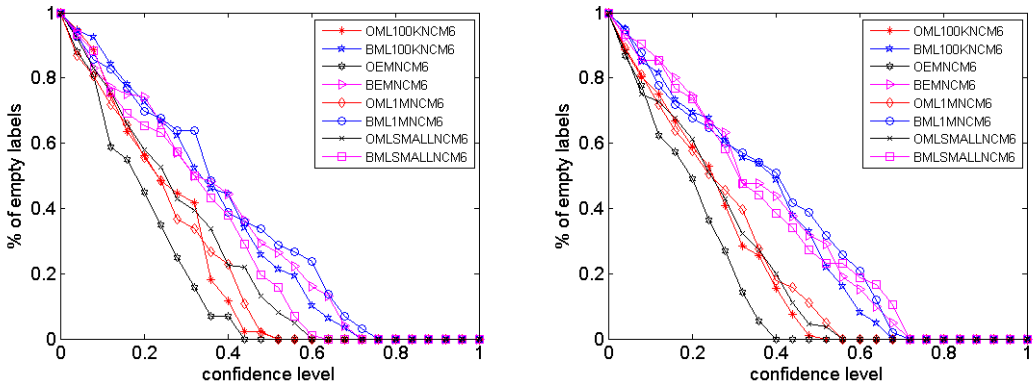
(a) UBTCMF (b) IBTCMF

(c) UBICMF (d) IBICMF

Figure 5.14: %of multiple labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets



(a) UBTCMF (b) IBTCMF



(c) UBICMF (d) IBICMF

Figure 5.15: %of empty labels produced by UBTCMF & IBTCMF and UBICMF & IBICMF for different data sets

is satisfied by both NCMS, NCM6 is producing more efficient predictions compared to NCM5. To check whether we improved the efficiency of the prediction regions with binary data over numeric data we compared the efficiency of prediction regions of both data sets. From the experiments, we show that CP algorithms for binary feedback data sets are outperforming their performance in numerical data sets. Only in case of percentage of empty labels numerical data sets are showing slightly better performance.

Chapter 6

Recommendation with Confidence in Neighbourhood-based Algorithms

In the previous two chapters, we have shown the application of conformal prediction to prediction algorithms in recommender systems. When applied for prediction task, conformal prediction attaches confidence values to each of the predictions made by the algorithm. Though conformal prediction algorithm is designed to associate confidence values to individual predictions, in this work, we use conformal prediction to associate confidence values to a set of recommended items L produced by neighborhood-based algorithms. Attaching the confidence values to a set of recommended items give the relevance of these items to the user tastes. Unlike the confidence estimation methods in recommender systems (all are limited to assigning the confidence values to predictions) which do not provide any guarantee on the error rate, conformal prediction makes it possible by allowing the users to select the appropriate significance level depending on the application. In recommendation scenario, error rate refers to the probability of excluding the relevant items form the set of recommended items at the given significance level.

As discussed above, the regions produced by any CP algorithm are automatically valid. The recommendation region is said to be valid if the error rate is bounded by the chosen significance level. But efficiency in terms of minimizing the number of irrelevant items in the recommendation regions depends on the nonconformity measure used by the CP algorithm [53]. In conformal prediction, nonconformity measure tells how strange the new item is with respect to the set of items in the training set. Instead

of using the nonconformity measure, it is appropriate to use conformity measure as our goal is to recommend the items amongst all the unused items that can conform to the training set. But in practice it is more common to use nonconformity instead of conformity. For example, when we compare a new example with an average of old examples we will usually first define a distance between the two rather than devising a way to measure their closeness. Doing so is consistent with the tradition in mathematical statistics and machine learning. In statistics, test statistics are usually defined so as to measure discrepancy rather than agreement and in machine learning, distance measures are used to find the nearest neighbours in K-NN instead of using proximity measures [75]. In our work, we used a simple conformity measure which is the score of the item. Major contributions of this chapter are:

- We describe application of conformal prediction (TCP) to recommendation task to provide confidence values to the set of recommended items produced by neighborhood-based collaborative filtering algorithms.
- We adopted false negative rate and false positive rate from statistics and machine learning to define validity and efficiency of proposed conformal recommendation algorithms.
- We empirically demonstrate validity and efficiency of our conformal recommendation algorithms.
- We experimentally show that though at 60-90% confidence levels the algorithms are producing low precision values, the recall values are sufficiently high while producing less number of recommended items (NRI) as compared to the total number of items in the data sets.
- We experimentally show that the performance of user-based conformal recommendation is better than that of item-based conformal recommendation.

In this chapter, we first discuss why we chose unary feedback for TOP-N recommendation task. Then, we present neighbourhood-based recommendation algorithms. Next, we describe application of conformal prediction to recommendation task to provide confidence values to the set of recommended items produced by the

neighbourhood-based algorithms. We also show how to adopt false negative rate and false positive rate from statistics and machine learning to define validity and efficiency of proposed conformal recommendation algorithms. Finally, we demonstrate the validity and efficiency of our conformal recommendation algorithms.

6.1 Motivation to Use Unary Feedback for Top-N Recommendation Task

For recommendation task, we use unary data (1 or 0) instead of using ratings (1-5 or 1-10) or binary data (+1 or -1). The reasons are

- [26, 48] considered user's rating as the noisy evidence of user's true rating. Users give different ratings at different times when asked to rate the same movie. Therefore, identifying k most similar items which are rated as 1 when the new item rating is assumed as 1 does not make sense. Moreover, the given ratings are not accurate enough to produce the correct predictions or recommendations.

- Users in general do not rate every item that they have used. Therefore, some of the ratings are missing in rating data sets [78].

- Users do not update their ratings. For example, assume that a customer bought a product from some e-commerce website. Initially, he is satisfied with that product as it works well for some time. Hence, he rated it 5. After a few days, he found some problem with that product and as a result he might think not to purchase that product or not to purchase the items form that e-commerce site in future, but might forget to update that rating. Other users still think that it is a top rated product [78]. One more reason is that the user who liked the item in the past may not like it in future. The ratings given by the user in the past might not reflect his/her present taste.

- When CP is applied on rating data sets, we need to find k neighbors belonging to each rating to compute the nonconformity score of each unused item. But the distribution of ratings in these data sets is uneven. For example, more than 80% of all ratings in MovieLens 100K are greater than 2 and nearly 70% of

all ratings in Eachmovie are greater than 3 [39]. As a result, it becomes very difficult to identify k most similar items for each rating. In case of User-based collaborative filtering, even with binary ratings (+1 or -1), it is very difficult to find the required number of neighbors belonging to both positive and negative ratings for each item in the training set and test item.

- In this work, our focus is not on predicting a rating for the item but on producing a fixed length of recommended items and the confidence of this recommendation list. Therefore, rating prediction is not mandatory for this task. We predict a score (instead of rating) for each unused item which tells the likelihood of this item being recommended.

6.2 Neighborhood-based CF Algorithms for Top-N Recommendation

Algorithms 5 and 6 describes User-based and Item-based collaborative filtering algorithms for recommending a fixed length of items which we call them as User-based Recommendation (UBR) and Item-based Recommendation (IBR) algorithms. There are a number of different ways to compute the similarity in neighborhood-based recommendation algorithms. Among them the most commonly used are Pearson, Cosine, Jaccard and Dice. In our work, we have chosen cosine similarity measure defined in Equation (3.11) to compute the similarities between users in UBCF and Equation (3.26) between items in IBCF as it is a very popular measure for *top-N* recommendation task. Both equations will give the value between [0,1]. Two users u and v are considered to be similar (i.e.,$similarity(u, v)$ = 1) when these two users rated the same set of items. Similarly, two items i and j are considered to be similar (i.e.,$similarity(i, j)$ = 1) when these two items are rated by same set of users.

Algorithm 5: : User-based Recommendation Algorithm

Input: U,I,k,u_t,C_t
Output: L
for all $u_i \in U \setminus u_t$ **do**
  Compute the similarity between u_i and u_t using Equation (3.11)
end for
for all $i \in I \setminus C_t$ **do**
  Find the set of users S who consumed item i
  Compute the score of i by adding the similarity scores of k most similar users from S.
end for
Recommend a set of Top-N items L with highest scores.

Algorithm 6: : Item-based Recommendation Algorithm

Input: I,k,C_t
Output: L
for all $i \in I$ **do**
  for all $j \in I$ **do**
    Compute the similarity between i and j using Equation (3.26)
  end for
end for
for all $i \in I \setminus C_t$ **do**
  Find k most similar items from C_t
  Compute the score of i by adding the similarity scores of these k items.
end for
Recommend a set of Top-N items L with highest scores.

6.3 Neighborhood-based Conformal Recommendation Algorithms

In this section, we discuss how to build Conformal Prediction on top of User-based and Item-based Recommendation Algorithms. We first discuss the algorithm setup and then the conformity measures for both algorithms. Finally, we present our proposed algorithms, User-based conformal recommendation (UBCR) and Item-based Conformal Recommendation (IBCR) along with the measures of validity and efficiency that are suitable for the recommendation task.

6.3.1 Problem Formulation

Conformal prediction is originally designed for prediction task where we have a training set of examples, a set of possible labels and a test example for which a set of labels (region prediction) or a single label (point prediction) is assigned along with a confidence value. In our work, we are adapting conformal prediction to recommendation task where the algorithm recommends a fixed number of items to the user instead of predicting a rating for an item. In order to make recommendations, first we need to select the neighbors (users in UBCR and items in IBCR) for every target item i_t of the target user u_t. This can be done based on the set of items consumed by the target user u_t which we denote with C_t. In other words, neighbors for i_t are selected based on the past history of the target user u_t. Therefore, the training set for user u_t is C_t i.e., each item in C_t constitutes an example in the training set. This means that the training set is different for different users and as a result we need to apply conformal prediction process for every user separately. Test examples are the unused items of the target user u_t which is $I \setminus C_t$. As ours is not the prediction task, we do not have labels for the test example. Instead every test example is a candidate item for recommendation. Our task is to find those test examples, in our case unused items, which conforms to the set of items in the training set and recommend them to the user with an associated confidence measure.

6.3.2 Conformity Measure (CM)

In this work, we have used a simple conformity measure which is the score of the item. As discussed above, score of the target item i_t for the target user u_t is the sum of similarities of k most similar users who have used this item in UBCR and sum of the similarities of k most similar items that are consumed by u_t in IBCR. Formally, this can be expressed as follows:

- UBCR: Let the k most similar users for u_t for i_t are $\{u_{t_1}, u_{t_2}, ..., u_{t_k}\}$ and their corresponding similarity scores with u_t are $\{s_{u_{t_1}}, s_{u_{t_2}}, ..., s_{u_{t_k}}\}$. Then conformity measure (CM) for the item i_t is calculated as

$$CM_{UBCR} = \sum_{j=1}^{k} s_{u_{t_j}} \tag{6.1}$$

where CM_{UBCR} is the conformity measure for User-based conformal recommendation.

- IBCR: Let the k most similar items for i_t are $\{i_{t_1}, i_{t_2}, ..., i_{t_k}\}$ and their corresponding similarity scores with i_t are $\{s_{i_{t_1}}, s_{i_{t_2}}, ..., s_{i_{t_k}}\}$. Then conformity measure (CM) for the item i_t is calculated as

$$CM_{IBCR} = \sum_{j=1}^{k} s_{i_{t_j}} \tag{6.2}$$

where CM_{IBCR} is the conformity measure for Item-based conformal recommendation.

6.3.3 Proposed Algorithms

In this subsection, we discuss how conformal prediction can be applied to User-based recommendation and Item-based recommendation algorithms. In order to find whether the new item conforms to the training set C_t, we need to append the item i to C_t to form extended training set E. Then, we compute the conformity scores of each item in E in order to find p-value for item i. As mentioned in Chapter 4, we can compute conformity scores in two ways:

- By excluding the item for which we are computing the conformity score from E.
- By including the item for which we are computing the conformity score from E.

First option is time consuming as it requires modification of similarity values of user in UBCR or item in IBCR $|C_t|+1$ number of times for each rating to be predicted. This is because in UBCR, in order to compute conformity score for j^{th} item, first we need to remove that item from E and then update the similarity values of target user u_t with other users. Similarly, to compute conformity score for the j^{th} item, we

have to remove that item from E and update the similarity values of the j^{th} item with other items in IBCR. Whereas in the second approach, we need to update similarity values of u_t with other users only once after appending the new item i to C_t in UBCR. Similarly, in IBCR, we update the similarity values of i with other items only once after appending the new item i to C_t . That means, in the first approach, we need to update for every item in E whereas in the second approach, we need to update the similarity values only once that is after appending i to C_t. Therefore, we have chosen the second option in our work.

