CSE 575 Final Project Report: Causal Inference for Recommender System

Dongqi Fu, Lu Cheng, Zhe Xu, Fumin He, Weihai Shen, Shuo Wang, Yilun Huang

April,26 2019

1 Introduction

Online commerce has presented unprecedented momentum in recent years and has outpaced the growth of traditional business. The recommendation functionality of online e-commerce such as Amazon, YouTube or Netflix is now accounting for around 35% of overall sales, in the case of Amazon [1]. Consequently, research work on recommender system within computer science has grown significantly in parallel.

The goal of a recommender system is to infer users' preferences for various items and recommend items that users are most likely to buy. Therefore, a standard input data for a recommender system then consists of two sources of information: items that users have looked at and items they actually liked/bought. For instance, for movies recommendation, the input data contains which movies each user watched and movies she liked. Another type of information that most datasets does not include is the social network represented by follow/followedby or trust/trustedby relationship. While recommender systems often use the click/rating data alone to infer the user preferences, a serious issue with this observational data is it is sort of contaminated by users' *selection biases*. For example, in a movie recommendation system, users watch and rate those movies they prefer, and rarely rate movies that they don't like [2]. Another example is when ad-placement system recommends ads, it shows ads that it believes to be of interest to the user, but will less often display other ads. Learning with the effect we would like to optimize (e.g., ratings) can lead to data that is Missing Not At Random (MNAR) [3].

In this project, we try to correct the selection biases in the observational data. Different from existing de-biased recommender system, we seek to de-bias the input data directly without changing the traditional recommender systems. Here, we leverage information extracted from the social networks. In particular, we seek to diversify the input data. Suppose we have a group of diverse users (i.e., users that are different in the social network), and surprisingly, they all are fans of Avengers, we may conclude that Avengers is very popular and similar movies should be recommended to these users. In this project, we propose two methods to diversify users: (1) *sampling* more diverse users using the social network, which further leading to more diverse ratings/clicks data; (2) *reweighting* each user based on the size of communities detected from the social network. The first method is easier to implement but the data size may be reduced. The second approach keeps the datasets same size but it needs to change the loss function in order to incorporate the weights, therefore, is more difficult to implement. Our contribution is we propose a novel and simple way to debias recommender system and experimental results reveal that the proposed method can outperform the baseline models most of the time.

Related Work. There has been a lot of works for recommendation system, [4], [5] and [6] shown different methods to correct the bias in observational recommendation data based on the joint likelihood of the missing data model and the rating model. These are very complex methods. In order to make the approach much easier, Tobias Schnabel [7] designed a fundamentally different approach that treats both models separately. Steck's research [8] is very close to Schnabel's work [7] since Steck defines a recall estimator that uses item popularity as a proxy for propensity.

Propensity-based approaches and Weighting approaches are both wildly used approaches. As for Propensity-based approaches, they are mainly used in causal inference from observational studies [9] and complete-case analysis for missing data [10]. According to [1], the basic idea of propensity scoring method is similar to importance sampling. There are many Propensity-based approaches, [11] introduces a *Clipped Inverse Propensity Scoring* method; [12] proposes a *Double Robust estimation* for *Inverse Propensity Scoring*; [13] and [14] introduce the *Counterfactual Risk Minimization* principle.

Furthermore, when it comes to domain adaptation and covariate shift, Weighting approaches will be wildly used. [15] and [16] make the causal inference problem into a domain adaption problem; [17] construct the causal inference problem to a covariate shift problem and propose a new method to get a better prediction.

Focusing on unbiased evaluation of the recommendation policy using biased data, [18], [19] and [20] solved recommendations using causality. Their method is very similar to ours, they typically uses importance sampling, weighting the probability of each observed click under the logging policy and under the recommendation policy [21]. And in this project, we also follow the sampling and reweighting mechanism for data. [21] is also similar to [18], [19] and [20]. But [21] focuses more on learning preferences rather than evaluating recommendation policies.

The remainder of this report is organized as follows: Section 2 introduces a formal definition of the problem of causal recommender system. Section 3 describes the proposed method in detail. Section 4 describes the datasets, settings, and results of our experimental evaluation. Section 5 concludes the paper and discusses key avenues for future work.

