Bias Disparity in Recommendation Systems

A Thesis

submitted to the designated
by the General Assembly of Special Composition
of the Department of Computer Science and Engineering
Examination Committee

by

Virginia Tsintzou

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

WITH SPECIALIZATION
IN SOFTWARE

University of Ioannina
October 2018

Examining Committee:

- **Panayiotis Tsaparas**, Associate Professor, Department of Computer Science & Engineering, University of Ioannina (Supervisor)
- **Nikos Mamoulis**, Associate Professor, Department of Computer Science & Engineering, University of Ioannina
- **Evaggelia Pitoura**, Professor, Department of Computer Science & Engineering, University of Ioannina

Table of Contents

List of Figures

List of Tables

Abstract

Virginia Tsintzou, M.Sc. in Computer Science, Department of Computer Science and Engineering, University of Ioannina, Greece, October 2018.
Bias Disparity in Recommendation Systems.
Advisor: Panayiotis Tsaparas, Associate Professor.

Recommender systems have been applied successfully in a number of different domains, such as, entertainment, commerce, and employment. Their success lies in their ability to exploit the collective behavior of users in order to deliver highly targeted, personalized recommendations. Given that recommenders learn from user preferences, they incorporate different *biases* that users exhibit in the input data. More importantly, there are cases where recommenders may amplify such biases, leading to the phenomenon of *bias disparity*. Amplifying bias for different groups of users can lead to isolating sensitive groups or indirect discrimination.

The goal of this thesis is to study bias disparity in recommender systems. To this end, we define metrics for bias and bias disparity for recommendation systems. Then, we consider variants of the K-Nearest Neighbors recommendation algorithms, and we perform a systematic analysis of their behavior using synthetic data. The goal is to understand the conditions under which those algorithms exhibit bias disparity, and the long-term effect of recommendations on data bias. We observe that even moderate amount of bias, and small biased groups can lead to significant bias amplification. Using the Movielens dataset, we also present cases of real data where bias is observed and confirm bias disparity on recommendations.

To address the problem of bias disparity, two algorithms that post-process recommendations are considered. The algorithms re-rank the results of any recommendation algorithm in order to produce new sets of recommendations where bias disparity is eliminated. Each bias correcting algorithm aims at providing useful recommendations by targeting the utility of the user group or the least satisfied user in the group.

We conclude that correcting bias in recommendations slows down the polarization of users in the long-term.

Εκτεταμενη Περιληψη

Βιργινία Τσίντζου, Μ.Δ.Ε. στην Πληροφορική, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Οκτώβριος 2018.
Προκατάληψη σε Συστήματα Συστάσεων.
Επιβλέπων: Παναγιώτης Τσαπάρας, Αναπληρωτής Καθηγητής.

Τα συστήματα συστάσεων έχουν μεγάλο πλήθος εφαρμογών, που συχνά βρίσκονται στο διαδίκτυο. Για παράδειγμα, ηλεκτρονικά καταστήματα τα χρησιμοποιούν για να προτείνουν προϊόντα στους χρήστες, μέσα κοινωνικής δικτύωσης προτείνουν τη σύνδεση με άλλους χρήστες, μηχανές αναζήτησης εργασίας προτείνουν θέσεις εργασίας σχετικές με τους χρήστες, κ.ά. Η χρησιμότητα και η αποτελεσματικότητα των αλγορίθμων συστάσεων οφείλεται στο γεγονός ότι βασίζονται στις ιδιαίτερες προτιμήσεις των χρηστών που συλλέγονται από τις αλληλεπιδράσεις τους με κάποιο σύστημα. Τα μοντέλα συστάσεων ενσωματώνουν τις προτιμήσεις αυτές και παράγουν πιο στοχευμένες προσωπικές συστάσεις. Οι επιλογές των χρηστών συχνά χαρακτηρίζονται από διάφορες προκαταλήψεις. Η εκπαίδευση ενός μοντέλου σε δεδομένα που χαρακτηρίζονται από τη μεροληψία των χρηστών, οδηγεί στην αναπαραγωγή και αύξηση της προκατάληψης στις συστάσεις. Η αύξηση της προκατάληψης στις συστάσεις σε συγκεκριμένες ομάδες χρηστών μπορεί να οδηγήσει στην αναπαραγωγή στερεοτύπων, διακρίσεις και απομόνωση ευαίσθητων κοινωνικών ομάδων.

Αυτή η εργασία έχει στόχο να μελετήσει την ανισότητα της προκατάληψης μεταξύ των δεδομένων προτιμήσεων των χρηστών και των συστάσεων των αλγορίθμων. Ορίζουμε μετρικές για την προκατάληψη και την ανισότητα της προκατάληψης των συστημάτων συστάσεων. Επιλέγουμε τέσσερα διαφορετικά μοντέλα συστάσεων που είναι διάφορες παραλλαγές του αλγορίθμου Κοντινότεροι Γείτονες και δημιουργώντας συνθετικά δεδομένα, παρατηρούμε τη συμπεριφορά των αλγορίθμων συστάσεων σε διάφορες περιπτώσεις με στόχο να κατανοήσουμε ποιες

συνθήκες προκαλούν αύξηση της προκατάληψης στις συστάσεις. Επιπλέον, μελετάμε την μακροπρόθεσμη επίδραση των μοντέλων ενσωματώνοντας τις συστάσεις στα δεδομένα. Διαπιστώνουμε ότι ακόμη και στις περιπτώσεις που τα δεδομένα προτιμήσεων των χρηστών χαρακτηρίζονται από μέτρια προκατάληψη ή υπάρχουν μικρές ομάδες χρηστών με προκατάληψη, τότε οι συστάσεις μπορεί να οδηγηθούν σε σημαντική αύξηση της προκατάληψης στα αποτελέσματα. Παρουσιάζουμε περιπτώσεις πραγματικών δεδομένων με προκατάληψη χρησιμοποιώντας το σύνολο δεδομένων Movielens, οι οποίες επιβεβαιώνουν τα ευρήματα.

Τέλος, για να αντιμετωπιστεί το πρόβλημα της ανισότητας της προκατάληψης στις συστάσεις, προτείνουμε δύο αλγορίθμους επεξεργασίας των συστάσεων από οποιοδήποτε υπάρχον μοντέλο. Οι αλγόριθμοι αναδιατάσσουν τα αποτελέσματα του αλγορίθμου συστάσεων και παράγουν νέα σύνολα συστάσεων που διατηρούν σταθερή την προκατάληψη των δεδομένων εισόδου. Οι αλγόριθμοι διόρθωσης παράλληλα στοχεύουν να επιστρέφουν αξιόλογες συστάσεις που έχουν υψηλή χρησιμότητα, η οποία ορίζεται είτε για το σύνολο μιας ομάδας χρηστών, είτε ατομικά για όλους τους χρήστες. Ο πρώτος αλγόριθμος ονομάζεται GULM και στοχεύει οι νέες συστάσεις να έχουν χαμηλή απώλεια χρησιμότητας σε σχέση με τις αρχικές συστάσεις, κατά μέσο όρο, για όλη την ομάδα χρηστών. Ο δεύτερος αλγόριθμος ονομάζεται MULM και παράγει νέες συστάσεις οι οποίες δεν αδικούν κάποιο χρήστη και ελαχιστοποιούν τη μέγιστη απώλεια χρήστη, δηλαδή την απώλεια χρησιμότητας στο νέο σύνολο συστάσεων για το χρήστη σε σχέση με το αρχικό. Εξετάζουμε τη συμπεριφορά των αλγορίθμων διόρθωσης και επιβεβαιώνουμε την αποδοτικότητά τους.

Chapter 1

Introduction

Decisions based on algorithmic results is a common phenomenon. However, blindly relying on algorithms has proven to be risky, because results are not always objective. In cases where a computer system reflects human values, we say that *algorithmic bias* occurs. For example, algorithms that assess the risk of a prisoner in the USA to reoffend, have been reported to produce biased results ([1], [2]). Blacks were 77% more likely to be classified as higher risk of commiting a future crime, while whites where more often classified as lower risk.

Biases have also occured in *recommendation systems*. Recommender systems have found applications in a wide range of domains, including e-commerce, entertainment, social media, news portals, and employment sites [3]. They are used for recommending products in shopping portals, entertainment content in video or music portals, information in content portals, job opportunities in employment portals. They have been proven to be extremely effective in predicting the preferences of the users, filtering the available content to provide a highly personalized and targeted experience. One of the most popular classes of recommendation systems is collaborative filtering. Collaborative Filtering (CF) uses the collective behavior of all users over all items to infer the preferences of individual users for specific items [3]. However, given the reliance of CF algorithms on the input preferences, they are susceptible to biases that may appear in the input data.

In this work, we consider biases with respect to the preferences of specific groups of users (e.g., men and women) towards specific categories of items (e.g., different movie

genres). For example, a case of gender discrimination in recommendations occured at the social platform Linkedin [4]. Following the search of the usually female name "Andrea", appeared the suggestion of the alternative search for "Andrew" which is commonly a male name. However, the oppositive suggestion for the search "Andrew" did not appear because of the bias in the input data used to train the recommendation model.

Bias in recommendations is not necessarily always problematic. For example, it is natural to expect gender bias when recommending clothes. However, gender bias is undesirable when recommending job postings, or information content. Furthermore, we want to avoid the case where the recommender system introduces bias in the data, by amplifying existing biases and reinforcing stereotypes. We refer to this phenomenon, where input and recommendation bias differ, as *bias disparity*.

The goal of the thesis is to understand the emergence of bias disparity under different conditions, and propose algorithms that correct it. We consider data that include implicit feedback and recommendation algorithms that are alternative approaches of K-Nearest Neighbors algorithm.

More specifically:

- We define notions of bias and bias disparity for recommender systems. Our definitions capture the increase in probability of a user to prefer items from specific categories.
- Using synthetic data we study different conditions under which bias disparity may appear. We observe that even moderate amount of bias, and small biased groups can lead to significant bias amplification. We also consider the effect of the iterative application of recommendation algorithms on the bias of the data.
- Using the MovieLens[1] dataset, we study cases where bias and bias amplification appears in real data.
- We consider two algorithms that re-rank the results of the recommenders for correcting bias disparity and study them experimentally.

The rest of the thesis is structured as follows. In Chapter 2, we review work related to recommendation systems, bias in recommendations and classification and fairness to users. In Chapter 3, we provide the main definitions of bias and bias

[1]MovieLens 1M: https://grouplens.org/datasets/movielens/1m/

disparity for recommendations, and describe the recommendation algorithms we use in our experiments. We present experiments on the recommendation algorithms in Chapter 4. In Chapter 5, we describe two algorithms that process recommendations and correct bias disparity and in Chapter 6, we review our work and provide our conclusions.

Chapter 2

Related Work

The problem of algorithmic bias, and its flip side, fairness in algorithms, has attracted considerable attention in the recent years [5, 6]. In [7] they explore the risks of data-driven algorithms as well as the requirements to ensure fairness. There is work related to different applications, such as online search engines([8], [9]) or text mining ([10]). Most existing work focuses on classification systems ([11], [12], [13], [14]), while there is limited work on recommendation systems. One type of recommendation bias that has been considered in the literature is popularity bias [15]. It has been observed that under some conditions popular items are more likely to be recommended leading to a rich get richer effect, and there are some attempts to correct this bias ([16], [17]). Related to this is also the quest for diversity [18], where the goal is to include different types of items in the recommendations.

These notions of fairness do not take into account the presence of different (protected) groups of users and different item categories that we consider in this work. In [19] they assume different groups of users and items, they define two types of bias and they propose a modification of the recommendation algorithm in [20] to ensure a fair output. Their work focuses on fairness, rather than bias disparity, and works with a specific algorithm. The notion of bias disparity is examined in [21] but in a classification setting. In [22], they mention different causes of unfairness in classification and they focus on indirect prejudice which they define as the statistical dependence between a sensitive feature, such as gender, and a target variable of the classifier, such as income.

Fairness in terms of correcting rating errors for specific groups of users was studied in [23] for a matrix factorization CF recommender. They use different measures of unfairness, such as the value of inconsistency in estimation error between an advantaged and a disadvantaged group of users. They focus on decreasing those inconsistencies of the predictions between the groups and treating them the same way. However, this does not guarantee that the recommendations will not become more biased, since recommendations that are equally biased are considered as fair.

The issue of fairness and bias is also relevant to ranking. In [24], they point out that ranking positions influence the amount of attention the ranked subjects receive and that position bias in rankings can lead to unfair distribution of opportunities such as jobs. They suggest that improving equity of attention that users give to equally relevant subjects is important and they propose a solution by swapping positions among equally relevant subjects while retaining high ranking quality. Fairness in ranking is also related to protected groups. In [25], they raise the issue of reducing biases in the representation of an under-represented group along a ranked list. They propose an algorithm for selecting a number of protected candidates over a proportion threshold while maximizing utility. That is, either selecting most qualified candidates or ranking with descending qualifications.

Application of recommenders has significant impact on user preferences in the long-term. Feeding the recommendation systems with data of user preferences affects the recommendations and consequently the future selections of the users. This feedback loop between user behavior and recommendations is examined in [26] where they suggest that this loop creates confounded data and causes homogenization of users, which is amplified every time the data go through the loop, without gaining utility. They measure and report their findings on the impact of algorithmic confounding on a range of recommendation systems using synthetic data.

In this work, we examine fairness in recommendation systems in cases that protected groups may be affected. We aim at measuring bias in the data and defining the conditions under which this is exaggerated, either in the short-term or in the long-term, and propose ways of mitigating this exaggeration.

Chapter 3

Problem Definition

3.1 Definitions

We consider a set of n users $\mathcal{U}$ and a set of m items $\mathcal{I}$. We are given an associations $n \times m$ matrix S, where $S(u, i) = 1$ if user u has selected item i, and zero otherwise. Selection may mean that user u liked post i, or that u purchased product i, or that u watched video i. These observations are also known as "implicit feedback data", where we do not have explicit ratings by the users on how much they liked (or not) an item, but rather only binary information on whether they selected the item or not. There are several applications where this kind of data are generated, and there is a need for generating recommendations from such data.

We assume that users are associated with an attribute A_U, e.g., the gender of the user. The attribute A_U partitions the users into *groups*, that is, subsets of users with the same attribute value, e.g., men and women. We will typically assume that we have two groups and one of the groups is the *protected group*. Similarly, we assume that items are associated with an attribute A_I, e.g., the genre of a movie, which partitions the items into *categories*, that is, subsets of items with the same attribute value, e.g., action and romance movies.

