The Dynamics of Recommender Systems

by

Karl M Krauth

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Summer 2022

The Dynamics of Recommender Systems

Abstract

The Dynamics of Recommender Systems

by

Karl M Krauth

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Michael I. Jordan, Chair

Over the past three decades, the reach of recommender systems has grown exponentially. Today, recommender systems are deployed on all major internet platforms, influencing our opinions, decisions, careers, and relationships. However, despite their far-reaching impact, these algorithms and their consequences are still poorly understood. In this thesis, we argue that this is due to the challenging dynamics of the recommendation problem. We outline four problems that distinguish the dynamics of recommendation from other dynamical systems, making them particularly hard to reason about: (1) direct *measurement* and experimentation are often infeasible, (2) *feedback* effects make it difficult to reason about cause and effect, (3) the *scale* of internet platforms requires increased algorithmic complexity, and (4) *incentives* created by recommender systems cause users to behave strategically. We build the foundations necessary to understand and remedy these four problems, paving the way for a complete understanding of the dynamics of recommender systems and their consequences.

To my friends and my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 2012, the size of the internet exceeded a zettabyte (10^{21} bytes) [241]. The internet's growth from a gentle stream to a torrential flood of information spurred the development of new methods to efficiently retrieve relevant data. Today, recommender systems and search engines allow us to find useful content at the click of a button while also filtering out a vast sea of informational noise. The year 2012 was also notable for the field of machine learning. In their seminal paper, Krizhevsky, Sutskever, and Hinton [137] developed a convolutional neural network that achieved a top-5 error rate of 15.3% in the ImageNet 2012 challenge [200], a 10 percentage point improvement over contemporary methods. This impressive feat launched a new wave of investment and development in machine learning.

Since machine learning algorithms excel at processing large amounts of data, new developments in the field were rapidly incorporated into recommender systems and deployed on large internet platforms such as Youtube [47], Facebook [112], Netflix [6], and Twitter [167]. These breakthroughs empower users to interact with recommender systems in new intuitive ways [77] while also improving the inferential accuracy of these systems. Unfortunately, these changes are not universally positive. Recommender systems now permeate every aspect of our lives: affecting our purchasing decisions, our media consumption, our political views, and even our social interactions. As a consequence, the societal impact of recommender systems has come under scrutiny in recent years, with detractors claiming that these systems contribute to societal ills such as addiction, polarization, radicalization, and misinformation [52, 65, 158].

The debate about the culpability of recommender systems is still ongoing [33, 98, 141]. This uncertainty reflects the complex nature of the environments in which these systems operate. Where much of this complexity stems from the dynamics of these environments since they are impacted by the changing preferences of massive and highly heterogeneous populations. Yet most recommendation research ignores the dynamics inherent to the problem, and instead makes the simplifying assumption of a static environment [143, 177, 191, 206, 218, 231, 248]. Given that most visible breakthroughs in machine learning have stemmed from accuracy improvements on standardized static benchmarks [186], this is perhaps unsurprising. However, we argue that the dynamics of recommendation are key to properly understanding recommender systems and their impact, and thus can not simply be ignored.

In this thesis we study the dynamics of recommender systems. We argue that recommendation environments are particularly challenging instances of dynamical systems due to the following four characteristics:

1. **Measurement:** Recommender systems are primarily deployed on large internet platforms where experimentation is often prohibitively expensive. Combined with the closed-source nature of these platforms, it is often infeasible to measure recommendation effects.

2. **Feedback:** Recommendations impact user preferences and interactions, which in turn impact future recommendations, thus creating a feedback loop that is challenging to account for.

3. **Scale:** Given the massive user populations recommender systems must contend with, platforms must deploy complex multi-stage pipelines where each stage is composed of a different filtering algorithm. These pipelines are much harder to analyze in unison than their individual components.

4. **Incentives:** Exposure on internet platforms is often tied to monetary compensation, incentivizing certain users to act strategically. Since recommender systems play a moderating role in directing user attention, users will adapt their behavior to the recommendation algorithm to maximize their exposure. These strategic agents will often act in a seemingly unpredictable fashion when their incentives aren't considered.

We will explore how to manage all four of these issues in subsequent chapters. However, we will begin by taking a deeper dive into each characteristic, explaining how they arise and what makes them uniquely challenging to handle.

1.1 The Measurement Problem

Developing a principled understanding of recommendation algorithms and their effects requires the ability to test hypotheses through experimentation. Unfortunately, experiments are particularly challenging to develop in this regime. Since recommender systems are primarily deployed on large internet platforms, experimental interventions often come at a prohibitively high engineering cost, often requiring the collaboration of multiple teams. For example, we might wish to understand whether expanding users' horizon by recommending novel content will increase their happiness. However, this would require making intrusive modifications to a large software system. Furthermore, the experiment isn't guaranteed to increase user engagement with the platform: potentially causing a loss in revenue.

To make matters worse, many quantities such as user happiness and content quality are only observable through indirect proxy measures such as click-through rate and ratings. And this is in the ideal case where a researcher is one of the lucky few to have direct access to these measures. Due to privacy concerns this data is often only available to researchers working at the company that owns the platform being experimented on. Most researchers must settle for even more indirect measurements that are publicly accessible to all users.

It should therefore come as no surprise that most researchers instead choose to study recommender systems through a static lens. In this regime, the performance of recommender systems is quantified through *offline metrics* which are usually evaluated with respect to a fixed held-out test set. The hope is that improvements in these offline metrics will reflect performance improvements of the recommender system once it is deployed.

In Chapter 2 we show how to mitigate the measurement problem through the use of simulation. We introduce RecLab: an extensible ready-to-use simulation framework. Using RecLab we study empirically how the effects of distribution shifts and feedback loops impact the validity of contemporary offline evaluation methodologies. We also explore what knowledge can be gained through these studies given the mismatch between simulations and real environments. This work first appeared as a standalone paper [134] which was co-authored alongside Sarah Dean, Alex Zhao, Mihaela Curmei, Wenshuo Guo, Ben Recht, and Michael I. Jordan.

1.2 The Feedback Problem

Feedback loops occur when the output of a system is the input of another system, the second system's output is then routed back as an input into the first system. Reasoning about cause and effect in this regime is often tricky due to the closely coupled nature of the systems. This notion is particularly relevant to the dynamics of recommendation wherein feedback is an inherent property of the dynamics.

Recommender systems alternate between (1) making recommendations (2) collecting user responses to these recommendations, and (3) retraining the recommendation algorithm based on this feedback. During this process the recommender system influences the user behavioral data that is subsequently used to update it, thus creating a feedback loop. Recent work has shown that feedback loops may compromise recommendation quality and homogenize user behavior, raising ethical and performance concerns when deploying recommender systems.

In Chapter 3 we propose the Causal Adjustment for Feedback Loops (CAFL), an algorithm that provably breaks feedback loops using causal inference and can be applied to any recommendation algorithm that optimizes a training loss. Our main observation is that a recommender system does not suffer from feedback loops if it reasons about causal quantities, namely the intervention distributions of recommendations on user ratings. This work first appeared as a standalone paper [135] which was co-authored alongside Yixin Wang and Michael I. Jordan.

1.3 The Scale Problem

In practice, recommender systems need to cater to millions or even billions of users. Each time a user needs a recommendation these systems must filter through a massive catalogue of items in only a few milliseconds. These requirements are too onerous for classical recommendation algorithms. As a result, two-stage recommenders have become a popular solution to the problem of scale.

Thanks to their scalability, two-stage recommenders are used by many of today's largest online platforms. These systems produce recommendations in two steps: (1) multiple nominators—tuned for low prediction latency—preselect a small subset of candidates from the whole item pool; (2) a slower but more accurate ranker further narrows down the nominated items, and serves to the user. Despite their popularity, the literature on two-stage recommenders is relatively scarce, and the algorithms are often treated as mere sums of their parts. Such treatment presupposes that the two-stage performance is explained by the behavior of the individual components in isolation.

In Chapter 4 we use synthetic and real-world data to demonstrate that interactions between the ranker and the nominators substantially affect the dynamics of recommendation. Motivated by these findings, we derive a generalization lower bound which shows that independent nominator training can lead to performance on par with uniformly random recommendations. We find that careful design of item pools, each assigned to a different nominator, alleviates these issues. As manual search for a good pool allocation is difficult, we propose to learn one instead using a Mixture-of-Experts based approach. This work first appeared as a standalone paper [101] which was co-authored alongside Jiri Hron, Michael I. Jordan, and Niki Kilbertus.

1.4 The Incentives Problem

Content creators compete for user attention. Their reach crucially depends on algorithmic choices made by developers on online platforms. To maximize exposure, many creators adapt strategically, as evidenced by examples like the sprawling search engine optimization industry. This begets competition for the finite user attention pool.

This strategic behavior is problematic when it comes to designing recommender systems. Changes in the recommendation algorithm can influence the type of content that is produced in unexpected and sometimes harmful ways. Furthermore, strategic behavior is inherently harder to model since we must account for the effect of the recommendation model on the creator's incentives.

In Chapter 5 we formalize these dynamics in what we call an exposure game, a model of incentives induced by algorithms including modern factorization and deep two-tower architectures. We prove that seemingly innocuous algorithmic choices significantly affect the existence and character of Nash equilibria in exposure games. We then use our creator behavior models as a pre-deployment audits. Such an audit can identify misalignment between desirable and incentivized content, and thus complement post-hoc measures like content filtering and moderation. This work first appeared as a standalone paper [103] which was co-authored alongside Jiri Hron, Michael I. Jordan, Niki Kilbertus, and Sarah Dean.

Chapter 2

Evaluating Recommender Systems under Distribution Shifts and Feedback Loops

2.1 Introduction

When machine learning systems are deployed in dynamical settings, their decisions often influence the outcomes they wish to predict. This feedback loop can cause a distribution shift not classically accounted for in supervised learning frameworks. This phenomena has not gone unnoticed by the machine learning community, with recent works characterizing general settings of performativity and concept drift [76, 183]. We take a complementary approach by considering the specific setting of recommendation systems. In particular, we focus on evaluation methodologies and examine their validity under such distribution shifts.

Though the recommender systems community is well aware of the challenges posed by dynamical interactions [133], most recommendation algorithms are primarily designed and evaluated on static datasets in an offline setting [143, 177, 191, 206, 218, 231, 248]. A typical offline-first evaluation methodology involves the following four steps:

1. **Dataset Creation:** An organization or research group creates a dataset by collecting user interactions with a set of items hosted on an internet platform. Two prominent examples are the Netflix Prize and MovieLens datasets.

2. **Offline Evaluation:** Algorithm developers use these datasets to evaluate their recommendation algorithms. They train their algorithm on a training split of the datasets, tune on a validation split, and evaluate on a test split using various offline metrics such as root-mean-squared error (RMSE) and normalized discounted cumulative gain (nDCG).

3. **Comparison with Baselines:** Developers compare these offline metrics with other algorithms, either by running the other algorithms themselves or by referring to previously published work. If their algorithm compares favorably the developers may publish their work, or, if they can, proceed to the next step.

4. **Online Evaluation:** The algorithm is deployed on a real platform and its performance is evaluated using online metrics, which are usually defined to capture some notion of utility. Examples of popular online metrics include click-through rate (CTR) and watch time.

Offline evaluations are sensible given the difficulty of evaluating algorithms online: most researchers do not have access to a platform on which to experiment, and even those that do may not be able to perform large-scale evaluations due to the engineering effort required or the potential lost revenue. However, offline evaluation comes with its own set of challenges. The data collected in step (1) is influenced by the recommender that is deployed at the time, leading to selection bias [205]. Dacrema, Cremonesi, and Jannach [51] demonstrate that baseline comparison in step (2) is often performed incorrectly, with practices like inconsistent dataset splitting leading to non-reproducible results. Furthermore, they show that the baselines in step (3) are often incorrectly tuned, leading to a false sense of progress, a finding corroborated by Rendle, Zhang, and Koren [192]. The difficulty of properly evaluating algorithms offline brings into question the relationship between steps (1) to (3) and step (4).

In this work, we study the validity of this evaluation pipeline in controlled simulated environments. We focus on the effect of the offline dataset and the relationship between online and offline evaluation. We aim to answer three main questions:

1. Does the distribution of offline datasets impact the conclusions of algorithm comparison?
2. Under what conditions can offline metrics reliably predict a recommender system's online performance?
3. Do feedback effects have a similar impact on differing recommender systems?

To answer these questions we evaluate eleven recommendation algorithms across six simulated environments.

By using large-scale simulations, we are able to isolate specific effects devoid of any confounding factors. The recommenders we evaluate encompass simple baselines (`Random`, `Oracle`, `TopPop`), neighborhood-based models (`ItemKNN`, `UserKNN` [107, 231]), kernel-based models (`LLORMA` [143]), linear models (`EASE` [218]), factorization models (`SGF MF`, `Bayes MF` [191]), and neural models (`CF-NADE` [248], `AutoRec` [206]). The simulated environments on which we evaluate represent a diverse set of scenarios including two settings implemented in prior work and one setting that is initialized using the MovieLens dataset [90]. We give further details of each environment in Section 2.3, and further details of each recommender system in the supplementary.

We begin by investigating the effects of shifts in the sampling distribution of offline datasets. We confirm and expand the finding of Schnabel et al. [205], demonstrating that biased sampling can lead to biased evaluation metrics for a wide range of recommendation algorithms. However, we find that the relative ordering of different recommendation algorithms is stable under these sampling shifts. We therefore do not find evidence that these biases affect the validity of the comparison.

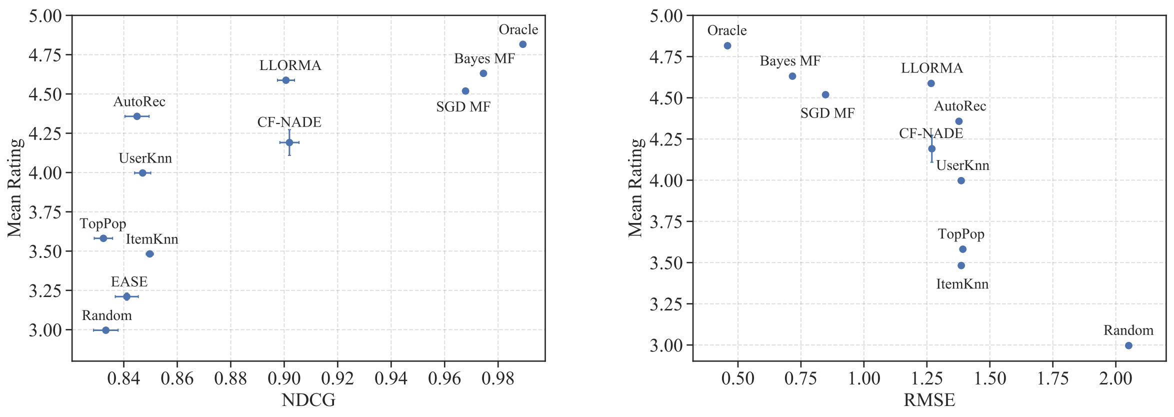


Figure 2.1: Left: The nDCG@20 plotted against the mean user ratings of all recommended items on the `topics-dynamic` environment. Right: The RMSE plotted against the mean user ratings. nDCG and RMSE are averaged across five folds on the offline dataset associated with the environment, user ratings are averaged across ten trials. Each point represents a single model evaluation with error bars representing 95% confidence intervals.

We then show that offline metrics can act as a good proxy for online performance by replicating the offline-first evaluation methodology in a controlled setting. We compute RMSE and nDCG on an offline dataset for each recommender, simulate the interactive process of recommendation, compute the average user ratings of recommended items, and then compare the offline and online metrics. Figure 2.1 shows such a comparison on the `topics-dynamic` environment. While there is a strong correlation between nDCG and mean user rating, we note that improvements in nDCG past a certain point suffer from diminishing improvement in mean user ratings. We examine these effects in more details in Section 2.6.

Taken together, our large-scale simulation results suggest that offline metrics are a reliable tool to supplement online evaluation. However, they also bring into question the value of chasing small improvements in predictive performance, since those results might only lead to very small noisy improvements in the online setting. This is especially true when data is plentiful and state-of-the-art recommendation algorithms can predict near-optimally. Instead, researchers should focus on a holistic evaluation of their algorithms; taking into account metrics that look beyond predictive accuracy and considering issues of measurement and sampling in the construction of the datasets they use.

Since there are many interesting research questions that can be studied through simulation, we open-source our simulation package. Our package is designed with large-scale evaluation in mind, and reproduces a number of popular and state-of-the-art recommenders. We also make available many environments, while making it easy to extend the package with new ones. A schematic view of the simulation and evaluation framework is shown in Figure 2.2. It is our hope that this package can be integrated into the development and evaluation of

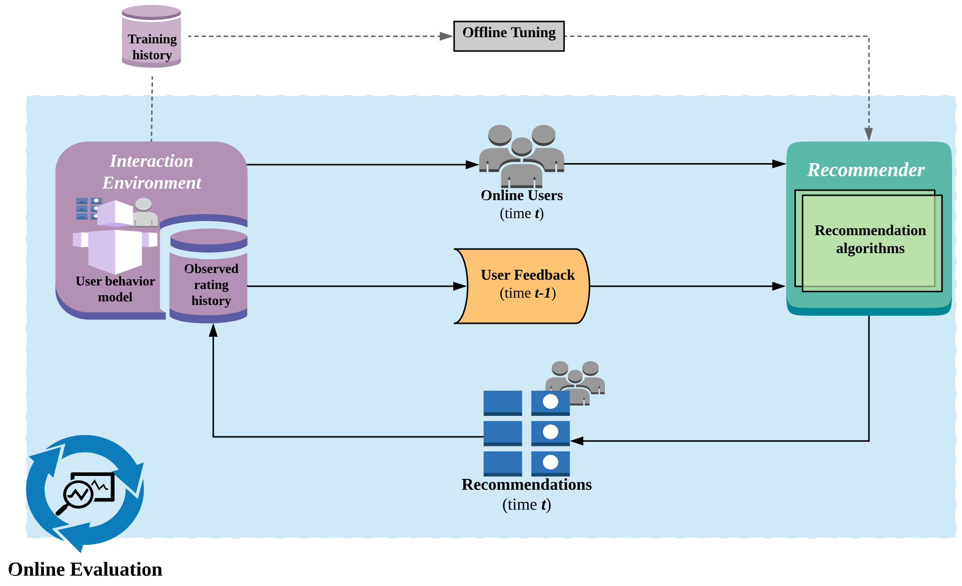


Figure 2.2: Illustration of the RecLab evaluation pipeline.

recommendation algorithms.

2.2 Related Work

Other simulations There has recently been increased interest in studying recommenders through simulation. Chaney, Stewart, and Engelhardt [43] propose an environment in which users have limited knowledge of their utility, and show that recommendation algorithms homogenize user behavior within their simulations. Schmit and Riquelme [204] evaluate the performance of ridge regression and matrix factorization in a two timestep simulated dynamical setting. Ie et al. [110] and Rohde et al. [195] both propose simulation frameworks that are focused on evaluating reinforcement-learning based recommenders. Mansoury et al. [158] and Jiang et al. [115] use simulation to study the negative effects of feedback loops in recommender systems. Our work distinguishes itself from prior simulation studies by being the first to investigate a wide range of recommendation algorithms across a large number of environments. Furthermore, we are the first to investigate simulations at the scale of common benchmark datasets, while still running for many timesteps.

Online recommendation Several empirical studies on deployed recommender systems identify inconsistencies in online and offline performance [14, 172, 196], while others show how richer sets of offline metrics can be used to predict online performance [157]. Our work brings a systematic lens to this problem to understand more broadly the validity of practices around algorithm design and evaluation.

Many recommendation algorithms have been designed to address the difficulty of online recommendation in dynamical environments. Classically, time-aware models exploit the sequential nature of recommendation by incorporating temporal context [37, 130, 229]. More recently, a body of work treats online recommendation as a causal inference problem where the recommender model must de-bias logged training data [161, 201, 205, 215]. Lastly, others seek to improve recommender systems by directly addressing the online problem either through exploration strategies or reinforcement learning algorithms [45, 111, 125, 146].

Assessing metrics Our work complements research on the societal effects of recommender systems, which considers alternative metrics including diversity, utility, serendipity and fairness [2, 69, 176, 213, 242]. Many authors have examined the limitations of accuracy as the sole metric for evaluating recommenders and have sought to define alternate metrics. Herlocker et al. [93] proposed a variety of metrics for assessing a recommender's coverage, diversity, novelty, and serendipity, while Kaminskas and Bridge [120] provide a more recent survey of the approaches for training recommenders with respect to these alternative metrics. Recently, Dean, Rich, and Recht [55] introduced a measure of reachability which combines ideas of coverage with user agency in interactive systems. Finally, recent meta-studies [166, 189, 192] address the reliability of benchmarks in machine learning, and the associated leaderboards of algorithms created by the research community. Our work extends this line of work by also investigating the relationship between these leaderboards and online performance.

2.3 The RecLab Simulated Environments

In this section we summarize the environments that we provide through our simulation framework. We consider both environments where users must consume the single item that is recommended to them and environments where users can choose from a slate of items. Unless mentioned otherwise we set our environments to have 1000 users and 1700 items, which is similar to the MovieLens 100K dataset.

`topics-static` In the `topics-static` environment, each item is assigned to one of K topics and users prefer certain topics. This is similar to the simulation presented by Ie et al. [111]. The preference of user u for items i of topic k_i is initialized as $\pi(u, k_i) \sim Unif(0.5, 5.5)$, while the topic k of item i is chosen randomly from the set of all topics. When user u is recommended item i it will rate the item as $r_t(u, i) = \mathsf{clip}(\pi(u, k_i) + \epsilon)$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ represents exogenous noise not modeled by the simulation, and clip truncates values to be between 1 and 5.

`topics-dynamic` In `topics-dynamic` items are rated in the same way as `topics-static`. In this setting however, user preferences can change as a result of the items they consume.

Since past work has shown that users might become more interested in a topic through repeated exposure [78, 158], we incorporate this phenomenon into our model. If item i is recommended to user u then their preferences are updated as

$$\pi_{t+1}(u,k) \leftarrow \mathsf{clip}(\pi_t(u,k)+a) \quad k=k_i,$$
$$\pi_{t+1}(u,k') \leftarrow \mathsf{clip}\left(\pi_t(u,k') - \frac{a}{K-1}\right) \quad \forall k' \neq k_i,$$

where a is a fixed affinity parameter. Another well-studied phenomenon is the fact that users get bored from being recommended the same topic in a short period of time [124, 236]. We model this as:

$$r_t(u,i) = \mathsf{clip}(\pi_t(u,k_i) - \lambda \mathbf{1}\{\text{topic } k_i \text{ observed} \geq \tau \text{ times within } m \text{ previous timesteps}\} + \epsilon).$$

The effect of boredom arises from three parameters: memory length m, boredom threshold τ, and boredom penalty λ. If a user observes the same topic more than τ times within the last m timesteps then their ratings is penalized by λ.

`latent-static` In the `latent-static` environment, we represent users and items with d-dimensional latent feature vectors. This is a common assumption when developing factor models [15, 131, 132], and allows us to investigate a different user-item representation than the topics-based simulations. The rating of user u on item i is computed using these latent vectors as well as bias terms: $r(u,i) = \mathsf{clip}(\mu_0 + c_u + b_i + \mathbf{p}_u^\top \mathbf{q}_i + \epsilon)$, where $\mu_0 = 3$ is a global bias, $c_u \sim \mathcal{N}(0, 0.25)$ is the bias of user u, $b_i \sim \mathcal{N}(0, 0.25)$ is the bias of item i, $\mathbf{p}_u \sim \mathcal{N}(0, \sqrt{0.5/d})$ is the latent factor of user u, $\mathbf{q}_i \sim \mathcal{N}(0, \sqrt{0.5/d})$ is the latent factor of item i, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. `RecLab` also includes a `latent-dynamic` environment with a similar concept of boredom and affinity change as `topics-dynamic`. However, we do not present experimental results on `latent-dynamic`.

`ML-100K` The `ML-100K` environment is identical to the `latent-static` environment, except that the parameters are generated based on the MovieLens 100K (ML_100K) dataset [90]. We train a LibFM factorization model on the ML_100K dataset, with hyperparameters tuned to achieve low RMSE through cross validation. We then extract the model's biases and latent factors to initialize the environment. We use `ML-100K` to confirm that our experiments generalize to situations where the simulator's parameters are initialized using real user interaction data.

`latent-score` The `latent-score` environment was proposed in Section 5.2.3 of Schmit and Riquelme [204]. In `latent-score` users have partial knowledge of an item's value. They use this information, with the recommender's predicted score, to select an item from a slate of recommended items. We evaluate `latent-score` with 170 users and 100 items due to computational limitations.

`beta-rank` This environment was introduced by Chaney, Stewart, and Engelhardt [43]. It is similar to `latent-score`: users know part of the value for each item and users/items are represented by latent vectors. In `beta-rank` the rating of a user i on an item j is given by $r(u,i) \sim Beta(\mathbf{p}_u^\top \mathbf{q}_i, \sigma^2)$, where $\mathbf{p}_u$ is the latent vector for user i, $\mathbf{q}_i$ is the latent vector for item i, and the Beta distribution is parameterized according to its mean and variance. In this setting users chose from a slate of items based upon their observed utility and the recommender's ranking. We evaluate this environment with 170 users and 100 items due to computational limitations.

2.4 Reproduced Recommenders

We evaluate eleven recommenders including baseline models, neighborhood models, factorization machine models, and several recent deep models. We choose to investigate recommenders mentioned by Rendle, Zhang, and Koren [192] as these models have all been run on the MovieLens10M dataset, giving us a starting point of comparison. The recommendation algorithms that we consider choose items to recommend based on a *relevance prediction*. Except when specified, recommendations are *greedy*, meaning that the item with the highest relevance prediction will be recommended. Therefore, the main difference between the following algorithms is their prediction component. All the models we reproduce only make use of ratings when making predictions, in future work we wish to also evaluated content-based and temporal models. We focus on the settings of rating prediction, where models are tuned so that relevance predictions match ratings (e.g. RMSE).

- `TopPop` - The TopPop algorithm recommends the most popular items to every user without personalization. The popularity of each item is measured by its average rating.
- `ItemKNN` - The ItemKNN algorithm is a collaborative filtering method using k-nearest neighborhood and item similarities [231]. We implement ItemKNN in the same way as Hug [107].
- `UserKnn` - The UserKnn algorithm is identical to ItemKnn, except that it uses user features instead of item features [231].
- `Oracle` - The oracle recommender has access to the internals of the simulated environment, and will recommend the item with the highest true rating at each time step. Since this oracle baseline is still greedy, it does not plan for environment dynamics. Additionally, since actual ratings are generated with some noise, the RMSE of the oracle baseline is not zero.
- `Random` - This baseline predicts ratings uniformly at random.
- `SGD MF` - A factorization machine implemented in LibFM [191]. The model is trained using SGD.