- User-based Conformal Recommendation Algorithm:

  Algorithm 7 describes the application of CP to UBR in detail. First, compute the similarity between the target user u_t and all other users. For every unused item i in $I \setminus C_t$ of the target user u_t, append this item to the set of consumed items C_t to form the appended training set E. Then update the similarity values of u_t using the training set as E. For every item in E compute the conformity score by adding the similarity scores of k most similar users from $U \setminus u_t$ who consumed that item. These conformity scores are then used to calculate the p-value for i which gives the typicalness of this sequence E. Finally, output all the items in $I \setminus C_t$ whose p-values are $> \epsilon$ in order to produce the recommendation region at the given confidence level $1 - \epsilon$ or select top N items with highest p-values in order to produce a *Top-N* recommendation list L with confidence 1 - $p(N + 1)$.

- Item-based Conformal Recommendation Algorithm:

  Algorithm 8 describes the application of CP to IBR in detail. First, compute the similarity between all items. Unlike in User-based conformal recommendation where the conformity scores of items in the training set are computed after appending every unused item to the training set, conformity scores of items in the training set are precomputed in Item-based conformal recommendation. For every unused item i in $I \setminus C_t$ of the target user u_t, append this item to the set of consumed items C_t to form the appended training set E. Next, update the similarity values of i with other items in C_t. Compute the conformity score of i by adding the similarity scores of k most similar items from C_t. Then, update the conformity scores of those items in the training set for which i is in their

Algorithm 7: : User-based Conformal Recommendation Algorithm

Input: U,I,k,u_t,C_t,ϵ
Output: Recommendation Region, L
for all $u_i \in U \setminus u_t$ **do**
  Compute the similarity between u_i and u_t using Equation (3.11)
end for
for all $i \in I \setminus C_t$ **do**
  $E = C_t \cup i$
  Update the similarity scores of u_t after appending i to C_t
  for all $j \in E$ **do**
    Find the set of users S from $U \setminus u_t$ who consumed item j
    Compute the conformity score of j using Equation (6.1)
  end for
  Compute the p-value *p(i)* using Equation (4.3)
end for
Recommendation Region = $\{i|p(i) > \epsilon\}$ with confidence 1-ϵ
OR
Recommend a set of *Top-N* items L with highest p-values with confidence 1 - $p(N+1)$

neighborhood. These conformity scores are then used to calculate the p-value for i which gives the typicalness of this sequence E. Finally, output all the items in $I \setminus C_t$ whose p-values are $> \epsilon$ in order to produce the recommendation region at the given confidence level $1 - \epsilon$ or select top N items with highest p-values in order to produce a *Top-N* recommendation list L with confidence 1 - $p(N+1)$.

6.3.4 Validity and Efficiency for Recommendation Task

Validity and Efficiency are used to measure the quality of prediction regions produced by CP. Error refers to the probability of excluding the true label from the prediction region and Validity refers to probability of including true label in the prediction region. In other words, validity = 1 - Error. Efficiency is the tightness of the prediction regions. The narrower the prediction region the more efficient the conformal predictor is. Efficiency can be measured in different ways for prediction task. Some of them are listed below:

- Observed Excess criterion which gives the the average number of false labels

Algorithm 8: : Item-based Conformal Recommendation Algorithm

Input: I,k,C_t,ϵ
Output: Recommendation Region, L
for all $i \in I$ **do**
  for all $j \in I$ **do**
    Compute the similarity between i and j using Equation (3.26)
  end for
end for
for all $i \in C_t$ **do**
  Find k most similar items from $C_t \setminus i$
  Compute the conformity score of i using Equation (6.2)
end for
for all $i \in I \setminus C_t$ **do**
  $E = C_t \cup i$
  Update the similarity scores of i after appending i to C_t
  Find k most similar items from C_t
  Compute the conformity score of i using Equation (6.2)
  for all $j \in C_t$ **do**
    if $similarity(i, j) > similarity(j, k)$ **then**
      Recompute the conformity score of item j
    end if
  end for
  Compute the p-value *p(i)* using Equation (4.3)
end for
Recommendation Region = $\{i|p(i) > \epsilon\}$ with confidence 1-ϵ
OR
Recommend a set of *Top-N* items L with highest p-values with confidence 1 - $p(N + 1)$

(ANFL) included in the prediction region at significance level ϵ.

- Percentage of test examples having prediction regions with single label.
- Percentage of test examples having prediction regions with more than one label.
- Percentage of test examples having empty prediction region.

In this Chapter, our focus is on recommendation task instead of prediction task. Therefore, we need to find suitable metrics for recommendation task. We adopted false negative rate (FNR) or miss rate and false positive rate (FPR) or fall out from

statistics and machine learning to define validity and efficiency of the recommendation regions respectively in recommendation task. Here relevant items are the items which are already used by the user (test set items denoted with S_t).

- Error Percentage (False Negative Rate) gives the probability that a relevant item is not present in the recommendation region at the given significance level ϵ. In other words, how many of the relevant items are missing in the recommendation region. Small values are preferable.

$$ErrorPercentage = \frac{|Relevant \setminus Recommended|}{|Relevant|} \tag{6.3}$$

  where $Recommended$ is the recommendation region produced at a given significance level ϵ and $relevant$ is the set of test items S_t.

  Validity in terms of error percentage is given by

$$Validity = 1 - ErrorPercentage \tag{6.4}$$

- Efficiency (False Positive Rate) which is also a variant of Observed Excess criterion gives the probability that an irrelevant item is recommended at the given ϵ. In other words, how many of the recommended items are not relevant. Here also, small values are preferable.

$$Efficiency = \frac{|Recommended \setminus Relevant|}{|Irrelevant|} \tag{6.5}$$

6.3.5 Time Complexity

In this section, we compare the time complexity of the proposed CP algorithms (UBCR & IBCR) with their respective underlying algorithms (UBR & IBR). Let m be the number of users and n be the number of items.

- Time complexity comparison between UBR and UBCR:

  The time complexity of UBR can be divided into two parts: (i) time required to compute the similarities between target user and all other users (similarity

computation) and (ii) time required to predict scores for all candidate items and make $Top-N$ recommendations (score prediction and recommendation).

- Similarity computation: For every target user u_t, we need to compute similarities between target user u_t and all other users. This requires $(m-1)$ computations and each requires n operations. If the number of target users are n_t, then the time complexity to compute the similarities for n_t target users is $n_t(m-1)n$ which is $\Theta(n_tmn)$.
- Score prediction: To reduce the complexity, we precomputed the users of all items which requires $\Theta(nm)$ time. The number of candidate recommended items for u_t is $n-|C_t|$ and for every candidate recommended item of the target user u_t, score computation takes $\Theta(1)$. For n_t target users, time complexity for this step is $\Theta(nm+n_t(n-|C_t|))$.
- Recommendation: Finally, recommending the $Top-N$ items with highest scores require $(n-|C_t|)N$ operations. Therefore, the time complexity for this step for n_t target users is $\Theta(n_t(n-|C_t|)N))$

The time complexity for UBR is $\Theta((n_tmn)+(nm+n_t((n-|C_t|)+(n-|C_t|)N)))$.

For UBCR, the time required for similarity computation is same as that of IBCF which is $\Theta(n_tmn)$. Time complexity for score prediction and recommendation is computed as follows:

- Compute the conformity scores and Score prediction: Similar to UBR, here also, we precomputed the users of all items which requires $\Theta(nm)$ time. The number of candidate recommended items for u_t is $n-|C_t|$. For every candidate recommended item i of the target user u_t, we need to update the similarity scores of u_t with other users after appending i to C_t. This requires $\Theta(mn)$ operations. Computing the conformity scores of all items in the set $C_t \cap i$ takes $|C_t|+1$ operations and computing the p-value for each candidate recommended item i takes another $|C_t|$ operations. So, the time complexity of this step for n_t target users is $\Theta(nm+n_t((n-|C_t|)(mn)))$

– Recommendation: Recommending the $Top-N$ items with highest scores require $(n-|C_t|)N$ operations and for n_t target users, the time complexity is $\Theta(n_t(n-|C_t|)N))$.

The time complexity of UBCR is $\Theta(n_t mn + (nm + n_t((n-|C_t|)(mn) + ((n-|C_t|)N)))$. Generally, the time complexity for the TCP is calculated as the number of test examples times the number of labels times the underlying algorithm time complexity. Here, we do not have labels and therefore, we are repeating the underlying UBR $(n-|C_t|)$ times for each target user. Although this is computationally expensive process, it is still practical to implement even for large data sets like Eachmovie and Movielens 1M.

- Time complexity comparison between IBR and IBCR:

  Similar to UBR, the time complexity of IBR can be divided into two parts: (i) time required to compute the similarities between all items (similarity computation) (ii) time required to predict scores for all candidate items and make $top-N$ recommendations (score prediction and recommendation).

  – Similarity computation: We need to compute $n(n-1)$ similarities to compute similarity between all n items. Each similarity computation requires m operations. Therefore, the time complexity for similarity computation is $n(n-1)m$ which is $\Theta(n^2m)$.

  – Score prediction: For every candidate recommended item of the target user u_t, we need to find k similar items from $|C_t|$ which requires $|C_t|k$ time and score computation takes $\Theta(1)$ time. Therefore, the time complexity for this step for n_t target users is $\Theta(n_t((n-|C_t|)|C_t|k))$.

  – Recommendation: Finally, recommending the $Top-N$ items with highest scores require $(n-|C_t|)N$ operations which for n_t target users is $\Theta(n_t(n-|C_t|)N)$

  The time complexity for IBR is $\Theta(n^2m + n_t((n-|C_t|)|C_t|k + (n-|C_t|)N))$.

  For IBCR, the time required for similarity computation is the same as that of IBR which is $\Theta(n^2m)$. Time complexity for score prediction and recommendation is computed as follows:

- Compute the conformity scores for all consumed items $|C_t|$ of the target user u_t: In order to compute the conformity score for each consumed item i in $|C_t|$, we need to find the most similar items from $|C_t|$ other than i which requires $(|C_t|-1)k$ operations and computing the conformity score takes $\Theta(1)$ time. Therefore the time complexity of this step is $|C_t|((|C_t|-1)k+1)$ operations which for n_t target users is $\Theta(n_t(|C_t|^2k))$.
- Score prediction: For each test item i in the set of candidate recommended items, we assume that the test item is consumed by the user. Therefore, we need to update the similarity values of the test item i with all the consumed items of the target user which takes $|C_t|m$ operations. In order to compute the conformity score of the test item i, we have to find the k similar items from the set C_t. This takes $|C_t|k$ operations. Computing the conformity score takes 1 operation. Then, we need to update the conformity scores of all items in C_t, if required, which takes $|C_t|$ time. Computing the p-value requires $\Theta(|C_t|)$ operations. The time required for this step for n_t target users is $\Theta(n_t((n-|C_t|)(|C_t|m+|C_t|k)))$
- Recommendation: Recommending the $Top-N$ items with highest scores require $(n-|C_t|)N$ operations which for n_t target users is $\Theta(n_t(n-|C_t|)N)$.

Finally, the time complexity of IBCR is $\Theta(n^2m+n_t(|C_t|^2k+(n-|C_t|)(|C_t|m+|C_t|k)+(n-|C_t|)N)$. Generally, the time complexity for the TCP is calculated as the number of test examples times the number of labels times the underlying algorithm time complexity. But, in our case, it is very less. Because, we do not have labels and also we can pre-compute the conformity scores of the consumed items for all target users and recompute them if and only if the similarity value of newly appended test item is greater than the similarity value of k^{th} most similar item of the consumed item. Therefore, even with large data sets like Eachmovie and Movielens 1M, it is practical to implement TCP on item-based Recommendation algorithm.

6.4 Experimental Results

6.4.1 Experimental Setup

We tested our algorithm on four data sets: MovieLens 100K, EachMovie, MovieLens 1M, MovieLens-latest-small. In order to convert these rating data sets to unary data sets, we converted all rating values to 1. We randomly selected 50 users and for each user 60% of the data is randomly selected for training set and remaining 40% is taken as the test set.