Given the association matrix S, we define the input *preference ratio* $PR_S(G, C)$ of

group G for category C as the fraction of selections from group G that are in category C. Formally:

$$PR_S(G,C) = \frac{\sum_{u\in G}\sum_{i\in C} S(u,i)}{\sum_{u\in G}\sum_{i\in \mathcal{I}} S(u,i)} \tag{3.1}$$

This is essentially the conditional probability that a selection is in category C given that it comes from a user in group G.

To assess the importance of this probability we compare it against the probability $P(C) = |C|/m$ of selecting from category C when selecting uniformly at random. We define the *bias* $B_S(G,C)$ of group G for category C as:

$$B_S(G,C) = \frac{PR_S(G,C)}{P(C)} \tag{3.2}$$

Bias values less than 1 denote *negative bias*, that is, group G on average tends to select less often from category C, while bias values greater than 1 denote *positive bias*, that is, group G favors category C disproportionately to its size.

Defining bias and deciding when a group of users is biased in favor of an item category, is complicated. Even though Eq. (3.2) takes into account the size of item categories, it ignores the size of user groups. Including the size of groups in the bias definition would allow us to compare the preference of different groups towards the same item category. In this work, we are interested in examining bias as the preference of a group towards an item category over other categories. Alternative definitions of bias are left for future work.

Assume that the recommendation algorithm outputs for each user u a ranked list of r items R_u. In our work we view the recommendation list as a set of r elements. The collection of all recommendations can be represented as an associations matrix R, where $R(u,i) = 1$ if item i is recommended for user u and zero otherwise. Given the matrix R, we can compute the output preference ratio of the recommendation algorithm, $PR_R(G,C)$, of group G for category C using Eq. (3.1), and the output bias $B_R(G,C)$ of group G for category C.

To compare the bias of a group G for a category C in the input data S and the recommendations R, we define the *bias disparity*, that is, the relative change of the bias value.

$$BD(G,C) = \frac{B_R(G,C) - B_S(G,C)}{B_S(G,C)} \tag{3.3}$$

Our definitions of preference ratios and bias are motivated by concepts of group proportionality, and group fairness considered in the literature [5, 6].

3.2 The Recommendation Algorithms

We consider the top-r recommendation problem, where we wish to determine which are the r most suitable items to recommend to a particular user. We use neighborhood-based collaborative filtering algorithms, and specifically, different variants of K-Nearest-Neighbors (KNN) algorithm adapted on implicit data. The notion of neighborhoods will be defined for either users or items. Therefore, the algorithms are divided in $user-based$ and $item-based$ according to the axis along which we consider neighbors, and similar entities. In the case of user-based algorithms, predictions are derived from the preferences of the users that are the most similar to the particular user for whom we produce recommendations. Respectively, in item-based algorithms, the user's own preference on items that are the most similar to a particular item contributes to the prediction for that item.

The algorithms use the preferences of users in the neighborhood or the association between the user and items in the neighborhood to compute a *utility value*. This value indicates how suitable is an item for a user, and therefore the items with the highest utility are recommended to the user.

For similarity, we use the $Jaccard$ similarity, *JSim*, computed using the matrix S. $Jaccard$ similarity between two sets of users or items I and J ,is the size of the intersection of I and J, divided by the size of their union.

$$JSim = \frac{|I \cap J|}{|I \cup J|} \tag{3.4}$$

In the selection of neighbors, ties of similarity may occur. Ties also happen at the selection of the top-r items for recommendation based on their utility values. The algorithms handle all cases of ties by selecting randomly.

3.2.1 User-based algorithms

We use two user-based algorithms that form neighborhoods of users that are similar to the user for whom we want to find the top-r recommendations. The UserKNN algorithm selects for each user the K most similar users, over all users. On the other hand, users that have selected the item we want to predict, are considered more relevant to the context of the particular prediction. Hence, we use the *context* algorithm, CUserKNN. In CUserKNN, neighbors are users that are similar to user u and have already selected the item i for which we want to calculate a utility value

for u.

As we see in Algorithm 3.1, the UserKNN first computes for each user, u, the set $N_K(u)$ of the K most similar users to u. For user u and item i not selected by u, it computes a *utility value*

$$V(u,i) = \frac{\sum_{n \in N_K(u)} \mathit{JSim}(u,n) S(n,i)}{\sum_{n \in N_K(u)} \mathit{JSim}(u,n)} \tag{3.5}$$

The utility value $V(u,i)$ is the fraction of the similarity scores of the top-K most similar users to u that have selected item i.

We expect UserKNN to define neighborhoods of users that are biased in favor of the same item categories.

In CUserKNN (Algorithm 3.2), the K nearest neighbors are selected from the pool of neighbors of u that have selected item i. To calculate the utility value $V(u,i)$ for a user u and an item i not selected by u, the algorithm finds the set of neighbors $N_K(u)$ and sums their similarities to u. So the utility value is computed as follows.

$$V(u,i) = \sum_{n \in N_K(u)} \mathit{JSim}(u,n) \tag{3.6}$$

The fact that the utility values are calculated without being normalized and are based on the similarity values between user u and the neighbors that selected item i, implies that it is easier for CUserKNN than the UserKNN to distinguish among items.

3.2.2 Item-based algorithms

The item-based algorithms form neighborhoods of items and recommend items that are similar to those that users have selected. The ItemKNN selects for each item the K most similar items, over all items, while the CItemKNN selects neighbors of the items from the pool of items that each user has selected. The algorithms compute utility values for every user u and every item i not selected by u.

Specifically, ItemKNN (Algorithm 3.3) computes for each item, i, the set $N_K(i)$ of the K most similar items to i. For user u and item i not selected by u, the algorithm computes the utility value

$$V(u,i) = \frac{\sum_{n \in N_K(i)} \mathit{JSim}(i,n) S(u,n)}{\sum_{n \in N_K(i)} \mathit{JSim}(i,n)} \tag{3.7}$$

Algorithm 3.1 UserKNN

Input: K: number of nearest neighbors, r: number of recommendations,
S: associations matrix

Output: V: utility values matrix, R: recommendations matrix

1: **for** each user x **do**
2: **for** each user $y \neq x$ **do**
3: $I(x) \leftarrow$ items i for which $S(x, i) = 1$
4: $I(y) \leftarrow$ items i for which $S(y, i) = 1$
5: calculate Jaccard similarity

$$JSim(x, y) \leftarrow \frac{|I(x) \cap I(y)|}{|I(x) \cup I(y)|}$$

6: **end for**
7: **end for**
8: **for** each user u **do**
9: **for** each item $i \notin S(u)$ **do**
10: $N_K(u) \leftarrow$ find K most similar users to u.
11: calculate utility values

$$V(u, i) \leftarrow \frac{\sum\limits_{n \in N_K(u)} JSim(u, n) \cdot S(n, i)}{\sum\limits_{n \in N_K(u)} JSim(u, n)}$$

12: **end for**
13: **end for**
14: $R \leftarrow$ recommend r items to each user with highest utility values

The utility value $V(u, i)$ is the fraction of the similarity scores of the top-K most similar items to i that user u has selected.

As mentioned in [27], item-based algorithms are enabled to provide more relevant recommendations since the recommendations occur based on each user's own ratings. We expect that this apllies also in our case of implicit feedback. Generally, ItemKNN enables items in the long tail to receive relatively higher utility values than in UserKNN. It is more possible that items in the long tail have neighbors that also have less ratings and belong in the long tail. This occurs due to the $Jaccard$ similarity calculation. If an item i in the long tail was rated by a number of common users with

another item j, then the size of the union of users that rated both items is smaller if j also comes from the long tail. That leads to higher similarity values among these items.

In Algorithm 3.4 we see that CItemKNN follows CUserKNN in the selection of neighbors. That is, for each user u and each item i not selected by u, it computes the set $N_K(i)$ of the K closest items to i that were selected by u consist the set of neighbors $N_K(i)$.

It then calculates the utility value

$$V(u,i) = \sum_{n \in N_K(i)} JSim(i,n) \tag{3.8}$$

which is the sum of similarities of item i with its neighbors.

Clearly, the value of K cannot be larger than the number of ratings of user u. Also, if K is equal to the number of selections of u in S, then the ratio PR of neighbors for an item i is equal to the preference ratio of the user $PR_S(u)$. This way, the item we predict, has more neighbors from the category of the user. In this case we expect that the algorithm enables items from the category of the user for recommendation.

Algorithm 3.2 CUserKNN

Input: K: number of nearest neighbors, r: number of recommendations,
S: associations matrix

Output: V: utility values matrix, R: recommendations matrix

1: **for** each user x **do**
2: **for** each user $y \neq x$ **do**
3: $I(x) \leftarrow$ items i for which $S(x, i) = 1$
4: $I(y) \leftarrow$ items i for which $S(y, i) = 1$
5: calculate Jaccard similarity

$$JSim(x, y) \leftarrow \frac{|I(x) \cap I(y)|}{|I(x) \cup I(y)|}$$

6: **end for**
7: **end for**
8: **for** each user u **do**
9: **for** each item $i \notin S(u)$ **do**
10: $N_K(u) \leftarrow$ find K most similar users to u that have selected item i.
$\forall n \in N_K(u) \implies S(n, i) = 1$
11: calculate utility values

$$V(u, i) \leftarrow \sum_{n \in N_K(u)} JSim(u, n)$$

12: **end for**
13: **end for**
14: $R \leftarrow$ recommend r items to each user with highest utility values

Algorithm 3.3 ITEMKNN

Input: K: number of nearest neighbors, r: number of recommendations,
S: associations matrix

Output: V: utility values matrix, R: recommendations matrix

1: **for** each item x **do**
2: **for** each item $y \neq x$ **do**
3: $U(x) \leftarrow$ users u for which $S(u, x) = 1$
4: $U(y) \leftarrow$ users u for which $S(u, y) = 1$
5: calculate Jaccard similarity

$$JSim(x, y) \leftarrow \frac{|U(x) \cap U(y)|}{|U(x) \cup U(y)|}$$

6: **end for**
7: **end for**
8: **for** each user u **do**
9: **for** each item $i \notin S(u)$ **do**
10: $N_K(i) \leftarrow$ find K most similar items to i.
11: calculate utility values

$$V(u, i) \leftarrow \frac{\sum\limits_{n \in N_K(i)} JSim(i, n) \cdot S(u, n)}{\sum\limits_{n \in N_K(i)} JSim(i, n)}$$

12: **end for**
13: **end for**
14: $R \leftarrow$ recommend r items to each user with highest utility values

Algorithm 3.4 CItemKNN

Input: K: number of nearest neighbors, r: number of recommendations,
S: associations matrix

Output: V: utility values matrix, R: recommendations matrix

1: **for** each item x **do**
2: **for** each item $y \neq x$ **do**
3: $U(x) \leftarrow$ users u for which $S(u, x) = 1$
4: $U(y) \leftarrow$ users u for which $S(u, y) = 1$
5: calculate Jaccard similarity

$$JSim(x, y) \leftarrow \frac{|U(x) \cap U(y)|}{|U(x) \cup U(y)|}$$

6: **end for**
7: **end for**
8: **for** each user u **do**
9: **for** each item $i \notin S(u)$ **do**
10: $N_K(i) \leftarrow$ find K most similar items to i that user u has selected.
$\forall n \in N_K(i) \implies S(u, n) = 1$
11: calculate utility values

$$V(u, i) \leftarrow \sum_{n \in N_K(i)} JSim(i, n)$$

12: **end for**
13: **end for**
14: $R \leftarrow$ recommend r items to each user with highest utility values

Chapter 4

Evaluating Bias Disparity

In this Chapter, we will evaluate experimentally bias amplification for recommendation algorithms we consider. To understand the conditions that lead to the emergence of bias amplification we will work with synthetic datasets with controlled degree of bias. We also measure bias and bias amplification on real data, using the MovieLens dataset.

4.1 Synthetic data generation

We create datasets of n users and m items. Each dataset consists of an $n \times m$ matrix S. All datasets are implicit feedback datasets, that is, the generated matrices are binary. A value $S[u, i] = 1$ denotes that user u has selected item i. Items are partitioned into two

categories C_1 and C_2 of size m_1 and m_2 respectively. Users consist one group G or in most cases they are split into two groups G_1 and G_2 of size n_1 and n_2 respectively. We assume that users in G_1 tend to favor items in category C_1, while users in group G_2 tend to favor items in category C_2. To quantify this preference, we give as input to the data generator two parameters ρ_1, ρ_2, where parameter ρ_i determines the preference ratio $PR_S(G_i, C_i)$ of group G_i for category C_i. For example, $\rho_1 = 0.7$ means that 70% of the ratings of group G_1 are in category C_1. In the case where we have one group of users, we assume they prefer category C_1 with preference ratio ρ.

The datasets we create consist of 1,000 users and 1,000 items. We assume that each user selects 5% of the items in expectation and we recommend $r = 10$ items per user. The presented results are average values of 10 experiments.

We perform sets of experiments that examine the role of the preference ratios, the group and category sizes, the number of K neighbors, and the number of recommendations r.

4.2 Single group

In this experiment, we have one user group G and two equal-size item categories C_1 and C_2. Users in G tend to favor C_1 items with preference ratio ρ, which takes values from 0.5 to 1, in increments of 0.05. In Figure 4.1, we plot the output preference ratio $PR_R(G, C_1)$ as a function of ρ for the four recommendation algorithms. Note that in this experiment, bias is the preference ratio scaled by a factor of two. We report preference ratios to be more interpretable. The dashed line shows when the output ratio is equal to the input ratio and thus there is no bias disparity. We consider different values for K, the number of neighbors. For CUserKNN and CItemKNN we use lower K values, since we select neighbors that we know are relevant to the prediction, unlike UserKNN and ItemKNN where neighbors are selected based on the highest similarities.

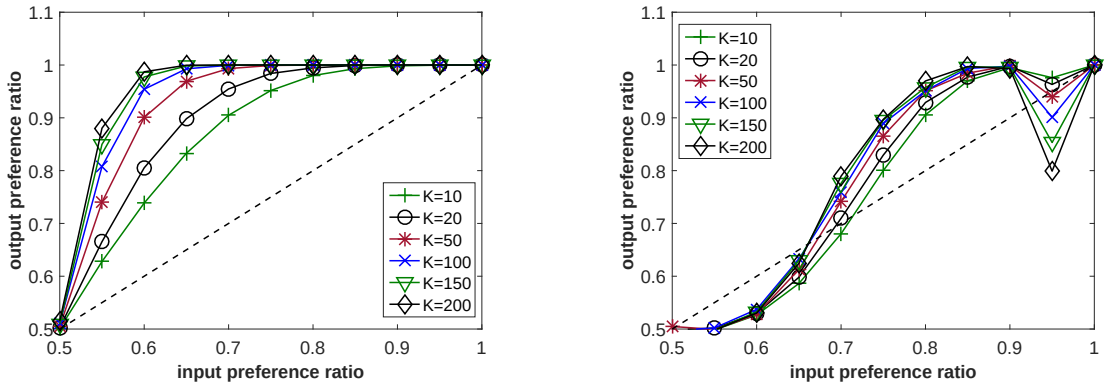
UserKNN (Figure 4.1a) amplifies bias as input bias increases. For large K, bias increases faster and for input $PR_S = 0.6$, recommendations become completely biased. Smaller K values amplify cause smaller bias disparity but also amplify the input bias.