- `Bayes MF` - Another variant of factorization machines implemented in LibFM. The model is trained and the hyperparameters are automatically tuned using MCMC.
- `AutoRec` - An autoencoder framework for collaborative filtering [206]. We train the algorithm with RMSProp and use the item-based version `I-AutoRec`. Our implementation makes use of the source code provided by the authors.[1]
- `CF-NADE` - A neural autoregressive architecture for collaborative filtering [248] trained with Adam. We adapt a publicly available implementation for our experiments.[2]
- `LLORMA` - LLORMA [143] is a generalization of low rank matrix factorization techniques. LLORMA approximates the rating matrix as a weighted sum of low-rank matrices. We adapt a publicly available implementation for our experiments.[3]
- `EASE` - EASE [218] is a linear model designed for sparse data, especially implicit feedback data in recommenders. We do not include this recommender when computing RMSE as it outputs non-normalized relevance scores.

2.5 Effects of Rating Sampling Distribution

The datasets used for the offline evaluation of recommendation algorithms are often collected from deployed systems and are therefore susceptible to sampling biases. How do these biases affect evaluations of algorithm performance? To investigate these effects, we construct three datasets. We derive each of them from the `topics-static` environment, we sample each one using one of three sampling schemes:

`UNIFORM-INIT` - In `UNIFORM-INIT` we sample each user-item pair uniformly.

`ML-100K-INIT` - We ensure `ML-100K-INIT` has a rating distribution that matches a real dataset by fitting two beta distributions to the MovieLens dataset. We fit the first distribution to the magnitude of each rating, and we fit the second distribution to the number of ratings each user has made. We then sample user-item pairs from the joint distribution of the two beta distributions. The probability of selecting user u and item i is thus

$$\mathbb{P}(u,i) = \frac{\mathrm{Beta}(r(u,i),\alpha_1,\beta_1)\,\mathrm{Beta}(\mathrm{rk}(u),\alpha_2,\beta_2)}{\sum_{u,i}\mathrm{Beta}(r(u,i),\alpha_1,\alpha_2)\,\mathrm{Beta}(\mathrm{rk}(u),\alpha_2,\beta_2)},$$

where $\alpha_1, \beta_1, \alpha_2, \beta_2$ are parameters fit to the MovieLens 100K dataset[4], r is the rating function, and rk is a random function that assigns a numerical rank to each each user. We shift and scale both functions between 0 and 1.

`ML-100K-INIT-FLIP` - This distribution is identical to `ML-100K-INIT`, except we input

[1] https://github.com/mesuvash/NNRec
[2] https://github.com/JoonyoungYi/CFNADE-keras
[3] https://github.com/JoonyoungYi/LLORMA-tensorflow
[4] We use $\alpha_1 = 3.61, \beta_1 = 2.57, \alpha_2 = 0.70, \beta_2 = 1.83$.

$1 - r(u, i)$ instead of $r(u, i)$ in the first beta distribution, flipping the distribution of ratings about $r = 3$. This results in a dataset with ratings biased toward lower values.

For each dataset our evaluation procedure is then as follows:

1. We create the dataset by sampling $100,000$ user-item pairs without replacement based on the above sampling scheme. We then query `topics-static` for the corresponding ratings.
2. We tune the hyperparameters on `UNIFORM-INIT` using grid search which minimizes the mean RMSE evaluated using five-fold cross-validation. We use the same hyperparameters for all datasets because we did not observe a meaningful difference in performance from tuning the hyperparameters on each dataset.
3. We evaluate each recommender by averaging the offline metrics (nDCG@20 and RMSE) across five folds.

The left plot of Figure 2.3 shows the nDCG@20[5] of the recommenders on `ML-100K-INIT` compared with `UNIFORM-INIT`, and the right plot shows the nDCG@20 of the recommenders on `ML-100K-INIT-FLIP` compared with `UNIFORM-INIT`.

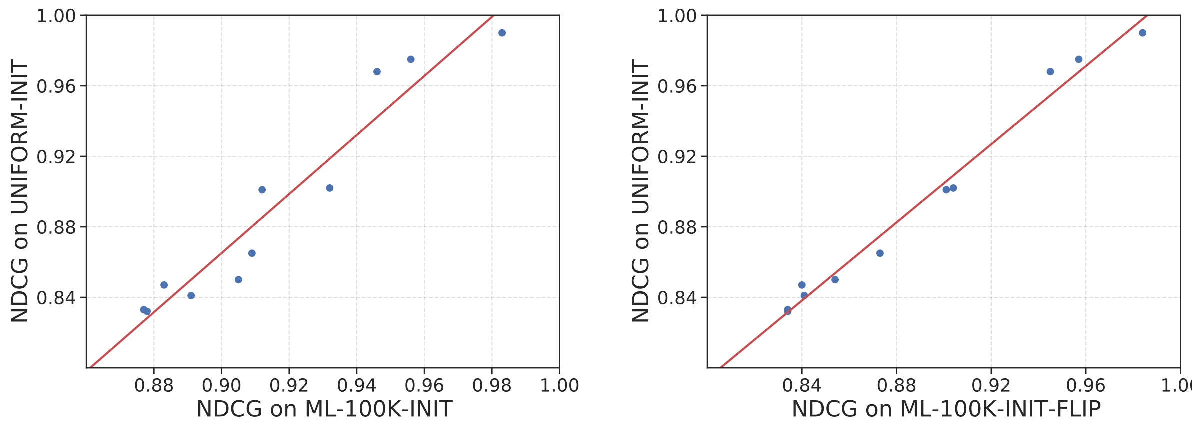


Figure 2.3: Left: The nDCG@20 on `ML-100K-INIT` plotted against the NDCG@20 on `UNIFORM-INIT`. Right: The nDCG@20 on `ML-100K-INIT-FLIP` plotted against the NDCG@20 on `UNIFORM-INIT`. nDCG is averaged across five folds. Each point represents a single model evaluation. A linear fit is shown in red.

Due to the uniform sampling, offline metrics evaluated on `UNIFORM-INIT` are unbiased estimators of a recommender's performance on the entire population of ratings. The nDCGs closely follow a linear function in both plots, indicating that the relative ranking of recommender systems evaluated on `ML-100K-INIT` and `ML-100K-INIT-FLIP` closely corresponds to the relative ranking of these systems with respect to the entire population of ratings. At first glance, this result seems to conflict with Schnabel et al. [205] who show that offline metrics evaluated over biased samples of ratings may result in a biased estimate of

[5]We tried many different values for nDCG@k and obtained near identical behavior for all k.

population-level performance. However, our results on `ML-100K-INIT` are actually in agreement with their findings. The nDCG evaluated on `ML-100K-INIT` is on average higher than the nDCG on `UNIFORM-INIT`. There are two possible reasons for this discrepancy: training data from `ML-100K-INIT` results in better models, or it is easier to rank items correctly on `ML-100K-INIT` than `UNIFORM-INIT`. We eliminate the first possibility by evaluating every model trained with `ML-100K-INIT` on a held-out uniformly-sampled dataset. This results in an nDCG on-par with models trained and evaluated in `UNIFORM-INIT`. Hence, only the second reason remains, which is reflected by the higher nDCG of the `Random` recommender on `ML-100K-INIT`.

We also observe that the nDCG on the `ML-100K-INIT-FLIP` dataset does not lead to a biased estimator of overall performance. These results suggest that the ranking of recommender systems are robust to natural changes in the rating sampling distribution. Furthermore, it is not a given that a non-uniformly sampled dataset will result in a biased estimator of population-level performance for recommender systems.

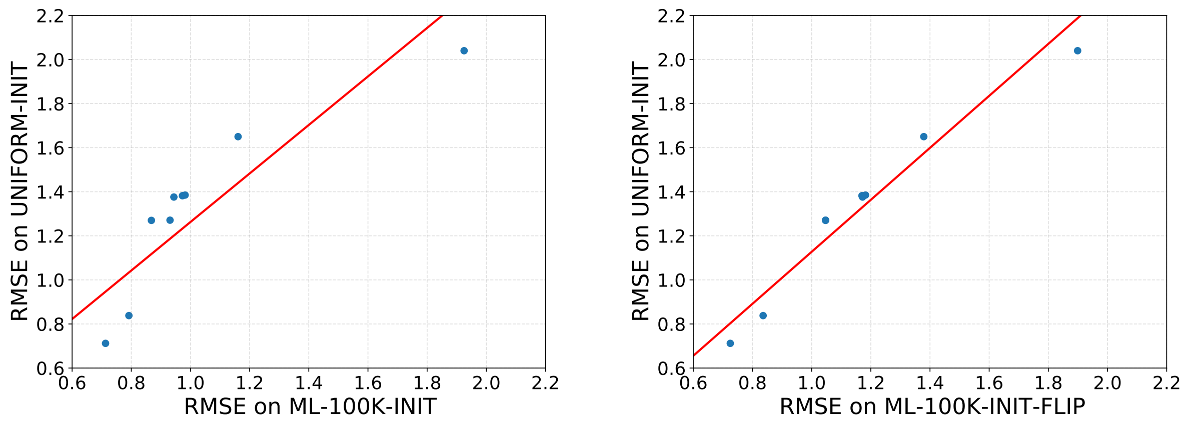
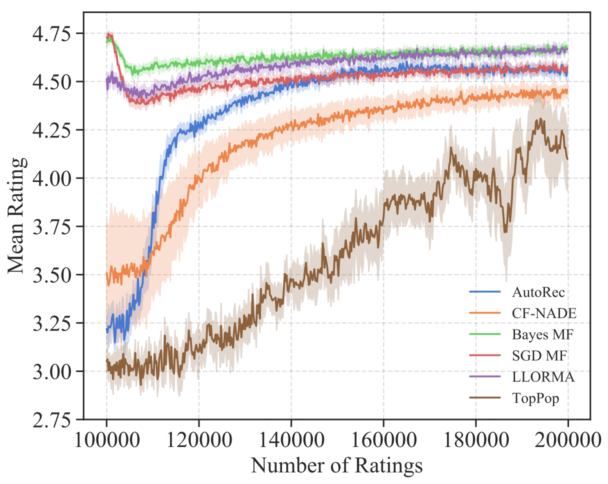


Figure 2.4: Left: The RMSE on `ML-100K-INIT` plotted against the RMSE on `UNIFORM-INIT`. Right: The RMSE on `ML-100K-INIT-FLIP` plotted against the RMSE on `UNIFORM-INIT`. RMSE is averaged across 5 folds. Each point represents a single model evaluation. A linear fit is shown in red.

Figure 2.4 shows the same trend holds for RMSE. Once again, we observe that the ranking of recommenders is robust to the change in sampling distribution. However, the RMSE on `ML-100K-INIT` once again overestimates the performance of models when compared to `UNIFORM-INIT`, while RMSE on `ML-100K-INIT-FLIP` is a better predictor of RMSE on `UNIFORM-INIT`.



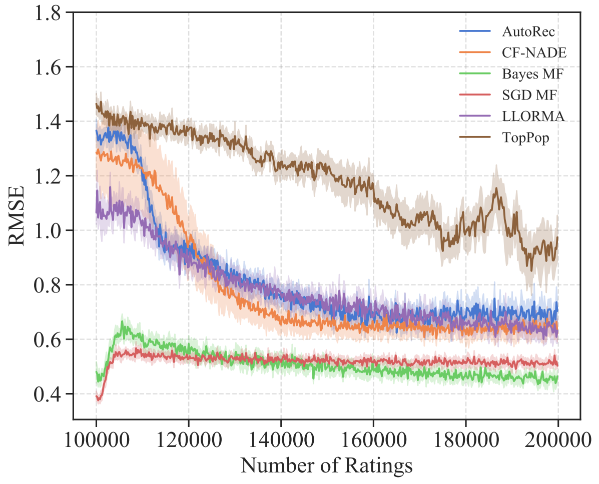


Figure 2.5: The performance of select recommenders over time on the `topics-dynamic` environment. The left plot shows the mean rating of the items that are recommended at each timestep. The right plot shows the RMSE between the predicted and true ratings of the recommended items at each timestep. Both plots are created by averaging over 10 experiment trials. The shaded areas represent 95% confidence intervals.

2.6 Effects of Feedback Loops

Having understood the validity of offline evaluation, we move to considering the distribution shifts that occur during deployment due to feedback and dynamics. We explore the relationship between offline metrics and online performance across all the environments described in Section 2.3. Given an environment, we evaluate each recommender by following these steps:

1. We create an offline dataset by sampling user-item pairs without replacement and get their ratings from the environment. We sample pairs uniformly, removing the sampling bias introduced by data collected when a recommendation algorithm is already deployed. Allowing us to focus on the effects of environment and recommender dynamics.
2. We tune the recommenders on the offline dataset using grid search on the hyperparameters. Our search aims to minimize the mean RMSE evaluated using five-fold cross-validation. We do not re-tune hyperparameters.
3. We begin by training each recommender using our offline dataset. At each timestep, online users are uniformly sampled from the set of all users. We recommend items to the sampled users and observe their ratings. Lastly, we retrain the recommenders on all the acquired data so far, and repeat until the end of the experiment. We repeat for 10 trials for every recommender on each environment.

This method of experimentation closely mimics the offline-first evaluation method mentioned in Section 2.1. Notice that we control for factors which may impact online performance, including the initial sampling distribution and the distribution of online users, so that our results illustrate exclusively the effect of environment and recommender dynamics. Furthermore, we consider simple online deployment, retraining from scratch whenever data is added and using the same hyperparameters throughout. While it is possible to improve these methods, for example by performing just a few gradient steps with new data, we wish to avoid any confounding factors whose effect on performance aren't well understood.

Throughout this section, we consider greedy recommendation: we always recommend the item with the highest predicted relevance that hasn't already been recommended to each user.

Dynamic Environment

In this section we discuss results on `topics-dynamic`, an environment where user's preferences dynamically change over time. We sample 100k initial ratings on which to tune the recommenders, at every timestep we sample 200 users to recommend items to, and we run the simulation until we observe 200k ratings. As shown in Figure 2.1, nDCG@20 and RMSE are predictive of mean user rating. While it is well known that no offline metric can perfectly track online performance, the reason for these discrepancies has not been extensively studied.

We explain some of these discrepancies by exploring the full time series of `topics-dynamic` for a select group of recommenders. The left plot in Figure 2.5 shows the average rating over the recommended items at each timestep, while the right plot shows the RMSE between the predicted and true ratings of the recommended items at each timestep. We emphasize that the RMSE shown in Figure 2.5 is only computed with respect to the recommended items at each timestep, and hence is not a measure of accuracy on the whole population of ratings. We are particularly interested in reasons why two recommenders might have: similar offline nDCG/RMSE but dissimilar mean user rating, similar mean user rating but dissimilar offline nDCG/RMSE, or dissimilar nDCG and RMSE. With these goals in mind we make the following observations:

- **Recommenders can affect user preferences in ways not captured by offline metrics.** `TopPop` improves its performance significantly as time progresses. There is a large difference in mean rating between it and `Random`, despite the fact that both recommenders have very close nDCG. As we discuss in Section 2.6, `TopPop`'s improvements are due to the environment's dynamics.

- **The size of the initial dataset affects the offline ranking of recommenders.** `LLORMA` performs well starting at the first timestep. This seems to contradict its relatively low NDCG on the offline dataset. However, further investigation reveals that LLORMA's NDCG increases to 0.948 when trained with $100,000$ datapoints instead of the $80,000$ available during five-fold cross-validation.

- **Recommender dynamics can affect their performance in ways not captured by offline metrics.** `AutoRec` performs much better than its offline nDCG and RMSE indicate. Unlike `LLORMA`, the initial performance of `AutoRec` is bad. At the first timestep, it barely performs better than `Random`. However, once it observes data that is sampled from its own recommendations, it is able to quickly match the performance of the best recommenders. This result also indicates that, while uniform sampling of ratings allows for an unbiased estimator of performance as discussed in Section 2.5, alternate sampling schemes can lead to improved online performance.

- **The same user dynamics can have a positive effect on one recommender, while simultaneously having a negative effect on another.** `CF-NADE` improves much more slowly than `LLORMA`, explaining the difference in rating between both algorithms despite the similar nDCG.[6] As we show in Section 2.6, this is primarily due to the negative effect of the user dynamics on `CF-NADE`.

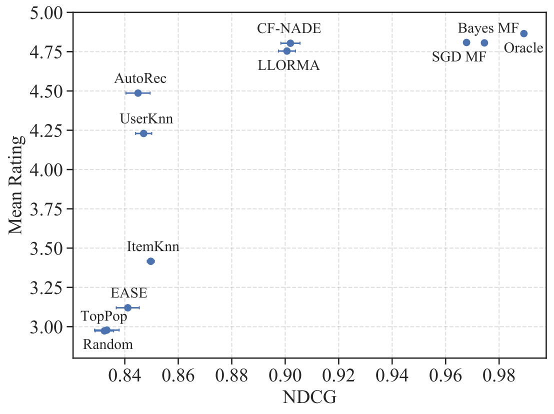
- **A low RMSE is sufficient for selecting high-value items, but it is not necessary.** `SGD MF` and `Bayes MF` both perform on-par with `LLORMA`, despite the fact that `LLORMA` has a worse offline RMSE and per-timestep RMSE. This is because RMSE captures a recommender's ability to predict ratings, whereas mean user rating captures a recommender's ability to identify high-value items.

Static Environment

In this section, we investigate the performance of recommenders on the `topics-static` environment. We focus on comparing these results with those obtained on `topics-dynamic` to disentangle phenomena caused by the environment dynamics and those caused by the recommender dynamics. We initialized `topics-static` with the same user preferences and item topics as `topics-dynamic`. Furthermore, we ensure the offline dataset is the same for both environments. As a result each recommender's hyperparameters are the same as in the `topics-dynamic` setting.

Figure 2.6 compares the nDCG@20 and RMSE of each recommender on the offline dataset with the average user rating across all timesteps. In this setting, nDCG and RMSE are still positively correlated with mean rating, and while most of the observations made in Section 2.6 are still a concern even when there are no environment dynamics at play, we identify two notable differences. First, `TopPop` performs on par with `Random`. As Figure 2.7 shows: without environment dynamics, the underlying preferences remain uniformly distributed, so popularity is not predictive. This is in contrast to `topics-dynamic` where the average preference for each topic is also initialized to be roughly 3, but by the end of the trial with `TopPop`, two of the topics have average affinities higher than 4.5. `TopPop` pushes user preferences toward certain topics, inducing item popularity effects in the data. Furthermore, `CF-NADE` performs significantly better, showing that the same user dynamics can have positive or negative effects on performance depending on the recommendation algorithm.

[6]This observation also holds for `UserKnn` and `ItemKnn` although we do not show all these recommenders in Figure 2.5 to reduce clutter.



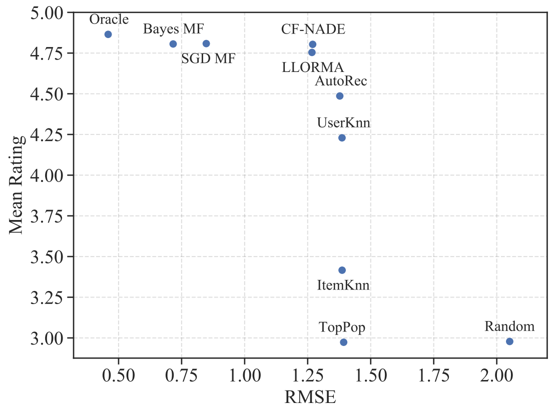


Figure 2.6: Left: The nDCG@20 plotted against the mean user ratings of all recommended items on the `topics-static` environment. Right: The RMSE plotted against the mean user ratings of all recommended items on the `topics-static` environment. NDCG and RMSE are averaged across five folds on the offline dataset associated with the environment, user ratings are averaged across 10 trials. Each point represents a single model with error bars representing 95% confidence intervals.

This emphasizes the importance of evaluating general-purpose recommenders across a diverse range of datasets and environments. We observed similar results when comparing RMSE to mean rating in the `topics-static` environment.

Just as in `topics-dynamic`, `AutoRec` performs much better than its offline nDCG@20 would indicate. We confirm that this improvement is caused by the distribution of the online data `AutoRec` observes by training the model with $200,000$ randomly sampled datapoints. We then evaluate the mean rating of `AutoRec`'s recommended items over a single timestep (repeated over ten different seeds). We observe that the mean rating is 3.718 ± 0.055, which is significantly lower than the recommender's performance in our online experiments.

Other Environments

So far we have only demonstrated that RMSE and nDCG are predictive of online performance for two environments: `topics-static` and `topics-dynamic`. To ensure that this observation is robust, we benchmark the recommendation models across all environments from Section 2.3. The left plot of Figure 2.8 shows the Spearman rank correlations between the nDCG@20 and the mean rating, while the right plot shows the correlations between the RMSE and the mean rating. We see that both nDCG and RMSE are predictive of online performance across all environments. For `latent-score` and `beta-rank` we sample 1000 initial ratings and run the simulation until we have 2000 ratings. For all other environments, we sample 100k ratings and run the simulation until we have 200k ratings.

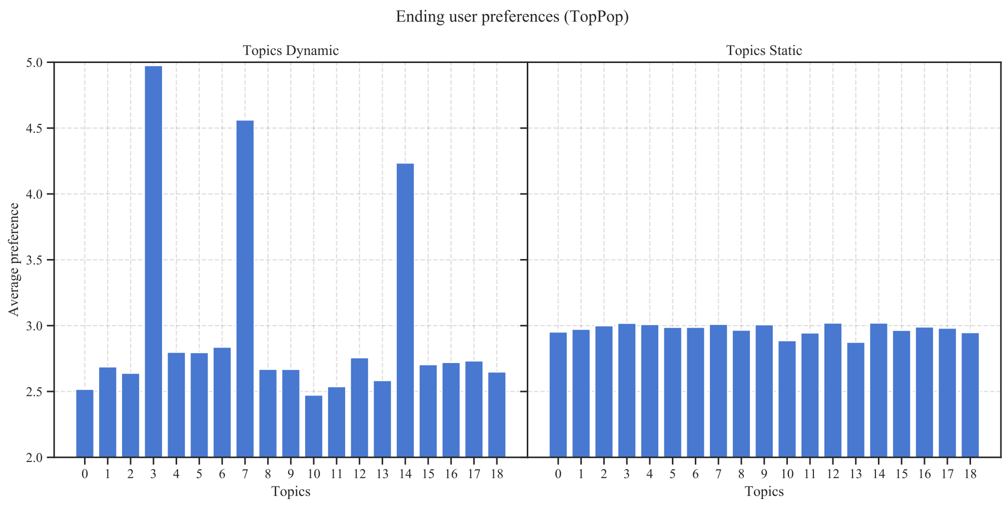


Figure 2.7: The mean affinity π for each topic averaged across all users after `TopPop` has recommended 100,000 items. Left: The mean affinity on the `topics-dynamic` environment. Right: The mean affinity on the `topics-static` environment.

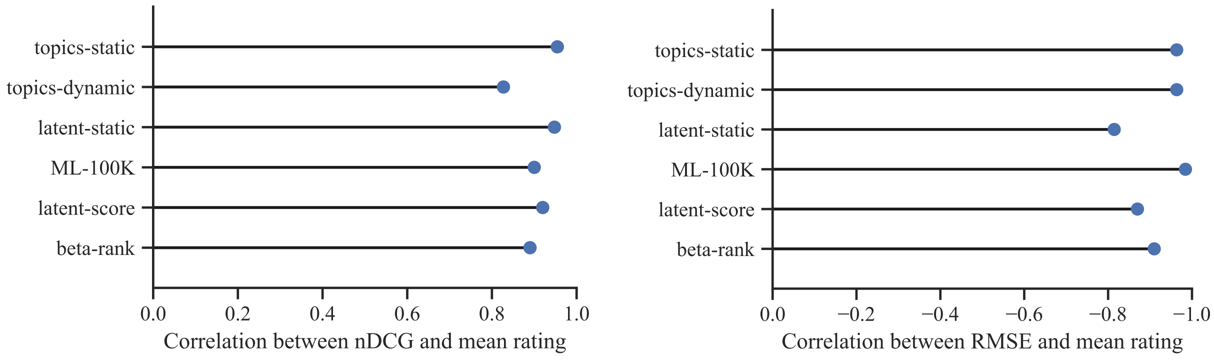


Figure 2.8: Left: The Spearman correlation between the nDCG@20 and the mean user ratings of all recommended items across all environments. Right: The Spearman correlation between the RMSE and the mean user ratings of all recommended items across all environments.

2.7 Discussion

Our experiments demonstrate encouraging results regarding the way recommender systems are currently evaluated. We showed that ranking recommender systems using offline metrics is a methodology that is surprisingly robust to changes in the sampling distribution of the offline dataset. Furthermore, the hyperparameter settings of recommender systems are also robust to such shifts in sampling distributions. These findings suggest that the sampling bias of popular benchmark datasets is likely not a concern for the reliability of the corresponding leaderboard of algorithms, although as we outlined in the introduction, prior work has surfaced a number of alternate issues with this methodology.

We also showed that offline metrics are predictive of online performance across a wide-variety of simulated environments, ranging from static environments where the feedback effects are limited to the ratings the recommender system chooses to reveal at each timestep, to dynamic environments where the recommender system also influences the behaviour of the environment with its predictions. Further justifying the use of offline metrics as a screening mechanism for recommendation algorithms.

However, our work also raises a number of concerns: the same feedback mechanism can have a surprisingly different effect on different recommender systems, increases in offline metrics can lead to diminishing returns in online performance, it is unclear what sampling scheme for the training set will lead to the best online performance, and even when the environment is static the dynamics created by the recommender system can lead to unexpected behavior.

Given these issues, in the following subsections, we discuss two major issues we hope future algorithm developers will have in mind when designing new recommender systems.

The Role of Simulation in Evaluation

In this work we designed simplified environments with reference to data-backed prior work and implemented existing environments that are vetted by the research community. These environments are not a perfect recreation of the real world. Nonetheless there is significant value in studying algorithms and metrics in a simplified setting, since the ability to control for many factors allows for a more mechanistic understanding of the complex dynamics in recommendations.

This is a widely accepted fact in theory-oriented work [46, 233], and is starting to be adopted within more empirical settings. For example, simulation has been widely accepted within the field of reinforcement learning as a tool to benchmark algorithms. These simulations vary from semi-realistic physical models [228] to video games with little grounding in reality [23, 171].

While simulation should not replace online evaluation, academic recommender systems are often developed as generalized rating prediction engines, with no specific platform in mind. Hence, a recommender system failing to perform well in simulation should be taken as strong negative evidence that such a system would fail to perform well in the real world and should probably be reworked; just as a supervised learning algorithm that fails to fit a simple toy problem would be seen with suspicion. As such, we hope that our simulation framework can be incorporated in the evaluation pipeline of recommender systems since simulations can catch many issues that would otherwise only surface when a recommender system is deployed.