6.4.2 Results and Discussion

We compared the performance of our proposed algorithms UBCR and IBCR with their underlying algorithms UBR and IBR respectively in terms of precision and recall. In our experiments, we take $k = 20$ (maximum number of nearest neighbors). In order to compare with the underlying algorithms, our proposed algorithms should produce fixed length recommendation lists at different *Top-N* values instead of producing the recommendation lists at the specified confidence level. Tables 6.1 and 6.2 show the performance comparison between UBR and UBCR and IBR and IBCR respectively. From these tables, we can observe that, though in some cases, UBR and IBR are showing better performance than their corresponding conformal recommendation algorithms, in most of the cases, the performance of our proposed conformal recommendation algorithms are very close to their underlying algorithms.

As mentioned above, we measure the quality of our conformal recommendation algorithms using two metrics: Validity and Efficiency. When CP is applied for recommendation task, training set and test set of items differ from user to user in contrast to CP in ML where the training set and test sets are fixed for the whole algorithm. So, it is necessary to show that the validity is satisfied for each user. Figures 6.1 and 6.2 show the validity of UBCR and IBCR of each target user for all data sets. We can see that the error values (validity) of most of the users are with in the bounds of ϵ.

Figures 6.3 and 6.4 show the average validity of UBCR and IBCR respectively for different data sets. From these figures, we can see that error values are always bounded by ϵ. Recommendation regions produced by our conformal recommenders are exactly

Dataset	Algorithm	Top-N									
		10	20	30	40	50	60	70	80	90	100
Movielens 100K	UBR	**0.542**	**0.454**	**0.4067**	**0.37**	**0.34**	**0.3137**	**0.2966**	**0.2793**	**0.2627**	**0.2496**
	UBCR	0.538	0.448	0.4	0.3655	0.3392	0.3123	0.294	0.2755	0.2604	0.2474
Eachmovie	UBR	**0.6600**	0.5090	**0.4367**	**0.3830**	**0.3412**	**0.3077**	**0.2800**	**0.2552**	**0.2364**	**0.2198**
	UBCR	0.6320	**0.5100**	0.4313	0.3820	0.3372	0.3037	0.2794	0.2537	0.2331	0.2172
Movielens 1M	UBR	**0.5080**	**0.4390**	**0.3953**	**0.3690**	**0.3396**	**0.3200**	**0.2994**	**0.2862**	**0.2756**	**0.2668**
	UBCR	0.5020	0.4370	0.3947	0.3670	0.3348	0.3117	0.2949	0.2797	0.2738	0.2648
MovieLens-latest-small	UBR	**0.3780**	**0.3310**	**0.2847**	0.2580	**0.2380**	0.2163	0.2023	0.1920	0.1820	**0.1726**
	UBCR	0.3680	0.3280	0.2807	**0.2585**	**0.2380**	**0.2190**	**0.2031**	**0.1952**	**0.1822**	0.1724

(a) Precision comparison of UBR with UBCR

Dataset	Algorithm	Top-N									
		10	20	30	40	50	60	70	80	90	100
Movielens 100K	UBR	0.1836	**0.2891**	**0.3713**	**0.4313**	**0.4823**	**0.5183**	**0.5695**	**0.5948**	**0.6186**	**0.6424**
	UBCR	**0.1862**	0.2879	0.3698	0.4259	0.4754	0.511	0.551	0.5793	0.6041	0.623
Eachmovie	UBR	**0.3239**	**0.4716**	**0.5856**	**0.6646**	**0.7206**	**0.7625**	**0.7949**	**0.8147**	**0.8380**	**0.8562**
	UBCR	0.3073	0.4679	0.5769	0.6628	0.7065	0.7457	0.7812	0.8021	0.8179	0.8391
Movielens 1M	UBR	**0.1282**	0.2030	0.2538	**0.3010**	**0.3416**	**0.3787**	**0.4034**	**0.4323**	**0.4606**	**0.4864**
	UBCR	0.1223	**0.2038**	**0.2557**	0.2985	0.3337	0.3633	0.3896	0.4113	0.4469	0.4753
MovieLens-latest-small	UBR	**0.1239**	**0.2060**	**0.2547**	**0.2927**	0.3277	0.3529	0.3789	0.4028	0.4254	**0.4503**
	UBCR	0.1211	0.2008	0.2440	0.2911	**0.3278**	**0.3617**	**0.3832**	**0.4143**	**0.4293**	0.4496

(b) Recall comparison of UBR with UBCR

Table 6.1: Performance comparison of UBR with UBCR

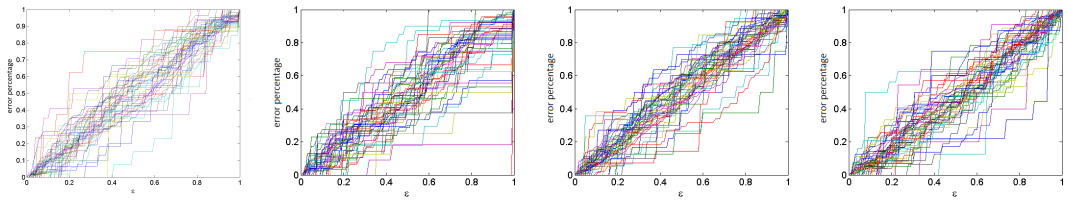


Figure 6.1: Validity of UBCR of individual users for different data sets (left to right: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

valid as they follow a straight line. This is true for both UBCR and IBCR and for all data sets. In other words, the percentage of relevant items not being recommended is same for both UBCR and IBCR in almost all cases. In Figures 6.5 and 6.6 we show the efficiency of our conformal recommendation algorithms. As can be seen from these figures, our proposed algorithms produce very efficient recommendation regions in such a way that the percentage of irrelevant items present in the recommendation regions is less than or around 20% at 80% confidence level (which is sufficiently high confidence level for movies domain). From Figures 6.5 and 6.6, we can say that UBCR is performing better than IBCR as it is producing more efficient recommendation re-

Dataset	Algorithm	Top-N									
		10	20	30	40	50	60	70	80	90	100
Movielens 100K	IBR	0.4580	**0.3900**	0.3327	0.2995	0.2752	0.2550	0.2386	0.2273	0.2169	0.2058
	IBCR	**0.4700**	0.3760	**0.3380**	**0.3055**	**0.2800**	**0.2633**	**0.2463**	**0.2332**	**0.2171**	**0.2066**
Eachmovie	IBR	**0.5720**	0.4620	0.3880	0.3430	0.2992	0.2730	0.2474	0.2260	0.2078	0.1936
	IBCR	0.5660	**0.4630**	**0.3980**	**0.3475**	**0.3048**	**0.2770**	**0.2537**	**0.2353**	**0.2187**	**0.2032**
Movielens 1M	IBR	**0.4340**	**0.3650**	**0.3240**	0.2930	0.2716	0.2553	0.2417	0.2295	0.2198	0.2128
	IBCR	0.4120	0.3640	0.3140	**0.2950**	**0.2748**	**0.2617**	**0.2463**	**0.2345**	**0.2256**	**0.2178**
MovieLens-latest-small	IBR	**0.2960**	**0.2560**	**0.2300**	**0.2040**	**0.1868**	**0.1713**	**0.1626**	**0.1550**	**0.1482**	**0.1426**
	IBCR	0.2840	0.2450	0.2227	0.1985	0.1812	0.1650	0.1514	0.1377	0.1298	0.1212

(a) Precision comparison of IBR with IBCR

Dataset	Algorithm	Top-N									
		10	20	30	40	50	60	70	80	90	100
Movielens 100K	IBR	0.1613	**0.2535**	0.3033	0.3522	**0.3905**	0.4189	0.4432	0.4761	**0.5066**	**0.5289**
	IBCR	**0.1639**	0.2429	**0.3053**	**0.3554**	0.3882	**0.4287**	**0.4590**	**0.4855**	0.4966	0.5189
Eachmovie	IBR	0.2834	0.4350	0.5314	0.6026	0.6464	**0.6932**	0.7253	0.7488	0.7685	0.7930
	IBCR	**0.2835**	**0.4418**	**0.5420**	**0.6101**	**0.6485**	0.6917	**0.7262**	**0.7589**	**0.7845**	**0.8039**
Movielens 1M	IBR	**0.0980**	**0.1539**	**0.1915**	0.2202	**0.2499**	0.2754	**0.2988**	0.3157	0.3353	0.3543
	IBCR	0.0900	0.1423	0.1788	**0.2234**	0.2494	**0.2778**	0.2986	**0.3210**	**0.3433**	**0.3624**
MovieLens-latest-small	IBR	**0.0934**	**0.1613**	**0.2104**	**0.2437**	**0.2741**	**0.3014**	**0.3262**	**0.3496**	**0.3716**	**0.3953**
	IBCR	0.0909	0.1516	0.2053	0.2356	0.2607	0.2847	0.3051	0.3174	0.3380	0.3483

(b) Recall comparison of IBR with IBCR

Table 6.2: Performance comparison of IBR with IBCR

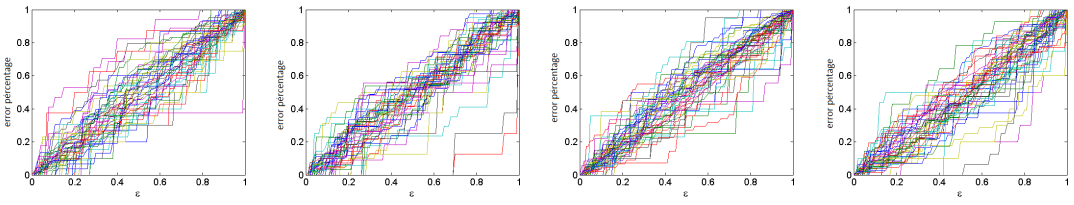


Figure 6.2: Validity of IBCR of individual users for different data sets (left to right: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

gions compared to IBCR.

In Figures 6.7 and 6.8, we show the precision and recall values of UBCR and IBCR at different confidence levels. We know that precision and recall are sensitive to the number of recommendations. If the number of recommended items is much greater than the number of relevant (used) items we will get low precision values. Similarly, if number of relevant (used) items are greater than the number of recommended items the resultant recall values are low. From Figures 6.7 and 6.8, we can observe that precision values are decreasing as we go from lower to higher confidence levels whereas the reverse is true for recall where the values are increasing as we go from lower to

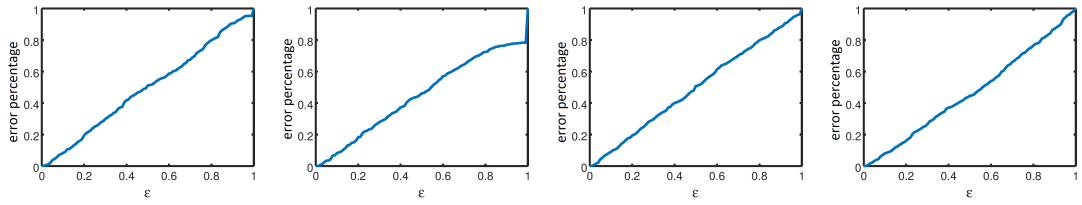


Figure 6.3: Validity of UBCR for different data sets (left to right: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

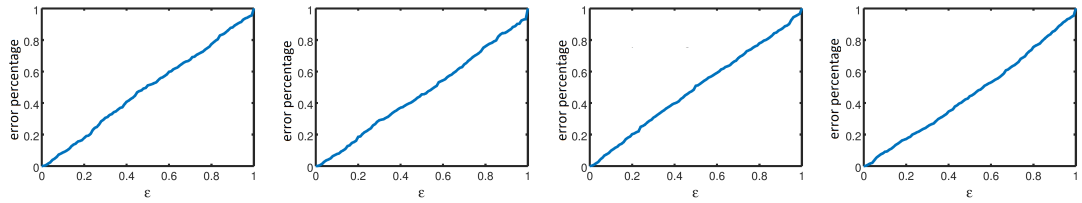


Figure 6.4: Validity of IBCR for different data sets (left to right: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

higher confidence levels. The reason is that the proposed algorithms are recommending more number of items at higher confidence levels ($> 50\%$) compared to the number of recommended items at lower confidence levels ($\leq 50\%$). As a result, the number of recommended items at higher confidence levels are much more than the number of test (relevant or used) items for every target user which leads to a larger number in the denominator (number of recommended items) while calculating the precision which in turn leads to small values of precision at higher confidence levels. Similarly, large number of recommended items at higher confidence levels means that there is a high