2 Problem Description

In this section, we give a formal definition for de-biased recommender system. We begin with some notations. Suppose we have a group of users $\mathcal{U} = \{1, 2, ..., U\}$ and a set of items $\mathcal{I} = \{1, 2, ..., I\}$. Each user is indexed by u and an item is indexed by i. The click/ratings data is the history of users clicking/rating an item, denoted as y_{ui} , an indicator of whether user u clicked on (liked) item i or decided to skip (disliked) the item. For rating data, y_{ui} is a positive value, e.g., 1, 3. These data capture the users' clicks/ratings. In addition, we are given the social network of all users. Given a social network $G = (V, E)$, where V is the set of users, E is the set of connections between users, e.g., trust/trustedby or follow/followedby. Now we define the problem of de-biased recommender system as follows:
Given the rating matrix Y from U users and I items, the graph $G = (V, E)$, the goal is to train a de-biased recommender system that accounts for users' selection biases to better predict the user preferences. In particular, during the learning phase, we would like to tailor the graph information to sampling more diverse users $\mathcal{U}_D$ or reweighting the input ratings from each user.

3 Methodology

3.1 Opinion leadership selection phase

The study of opinion leadership has its origins in work by Lazarsfeld, Berelson and Gaudet (1948) in which they discovered that voting decisions were heavily influenced by relatives, friends and co-workers. Rogers (1962) defined opinion leadership as 'the degree to which an individual is able to influence other individuals' opinions or behavior in a preferred way with relative frequency' (Jamrozy, Backman & Backman, 1996). In our project, we will find those opinion leaders to eliminate inherent bias in our datasets.

At first, we divide our users from social networks into several clusters in which each cluster has its own opinion leader who has huge influence on its clustering users by **Opinion Leader Algorithm**. The algorithm described as follows [1]

We adopt the algorithm proposed by Vincent D Blondel(2008) to extract the community structure of the big networks which outperforms all other known community detection method at that time. One primary reason we use it is that the quality of the communities detected is very good. This algorithm is divided in two phases that are repeated iteratively. Assume that we start with

Algorithm 1 Locating Opinion Leaders in the social networks

```
Input: Social Network Graph $G = (V, E)$, where $V$ is the set of users, $E$ is
    the set of connections between users.
Output: Opinion Leaders list.
1: Run clustering algorithms to obtain the global optimal number of clusters,
   $c_j$, $j = 1, \dots, k$.
2: for j = 1 to k do
3:     Sort the users in $c_j$ descendingly according to their degree.
4:     Opinion Leaders list appends top 1% users of $c_j$.
5: end for
6: return Opinion Leaders list
```

a weighted network of $\boldsymbol{N}$ nodes. First, we assign a different community to each node of the network. So, in this initial partition there are as many communities as there are nodes. Then, for each node i we consider the neighbours j of i and we evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of j. The node i is then placed in the community for which this gain is maximum (in case of a tie we use a breaking rule), but only if this gain is positive. If no positive gain is possible, i stays in its original community. This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first phase is then complete.

The next step, we eliminate all their rating records from rating data. For example, if we can obtain the global partition, and find the opinion in each cluster Figure [1], we could locate their record coordinately. When their record is using for collaborative filtering, we can control its weight to a lower level when used for users recommendations from other clusters.

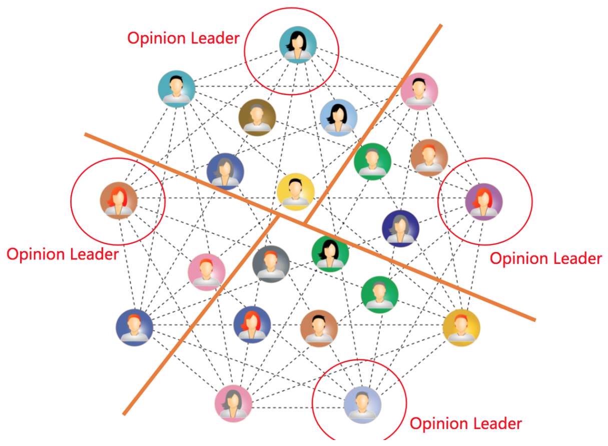


Figure 1: Opinion leaders in each cluster

3.2 User weighting with community detection

Here, we give details of the weighting method based on community detection. This is really inspired by the Inverse Probability Weighting [9] in causal inference, a statistical technique for calculating statistics standardized to a population different from that in which the data was collected. It is often used to reduce the bias of unweighted estimators and often calculated by the inverse of the data size.