In Figure 4.1b, we see that ItemKNN decreases bias for smaller input preference ratios. For $PR_s > 0.65$ bias is amplified. For larger K values when $PR_S = 0.95$, bias

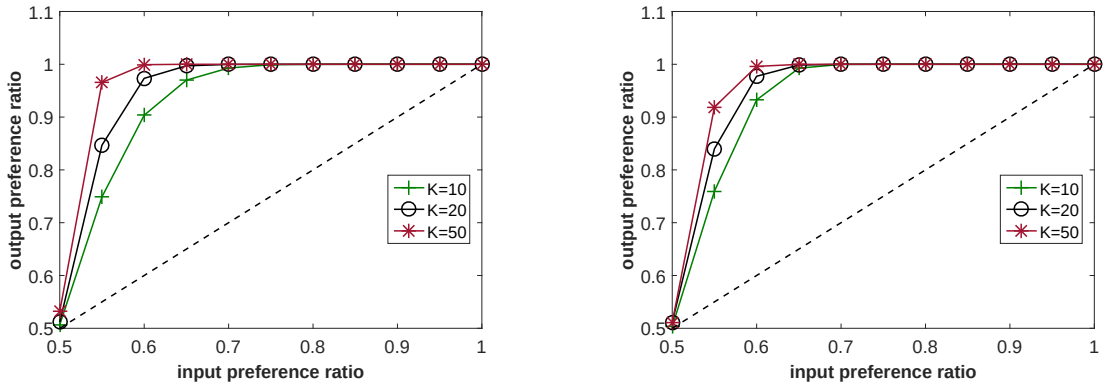
decreases significantly. Items in C_2 have higher similarity with C_1 items than with other C_2 items for high input bias. For $PR_S = 0.95$, the 95% of the total ratings are given to C_1 items, so category C_1 has stronger signal. It is more likely for a C_2 item to be selected by the same user at the same time with a C_1 item than another C_2 item. Since C_2 items have as neighbors mostly C_1 items that are highly preferred by the users, the utility values of C_2 items become higher than less biased cases, so C_2 is competitive to C_1 and is recommended more often. This explains the sudden drop of output bias at that point. However, for complete input bias, C_2 items have zero similarity with any item so it is not possible to be recommended.

We examine the results for CUserKNN and CItemKNN (Figures 4.1c & 4.1d) and even though they seem to have similar behavior, they have subtle but important differences. In both cases bias increases rapidly and reaches its peak for $PR_S \geq 0.6$ but CUserKNN with large K values is more prone to amplifying bias, since it provides almost completely biased recommendations for the least biased input ($PR_S = 0.55$). On the other hand, even though CItemKNN has less sharp increase of bias disparity for smaller bias input, it appears to depend less on the number of neighbors K.

Generally, ItemKNN is more resilient than the other algorithms and has the smaller bias disparity.



(a) UserKNN (b) ItemKNN



(c) CUserKNN (d) CItemKNN

Figure 4.1: Output preference ratio PR_R, single group case

4.3 Symmetric Preferences

For the following experiments we will have two groups of users, G_1 and G_2, 1000 in total, and 1000 items split into two categories. we assume that the two groups have equal size and the same preference ratios by setting $\rho_1 = \rho_2 = \rho$, where ρ takes values from 0.5 to 1, in increments of 0.05. Categories C_1 and C_2 have equal size also. We call this setting *symmetric*. In this section we perform experiments to evaluate the behavior of the recommenders on datasets with the symmetric setting.

4.3.1 Preference ratio in recommendations

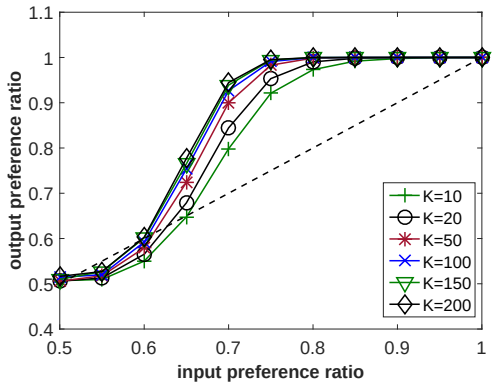
In Figure 4.2, we plot the output preference ratio of the recommenders $PR_R(G_1, C_1)$ (eq. $PR_R(G_2, C_2)$) as a function of ρ. A first observation for UserKNN (Figure 4.2a) is that when the input bias is small ($PR_S \leq 0.6$), the output bias decreases or stays the same. In this case, users have neighbors from both groups. For higher input bias ($PR_S > 0.6$), we have a sharp increase of the output bias, which reaches its peak for $PR_S = 0.8$. In these cases, the recommender polarizes the two groups, recommending items only from their favored category.

Increasing the value of K increases the output bias. Adding neighbors increases the strength of the signal, and the algorithm discriminates better between the items in the different categories.

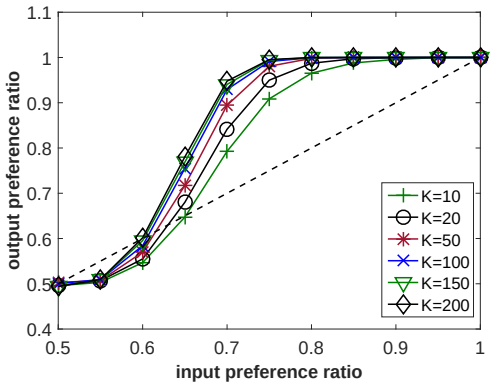
The ItemKNN (Figure 4.2b) is almost identical to UserKNN. This is reasonable because of the symmetry of the associations of the input datasets and the symmetry of the algorithms.

This applies to CUserKNN and CItemKNN also (Figures 4.2c & 4.2d). They exhibit almost identical behavior. For small input bias ($PR_S \leq 0.6$), the output bias once again decreases slightly or stays the same, while for $PR_S \geq 0.75$ recommendations are completely biased. The main difference that we notice between the primary and the context recommenders is that K has a different effect. The context algorithms are more resilient to the value of K and even for smaller K the bias amplification is sharp.

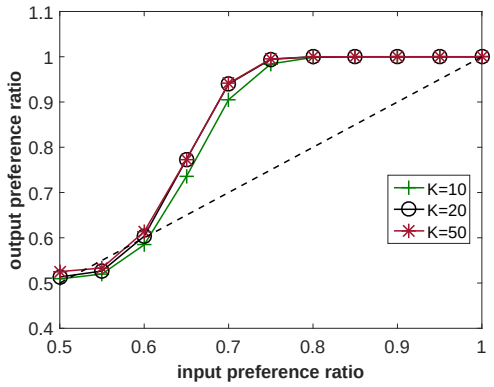
In Figure 4.3, we report the preference ratio for all candidate items for recommendation for each user in the UserKNN algorithm. These are the items that have non-zero utility according to our algorithm. We can think of this bias, as the bias in the case where r takes the maximum possible value. The plot of the candidate items



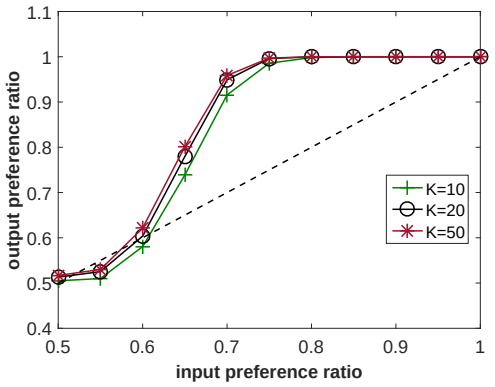
(a) UserKNN



(b) ItemKNN



(c) CUserKNN



(d) CItemKNN

Figure 4.2: PR_R, symmetric case

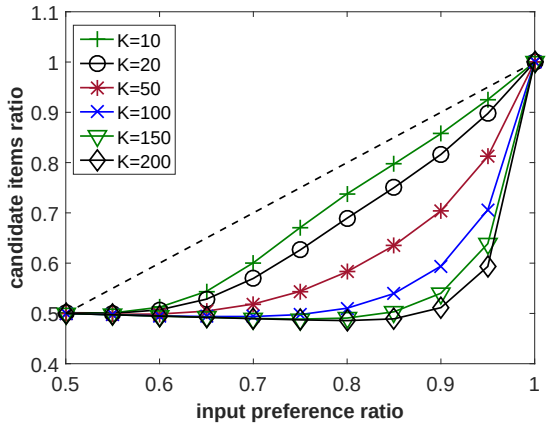


Figure 4.3: Ratio of candidate items in UserKNN, symmetric case

in ItemKNN is identical and we do not report it. Surprisingly, the candidate items are less biased even for high values of the input bias. This shows that (a) utility proportional to user or item similarity increases bias, (b) re-ranking may help in decreasing bias.

It is not useful to report the ratio of candidate items in CUserKNN and CItemKNN. In our datasets, the ratio of candidate items for these algorithms is always around 0.5, with the exception of the complete biased case ($PR_S(G_i, C_i) = 1$). In that case, items in one category C_i have zero similarity with items in the other category $\overline{C_i}$, so candidate items come only from each group's preferred category.

4.3.2 Prediction precision for K neighbors

As we have seen in our experiments, our algorithms are sensitive to the number of neighbors K that we consider. In order to understand what is the "correct" number of K, we use as a guide the performance of the recommender. We perform 5-fold cross validation on each dataset and measure the precision of the prediction of the four recommendation algorithms. We test UserKNN and ItemKNN for K up to 500. For the context algorithms CUserKNN and CItemKNN, we do not consider values of K larger than 50. That is, because they select as neighbors users that have rated an item and items that were rated by a user respectively, and users and items in our data have 50 ratings in expectation.

If R is the set of recommendations of an algorithm A for the $train$ set, and $test$ is the set of the true selections of users left out of the training, then we measure

precision of the algorithm as follows:

$$precision_A = \frac{1}{5} \cdot \sum_{fold=1}^{5} \frac{|R \cap test|}{|R|}$$

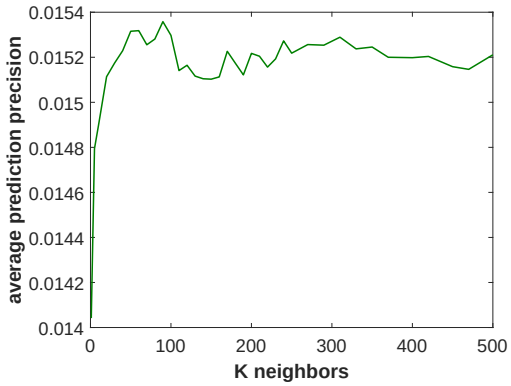
The plot in Figure 4.4 is the average precision over all datasets with varying input bias. UserKNN has higher precision in its predictions for K values close to 100. Even though ItemKNN appears to have better precision for $K \geq 200$, we want to avoid using extremely large K values because normalizing the utility values with the sum of similarities will lead to very small utility values and it will be difficult for the recommender to discriminate among items. CUserKNN and CItemKNN have better precision for very small K values. $K = 10$ is the preferred number of neighbors, since precision is good and smaller values would not allow diversity among neighbors.

Any item in our datasets has probability 5% to be selected by a user and each user selects items independently. Applying the UserKNN on the symmetric data, if we expect at least 5 users that are neighbors to user u to have selected the same item i, then we need K equal or larger than 100 because $K \cdot 5\% \geq 5 \implies K \geq 100$. It is interesting that for K values smaller than 40, the recommended items are probably suggested by only one user. This applies to ItemKNN also, due to the symmetry of the data and of the selection of neighbors between the two algorithms. That is why we will use $K = 100$ as the default value of K. For CUserKNN and CItemKNN we select $K = 10$ as the default value.

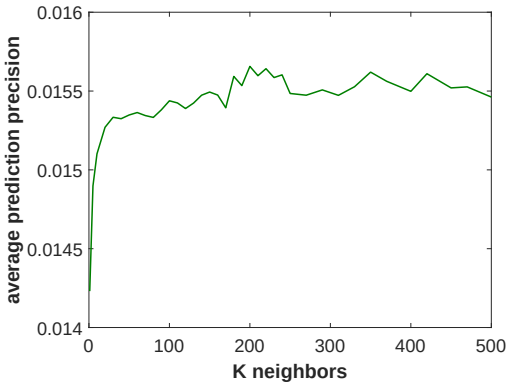
4.3.3 Impact of the number of recommendations

To understand the role of the number of recommendations r, we will measure the recommendation preference ratio for various values of r. We have G_1 and G_2 user groups of equal size and C_1 and C_2 item categories of also equal size. The input preference ratio for each group is $PR_S(G_i, C_i) = 0.7$. We plot the output preference ratio of the recommendation algorithms PR_R for values of r from 1 to 50. For UserKNN and ItemKNN we use $K = 100$ and for CUserKNN and CItemKNN we use $K = 10$ number of neighbors. As we see in Figure 4.5, for any r, recommendations of any recommender have amplified bias. The dashed line shows the input preference ratio and as long as the ratio of recommendations is above it, they have bias increased.