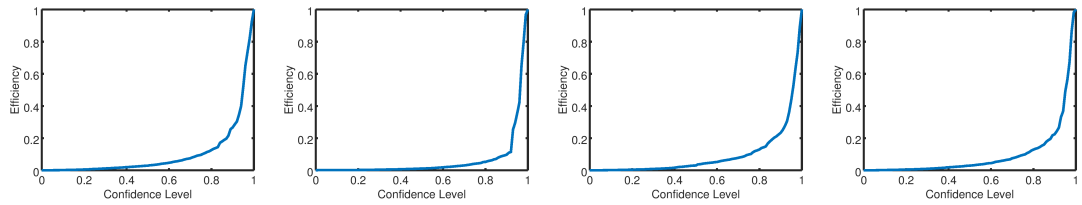


Figure 6.5: Efficiency of UBCR for different data sets (left to right: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

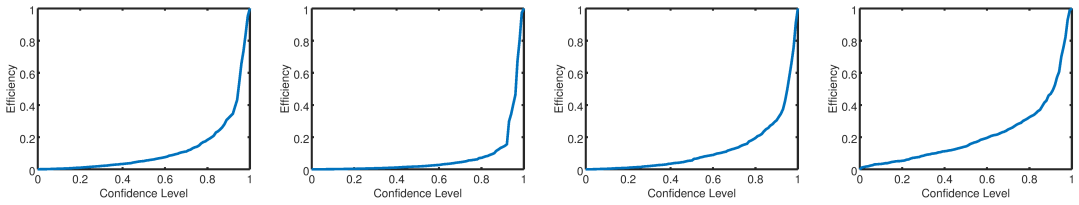


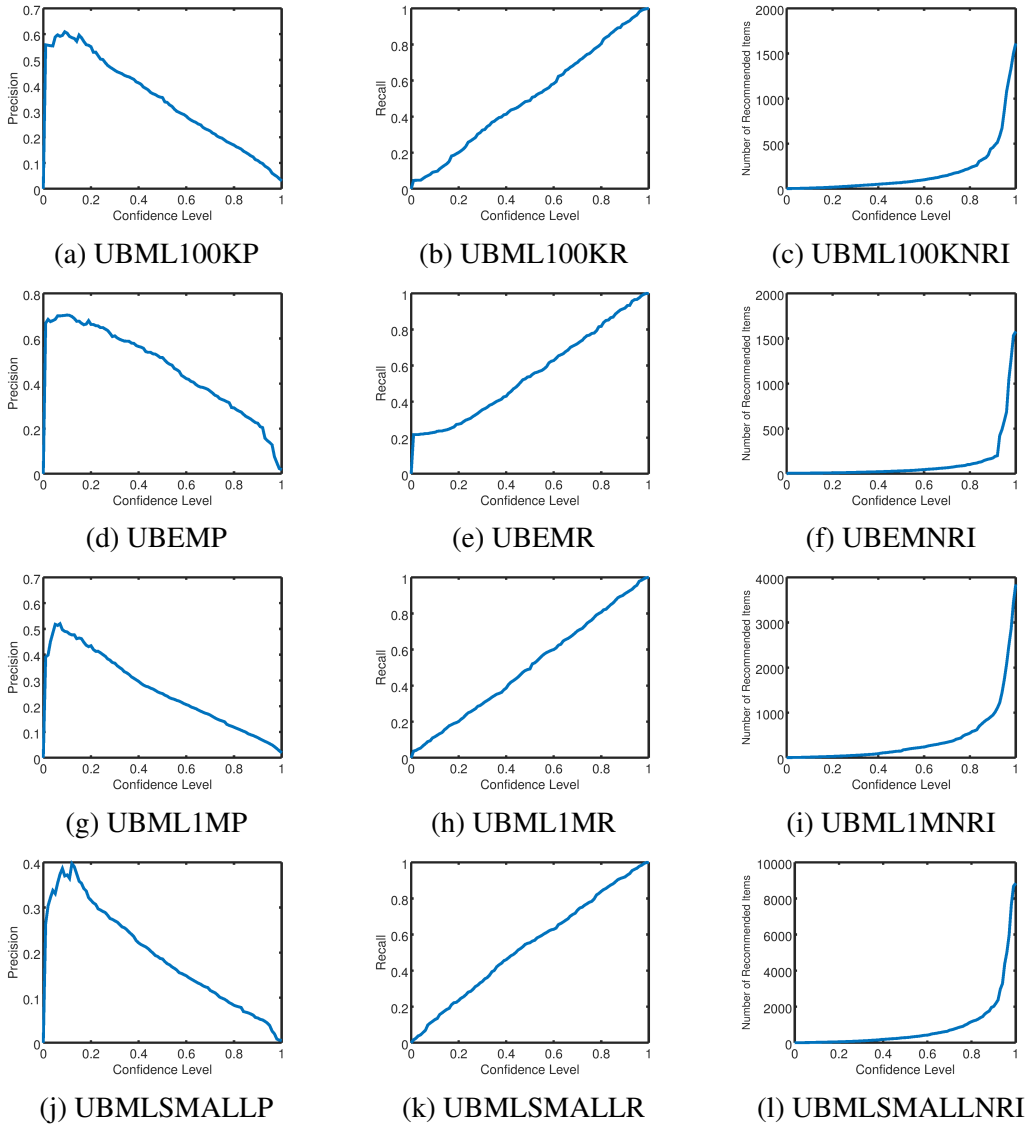
Figure 6.6: Efficiency of IBCR for different data sets (left to right: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

probability that many of the relevant items are being recommended which leads to high recall values at higher confidence levels. To support this, we also show the number of recommended items at each confidence level along with precision and recall in Figures 6.7 and 6.8. Though the algorithms are producing large number of recommended items at higher confidence levels, this number is small compared to total number of items in the data sets. From Figures 6.7 and 6.8, we can observe that UBCR is performing better than IBCR in terms of precision and recall at different confidence levels while at the same time producing less number of recommended items especially at higher confidence levels.

To clearly show the superiority of UBCR over IBCR especially at 60-90% confidence levels, in Table 6.3 we give the precision, recall and percentage of recommended items (%RI) along with the irrelevant items present in the recommendation regions (in our case, efficiency in terms of false positive rate (FPR)) of UBCR and IBCR (in %) at confidence levels 60,70,80 and 90 (from Figures 6.5 to 6.8). From this table, we can clearly observe that though, recall values of IBCR are very close to UBCR, UBCR is producing efficient recommendation regions and giving better precision values compared to IBCR while producing less number of recommended items.

We also measured confidence values for recommendation lists at different Top-N values. Tables 6.4 and 6.5 gives the confidence values for recommendation lists produced by UBCR and IBCR respectively at different *Top-N* values and for different data sets. Here also, the confidence values produced by UBCR are greater than those produced by IBCR.

In summary, though IBCF shows better performance compared to UBCF for prediction task in rating data sets [66], the performance of UBCF for recommendation



(a) UBML100KP (b) UBML100KR (c) UBML100KNRI

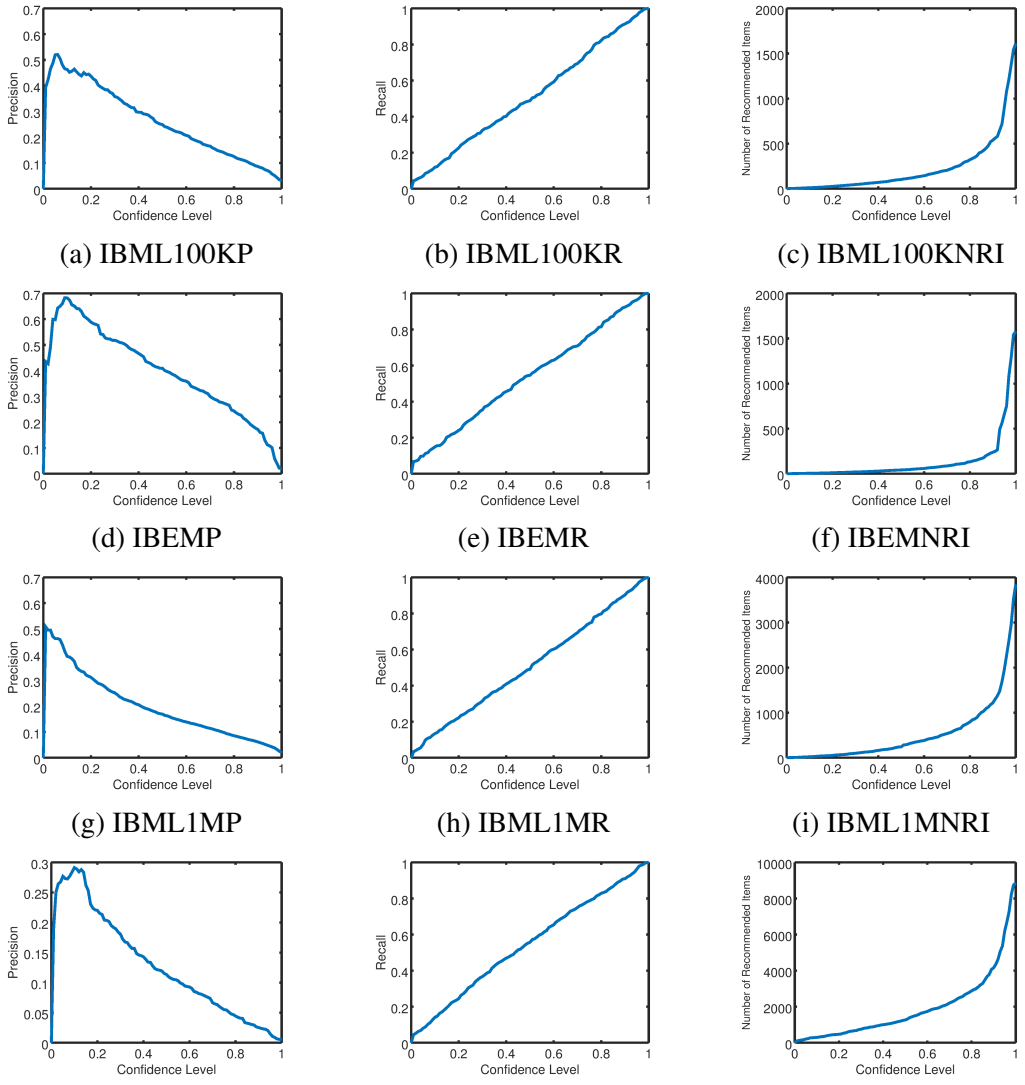
(d) UBEMP (e) UBEMR (f) UBEMNRI

(g) UBML1MP (h) UBML1MR (i) UBML1MNRI

(j) UBMLSMALLP (k) UBMLSMALLR (l) UBMLSMALLNRI

Figure 6.7: Precision,Recall and Number of Recommended Items of UBCR at different confidence levels for different data sets (top to bottom: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

task in unary data sets is better than that of IBCF as shown in Tables 6.1 and 6.2. We have also shown that UBR is better than IBR in terms of producing efficient recommendation regions as shown in 6.4 and also in terms of precision and recall at different confidence levels while producing less number of recommended items especially at higher confidence levels as shown in Figures 6.5 and 6.6. Moreover, the mean confi-



(a) IBML100KP (b) IBML100KR (c) IBML100KNRI

(d) IBEMP (e) IBEMR (f) IBEMNRI

(g) IBML1MP (h) IBML1MR (i) IBML1MNRI

(j) IBMLSMALLP (k) IBMLSMALLR (l) IBMLSMALLNRI

Figure 6.8: Precision,Recall and Number of Recommended Items of IBCR at different confidence levels for different data sets (top to bottom: Movielens 100K, Eachmovie, Movielens 1M and Movielens-latest-small)

dence values produced by UBCR are greater than that of IBCR as shown in Tables 6.4 and 6.5. From this, we can conclude that UBCR is performing better than IBCR for recommendation task.