Diminishing Returns

Increases in offline metrics lead to diminishing returns in terms of online performance, as shown in Figure 2.1. The hesitance of technology companies to adopt expressive yet computationally expensive models [6] indicates that we may already operate in this regime of diminished returns for real-world recommendation tasks. Furthermore, recommendation tasks customarily involve an abundance of data with which most current algorithms might already be predicting near optimally. Instead, large accuracy gains in these high-data settings could be obtained through the measurement of new user and item features rather than algorithmic innovation.

This issue is compounded by another source of diminishing returns: as predictive models achieve higher-and-higher scores on offline benchmarks, their complexity grows exponentially [7]. This is not a feasible solution for deployed recommenders where models must handle up to billions of users and items [47]. Even our simulations, which are modest in size when compared to a production system, took many thousands of compute hours due to the computational complexity of state-of-the-art models. These issues indicate that further research effort would be better spent gaining deeper understanding of existing algorithms and datasets to guide the focus of algorithmic improvements.

Chapter 3

Breaking Feedback Loops in Recommender Systems with Causal Inference

3.1 Introduction

In the previous chapter we started to examine feedback loops in recommender systems using simulations. In this chapter we will show how to address feedback loops in a principled manner using causal inference. First, a brief refresher on how feedback loops occur. At each time step, a recommender system makes recommendations and collects user feedback. At the next time step, the recommendation algorithm is updated based on this feedback. The process alternates between these two steps, inducing a feedback loop: the recommendation algorithm influences what user behavior data it observes; this data in turn affects the recommendation algorithm, since the algorithm is trained on this data. Over time, the issue is exacerbated: the recommender system is trained on a growing set of data points that have been biased by recommendations.

Uncontrolled feedback loops create negative externalities that are shouldered by consumers and producers. For example, they compromise recommendation quality as they bias the behavioral data collected by the system. They also exacerbate homogenization effects [43]: if a user interacts with an item early on, the recommendation system is more likely to recommend similar items at the expense of dissimilar items that the user might prefer. A related issue is the "rich-get-richer" problem where items that are popular early on are undeservedly recommended over newer items since the recommender system has observed more data about them [41, 202].

A naive way to break feedback loops is to recommend random items. Such a system does not suffer from negative feedback effects because its recommendations no longer depend on past data, but it does not learn from user behavior and would unacceptably degrade recommendation quality. So how can we break feedback loops without making recommendations useless?

In this work, we study the *causal* mechanism underlying the recommendation process and propose the *causal adjustment for feedback loops (*CAFL*)*, an algorithm that can provably break feedback loops in recommender systems. Studying feedback loops with a causal lens leads to a key observation: recommendation algorithms do not suffer from feedback loops if they reason about causal quantities, namely the intervention distributions of recommendations on user ratings. The reason is that a *do* intervention on a causal graph, by definition, breaks the connection between the variable being intervened on (e.g., the recommendation) and its normal causes (e.g., user feedback) [181].

The causal mechanism of recommendation also reveals that the intervention distributions of recommendations are identifiable from observational data. To calculate the intervention distributions, it is sufficient to adjust for the recommender system's predictions, since feedback loops in recommender systems only occur through this quantity (see Fig. 3.1b). Following this observation, we show how to design an algorithm, CAFL, that estimates intervention distributions in training any loss-minimizing recommendation algorithms. CAFL enables recommender systems to break feedback loops without resorting to random recommendations. In particular, it can be applied to situations where common causal assumptions (e.g. positivity [113]) are violated. For example, CAFL can be applied when a recommender system requires that all items can only be recommended at most once to each user, which violates the positivity conditions required by standard causal adjustment methods (e.g. backdoor adjustment or inverse propensity weighting).

Contributions. The contributions of this work are three-fold. (1) We formalize the operation of recommender systems over time as a structural causal model over multiple time steps. (2) We introduce CAFL: a causal adjustment algorithm that can provably break feedback loops for existing recommendation algorithms. CAFL is easy to implement in existing recommender systems as it only requires changing the weights of their loss function. (3) Across multiple simulation environments, we show that CAFL not only corrects the dataset bias caused by feedback loops, but also improves predictive performance, moreso than prior correction methods. CAFL can also reduce homogenization when feedback loops induce recommendation homogenization.

Related work. This work is motivated by a recent line of research that aims to understand the effect of feedback loops in recommender systems. Using simulations, Schmit and Riquelme [204] and Krauth et al. [134] have shown that ignoring feedback effects will negatively impact performance. As recommender systems observe more data, they are also prone to homogenization and bias amplification, which researchers often attribute to feedback effects [43, 158]. Another related line of work studies the effects of feedback loops theoretically: under assumptions of preference drifts, they show that feedback loops can have undesirable effects on users [115, 119]. Hosseinmardi et al. [99] and Sharma, Hofman, and Watts [209] further illustrate the impacts of recommender systems and feedback loops using observational data from large internet platforms.

Feedback loops can induce a sampling bias in the collected user behavior dataset, so the feedback loop problem is a form of the missing-not-at-random (MNAR) problem over multiple time steps [160]. The MNAR problem has been extensively studied in single-step recommender systems: the recommender system aims to infer missing ratings over a single timestep using a static dataset. One approach to this problem is to assume an exposure model between users and items and corrects the bias such an exposure model would induce [94, 147, 215]. Another approach is to use tools from causal inference to correct this bias under different assumptions [30, 205, 234]. In contrast to these works that focus on single-step recommender systems, this work studies the setting over multiple time steps and proposes adjustments based on the special structure of feedback loops.

Correcting feedback effects in recommender system is comparatively less studied than the MNAR problem. Earlier work augments algorithms with temporal information, but does not model feedback effects [130]. More recently, Sun et al. [220] combines inverse propensity weighting (IPW) with active learning to correct for feedback effects, assuming the same rating model as Liang et al. [147]. Pan et al. [178] modify their IPW correction to account for sequential effects. However, their method requires the marginal probability of an item being observed at each time step. Estimating these probabilities requires access to observational user data over a long period of time; the algorithm is thus unable to correct for feedback effects at the initial period when estimation is performed. Further, estimating these marginal probabilities is a challenging task, which compromises the prediction accuracy of algorithms even at later time steps. In contrast to these works, the CAFL algorithm only requires calculating the probabilities of an item being recommended conditioned on the history of observed data, which is available to the recommender system. As a result, CAFL is data-efficient, simple, and does not require strong modeling assumptions.

Finally, the problem of feedback loops in training machine learning algorithms is not unique to recommender systems. Farquhar, Gal, and Rainforth [67] focus on the problem of active learning where one collect one data point at each time step; they propose two unbiased weighting estimators for empirical risk minimization. Perdomo et al. [183] present a general framework to study the impact of feedback loops in machine learning predictions. In contrast to these works, the CAFL algorithm takes an explicitly causal view of the feedback loop. It leads to unbiased weighting estimators that are applicable to recommender systems with multiple data points acquired in a single time step. The explicit causal view also enables the use of other causal adjustment strategies for breaking feedback loops, e.g., backdoor adjustment.

3.2 Feedback loops in recommender systems

We describe feedback loops and their consequences in recommender systems. We focus on *multi-step* recommender systems; these systems are regularly retrained as they collect more data.

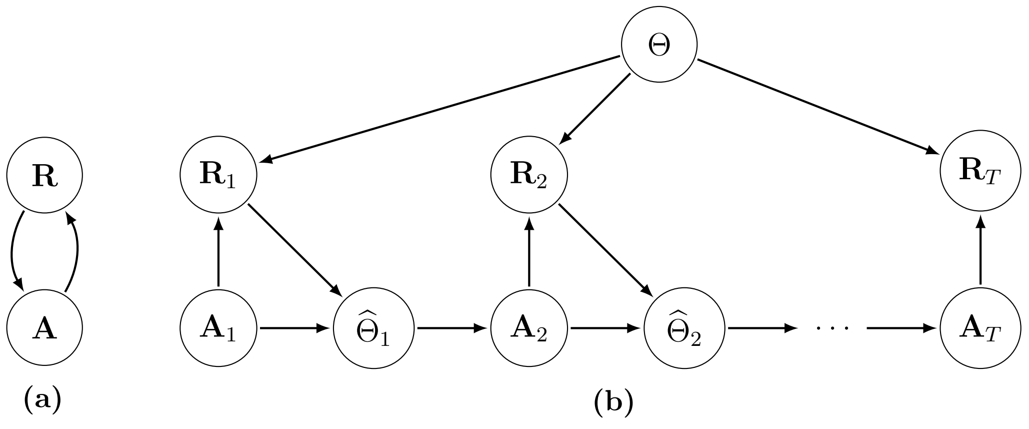


Figure 3.1: (a) Feedback loops depict the process where the recommendations $\mathbf{A}$ affect the ratings $\mathbf{R}$. They in turn affect the recommendation $\mathbf{A}$ because recommendation algorithms are trained on the collected ratings data. (b) The causal graph of recommender systems over multiple times steps with feedback loops unrolled.

Multi-step recommender systems

A multi-step recommender system operates over T time steps. At each time step, it makes recommendations to users, observes their feedback, and is then retrained on all the data observed so far.

A multi-step recommender system begins with a set of U new users and I new items. At time step $t = 1$, it recommends N items to a random set of users. Denote $\mathbf{A}_t$ as the $U \times I$ recommendation matrix, where its (u, i)-entry $A_{t,ui}$ is a binary variable that indicates whether user u was recommended item i at time t. Then, at time $t = 1$, we have $\mathbf{A}_1 \overset{iid}{\sim} \text{Multinomial}(n = N, p = \mathbf{p}_0)$, where $\mathbf{p}_0$ is a matrix of the initial probabilities of recommendation each item to each user.

After making recommendations, the multi-step recommender then collects user ratings on the items consumed at this time step. We denote the data we collect at time t as $\mathbf{R}_t$, which is a $U \times I$ rating matrix; its (u, i)-entry $R_{t,ui}$ is user u's rating of item i if the item is consumed at time step t, and $R_{t,ui} = 0$ otherwise.

Given the user ratings, the multi-step recommender then infers users' preferences based on the collected data from this first time step, $\{\mathbf{A}_1, \mathbf{R}_1\}$. Denote $\widehat{\Theta}_t$ as the inferred user preference and item attribute parameters at time t, which are usually minimizers of some loss function. More formally, the inferred user preference $\widehat{\Theta}_t$ can be seen as a maximum likelihood estimator under some parametric probability model P_Θ with parameter $\widehat{\Theta}$,

$$\begin{aligned}\widehat{\Theta}_1 &= \arg\min_{\Theta} \mathbb{E}_{\mathbf{A}_1} \left[\mathbb{KL}\left(P\left(\mathbf{R}_1 \,|\, \mathbf{A}_1\right) || P_\Theta\left(\mathbf{R}_1 \,|\, \mathbf{A}_1\right)\right)\right] \\ &= \arg\max_{\Theta} \mathbb{E}_{\mathbf{A}_1} \left[\mathbb{E}_{P(\mathbf{R}_1 \,|\, \mathbf{A}_1)} \left[\log P_\Theta\left(\mathbf{R}_1 \,|\, \mathbf{A}_1\right)\right]\right],\end{aligned}$$

where $P\left(\mathbf{R}_1 \mid \mathbf{A}_1\right)$ is the (population) conditional distribution of $\mathbf{R}_1$ given $\mathbf{A}_1$, from which the collected data $\{\mathbf{A}_1, \mathbf{R}_1\}$ is drawn, and $P_\Theta\left(\mathbf{R}_1 \mid \mathbf{A}_1\right)$ is a parametric model we posit. $\mathbb{KL}(\cdot||\cdot)$ denotes the Kullback–Leibler divergence between the two distributions.

Many existing recommendation algorithms fall into this setup. For example, assuming a probabilistic matrix factorization (PROB-MF) model for ratings [170] is equivalent to fitting $\widehat{\Theta}$ through a Gaussian linear factor model with maximum likelihood estimation, $P_\Theta\left(\mathbf{R}_1 \mid \mathbf{A}_1\right) = \prod_{u=1}^{U} \prod_{i=1}^{I} \mathcal{N}\left(R_{1,ui} \mid \theta_u^\top \beta_i \cdot A_{1,ui}, \sigma^2\right)$, where the $K \times 1$ user vectors θ_u and the $K \times 1$ item vectors β_i constitute the parameters $\Theta = \left((\theta_u)_{u=1}^{U}, (\beta_i)_{i=1}^{I}\right)$, and σ^2 is assumed known. Similarly, performing weighted matrix factorization [105] and nonnegative factorization [83, 84] also correspond to fitting parametric probability models using maximum likelihood [234].

After fitting a parametric model of user preferences, the recommender system then recommends items based on the inferred user preferences; that is, $\mathbf{A}_{2,u} \sim P_u(\mathbf{A}_{2,u} \mid \widehat{\Theta}_1), u = 1, \ldots, U$, where $P_u(\mathbf{A}_{2,u} = 1 \mid \widehat{\Theta}_1)$ describes the probability of each item being recommended to user u based on the inferred parameters $\widehat{\Theta}_1$. This distribution $P_u(\cdot \mid \widehat{\Theta})$ captures how recommender systems make decisions based on user preferences; it is usually specified by recommender systems *a priori* and can target different goals. For example, a recommender system may set up $P_u(\cdot \mid \widehat{\Theta})$ to maximize the potential rating of recommended items, or to increase user consumption, or to maximize the diversity of the recommendations without sacrificing users' utility by more than 10%.

Going from time $t = 1$ to time $t = 2, \ldots, T$, a *multi-step* recommender system further updates its inferred preferences, unlike a *one-step* recommender system which does not update its inferred user preferences. At time t, it makes new recommendations $\mathbf{A}_t$, collects new data $\{\mathbf{A}_t, \mathbf{R}_t\}$, and updates its parameters $\widehat{\Theta}_t$ by fitting the model to $P(\{\mathbf{R}_s\}_{s=1}^{t} \mid \{\mathbf{A}_s\}_{s=1}^{t})$ using all the data collected up to this point, $\{(\mathbf{A}_1, \mathbf{R}_1), (\mathbf{A}_2, \mathbf{R}_2) \ldots, (\mathbf{A}_t, \mathbf{R}_t)\}$.

But how should a recommender system update its user preference parameters $\widehat{\Theta}_t$ after the first time step? In particular, the data collected from different time steps are not independent. Future recommendations depend on past ratings because the recommender system tries to recommend items that users will like, which is inferred from past ratings. Handling this dependence over time is a core challenge in developing multi-step recommender systems since naively repeating the optimization problem in Equation 3.2 would be sub-optimal.

Feedback loops in recommender systems

To handle the dependence between data points collected at different time steps, we study their dependency structure. This dependency structure is often referred to as *feedback loops* in recommender systems, describing how $(\mathbf{A}_t, \mathbf{R}_t)$—the data collected at time t—depends on the data collected at all previous time steps, $\{(\mathbf{A}_1, \mathbf{R}_1), \ldots, (\mathbf{A}_{t-1}, \mathbf{R}_{t-1})\}$.

In more detail, feedback loops refer to the $\mathbf{A} \rightarrow \mathbf{R} \rightarrow \mathbf{A}$ loop, going from the recommendation $\mathbf{A}_t$, to the rating $\mathbf{R}_t$, and finally back to the recommendation $\mathbf{A}_{t+1}$ (Fig. 3.1a). The recommender system begins by making recommendations $\mathbf{A}_t$, these recommendations increase the probability of the recommended items being rated, hence affecting the rating we observe $\mathbf{R}_t$. The collected ratings $\mathbf{R}_t$—together with all past collected ratings—in turn, affect the recommendation matrix $\mathbf{A}_{t+1}$, because the recommender infers user preferences from this data and makes recommendations based on this inference.

To learn user preferences from this sequentially collected data, the most common approach is to aggregate the data from different time steps [143, 177, 191, 206, 218, 231, 248]. We fit the probability model for the first time step (i.e. Section 3.2) to all the data collected up to time t, and infer user preferences as follows:

$$\begin{aligned}\widehat{\Theta}_t^{\text{naive}} &= \arg\max_{\Theta} \sum_{s=1}^{t} \mathbb{E}_{\mathbf{A}_s}\left[\mathbb{E}_{P(\mathbf{R}_s \,|\, \mathbf{A}_s)}\left[\log P_\Theta\left(\mathbf{R}_s \,|\, \mathbf{A}_s\right)\right]\right] \\ &= \arg\min_{\Theta} \mathbb{E}\left[\mathbb{KL}\left(\prod_{s=1}^{t} P\left(\mathbf{R}_s \,|\, \mathbf{A}_s\right) || \prod_{s=1}^{t} P_\Theta\left(\mathbf{R}_s \,|\, \mathbf{A}_s\right)\right)\right].\end{aligned}$$

This approach considers a product of the likelihood terms from each time step, implicitly assuming data $\{\mathbf{A}_t, \mathbf{R}_t\}$ observed at different time steps are independently collected. However, they are in general *not* independent in multi-step recommender systems, that is,

$$P(\{\mathbf{R}_s\}_{s=1}^t \,|\, \{\mathbf{A}_s\}_{s=1}^t) = \prod_{s=1}^{t} P(\mathbf{R}_s \,|\, \mathbf{A}_s, \{\mathbf{R}_{s'}, \mathbf{A}_{s'}\}_{s'=1}^{s-1}) \neq \prod_{s=1}^{t} P(\mathbf{R}_s \,|\, \mathbf{A}_s)$$

since $P(\{\mathbf{A}_s, \mathbf{R}_s\}_{s=1}^t) \neq \prod_{s=1}^t P(\mathbf{R}_s, \mathbf{A}_s)$. The only exception is when all recommendations $\mathbf{A}_s$'s are completely random, and thus they do not depend on past observations. By treating dependent data as if they were independent, $\widehat{\Theta}_t^{\text{naive}}$ is a biased estimator of user preferences.

Beyond biased estimation, feedback loops are also known to produce a "rich get richer" phenomenon, leading to homogenization in recommendations over time [43]. As an example, we contrast two toy movie recommender systems, one with feedback loops (making recommendations based on $\widehat{\Theta}_t^{\text{naive}}$ at time t) and one without feedback loops (making random recommendations at time t). We then consider their recommendations to two different users: user A likes both drama and horror movies, and user B likes both drama and sci-fi movies.

The recommender with feedback loops often experiences recommendation homogenization over time. For example, suppose it recommends drama movies to both users at $t = 1$ by chance. Both users will rate it highly because the recommendation aligns with (part of) their preferences. The recommender then collects these ratings and infers that both users like drama movies. It thus continues to recommend drama movies at $t = 2$, and again both users will rate it highly. Continuing with this process, the recommender with feedback loops will incorrectly infer that the two users have the same preferences, hence homogenizing user recommendations.

In contrast, the recommender without feedback loops does not homogenize recommendations. Again suppose it recommends a drama movie to both users at $t = 1$ by chance; both users rate it highly; the recommender will then infer that both like drama movies, as with the recommender with feedback loops. At time $t = 2$, however, the recommender will not solely recommend drama movies. Rather, it may recommend some other movies like a sci-fi movie. In this case, only user B may rate it highly. With this data from $t = 2$, the recommender will correctly infer that the two users have different user preferences.

Given these concerns about biased and homogenized recommendations, multi-step recommender systems ought to avoid feedback loops [43]. One immediate way to avoid feedback loops is to adopt a random recommendation mechanism. Such recommendations are not affected by past ratings and thus are feedback-free. But the user experience will suffer with random recommendations. How can we then break the feedback loop without making random recommendations? In the next sections, we analyze the causal mechanism of feedback loops and show that a recommender system does not suffer from feedback loops if it reasons about causal quantities, namely the intervention distributions of recommendations $\mathbf{A}_s$ on ratings $\mathbf{R}_s$. This observation leads us to develop a causal adjustment algorithm that can provably break feedback loops without resorting to random recommendations.

3.3 Breaking Feedback Loops in Recommendation Systems with Causal Inference

We begin with a description of the causal mechanism of feedback loops in recommender systems. This causal perspective will lead to an adjustment algorithm that can provably break feedback loops.

The causal mechanism of feedback loops

To break feedback loops, we first study its causal mechanism by unrolling it over t time steps.

Begin by writing down the causal graphical model [181] of recommender systems in Fig. 3.1b. At time $t = 1$, the recommender system begins with some recommendations $\mathbf{A}_1$. These recommendations causally affect the observed user ratings $\mathbf{R}_1$ by increasing the probability of the recommended items being rated. Both the recommendations and the observed ratings also affect the inferred user preference parameters $\widehat{\Theta}_1$ since the recommender optimizes $\widehat{\Theta}_1$ based on $\{\mathbf{A}_1, \mathbf{R}_1\}$. This inferred user preference $\widehat{\Theta}_1$ then affects $\mathbf{A}_2$, i.e. what items are recommended at $t = 2$.

The causal structure of $\left\{\mathbf{A}_1, \mathbf{R}_1, \widehat{\Theta}_1, \mathbf{A}_2\right\}$ then repeats itself at each time step t. In particular, the inferred user preferences $\widehat{\Theta}_t$ generally depends on all past recommendations and ratings $\left\{\mathbf{A}_{1:(t-1)}, \mathbf{R}_{1:(t-1)}\right\}$. Finally, the (unobserved) true user preferences Θ affects all ratings across all time steps, hence there is an arrow from Θ to all $(\mathbf{R}_t)_{t=1}^{T}$.

The existence of feedback loops is evident in the causal graph (Fig. 3.1b), where past recommendations and ratings constantly inform future recommendations. The data $\{\mathbf{A}_t, \mathbf{R}_t\}$ collected at different time steps are thus not independent, i.e. $P(\{\mathbf{R}_s\}_{s=1}^{t} \mid \{\mathbf{A}_s\}_{s=1}^{t}) \neq \prod_{s=1}^{t} P(\mathbf{R}_s \mid \mathbf{A}_s)$ as in Section 3.2; they causally depend on each other, preventing us from fitting a single model to the data from all time steps. Given the causal graph from Fig. 3.1b, how can we break the feedback loop $\mathbf{A} \to \mathbf{R} \to \mathbf{A} \to \mathbf{R} \to \mathbf{A} \to \cdots$ in updating recommendation algorithms and fitting $\widehat{\Theta}_t$?

Breaking feedback loops in recommender systems using causal inference

To break the feedback loops in Fig. 3.1b, we need to find a distribution $\widetilde{P}$ about recommendations and ratings $\{\mathbf{A}_s, \mathbf{R}_s\}_{s=1}^t$ such that it does achieve independence over time steps, i.e. $\widetilde{P}(\{\mathbf{R}_s\}_{s=1}^t \,|\, \{\mathbf{A}_s\}_{s=1}^t) = \prod_{s=1}^t \widetilde{P}(\mathbf{R}_s \,|\, \mathbf{A}_s)$. Such a distribution does not suffer from feedback loops or their resulting model-fitting bias in Section 3.2; it would allow us to learn user preference parameters $\widehat{\Theta}_t$ from all past time steps.

Causal inference provides a solution to this challenge: the intervention distribution of recommendations on ratings $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{a}))$ achieves this independence and does not suffer from feedback loops [181]. In more detail, $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{a}))$ denotes the distribution of $\mathbf{R}_s$ under the intervention of setting $\mathbf{A}_s$ to be equal to the value a. It does not suffer from feedback loops because, by definition, a *do* intervention $\mathrm{do}(\mathbf{A}_s = \mathbf{a})$ breaks the connection between the variables being intervened on $\mathbf{A}_s$ and its parents $\widehat{\Theta}_{s-1}$ in the causal graph (Fig. 3.1b), thus breaking the $\mathbf{A}_s \to \mathbf{R}_s \to \mathbf{A}_{s+1}$ feedback loop. The following lemma formalizes this argument.

Lemma 1. *Assuming the causal graph in Fig. 3.1b, we have*

$$P(\{\mathbf{R}_s\}_{s=1}^t \,|\, \mathrm{do}(\{\mathbf{A}_s\}_{s=1}^t = \{\mathbf{a}_s\}_{s=1}^t) = \prod_{s=1}^t P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{a}_s)),$$
$$\forall \mathbf{a}_s \in \{0,1\}^{U\times I}, t \in \{2,\ldots,T\}.$$

Lemma 1 is an immediate consequence of the *do* calculus [181].

Moreover, the intervention distribution is the distribution with the highest fidelity to the observational data while staying immune from feedback loops. More precisely, among all distributions $\widetilde{P}$ that satisfy independence across time steps (Lemma 1), the intervention distribution $\prod_{s=1}^t P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{a}))$ is the distribution that is closest to $\widetilde{P}\left(\{\mathbf{R}_s\}_{s=1}^t \,|\, \{\mathbf{A}_s\}_{s=1}^t\right)$ in Kullback-Leibler (KL) divergence:

Lemma 2. *Assuming the causal graph in Fig. 3.1b, we have, for all $\mathbf{a}_s \in \{0,1\}^{U\times I}$ and $t \in \{2,\ldots,T\}$,*

$$\prod_{s=1}^t P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{a}_s))$$
$$= \operatorname*{arg\,min}_{\widetilde{P}\in\mathcal{Q}} \mathbb{KL}\left(P(\{\mathbf{R}_s\}_{s=1}^t \,|\, \{\mathbf{A}_s\}_{s=1}^t = \{\mathbf{a}_s\}_{s=1}^t) \middle|\middle| \widetilde{P}(\{\mathbf{R}_s\}_{s=1}^t \,|\, \{\mathbf{A}_s\}_{s=1}^t = \{\mathbf{a}_s\}_{s=1}^t)\right),$$

where $\mathcal{Q} = \left\{\widetilde{P} : \widetilde{P}(\{\mathbf{R}_s\}_{s=1}^t \,|\, \{\mathbf{A}_s\}_{s=1}^t = \{\mathbf{a}_s\}_{s=1}^t) = \prod_{s=1}^t \widetilde{P}(\mathbf{R}_s \,|\, \mathbf{A}_s = \mathbf{a}_s)\right\}$ *is the set of all distributions that satisfies the independence relationship in Lemma 1.*

Lemma 2 is an immediate consequence of Theorem 1 in Wang, Sridhar, and Blei [235].

Taken together, Lemmas 1 and 2 imply that, to avoid feedback loops in recommender system, one should fit the parametric model P_Θ to the intervention distributions $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{a}_s))$ instead of the observational distributions $P(\mathbf{R}_s \,|\, \mathbf{A}_s = \mathbf{a}_s)$ in Section 3.2,

$$\begin{aligned} \widehat{\Theta}_t^{\text{causal}} &= \arg\min_{\Theta} \mathbb{E}_{\{\mathbf{A}_s\}_{s=1}^t}\left[\mathbb{KL}\left(\prod_{s=1}^t P\left(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{A}_s)\right) \,||\, \prod_{s=1}^t P_\Theta\left(\mathbf{R}_s \,|\, \mathbf{A}_s\right)\right)\right] \\ &= \arg\max_{\Theta} \qquad L^{\text{causal}}, \\ \text{where} \qquad & L^{\text{causal}} \triangleq \sum_{s=1}^t \mathbb{E}_{\mathbf{A}_s}\left[\mathbb{E}_{P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{A}_s))}\left[\log P_\Theta\left(\mathbf{R}_s \,|\, \mathbf{A}_s\right)\right]\right]. \end{aligned}$$

We term Section 3.3 as the *feedback-free causal objective*. How can we solve the optimization problem in Section 3.3 then? How can we estimate the intervention distributions $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{A}_s))$ using the observational data $\{(\mathbf{A}_1, \mathbf{R}_1), \ldots, (\mathbf{A}_t, \mathbf{R}_t)\}$? We discuss these questions in the next section.