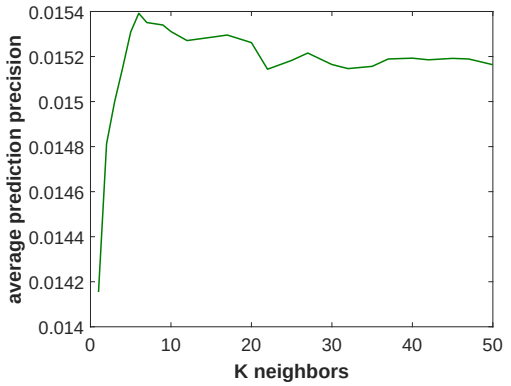
UserKNN and ItemKNN always provide more biased recommendations than the context recommenders. However, it is obvious that for biased input data, the bias



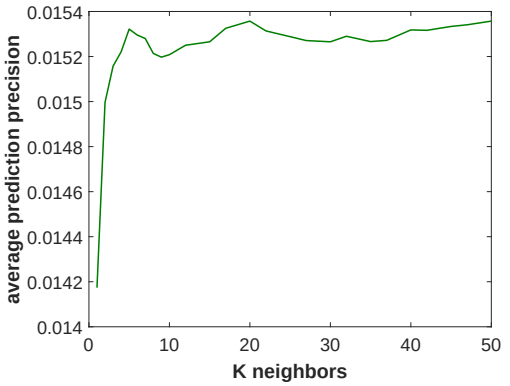
(a) UserKNN



(b) ItemKNN



(c) CUserKNN



(d) CItemKNN

Figure 4.4: Average prediction precision for various K values

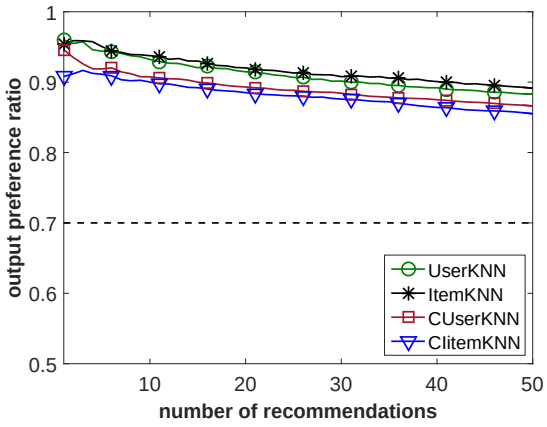


Figure 4.5: PR_R for r from 1 to 50; input preference ratio $PR_S = 0.7$

in recommendations increases significantly regardless the amount of recommended items. In any case, as r takes larger values, bias decreases and since the ratio of candidate items (Figure 4.3) has negative bias disparity, for large values of r, recommendations will also have negative bias disparity.

4.3.4 MAP of recommendations

In this experiment we want to understand how item categories rank according to their utility value if we see recommendations as ranked lists per user. Users are more likely to accept items that are ranked higher. Assume that a user will accept 6 out of 10 recommendations and that the PR_R of the set of recommendations is 0.7. The ranking where all 7 C_i items are positioned on the top of the list will result in the user accepting only C_i items. We consider this a highly biased ranking even though the preference ratio of the top-10 recommendations may be 0.7, equal to that of the input. On the other hand, if at the top 6 positions of the list we have 3 C_i items and 3 $\overline{C_i}$, then the set of recommendations that the user will accept, is unbiased.

We use the symmetric setting of equal-sized user groups G_1 and G_2 and equal-sized item categories C_1 and C_2 for varying input preference ratios from 0.5 to 1. For number of users n and number of recommended items per user r, we will measure the mean average precision (MAP) values of any recommendation algorithm A, as follows:

$$MAP_A = \frac{1}{n} \cdot \frac{1}{r} \cdot \sum_{u=1}^{n} \sum_{p=1}^{r} \frac{I(p,u)}{p}$$

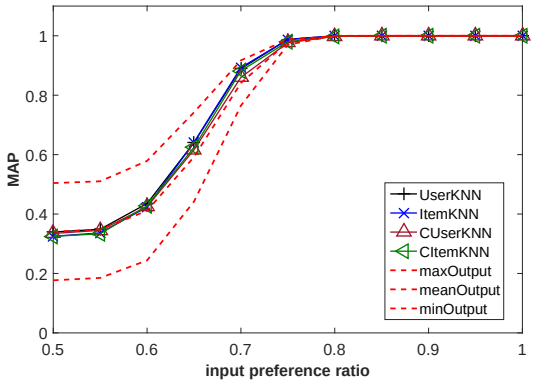


Figure 4.6: Average MAP of recommendations (G_i, C_i); the dashed lines show the minimum, maximum and mean possible MAP according to the input bias.

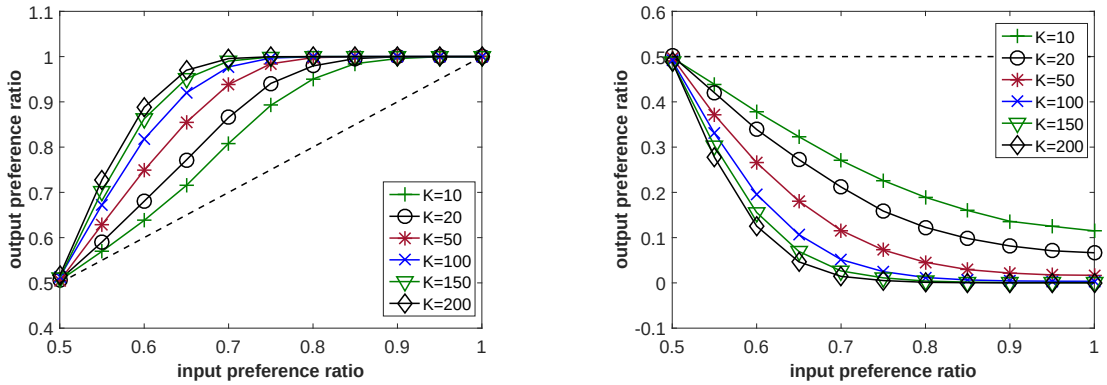
, where $I(p, u)$ is the number of C_i items in recommendations of user $u \in G_i$, $R(u)$, at positions from 1 to p.

In Figure 4.6, we plot the MAP values of the recommendations of each algorithm. The red dashed lines "minOutput", "meanOutput" and "maxOutput", show the minimum, the mean and the maximum possible MAP value for the average of the output preference ratios of the algorithms. The "maxOutput" shows the MAP values in the case where all the recommended C_i items are ranked on the top of the recommendation list, and the "minOutput" shows the MAP values when all C_i items are positioned on the bottom of the recommendation list. The "meanOutput" represents a random ranking.

We observe that for small preference ratio the ranking of items is random, which is reasonable since there is not that much output bias either (Figure 4.2). There is a point for $PR_S > 0.6$ and for $PR_S < 0.75$ where the recommenders have increasing MAP values and get closer to the maximum MAP. This indicates that for G_i users, C_i items are ranked a bit higher than $\overline{C_i}$. Finally, for large input preference ratio, the recommendations are completely biased and in that case we have MAP values equal to 1 since there are only C_i items in the lists.

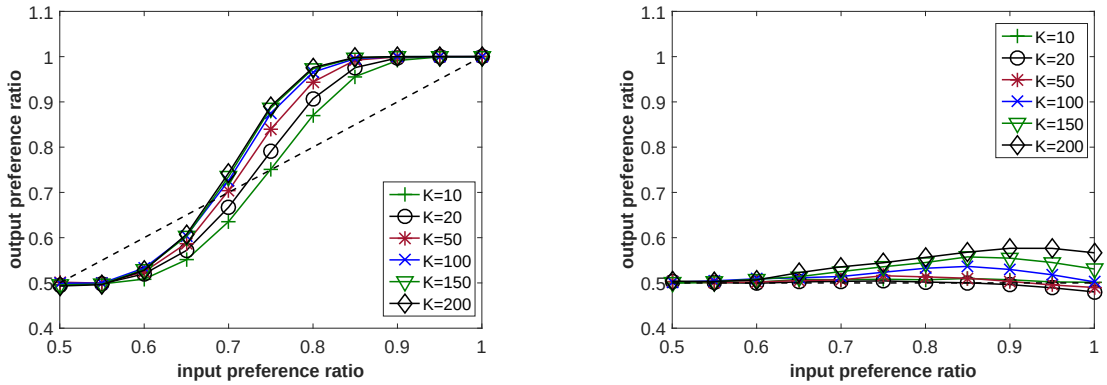
4.4 Asymmetric Preferences

In this set of experiments, the two groups have different bias preference ratios. We want to understand how the degree of bias in one group affects the bias of the other



(a) group G_1 (b) group G_2

Figure 4.7: UserKNN, $PR_R(G_i, C_i)$, asymmetric case



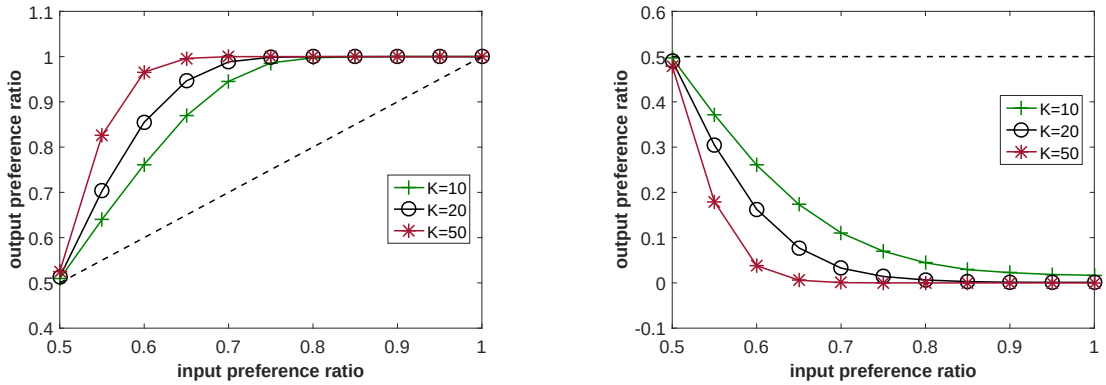
(a) group G_1 (b) group G_2

Figure 4.8: ItemKNN, $PR_R(G_i, C_i)$, asymmetric case

group. In our first experiment, group G_1 has preference ratio ρ_1 ranging from 0.5 to 1 while G_2 has fixed preference ratio $\rho_2 = 0.5$, that is, G_2 is unbiased. We show the recommendation preference ratio for groups G_1 and G_2 as a function of ρ_1.

We observe that the UserKNN output bias of group G_1 (Figure 4.7a) is amplified at a rate much higher than in Figure 4.2a, while group G_2 (Figure 4.7b) becomes biased towards category C_1. Surprisingly, the presence of the unbiased group G_2, rather than moderating the overall bias, it has an amplifying effect on the bias of G_1, more so than an opposite-biased group.

Furthermore, the unbiased group adopts the biases of the bias group. This is due to the fact that the users in the unbiased group G_2 provide a stronger signal in favor of category C_1 compared to the symmetric case where group G_2 is biased over C_2. This reinforces the overall bias in favor of category C_1.



(a) group G_1 (b) group G_2

Figure 4.9: CUserKNN, $PR_R(G_i, C_i)$, asymmetric case

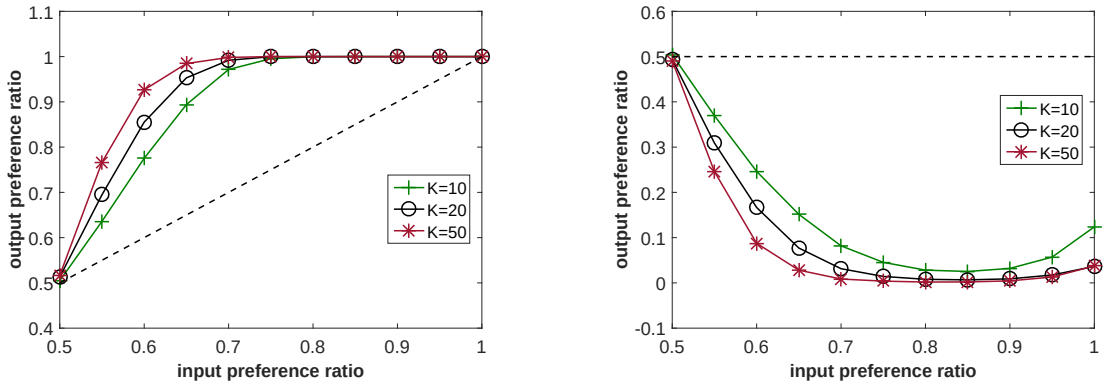
On the other hand, ItemKNN exhibits a more neutral behavior. The output bias of group G_1 in Figure 4.8a, decreases for smaller input bias ($PR_S \leq 0.7$) and increases with a lower rate than UserKNN for larger input bias. Bias becomes amplified for higher input bias than in the symmetric case (Figure 4.2b) because the unbiased group shares its ratings in both categories and it is easier for items of category C_2 to have neighbors in C_1.

The unbiased group G_2 (Figure 4.8b) receives almost unbiased recommendations from ItemKNN. When G_1 has higher input bias, then G_2 is pushed towards category C_2 for larger K values, rather than being drawn to C_1.

Unlike ItemKNN, the context algorithms once again result in positive bias disparity and behave similarly to UserKNN even though they result in biased recommendations faster. In Figure 4.9 we see that CUserKNN has fully biased recommendations for group G_1 after a point, depending on K, and it is more sharp for larger K values, while the CItemKNN recommendations are less affected by K but have also large bias disparity.

The unbiased group G_2 is completely drawn to category C_1 by both context algorithms because it is easier for them to discriminate items. We notice an interesting point where CItemKNN for the unbiased group G_2 has a decrease of bias disparity for large input bias of group G_1.

An interesting observation from this experiment is that ItemKNN behaves differently from the other recommenders. It tends to balance recommendations, and the more biased group does not draw the unbiased group to its category. The unbiased group impedes to a small extent the bias amplification of the biased group. UserKNN



(a) group G_1 (b) group G_2

Figure 4.10: CItemKNN, $PR_R(G_i, C_i)$, asymmetric case

and the context recommenders are more prone to input bias and affect the unbiased group significantly.

4.5 Varying group and category sizes

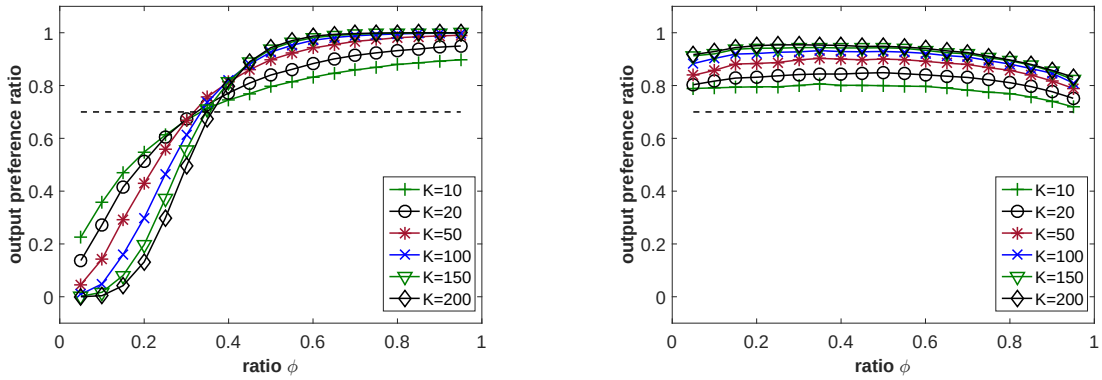
In this experiment we examine bias disparity with unbalanced groups and categories.