Based on this, a simple solution is to use community detection algorithm to find different communities in the graph and obtain the weights for each user by simply inversing the size of each community. Community detection is that if the nodes of the network can be easily grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally. The more general definition is based on the principle that pairs of nodes are more likely to be connected if they are both members of the same community(ies), and less likely to be connected if they do not share communities. Suppose we the size of a community is C_1, then the weights for all the users in this community are $\frac{1}{C_1}$. At last in the training phase, we multiply the user weight with the prediction loss. The final loss function is then a sum of weighted loss.

$$Loss = \sum_{u=1}^{U} \sum_{i=1}^{I} w_u \cdot (y_{ui} - \hat{y}_{ui})^2 + \alpha \mathcal{R}, \tag{1}$$

where w_u is the weight for user u, $\mathcal{R}$ is the regularization for model complexity and α controls the balance between prediction error and regularization.

4 Experiment

4.1 DATASETS AND DATA ANALYSIS

The dataset used in our experiments are Ciao and Douban from two popular product review sites Ciao[1] and Douban[2]. On both sites, people not only write critical reviews for various products but also read and rate the reviews written by others. Furthermore, people can add members to their trust networks or “Circle of Trust”, if they find their reviews consistently interesting and helpful.

In these two datasets, both of them contain two files with same format: ratings.txt and trusts.txt. For ratings.txt, every line has the format: **user_id item_id rating_value**. For example, **23 387 5** represents the fact ”user 23 has rated item 387 as 5”. For trusts.txt, every line has the format: **source_user_id target_user_id trust_statement_value**. For example, **22605 18420 1** represents the fact ”user 22605 has expressed a positive trust statement on user

[1]http://www.ciao.co.uk
[2]http://movie.douban.com

18420".

1) Ciao dataset: Guo et al. [22] crawled Ciao DVD ratings and trust values for a small dataset. In this dataset, users can write textual reviews to products that they purchased or used in the past; and some other users can rate the helpfulness of these reviews in terms of rating scales. If one's reviews are consistently valuable to a specific user, the user may specify the review writer as trustworthy and add him/her in the trust list. Here are some statistics.

* 284,086 ratings (1–5 scale) rated by
* 7,375 users on
* 105,114 items, and
* 111,781 trust statements

2) Douban Movie dataset: Douban Movie is a Chinese website that allows Internet users to share their comments and viewpoints about movies. Users are able to post short or long comments on movies and give them marks. This dataset mainly focus on the short comments of Douban Movie. It is crawled by G. Zhao [23]. This dataset contains

* 894,887 ratings (1–5 scale) rated by
* 2,848 users on
* 39,586 items, and
* 35,770 trust statements.

These two datasets reflect typical social rating networks and are often adopted by previous studies [22, 24–26]. Figure 2 presents the visualization of the social networks in both datasets. As we can see that both networks are condense and from the results, we actually find that the *Douban* dataset is fully connected.

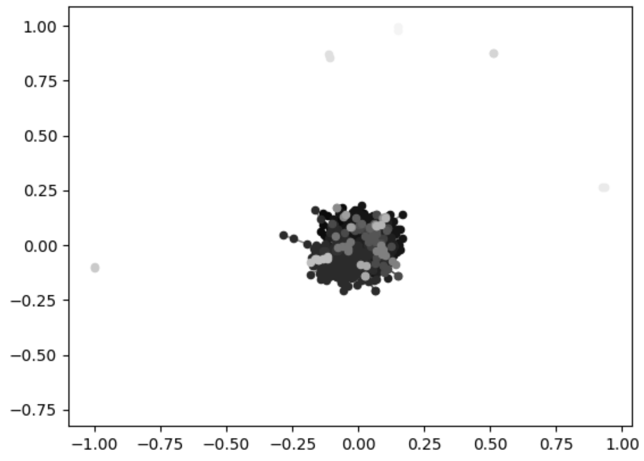
4.2 Metrics

The performance of our algorithm will be embodied by the errors. In our experiments, we introduce two of the most common metrics used to measure accuracy for continuous variables: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