Confidence Level	Algorithm	Movielens 100K				Eachmovie			
		FPR	Precision	Recall	RI	FPR	Precision	Recall	RI
90%	UBCR	**26.69**	**10.96**	**91.69**	**27.17**	**9.77**	**22.39**	91.84	**10.73**
	IBCR	31.92	8.67	91.39	32.04	13.93	17.20	**92.40**	14.68
80%	UBCR	**12.64**	**16.94**	80.43	**13.73**	**5.26**	**29.25**	**81.71**	**6.29**
	IBCR	18.35	12.53	**82.20**	19.14	7.15	24.37	81‘.47	8.08
70%	UBCR	**7.51**	**22.44**	**70.05**	**8.74**	**3.06**	**36.66**	**72.14**	**4.07**
	IBCR	11.10	16.55	69.41	12.07	4.45	29.97	70.90	5.37
60%	UBCR	**4.71**	**28.15**	58,29	**5.88**	**1.87**	**42.19**	62.78	**2.78**
	IBCR	7.59	20.72	**59.22**	8.56	2.73	35.93	**62.96**	3.58

(a) Movielens 100K and Eachmovie

Confidence Level	Algorithm	Movielens 1M				Movielens-latest-small			
		FPR	Precision	Recall	RI	FPR	Precision	Recall	RI
90%	UBCR	**23.34**	**7.80**	**90.98**	**23.91**	**22.47**	**5.50**	**91.92**	**22.59**
	IBCR	30.52	6.14	90.27	30.79	46.96	2.53	90.95	46.65
80%	UBCR	**13.00**	**11.86**	**80.70**	**13.84**	**12.91**	**8.31**	**83.94**	**13.12**
	IBCR	19.52	8.56	79.78	20.12	32.44	4.44	82.87	32.26
70%	UBCR	**7.80**	**16.59**	**70.35**	**8.68**	**7.53**	**11.63**	**73.51**	**7.78**
	IBCR	12.86	11.42	69.31	13.56	24.84	6.67	74.98	24.72
60%	UBCR	**5.20**	**20.63**	59.98	**6.05**	**4.51**	**14.85**	62.93	**4.76**
	IBCR	8.94	13.89	**60.21**	9.66	19.57	9.28	**65.39**	19.48

(b) Movielens 1M and Movielens-latest-small

Table 6.3: Comparison between UBCR and IBCR

Dataset	Top-N									
	10	20	30	40	50	60	70	80	90	100
Movielens 100K	0.1985	0.2903	0.3617	0.4225	0.4633	0.5093	0.5456	0.5813	0.6155	0.6422
Eachmovie	0.2793	0.4572	0.5514	0.6259	0.6770	0.7131	0.7536	0.7816	0.7998	0.8164
Movielens 1M	0.1333	0.2077	0.2743	0.3084	0.3445	0.3743	0.4057	0.4317	0.4506	0.4706
Movielens-latest-small	0.1186	0.1787	0.2339	0.2715	0.3062	0.3314	0.3566	0.3768	0.3979	0.4143

Table 6.4: Mean confidence values produced by UBCR for fixed length recommendation lists

6.5 Conclusions

In this chapter, we show the adaptation of CP to recommendation task. We applied CP on top of UBCR and IBCR in order to associate confidence values to a set of recommended items. We tested our proposed algorithms on different data sets and we experimentally proved that our algorithms produce efficient recommendation regions at different confidence levels while satisfying the validity property which guarantees

Dataset	Top-N									
	10	20	30	40	50	60	70	80	90	100
Movielens 100K	0.1488	0.2407	0.2943	0.3454	0.3844	0.4145	0.4453	0.4764	0.5053	0.5262
Eachmovie	0.2462	0.4150	0.5193	0.5893	0.6307	0.6693	0.7020	0.7324	0.7495	0.7645
Movielens 1M	0.0914	0.1509	0.1904	0.2225	0.2492	0.2728	0.2935	0.3161	0.3316	0.3514
Movielens-latest-small	0.0731	0.1323	0.1802	0.2057	0.2262	0.2498	0.2650	0.2806	0.2990	0.3141

Table 6.5: Mean confidence values produced by IBCR for fixed length recommendation lists

that the percentage of relevant items not being recommended will not exceed the specified significance level. We also calculate the precision and recall values at different confidence levels. Though at higher confidence levels the algorithms are producing low precision values, the recall values are sufficiently high while producing less number of recommended items compared to the total number of items in the data sets. We also associated confidence values to individual fixed length recommendation lists at different *Top-N* values. Through our experiments we have shown that UBCR is performing better than IBCR in terms of precision and recall while producing less number of recommended items especially at higher confidence values and also in producing efficient recommendation regions. Moreover, the mean confidence values produced by UBCR is greater than that of IBCR. Though, the recommendation regions produced by our algorithms at very high confidence levels are somewhat less efficient especially at 90-99% confidence levels (desired confidence levels in life-critical applications) we can use this algorithm to make predictions in certain recommendation domains such as movies, books, news articles, restaurants, music and in tourism where the confidence level of 60%-90% is also acceptable.

Chapter 7

Conclusions and Future Work

In this thesis, we focus on estimating the confidence of predictions and recommendations with guaranteed error rate in collaborative filtering recommendation algorithms. We start this thesis with a brief introduction of recommender systems and their tasks. We also discussed different kinds of data, algorithms, specifically collaborative filtering algorithms, evaluation approaches and evaluation measures relevant to recommender systems. Neighborhood-based algorithms are the collaborative filtering algorithms which are very simple and easy to understand and at the same time they generate reasonably accurate results. But their performance depends on different aspects such as similarity measures, prediction techniques, data sets and the type of feedback. Therefore, before discussing the confidence estimation we conducted an experimental study on the performance of neighborhood-based algorithms with different similarity measures, prediction techniques and data sets with different types of feedback. We also conducted significance tests to measure the statistical significance of the performance differences between algorithms with different similarity measures, prediction techniques and data sets with different kinds of feedback.

Performance of collaborative filtering algorithms depends on the number of available ratings in the data set as they make predictions and recommendation based only on the available rating information. But most of the real-world data sets used in recommender systems literature tend to be sparse. In addition to sparseness of data sets, factors such as noisy data, untrusted users, change of user preferences over time, choice of different parameter values in the algorithm and suitability of the algorithm for the given data also contribute uncertainty in predictions and recommendations. As a result,

the predictions and recommendations produced by collaborative filtering algorithms are not reliable. Therefore, measuring the reliability of predictions and recommendations generated by collaborative filtering algorithms is a crucial aspect in evaluating those algorithms. In chapters 4, 5 and 6 we adopted conformal prediction framework to different collaborative filtering algorithms for prediction and recommendation tasks in order to measure the confidence of predictions and recommendations generated by the algorithms.

In Chapters 4, 5 and 6 we have shown how conformal prediction provides a standard and reliable way to measure the confidence values for various collaborative filtering algorithms for both prediction and recommendation tasks. We discussed how application of conformal prediction to recommender systems, specifically collaborative filtering algorithms, is different from its application to machine learning and proposed different nonconformity measures or conformity measures suitable to the given underlying algorithm. This is only a first step towards the adaptation of conformal prediction to recommender systems. We can extend this work by proposing new and more efficient nonconformity or conformity measures for the algorithms proposed in Chapters 4, 5 and 6 in order to improve the efficiency of the prediction and recommendation regions. For conformal recommendation algorithms proposed in Chapter 6, we have adapted False Positive Rate to measure the efficiency of recommendation regions. We can also define more efficiency measures based on the ranking ability of the recommendation algorithms. Furthermore, we apply both TCP and ICP to only matrix factorization algorithm in Chapter 5. But in Chapters 4 and 6, we have shown only the application of TCP to the underlying algorithms (IBCF in Chapter 4 and UBR and IBR in Chapter 6). We can also apply ICP for the underlying algorithms in Chapters 4 and 6 and compare its results with that of TCP. Other extensions are adapting the conformal prediction to the recommendation algorithms other than collaborative filtering algorithms such as content-based and context-aware recommendation algorithms. and also investigating the application of conformal prediction to cross-domain recommendation algorithms. These extensions are briefly described below:

1. Improving the efficiency of conformal predictors or recommenders by defining more efficient nonconformity measures

In chapters 4 and 5 we tried to improve the efficiency of the prediction regions produced by the proposed conformal prediction algorithms by defining a number of

nonconformity measures based on the underlying algorithm. Though, the proposed algorithms are producing sufficiently enough number of single labels and correct predictions at 60%-90% confidence levels which can be acceptable in some recommendation domains such as movies, books, news articles, restaurants, music, but they are not acceptable when used in medical applications where the desirable confidence level is 90-99%. Therefore, in order to produce large number of single labels and large percentage of correct predictions at higher confidence levels i.e., 90-99% in medical recommender systems, we need to define more efficient nonconformity measures.

In chapter 6, our main aim is to show the applicability of conformal prediction to recommendation task rather than the prediction task for which the conformal prediction algorithms are originally designed for. Therefore, we define only a simple conformity score based on the underlying algorithm. Though, our proposed algorithms are producing sufficiently high recall values and moderate values of precision at 60%-90% confidence levels , we can improve this by defining more efficient conformity measures. Furthermore, we can also define efficiency measures for recommendation task based on the ranking ability of the recommendation algorithm.

2. Application of ICP to IBCF for prediction task and UBR and IBR for recommendation task

In chapters 4 and 6 we have shown only the applicability of TCP to IBCF for prediction task and UBR and IBR for recommendation task. But we can also apply ICP to these algorithms and analyze the performance difference between TCP and ICP. We can check whether the results are consistent with the results that we got for matrix factorization. Unlike, in matrix factorization, where we got good results for ICP compared to TCP, in neighborhood-based algorithms, there may be a chance that we get exactly opposite results for both prediction and recommendation tasks due to the following reasons:

- In matrix factorization the labels of many training examples are zero in TCP (main problem while computing NCM5 and NCM6). But this problem is not present in neighbourhood-based algorithms because ground truth is always available to compute the nonconformity or conformity scores of training examples.
- In neighborhood-based algorithms, the number of neighbours to compute the nonconformity or conformity scores will be decreased in ICP compared to TCP.

Because we need to take out some items from the training set to form the calibration set. Alternatively, we can select only those users having at least the minimum number of neighbours for each item as we did in Chapter 4.

- In neighborhood-based algorithms, the number of training examples in ICP with which we are comparing the nonconformity or conformity score of a test example is less compared to TCP. But this is not the case in matrix factorization as discussed in Chapter 5.

3. Application of conformal prediction to content-based [60] and context-aware [6] recommendation algorithms

In our thesis, we have limited our attention to collaborative filtering algorithms. But, there are several algorithms other than the collaborative filtering algorithms to make recommendations such as content-based and context-based algorithms. Content-based algorithms recommend items based on the similarity between the items' attributes and user profiles. To do this similarity computation they make use of additional information other than the rating information. The additional information used here is item attributes such as genre, date, price etc. and user profile is generated from usage history of the user and item attributes. Context-aware recommender systems generate recommendations by incorporating the contextual information of the user in the recommendation algorithms. Context in recommender systems generally refers several factors such as time, location, companion (for watching the movies, going to restaurants etc.), purpose etc. By incorporating the additional information such as item attributes in content-based recommender systems and contextual information in context-aware recommender systems we can provide more relevant recommendations to the users. But, missing and erroneous item attributes and contextual factors along with the sparse data might lead to uncertainty. Therefore, adapting the conformal prediction to these algorithms along with the appropriate and more efficient nonconformity measures to determine the reliability of the predictions and recommendations produced by the algorithms is very useful.

4. Conformal prediction in cross-domain recommendation [10, 13, 14, 19, 40]

Most of the recommender systems based on collaborative filtering recommend items to the users using only the information that is limited to single domain. But, in general users can rate only a limited number of items which leads to extremely

sparse rating matrices. This prevents the system from generating accurate recommendations. To address this data sparsity problem cross domain recommender systems transfer knowledge from source domain to target domain by using some machine learning techniques or they extract information from both domains to improve the accuracy of recommendations produced by both domains and to make joint recommendations. Cross-domain recommender systems are also used to improve the diversity in recommendations. The assumption in cross domain recommender systems is there exists relationship between users and items in source and target domains.

We can use conformal prediction to transfer learning recommendation algorithms in order to improve the accuracy of predictions and recommendations made in the target domain. Accuracy of the transfer learning recommender systems depends on the information that is extracted from the source domain which is then transferred to the target domain. In other words, the more relevant the extracted information from source domain is to the target domain, the more accurate the predictions and recommendations are. For this reason, we use conformal prediction in transfer learning recommender systems to extract the information from source domain that is relevant to target domain.

Several cross-domain recommendation approaches are proposed in recommender systems literature. Among them Codebook transfer (CBT) [41] and Rating-matrix generative model (RMGM) [42] are the popular algorithms used to make comparisons with the other cross-domain algorithms. Therefore, we give some brief ideas on how to apply conformal prediction to these algorithms.