Inferring user preferences from intervention distributions

To estimate the intervention distributions $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{A}_s)$ in recommender systems, we first write them as a functional of the observational data distribution $P(\{\mathbf{A}_s, \mathbf{R}_s, \widehat{\Theta}_s\}_{s=1}^t)$. This procedure is also known as *causal identification* [181]. These results will lead to unbiased estimates of the optimization objectives in Section 3.3, enabling us to infer the parameters $\widehat{\Theta}_t^{\text{causal}}$ from observational data and perform recommendation with the inferred parameters.

To identify the intervention distributions $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{A}_s)$, we return to the causal graph (Fig. 3.1b) of multi-step recommender systems. The causal graph implies that $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{A}_s)$ is identifiable from the observational data. Moreover, it is sufficient to adjust for the inferred user preferences $\widehat{\Theta}_{s-1}$ to calculate the intervention distribution $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{A}_s)$, since the feedback loop $\mathbf{A} \to \mathbf{R} \to \mathbf{A}$ occurs only through $\widehat{\Theta}_{s-1}$ in the causal graph (Fig. 3.1b). We first state the key assumption this argument relies on, namely the positivity condition, and then state this argument formally.

Assumption 1 (Positivity, a.k.a. overlap [113])**.** *The random variables* $\mathbf{A}_s$ *satisfies the positivity condition if, for all values of* $s, \mathbf{a}_s$ *and* $\widehat{\Theta}_{s-1}$*, we have*

$$P(\mathbf{A}_s = \mathbf{a}_s \,|\, \widehat{\Theta}_{s-1}) > 0.$$

Loosely, the positivity condition requires that it must be possible to recommend any subset of items to any subset of users at any time steps. Under positivity, we can identify the intervention distributions as follows.

Lemma 3. *Assuming the causal graph in Fig. 3.1b. Then under Assumption 1, we have*

$$P(\mathbf{R}_s \mid \mathrm{do}(\mathbf{A}_s = \mathbf{a}_s)) = \int P(\mathbf{R}_s \mid \mathbf{A}_s = \mathbf{a}_s, \widehat{\Theta}_{s-1}) P(\widehat{\Theta}_{s-1}) \,\mathrm{d}\widehat{\Theta}_{s-1}.$$

Lemma 3 is an immediate consequence of the backdoor adjustment formula for identifying intervention distributions [181]. Plugging Lemma 3 into Section 3.3, Lemma 3 implies that, when positivity holds, one can estimate $\widehat{\Theta}_t^{\text{causal}}$ by solving

$$\widehat{\Theta}_t^{\text{causal}} = \arg\max_{\Theta} \sum_{s=1}^{t} \int\int\int P(\mathbf{A}_s) P(\mathbf{R}_s \mid \mathbf{A}_s, \widehat{\Theta}_{s-1}) P(\widehat{\Theta}_{s-1}) \log P_\Theta(\mathbf{R}_s \mid \mathbf{A}_s) \,\mathrm{d}\widehat{\Theta}_{s-1} \,\mathrm{d}\mathbf{A}_s \,\mathrm{d}\mathbf{R}_s$$

$$= \arg\max_{\Theta} \sum_{s=1}^{t} \mathbb{E}_{P(\mathbf{A}_s, \mathbf{R}_s, \widehat{\Theta}_{s-1})} \left[\frac{P(\mathbf{A}_s)}{P(\mathbf{A}_s \mid \widehat{\Theta}_{s-1})} \log P_\Theta(\mathbf{R}_s \mid \mathbf{A}_s) \right],$$

where Section 3.3 is due to the chain rule $P(\mathbf{A}_s, \mathbf{R}_s, \widehat{\Theta}_{s-1}) = P(\mathbf{R}_s \mid \mathbf{A}_s, \widehat{\Theta}_{s-1}) \cdot P(\mathbf{A}_s \mid \widehat{\Theta}_{s-1}) \cdot P(\widehat{\Theta}_{s-1})$.

The expectation in the optimization objective for $\widehat{\Theta}_t^{\text{causal}}$ (Section 3.3) can be unbiasedly estimated from observational data. Specifically, Section 3.3 implies that, under positivity, we can solve for $\widehat{\Theta}_t^{\text{causal}}$ via weighted maximum likelihood estimation,

$$\widehat{\Theta}_t^{\text{causal}} = \arg\max_{\Theta} \quad \widehat{L}_{\text{IPW}}^{\text{causal}}(\Theta),$$

$$\text{where} \quad \widehat{L}_{\text{IPW}}^{\text{causal}}(\Theta) \triangleq \sum_{s=1}^{t} \sum_{\mathbf{a}_s \in \mathcal{A}} \frac{\mathbb{1}\{\mathbf{A}_s = \mathbf{a}_s\}}{P(\mathbf{A}_s = \mathbf{a}_s \mid \widehat{\Theta}_{s-1})} \cdot \log P_\Theta(\mathbf{R}_s \mid \mathbf{A}_s = \mathbf{a}_s).$$

The set $\mathcal{A} = \{0,1\}^{U \cdot I}$ denotes the set of all possible values that $\mathbf{A}_s$ can take. The next proposition justifies this estimator.

Proposition 1 (Unbiased estimation of the causal objective (Section 3.3) under positivity assumptions)**.** *If positivity (Assumption 1) holds, then $\widehat{L}_{\text{IPW}}^{\text{causal}}(\Theta)$ is an unbiased estimator of the causal objective in Section 3.3,*

$$\mathbb{E}\left[\widehat{L}_{\text{IPW}}^{\text{causal}}(\Theta)\right] = L^{\text{causal}}(\Theta).$$

Proposition 1 is an immediate consequence of Section 3.3. It implies that $\widehat{L}_{\text{IPW}}^{\text{causal}}(\Theta)$ provides a causal adjustment to the maximum likelihood objective that does not suffer from feedback loops. It takes the form of the standard inverse probability weighting (IPW) estimator in causal inference [113], where $\widehat{L}_{\text{IPW}}^{\text{causal}}$ is a weighted sum of the one-step optimization objective for recommendations (Section 3.2) with weights being the inverse of the probabilities $P(\mathbf{A}_s \mid \widehat{\Theta}_{s-1})$.

The weighting estimator $\widehat{L}^{\text{causal}}_{\text{IPW}}(\Theta)$ is different from the other IPW estimators targeting MNAR issues in one-step recommender systems (e.g. Marlin and Zemel [160] and Schnabel et al. [205]): the weights in $\widehat{L}^{\text{causal}}_{\text{IPW}}$ are inverses of the probabilities given the inferred user preferences at the previous time step $\widehat{\Theta}_{s-1}$; in contrast, the weights in other IPW estimators are often inverses of the probabilities given user covariates like their demographics or characteristics. This difference is due to the particular causal structure of feedback loops in multi-step recommender systems (Fig. 3.1b); this structure is not present in one-step recommender systems.

Taking Section 3.3 together, we can infer user preferences $\widehat{\Theta}_t$ by fitting matrix factorization models P_Θ to intervention distributions, when the positivity condition holds. For example, we can extend the PROB-MF in Section 3.2 from one-step recommender systems to multi-step recommender systems with feedback loops: we infer user preferences at each time step by solving

$$
\begin{aligned}
&\widehat{\Theta}^{\text{causal}}_{t,\text{PROB-MF}} \\
&= \arg\max_{\Theta} \sum_{s=1}^{t}\sum_{u=1}^{U}\sum_{i=1}^{I}\sum_{a_{s,ui}\in\{0,1\}} \frac{\mathbb{1}\{A_{s,ui}=a_{s,ui}\}}{P(A_{s,ui}=a_{s,ui}\,|\,\widehat{\Theta}_{s-1})}\cdot \log\mathcal{N}(R_{s,ui}\,|\,\theta_u^\top\beta_i\cdot A_{s,ui},\sigma^2) \\
&= \arg\max_{\Theta} \sum_{s=1}^{t}\sum_{u=1}^{U}\sum_{i=1}^{I} \frac{\mathbb{1}\{A_{s,ui}=1\}}{P(A_{s,ui}=1\,|\,\widehat{\Theta}_{s-1})}\cdot \log\mathcal{N}(R_{s,ui}\,|\,\theta_u^\top\beta_i,\sigma^2),
\end{aligned}
$$

where $\Theta = ((\theta_u)_{u=1}^U, (\beta_i)_{i=1}^I)$, and the second equation is due to $R_{s,ui} = 0$ if and only if $A_{s,ui} = 0$.

Estimating user preferences under violations of positivity

The IPW estimator $\widehat{L}^{\text{causal}}_{\text{IPW}}$ in the previous section provides an unbiased estimator of the maximum likelihood objective for fitting intervention distribution. However, the validity of $\widehat{L}^{\text{causal}}_{\text{IPW}}$ relies on a core causal assumption, namely the positivity condition (Assumption 1) in Lemma 3. It requires that, for all $\mathbf{a}, s, \widehat{\Theta}_{s-1}$, we have $P(\mathbf{A}_s = \mathbf{a}_s\,|\,\widehat{\Theta}_{s-1}) > 0$. In other words, it must be possible to recommend any subset of items to any subset of users at any time steps, no matter what the inferred user preferences are.

This positivity condition is often violated in multi-step recommender systems. For example, multi-step recommender systems often require that an item cannot be recommended to the same user twice. In such cases, an item cannot possibly be recommended to a user at a later time step if it has already been recommended, constituting a violation of the positivity condition. In such cases, the IPW estimator does not apply since the probability of some recommendation configurations $P(\mathbf{A}_s = \mathbf{a}_s\,|\,\widehat{\Theta}_{s-1})$ is zero, and thus the inverse probability weight is infinite.

In this section, we extend the IPW estimator $\widehat{L}^{\text{causal}}_{\text{IPW}}$ to settings where positivity is violated. We construct an unbiased estimator of causal objective (Section 3.3) for these settings. The key idea is to leverage additional invariance structures of the intervention distributions $P(R_{s,ui} \,|\, \mathrm{do}(A_{s,ui}))$ over time to overcome the challenge of positivity violation. Specifically, we assume that $P(R_{s,ui} \,|\, \mathrm{do}(A_{s,ui}))$ is stationary over time, and all user-item pairs have a non-zero probability of recommendation at a non-empty subset of time steps. Then, for (u,i) pairs with probability zero of recommendation at time t, we could form an unbiased estimator for time t using other time steps with non-zero recommendation probabilities.

Theorem 2 (Unbiased estimation of the causal objective (Section 3.3) under positivity violations)**.** *Suppose the following assumptions hold:*

1. *There is no interference between users or items at a single time step, i.e. recommending an item to a user does not affect the ratings of other users or items at the same time step,* $P(\mathbf{R}_s \,|\, \mathrm{do}(\mathbf{A}_s = \mathbf{a}_s)) = \prod_{u=1}^{U}\prod_{i=1}^{I} P(R_{s,ui} \,|\, \mathrm{do}(A_{s,ui} = a_{s,ui}))$*. Moreover, the parametric model* P_Θ *satisfies a similar factorization.* $P_\Theta(\mathbf{R}_s \,|\, \mathbf{A}_s = \mathbf{a}_s) = \prod_{u=1}^{U}\prod_{i=1}^{I} P_\Theta(R_{s,ui} \,|\, A_{s,ui} = a_{s,ui})$*.*

2. *The intervention distributions of recommendations on ratings are stationary over time,* $P(R_{s,ui} \,|\, \mathrm{do}(A_{s,ui} = a_{s,ui})) = P(R_{s',ui} \,|\, \mathrm{do}(A_{s,ui} = a_{s',ui}))$ *for any* s, s'*.*

3. *For each user-item pair, there exists some time step when there is nonzero probability for this pair to be recommended: for each* (u,i)*, there exists some* $s \in \{1,\dots,t\}$ *such that* $P(A_{s,ui} \,|\, \widehat{\Theta}_{s-1}) > 0$*.*

Then any $\widehat{L}^{\text{causal}}_{\text{CAFL}}(\Theta\,;\,\mathbf{c})$ *is an unbiased estimator of the causal objective (Section 3.3): for any* $\mathbf{c} = (c_1,\dots,c_t)$ *with* $\sum_{s=1}^{t} c_s = t$*, we have*

$$\mathbb{E}\left[\widehat{L}^{\text{causal}}_{\text{CAFL}}(\Theta\,;\,\mathbf{c})\right] = L^{\text{causal}}(\Theta),$$

where

$$\begin{aligned}&\widehat{L}^{\text{causal}}_{\text{CAFL}}(\Theta\,;\,\mathbf{c})\\ &\triangleq \sum_{s=1}^{t} c_s \left[\sum_{\substack{u,i:\\ P(A_{s,ui}=\\ a_{s,ui}\,|\,\widehat{\Theta}_{s-1})=0}} \sum_{a_{s,ui}\in\{0,1\}} \widehat{L}^{\text{unobs}}_{s,u,i}(\Theta\,;\,a_{s,ui}) + \sum_{\substack{u,i:\\ P(A_{s,ui}=\\ a_{s,ui}\,|\,\widehat{\Theta}_{s-1})>0}} \sum_{a_{s,ui}\in\{0,1\}} \widehat{L}^{\text{obs}}_{s,u,i}(\Theta\,;\,a_{s,ui}) \right],\end{aligned}$$

with

$$\widehat{L}^{\text{unobs}}_{s,u,i}(\Theta\,;\,a_{s,ui}) \triangleq \frac{\sum_{r:P(A_{r,ui}=a_{s,ui}\,|\,\widehat{\Theta}_{r-1})>0} \mathbb{1}\{A_{r,ui} = a_{s,ui}\} \cdot \log P_\Theta\left(R_{r,ui} \,|\, A_{r,ui} = a_{s,ui}\right)}{\sum_{r:P(A_{r,ui}=a_{s,ui}\,|\,\widehat{\Theta}_{r-1})>0} \mathbb{1}\{A_{r,ui} = a_{s,ui}\}},$$

$$\widehat{L}^{\text{obs}}_{s,u,i}(\Theta\,;\,a_{s,ui}) \triangleq \frac{\mathbb{1}\{A_{s,ui} = a_{s,ui}\} \cdot \log P_\Theta\left(R_{s,ui} \,|\, A_{s,ui} = a_{s,ui}\right)}{P\left(A_{s,ui} = a_{s,ui} \,|\, \widehat{\Theta}_{s-1}\right)}.$$

The proof of Theorem 2 is in Section 3.6. Loosely, $\widehat{L}^{\text{causal}}_{\text{CAFL}}(\Theta\,;\,\mathbf{c})$ treats the entries where positivity holds in $\widehat{L}^{\text{obs}}_{s,u,i}$ and the other entries where positivity is violated in $\widehat{L}^{\text{unobs}}_{s,u,i}$. The term $\widehat{L}^{\text{obs}}_{s,u,i}$ is the standard IPW estimator as in Proposition 1. The term $\widehat{L}^{\text{unobs}}_{s,u,i}$ leverages the stationary assumption on intervention distributions to form an unbiased estimate for entries with positivity violations, namely the empirical average of the likelihood term at other time steps.

Theorem 2 suggests that, to avoid feedback loops in recommender systems, we should make recommendations at each time step by solving for

$$\widehat{\Theta}^{\text{causal}}_{t} = \arg\max \qquad \widehat{L}^{\text{causal}}_{\text{CAFL}}(\Theta\,;\,\mathbf{c})$$

at each time step t.

In the special case where the recommender systems do not allow the same item to be recommended twice, we can instantiate Theorem 2 in the following corollary.

Corollary 1. *When (1) only one user-item pair is recommended at each time step, and no item can be recommended twice to the same user, (2) all assumptions of Theorem 2 hold, we have*

$$\begin{aligned}&\widehat{L}^{\text{causal}}_{\text{CAFL}}(\Theta\,;\,\mathbf{c})\\&=\sum_{u,i}\sum_{s=1}^{t} c_s\left[\sum_{r<s}\mathbb{1}\{A_{r,ui}=1\}\log P_\Theta\left(R_{r,ui}\,|\,A_{r,ui}\right)+\frac{\mathbb{1}\{A_{s,ui}=1\}\log P_\Theta\left(R_{s,ui}\,|\,A_{s,ui}\right)}{P\left(A_{s,ui}\,|\,\widehat{\Theta}_{s-1}\right)}\right]\\&\qquad+\text{constant}\end{aligned}$$

Corollary 1 is an immediate consequence of Theorem 2. The first term in Corollary 1 considers the terms where $A_{s,ui}=1$, and the constant term absorbs the $A_{s,ui}=0$ terms because $R_{s,ui}=0$ if and only if $A_{s,ui}=0$.

Choosing the constants c. A final challenge is to choose the constants $\mathbf{c}$ in the unbiased estimators (Theorem 2 and corollary 1), since any constant $\mathbf{c}$ with $\sum_{s=1}^{t}c_s=t$ will lead to an unbiased estimator of Section 3.3. A common choice is to choose $\mathbf{c}$ such that the expectation of the weights in front of each term $\log P_\Theta(R_{s,ui}\,|\,A_{s,ui})$ is the same, since all $(R_{s,ui},A_{s,ui})$ pairs shall be similarly informative for Θ.

In the special case of Corollary 1, one can obtain the $\mathbf{c}$ vector by calculating thet expected weights in front of each term. In more detail, the expected weight of the s-th term $\log P_\Theta(R_{s,ui}\,|\,A_{s,ui})$ is $\sum_{s'=s+1}^{t}c_{s'}+c_s\cdot(UI-s+1)/UI$. The reason is that the sth term does not contribute to $\widehat{L}^{\text{causal}}_{\text{CAFL}}(\Theta\,;\,\mathbf{c})$ in the first $s'=1,\ldots,s-1$ time steps, i.e. before the first occurrence where $A_{s,ui}=1$. It then contributes with weight $c_s/P(A_{s,ui}\,|\,\widehat{\Theta}_{s-1})$ at the sth time step, where $P(A_{s,ui}\,|\,\widehat{\Theta}_{s-1})$ has an expectation of $(UI-s+1)/UI$ since $UI-s+1$ items out of the total UI items remains to have nonzero probability of being recommended. Finally, it contributes weight c'_s for all future time steps $s'=s+1,\ldots,t$ due to the first term of Corollary 1. This calculation leads to $c_s = (UI(UI-t))/((UI-s)(UI-s+1))$, which repeats the calculation in Appendix B.4. of Farquhar, Gal, and Rainforth [67].

Intuitive understanding of the weighting estimator. Given the weighting estimator in Theorem 2, we next develop some intuitive understanding of the weights in the special case of Corollary 1. Focusing on the choice of $c_s = (UI(UI-t))/((UI-s)(UI-s+1))$ above, the weight W_{ui} of each log-likelihood term $\log P_\Theta\left(R_{s,ui} \mid A_{s,ui}=1\right)$ is

$$W_{ui} = \underbrace{\frac{UI}{UI-s}}_{\text{Normalization}} \left(\underbrace{t-s}_{\text{Fix 1}} + \underbrace{\frac{UI-t}{UI-s+1}}_{\text{Fix 2}} \underbrace{P\left(A_{s,ui}=a \mid \widehat{\Theta}_{t-1}\right)^{-1}}_{\text{IPW Weight}} \right),$$

where s is the time at which that user-item pair (u,i) was observed and t is the current time step. To understand these weights intuitively, we first notice that, in Section 3.3, the IPW weight ensures that likely (u,i)-pairs will be down-weighted while unlikely pairs will be up-weighted. This weighting scheme mimics a uniformly sampled distribution over the set of remaining items, as with standard IPW weighting. It ensures $\mathbb{E}\left[W_{ui}P\left(A_{t,ui} \mid \widehat{\Theta}_{t-1}\right)\right]$ is constant across all unobserved (u,i)-pairs at time t.

Beyond the IPW weighting, W_{ui} also exerts several fixes. Fix 1 upweights the observed (u,i)-pair since it has not had a chance at being recommended for $t-s$ time steps, this mimics a uniformly sampled distribution over the set of *all* items. In combination with the IPW weight, Fix 1 ensures $\mathbb{E}\left[W_{ui}P\left(A_{t,ui} \mid \widehat{\Theta}_{t-1}\right)\right]$ is constant across all (u,i)-pairs both observed and unobserved. Next, Fix 2 accounts for the fact that if a (u,i)-pair is recommended earlier on, it likely had a smaller chance of being recommended at that time step than if it were recommended at a later time step. Hence, IPW weights will tend to be larger for (u,i)-pairs recommended early, which implies their IPW weight should be downweighted as time progresses. By rewriting Fix 2 as $\frac{1}{UI-s+1}\left(\frac{1}{UI-t}\right)^{-1}$, we observe that it is equal to the IPW weight of a random recommender at time step t multiplied by the inverse of the IPW weight of a random recommender at time step $s-1$. In this sense, we are effectively canceling out the effect of the sample space's size on the IPW weight at time $s-1$ and replacing it with the effect of the sample space's size at time t.

Applicability of CAFL to general probability models and time-varying user preferences. Zooming out from the special case of Corollary 1, we note that the CAFL algorithm can be applied to any common parametric probability model originally developed as a one-step recommender system. The reason is that most recommendation models satisfy the constraints (especially the first assumption on $P_\Theta(\cdot)$) in Theorem 2. While we mainly illustrated the adjustment with PROB-MF in Section 3.3, CAFL can be applied to general matrix factorization models, including weighted matrix factorization [105], Poisson matrix factorization [84], variational autoencoders [148].

The CAFL algorithm can also be extended to accommodate time-varying user preferences. We simply replace the time-invariant parametric model P_Θ with a time-varying one P_Θ^t, indicating the user behavior patterns at time t. To infer P_Θ^t, we can extend the parametric model to take in time as a parameter, e.g. $P^t_\Theta(\mathbf{R}_t \mid \mathbf{A}_t; t)$. For example, one can extend PROB-MF to its time-varying counterpart with $R_{s,ui} \sim \mathcal{N}((g(t, \theta_u)^\top \beta_i \cdot A_{s,ui}, \sigma^2)$ for some parametric function g (e.g., a neural network). Such a time-varying parametric model, together with CAFL, will enable us to handle time-varying user preferences in the presence of feedback loops.

Connections between CAFL and existing IPW estimators. We conclude the section by discussing the connections between CAFL and other similar-looking IPW estimators. We begin with the connection between CAFL and existing IPW weigting estimators in one-step recommender systems [160, 205]. The $\widehat{L}_{\text{cafl}}(\Theta)$ objective differs from these standard IPW estimators in one-step recommender systems, because these latter estimators often rely on adjusting for the probability of recommendation given external user characteristics or covariates. They are different from CAFL, which adjusts for the probability given inferred user preferences from the previous time step. This adjustment of CAFL, in particular its sufficiency for breaking feedback loops, is unique to the multi-step recommender systems we consider here. Moreover, existing estimators always assume positivity (Assumption 1); they cannot provide unbiased estimation in multi-step recommendation settings where some variables may not be freely manipulated at certain time steps.

Finally, in the special case of Corollary 1, the $\widehat{L}_{\text{cafl}}(\Theta)$ objective coincides with the R_{LURE} estimator of Farquhar, Gal, and Rainforth [67]. However, the $\widehat{L}_{\text{cafl}}(\Theta)$ in Theorem 2 can be applied to general statistical estimation and causal inference settings when feedback loops are present. For example, it extends to settings where multiple data points are acquired at each time step and/or where data points acquired need not be distinct across time steps. The derivations of the two estimators are also complementary to each other: R_{LURE} is constructed by finding weights that do not depend on the time at which data points are collected. In contrast, Corollary 1 is derived from an explicitly causal perspective and finding the optimal distribution that does not suffer from feedback loops.

3.4 Empirical Studies

In this section, we evaluate the proposed CAFL algorithm using two environments from the RecLab simulation framework [134]. We show that CAFL mitigates negative feedback effects:

1. CAFL corrects the dataset bias caused by feedback loops and improves the predictive performance and recommendation quality of our model.
2. CAFL reduces homogenization when feedback loops induce homogenization.
3. In settings where feedback loops do not cause homogenization, we show that the behavior of CAFL tracks random sampling, suggesting that CAFL breaks feedback loops.

Furthermore, we compare CAFL in the experiment proposed by Pan et al. [178] in Section 6.3 of their paper. We show that CAFL significantly outperforms both the correction method proposed by Pan et al. [178], Poisson factorization [83], a popularity-based correction, and uncorrected matrix factorization.

Evaluation of feedback effects in recommender systems

In this section we detail the experimental setup in Sections 3.4 and 3.4. We outline the experimental setup comparing CAFL to prior work in Section 3.4.

Metrics of feedback effects. We begin by defining the metric we use for measuring feedback effects.

Definition 1. *Let $\widetilde{\mathbf{A}}_t$ be the recommendation matrix from a random recommender and let $\widetilde{\mathbf{R}}_t$ be the corresponding rating matrix. Furthermore, let $\widehat{\mathbf{A}}_t \sim P(\widetilde{\Theta}_{t-1})$ be the recommendation matrix where $\widetilde{\Theta}_{t-1}$ are the parameters derived by observing the "shadow" randomly sampled dataset $\{(\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{R}}_1), (\widetilde{\mathbf{A}}_2, \widetilde{\mathbf{R}}_2), \ldots, (\widetilde{\mathbf{A}}_{t-1}, \widetilde{\mathbf{R}}_{t-1})\}$. Then the effect of feedback at time t with respect to some metric $\mathcal{M}$ is defined as:*

$$\mathcal{M}\left(\mathbf{R}_t\right) - \mathcal{M}\left(\widehat{\mathbf{R}}_t\right).$$

This definition states that the effect of feedback is the difference in performance between a model that observes the ratings of the items it recommends, and a model that observes the ratings of items drawn at random. This definition allows us to differentiate between true feedback effects and phenomena caused by confounding factors, such as the inductive bias of matrix factorization models.

While $\widehat{\mathbf{R}}_t$ is not usually observable and must be approximated, it can easily be computed in simulated environments. Therefore, we report $\mathcal{M}\left(\widehat{\mathbf{R}}_t\right)$ in all the experiments to quantify feedback effects.