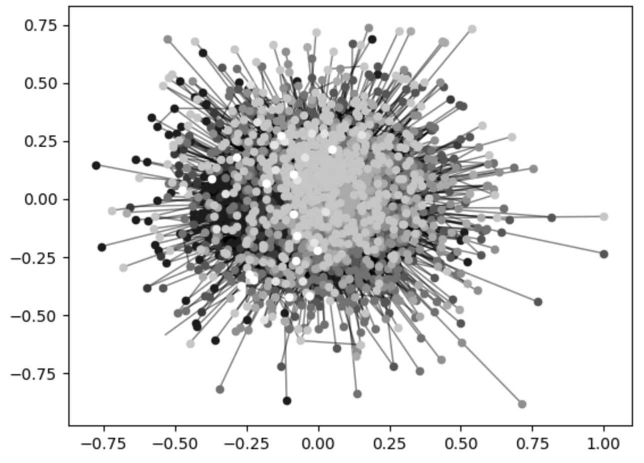
MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$MAE = \sum_{(u,i)\in\mathbb{A}_{\text{test}}} \left|R_{u,i} - \hat{R}_{u,i}\right| / |\Re_{\text{test}}| \tag{2}$$

RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences



(a) *ciao* dataset



(b) *Douban* dataset

Figure 2: Visualization of networks for both datasets.

between prediction and actual observation.

$$RMSE = \sqrt{\sum_{(u,i)\in\Re_{\text{tost}}} \left(R_{u,i} - \hat{R}_{u,i}\right)^2 / |\Re_{\text{test}}|} \tag{3}$$

where $R_{u,i}$ is the real rating value of user u to item i, $\hat{R}_{u,i}$ is the corresponding predicted rating value. $\Re_{\text{test}}$ is the set of all user-item pairs in the test set. $|\Re_{\text{test}}|$ denotes the number of user-item pairs in the test set.

4.3 Baseline Algorithms

We select four algorithms as our baselines: UserMean, UserKNN, Basic matrix factorization, and EE. Here we will give a brief introduction about all those algorithms.

4.3.1 UserMean

As the name shows, it is an extremely easy algorithm. When decide the rating from a user to an unknown item, the rating is the average rating of the user's history.

4.3.2 UserKNN

KNN is a famous and simple classification algorithm. UserKNN works based on that which means that the rating for test set data is based on the most similar users' rating in training set. The similarity between two users can be used as Pearson correlation coefficient, Reciprocal of Euclidean distance, or Cosine similarity, and the features are their rating history. After we have the similarity for the most K similar users, we can predict the rating for test set based on the following formula:

$$R_j = \frac{\sum_i^K Similarity(U_i, U_j) * AverageRating(U_i)}{\sum_i^K Similarity(U_i, U_j)} \tag{4}$$

4.3.3 BasicMF

Matrix factorization is a classic and basic recommender system algorithm, and seriously it is a model-based method, which means that we need to train a model and actually the model in this method is the latent vector presentation of each users and items in a same dimension space. For example, a user i is represented as a $1 \times d$ vector $user_i$ and item j is represented as a $d \times 1$ vector $item_j$. Then we have the predict rating from user i to item j as:

$$\hat{r_{ij}} = user_i \cdot item_j \tag{5}$$

and the training process is try to minimize the error between r_{ij} and $\hat{r_{ij}}$.

4.3.4 SVD

SVD is an modified algorithm from Basic matrix factorization, which also consider the average bias from user and items. It is obvious that some users may rate all the items high, and some items may get high ratings no matter who rates them. For this algorithm the prediction ratings from user i to item j is:

$$\hat{r_{ij}} = user_i \cdot item_j + globalBias + UserBias(i) + ItemBias(j) \tag{6}$$

4.4 Results

Here we present the experiment results for four baseline algorithms under our sampling framework and without our framework on two datasets in Tab. 1 and 2.

It shows that our framework can handle most of algorithms but not so well-done on every dataset. For that, we have some analyses:

- The objective of debiasing and accurate prediction have some conflicts.
- Removing the leader points may not be the best solution to debias
- Our existing community detection algorithm can't detect the community with almost same bias.