- Conformal prediction in CBT:

  The assumption in CBT is that source domain and target domain share a common cluster level rating pattern which we call it as the codebook. This pattern should be extracted from the source domain and the users and items from the target domain should be mapped to the corresponding user and item clusters in the source domain in order to make accurate predictions. The disadvantage of this approach is that the source domain should be dense which is not the case in real world scenarios. We can use CP to extract a set of users and items from the source domain which are relevant to the target domain. But as most of the real world data sets are sparse, the matrix formed by a selected subset of users and items is also sparse. As a result, most of the entries in the resulting codebook

are zeros. The remaining nonzero entries contain very small numbers (approximately equal to zero) like 0.0001, 0.0004, 0.0052 and so on. As a result, all the predicted rating values are also approximately equal to zero. Therefore, it is not the appropriate way to apply conformal prediction to CBT.

Another way to use conformal prediction in CBT is to map users and items of the target domain to the corresponding user and item clusters in the target domain.

- Conformal prediction in RMGM:

  Here also, the assumption is that source and target domains share a common cluster level rating pattern. But, unlike in CBT, where the cluster level rating pattern is extracted form the source domain, in RMGM, this pattern is extracted from both domains and use this pattern to improve the accuracy in both domains. Here both domains are assumed to be sparse. In this paper, authors have selected subset of matrices from the source domain and target domain by randomly choosing the 500 users with more than 20 ratings and 1000 items from each domain. Though the main goal in this paper is to improve the accuracy in both domains, in transfer learning, we focus only on improvement of accuracy in target domain. Therefore, instead of selecting source domain users and items randomly, we can use conformal prediction to choose the users and items from source domain which are relevant to the target domain.

We can extend this work to other cross-domain collaborative algorithms to improve the accuracy of predictions or recommendations or to improve the diversity in recommendations in target domain and also to make joint recommendations.

References

[1] **EachMovie collaborative filtering data set**. `http://www.cs.cmu.edu/~lebanon/IR-lab/data.html`. (23)

[2] **MovieLens**. `https://grouplens.org/datasets/movielens/`. (23)

[3] PANAGIOTIS ADAMOPOULOS AND ALEXANDER TUZHILIN. **On Unexpectedness in Recommender Systems: Or How to Better Expect the Unexpected**. *ACM Transactions on Intelligent Systems and Technology*, **5**(4):54:1–54:32, 2014. (33)

[4] GEDIMINAS ADOMAVICIUS, SREEHARSHA KAMIREDDY, AND YOUNGOK KWON. **Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance**. In *Workshop on Information Technology and Systems*, 2007. (6, 101, 105, 110)

[5] GEDIMINAS ADOMAVICIUS AND ALEXANDER TUZHILIN. **Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions**. *IEEE Transactions on Knowledge and Data Engineering*, **17**(6):734–749, 2005. (10)

[6] GEDIMINAS ADOMAVICIUS AND ALEXANDER TUZHILIN. **Context-Aware Recommender Systems**. In *Recommender Systems Handbook*, pages 191–226. 2015. (205)

[7] ALIJAH AHMED. *Cochran's Q Test Calculator with Post-hoc Analysis*, 2013. (57)

[8] VINEETH BALASUBRAMANIAN, SHEN-SHYANG HO, AND VLADIMIR VOVK. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and*

Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2014. (6, 110)

[9] TONY BELLOTTI, ZHIYUAN LUO, ALEX GAMMERMAN, FREDERICK W.VAN DELFT, AND VASKAR SAHA. **Qualified predictions for microarray and proteomics pattern diagnostics with confidence machines**. *Neural Systems*, **15**(4):247–258, 2005. (118)

[10] SHLOMO BERKOVSKY, TSVI KUFLIK, AND FRANCESCO RICCI. **Cross-Domain Mediation in Collaborative Filtering**. In *International Conference on User Modeling*, pages 355–359, 2007. (205)

[11] SIDDHARTHA BHATTACHARYYA. **Confidence in predictions from random tree ensembles**. *Knowledge and Information Systems*, **35**(2):391–410, 2013. (117)

[12] LAURENT CANDILLIER, MAX CHEVALIER, DAMIEN DUDOGNON, AND JOSIANE MOTHE. **Diversity in Recommender Systems : Bridging the gap between users and systems**. In *International Conference on Advances in Human-oriented and Personalized Mechanisms*, 2011. (33)

[13] IVÁN CANTADOR, , AND PAOLO CREMONESI. **Tutorial on cross-domain recommender systems**. pages 401–402, 2014. (205)

[14] IVÁN CANTADOR, IGNACIO FERNÁNDEZ-TOBÍAS, SHLOMO BERKOVSKY, AND PAOLO CREMONESI. *Cross-Domain Recommender Systems*, pages 919–959. 2015. (205)

[15] PABLO CASTELLS, NEIL J. HURLEY, AND SAUL VARGAS. **Novelty and Diversity in Recommender Systems**. In *Recommender Systems Handbook*, pages 881–918. 2015. (32, 33)

[16] PABLO CASTELLS AND SAL VARGAS. **Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance**. In *International Workshop on Diversity in Document Retrieval (DDR*, pages 29–37, 2011. (32, 33)

[17] MIKHAIL DASHEVSKIY AND ZHIYUAN LUO. **Network Traffic Demand Prediction with Confidence**. In *Proceedings of the Global Communications Conference*, pages 1453–1457, 2008. (118)

[18] DMITRY DEVETYAROV AND ILIA NOURETDINOV. **Prediction with Confidence Based on a Random Forest Classifier**. In *Artificial Intelligence Applications and Innovations*, pages 37–44, 2010. (117)

[19] IGNACIO FERNNDEZ-TOBAS, IVN CANTADOR, MARIUS KAMINSKAS, AND FRANCESCO RICCI. **Cross-domain recommender systems: A survey of the State of the Art**. 2012. (205)

[20] SIMON FUNK. **Netflix update: try this at home**. 2006. (20, 134)

[21] ALEXANDER GAMMERMAN, VOLODYA VOVK, BRIAN BURFORD, ILIA NOURETDINOV, ZHIYUAN LUO, ALEXEY YA. CHERVONENKIS, MIKE WATERFIELD, RAINER CRAMER, PAUL TEMPST, JOSEP VILLANUEVA, MUSARAT KABIR, STEPHANE CAMUZEAUX, JOHN TIMMS, USHA MENON, AND IAN JACOBS. **Serum Proteomic Abnormality Predating Screen Detection of Ovarian Cancer**. *Computer*, **52**(3):326–333, 2009. (117)

[22] FLORENT GARCIN, BOI FALTINGS, OLIVIER DONATSCH, AYAR ALAZZAWI, CHRISTOPHE BRUTTIN, AND AMR HUBER. **Offline and online evaluation of news recommender systems at swissinfo.ch**. In *ACM Conference on Recommender Systems*, pages 169–176, 2014. (28)

[23] MOUZHI GE, CARLA DELGADO-BATTENFELD, AND DIETMAR JANNACH. **Beyond accuracy: evaluating recommender systems by coverage and serendipity**. In *ACM Conference on Recommender Systems*, pages 257–260, 2010. (32, 33)

[24] JONATHAN L. HERLOCKER, JOSEPH A. KONSTAN, AL BORCHERS, AND JOHN RIEDL. **An Algorithmic Framework for Performing Collaborative Filtering**. *SIGIR Forum*, **51**(2):227–234, 1999. (32, 40, 59)

[25] JONATHAN L. HERLOCKER, JOSEPH A. KONSTAN, LOREN G. TERVEEN, AND JOHN RIEDL. **Evaluating collaborative filtering recommender systems**. *Transactions on Information Systems*, **22**(1):5–53, 2004. (2, 29, 101)

[26] WILL HILL, LARRY STEAD, MARK ROSENSTEIN, AND GEORGE FURNAS. **Recommending and Evaluating Choices in a Virtual Community of Use**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 194–201, 1995. (5, 104, 118, 181)

[27] GAWESH JAWAHEER, MARTIN SZOMSZOR, AND PATTY KOSTKOVA. **Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service**. In *Information Heterogeneity and Fusion in Recommender Systems*, pages 47–51, 2010. (12)

[28] ULF JOHANSSON, HENRIK BOSTRÖM, AND TUVE LÖFSTRÖM. **Conformal Prediction Using Decision Trees**. In *International Conference on Data Mining*, pages 330–339, 2013. (117)

[29] ULF JOHANSSON, RIKARD KÖNIG, TUVE LÖFSTRÖM, AND HENRIK BOSTRÖM. **Evolved decision trees as conformal predictors**. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1794–1801, 2013. (117)

[30] MARIUS KAMINSKAS AND DEREK BRIDGE. **Measuring Surprise in Recommender Systems**. In *ACM RecSys Workshop on Recommender Systems Evaluation: Dimensions and Design*, 2014. (33)

[31] MARIUS KAMINSKAS AND DEREK BRIDGE. **Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems**. *Transactions on Internet and Information Systems*, **7**(1):2:1–2:42, 2017. (32, 33)

[32] BART P. KNIJNENBURG, MARTIJN C. WILLEMSEN, ZENO GANTNER, HAKAN SONCU, AND CHRIS NEWELL. **Explaining the user experience of recommender systems**. *User Modeling and User-Adapted Interaction*, **22**(4-5):441–504, 2012. (26)

[33] RON KOHAVI, ROGER LONGBOTHAM, DAN SOMMERFIELD, AND RANDAL M. HENNE. **Controlled experiments on the web: survey and practical guide**. *Data Mining and Knowledge Discovery*, **18**(1):140–181, 2009. (28)

[34] YEHUDA KOREN, ROBERT M. BELL, AND CHRIS VOLINSKY. **Matrix Factorization Techniques for Recommender Systems**. *IEEE Computer*, **42**(8):30–37, 2009. (1, 5, 19, 20, 134)

[35] YEHUDA KOREN AND JOE SILL. **OrdRec: an ordinal model for predicting personalized item rating distributions**. In *ACM Conference on Recommender Systems*, pages 117–124, 2011. (6, 108, 110)

[36] DENIS KOTKOV, SHUAIQIANG WANG, AND JARI VEIJALAINEN. **A survey of serendipity in recommender systems**. *Knowledge-Based Systems*, **111**:180–192, 2016. (33)

[37] MATEVZ KUNAVER AND TOMAZ POZRL. **Diversity in recommender systems - A survey**. *Knowledge-Based Systems*, **123**:154–162, 2017. (33)

[38] DANIEL D. LEE AND H. SEBASTIAN SEUNG. **Learning the parts of objects by non-negative matrix factorization**. *Nature*, **401**(6755):788–791, 1999. (20, 134, 135)

[39] DANIEL LEMIRE AND ANNA MACLACHLAN. **Slope One Predictors for Online Rating-Based Collaborative Filtering**. *CoRR*, **abs/cs/0702144**, 2007. (22, 119, 182)

[40] BIN LI. **Cross-Domain Collaborative Filtering: A Brief Survey**. In *International Conference on Tools with Artificial Intelligence*, pages 1085–1086, 2011. (205)

[41] BIN LI, QIANG YANG, AND XIANGYANG XUE. **Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction**. In *International Joint Conference on Artificial Intelligence*, pages 2052–2057, 2009. (206)

[42] BIN LI, QIANG YANG, AND XIANGYANG XUE. **Transfer learning for collaborative filtering via a rating-matrix generative model**. In *International Conference on Machine Learning*, pages 617–624, 2009. (206)

[43] KENT STATE UNIVERSITY LIBRARIES. *SPSS Tutorials: Paired Samples t Test*, 2013 (accessed October 6, 2017). `https://libguides.library.kent.edu/SPSS/PairedSamplestTest`. (53)

[44] RICHARD LOWRY. *One-Way Analysis of Variance for Correlated Samples*, 1999 (accessed 2017). (54)

[45] RICHARD LOWRY. *McNemar's test*, 2001 (accessed 2017). (56)

[46] RICHARD LOWRY. *One-Way Analysis of Variance for Independent or Correlated Samples*, 2001 (accessed 2017). `http://vassarstats.net/anova1u.html`. (54)

[47] MACIEJ A. MAZUROWSKI. **Estimating confidence of individual rating predictions in collaborative filtering recommender systems**. *Expert Systems with Applications*, **40**(10):3847–3857, 2013. (6, 108)