4.5.1 Varying Group Sizes

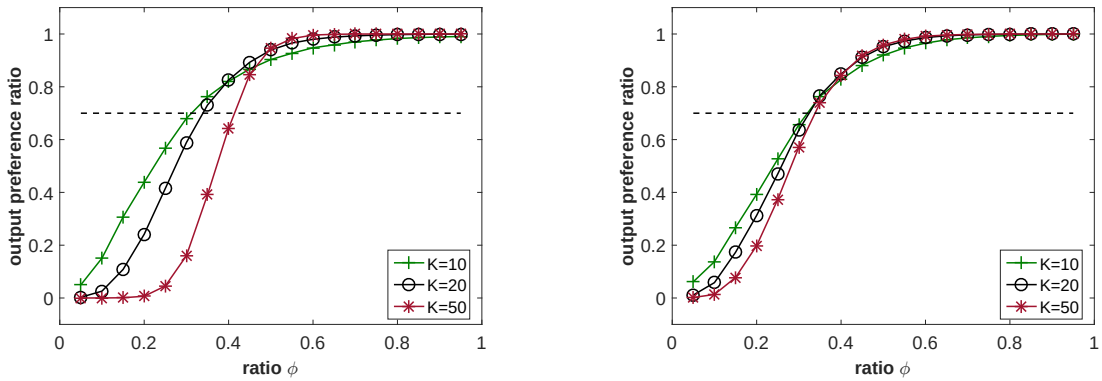
We first consider groups of uneven size. We set the size n_1 of G_1 to be a fraction ϕ of the number of all users n, ranging from 5% to 95%. Both groups have fixed preference ratio $\rho_1 = \rho_2 = 0.7$. Figure 4.11 shows the output preference ratio $PR_R(G_1, C_1)$ of the recommendation algorithms as a function of ϕ. The plots of $PR_R(G_2, C_2)$ are the mirror images of G_1 plots, so we do not report them.

We observe that for $\phi \leq 0.3$ group G_1 has negative bias disparity ($PR_R(G_1, C_1) < 0.7$) when we apply the UserKNN recommender. For medium values of ϕ in $[0.35, 0.5]$ the bias of both groups is amplified, despite the fact that G_1 is smaller than G_2. The increase is larger for the larger group, but there is increase for the smaller group as well.

The context recommendation algorithms have also negative bias disparity for small ϕ. That is, the small group is drawn by the larger group. It is reasonable that in UserKNN and CUserKNN users in the small group have more neighbors from the



(a) UserKNN (b) ItemKNN



(c) CUserKNN (d) CItemKNN

Figure 4.11: Unbalanced group sizes; input preference ratio $PR_S(G_i, C_i) = 0.7$

larger group. In the case of CItemKNN, items from the category preferred by the larger group have higher utility values because they are not normalized compared to ItemKNN. Again, the outputs of the context algorithms have resemblance with those of the UserKNN and have complete bias in recommendations for the larger group. However, CUserKNN shows more dependence on the value of K neighbors than CItemKNN.

ItemKNN as we see in Figure 4.11b, behaves differently from all other the recommenders. For any group size, the recommendations have positive bias disparity. That is, the two biased groups are polarized regardless of their size. However, ItemKNN is more vulnerable to small biased user groups. For $\phi > 0.6$, where G_1 is the larger group, bias disparity drops even though it remains positive. For small K and $\phi = 0.95$, we have $PR_R \approx PR_S$. Smaller user groups with more gathered ratings in one category, allow ItemKNN to select neighbor items from the same category and support recommendations from the same category.

4.5.2 Varying Category Sizes

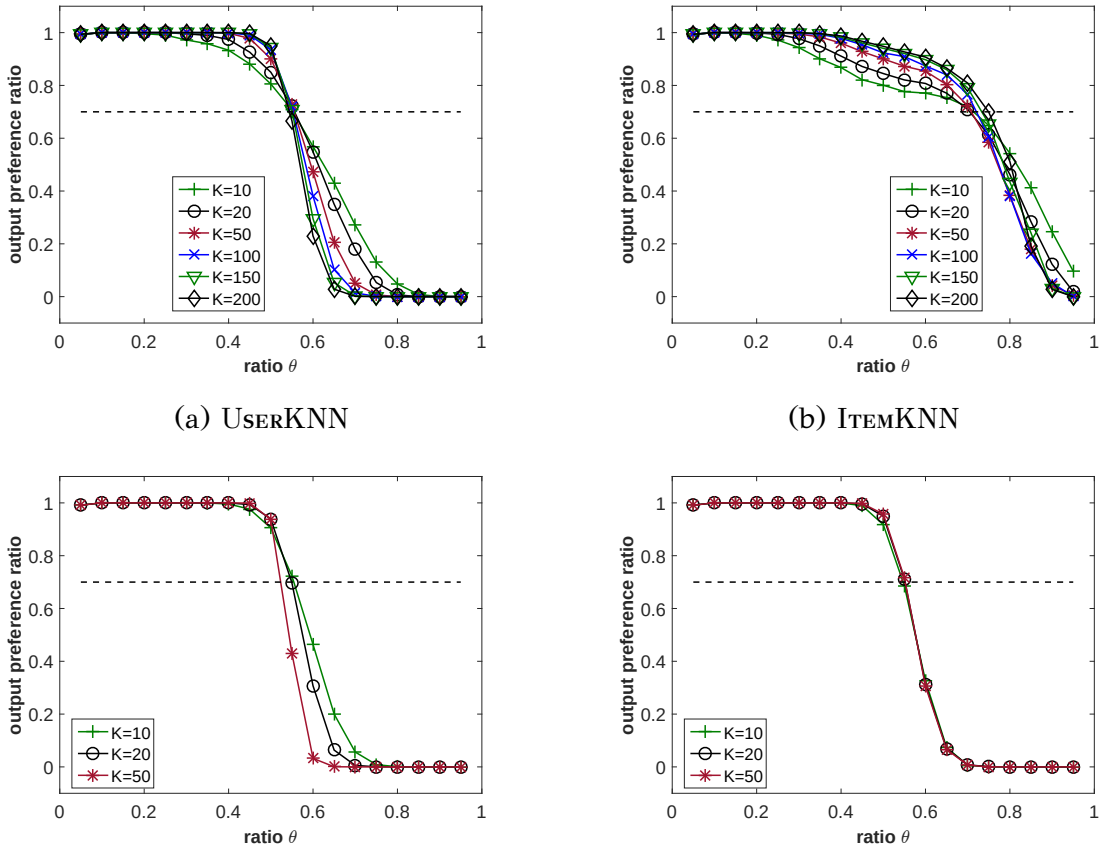
We now consider categories of uneven size. We set the size m_1 of C_1 to be a fraction θ of the number items m, ranging from 5% to 95%. We assume that both groups have fixed preference ratio $\rho_1 = \rho_2 = 0.7$. Figure 4.12 shows the recommendation preference ratios $PR_R(G_1, C_1)$ of the four algorithms, as a function of θ. The plots of $PR_R(G_2, C_2)$ are again the mirror images of these.

Note that as long as $\theta \leq 0.7$, group G_1 has positive bias (greater than 1) for category C_1 since bias is equal to ρ_1/θ. However, it decreases as the size of the category increases.

We observe that when the category size is not very large ($\theta \leq 0.5$), the output bias of UserKNN is amplified regardless of the category size. For $\theta > 0.7$, G_1 is actually biased in favor of C_2, and this is reflected in the output. There is an interesting range $[0.6, 0.7]$ where G_1 is positively biased towards C_1 but its bias is weak, and thus the recommendation output is drawn to category C_2 by the more biased group.

ItemKNN shows a similar behavior with UserKNN, since bias decreases when the category becomes larger. Bias is amplified for $\theta < 0.8$. The threshold where bias disparity becomes negative is greater than the other recommenders ($\theta = 0.6$) which are more vulnerable to category size.

The context algorithms are very sharp and also amplify the output bias for $\theta < 0.6$. It s interesting to note that CItemKNN is not affected by the value of K.



(a) UserKNN (b) ItemKNN

(c) CUserKNN (d) CItemKNN

Figure 4.12: Unbalanced category sizes; input preference ratio $PR_S(G_i, C_i) = 0.7$

4.6 Iterative Application of Recommendations

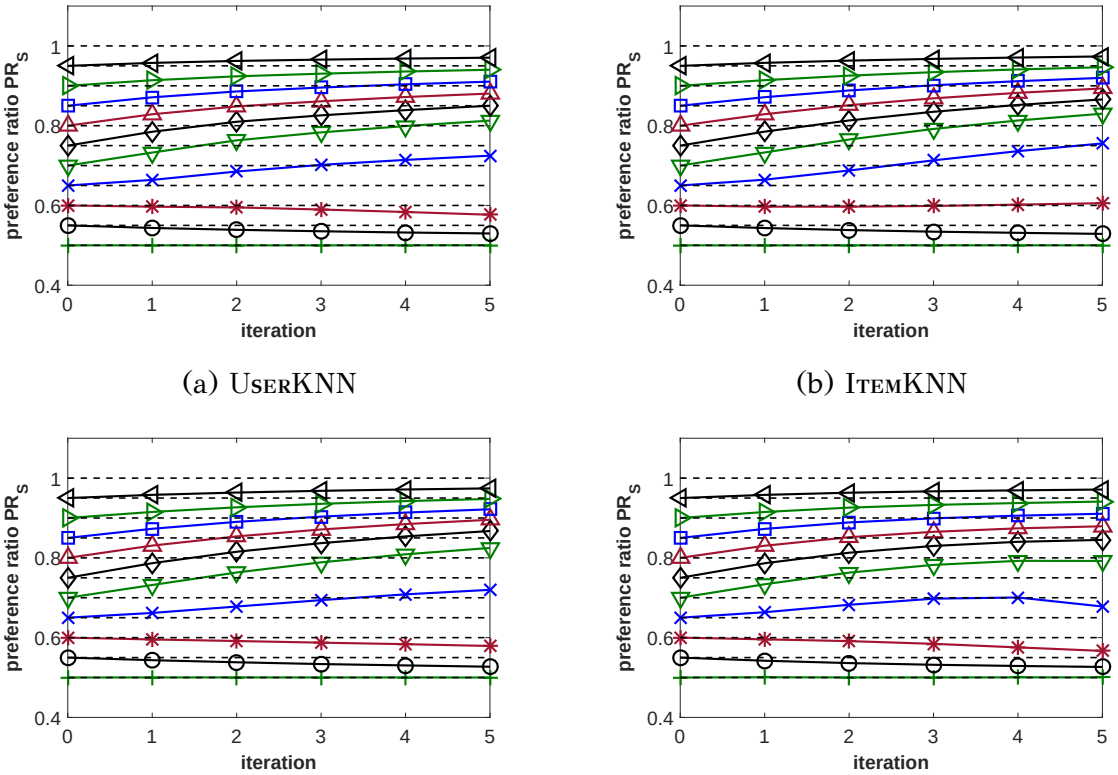
We observed bias disparity in the output of the recommendation algorithm. However, how does this affect the bias in the data? To study this we consider a scenario where the users accept (some of) the recommendations of the algorithm, and we study the long-term effect of the iterative application of the algorithm on the bias of the data. More precisely, at each iteration, we consider the top-r recommendations of the algorithm ($r = 10$) to a user u, and we normalize their utility values, by the utility value of the top recommendation. We then assume that the user accepts a recommendation with probability equal to the normalized score. The accepted recommendations are added to the data, and they are fed as input to the next iteration of the recommendation algorithm.

We apply this iterative algorithm on a dataset with two equally but oppositely biased groups, as described in Section 4.3. The results of this iterative experiment are shown in Figure 4.13, where we plot the average preference ratio for each iteration. Iteration 0 corresponds to the input data. For this experiment we set the value of K to the default values we selected, 100 for UserKNN and ItemKNN, and 10 for CUserKNN and CItemKNN.

We observe that even with the probabilistic acceptance of recommendations, there is a clear long-term effect of the recommendation bias. For small values of input bias, we observe a decrease in line with the observations in Figure 4.2. For these values of bias, the recommender will result in reducing bias and smoothing out differences. For larger values of preference ratio the bias in the data increases. Therefore, for large values of bias the recommender has a reinforcing effect, which in the long term will lead to polarized groups of users. However, it is interesting that CItemKNN (Figure 4.13d) has the most modest behavior. After a number of iterations, the bias disparity decreases. After iteration 4, it appears to have zero bias disparity for large input preference ratios.

The most vulnerable algorithm in the long-term, is ItemKNN (Figure 4.13b). The tendency to polarize biased groups that we have observed in Section 4.4, support these results.

In Figure 4.14 we report the average number of recommendations that each user accepts after every iteration of the recommender. We see that UserKNN and CItemKNN from more than 8 average accepted recommendations per user at the



(a) UserKNN (b) ItemKNN

(c) CUserKNN (d) CItemKNN

Figure 4.13: The evolution of the preference ratio in the data for different input preference ratios (PR_S), after 5 iterations of (a) UserKNN, (b) ItemKNN, (c) CUserKNN and (d) CItemKNN. Iteration 0 shows the original preference ratio of each dataset.

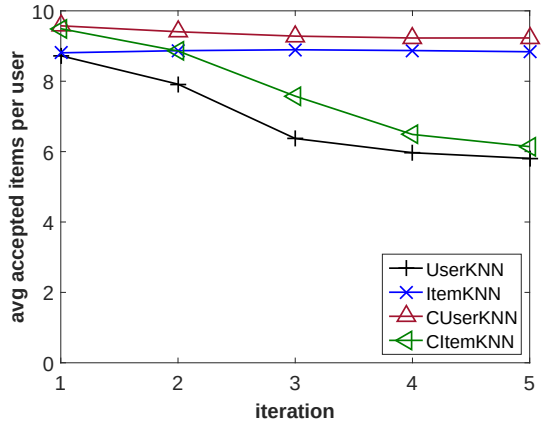


Figure 4.14: Average of accepted items per user on every iteration

first iteration, drop to 6 at iteration 5. However, ItemKNN and CUserKNN remain constantly above 8, which is why they have larger bias disparity in Figure 4.13.

4.7 Bias disparity on real data

4.7.1 Data and settings

In this set of experiments, we use the Movielens 1M dataset[1]. We select two groups of users and two movie genres. To create the associations matrix S, we exclude items that do not belong to one of the genres we selected and items that belong to both. We also exclude users with no ratings for the items selected. We ignore the values of ratings, and all specified ratings are considered positive associations with value 1 in matrix S. Unspecified ratings are also considered unspecified associations with value 0 in S. In every experiment, we first measure input bias, bias in recommendations and bias disparity. Secondly, we take a random sample of users from the larger group, equal to the small group, in order to balance the user groups, and measure bias again.