	MAE	**RMSE**
UserKNN	0.780	**1.045**
UserKNN-debiased	**0.784**	1.043
BasicMF	**0.788**	**1.061**
BasicMF-debiased	0.785	1.057
EE	**0.805**	**1.090**
EE-debiased	0.804	1.088
UserMean	**0.782**	**1.033**
UserMean-debiased	0.788	1.039

Table 1: Experiment on Ciao Dataset

5 Conclusions & Future Work

We design a framework to eliminate the bias for recommendation systems by reducing the influence of some opinion leader and reweighting. Moreover, we provide a method for locating opinion leaders. Then, we applied our methodology to the real world data set with the support several efficient recommend

	MAE	**RMSE**
UserKNN	0.703	0.884
UserKNN-debiased	**0.701**	**0.883**
BasicMF	0.625	**0.801**
BasicMF-debiased	**0.622**	0.803
EE	**0.605**	**0.777**
EE-debiased	0.607	0.779
UserMean	0.699	0.873
UserMean-debiased	**0.696**	**0.869**

Table 2: Experiment on Douban Dataset

system models. Finally, the experiments show the effectiveness of our framework.

For future work, it is interesting to figure out why reweighting method does not work properly: is it because of the parameters setting? In addition, more advanced sampling methods can be explored such as determinantal point processes [27].

References

[1] Stephen Bonner and Flavian Vasile. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 104–112. ACM, 2018.

[2] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. *arXiv preprint arXiv:1602.05352*, 2016.

[3] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. Wiley, 2019.

[4] Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'07, pages 267–275, Arlington, Virginia, United States, 2007. AUAI Press.

[5] Benjamin M. Marlin and Richard S. Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 5–12, New York, NY, USA, 2009. ACM.

[6] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31st International Conference on International Confer-*

ence on Machine Learning - Volume 32, ICML'14, pages II–1512–II–1520. JMLR.org, 2014.

[7] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. *CoRR*, abs/1602.05352, 2016.

[8] Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 125–132, New York, NY, USA, 2011. ACM.

[9] Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction.* Cambridge University Press, 2015.

[10] Shaun R Seaman and Ian R White. Review of inverse probability weighting for dealing with missing data. *Statistical Methods in Medical Research*, 22(3):278–295, 2013. PMID: 21220355.

[11] Léon Bottou, Jonas Peters, Joaquin Qui nonero Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14:3207–3260, 2013.

[12] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *CoRR*, abs/1103.4601, 2011.

[13] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16:1731–1755, 2015.

[14] Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. *CoRR*, abs/1502.02362, 2015.

[15] Fredrik D. Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *33rd International Conference on Machine Learning, ICML 2016*, volume 6, pages 4407–4418. International Machine Learning Society (IMLS), 2016.

[16] Uri Shalit, Fredrik D. Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3076–3085, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[17] Nir Rosenfeld, Yishay Mansour, and Elad Yom-Tov. Predicting counterfactuals from large historical data and small randomized trials. *CoRR*, abs/1610.07667, 2016.

[18] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, abs/1003.0146, 2010.

[19] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Conference on information &#38; knowledge management*, CIKM '13, pages 1411–1420, New York, NY, USA, 2013. ACM.

[20] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 929–934, New York, NY, USA, 2015. ACM.

[21] Dawen Liang, Laurent Charlin, and David M. Blei. Causal inference for recommendation. 2016.

[22] G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith. Etaf: An extended trust antecedents framework for trust prediction. In *Proceedings of the 2014 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 540–547, 2014.

[23] X. Qian G. Zhao and X. Xie. User-service rating prediction by exploring social users' rating behaviors. volume 18, pages 496–506, 2016.

[24] Guibing Guo, Jie Zhang, and Daniel Thalmann. A simple but effective method to incorporate trusted neighbors in recommender systems. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 114–125. Springer, 2012.

[25] Jiliang Tang, Huiji Gao, and Huan Liu. mtrust: discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 93–102. ACM, 2012.

[26] Nan Ma, Ee-Peng Lim, Viet-An Nguyen, Aixin Sun, and Haifeng Liu. Trust relationship prediction using online product review data. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 47–54. ACM, 2009.

[27] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.