Simulation environments. The two simulated environments we consider are beta-rank-v1 and ml100k-v1. Beta-rank-v1 is an implementation of the environment developed by Chaney, Stewart, and Engelhardt [43], it is of significance since it was used to demonstrate that recommender systems under feedback loops homogenize user interactions. Ml100k-v1 represents each user and item as a 100-dimensional latent vector. These latent vectors were generated by fitting a matrix factorization model to the MovieLens 100K dataset [90]. Ml100k-v1 allows us to evaluate the algorithms in a simulation initialized with a real-world dataset.

Across all experiments, we use matrix factorization trained using alternating least squares as the parametric model P_Θ [16]. We implement CAFL in this setting by running weighted least squares, where the weight for each observed rating $R_{t,ui}$ is as defined in Equation 1. We repeat each experiment 10 times and report results averaged across all runs.

Competing methods. We compare CAFL with two other algorithms: (1) matrix factorization re-trained on the newly collected data at each time step; (2) matrix factorization trained on uniformly sampled unrecommended items at each time step. The first algorithm is the usual multi-step recommendation algorithm that suffers from feedback loops; it is a baseline algorithm on which CAFL performs causal adjustment and improves upon. The second uniform sampling approach does not suffer from feedback loops because it observes randomly sampled data (although makes non-random recommendations). If the performance of an algorithm tracks uniform sampling, then it suggests that it does not suffer from feedback loops.

CAFL improves recommendation quality

In this section, we evaluate how CAFL impacts the model's accuracy over time. To evaluate the algorithm in an unbiased manner, we create a test set by randomly sampling user-item pairs. User-item pairs in the test set can not be recommended during the run.

Evaluation metrics for recommendation quality. We evaluate each algorithm using root mean squared error (RMSE), which measures predictive accuracy, and normalized discounted cumulative gain (NDCG), which measures ranking accuracy and places a heavier emphasis on higher rankings. We report RMSE and NDCG with respect to the held-out test set.

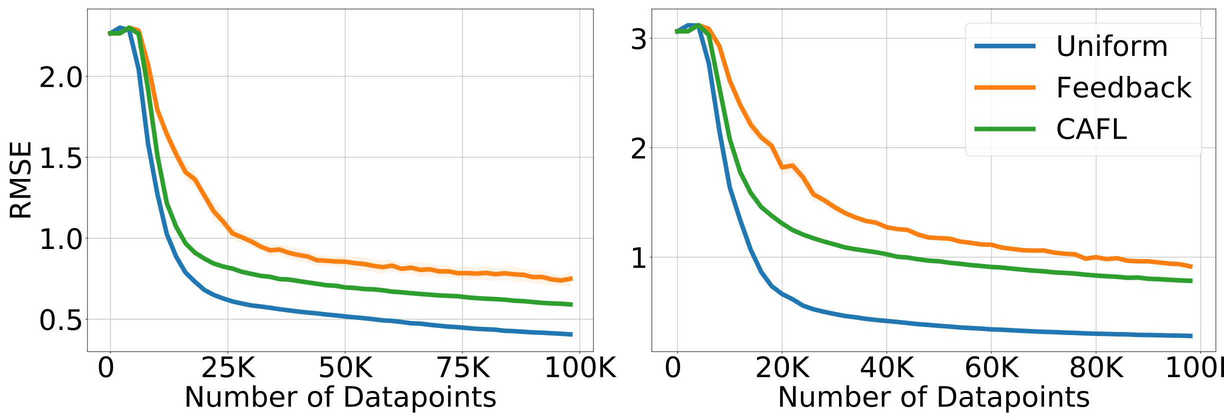


Figure 3.2: The mean RMSE of the models in the beta-rank-v1 (left) and ml-100k-v1 (right) environments averaged across 10 runs. Shaded areas indicate 95% confidence intervals. RMSE was evaluated with respect to a randomly sampled test set of size 100,000. Both CAFL and Feedback observe their own recommendations, while Uniform observes randomly chosen user-item pairs. Lower RMSE is better.

Results. As shown in Figs. 3.2 and 3.3, CAFL increases the model's predictive (RMSE) and ranking (NDCG) accuracy, when compared to the uncorrected version (Feedback).We note that the model that observes uniformly chosen datapoints (Uniform) still outperforms CAFL in most cases. This is expected since the CAFL correction is attempting to use the observed feedback data to approximate the empirical risk that Uniform observes. Uniform effectively observes more datapoints than CAFL at any given timestep.

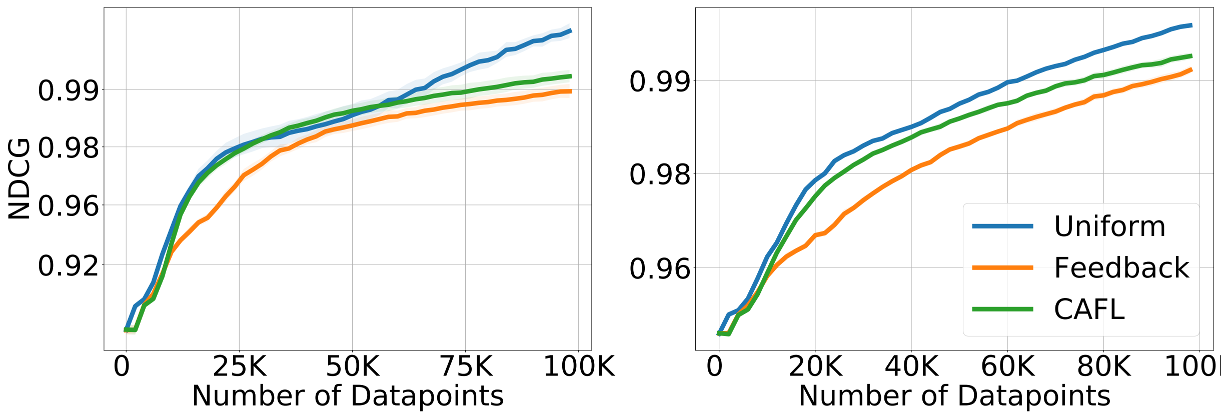


Figure 3.3: Left: The mean NDCG of the models averaged across all users and across 10 runs in the beta-rank-v1 (left) and ml-100k-v1 (right) environments. Shaded areas indicate 95% confidence intervals. NDCG was evaluated with respect to a randomly sampled test set of size 100,000. Both CAFL and Feedback observe their own recommendations, while Uniform observes randomly chosen user-item pairs. We use a logit scale for the Y-axis for readability. Higher NDCG is better.

CAFL, feedback loops, and homogenization

Recommendation systems and their feedback loops have been shown to homogenize the set of items that users will observe beyond what is necessary to achieve optimal utility [43]. This is troublesome since it implies algorithmic minutia may have an undeservedly large impact on the popularity of different items.

Here we evaluate the homogenization effect of uniform sampling, CAFL, and the vanilla recommender with feedback loops. We show that CAFL reduces homogenization when feedback loops induce homogenization. In settings where feedback loops do not induce homogenization (i.e. when feedback loops induce the same or less homogenization than uniform sampling), we show that the behavior of CAFL tracks random sampling, suggesting that CAFL breaks feedback loops in those settings too.

Evaluation metrics for homogenization. We define homogenization as the mean similarity between every pair of users' recommended items, for which we use the Jaccard coefficient as the measure of similarity between two different users u_1 and u_2:

$$S(u_1, u_2) = \frac{\sum_i \mathbf{A}_{t,u_1 i} \wedge \mathbf{A}_{t,u_2 i}}{\sum_i \mathbf{A}_{t,u_1 i} \vee \mathbf{A}_{t,u_2 i}}.$$

Results. When feedback loops increase homogenization, CAFL successfully mitigates homogenization. The right plot of Figure 3.4 shows the Jaccard index over time for the ml-100k-v1 environment. In this setting, feedback effects cause the uncorrected recommender system to further homogenize the user experience when compared to a recommender system that observes uniformly sampled data. CAFL is able to reduce homogenization in this setting.

We note that this outcome is not self-evident. In particular, the CAFL correction only leads to a more accurate empirical risk estimate and does not explicitly consider homogenization.

Turning to the beta-rank-v1 homogenization results, we observe that CAFL is unable to reduce homogenization when it is not caused by feedback effects. As shown in the left plot of Figure 3.4, CAFL increased homogenization in this setting when compared to the uncorrected feedback recommender. Surprisingly, the uniform recommender also leads to higher homogenization. This suggests that homogenization is not always caused by feedback effects, since we would otherwise expect the feedback recommender to have the highest homogenization if that were the case. In fact, these results suggest that feedback can sometimes reduce homogenization.

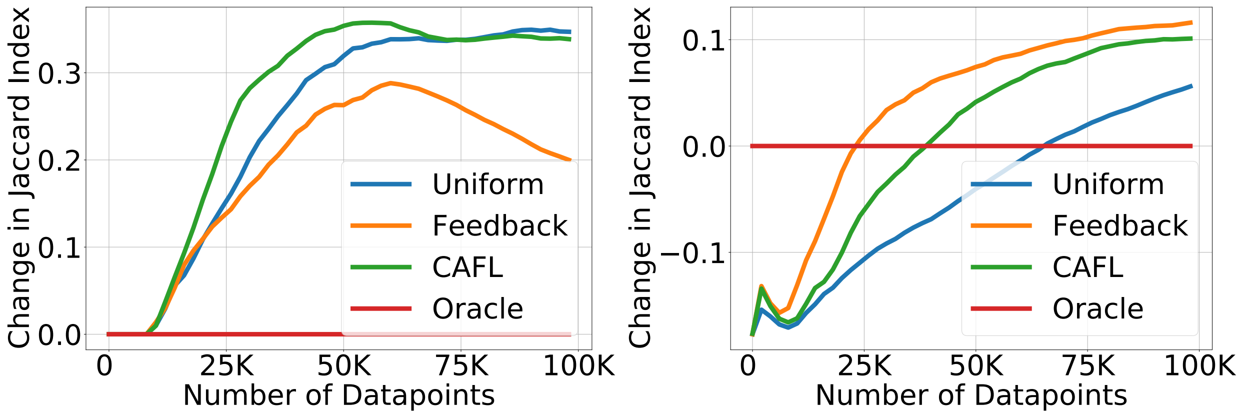


Figure 3.4: The mean Jaccard coefficient between the set of recommended items of each user-item pair at each timestep minus the Jaccard coefficient of an oracle recommender system on the beta-rank-v1 (left) and ml100k-v1 (right) environments. Both CAFL and Feedback observe their own recommendations, while Uniform observes randomly chosen user-item pairs. Lower change in Jaccard index is better.

Comparison with Prior Work

We replicated the experimental setup of Pan et al. [178] to compare CAFL with prior correction methods. We evaluate CAFL on a variation of the simulated environment first proposed by Chaney, Stewart, and Engelhardt [43]. In this setup if user u interacts with item i at time t we have

$$R_{t,ui} \sim \mathrm{Beta}'\left(\theta_u^\top \beta_i\right)$$

where $\mathrm{Beta}'(\mu)$ is a reparametrized beta distribution with variance $\sigma^2 = 0.01$ that is equivalent to $Beta(a, b)$ where $a = \left(\frac{1-\mu}{\sigma^2} - \frac{1}{\mu}\right)\mu$ and $b = a\left(\frac{1}{\mu} - 1\right)$. The latent user and item vectors have distribution

$$\theta_u \sim \text{Dirichlet}(\mu_\theta), \quad \mu_\theta \sim \text{Dirichlet}(\mathbf{20})$$
$$\beta_i \sim \text{Dirichlet}(\mu_\beta), \quad \mu_\beta \sim \text{Dirichlet}(\mathbf{100}).$$

We consider $U = 3000$ users and $I = 1000$ items. We sample one item for each user over 30 timesteps, where items are selected uniformly at random when $t = 1$ and for $t > 1$ we have

$$P\left(A_{t,ui} = 1\right) \propto \begin{cases} 0 & \text{if user } u \text{ already interacted with item } i \\ 10 & \text{if } \text{rank}_t(u, i) <= 100 \\ 1 & \text{otherwise,} \end{cases}$$

where the ranking function $\text{rank}_t(u, i)$ orders items from largest to smallest based on a score function intended to mimic the recommendation process:

$$\text{score}_t(u, i) \propto \sum_{s=1}^{t-1} \sum_{j=1}^{I} A_{s,uj} R_{s,uj} \exp\left(S_{ij}\right),$$

where S_{ij} is an item-item similarity matrix with distribution

$$S_{ij} \sim \text{Beta}'\left(\beta_i^\top \beta_j\right).$$

We use the first 20 sampled items for each user as the training set and do not consider the last 10 for consistency with Pan et al. [178].[1] Finally, we sample an additional 20 unobserved ratings uniformly at random for each user to create the test set.

We then train a generalized matrix factorization model [92] using Adam with identical hyperparameter settings to Pan et al. [178] but with each observation in the training loss weighted according to CAFL. We then evaluate the recommender's predictions on the test set, repeating the entire simulation procedure 10 times.

Table 3.1 shows the performance of CAFL averaged across all 10 runs compared to the correction methods evaluated by Pan et al. [178]. CAFL outperforms all prior methods both in terms of MSE and MAE. We observe that the improvement in MSE/MAE when comparing CAFL to Pan is larger than the improvement in MSE/MAE when comparing Pan to Poisson Factorization. Furthermore, we note that the MSE gap between the simple popularity-based re-weighting scheme and Pan is equal to the MSE gap between CAFL and Pan, indicating that the CAFL algorithm proposed in this work leads to significant performance improvements.

[1]Pan et al. [178] use half of the last 10 items for evaluations and the other half as a validation set. This requirement does not apply to our algorithm: we do not need a validation set since we use the same hyperparameter settings as Pan et al. [178].

Correction	MSE	MAE
Naive	2.001 ± 0.066	1.087 ± 0.021
Pop	1.990 ± 0.035	1.080 ± 0.010
PF [83]	1.945 ± 0.038	1.065 ± 0.010
Pan [178]	1.896 ± 0.042	1.042 ± 0.011
CAFL (This paper)	$\mathbf{1.818 \pm 0.019}$	$\mathbf{1.015 \pm 0.005}$

Table 3.1: Predictive performance (MSE and MAE) of generalized matrix factorization on a benchmark derived from a modification of the simulation proposed by Chaney, Stewart, and Engelhardt [43] when trained with: no correction (Naive), a correction that scales inversely with item popularity (Pop), Poisson Factorization (PF), the correction by Pan et al. [178] (Pan), and the correction algorithm proposed in this work (CAFL).

3.5 Discussion

Feedback loops are endemic in multi-step recommender systems. Recommendations affect user behavior; which in turn affect future recommendations through the retraining process. Feedback loops in recommender systems bias the inference of user preferences, compromise recommendation quality, and can homogenize recommendations. To this end, we propose CAFL, a causal adjustment algorithm that can provably break feedback loops. Across empirical studies, we find that CAFL improves recommendation quality and mitigates negative feedback effects. It also significantly improves predictive performance when compared to prior correction methods.

Furthermore, our results on homogenization show the importance of isolating feedback effects when evaluating models in dynamic setting. Our results indicate that the model's inductive bias and the number of datapoints, can sometimes have a stronger effect on homogenization than feedback loops. The picture of how and when homogenization occurs in recommender systems still remains incomplete. Future work that meticulously evaluates recommender systems in dynamic settings will likely shed light on this phenomenon.

3.6 Proofs

Proof of Theorem 2

Proof. We decompose the causal objective into terms where positivity holds and those where positivity is violated:

$$
\begin{aligned}
&L^{\text{causal}}(\Theta) \\
&= \frac{1}{t}\sum_{s=1}^{t}\sum_{u=1}^{U}\sum_{i=1}^{I}\mathbb{E}_{\mathbf{A}_s}\left[\mathbb{E}_{P(R_{s,ui}\,|\,\mathrm{do}(A_{s,ui}=A_{s,ui}))}\left[\log P_\Theta\left(R_{s,ui}\,|\,A_{s,ui}\right)\right]\right] \\
&= \sum_{s=1}^{t} c_t\left[\sum_{u=1}^{U}\sum_{i=1}^{I}\mathbb{E}_{\mathbf{A}_s}\left[\mathbb{E}_{P(R_{s,ui}\,|\,\mathrm{do}(A_{s,ui}=A_{s,ui}))}\left[\log P_\Theta\left(R_{s,ui}\,|\,A_{s,ui}\right)\right]\right]\right] \\
&= \sum_{s=1}^{t} c_t\left[\sum_{u=1}^{U}\sum_{i=1}^{I}\sum_{a_{s,ui}\in\{0,1\}}\mathbb{E}_{P(R_{s,ui}\,|\,\mathrm{do}(A_{s,ui}=a_{s,ui}))}\left[\mathbb{1}\{A_{s,ui}=a_{s,ui}\}\cdot\log P_\Theta\left(R_{s,ui}\,|\,A_{s,ui}\right)\right]\right] \\
&= \sum_{s=1}^{t} c_t\left[\sum_{\substack{u,i:\\ P(A_{s,ui}=\\ a_{s,ui}\,|\,\widehat{\Theta}_{s-1})>0}}\sum_{a_{s,ui}\in\{0,1\}}\mathbb{E}_{P(R_{s,ui}\,|\,\mathrm{do}(A_{s,ui}=a_{s,ui}))}\left[\mathbb{1}\{A_{s,ui}=a_{s,ui}\}\cdot\log P_\Theta\left(R_{s,ui}\,|\,A_{s,ui}\right)\right]\right] \\
&\quad+\sum_{s=1}^{t} c_t\left[\sum_{\substack{u,i:\\ P(A_{s,ui}=\\ a_{s,ui}\,|\,\widehat{\Theta}_{s-1})=0}}\sum_{a_{s,ui}\in\{0,1\}}\mathbb{E}_{P(R_{s,ui}\,|\,\mathrm{do}(A_{s,ui}=a_{s,ui}))}\left[\mathbb{1}\{A_{s,ui}=a_{s,ui}\}\cdot\log P_\Theta\left(R_{s,ui}\,|\,A_{s,ui}\right)\right]\right].
\end{aligned}
$$

The first equation is due to Section 3.3; the second equation is due to the stationary assumption of intervention distributions (i.e. the second assumption of Theorem 2); the third equation is an unbiased estimator of the expectation over $\mathbf{A}_s$; the fourth equation separates the loss into two terms, one where positivity holds and the other where positivity fails. $\widehat{L}^{\text{obs}}_{s,u,i}(\Theta\,;\,a_{s,ui})$ in the theorem is an unbiased estimator of the second term, following the same inverse probability argument as in Proposition 1 and section 3.3. $\widehat{L}^{\text{unobs}}_{s,u,i}(\Theta\,;\,a_{s,ui})$ is an unbiased estimator of the first term due to the stationary assumption of intervention distributions, together with the inverse probability argument as in Proposition 1 and section 3.3. □

Chapter 4

Component Interactions in Two-Stage Recommender Systems

4.1 Introduction

In the last two chapters we studied recommendation dynamics in the idealized setting of single-stage recommender systems. However, in practice these single-stage systems are unable to handle the scale of data with which internet platforms must contend with. A key technical challenge is ensuring recommender systems can sift through billions of items to deliver a personalized experience to millions of users with *low response latency*. A widely adopted solution to this problem are two-stage recommender systems [31, 47, 64, 243, 247] where (i) a set of computationally efficient *nominators* (or *candidate generators*) preselects a small number of candidates, which are then (ii) further narrowed down, reranked, and served to the user by a slower but more statistically accurate *ranker*.

Nominators are often heterogeneous, ranging from associative rules to recurrent neural networks [45]. A popular choice are matrix factorization [132, 170] and two-tower [243] architectures which model user feedback by the dot product between user and item embeddings. While user embeddings often evolve with the changing context of user interactions, item embeddings can typically be precomputed before deployment. The cost of candidate generation is thus dominated by the (approximate) computation of the embedding dot products. In contrast, the ranker often takes *both* the user and item features as input, making the computational cost linear in the number of items even at deployment [47, 155].

With few exceptions [102, 121, 155], two-stage specific literature is sparse compared to that on *single-stage* systems (i.e., recommenders which do not construct an explicit candidate set within a separate nominator stage [e.g., 44, 92, 111, 132, 170, 190, 205]). This is especially concerning given the considerable ethical challenges entailed by the enormous reach of two-stage systems: according to the recent systematic survey by Milano, Taddeo, and Floridi [165], recommender systems have been (partially) responsible for unfair treatment of disadvantaged groups, privacy leaks, political polarization, spread of misinformation, and 'filter bubble' or 'echo chamber' effects. While many of these issues are primarily within the realm of 'human–algorithm' interactions, the additional layer of 'algorithm–algorithm' interactions introduced by the two-stage systems poses a further challenge to understanding and alleviating them.

The main aim of our work is thus to narrow the knowledge gap between single- and two-stage systems, particularly in the context of *score-based* algorithms. Our main contributions are:

1. We show two-stage recommenders are significantly affected by interactions between the ranker and the nominators over a variety of experimental settings (Section 4.3).

2. We investigate these interactions theoretically (Section 4.3), and find that while independent ranker training typically works well (Proposition 1), the same is not the case for the nominators where two popular training schemes can both result in performance no better than that of a uniformly random recommender (Proposition 2).

3. Responding to the highlighted issues with *independent* training, we identify specialization of nominators to smaller subsets of the item pool as a source of potentially large performance gains. We thus propose a *joint* Mixture-of-Experts [114, 118] style training which treats each nominator as the expert for its own item subset. The ability to learn the item pool division alleviates the issues caused by the typically lower modeling capacity of the nominators, and empirically leads to improved precision and recall at K (Section 4.4).

4.2 Two-stage recommender systems

The goal of recommender systems is to learn a policy π which maps contexts $x \in \mathcal{X}$ to distributions $\pi(x)$ over a finite set of items (or *actions*) $a \in \mathcal{A}$, such that the expected reward $\mathbb{E}_x \left[\mathbb{E}_{a \sim \pi(x)} \left[r_a \,|\, x\right]\right]$ is maximized. The context x represents information about the user and items (e.g., interaction history, demographic data), and r_a is the user feedback associated with item a (e.g., rating, clickthrough, watch-time). We assume that $r_a | x$ has a well-defined fixed mean $f^\star(x, a) \coloneqq \mathbb{E}\left[r_a \,|\, x\right]$ for all the (x, a) pairs. To simplify, we further assume only one item a_t is to be recommended for each given context x_t, where $t \in [T]$ with $[T] \coloneqq \{1, \ldots, T\}$ for $T \in \mathbb{N}$.

Two-stage systems differ from the single-stage ones by the two-step strategy of selecting a_t. First, each nominator $n \in [N]$ picks a single candidate $a_{n,t}$ from its *assigned pool* $\mathcal{A}_n \subseteq \mathcal{A}$ ($\mathcal{A}_n \neq \varnothing$, $\bigcup_n \mathcal{A}_n = \mathcal{A}$). Second, the ranker chooses an item a_t from the *candidate pool* $\mathcal{C}_t \coloneqq \{a_{1,t}, \ldots, a_{N,t}\}$, and observes the reward $r_t = r_{ta_t}$ associated with a_t. Since the goal is *expected* reward maximization, recommendation quality can be measured by *instantaneous regret* $r_t^\star - r_t$ where $r_t^\star = r_{ta_t^\star}$ is the reward associated with an optimal arm $a_t^\star \in \arg\max_{a \in \mathcal{A}} f^\star(x_t, a)$. This leads us to an important identity for the *(cumulative) regret*

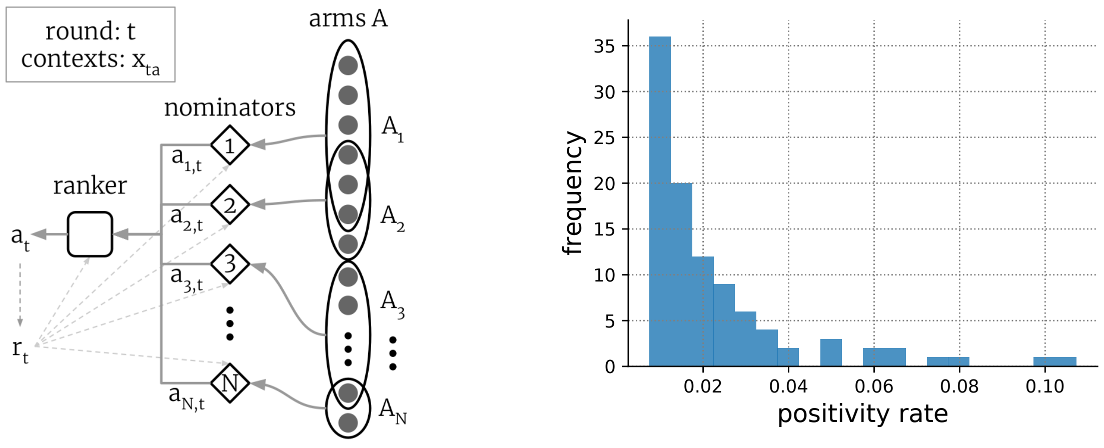


Figure 4.2: Left: The two-stage recommendation setup. **Right:** Amazon reward histogram. The top 5 arms are responsible for 19.22% whereas the bottom 50 only for 19.85% of the positive rewards.

in two-stage systems which is going to be used throughout:

$$\mathrm{R}_T^{\mathtt{2s}} = \sum_{t=1}^{T} r_t^\star - r_t = \underbrace{\sum_{t=1}^{T}(r_t^\star - r_{t\widetilde{a}_t})}_{=:\mathrm{R}_T^{\mathtt{N}}} + \underbrace{\sum_{t=1}^{T}(r_{t\widetilde{a}_t} - r_t)}_{=:\mathrm{R}_T^{\mathtt{R}}} ,$$

with $\widetilde{a}_t \in \arg\max_{a \in \mathcal{C}_t} f^\star(x_t, a)$. In words, $\mathrm{R}_T^{\mathtt{N}}$ is the *nominator regret*, quantifying the difference between the best action presented to the ranker $\widetilde{a}_t$ and the best overall action $a_t^\star$, and $\mathrm{R}_T^{\mathtt{R}}$ is the *ranker regret* which measures the gap between the choice of the ranker a_t and the best action in the candidate set $\mathcal{C}_t$. The two-stage recommendation process is summarized in Fig. 4.2 (left).

While Section 4.2 is typical for the *bandit* literature (data collected interactively, only r_{ta_t} revealed for each t), we also consider the *supervised* learning case (a dataset with all $\{r_{ta}\}_{ta}$ revealed is given). In particular, Section 4.3 presents an empirical comparison of single- and two-stage systems in the bandit setting, followed by a theoretical analysis with implications for both the learning setups.

4.3 Comparing single- and two-stage systems

Since the ranker and nominators could each be deployed independently, one may wonder whether the performance of a two-stage system is significantly affected by factors beyond the hypothetical single-stage performance of its components. This question is both theoretically (new developments needed?) and practically interesting (e.g., training components independently, as common, assumes targeting single-stage behavior is optimal). In Section 4.3, we empirically show that while factors known from the single-stage literature also affect two-stage systems, there are *two-stage specific properties* which can be even more important. Section 4.3 then investigates these properties theoretically, revealing a non-trivial interaction between the nominator training objective and the item pool allocations $\{\mathcal{A}_n\}_n$.