4.7.2 Gender bias

Bias disparity of recommenders

We consider as categories the genres Action and Romance, with 468 and 463 movies. We extract a subset of users $\mathcal{U}$ that have at least 90 ratings in these categories, resulting in 1,259 users. Users in $\mathcal{U}$ consist of 981 males and 278 females. In this experiment we use number of neighbors $K = 100$ for UserKNN and ItemKNN, and $K = 10$ for CUserKNN and CItemKNN.

In Table 4.1, we show the input/output bias and in parentheses the bias disparity for each group-category combination. The right part of the table reports these numbers when the user groups are balanced, by selecting a random sample of 278 males.

We observe that males are biased in favor of Action movies while females prefer Romance movies. The application of UserKNN increases the output bias for males for which group the input bias is strong. Females are moderately biased in favor of Romance movies. Hence, their output bias is drawn to Action items. We observe a

[1]MovieLens 1M: https://grouplens.org/datasets/movielens/1m/

	Unbalanced Groups		Balanced Groups	
	Action	Romance	Action	Romance
	UserKNN			
M	1.39/1.77 (0.27)	0.58/0.17 (-0.70)	1.40/1.73 (0.24)	0.57/0.22 (-0.62)
F	0.97/1.43 (0.47)	1.03/0.54 (-0.48)	0.97/1.33 (0.37)	1.03/0.64 (-0.38)
	CUserKNN			
M	1.39/1.67 (0.20)	0.58/0.28 (-0.51)	1.41/1.68 (0.19)	0.56/0.27 (-0.51)
F	0.97/1.15 (0.18)	1.03/0.84 (-0.19)	0.97/1.06 (0.09)	1.03/0.94 (-0.09)
	ItemKNN			
M	1.39/1.43 (0.03)	0.58/0.54 (-0.07)	1.39/1.51 (0.09)	0.59/0.45 (-0.23)
F	0.97/0.60 (-0.39)	1.03/1.43 (0.39)	0.97/0.68 (-0.30)	1.03/1.34 (0.30)
	CItemKNN			
M	1.39/1.73 (0.24)	0.58/0.22 (-0.62)	1.40/1.42 (0.01)	0.57/0.55 (-0.04)
F	0.97/1.85 (0.91)	1.03/0.09 (-0.92)	0.97/1.71 (0.76)	1.03/0.24 (-0.77)

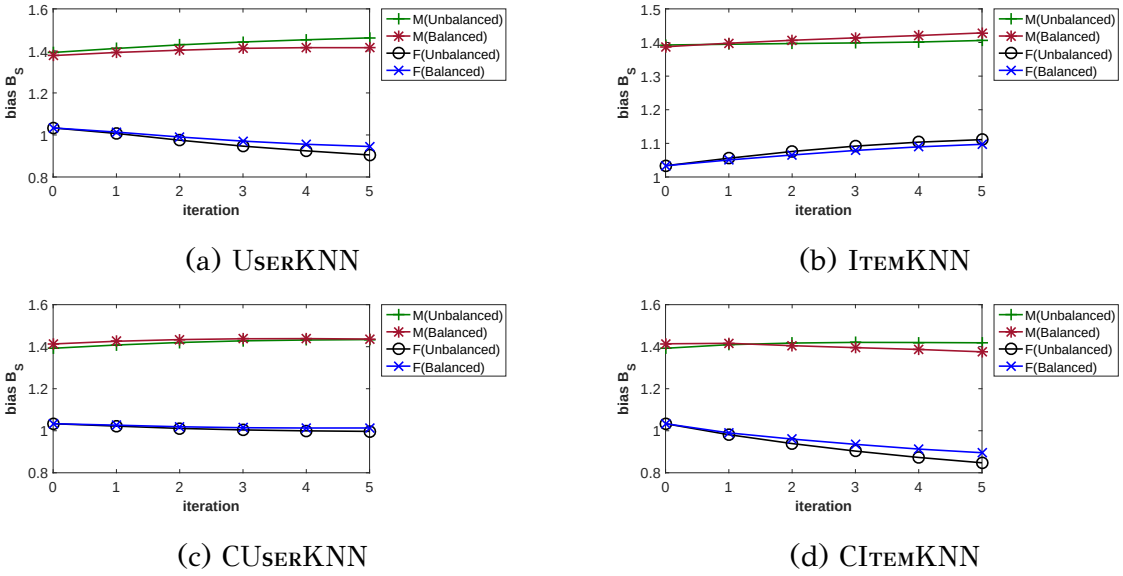
Table 4.1: Gender bias in action and romance

very similar picture for balanced data, indicating that the changes in bias are not due to the group imbalance. CUserKNN has similar behavior but exhibits lower bias disparity. The ItemKNN algorithm, as we observed in the experiments with the synthetic data, polarizes the two groups by increasing their biases in both the unbalanced and balanced cases. The stronger effect on the less biased group is observed in CItemKNN, which results in almost complete opposite bias for the female group in the unbalanced case. In the balanced case we have lower bias disparity but still significant.

Iterative application of recommenders

On the same setting of gender groups, we apply iteratively the recommendation algorithms. We assume that after every iteration, users accept recommendations with a probablility. We normalize the utility values of the recommendations of each user, with the highest utility in each user's recommendations list and we accept items with that probability. In the associations matrix, we add the accepted items for each user and measure the new input bias B_S. Then we apply the recommenders on the new associations.

In Figure 4.15, we plot the bias for 5 iterations of the four recommendation algorithms. Iteration 0 indicates the original input bias. We consider again the cases



(a) UserKNN (b) ItemKNN

(c) CUserKNN (d) CItemKNN

Figure 4.15: The evolution of bias in the data, after 5 iterations of the recommenders: (a) UserKNN, (b) ItemKNN, (c) CUserKNN and (d) CItemKNN. M: $B_S(Male, Action)$, F: $B_S(Female, Romance)$. Iteration 0 shows the original bias.

of unbalanced and balanced groups. The line M shows the bias for males to action ($B_S(Males, Action)$), while F shows the bias for females and romance ($B_S(Females, Romance)$).

We observe first that there is no significant difference between the balanced and the unbalanced setting. We see that in the long-term, bias increases for Males that are originally more biased, while Females have usually negative bias disparity and they are drawn to the Action category. Once again, the ItemKNN has distinctive behavior and polarizes the two groups, since they both have positive bias disparity.

4.7.3 Social bias

Biased groups occur also when we examine different occupations. An interesting example includes students from kindergarten to 12th grade and people that have retired. Students group has 195 users and Retired has 142. We apply the recommenders for $K = 5$ for CUserKNN and CItemKNN, and for $K = 10$ for UserKNN and ItemKNN, since we have a smaller dataset.

As we see in Table 4.2, these groups are oppositely biased in the categories of Children's and War movies respectively. Children's category has 249 items and War has 141. The Retired group is completely biased in favor of War movies which is a smaller and more dense item category. This results in ItemKNN increasing the bias

and drawing Students to War items. UserKNN and CUserKNN mitigate biases while CItemKNN enables more recommendations from the larger category Children's where we have very large bias in the input. Similar to Table 4.1, the CItemKNN has high bias disparity for the Retired group which is smaller and is drawn to the large category.

	Unbalanced Groups		Balanced Groups	
	Children's	War	Children's	War
	UserKNN			
S	1.45/1.27 (-0.12)	0.67/0.80 (0.19)	1.45/1.28 (-0.12)	0.67/0.80 (0.19)
R	0.47/0.62 (0.33)	1.39/1.28 (-0.08)	0.47/0.60 (0.29)	1.39/1.29 (-0.07)
	CUserKNN			
S	1.45/1.24 (-0.14)	0.67/0.82 (0.23)	1.46/1.25 (-0.14)	0.66/0.82 (0.23)
R	0.47/0.60 (0.30)	1.39/1.29 (-0.07)	0.47/0.57 (0.23)	1.39/1.31 (-0.06)
	ItemKNN			
S	1.45/1.13 (-0.22)	0.67/0.90 (0.34)	1.41/1.14 (-0.19)	0.70/0.90 (0.29)
R	0.47/0.39 (-0.15)	1.39/1.44 (0.04)	0.47/0.45 (-0.03)	1.39/1.40 (0.01)
	CItemKNN			
S	1.45/1.63 (0.12)	0.67/0.54 (-0.19)	1.47/1.68 (0.14)	0.65/0.50 (-0.23)
R	0.47/0.82 (0.76)	1.39/1.13 (-0.19)	0.47/0.79 (0.70)	1.39/1.15 (-0.17)

Table 4.2: Occupation bias in children's and war

Finally, we examine the group of Homemakers with 92 users and the Retired with 142 users. We have two almost equal categories. Animation has 104 items and Mystery has 105. We again use $K = 5$ for the context algorithms and $K = 10$ for the primary.

In Table 4.3, we observe some bias amplification for CItemKNN in the balanced case, where it has positive bias amplification for both groups in Animation, due to the fact that there are more ratings in that category, therefore stronger signal. The rest of the recommenders mitigate biases in any case. In the unbalanced case though, we have more bias decrease for the smaller group than in the balanced case.

	Unbalanced Groups		Balanced Groups	
	Animation	Mystery	Animation	Mystery
	UserKNN			
H	1.28/1.07 (-0.17)	0.72/0.93 (0.30)	1.28/1.13 (-0.12)	0.72/0.87 (0.22)
R	0.59/0.76 (0.27)	1.40/1.24 (-0.11)	0.62/0.79 (0.28)	1.38/1.21 (-0.12)
	CUserKNN			
H	1.28/1.08 (-0.16)	0.72/0.92 (0.28)	1.28/1.19 (-0.07)	0.72/0.81 (0.13)
R	0.59/0.70 (0.19)	1.40/1.29 (-0.08)	0.72/0.89 (0.23)	1.27/1.11 (-0.13)
	ItemKNN			
H	1.28/1.08 (-0.16)	0.72/0.92 (0.28)	1.28/1.11 (-0.14)	0.72/0.90 (0.25)
R	0.59/0.61 (0.03)	1.40/1.38 (-0.01)	0.58/0.62 (0.07)	1.42/1.38 (-0.03)
	CItemKNN			
H	1.28/1.27 (-0.01)	0.72/0.73 (0.02)	1.28/1.34 (0.04)	0.72/0.67 (-0.07)
R	0.59/0.67 (0.12)	1.40/1.33 (-0.05)	0.68/0.86 (0.26)	1.31/1.14 (-0.14)

Table 4.3: Occupation bias in animation and mystery

Chapter 5

Correcting Bias Disparity

To address the problem of bias disparity, we consider two algorithms that performs post-processing of the recommendations. Our goal is to adjust the set of items recommended to users so as to ensure that there is no bias disparity. Each algorithm aims at producing new recommendations while maintaining high utility for the recommendation set.

We consider two algorithms: the GULM (Group Utility Loss Minimization) algorithm, and the MULM (Maximum User Utility Loss Minimization). The GULM algorithm targets on providing new recommendations that have the maximum possible average utility for the group of users, while MULM processes recommendations in a way that the minimum utility of all recommendations is the maximum possible.

5.1 GULM: **Group Utility Loss Minimization**

The GULM algorithm processes a set of recommendations R and produces a new set of recommendations R_{GULM} that minimize the bias disparity of a group G_i in categories C_i and $\overline{C_i}$, while minimizing the *group utility loss* for G_i.

Abusing the notation, let R denote the set of user-item pairs produced by our recommendation algorithm, where $(u, i) \in R$ denotes that u was recommended item i. We will refer to the pair (u, i) as a recommendation. The set R contains r recommendations for each user, thus, rn recommendations in total. Let $V(R) = \sum_{(u,i)\in R} V(u, i)$ denote the total utility of the recommendations in set R. Since R contains for each user u the top-r items with the highest utility, R has the maximum utility.

We want to adjust the set R so as to minimize the bias disparity $BD(G_i, C_i)$. Since we have two categories, it suffices to minimize $|B_R(G_i, C_i) - B_S(G_i, C_i)|$. That is, the absolute value of the difference of the bias of each group in R and the one in the input data. Without loss of generality assume that $B_R(G_i, C_i) > B_S(G_i, C_i)$. Let $\overline{C_i}$ denote the category other than C_i. We decrease the output bias B_R by swapping recommendations (u, i) of category C_i with recommendations (u, j) of category $\overline{C_i}$. The GULM algorithm applies a simple greedy rule where at each step it swaps the pair of recommendations that incur the minimum utility loss.

We define swapping (u, i) with (u, j) as swap s, and we say that s *involves* user u and items i, j. That is,

$$s = \langle (u, i), (u, j) \rangle \tag{5.1}$$

The *utility loss* incurred by a swap s is

$$loss(s) = V(u, i) - V(u, j) \tag{5.2}$$

That is, the difference of the utility values of item i that is replaced and the new item j, for user u. Assume that S is the set of swaps for all users in the group G. If $S_u \subseteq S$ is the set of swaps involving user u, then the *user loss* is the sum of losses incurred by each swap for u. That is,

$$loss(u, S) = \sum_{s\in S_u} V(u, i_s) - V(u, j_s) \tag{5.3}$$

If we perform all swaps in S, the *group utility loss* for G is the sum of losses incurred by all swaps in S, or else the sum of user losses for all users in G.

$$loss(G, S) = \sum_{s\in S} V(u_s, i_s) - V(u_s, j_s) \tag{5.4}$$

From all the possible swaps that involve users in G_i, we select as candidate swaps those that incur the minimum possible utility loss. The candidate swaps can be computed by pairing for each user u the lowest-ranked unpaired recommendation (u, i) in

R from category C_i, with the highest-ranked unpaired recommendation (u, j) not in R from category $\overline{C_i}$ as you can see in Algorithm 5.1 (lines:8-16). We perform swaps like that with ascending loss of utility (Alg.5.1, line:17) until the desired number of swaps has been performed (Alg.5.1, lines:20-23). The number of desired swaps (Alg.5.1, line:19) is the one that minimizes the bias disparity $BD(G_i, C_i)$. It is not always possible to obtain the exact bias as in the input data, because the number of necessary swaps may not be an integer number. The number of desired swaps is the one that minimizes $|B_R(G_i, C_i) - B_S(G_i, C_i)|$. It is reasonable to expect a small error in the results.

This algorithm is efficient, and it is easy to show that it is optimal, in the sense that it will produce the set of recommendations with the highest utility among all sets with no bias disparity.