[48] MATTHEW R. MCLAUGHLIN AND JONATHAN L. HERLOCKER. **A collaborative filtering algorithm and evaluation metric that accurately model the user experience**. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 329–336, 2004. (6, 103, 104, 110, 181)

[49] SEAN M. MCNEE, SHYONG K. LAM, CATHERINE GUETZLAFF, JOSEPH A. KONSTAN, AND JOHN RIEDL. **Confidence Displays and Training in Recommender Systems**. In *Human-Computer Interaction*, 2003. (5, 6, 100, 101, 102, 110)

[50] SEAN M. MCNEE, JOHN RIEDL, AND JOSEPH A. KONSTAN. **Being accurate is not enough: how accuracy metrics have hurt recommender systems**. In *Conference on Human Factors in Computing Systems*, pages 1097–1101, 2006. (14)

[51] ILIA NOURETDINOV, THOMAS MELLUISH, AND VOLODYA VOVK. **Ridge Regression Confidence Machine**. In *International Conference on Machine Learning*, pages 385–392, 2001. (117)

[52] ILIA NOURETDINOV, VOLODYA VOVK, MICHAEL V. VYUGIN, AND ALEXANDER GAMMERMAN. **Pattern Recognition and Density Estimation under the General i.i.d. Assumption**. In *European Conference on Computational Learning Theory*, pages 337–353, 2001. (111)

[53] HARRIS PAPADOPOULOS. **Inductive conformal prediction: theory and application to neural networks**. In *Tools in Artificial Intelligence*, pages 315–330, 2008. (112, 117, 179)

[54] HARRIS PAPADOPOULOS, ALEXANDER GAMMERMAN, AND VOLODYA VOVK. **Confidence Predictions for the Diagnosis of Acute Abdominal Pain**. In *Artificial Intelligence Applications and Innovations*, pages 175–184, 2009. (117)

[55] HARRIS PAPADOPOULOS, EFI PAPATHEOCHAROUS, AND ANDREAS S. ANDREOU. **Reliable Confidence Intervals for Software Effort Estimation**. In *Artificial Intelligence Applications & Innovations*, pages 211–220, 2009. (118)

[56] HARRIS PAPADOPOULOS, KOSTAS PROEDROU, VOLODYA VOVK, AND ALEXANDER GAMMERMAN. **Inductive Confidence Machines for Regression**. In *International Symposium on Artificial Intelligence and Mathematics*, 2002. (117)

[57] HARRIS PAPADOPOULOS, VLADIMIR VOVK, AND ALEXANDER GAMMERMAN. **Qualified Prediction for Large Data Sets in the Case of Pattern Recognition**. In *International Conference on Machine Learning and Applications*, pages 159–163, 2002. (117)

[58] HARRIS PAPADOPOULOS, VLADIMIR VOVK, AND ALEXANDER GAMMERMAN. **Conformal prediction with neural networks**. In *International Conference on Tools with Artificial Intelligence*, pages 388–395, 2007. (117, 128, 154)

[59] HARRIS PAPADOPOULOS, VLADIMIR VOVK, AND ALEXANDER GAMMERMAN. **Regression Conformal Prediction with Nearest Neighbours**. *Computing Research Repository*, **abs/1401.3880**, 2014. (158)

[60] MICHAEL J. PAZZANI AND DANIEL BILLSUS. **The Adaptive Web**. chapter Content-based Recommendation Systems, pages 325–341. 2007. (205)

[61] KOSTAS PROEDROU, ILIA NOURETDINOV, VOLODYA VOVK, AND ALEXANDER GAMMERMAN. **Transductive Confidence Machines for Pattern Recognition**. In *European Conference on Machine Learning*, pages 381–390, 2002. (117, 122, 156)

[62] PEARL PU, LI CHEN, AND RONG HU. **A user-centric evaluation framework for recommender systems**. In *ACM Conference on Recommender Systems*, pages 157–164, 2011. (26)

[63] JASON D. M. RENNIE AND NATHAN SREBRO. **Fast maximum margin matrix factorization for collaborative prediction**. In *International Conference on Machine Learning*, pages 713–719, 2005. (20, 134, 135)

[64] PAUL RESNICK, NEOPHYTOS IACOVOU, MITESH SUCHAK, PETER BERGSTROM, AND JOHN RIEDL. **GroupLens: An Open Architecture for Collaborative Filtering of Netnews**. In *Computer Supported Cooperative Work*, pages 175–186, 1994. (1, 17, 38)

[65] SAL VARGAS SANDOVAL. *Novelty and Diversity Evaluation and Enhancement in Recommender Systems*. PhD thesis, Autonomous University of Madrid, 2015. (26)

[66] BADRUL M. SARWAR, GEORGE KARYPIS, JOSEPH A. KONSTAN, AND JOHN RIEDL. **Item-based collaborative filtering recommendation algorithms**. In *World Wide Web*, pages 285–295, 2001. (1, 17, 18, 45, 197)

[67] CRAIG SAUNDERS, ALEXANDER GAMMERMAN, AND VOLODYA VOVK. **Transduction with Confidence and Credibility**. In *International Joint Conference on Artificial Intelligence*, pages 722–726, 1999. (117)

[68] CRAIG SAUNDERS, ALEXANDER GAMMERMAN, AND VOLODYA VOVK. **Computationally Efficient Transductive Machines**. In *Algorithmic Learning Theory*, pages 325–333, 2000. (117)

[69] GLENN SHAFER AND VLADIMIR VOVK. **A Tutorial on Conformal Prediction**. *Journal of Machine Learning Research*, **9**:371–421, 2008. (6, 110, 111, 139)

[70] GUY SHANI AND ASELA GUNAWARDANA. **Evaluating Recommendation Systems**. In *Recommender Systems Handbook*, pages 257–297. 2011. (4, 5, 6, 23, 29, 33, 100, 107)

[71] GUY SHANI, LIOR ROKACH, BRACHA SHAPIRA, SARIT HADASH, AND MORAN TANGI. **Investigating confidence displays for top-*N* recommendations**. *Journal of the Association for Information Science and Technology*, **64**(12):2548–2563, 2013. (100)

[72] NATHAN SREBRO, JASON D. M. RENNIE, AND TOMMI S. JAAKKOLA. **Maximum-margin matrix factorization**. In *Advances in Neural Information Processing Systems*, pages 1329–1336, 2004. (20, 134, 135)

[73] SAUL VARGAS AND PABLO CASTELLS. **Rank and relevance in novelty and diversity metrics for recommender systems**. In *Recommender Systems*, pages 109–116, 2011. (32, 33)

[74] VLADIMIR VOVK, VALENTINA FEDOROVA, ILIA NOURETDINOV, AND ALEX GAMMERMAN. **Criteria of efficiency for conformal prediction**. Technical report, Technical report, Royal Holloway University of London, 2014. (116, 153, 154)

[75] VLADIMIR VOVK, ALEX GAMMERMAN, AND GLENN SHAFER. *Algorithmic Learning in a Random World*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. (6, 110, 141, 180)

[76] WIKIPEDIA. *Cochran's Q test*, (accessed August 8, 2017). (57)

[77] WIKIPEDIA. *McNemar's test*, (accessed November 11, 2017). (56)

[78] RON ZACHARSKI. *A Programmer's Guide to Data Mining: The Ancient Art of the Numerati*. 2015. (13, 181)

[79] JIN ZHANG, GUANG LI, MENG HU, JIAOJIE LI, AND ZHIYUAN LUO. **Recognition of hypoxia EEG with a preset confidence level based on EEG analysis**. In *International Joint Conference on Neural Networks*, pages 3005–3008, 2008. (117)

[80] LIANG ZHANG. **The Definition of Novelty in Recommendation System**. **6**:141–145, 2013. (32)

[81] CAI-NICOLAS ZIEGLER, SEAN M. MCNEE, JOSEPH A. KONSTAN, AND GEORG LAUSEN. **Improving recommendation lists through topic diversification**. In *World Wide Web*, pages 22–32, 2005. (33)

SYNOPSIS OF

Prediction and Recommendation with Confidence in Collaborative Filtering Recommender Systems

A thesis submitted during 2018 to the University of Hyderabad in partial fulfillment of the award of a Ph.D. degree in Computer Science

to be submitted by

TADIPARTHI V R HIMABINDU (12MCPC04)

SCHOOL OF COMPUTER AND INFORMATION SCIENCES

UNIVERSITY OF HYDERABAD

(P.O.) CENTRAL UNIVERSITY

HYDERABAD - 500 046, INDIA

July-2018

1 Introduction

With the increasing amount of data on the World Wide Web, the problem of finding useful information has become more and more difficult. To deal with this information overload problem, recommender systems have been developed. Recommender systems (RS) help users make more effective decisions by recommending the items of their interest based on users' past behavior. Recommender sytems are now being used by many websites to increase their revenue while maintaining the user satisfaction by producing more effective recommendations. Collaborative filtering recommendation algorithms are the most widely adopted and successful techniques used to build recommender systems both in academic research and commercial applications. Classic methods of collaborative filtering include *neighborhood-based* methods and recent methods are revolving around *model-based* approaches especially matrix factorization techniques. In neighborhood-based approaches, prediction and recommendations can be done either by computing the similarities between users (user-based collaborative filtering [10]) or similarities between items (item-based collaborative filtering [11]). User-based approach identifies users whose tastes are similar to those of the active user and recommends items they have liked. Item-based approach uses item-item similarity to make the predictions. On the other hand, model-based approaches use mathematical models to make the predictions. Particularly, matrix factorization techniques [5] have recently gained much popularity because of their scalability, accuracy and also their successful application in the Netflix Prize competition.

Over the last two decades, the research in recommender systems has been focusing on developing new algorithms and enhancing the existing algorithms to improve the performance. In order to measure the performance of an algorithm and to choose the best algorithm among the available algorithms one needs to evaluate and compare these algorithms. Several measures have been proposed to evaluate recommender systems from different perspectives. Still, evaluating recommender systems is inherently difficult due to the following reasons [3]:

1. Which measure should be used to determine the quality of recommendation algorithms?

Majority of the papers in RS literature focus on accuracy of recommender systems.

But, most accurate predictions are not always the most useful ones to the users. For example, though RS predicts correct ratings for items, it may not be useful to the user if the user is not interested in ratings and expects only those items which he likes. Similarly, the recommendations may not be useful to the user, if the recommender system recommends the most popular items. Because, the user can find those items by him/herself. Likewise, if the recommended items are very much similar to those which are already consumed by the user, the user will get bored of similar recommendations. Similarly, recommending similar items for instance, similar places in tourism recommendation will not be helpful to the user as the recommendation list should be diverse in tourism domain. Therefore, the measure that should be used to determine the quality of RS depends on the users' task and their requirements and the recommendation domain.

2. The performance of RS varies depending on the characteristics of the data set.

The performance of RS is effected by many characteristics of the data set like number of users, number of items and sparsity of the data set. An algorithm which performs well on a data set with number of users greater than the number of items may not perform well on a data set with number of items greater than the number of users. Similarly, an algorithm which performs well on dense data set may perform worse on sparse data set.

3. The goal for which the RS is evaluated.

The goal for which the RS is designed will also play a significant role in evaluation. For example, improving user satisfaction and increasing revenue are two important goals of a recommender system. These goals require completely different approaches and different measures for evaluation as compared to the traditional ones.

A typical recommender system can perform two tasks: *prediction* and *recommendation*. In prediction task, the algorithm predicts how the user will rate a particular item. In recommendation task, the algorithm presents a list of items that the user likes. The recommendation can be done in two ways: 1. Prediction-based recommendation: Here, like in prediction task, the algorithm predicts rating for every unused item and recommends items with highest ratings. 2. Score-based recommendation: For every unused item, a score will be calculated which tells the likelihood of this item being recommended. The higher the score, the better to consider it for recommendation. Fi-

nally, items with high scores will be recommended. Type of feedback in the data set will also effect the performance of the algorithm. Therefore, choosing the data set with appropriate feed back for the given task is also very important in evaluation. We have mainly two types of feedback available: *explicit* and *implicit*. Explicit feedback is the feedback given by the users explicitly, whereas implicit feedback is derived from the users' behavior. We can divide explicit feedback data set into two types: Numerical feedback where the feedback is in the form of ratings like 1, 2, 3, 4 and 5 and binary feedback which contains information about whether the user likes or dislikes the item instead of having ratings. Implicit feedback data sets which are also called unary data sets contain information about whether the user has used that item or not. But, unary feedback can be explicitly given by the user and binary feedback can be implicitly derived from user's behavior. In this case, unary feedback comes under explicit feedback and binary feedback comes under implicit feedback. Rating prediction and prediction-based recommendation will be generally performed on data sets with explicit feedback and score computation and score-based recommendation on implicit feedback data sets.