Empirical observations

Setup

We study the effects of item pool size, dimensionality, misspecification, nominator count, and the choice of ranker and nominator algorithms *in the bandit setting*. We compare single- and two-stage systems where each (component) models the expected reward as a *linear* function $f_t(x, a) = \langle \widehat{\theta}_t, x_a \rangle$ (x_a differs for each a). Abbreviating $x_t = x_{ta_t}$, the estimates are converted into a policy via either:

1. **UCB (U)** [9, 146] which computes the ridge-regression estimate with regularizer $\lambda > 0$

$$\widehat{\theta}_t \coloneqq \Sigma_t \sum_{i=1}^{t-1} x_i r_i \, , \qquad \Sigma_t \coloneqq \left(\lambda I + \sum_{i=1}^{t-1} x_i x_i^\top \right)^{-1} ,$$

   and selects actions with exploration bonus $\alpha > 0$: $a_t \in \arg\max_a \langle \widehat{\theta}_t, x_{ta} \rangle + \alpha \sqrt{x_{ta}^\top \Sigma_t x_{ta}}\, .$

2. **Greedy (G)** [13, 123] which can be viewed as a special case of UCB with $\alpha = 0$.

The $\arg\max$ is restricted to $\mathcal{A}_n$ (resp. $\mathcal{C}_t$) in two-stage systems, with pool allocation $\{\mathcal{A}_n\}_n$ designed to minimize overlaps and approximately equalize the number of items in each pool (see Section 4.8).

We chose to make the above restrictions of our experimental setup to limit the large number of design choices two-stage recommenders entail (architecture and hyperparameters of each nominator and the ranker, item pool allocation, number of nominators, etc.), and with that isolate the variation in performance to only a few factors of immediate interest.

We use one synthetic and one real-world dataset. The **synthetic dataset** is generated using a linear model $r_{ta} = \langle \theta^\star, x_{ta} \rangle + \varepsilon_{ta}$ for each time step $t \in [T]$ and action $a \in \mathcal{A}$. The vector $\theta^\star$ is drawn uniformly from the d-dimensional unit sphere at the beginning and then kept fixed, the contexts x_{ta} are sampled independently from $\mathcal{N}(0, I)$, and $\varepsilon_{ta} \sim \mathcal{N}(0, 0.01)$ is independent observation noise.

The **real-world dataset** 'AmazonCat–13K' contains Amazon reviews and the associated product category labels [27, 163].[1] Since 'AmazonCat–13K' is a multi-label classification dataset, we convert it into a bandit one by assigning a reward of one for correctly predicting any one of the categories to which the product belongs, and zero otherwise. An $|\mathcal{A}|$–armed linear bandit is then created by sampling $|\mathcal{A}|$ reviews uniformly from the whole dataset,

[1]We did not use 'MovieLens' [90] since it contains little useful contextual information as evidenced by its absence even from the state-of-the-art models [192]. Two-stage recommenders are only used when context matters as otherwise all recommendations could be precomputed and retrieved from a database at deployment.

and treating the associated features as the contexts $\{x_{ta}\}_{a \in \mathcal{A}}$. This method of conversion is standard in the literature [63, 79, 152, 155].

We use only the raw text features, and convert them to 768-dimensional embeddings using the HuggingFace pretrained model `'bert-base-uncased'` [58, 239];[2] we further subset to the first $d = 400$ dimensions of the embeddings, which does not substantially affect the results. Because we are running thousands of different experiment configurations (counting the varying seeds), we further reduce the computational complexity by subsetting from the overall 13K to only 100 categories. Since most of the products belong to 1–3 categories, we take the categories with 3rd to 102nd highest occurrence frequency. This ensures less than 5% of the data points belong to none of the preserved categories, and overall 10.73% reward positivity rate with strong power law decay (Fig. 4.2, right).

While the ranker can always access all d features, the usual lower flexibility of the nominators (*misspecification*) is modelled by restricting each to a different *random subset* of s out of the total d features on both datasets. This is equivalent to forcing the corresponding regression parameters to be zero. Both UCB and Greedy are then run with x_t replaced by the s-dimensional $x_{n,t} = x_{n,ta_t}$ everywhere. The same restriction is applied to the single-stage systems for comparison. In all rounds, each nominator is updated with $(x_{n,t}, a_t, r_t)$, regardless of whether $a_t \in \mathcal{A}_n$ (this assumption is revisited in Section 4.3). Thirty independent random seeds were used to produce the (often barely visible) two-sigma standard error regions in Figs. 4.3 and 4.4. More details—including the hyperparameter tuning protocol and additional results—can be found in Sections 4.8 and 4.9.

Results

Starting with the synthetic results in Fig. 4.3, we see that the number of arms and the feature dimension d are both correlated with increased regret in single- *and* two-stage systems. Another similarity between all the algorithms is that misspecification—as measured by d/s—also has a significant effect on performance.[3] This is also the case for the Amazon dataset in Fig. 4.4.

The influence of the number of arms, dimensionality, and misspecification on single-stage systems is well known [139]. Figs. 4.3 and 4.4 suggest similar effects also exist for two-stage systems. On the other hand, while the directions of change in regret agree, the magnitudes do not. In particular, two-stage systems perform significantly better than their single-stage counterparts. This is possible because the ranker can exploit its access to all d features to improve upon even the best of the nominators (recall that nominators and single-stage

[2] Encoded dataset: https://twostage.s3-us-west-2.amazonaws.com/amazoncat-13k-bert.zip.

[3] Misspecification error typically translates into a linear regret term ϵT [48, 61, 70, 71, 80, 136, 140]. We can thus gain some *intuition* for the concavity of $d/s \mapsto \mathrm{R}_T$ from the L^2 error $\epsilon = \min_{\theta_n} (\mathbb{E}\left[(r_a - \langle \theta_n, x_{n,a} \rangle)^2\right])^{1/2}$ where $a \sim \mathrm{Unif}(\mathcal{A})$ [136]. Using $x_{ta} \sim \mathcal{N}(0, I)$, the minimum is achieved by $(\mathbb{E}\left[x_{n,a} x_{n,a}^\top\right])^{-1} \mathbb{E}\left[x_{n,a} r_a\right] = \theta_n^\star$, with $\theta_n^\star$ the s entries of $\theta^\star$ corresponding to the dimensions available to n. The L^2 error is thus a concave function of d/s by symmetry: $\epsilon = \sqrt{\mathbb{E}\left[\mathbb{E}\left[(r_{ta} - \langle \theta_n^\star, x_{n,ta} \rangle)^2 \mid \theta^\star\right]\right]} = \sqrt{\mathbb{E}\left[\|\theta^\star\|_2^2 - \|\theta_n^\star\|_2^2\right]} = \sqrt{1 - s/d}$.

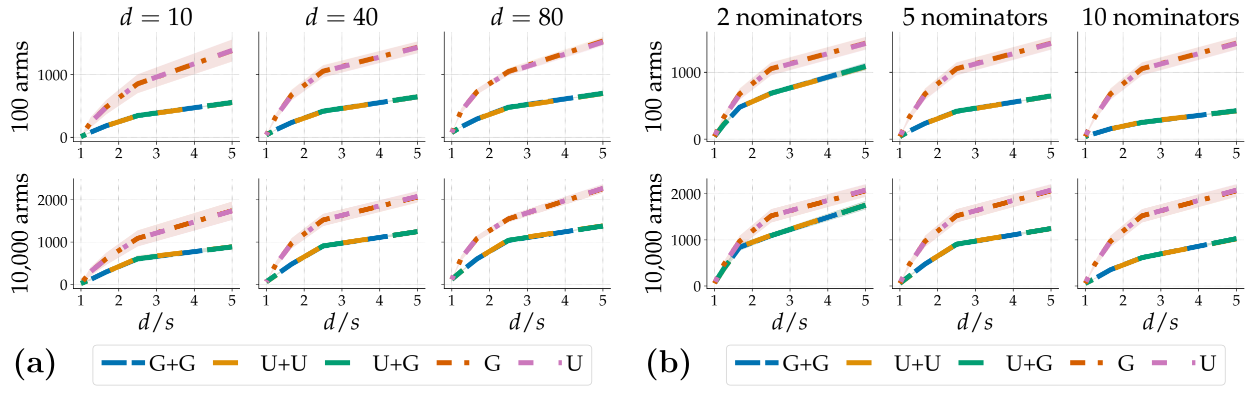


Figure 4.3: Synthetic data results. The x-axis is the ratio between the true feature dimension d and the size of the subset available to the nominators and the single-stage systems s. The y-axis shows the expected regret at $T = 1000$. In plot **(a)**, $N = 5$ nominators are used, and columns represent the total number of features d. In plot **(b)**, $d = 40$ features are used, and columns show the number of nominators N. The legend describes model architectures, where two-stage systems are labeled by '[ranker]+[nominator]' (e.g., 'U+G' is a UCB ranker with Greedy nominators).

systems share the same model architecture). In other words, *the single-stage performance of individual components does not fully explain the two-stage behavior.*

To develop further intuition about the differences between single- and two-stage systems, we turn our attention to the Amazon experiments in Fig. 4.4. The top row suggests the performance of two-stage systems improves as the number of nominators grows. Strikingly, the accompanying UCB ranker + nominator plots in the bottom row show the nominator regret $\mathrm{R}_T^{\mathtt{N}}$ dominates when there are few nominators, but gives way to the ranker regret $\mathrm{R}_T^{\mathtt{R}}$ as their number increases.

To explain why, first note that the single-stage performance of the ranker can be read off from the bottom left corner of each plot where $d = s$ (because all the components are identical at initialization, and then updated with the same data). Since the size of the candidate set $\mathcal{C}_t$ increases with the number of nominators, the two-stage performance in the $d > s$ case eventually approaches that of the single-stage UCB ranker as well, even if the nominators are no better than random guessing. In fact, because $\approx 10\%$ of the items yield optimal reward, the probability that a set of ten uniformly random nominators with non-overlapping item pools nominates at least one optimal arm is on average $1 - (\frac{9}{10})^{10} \approx 0.65$, i.e., the instantaneous nominator regret would be zero 65% of the time.

To summarize, we have seen evidence that properties known to affect single-stage performance—number of arms, feature dimensionality, misspecification—have similar qualitative effects on two-stage systems. However, two-stage recommenders perform significantly better than any of the nominators alone, especially as the nominator count and the size of the candidate pool increase. Complementary to the evidence from offline learning [155],

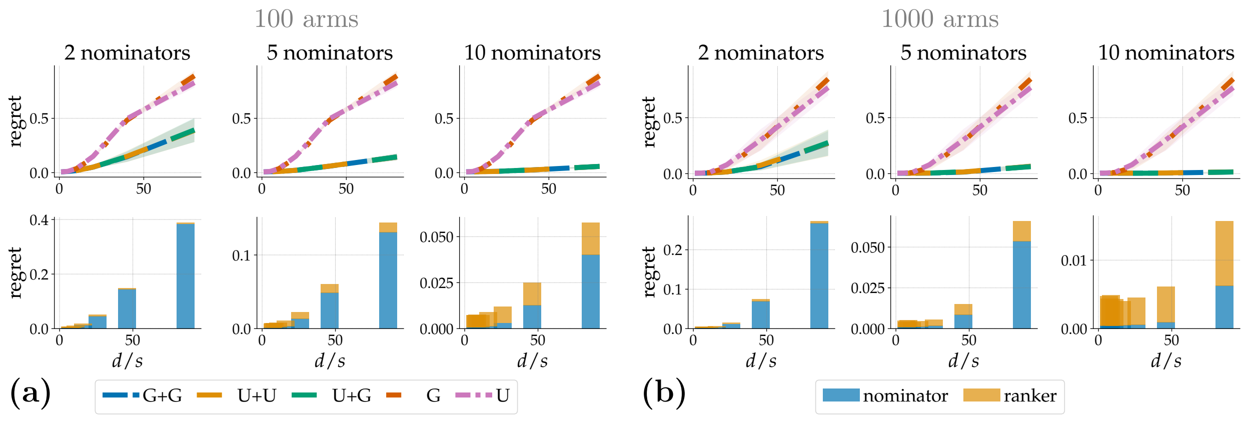


Figure 4.4: Amazon data results. The axes are the same as in Fig. 4.3, except the y-axis is plotted at $T = 5000$ with the regret divided by that of a uniformly random agent. The feature dimension is fixed to $d = 400$, and the number of arms to 100 in plot **(a)**, and to 1000 in plot **(b)**. The columns represent varying number of nominators N. The legend is shared, where the one in **(a)** corresponds to the top row plots and has the same interpretation as in Fig. 4.3, and the one in **(b)** belongs to the bottom row plots which show the proportion of ranker and nominator regret (see Section 4.2) for the `'U+U'` two-stage system as a representative example (all two-stage systems perform similarly here).

and the effect of ranker pretraining [102], these observations add to the case that two-stage systems should not be treated as just the sum of their parts. We add theoretical support to this argument in the next section.

Theoretical observations

The focus of Section 4.3 was on linear models in the bandit setting. We lift the bandit assumption later in this section, and relax the class of studied models to *least-squares regression oracle* (LSO) based algorithms, which estimate the expected reward by minimizing the sum of squared errors and a regularizer $\|\cdot\|_{\mathcal{F}}$ over a given model class $\mathcal{F}$

$$f_t \in \arg\min_{f \in \mathcal{F}} \left\{ \|f\|_{\mathcal{F}} + \sum_{i=1}^{t-1} (r_i - f(x_i, a_i))^2 \right\} .$$

These estimates are then converted into a policy either greedily, $\pi_t(x) = \text{Unif}(\arg\max_a f_t(x, a))$, or by incorporating an exploration bonus as in LinUCB [9, 46, 53, 146, 199], or the more recent class of black-box reductions from bandit to online or offline regression [70, 71, 72, 136, 211]. The resulting algorithms are often minimax optimal, and (some) also perform well on real-world data [28].

We choose LSO based algorithms because they (i) include the Greedy and (Lin)UCB models studied in the previous section, and (ii) allow for an easier exposition than the similarly popular cost-sensitive classification approaches [e.g., 3, 45, 47, 62, 138]. The following proposition is an application of the fact that algorithms like LinUCB or SquareCB [1, 71, 72] provide regret guarantees robust to contexts chosen by an *adaptive* adversary, and thus also to those chosen by the nominators.

Proposition 1. *Assume the ranker achieves a* single-stage *regret guarantee* $\mathrm{R}_T \leq \mathrm{B}_T^{\mathtt{R}}$ *for some constant* $\mathrm{B}_T^{\mathtt{R}} \in \mathbb{R}$ *(either in expectation or with high probability), even if the contexts* $\{x_t\}_{t=1}^T$ *are chosen by an adaptive adversary. The ranker regret then satisfies*

$$\mathrm{R}_T^{\mathtt{R}} = \sum_{t=1}^{T} r_{t\tilde{a}_t} - r_t \leq \mathrm{B}_T^{\mathtt{R}} \,,$$

in the sense of the original bound (i.e., in expectation, or with high probability).

While proving Proposition 1 is straightforward, its consequences are not. First, if $\mathrm{B}_T^{\mathtt{R}}$ is in some sense optimal, then Section 4.2 implies the two-stage regret $\mathrm{R}_T^{\mathtt{2s}}$ will be dominated by the nominator regret $\mathrm{R}_T^{\mathtt{N}}$ (unless it satisfies a similar guarantee). Second, $\mathrm{R}_T^{\mathtt{R}} \leq \mathrm{B}_T^{\mathtt{R}}$ holds exactly when the ranker is trained in the 'single-stage mode', i.e., the tuples (x_t, a_t, r_t) are fed to the algorithm without any adjustment for the fact $\mathcal{C}_t$ is selected by a set of adaptive nominators from the whole item pool $\mathcal{A}$.

The above however does not mean that the ranker has no substantial effect on the overall behavior of the two-stage system. In particular, the feedback observed by the ranker also becomes the feedback observed by the nominators, which has the *primary effect* of influencing the nominator regret $\mathrm{R}_T^{\mathtt{N}}$, and the *secondary effect* of influencing the candidate pools $\mathcal{C}_t$ (which creates a feedback loop). The rest of this section focuses on the primary effect, and in particular its dependence on how the nominators are trained and the item pools $\{\mathcal{A}_n\}_n$ allocated.

Pitfalls in designing the nominator training objective

The *primary effect* above stems from a key property of two-stage systems: unlike the ranker, nominators do not observe feedback for all items they choose. While importance weighting can be used to adjust the nominator training objective [155], it does not tell us what adjustment would be optimal.

We thus study two major types of updating strategies: (i) 'training-on-all,' and (ii) 'training-on-own.' Both can be characterized in terms of the following weighted oracle objective for the n^{th} nominator

$$f_{n,t} \in \operatorname*{arg\,min}_{f_n \in \mathcal{F}_n} \left\{ \|f_n\|_{\mathcal{F}_n} + \sum_{i=1}^{t-1} w_{n,i} (r_i - f_n(x_{n,i}\,, a_i))^2 \right\} ,$$

where $\mathcal{F}_n$ is the class of functions the nominator can fit, $\|\cdot\|_{\mathcal{F}_n}$ the regularizer, and $w_{n,t} = w_{n,a_t} \geq 0$ the weight. **'Training-on-all'**—used in Section 4.3—takes $w_{n,a} = 1$ for all (n, a), which means *all* data points are valued equally regardless of whether a particular a_t belongs to the nominator's pool $\mathcal{A}_n$. 'Training-on-all' may potentially waste the already limited modelling capacity of the nominators if the pools $\mathcal{A}_n$ are not identical. The **'training-on-own'** alternative therefore uses $w_{n,a} = \mathbb{1}\{a \in \mathcal{A}_n\}$ so that only the data points for which $a_t \in \mathcal{A}_n$ influence the objective.[4]

While 'training-on-all' and 'training-on-own' are not the only options we could consider, they are representative of two very common strategies. In particular, 'training-on-all' is the default easy-to-implement option which sometimes performs surprisingly well [28, 198]. In contrast, 'training-on-own' approximates the (on-policy) 'single-stage mode' where the nominator observes feedback only for the items it selects (in particular, $a_t \in \mathcal{A}_n$ only if $a_t = a_{n,t}$ when the pools are non-overlapping).

Proposition 2 below shows that neither 'training-on-all' nor 'training-on-all' is guaranteed to perform better than random guessing in the infinite data limit ($T \to \infty$). We consider the *linear* setting $f^\star(x_t, a) = \langle \theta^\star, x_{ta} \rangle$ for all (x_t, a), $\theta^\star$ fixed, with nominators using ridge regression oracles $f_{n,t}(x_{n,t}, a) = \langle \widehat{\theta}_{n,t}, x_{n,ta} \rangle$ as defined in Item 1, $\lambda \geq 0$ fixed, and $x_{n,ta}$ again a *subset* of the full feature vector x_{ta}. We also assume the nominators take the predicted best action with non-vanishing probability (Assumption 2), which holds for all the cited LSO based algorithms.

Assumption 2. *Let $f_{n,t}$ be as in Section 4.3, and denote $\mathcal{A}^{\mathtt{G}}_{n,t} := \arg\max_{a \in \mathcal{A}_n} f_{n,t}(x_{n,t}, a)$. We assume there is a universal constant $\delta > 0$ such that for all $n \in [N]$ and $a \in \mathcal{A}_n$ with $\limsup T^{-1} \sum_{1 \leq t \leq T} \mathbb{P}(a_t^\star = a) > 0$, we have $\limsup T^{-1} \sum_{1 \leq 1 \leq T} \mathbb{P}(a_{n,t} \in \mathcal{A}^{\mathtt{G}}_{n,t} \,|\, a_t^\star = a) \geq \delta$.*

Proposition 2. *In both the supervised and the bandit learning setup, there exist two* distinct *context distributions with pool allocations $\{\mathcal{A}_n\}_n$, and $r_a \in [0, 1]$ almost surely (a.s.) for all $a \in \mathcal{A}$, such that 'training-on-own' (resp. 'training-on-all') leads to asymptotically linear two-stage regret*

$$\limsup_{T \to \infty} \frac{\mathbb{E}\left[\mathrm{R}_T^{\mathtt{2s}}\right]}{T} > 0 \,.$$

Moreover, the asymptotic regret of 'training-on-all' is sublinear under the context distribution and pool allocation where 'training-on-own' suffers linear regret, and vice versa.

Proof. Throughout, we use $\widehat{\theta}_{n,t} \to (\mathbb{E}\left[x_{n,a} x_{n,a}^\top\right])^{-1} \mathbb{E}\left[x_{n,a} r_a\right] =: \theta_n^\star$ (a.s.) by Lemma 4 (Section 4.7), assuming invertibility and that a_t is i.i.d.; note $\theta_n^\star = \arg\min_{\theta_n} \mathbb{E}\left[(r_a - \langle \theta_n, x_{n,a} \rangle)^2\right]$. We allow any zero mean reward noise which satisfies $r_a \in [0, 1]$ (a.s.) for all $a \in \mathcal{A}$.

(I) Supervised setup. Take two nominators, $\mathcal{A}_1 = \{a_{(1)}\}$, $\mathcal{A}_2 = \{a_{(2)}, a_{(3)}\}$, a single context

$$X := \begin{bmatrix} — & x_{a_{(1)}} & — \\ — & x_{a_{(2)}} & — \\ — & x_{a_{(3)}} & — \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} ,$$

[4]There are two possible definitions of 'training-on-own': (i) $w_{n,t} = \mathbb{1}\{a_t \in \mathcal{A}_n\}$; (ii) $w_{n,t} = \mathbb{1}\{a_t = a_{n,t}\}$. While the main text considers the former, Proposition 2 can be extended to the latter with minor modifications.

and *restrict the nominators to the last two columns of X*. As $|\mathcal{A}_1| = 1$, the first nominator always proposes $a_{(1)}$, disregards of its fitted model. Since all rewards are revealed and used to update the model in the supervised setting, 'training-on-all' $t \to \infty$ limit for the second nominator's $\widehat{\theta}_{2,t}$ is

$$\begin{aligned}\theta_2^\star &= \operatorname*{arg\,min}_{\beta \in \mathbb{R}^2} \mathbb{E}_{a \sim \text{Unif}(\mathcal{A})} \left[(r_a - \langle \beta, x_{2,a} \rangle)^2 \right] \\ &= \operatorname*{arg\,min}_{\beta \in \mathbb{R}^2} \left\{ (\overline{r}_1 + \beta_2)^2 + (\overline{r}_2 - \beta_1)^2 + (\overline{r}_3 - \beta_2)^2 \right\} = [\overline{r}_2, \tfrac{\overline{r}_3 - \overline{r}_1}{2}]^\top ,\end{aligned}$$

where $\overline{r}$ is the mean reward vector for the single context ($\theta^\star$ is then $X^{-1}\overline{r}$). If we take, e.g., $\overline{r} = [\frac{1}{4}, \frac{1}{2}, 1]^\top$, then $a_{(3)} \neq \operatorname{arg\,max}_{a \in \mathcal{A}_2} \langle \theta_2^\star, x_{2,a} \rangle = a_{(2)}$. On the other hand, 'training-on-own' would yield $\theta_2^\star = [\overline{r}_2, \overline{r}_3]^\top$, and thus correctly identify $a_{(3)}$ via $\operatorname{arg\,max}_{a \in \mathcal{A}_2} \langle \theta_2^\star, x_{2,a} \rangle$.

In contrast, consider the modified setup $\mathcal{A}_1 = \{a_{(1)}\}$, $\mathcal{A}_2 = \{a_{(2)}, a_{(3)}, a_{(4)}\}$

$$X := \begin{bmatrix} — & x_{a_{(1)}} & — \\ — & x_{a_{(2)}} & — \\ — & x_{a_{(3)}} & — \\ — & x_{a_{(4)}} & — \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .$$

Restricting nominators to the last two columns of X, 'training-on-own' would yield $\theta_2^\star = [\overline{r}_3, \frac{\overline{r}_4 - \overline{r}_2}{2}]^\top$ under full feedback access, whereas 'training-on-all' would converge to $\theta_2^\star = [\overline{r}_3, \frac{\overline{r}_4 - \overline{r}_2 - \overline{r}_1}{3}]^\top$. Hence with, e.g., $\overline{r} = [\frac{3}{4}, 1, \frac{1}{6}, \frac{7}{8}]^\top$, 'training-on-own' would make the second nominator pick $a_{(3)}$ via arg max, but 'training-on-all' would successfully identify the optimal $a_{(2)}$.

(II) Bandit setup. Take X from Section 4.3, but use $\overline{r} = [\frac{3}{4}, \frac{7}{8}, \frac{1}{6}, 1]^\top$ and the associated $\theta^\star = X^{-1}\overline{r}$. For each $j \in [4]$, let $X_{(j)}$ be a deterministic context matrix which is the same as X except all but the j^{th} row are replaced by zeros. Observe that for each j, the mean reward vector $X_{(j)}\theta^\star$ has exactly one *strictly* positive component, and thus $a_t^\star = a_{(j)}$ when $X_{(j)}$ is drawn.

Let $\text{Unif}(\{X_{(j)}\}_j)$ be the context distribution, $\mathcal{A}_1 = \{a_{(1)}\}$, $\mathcal{A}_2 = \{a_{(2)}, a_{(3)}, a_{(4)}\}$, and restrict nominators to the last two columns of each sampled $X_{(j)}$. We employ a proof by contradiction. Assume $\limsup T^{-1}\mathbb{E}\left[\mathrm{R}_T^{\mathtt{2s}}\right] \to 0$. Then $\widehat{\theta}_{2,t} \to \theta_2^\star$ in probability by Lemma 5 (Section 4.7), with $\theta_2^\star$ as stated right after Section 4.3 for both the update rules. Since $\theta_{2,2}^\star = \frac{1}{16} > 0$ under 'training-on-own', resp. $\theta_{2,2}^\star = -\frac{5}{24} < 0$ under 'training-on-all', $\operatorname{arg\,max}_{a \in \mathcal{A}_2} \langle \theta_2^\star, x_{2,ta} \rangle$ would fail to select $a_t^\star$ when $X_{(2)}$, resp. $X_{(4)}$, is sampled (see Section 4.3). This would translate into an expected instantaneous regret of at least $\Delta := \min_i \overline{r}_i = \frac{1}{6} > 0$. Hence by Section 4.2 and $\mathbb{P}(a_t^\star = a) = |\mathcal{A}|^{-1}$

$$\mathbb{E}\left[\mathrm{R}_T^{\mathtt{2s}}\right] \geq \mathbb{E}\left[\mathrm{R}_T^{\mathtt{N}}\right] \geq \frac{\Delta}{|\mathcal{A}|} \sum_{a \in \mathcal{A}_2} \mathbb{P}(a_{2,t} \neq a \,|\, a_{2,t} \in \mathcal{A}_{2,t}^{\mathtt{G}}, a_t^\star = a)\, \mathbb{P}(a_{2,t} \in \mathcal{A}_{2,t}^{\mathtt{G}} \,|\, a_t^\star = a) .$$

For 'training-on-own', $\mathbb{P}(a_{2,t} \neq a_{(2)} \,|\, a_{2,t} \in \mathcal{A}_{2,t}^{\mathtt{G}}, a_t^\star = a_{(2)}) = \mathbb{P}(\widehat{\theta}_{2,t2} \cdot (-1) < 0) \to 1$ by the above established $\widehat{\theta}_{2,t} \to \theta_2^\star$ in probability, and the continuous mapping theorem.