Lemma 5.1. *The algorithm* GULM *has the optimal group utility loss for a number of swaps ℓ, compared to any algorithm that performs equal number of swaps.*

Proof. Let P be the set of all possible swaps, and W the set of swaps we defined through the matching process. For a user u, a set of possible swaps S, and a number k, let

$$loss_o(u, S, k)$$

be the minimum (optimal) loss of performing k swaps from set S for user u. It is easy to show that for any k, we have

$$loss_o(u, W, k) \leq loss_o(u, P, k) \tag{5.5}$$

If the optimal K swaps from S for user u are the K pairs of items i with items j,

$$(i_1, j_1),\ ...,\ (i_K, j_K)$$

then the loss of these swaps is $loss_o(u, S, K)$:

$$loss_o(u, S, K) = \sum_{k=1}^{K} V(u, i_k) - \sum_{k=1}^{K} V(u, j_k) \tag{5.6}$$

$loss_o(u, S, K)$ is minimum when $\sum_{k=1}^{K} V(u, i_k)$ is the minimum possible and $\sum_{k=1}^{K} V(u, j_k)$ is the maximum possible. Since, in W we select the i items for user u that minimize the sum of utilities and the j items that maximize the sum of utilities for u, for K, they are the swaps with the optimal loss.

Let $loss_o(S, k)$ be the minimum group loss of performing k swaps selected from set S. We will show that

$$loss_o(W, k) \leq loss_o(P, k) \tag{5.7}$$

We have

$$loss_o(P, k) = \sum_u loss_o(u, P, k_u) \tag{5.8}$$

where k_u is the number of swaps for user u and the sum is over all involved users. For the same users, using swaps only from W, we have

$$loss_o(W, k) = \sum_u loss_o(u, W, k_u) \leq \sum_u loss_o(u, P, k_u) = loss_o(P, k) \tag{5.9}$$

Now for the set W it is trivial to show that the greedy is optimal, since we are simply adding up the losses of independent swaps, and we just select the k smallest numbers. □

Algorithm 5.1 The GULM Algorithm

Input: S: associations matrix, R: recommendations matrix, V: utility values

Output: R_{GULM} recommendations with $B_{GULM} \approx B_S$

calculate biases B_R and B_S
for each item i **do**
  if $S(u,i) = 0$ & $R(u,i) = 0$ **then**
    $C(u,i) \leftarrow 1$ find candidate pairs
  end if
end for
if $B_R > B_S$ **then**
  for each user u **do**
    $L_{C_i} \leftarrow$ list of items $i \in C_i$, for which $R(u,i) = 1$, sorted by V ascending
    $L_{\overline{C_i}} \leftarrow$ list of items $i \in \overline{C_i}$, for which $C(u,i) = 1$, sorted by V descending
    $possible\_swaps \leftarrow \min(|L_{C_i}|, |L_{\overline{C_i}}|)$
    for i from 1 to $possible\_swaps$ **do**
      $x\_item \leftarrow L_{C_i}(i)$; $y\_item \leftarrow L_{\overline{C_i}}(i)$; $loss \leftarrow V(x\_item) - V(y\_item)$
      $Swaps(i) \leftarrow (u, x\_item, y\_item, loss)$ keep candidate swaps
    end for
  end for
  sort $Swaps$ by $loss$ in ascending order
  $R_{GULM} \leftarrow R$
  $desired\_swaps \leftarrow$ calculate number of swaps that minimize $|B_R - B_S|$
  for i from 1 to $desired\_swaps$ **do**
    $R_{GULM}(Swaps(i).u,\ Swaps(i).x\_item) \leftarrow 0$
    $R_{GULM}(Swaps(i).u,\ Swaps(i).y\_item) \leftarrow 1$
  end for
else
  replace $\overline{C_i}$ items with C_i
end if

5.2 MULM: **Maximum User Utility Loss Minimization**

The Minimize Maximum User Utility Loss MULM algorithm processes a set of recommendations R and produces a new set of recommendations R_{MULM} that minimize the bias disparity of a group G_i in categories C_i and $\overline{C_i}$, while minimizing the *max user utility loss* for G_i.

As described in Section ??, the GULM algorithm swaps recommendations of one category with another, while we keep the utility loss for the group to the minimum possible. However, this may result in some users having many swaps and incurring high user utility loss, while others receive the original set of recommendations as they were selected by the recommender, with no utility loss. To avoid mistreating a portion of the users, we consider a different algorithm that adjusts the set of recommendations R, so as to ensure that the bias in R is the same as the input bias, while minimizing the maximum over all users utility loss. We call this algorithm Minimize Maximum User utility Loss (MULM). MULM is optimal and results in the minimum possible maximum user loss.

Assume that $B_R(G_i, C_i) > B_S(G_i, C_i)$. We swap recommendations (u, i) of category C_i with recommendations (u, j) of category $\overline{C_i}$ with a greedy algorithm. Unlike GULM, this algorithm at each step takes into account the utility loss that each user incurred in previous steps and selects the swap with the minimum total loss for a user. In the Algorithm 5.2, you can see that we keep the additive loss for the user after we perform a swap (lines:24-27).

Again the candidate swaps are computed by pairing for each user u the lowest-ranked recommendation (u, i) in R from category C_i, with the highest ranked recommendation (u, j) not in R from category $\overline{C_i}$ and the number of the desired swaps is the one that minimizes $|B_R(G_i, C_i) - B_S(G_i, C_i)|$. We perform the desired number of swaps while we re-order the candidate swaps after each swap (Alg.5.2, line:28). At step k, the swap s_k is on the top of the list of candidate swaps and it will result in the minimum increase of the maximum user utility loss. If swap s_k replaces (u, i) with (u, j) and has total user loss for u, $loss_k(u)$, then after we perform the swap, we remove it from the candidate swaps list and update all other swaps for user u that remain in the list, by adding $loss_k(u)$ to the loss of each swap. This way, we keep track of the total loss every user incurred after a number of swaps. Finally, we sort again the list of candidate swaps by ascending total user loss and proceed to the next

step.

After the desired number of the swaps, the bias of the updated recommendations is the same as the input bias B_S and the total utility loss is shared by as many users as possible.

Lemma 5.2. *The algorithm* MULM *has the optimal maximum user utility loss for a number of swaps s, compared to any algorithm that performs equal number of swaps.*

Proof. To prove the algorithm's optimality, we assume that MULM is not optimal and there is another solution that results in the optimal maximum user loss. We denote with $loss_o(u)$ and $loss_m(u)$ the utility loss of user u in the case of optimal and MULM respectively. Let u_o be the user with the maximum loss for optimal and u_m the user with the maximum loss for MULM. We have

$$loss_o(u_o) \geq loss_o(u), \forall u \tag{5.10}$$

and

$$loss_m(u_m) \geq loss_m(u), \forall u \tag{5.11}$$

Assume for the purpose of contradiction, that

$$loss_m(u_m) > loss_o(u_o) \tag{5.12}$$

From (5.10) and (5.12),

$$loss_m(u_m) > loss_o(u_m) \tag{5.13}$$

From (5.13), it must be that MULM makes more swaps for u_m than optimal. Since both algorithms make the same number of swaps, then optimal must make at least one more swap for some user than MULM. Let w be this user, then

$$loss_o(w) \geq loss_m(w) \tag{5.14}$$

Take the point at which MULM makes the last swap that includes w. Since w is never included in any further swaps, it must hold

$$loss_m(w) \geq loss_m(u), \forall u \tag{5.15}$$

otherwise w should have been swapped. From (5.14) and (5.15),

$$loss_o(w) \geq loss_m(u), \forall u \tag{5.16}$$

and therefore,

$$loss_o(w) \geq loss_m(u_m) \tag{5.17}$$

Using (5.10)

$$loss_o(u_o) \geq loss_m(u_m) \tag{5.18}$$

which contradicts our assumption (5.12).

□

Algorithm 5.2 The MULM Algorithm

Input: S: associations matrix, R: recommendations matrix, V: utility values

Output: recommendations R_{MULM} with $B_{MULM} \approx B_S$

calculate biases B_R and B_S
for each item i **do**
  if $S(u,i) = 0$ & $R(u,i) = 0$ **then**
    $C(u,i) \leftarrow 1$ find candidate pairs
  end if
end for
if $B_R > B_S$ **then**
  for each user u **do**
    $L_{C_i} \leftarrow$ list of items $i \in C_i$, for which $R(u,i) = 1$, sorted by V ascending
    $L_{\overline{C_i}} \leftarrow$ list of items $i \in \overline{C_i}$, for which $C(u,i) = 1$, sorted by V descending
    $possible\_swaps \leftarrow \min(|L_{C_i}|, |L_{\overline{C_i}}|)$
    for i from 1 to $possible\_swaps$ **do**
      $x\_item \leftarrow L_{C_i}(i)$; $y\_item \leftarrow L_{\overline{C_i}}(i)$; $loss \leftarrow V(x\_item) - V(y\_item)$
      $Swaps(i) \leftarrow (u, x\_item, y\_item, loss)$ keep candidate swaps
    end for
  end for
  sort $Swaps$ by $loss$ in ascending order
  $R_{MULM} \leftarrow R$
  $desired\_swaps \leftarrow$ calculate number of swaps that minimize $|B_R - B_S|$
  for i from 1 to $desired\_swaps$ **do**
    $R_{MULM}(Swaps(1).u,\ Swaps(1).x\_item) \leftarrow 0$
    $R_{MULM}(Swaps(1).u,\ Swaps(1).y\_item) \leftarrow 1$
    for i from 1 to $size(Swaps)$ **do**
      if $Swaps(i).u = Swaps(1).u$ **then**
        $Swaps(i).loss \leftarrow Swaps(i).loss + Swaps(1).loss$
      end if
    end for
    remove $Swaps(1)$ and sort $Swaps$ by $loss$ in ascending order
  end for
else
  replace $\overline{C_i}$ items with C_i
end if

5.3 Evaluation of bias correcting algorithms

In this Section, we present the experiments that evaluate the behavior of the correcting algorithms GULM and MULM.

5.3.1 Prediction precision of correcting algorithms

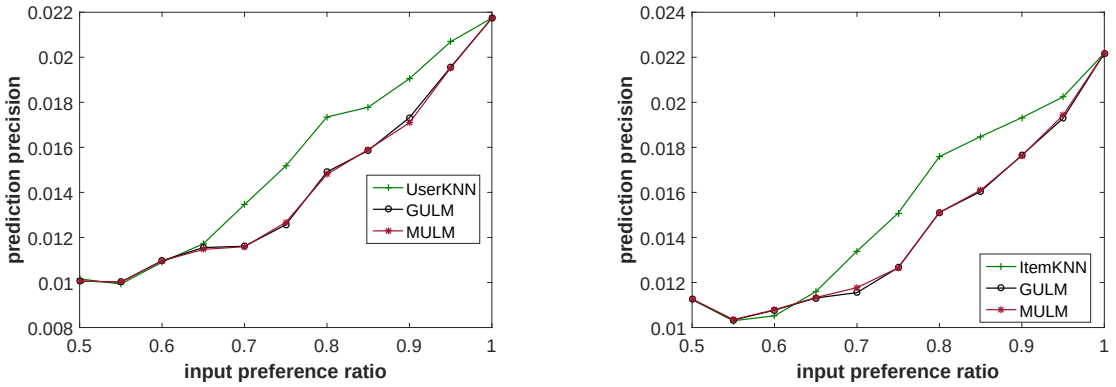
First, we measure the prediction precision of GULM and MULM applied on the recommendations of each recommender. We use datasets with symmetric setting of equal-size groups (G_1, G_2) and equal-size categories (C_1, C_2) with equal preference ratios $\rho_1 = \rho_2 = \rho$, where ρ varies from 0.5 to 1, in increments of 0.05. To measure the precision of predictions, we perform 5-fold cross validation on each dataset and measure the average precision of recommendations R compared to the $test$ set, for each algorithm A:

$$precision_A = \frac{1}{5} \cdot \sum_{fold=1}^{5} \frac{|R \cap test|}{|R|}$$

In Figure 5.1 we present the results for each recommender and the correcting algorithms applied to their recommendations. We observe that the correcting algorithms have equal precision in every case. For input preference ratio $PR_S \leq 0.65$, they have equal precision with the recommendation algorithm. This is reasonable because as we observed in Section 4.3 and Figure 4.2, for small input preference ratio the absolute value of bias disparity is small. Therefore, the correcting algorithms need to perform only a small number of swaps that do not affect the precision. However, after that point and for $PR_S > 0.65$, the precision of the correcting algortihms drops. This confirms that recommendations incur utility loss in order to minimize the bias disparity.

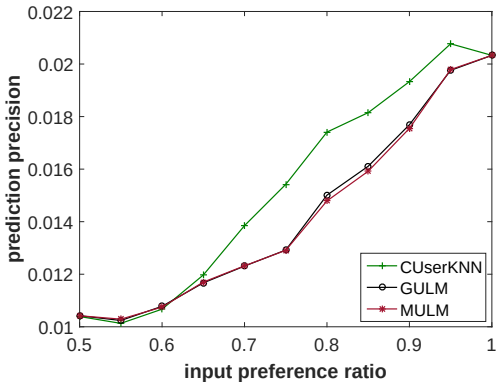
5.3.2 Iterative application of correcting algorithms

By design, when we apply the GULM algorithm on the output of the recommendation algorithm, we eliminate bias disparity (modulo rounding errors) in the recommendations. We consider the iterative application of the recommendation algorithm, in the setting described in Section 4.6, again assuming that the probability of a recommendation being accepted depends on its utility. The results are shown in Figures 5.2 & 5.3.