Despite having several measures for evaluating the RS, accuracy measures are the most sought after evaluation measures. Several accuracy measures have been adapted from statistics and information retrieval to measure the accuracy of recommender systems. Among them, prediction accuracy measures like *Mean Absolute Error (MAE)* and *Root Mean Squared Error (RMSE)* and recommendation accuracy measures like *precision* and *recall* are very popular. Choosing the appropriate accuracy measure for the given task as well as the type of feedback in the data set is one of the crucial aspects involved in developing recommender systems.

Though accuracy is very important, it is not the only measure that determines the quality of a recommendation algorithm. Several other measures such as coverage, novelty, unexpectedness, serendipity, diversity and confidence were proposed to evaluate the recommender systems from different perspectives [13]. Among them confidence is very important as it acts as a reliability measure which measures the uncertainty associated with the predictions made by the recommendation algorithm. This in turn enables the users to make more effective decisions about which items to buy, which books to read, which movies to watch etc. Confidence tells us the system's trust in its predictions or recommendations. It helps users to distinguish between confident and inadequate recommendations thus by enabling them to make intelligent choices.

2 Motivation

Evaluation measures are used to find how well a given algorithm performs, to compare different recommendation algorithms along with their variants and to select the appropriate algorithm in a given context. There is no single measure to determine the quality of the recommendation algorithms suitable to all contexts. Several evaluation measures have been proposed in the literature to measure the recommender systems' performance. All these different measures assess the performance of recommendation algorithms from different perspectives. We need to choose a suitable evaluation measure for a given context.

Though accuracy measures are the most popular for evaluation in recommender systems, there are many aspects of the recommender systems which are not taken into consideration by accuracy measures. For example, accuracy measures do not measure the percentage of items for which the algorithm can make predictions, how many of the items in the recommendation list are not known to the users, how many of them are unexpected and interested items, diversity of the items in the recommendation list, how much uncertainty is involved in the predictions etc. We can not measure all these aspects by a single evaluation measure. We need a different evaluation metric to measure each of these aspects. Therefore, besides accuracy, several other measure like coverage, novelty, unexpectedness, serendipity, diversity and confidence were proposed to judge the quality of recommendations from different perspectives [13].

Among them confidence is very important which measures the uncertainty of the predictions or recommendations made by the algorithm. There are several factors that leads to uncertainty in recommender systems:

1. Data: The first and foremost reason is sparse data. If the data is sparse, we do not have enough information to make predictions or recommendations. It is difficult to make accurate predictions or recommendations for users who have rated very few items. Similarly, the predictions made for the items with many ratings are more accurate than the items with few ratings. Therefore, the more ratings we have in the data set, the more confident the algorithm is in making its recommendations or predictions [9]. Second reason is noisy data. Users will rate same items differently at different times. Therefore user's given rating can be considered as the noisy evidence of the user's true rating [4].

Another reason is data from untrusted users.

2. Change of user preferences over time: In recommender systems, user preferences may vary dynamically. As users tend to explore more and more items, their tastes change over time accordingly [5]. User preferences will also be changed depending on his current situation like mood, location, season, weather condition etc.

3. Algorithmic parameters and Model selection: Choice of different parameter values in the algorithm, suitability of the algorithm for the given data, assumptions used in the model and appropriateness of model to the given data are also the reasons which causes uncertainty.

Several approaches have been proposed to measure the uncertainty involved in predictions or recommendations made by the recommendation algorithms. One of the most common methods which can be used to estimate the confidence in machine learning algorithms is probability. As most of the recommendation algorithms use machine learning techniques, we can use probability as a confidence measure in recommender systems. Besides probability, several other confidence estimation approaches based on number of ratings [9], belief distributions [8], rating variance [1], percentage of correct predictions to compare two confidence estimation algorithms [13], binary classification problem with full probability distribution [6] and resampling [7] have been proposed in the recommender systems literature. Some of the them are non-personalized [9; 1] and some of them are applicable only to specific algorithms and are not generalized [8; 6]. Furthermore, none of the above algorithms provide guarantees on the error rate of the predictions, where error rate is the probability of excluding the correct class label and there is no possibility of controlling the erroneous predictions.

Conformal prediction (CP) [12; 14; 2] is a framework used to provide confidence values to individual predictions made by the machine learning algorithms. CP can be generalized to any algorithm and can produce personalized recommendations with guaranteed error rate. Furthermore, with CP we can control the number of erroneous predictions by varying the significance level, thus making it suitable to different kinds of applications. This motivated us to develop conformal prediction based confidence estimation algorithms for making the reliable predictions and recommendations in recommender systems.

3 Thesis Contributions and Chapter Organization

This dissertation focuses on developing the collaborative filtering recommendation algorithms based on conformal prediction to associate confidence values to predictions and recommendations produced by the collaborative filtering algorithms.

Chapter 1 is the **Introduction**, where we give a brief introduction of recommender systems and their tasks and discuss the reasons that motivated us to develop conformal prediction based recommendation algorithms to estimate the confidence of predictions and recommendations. We end up this chapter with contributions and an outline of the thesis.

Chapter 2 introduces fundamental concepts related to recommender systems and their evaluation. We first give a brief overview of recommender systems and their tasks and different kinds of data on which they work. We then review the most popular recommendation algorithms in the literature, namely collaborative filtering algorithms (especially, neighborhood-based algorithms and matrix factorization algorithm) which are solely based on the rating information to make predictions and recommendations. Finally, we discuss different approaches and measures used to evaluate collaborative filtering based recommender systems along with the data sets used in our experiments throughout this thesis.

Chapter 3 Task-based Evaluation of Neighborhood-Based collaborative filtering Algorithms with Multiple Feedback Types The performance of the neighborhood-based algorithms vary depending on the similarity measures used to find the similarity between two entities, prediction techniques used to predict the ratings or scores, data set used in evaluation and type of feedback in the data set. Therefore, we present an empirical analysis on neighborhood-based algorithms with different similarity measures, prediction techniques and data sets with different types of feedback. In this chapter, we discuss the following points:

- How does the different neighborhood-based algorithms proposed in the literature work for different feedback data sets?
- What similarity measures are available for each of these algorithms and for different feedback data sets?
- How to predict the rating or score for different feedback data sets?

- How the prediction and recommendation ability of the neighborhood-based algorithms are measured for different kinds of feedback?
- What Significance tests are used to determine whether the performance differences of different algorithms are statistically significant?
- Which algorithm best suits for prediction and recommendation tasks and why?
- Why unary feedback data sets perform better than numerical and binary for recommendation task?
- Analyze the performance differences between the different prediction techniques and similarity measures of different algorithms for different tasks, different kinds of feedback and different data sets.

Chapter 4 Prediction with confidence in Item-based Collaborative Filtering In this chapter, first, we explain the importance of confidence values in recommender systems, how these confidence values will be practically shown to the users and how the users' decisions will be affected by that. We also review different methods proposed in the literature to estimate the confidence of collaborative filtering algorithms. Then, we introduce conformal prediction which is a generalized framework used in machine learning to estimate the confidence values of individual predictions produced by any machine learning algorithm. We also discuss variants of CP along with the validity and efficiency measures used to determine the quality of predictions made by the CP algorithms. Then, we propose an algorithm based on CP to associate confidence values to the predictions made by the neighborhood-based algorithms. We do not apply CP to user-based collaborative filtering algorithm as it is very difficult to find the required number of neighbors belonging to each class. This problem arises in user-based algorithm for both numerical and binary data sets. This same problem arises when CP is applied on top of item-based collaborative filtering algorithm on numerical data sets. But for binary data sets, it is not a problem to find the required number of neighbors for each class in item-based algorithm in many cases. Therefore, we have used binary feedback data sets while applying CP on top of item-based collaborative filtering algorithm. We define different nonconformity measures based on similarity values between the items and then we empirically determine the best nonconformity measure.

Chapter 5 Prediction with confidence in Matrix factorization Algorithm In this chapter, we have shown different ways of applying conformal prediction to matrix factorization algorithm. We also define various nonconformity measures based on the

deviation between true and predicted values and analyzed the results and found that even the CP algorithm with the best nonconformity measure among the proposed nonconformity measures is not producing efficient prediction regions at higher confidence levels due to very close possible rating values in numerical or rating data sets. Therefore, to improve the efficiency we applied CP on binary data sets and got better results compared to the numerical data sets.

The application of conformal prediction to recommender systems is very different from its application to machine learning as the characteristics of data sets used in machine learning and recommender systems are different. Therefore, we also discussed major differences between machine learning algorithms and recommendation algorithms, how to adapt CP to different recommendation algorithms and what constitutes an example, features and labels in recommender systems in chapters 4 and 5.

Chapter 6 Recommendation with confidence in Neighbourhood-based Algorithms Conformal prediction is originally designed to associate confidence values to each prediction. In addition to prediction task, recommendation task is equally important in recommender systems. Attaching confidence values to a set of recommended items gives the relevance of the recommended items to user tastes. In this chapter, we discuss the following queries regarding the application of conformal prediction for recommendation task:

- How conformal prediction can be adapted to recommendation task in neighborhood-based algorithms?
- How error rate is redefined in recommendation scenario?
- How measures of machine learning can be used to measure validity and efficiency of recommendation lists produced at the given confidence levels?
- How to find confidence values for the *top-n* recommendation list for each user?
- What is the trade-off between precision and number of recommended items at different significance levels?
- We have also shown the performance differences between two neighborhood-based conformal recommendation algorithms, i.e., *user-based* and *item-based*.

Chapter 7 concludes our thesis by summarizing the contributions along with some directions for future work.

4 Publications of the Thesis

1. Tadiparthi V.R. Himabindu and Vineet Padmanabhan and Arun K. Pujari. "Estimating Confidence for Top-N Recommendations in Neighbourhood-based Algorithms". Yet to be submitted.

2. Tadiparthi V.R. Himabindu, Vineet Padmanabhan and Arun K. Pujari. "Conformal matrix factorization based recommender system", Information Sciences, pp. 1-23, 2018. In Press.

3. Tadiparthi V. R. Himabindu, Vineet Padmanabhan, Arun K. Pujari, and Abdul Sattar. "Prediction with Confidence in Item Based Collaborative Filtering". Pacific Rim International Conference on Artificial Intelligence, pp. 125-138. Springer, 2016.

REFERENCES

[1] Gediminas Adomavicius, Sreeharsha Kamireddy, and YoungOk Kwon. Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance. In *Workshop on Information Technology and Systems*, 2007.

[2] Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2014.

[3] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *Transactions on Information Systems*, 22(1):5–53, 2004.

[4] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 194–201, 1995.

[5] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[6] Yehuda Koren and Joe Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In *ACM Conference on Recommender Systems*, pages 117–124, 2011.

[7] Maciej A. Mazurowski. Estimating confidence of individual rating predictions in collaborative filtering recommender systems. *Expert Systems with Applications*, 40(10):3847–3857, 2013.

[8] Matthew R. McLaughlin and Jonathan L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 329–336, 2004.

[9] Sean M. McNee, Shyong K. Lam, Catherine Guetzlaff, Joseph A. Konstan, and John Riedl. Confidence displays and training in recommender systems. In *Human-Computer Interaction*, 2003.

[10] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Computer Supported Cooperative Work*, pages 175–186, 1994.

[11] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web Conference*, pages 285–295, 2001.

[12] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008.

[13] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. 2011.

[14] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

Prediction and Recommendation with Confidence in Collaborative Filtering Recommender Systems

ORIGINALITY REPORT

34%	12%	31%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Tadiparthi V.R. Himabindu, Vineet Padmanabhan, Arun K. Pujari. "Conformal matrix factorization based recommender system", Information Sciences, 2018 Publication	15%
2	"PRICAI 2016: Trends in Artificial Intelligence", Springer Nature, 2016 Publication	6%
3	link.springer.com Internet Source	<1%
4	net.pku.edu.cn Internet Source	<1%
5	repositorio.uam.es Internet Source	<1%
6	www.plantcell.org Internet Source	<1%
7	www-users.cs.umn.edu Internet Source	<1%