Analogously for 'training-on-all'. Hence $\limsup T^{-1}\mathbb{E}\left[\mathrm{R}_T^{\mathtt{2s}}\right] \geq |\mathcal{A}|^{-1}\Delta\delta > 0$ by Assumption 2, a contradiction, meaning both modes of training fail, but for a different item ($a_{(3)}$ is picked out correctly by both again by the convergence in probability). To make $\limsup T^{-1}\mathbb{E}\left[\mathrm{R}_T^{\mathtt{N}}\right] \to 0$ for exactly one of the two setups, add a *third nominator* with $\mathcal{A}_3 = \{a_{(2)}\}$, resp. $\mathcal{A}_3 = \{a_{(4)}\}$, so that $\mathbb{P}(a_t^\star \in \mathcal{C}_t) \to 1$. □

Proposition 2 shows that the nominator training objective can be all the difference between poor and optimal two-stage recommender.[5] Moreover, neither 'training-on-own' nor 'training-on-all' guarantees sublinear regret, and one can fail exactly when the other works. The main culprit is the difference between context distribution in and outside of each pool: combined with the misspecification, either one can result in more favorable optima from the overall two-stage performance perspective. This is the case *both in the supervised and the bandit setting.*

Proposition 2 can be trivially extended to higher number of arms and nominators (add embedding dimensions, let the new arms have non-zero embedding entries only in the new dimensions, and the expected rewards to be lower than the ones we used above). We think that the difference between the in- and out-pool distributions could be exploited to derive analogous results to Proposition 2 for non-linear (e.g., factorization based) models, although the proof complexity may increase.

To summarize, beyond the actual number of nominators identified in the previous section, we have found that the combination of training objective and pool allocation can heavily influence the overall performance. We use these insights to improve two-stage systems in the next section.

4.4 Learning pool allocations with Mixture-of-Experts

Revisiting the proof of Proposition 2, we see the employed pool allocations are essentially adversarial with respect to the context distributions. However, we are typically free to design the pools ourselves, with the only constraints imposed by computational and statistical performance requirements. Proposition 2 thus hints at a positive result: a good pool allocation can help us achieve an (asymptotically) optimal performance *even in cases where this is not possible using any one of the nominators alone.*

Crafting a good pool allocation *manually* may be difficult, and could lead to very bad performance if not done carefully (Proposition 2). We thus propose to *learn* the pool allocation using a **Mixtures-of-Experts** (MoE) [114, 117, 118, 244] based approach instead. A MoE computes predictions by weighting the individual expert (nominator) outputs using a trainable gating mechanism. The weights can be thought of as a *soft pool allocation* which

[5]Covington, Adams, and Sargin [47] reported empirically observing that the the training objective choice has an outsized influence on the performance of two-stage recommender systems. Proposition 2 can be seen as a theoretical complement which shows that the range of important choices goes beyond the selection of the objective.

allows each expert to specialize on a different subset of the input space. This makes the MoE more flexible than any one of the experts alone, alleviating the lower modeling flexibility of the nominators due to the latency constraints.

We focus on the Gaussian MoE [114], trained by likelihood maximization (Section 4.4). We employ gradient ascent which—despite its occasional failure to find a good local optimum [156]—is easy to scale to large datasets using a stochastic approximation of gradients

$$\frac{1}{T}\sum_{t=1}^{T}\log\sum_{n=1}^{N}p_{n,t}\exp\left\{-\frac{(r_t-\widehat{r}_{n,t})^2}{2\sigma^2}\right\}\approx\frac{1}{B}\sum_{t=1}^{B}\log\sum_{n=1}^{N}p_{n,t}\exp\left\{-\frac{(r_t-\widehat{r}_{n,t})^2}{2\sigma^2}\right\},$$

with $p_{n,t} \geq 0$ ($\sum_n p_{n,t} = 1$) the gating weight assigned to expert n on example t, $\widehat{r}_{n,t}$ the matching expert prediction, $\sigma > 0$ a hyperparameter approximating reward variance, and $B \in \mathbb{N}$ the batch size.

MoE provides a compelling alternative to a policy gradient style approach applied to the joint two-stage policy $\pi^{\mathtt{2s}}(a \,|\, x) = \sum_{a_1,\ldots,a_N} \pi^{\mathtt{R}}(a \,|\, x, \mathcal{C}) \prod_{n=1}^{N} \pi_n^{\mathtt{N}}(a_n \,|\, x)$ as done in [155]. In particular, a significant advantage of the MoE approach is that the sum over the *exponentially many* candidate sets $\mathcal{C} = \{a_1, \ldots, a_N\}$ is replaced by a sum over *only* N experts, which can be either computed exactly or estimated with dramatically smaller variance than in the candidate set case.

There are (at least) two ways of incorporating MoE into existing two-stage recommender deployments:

1. Use a *provisional* gating mechanism, and then distill the pool allocations from the learned weights, e.g., by thresholding the per arm average weight assigned to each nominator, or by restricting the gating network only to the item features. Once pools are divided, nominators and the ranker may be finetuned and deployed *using any existing infrastructure*.

2. Make the gating mechanism *permanent*, either as (i) a replacement for the ranker, or (ii) part of the nominator stage, reweighting the predictions before the candidate pool is generated. This necessitates change of the existing infrastructure but can yield better recommendations.

Unusually for MoEs, we may want to use a different input subset for the gating mechanism and each expert depending on which of the above options is selected. We would like to emphasize that the MoE approach can be used with any *score-based* nominator architecture including but not limited to the linear models of the previous section. If some of the nominators are *not* trainable by gradient descent but *are* score-based, they can be pretrained and then plugged in during the MoE optimization, allowing the other experts to specialize on different items.

We use the 'AmazonCat-13K' dataset [27, 163] to investigate the setup with a logistic gating mechanism as a part of the nominator stage. We employ the same preprocessing as in Section 4.3. Due to the success of greedy methods in Section 4.3, and the existence of black-box reductions from bandit to offline learning [70, 211], we simplify by focusing only on offline evaluation. We compare the MoE against the same model except with the gating replaced by a random pool allocation fixed at the start.

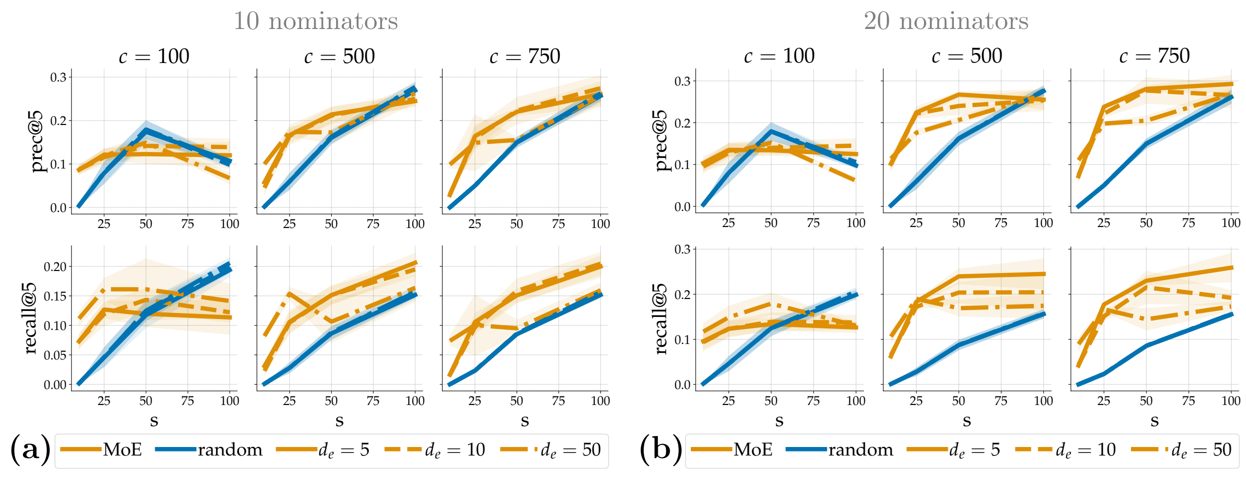


Figure 4.5: Mixture-of-Experts results on the 100-item Amazon dataset. The x-axis is the size of the BERT embedding dimension subset. The y-axis shows the average `precision@5` (top row) and `recall@5` (bottom row) over 50,000 entries from an independent test set (both set to zero for entries with no positive labels—about 5.5% of the test set). The columns in both plots **(a)** and **(b)** correspond to the number of examples per arm c in the training set. Ten (resp. twenty) nominators were used in **(a)** (resp. **(b)**). The legend shows whether pool allocations were learned (MoE) or randomly assigned (random), and the dimension of item embeddings d_e employed by both model types.

The experts in both models use a simple two-tower architecture, where d_e-dimensional dense embeddings are learned for each item, the s-dimensional subset of the BERT embeddings is mapped to $\mathbb{R}^{d_e}$ by another trained matrix, and the final prediction is computed as the dot product on $\mathbb{R}^{d_e}$. To enable low latency computation of recommendations, the gating mechanism models the logits $\{\log p_n\}_n$ as a sum of learned user and item embeddings. Further details are described in Section 4.8.

Fig. 4.5 shows that MoEs are able to outperform random pool allocation for most combinations of model architecture and training set size. The improved results in recall suggest that the specialization allows nominators to produce a more diverse candidate set. Since *the gating mechanism can learn to exactly recover any fixed pool allocation*, the *MoE can perform worse only when the optimizer fails or the model overfits*. This seems to be happening for the smallest training set size ($c = 100$ samples per arm), and also when the item embedding dimension d_e is high. In practice, these effects can be counteracted by tuning hyperparameters for the specific setting, regularization, or alternative training approaches based on expectation–maximization or tensor decomposition [117, 156, 244].

4.5 Other related work

Scalable recommender systems. Interest in scalable recommenders has been driven by the continual growth of available datasets [83, 145, 203, 223]. The two-stage architectures examined in this paper have seen widespread adoption in recommendation [31, 47, 64, 247, 249], and beyond [8, 246]. Our paper is specifically focused on recommender systems which means our insights may not transfer to application areas like information retrieval without adaptation.

Off-policy learning and evaluation. Updating the recommendation policy online, without human oversight, runs the risk of compromising the service quality, and introducing unwanted behavior. Offline learning from logged data [63, 173, 221, 226] is an increasingly popular alternative [45, 116, 155, 222]. It has also found applications in search engines, advertising, robotics, and more [5, 116, 144, 219].

Ensembling and expert advice. The goal of 'learning with expert advice' [4, 10, 149, 214] is to achieve performance comparable with the best expert if deployed on its own. This is not a good alternative to our MoE approach since two-stage systems typically outperform any one of the nominators alone (Section 4.3). A better alternative may possibly be found in the literature on 'aggregation of weak learners' [35, 36, 73, 74, 96], or recommender ensembling (see [38] for a recent survey).

4.6 Discussion

We used a combination of empirical and theoretical tools to investigate the differences between single- and two-stage recommenders. Our **first major contribution** is demonstrating that besides common factors like item pool size and model misspecification, the nominator count and training objective can have even larger impact on performance in the two-stage setup. As a consequence, two-stage systems *cannot* be fully understood by studying their components in isolation, and we have shown that the common practice of training each component independently may lead to suboptimal results. The importance of the nominator training inspired our **second major contribution**: identification of a link between two-stage recommenders and Mixture-of-Experts models. Allowing each nominator to specialize on a different subset of the item pool, we were able to significantly improve the two-stage performance. Consequently, splitting items into pools within the nominator stage is not just a way of lowering latency, but can also be used to improve recommendation quality.

Due to the the lack of access, a major limitation of our work is not evaluating on a production system. This may be problematic due to the notorious difficulty of offline evaluation [14, 134, 196]. We further assumed that recommendation performance is captured by a few measurements like regret or precision/recall at K, even though design of meaningful evaluation criteria remains a challenge [55, 93, 120, 167]; we caution against deployment without careful analysis of downstream effects and broader impact assessment. Several topics were left to future work: (i) extension of the linear regret proof to non-linear models such as those used in the MoE experiments; (ii) slate (multi-item) recommendation; (iii) theoretical understanding of how much can the ranker reduce the regret compared to the best of the (misspecified) nominators; (iv) alternative ways of integrating MoEs, including explicit distillation of pool allocations from the learned gating weights, learning the optimal number of nominators [187], using categorical likelihood [244], and sparse gating [68, 210].

Overall, we believe better understanding how two-stage recommenders work matters due to the enormous reach of the platforms which employ them. We hope our work inspires further inquiry into two-stage systems in particular, and the increasingly more common 'algorithm-algorithm' interactions between independently trained and deployed learning algorithms more broadly.

4.7 Auxiliary lemmas

Throughout the paper, we assume the **'stack of rewards model'** from chapter 4.6 of [139].

Lemma 4. *Let $\widehat{\theta}_t = \Sigma_t \sum_{i=1}^{t-1} x_i r_i$ where $\Sigma_t = (\lambda I + \sum_{i=1}^{t-1} x_i x_i^\top)^{-1}$ for some fixed $\lambda \geq 0$. Assume (x_i, r_i) are i.i.d. with $\mathbb{E}[x_1 r_1]$ and $\mathbb{E}\left[x_1 x_1^\top\right]$ well-defined, and the latter invertible. Then*

$$\widehat{\theta}_t \to (\mathbb{E}\left[x_1 x_1^\top\right])^{-1}\mathbb{E}[x_1 r_1] \qquad a.s.$$

Proof. Rewriting $\widehat{\theta}_{t+1} = (\frac{\lambda}{t} I + \frac{1}{t}\sum_{i=1}^{t} x_i x_i^\top)^{-1}\frac{1}{t}\sum_{i=1}^{t} x_i r_i$, $\frac{\lambda}{t} I + \frac{1}{t}\sum_{i=1}^{t} x_i x_i^\top \to \mathbb{E}\left[x_1 x_1^\top\right]$ and $\frac{1}{t}\sum_{i=1}^{t} x_i r_i \to \mathbb{E}[x_1 r_1]$ a.s. by the strong law of large numbers. Since $A \mapsto A^{-1}$ is continuous on the space of invertible matrices, the result follows by the continuous mapping theorem. □

Lemma 5. *Consider the setup from Part II of the proof of Proposition 2. Define*

$$\widehat{\theta}_{n,t} = \Sigma_{n,t} \sum_{i=1}^{t-1} w_{n,t} x_{n,i} r_i \,, \quad \Sigma_{n,t} = (\lambda I + \sum_{i=1}^{t-1} w_{n,i} x_{n,i} x_{n,i}^\top)^{-1} \,,$$

with $\lambda \geq 0$ fixed, and $\theta_n^\star = (\mathbb{E}\left[w_{n,a} x_{n,a} x_{n,a}^\top\right])^{-1}\mathbb{E}[w_{n,a} x_{n,a} r_a]$, $a \sim$ Unif$(\mathcal{A})$. Then $\widehat{\theta}_{n,t} \to \theta_n^\star$ in probability, if $\limsup T^{-1}\mathbb{E}[\mathrm{R}_T^{2\mathsf{s}}] \to 0$.

Proof. Since $x_{n,t} = 0$ unless $a_t = a_t^\star$ by construction, $\widehat{\theta}_{n,t+1}$ is equal to

$$\left(\frac{\lambda}{t} I + \frac{1}{t}\sum_{i=1}^{t}\sum_{j=1}^{|\mathcal{A}|} \mathbb{1}\{a_i^\star = a_i = a_{(j)}\} w_{n,i} x_{n,i} x_{n,i}^\top\right)^{-1} \frac{1}{t}\sum_{i=1}^{t}\sum_{j=1}^{|\mathcal{A}|} \mathbb{1}\{a_i^\star = a_i = a_{(j)}\} w_{n,i} x_{n,i} r_i \,.$$

Define $S_{t(j)}^\star \coloneqq \sum_{i=1}^{t-1} \mathbb{1}\{a_i^\star = a_{(j)}\} w_{n,i}$ for each $a_{(j)} \in \mathcal{A}$, and take, for example, the term

$$\sum_{j=1}^{|\mathcal{A}|} \frac{S_{t(j)}^\star}{t-1} \frac{1}{S_{t(j)}^\star} \sum_{i=1}^{t-1} \mathbb{1}\{a_i^\star = a_i = a_{(j)}\} w_{n,i} x_{n,i} r_i \,.$$

Since $a_t^\star \overset{\text{i.i.d.}}{\sim} \text{Unif}(\mathcal{A})$ by construction, $\frac{S_{t(j)}^\star}{t-1} \to |\mathcal{A}|^{-1}$ a.s. by the strong law of large numbers, and $S_{t(j)} \to \infty$ a.s. by the second Borel-Cantelli lemma. Furthermore, defining $S_{t(j)} \coloneqq \sum_{i=1}^{t-1} \mathbb{1}\{a_i^\star = a_i = a_{(j)}\} w_{n,i}$, $S_{t(j)}^\star - S_{t(j)} \geq 0$ is the number of '$a_{(j)}$ mistakes', and is associated with positive regret when the inequality is strict. Observe that we must have $t^{-1}(S_{t(j)}^\star - S_{t(j)}) \to 0$ in probability, as otherwise there would be $c, \epsilon > 0$ such that $\limsup \mathbb{P}(t^{-1}(S_{t(j)}^\star - S_{t(j)}) > \epsilon) > c$, implying

$$\limsup T^{-1} \mathbb{E}\left[\mathrm{R}_T^{\mathtt{2s}}\right] \geq \limsup T^{-1} \mathbb{E}\left[\mathrm{R}_T^{\mathtt{N}}\right] \geq \Delta \limsup \mathbb{E}\left[\frac{S_{T(j)}^\star - S_{T(j)}}{T}\right] > \Delta c \epsilon > 0\,,$$

which contradicts the assumption $\limsup T^{-1} \mathbb{E}\left[\mathrm{R}_T^{\mathtt{2s}}\right] \to 0$ (recall $\Delta = \min_i \overline{r}_i > 0$).

Finally, $t^{-1}(S_{t(j)}^\star - S_{t(j)}) \to 0$ implies $\frac{S_{t(j)}}{S_{t(j)}^\star} \to 1$ and $S_{t(j)} \to \infty$ in probability, and therefore

$$\frac{S_{t(j)}^\star}{t-1} \sum_{j=1}^{|\mathcal{A}|} \frac{S_{t(j)}}{S_{t(j)}^\star} \frac{1}{S_{t(j)}} \sum_{i=1}^{t-1} \mathbb{1}\{a_i^\star = a_i = a_{(j)}\} w_{n,i} x_{n,i} r_i \;\to\; \mathbb{E}_{a \sim \text{Unif}(\mathcal{A})}\left[w_{n,a} x_{n,a} r_a\right]\,,$$

in probability by the law of large numbers, the continuous mapping theorem, and $|\mathcal{A}| < \infty$. Since an analogous argument can be made for the covariance term, and $A \mapsto A^{-1}$ is continuous on the space of invertible matrices, $\widehat{\theta}_{n,t} \to \theta_n^\star$ in probability by the continuous mapping theorem, as desired.

□

4.8 Experimental details

The experiments were implemented in Python [197], using the following packages: abseil-py, h5py, HuggingFace Transformers [239], JAX [34], Jupyter [128], matplotlib [108], numpy [91], Pandas [164, 225], PyTorch [179], scikit-learn [182], scipy [230], seaborn [237], tqdm. The bandit experiments in Section 4.3 were run in an embarrassingly parallel fashion on an internal academic CPU cluster running CentOS and Python 3.8.3. The MoE experiments in Section 4.4 were run on a single desktop GPU (Nvidia GeForce GTX 1080). While each experiment took under five minutes (most under two), we evaluated hundreds of thousands of different parameter configurations (including random seeds in the count) over the course of this work. Due to internal scheduling via slurm and the parallel execution, we cannot determine the overall total CPU hours consumed for the experiments in this work.

Besides the UCB and Greedy results reported in the main text, some of the experiments we ran also included *policy gradient* (PG) where at each step t, the agent takes a single gradient step along $\nabla \mathbb{E}_x\left[\mathbb{E}_{a \sim \pi(x)}\left[r_a\right]\right] = \mathbb{E}_x\left[\mathbb{E}_{a \sim \pi(x)}\left[r_a \nabla \log \pi_a(x)\right]\right]$ where the policy is parametrised by logistic regression, i.e., $\log \pi_a(x) = \langle \theta, x_a \rangle - \log \sum_{a'} \exp\{\langle \theta, x_{a'} \rangle\}$, and the expectations are approximated with the *last* observed tuple (x_t, a_t, r_t). PG typically performs much worse than UCB and Greedy in our experiments which is most likely the result of not using a replay buffer, or any of the other standard ways of improving PG performance. We eventually decided not include the PG results in the main paper as they are not covered by the theoretical investigation in Section 4.3.

For the bandit experiments, the arm pools $\{\mathcal{A}_n\}_n$, and feature subsets $s < d$, were divided to minimize overlaps between the individual nominators. The corresponding code can be found in the methods `get_random_pools` and `get_random_features` within `run.py` of the supplied code:

- **Pool allocation:** Arms are randomly permuted and divided into N pools of size $\lfloor |\mathcal{A}|/N \rfloor$ (floor). Any remaining arms are divided one by one to the first $|\mathcal{A}| - N\lfloor |\mathcal{A}|/N \rfloor$ nominators.

- **Feature allocation:** Features are randomly permuted and divided into N sets of size $s' = \min\{s, \lfloor d/N \rfloor\}$. If $s' < s$, the $s - s'$ remaining features are chosen uniformly at random without replacement from the $d - s'$ features not already selected.

To adjust for the varying dimensionality, the regularizer λ was multiplied by the input dimension for UCB and Greedy algorithms, throughout. The λ values reported below are prior to this scaling.

Synthetic bandit experiments (Fig. 4.3)

Hyperparameter sweep: We used the single-stage setup, no misspecification ($d = s$), 100 arms, $d = 20$ features, and 0.1 reward standard deviation, to select hyperparameters from the grid in Table 4.1, based on the average regret at $T = 1000$ rounds estimated using 30 different random seeds.

Table 4.1: Hyperparameter grid for the synthetic dataset. Bold font shows the selected hyperparameters.

algorithm	parameter	values
UCB	regularizer λ	$[10^{-4}, 10^{-3}, \mathbf{10^{-2}}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$
	exploration bonus α	$[10^{-4}, 10^{-3}, \mathbf{10^{-2}}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$
Greedy	regularizer λ	$[10^{-4}, 10^{-3}, \mathbf{10^{-2}}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$
PG	learning rate	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, \mathbf{10^{0}}, 10^{1}, 10^{2}]$

With the hyperparameters fixed, we ran 30 independent experiments for each configuration of the UCB, Greedy, and PG algorithms in the single-stage case, and 'UCB+UCB', 'UCB+PG', 'UCB+Greedy', 'PG+PG', and 'Greedy+Greedy' in the two-stage one. Other settings we varied are in Table 4.2. The 'misspecification' ρ was translated into the nominator feature dimension via $s = \lfloor d/\rho \rfloor$. For the nominator count N, the configurations with $N > |\mathcal{A}|$ were not evaluated.

Table 4.2: Evaluated configurations for the synthetic dataset.

parameter	values
arm count $\|\mathcal{A}\|$	$[10, 100, 10000]$
feature count d	$[5, 10, 20, 40, 80]$
nominator count N	$[2, 5, 10, 20]$
reward std. deviation	$[0.01, 0.1, 1.0]$
misspecification ρ	$[0.2, 0.4, 0.6, 0.8, 1.0]$

Amazon bandit experiments (Fig. 4.4)

The features were standardized by computing the mean and standard deviation over *all* dimensions.

Hyperparameter sweep: We used the single-stage setup, $s = 50$ features, 100 arms, to select hyperparameters from the grid in Table 4.3, based on the average regret at $T = 5000$ rounds estimates using 30 different random seeds.

Table 4.3: Hyperparameter grid for the Amazon dataset. Bold font shows the selected hyperparameters.

algorithm	parameter	values
UCB	regularizer λ	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, \mathbf{10^{0}}, 10^{1}, 10^{2}]$
	exploration bonus α	$[10^{-4}, \mathbf{10^{-3}}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$
Greedy	regularizer λ	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, \mathbf{10^{0}}, 10^{1}, 10^{2}]$
PG	learning rate	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, \mathbf{10^{1}}, 10^{2}]$

With the hyperparameters fixed, we again ran 30 independent experiments for the same set of algorithms as in Section 4.8, but now with fixed $d = 400$ as described in Section 4.3. Since d is fixed, we vary the nominator feature dimension s directly. Other variables are described in Table 4.4.

Table 4.4: Evaluated configurations for the Amazon dataset.

parameter	values
arm count $\|\mathcal{A}\|$	$[10, 100, 1000]$
nominator count N	$[2, 5, 10, 20]$
nominator feature dim s	$[5, 10, 20, 40, 80, 150, 250, 400]$

Mixture-of-Experts offline experiments (Section 4.4)

Hyperparameter sweep: We ran a separate sweep for the MoE and the random pool models using 100 arms, $N = 10$ experts, and $c_k = 500$ training examples per arm. We swept over optimizer type (`'RMSProp'` [95], `'Adam'` [126]), learning rate ($[0.001, 0.01]$), and likelihood variance σ^2 ($[0.01, 1.0]$). The selection was made based on average performance over three distinct random seeds. The 0.01 learning rate was best for both models. `'RMSProp'` and $\sigma^2 = 1$ were the best for the random pool model, whereas `'Adam'` and $\sigma^2 = 0.01$ worked better for the MoE, except for the embedding dimension $d_e = 50$ where $\sigma^2 = 1$ had to be used to prevent massive overfitting.

Evaluation: We varied the number of training examples per arm $c_k \in \{100, 500, 750\}$, number of dimensions of the BERT embedding revealed to the nominators $s \in \{10, 25, 50, 100\}$, the dimension of the learned item embeddings $d_e \in \{5, 10, 50\}$, and the number of experts $N \in \{5, 10, 20\}$. We used 50000 optimization steps, batch size of 4096 to adjust for the scarcity of positive labels, and no early stopping. Three random seeds were used to estimate the reported mean and standard errors.

4.9 Additional results

Synthetic bandit experiments (Fig. 4.3)

The interpretation of all axes and the legend is analogous to that in Fig. 4.3, except the relative regret (divided by that of the uniformly guessing agent) is reported as in Fig. 4.4.