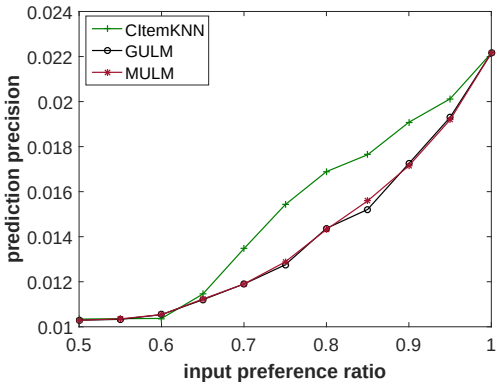


(a) UserKNN

(b) ItemKNN

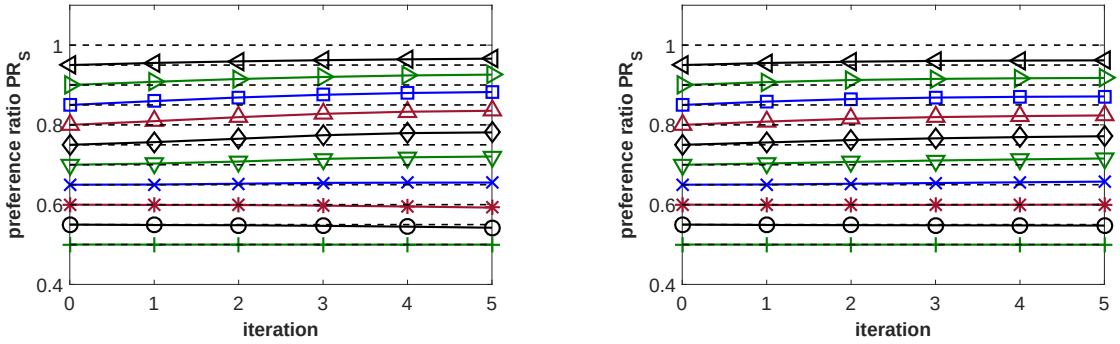


(c) CUserKNN



(d) CItemKNN

Figure 5.1: Prediction precision of correcting algorithms applied on (a) UserKNN, (b) ItemKNN, (c) CUserKNN and (d) CItemKNN.



(a) GULM on UserKNN

(b) GULM on ItemKNN

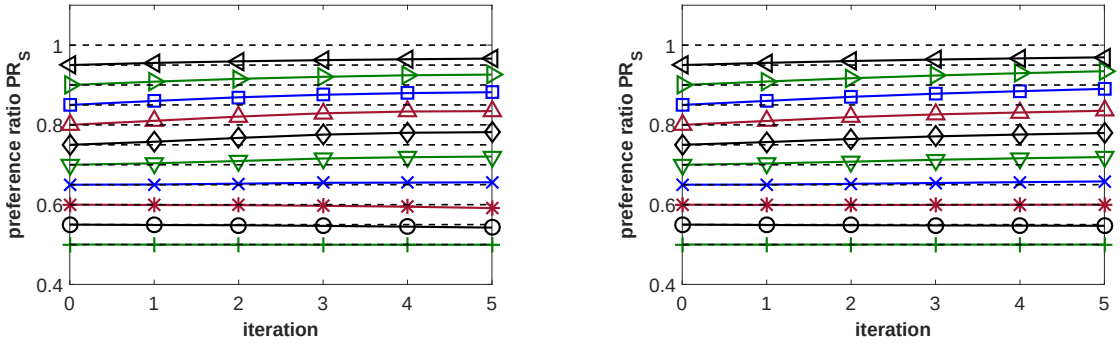
(c) GULM on CUserKNN

(d) GULM on CItemKNN

Figure 5.2: The evolution of the preference ratio in the data for different input preference ratios (PR_S), after 5 iterations of GULM on recommendations of (a) UserKNN, (b) ItemKNN, (c) CUserKNN and (d) CItemKNN. Iteration 0 shows the original preference ratio of each dataset.

For values of preference ratio up to 0.65, we observe that bias remains more or less constant after re-ranking. For larger values of preference ratio, there is some noticeable increase in the bias when we apply GULM on recommendations of UserKNN and ItemKNN, due to the fact that the recommendations introduced by GULM have low probability to be accepted. Surprisingly, for UserKNN (Figure 5.2a) we have the larger bias disparity even though it performed well without the correcting. In general, as expected the bias increase id significantly smaller than before re-ranking. The GULM appears to be effective on CUserKNN and CItemKNN.

MULM has similar results as we show in Figure 5.3. ItemKNN with MULM has the largest bias disparity among the recommenders, as before re-ranking. Again, correcting on UserKNN performs poorly. It is interesting that in any case of correcting, the CUserKNN has little to none bias increase, even though without the re-ranking



(a) MULM on UserKNN

(b) MULM on ItemKNN

(c) MULM on CUserKNN

(d) MULM on CItemKNN

Figure 5.3: The evolution of the preference ratio in the data for different input preference ratios (PR_S), after 5 iterations of MULM on recommendations of (a) UserKNN, (b) ItemKNN, (c) CUserKNN and (d) CItemKNN. Iteration 0 shows the original preference ratio of each dataset.

it has significant positive bias disparity for large preference ratios.

In Figure 5.4, we show the average number of items that each user accepts after every iteration. We notice that applying a correcting algorithm does not necessarily result in users accepting less recommendations. The average accepted items after applying a correcting algorithm is close to the average accepted items for the recommender and in fact, the CItemKNN has less accepted items before the application of the correcting algorithms, after the third iteration.

5.3.3 MAP of recommendations

With this experiment we want to understand how item categories rank according to their utility value if we see recommendations as ranked lists per user. We again use the symmetric setting of equal-sized user groups G_1 and G_2 and equal-sized item

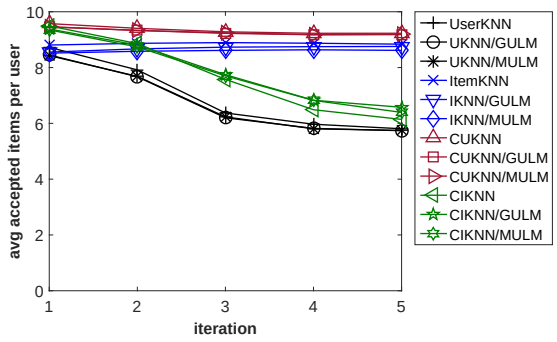


Figure 5.4: Average of accepted items per user on every iteration

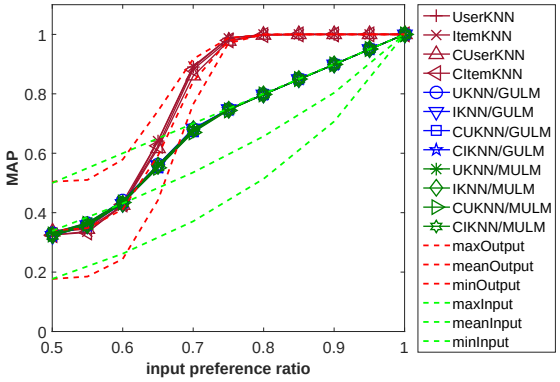


Figure 5.5: Average MAP of recommendations (G_i, C_i); the dashed lines show the minimum, maximum and mean possible MAP according to the input bias.

categories C_1 and C_2 for varying input preference ratios from 0.5 to 1. For n users and r recommendations per user, we measure the mean average precision (MAP) values of algorithm A, as follows:

$$MAP_A = \frac{1}{n} \cdot \frac{1}{r} \cdot \sum_{u=1}^{n} \sum_{p=1}^{r} \frac{I(p,u)}{p}$$

, where $I(p,u)$ is the number of C_i items in recommendations of user $u \in G_i$, $R(u)$, at positions from 1 to p.

In Figure 5.5, we plot the MAP values of the recommendation algorithms and the correcting algorithms applied on the recommendations of each recommender as well. We remind that the red dashed lines "minOutput", "meanOutput" and "maxOutput", show the minimum, the mean and the maximum possible MAP value for the average of the output preference ratios of the algorithms. The green dashed lines "minInput", "meanInput" and "maxInput" indicate respectively the minimum, mean and maximum possible MAP values for the input preference ratio. In the case of the

correcting algorithms, recommendations have preference ratio equal to the input. So "maxInput" shows the MAP values in the case where all the recommended C_i items are ranked on the top of the recommendation list, and the "minInput" shows the MAP values when all C_i items are positioned on the bottom of the recommendation list. The "meanInput" represents a random ranking.

We observe that the MAP values for the correcting algorithms show a random ranking for $PR_S <= 0.6$. For an average input preference ratio from $PR_S > 0.6$ to $PR_S < 0.75$, the MAP values increase and for large input preference ratio we have the maximum possible MAP. Ranking of items for the correcting algorithms is similar to the one of the recommendation algorithms. For large input bias the C_i items are positioned at the top of the recommendations lists. This is reasonable since we swap recommendations of C_i that are ranked low, with items of $\overline{C_i}$ with less utility values. This results in providing recommendations to the group with bias equal to the group's preference bias but retaining a position bias.

Chapter 6

Conclusions

In this thesis, we performed an experimental study of bias disparity in recommender systems, and the conditions under which it may appear. Using synthetic data, we observed that bias disparity can appear for moderately biased groups. Bias disparity becomes stronger in the presence of unbiased users at most cases, even when the biased group is small. Larger groups in general bias smaller groups, while smaller categories lead to higher bias. The iterative application of the recommendation algorithms will eventually amplify the data bias, when it is relatively large, even when recommendations are accepted with a probability.

Our findings are confirmed by real data. However, they are characterized by various uncontrolled conditions, such as the uneven distribution of ratings among users of a group, and we need to explore them more.

Finally, we introduced two bias correcting algorithms that re-rank the results of recommenders in order to minimize the bias disparity. The GULM algorithm produces new recommendations that minimize the utility loss for the group of users while the MULM minimizes the maximum over all users loss. We found that the correcting algorithms are efficient and in the long-term they avert large bias disparity.

We view this analysis as a first step towards a systematic analysis of the factors that cause bias disparity. We intend to investigate more recommendation algorithms, and the case of numerical, rather than binary, ratings. Also, we want to examine alternative definitions of bias.

Bibliography

[1] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, "Machine bias," *ProPublica*, 23 May 2016.

[2] J. L. SKEEM and C. T. LOWENKAMP, "Risk, race, and recidivism: Predictive bias and disparate impact*," *Criminology*, vol. 54, no. 4, pp. 680–712, 2016.

[3] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, vol. 2009, pp. 4:2–4:2, Jan. 2009.

[4] M. Day, "How linkedin's search engine may reflect a gender bias," *The Seattle Times*, 31 August 2016.

[5] S. Hajian, F. Bonchi, and C. Castillo, "Algorithmic bias: From discrimination discovery to fairness-aware data mining," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 2125–2126, ACM, 2016.

[6] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, (New York, NY, USA), pp. 214–226, ACM, 2012.

[7] B. Lepri, J. Staiano, D. Sangokoya, E. Letouzé, and N. Oliver, "The tyranny of data? the bright and dark sides of data-driven decision-making for social good," *CoRR*, vol. abs/1612.00323, 2016.

[8] A. Mowshowitz and A. Kawaguchi, "Measuring search engine bias," *Inf. Process. Manage.*, vol. 41, pp. 1193–1205, 2005.

[9] S. Fortunato, A. Flammini, F. Menczer, and A. Vespignani, "Topical interests and the mitigation of search engine bias," *Proceedings of the National Academy of Sciences*, vol. 103, no. 34, pp. 12684–12689, 2006.

[10] T. Bolukbasi, K. Chang, J. Y. Zou, V. Saligrama, and A. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," *CoRR*, vol. abs/1607.06520, 2016.

[11] G. Pleiss, M. Raghavan, F. Wu, J. M. Kleinberg, and K. Q. Weinberger, "On fairness and calibration," *CoRR*, vol. abs/1709.02012, 2017.

[12] M. B. Zafar, I. Valera, M. Gomez Rodriguez, and K. P. Gummadi, "Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment," in *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, (Republic and Canton of Geneva, Switzerland), pp. 1171–1180, International World Wide Web Conferences Steering Committee, 2017.

[13] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, (USA), pp. 3323–3331, Curran Associates Inc., 2016.

[14] M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi, "Fairness Constraints: Mechanisms for Fair Classification," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 962–970, PMLR, 20–22 Apr 2017.

[15] O. Celma and P. Cano, "From hits to niches?: Or how popular artists can bias music recommendation and discovery," in *Proceedings of the 2Nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, NETFLIX '08, (New York, NY, USA), pp. 5:1–5:8, ACM, 2008.

[16] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Correcting popularity bias by enhancing recommendation neutrality," in *Poster Proceedings of the 8th ACM Conference on Recommender Systems, RecSys 2014, Foster City, Silicon Valley, CA, USA, October 6-10, 2014*, 2014.

[17] G. Adomavicius, J. Bockstedt, S. Curley, and J. Zhang, "De-biasing user preference ratings in recommender systems completed research paper," in *24th Workshop on Information Technology and Systems*, University of Auckland Business School, 2014.

[18] M. Kunaver and T. Porl, "Diversity in recommender systems a survey," *Know.-Based Syst.*, vol. 123, pp. 154–162, May 2017.

[19] R. Burke, N. Sonboli, and A. Ordonez-Gauger, "Balanced neighborhoods for multi-sided fairness in recommendation," in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency* (S. A. Friedler and C. Wilson, eds.), vol. 81 of *Proceedings of Machine Learning Research*, (New York, NY, USA), pp. 202–214, PMLR, 23–24 Feb 2018.

[20] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, (Washington, DC, USA), pp. 497–506, IEEE Computer Society, 2011.

[21] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K. Chang, "Men also like shopping: Reducing gender bias amplification using corpus-level constraints," *CoRR*, vol. abs/1707.09457, 2017.

[22] T. Kamishima, S. Akaho, and J. Sakuma, "Fairness-aware learning through regularization approach," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, (Washington, DC, USA), pp. 643–650, IEEE Computer Society, 2011.

[23] S. Yao and B. Huang, "Beyond parity: Fairness objectives for collaborative filtering," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 2921–2930, Curran Associates, Inc., 2017.

[24] A. J. Biega, K. P. Gummadi, and G. Weikum, "Equity of attention: Amortizing individual fairness in rankings," *CoRR*, vol. abs/1805.01788, 2018.

[25] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. A. Baeza-Yates, "Fa*ir: A fair top-k ranking algorithm," *CoRR*, vol. abs/1706.06368, 2017.

[26] A. J. B. Chaney, B. M. Stewart, and B. E. Engelhardt, "How algorithmic confounding in recommendation systems increases homogeneity and decreases utility," *CoRR*, vol. abs/1710.11214, 2017.

[27] C. C. Aggarwal, *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st ed., 2016.

Author's Publications

V. Tsintzou, E. Pitoura and P. Tsaparas, "Bias disparity in recommendation systems", in *International Workshop of Data and Algorithm Bias*, *Conference on Information and Knowledge Management*, CIKM'18, (Turin, Italy), 2018

Short Biography

Virginia Tsintzou was born in 1989. She comes from Zitsa, Greece, but she grew up in Ioannina. She received her Engineer's diploma in 2016 from the Department of Computer Science & Engineering in the University of Ioannina. Her diploma thesis has title "Analysis and Prediction of Election Results with Twitter" and was completed under the supervision of Professor Panayiotis Tsaparas. She continued her studies as a M.Sc. student at the same department, focusing on Data Mining. She works under the supervision of P.Tsaparas and the guidance of Professor Evaggelia Pitoura. She is expected to graduate in 2018.