Amazon bandit experiments (Fig. 4.4)

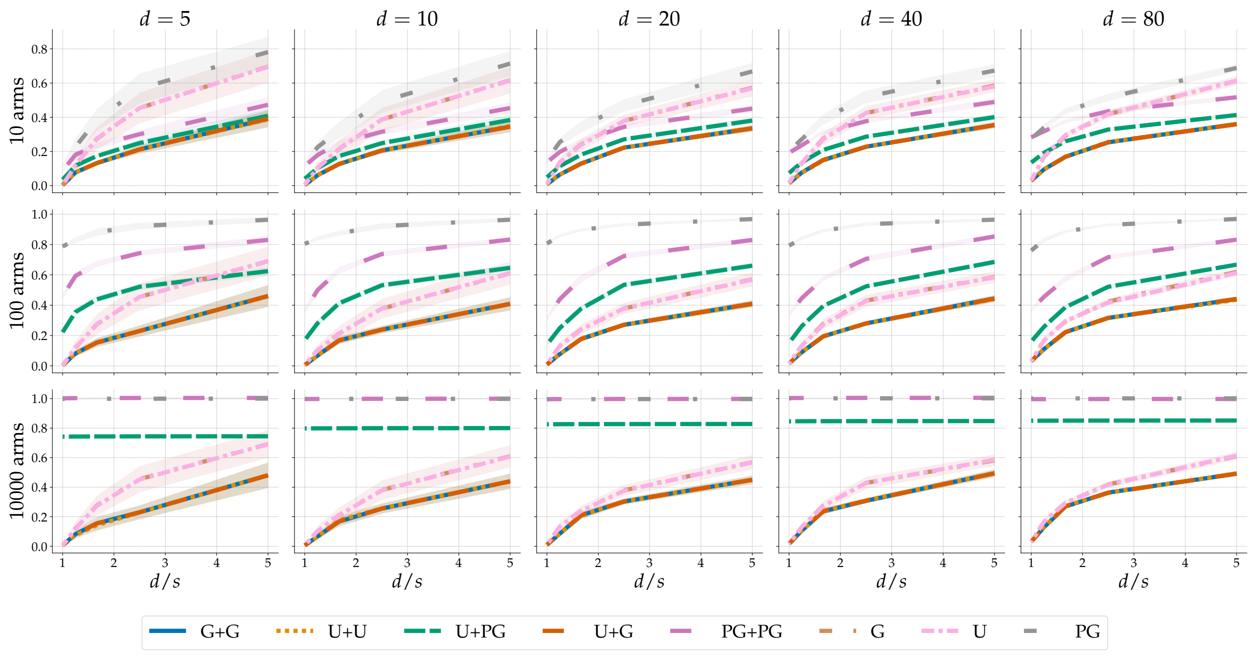


Figure 4.6: Relative regret for *two* nominators on the synthetic dataset.

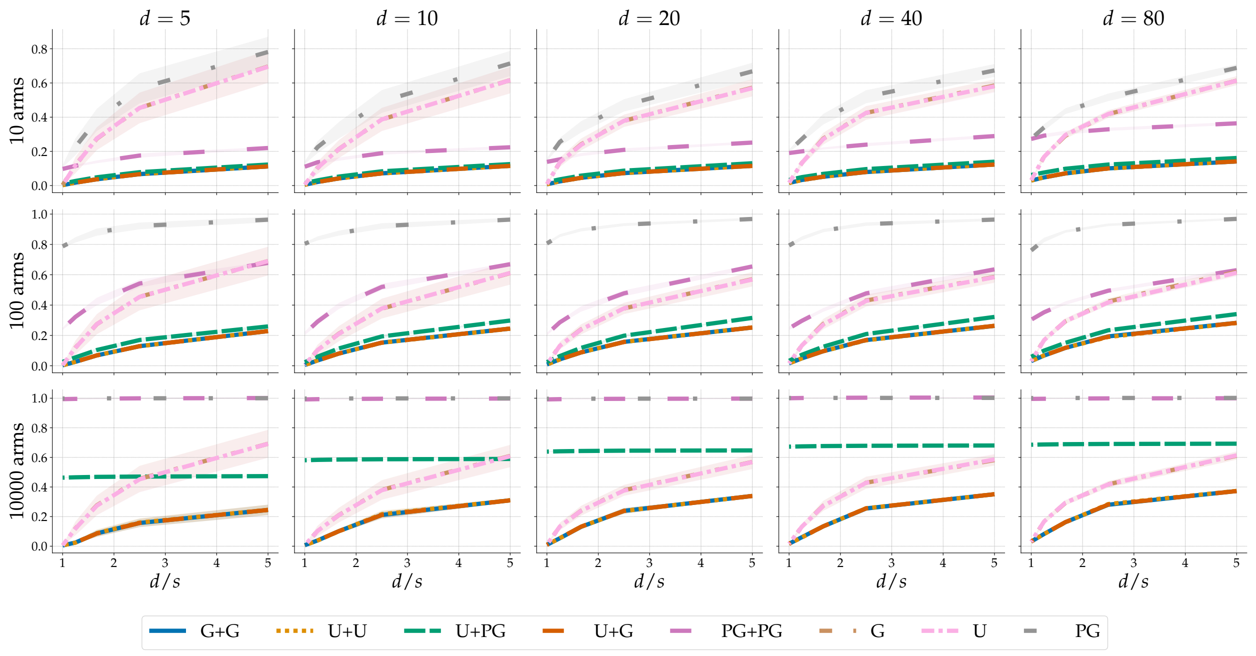


Figure 4.7: Relative regret for *five* nominators on the synthetic dataset.

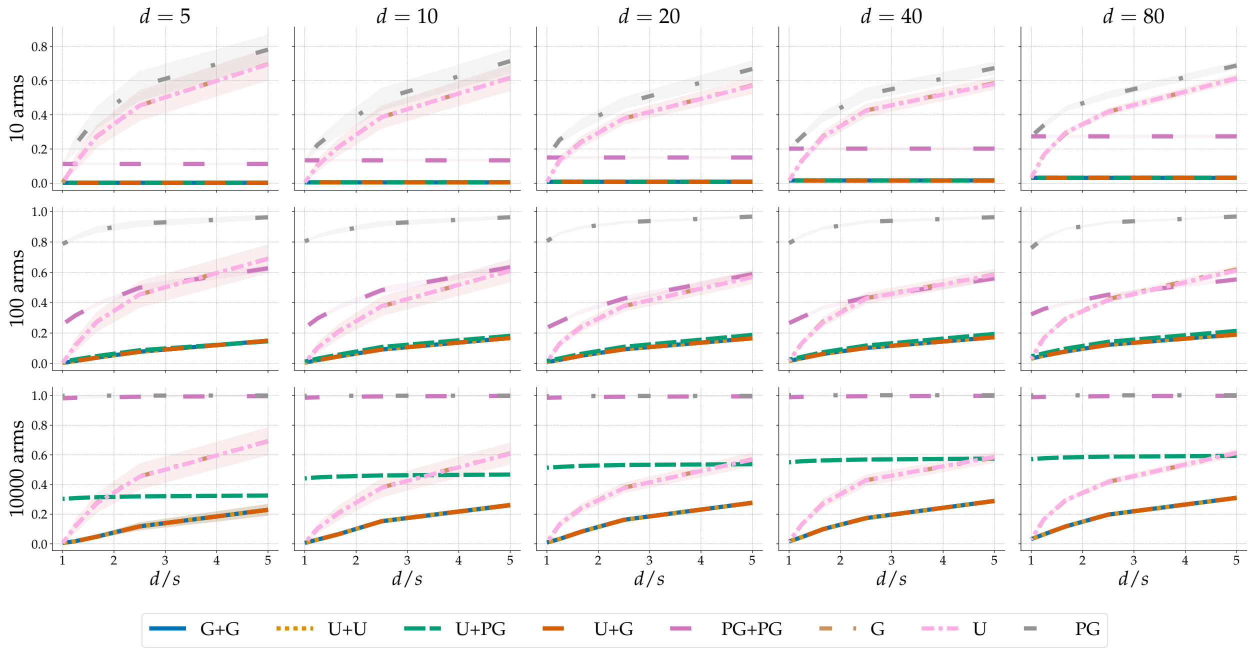


Figure 4.8: Relative regret for *ten* nominators on the synthetic dataset.

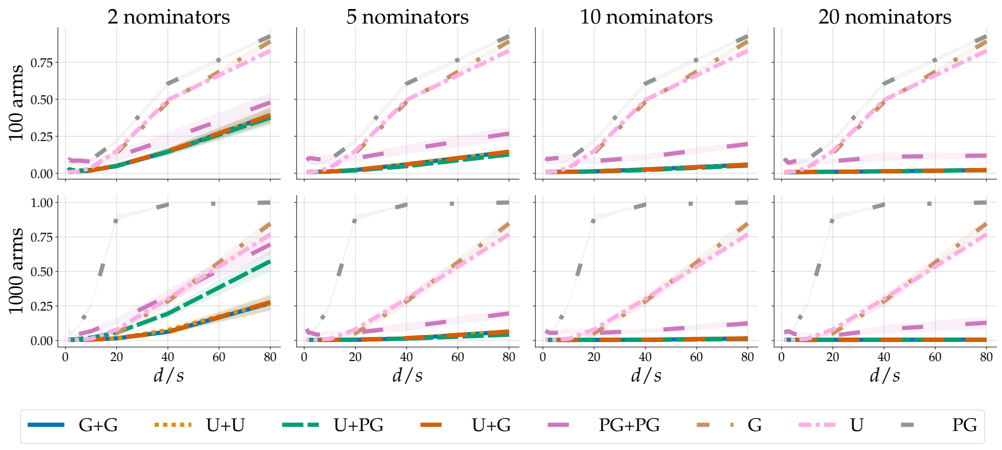


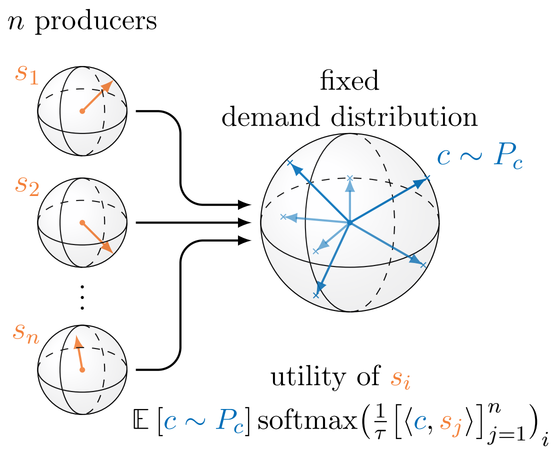
Figure 4.9: Relative regret on the Amazon dataset.

Chapter 5

Content Creator Incentives Induced by Recommender Systems

5.1 Introduction

Having studied recommendation dynamics with a focus on users that consume content, we now turn to understand how recommender systems impact users that produce content. In 2018, Jonah Peretti (CEO, Buzzfeed) noticed that junk and divisive content was increasingly elevated following a change to the Facebook main feed algorithm [87]. In Poland, politicians increased negative messaging following the same update [87]. Adapting content in response to an algorithm is not limited to social media. An example can be found in search engine optimization (SEO), where some professionals specialize on managing impacts of specific Google Search updates [57, 82, 159, 180, 207]. While the motivation for adapting content to an algorithm may range from economic to socio-political, it often translates into the same


n producers
s_1
s_2
s_n
fixed demand distribution
$c \sim P_c$
utility of s_i
$\mathbb{E}\,[c \sim P_c]\,\text{softmax}\big(\frac{1}{\tau}\,[\langle c, s_j \rangle]_{j=1}^{n}\big)_i$

R7cm

Figure 5.1: Exposure game. Items $s_i \in S^{d-1}$ placed to maximize exposure to consumers $c \sim P_c$.

operative goal: *exposure maximization*.

We therefore study how different algorithmic choices affect the incentives of exposure-maximizing content producers. We propose an incentive-based behavior model called an *exposure game*, where producers compete for a *finite* user attention pool by crafting content ranked highly by a given algorithm (Section 5.1). When producers act strategically, a steady state—Nash equilibrium (NE)—may be reached, with no one able to unilaterally improve their exposure (utility). The content produced in a NE can thus be interpreted as what the algorithm implicitly incentivizes.

We focus on algorithms which model user preferences as an inner product of d-dimensional user and item embeddings, and rank items by the estimated preference. Section 5.2 presents theoretical results on the NE induced by these algorithms. We identify cases where algorithmic changes seemingly unconnected to producer incentives—e.g., switching from non-negative to unconstrained embeddings—determine whether there are zero, one, or multiple NE. The character of NE is also affected by the level of algorithmic exploration. Perhaps counter-intuitively, we show that high levels of exploration incentivize broadly appealing content, whereas low levels lead to specialization.

In Section 5.3, we explore how a creator behavior model can facilitate a pre-deployment audit. Such an audit could be particularly useful for assessing the producer impact of algorithmic changes, which is hard to measure by A/B testing for two important reasons: (1) producers cannot be easily randomized to distinct treatment groups, and (2) there is often a delay between deployment and content adaptation. Our hope is that this style of auditing will enable detection of misalignment between the induced and desired incentives, and thus flag issues to either immediately address, or monitor in content filtering and moderation. For demonstration, we execute a pre-deployment audit on the MovieLens and LastFM datasets using the exposure game model. We find the incentivized content exhibits a strong dependence between algorithmic exploration and content diversity (confirming our theory), and between model expressivity and bias towards gender-based user and creator groups.

Setting and the exposure game incentive model

We assume there is a *fixed* recommender system trained on past data, and a *fixed* population of users (*consumers*). Together, these induce a *demand distribution* P_c which represents typical traffic on the platform over a predefined period of time. Content is created by $n \in \mathbb{N}$ *producers* who try to maximize their expected exposure (*utility*). Denoting consumers by $c \sim P_c$, an item created by the i^{th} producer by s_i (*strategy*), $s := (s_i)_{i \in [n]}$, and $s_{\setminus i} := (s_j)_{j \neq i}$, we define (expected) *exposure* as the proportion of the "user attention pool" captured by the i^{th} producer

$$u_i(s) = u_i(s_i, s_{\setminus i}) := \mathbb{E}_{c \sim P_c}\left[\mathbb{1}\{c \text{ is exposed to } s_i\}\right] \overset{\star}{=} \mathbb{E}_{c \sim P_c}\left[p_i(c)\right] ,$$

with $p_i(c) \geq 0$ the probability that the algorithm exposes c to s_i rather than any $s_{\setminus i}$. As common in game theory, we can extend from deterministic single item strategies to random strategies $s_i \sim P_i$ for some distribution P_i. The interpretation of this extension is discussed in detail in Section 5.2.

The assumption that $\mathbb{E}\left[\mathbb{1}\{c \text{ is exposed to } s_i\}\right] \overset{\star}{=} \mathbb{E}\left[p_i(c)\right]$ ignores cases where some interactions are not mediated by the algorithm (e.g., YouTube videos linked to by an external website). This may be a reasonable approximation for infinite feed platforms (e.g., Twitter, Facebook, TikTok) where most consumers scroll through items in the algorithm-defined order, and search engines (e.g., Google, Bing) where first-page bias is well documented [49]. While similar assumptions are common in the literature [e.g., 19, 45, 50, 146], alternative interaction models are an important future research direction.

Unlike previous work (Section 5.1), we focus on the popular class of factorization-based algorithms. These models rank items by a score estimated by the inner product of user and item embeddings $c, s_i \in \mathbb{R}d$. The larger this score, the higher the probability of exposure, which we model as

$$p_i(c) = \frac{\exp(\tau^{-1}\langle c, s_i\rangle)}{\sum_{i'=1}^{n} \exp(\tau^{-1}\langle c, s_{i'}\rangle)} = \operatorname{softmax}\big(\left[\tau^{-1}\langle c, s_{i'}\rangle\right]_{i'=1}^{n}\big)_i \,,$$

where $\tau \geq 0$ is a temperature parameter which controls the spread of exposure probabilities over the top scoring items. When $\tau = 0$ (i.e., hardmax), these probabilities correspond to top-1 recommendation or absolute first-position bias. Taking $\tau > 0$ models the effects of ranked position, injected randomness for exploration, and even interactions not mediated by the algorithm. While an approximation in some settings, Section 5.1 has been directly used, e.g., by YouTube [45]. We emphasize that we make no assumption on how the embeddings are obtained, and our conclusions thus apply equally to classical matrix factorization and deep learning-based systems.

We are now ready to formalize *exposure games*, an incentive-based model of creator behavior.

Definition 1. *An* exposure game *consists of an embedding dimension $d \in \mathbb{N}$, a demand distribution $P_c \in \mathcal{P}(\mathbb{R}d)$, and $n \in \mathbb{N}$ producers (players), each with an associated (pure) strategy space $s_i \in S^{d-1}$ and utility function $u_i(s) = \mathbb{E}_{c\sim P_c}\left[p_i(c)\right]$ with $p_i(c)$ as in Section 5.1 for a given $\tau \geq 0$.*

We restrict items s_i to the unit sphere $S^{d-1} = \{v \in \mathbb{R}d \colon \|v\| = 1\}$. A norm constraint is necessary as otherwise the rational behavior would often be to infinitely inflate $\|s_i\|$, which is not observed in practice.[1] We further distinguish *non-negative* games where all embeddings lie in the positive orthant. This includes algorithms ranging from TF-IDF and bag-of-words, to non-negative matrix factorization [142], topic models [29], and constrained neural networks [11].

Definition 2. *A* non-negative exposure game *is an exposure game where the support of P_c is restricted to the positive orthant, i.e., $P_c(\{c \in \mathbb{R}d \colon c_j \geq 0\,, \forall j \in [d]\}) = 1$.*

[1]Possibly due to the often finite rating scale, and use of gradient clipping and various forms of regularization.

We assume all producers are rational, have complete information, and full control over placing s_i in S^{d-1}. Full control is perhaps the least realistic, since producers can only modify their features ingested by the algorithm in practice. This assumption has a significant advantage though: it abstracts away an explicit model of producer actions (cf. the variety of SEO techniques). Appropriateness of rationality and complete information are then context-dependent; they may be respectively reasonable in environments where strong profit motives or user profiling tools are common. However, investigating alternatives to each of the above assumptions is an important direction of future work.

Related work

Most relevant to our theoretical results are several works studying incentives of exposure-maximizing creators induced by recommender and information retrieval systems [17, 18, 19, 20, 21, 185]. Interesting features which we do not consider include (i) repeated interaction with the algorithm [19, 20, 185], (ii) user welfare [17, 18, 19, 21], and (iii) incomplete information scenarios [185].

The most important distinction of our approach is that the above works only allow production of items from a predefined finite catalog. This excludes the popular *factorization-based* algorithms—ranging from standard matrix factorization [132] to (deep) two-tower architectures [106, 243]—whose continuous embedding space translates into an *infinite* number of possible items. The only exception is [18] where users and items are represented by numbers in $[0, 1]$, which is equivalent to the special case of two-dimensional non-negative exposure games. Continuous embedding spaces were recently studied in [169, 245], though neither focuses on producer competition. Mladenov et al. [169] consider producers who decide whether to stay or leave the platform if their exposure is too low. Zhan et al. [245] study design of recommender systems which optimize for *both* user and producer utility.

Our work is also related to literature on adaptive behavior in the presence of a decision-making algorithm [88, 127, 183]. The social impact and potential disparate effects of strategic adaptation have been analyzed in [104, 151, 168]. Most relevant for us is a recent paper by Liu, Garg, and Borgs [150] which studies strategic adaptation in the context of *finite* resources (e.g., number of accepted college applicants). Unlike us, the authors assume a *single* score for each competitor, who can pay *cost* to improve it. A principal then *designs* a reward function which allocates the finite resource based on the scores, and the authors study how different choices affect various notions of welfare. The preliminary results on multi-dimensional scores (appendix B) assume that the scores and individual improvements are *independent*, whereas our scores—$\langle c, s_i \rangle$ for each c—imply complex dependence and trade-offs.

Finally, our proposed methods for auditing recommender and information retrieval systems belong to a rapidly growing algorithm auditing toolbox. We focus on understanding producer incentives caused by a known algorithm. Thus, we complement prior work that aims to audit these systems based upon: the degree of consumer control [50], fairness [59], compliance with regulations [39], and dynamical behavior in simulations [134, 153] or deployed systems [89].

5.2 Equilibria in exposure games

This section presents theoretical results on incentives in exposure games. We focus on the impact of the recommender/information retrieval model on the competitive equilibria. Throughout, we find that one of the most important factors determining existence and character of equilibria is the temperature τ (see Section 5.1). We thus distinguish the *softmax* ($\tau > 0$) and the *hardmax* ($\tau = 0$) case.

In competitive settings, a key question is whether there are equilibria in which players are satisfied with their strategies, as otherwise there may be never-ending oscillation in search for better outcomes. We thus consider several *solution concepts* (i.e., definitions of equilibria) related to NE. A *pure NE* (PNE) is a point in strategy space $s^{\mathsf{NE}} \in (S^{d-1})^n$ where no player i can increase their utility by unilaterally deviating from $s_i^{\mathsf{NE}} \in S^{d-1}$. In other words, no content producer can increase their exposure by modifying their content. *Mixed NE* (MNE) refer to the setting where players are allowed to choose randomized (*mixed*) strategies $P_i \in \mathcal{P}(S^{d-1})$. Rather than selecting a single piece of content, a creator following a mixed strategy samples $s_i \sim P_i$. Alternative interpretation is that producers create multiple items, splitting their time/budget proportionally to the P_i-probabilities.

In later sections, we explore the weaker solution concepts of ϵ-NE, *local NE* (LNE), and their combination ϵ-LNE. An ϵ-NE is an approximate NE where no producer can unilaterally increase their utility by more than ϵ (MNE/PNE are "0-NE"). LNE are analogous to local optima: points where no player benefits from small deviations from their strategy. The approximate and local perspectives are relevant when deploying local search algorithms to find NE numerically (Section 5.3).

Exposure games are symmetric, meaning that any permutation of strategies forming an equilibrium produces another equilibrium. Our statements on the existence and uniqueness of equilibria hold up to player permutation. All proofs for the results in this section are presented in the appendix.

Pure and mixed Nash equilibria

We begin by characterizing the existence of pure and mixed NE in general exposure games.

Theorem 2. *Every exposure game has at least one* mixed *Nash equilibrium.*

A key property of *softmax* games is that the utilities u_i are continuous in s. This fact, combined with the compactness of the strategy space S^{d-1}, guarantees existence of MNE via a classic result by Glicksberg [81]. In the *hardmax* case ($\tau = 0$), we can show that MNE are guaranteed to exist through a direct application of proposition 4 due to Simon [212]. The producer utilities u_i are not differentiable in the hardmax case though, which means we cannot use gradient information to find NE as in the softmax case. The only procedure we know for finding NE in hardmax games requires solving the hitting set problem which is NP-complete [54]. See Section 5.6 for further discussion.

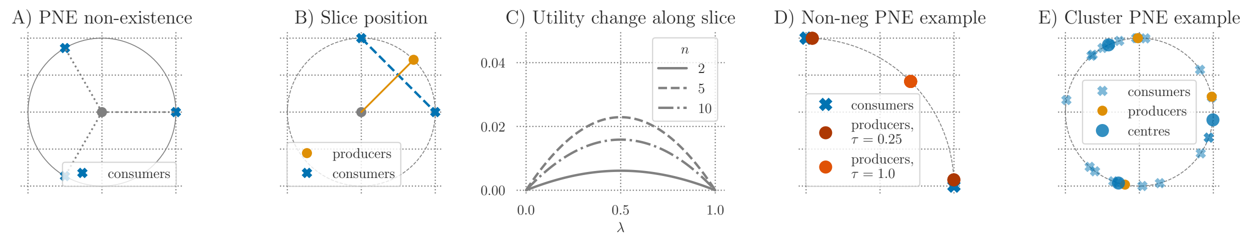


Figure 5.2: A) A game with no PNE. **B)** $n - 1$ producers at midpoint, s_1 along slice $\lambda c_1 + (1 - \lambda) c_2$ (dashed line). **C)** Change in utility along the slice in B) demonstrates lack of quasi-concavity. **D)** A non-negative game with very different PNE depending on τ. **E)** PNE with "protective positioning."

We now turn to existence of *pure* NE, which is the setting where creators strategically design a *single* piece of content. Unlike MNE, PNE are not guaranteed to exist even in the softmax case.

Theorem 3. *PNE need not exist in either the hardmax ($\tau = 0$) or softmax ($\tau > 0$) exposure games.*

Fig. 5.2A illustrates this non-existence result. The counter-example holds even for $n = 2$ players and planar ($d = 2$) strategies. A reader familiar with classic PNE existence results [56, 66, 81] may wonder if PNE would appear if we relaxed the strategy space from S^{d-1} to the *convex* $B^d = \{v : \|v\| \leq 1\}$. This is not the case because the exposure utility is not quasi-concave (Fig. 5.2B and C).

Our next point of interest is the special case of *non-negative* exposure games (Definition 2). For the $n = d = 2$ case, non-negative hardmax exposure games are equivalent to Hotelling games [100], and more generally to facility location games on a line [18, 184]. The following proposition lists several special cases in which we understand existence and character of PNE.

Proposition 3. *A PNE always exists in $n = d = 2$ non-negative* hardmax *games, but may not without non-negativity or when $d > 2$. For $n = 2$ non-negative* softmax *games with $\widehat{c} := \frac{1}{n}(1 - \frac{1}{n})\mathbb{E}[c] \neq 0$, the only possible PNE is $s_1 = s_2 = \overline{c}$ with $\overline{c} := \frac{\widehat{c}}{\|\widehat{c}\|}$ (independently of d), but a PNE may not exist. When $n > 2$, non-negative softmax games can have a PNE other than $s_1 = \cdots = s_n = \overline{c}$.*

Fig. 5.2D illustrates a 4-player non-negative exposure game. Depending on the temperature, we observe either the collapsed $s_i = \overline{c}$ (large τ), or what we term "protective positioning" (small τ). In Fig. 5.2D, players place their strategies *between* a consumer and the next closest producer. Fig. 5.2E illustrates protective positioning for a higher number of consumers and $n = 3$. Here, consumers are roughly clustered around three centers (blue dots). The producer strategies are close to these centers, but again offset towards the most contested consumers.

ϵ-Nash equilibria

While existence of NE is not guaranteed, the situation changes when we adopt the weaker solution concept of ϵ-NE, in which no producer can unilaterally increase their utility by more than ϵ.

The existence and character of such equilibria strongly depends on the temperature parameter τ. When $\tau = \infty$, exposure is equally likely $p_i(c) = \frac{1}{n}$ for all i and c regardless of the adopted strategies. Thus, every strategy profile is an NE. Considering a sequence of increasing $(\tau_i)_{i \geq 1}$, we can therefore argue that the limit of any convergent sequence of NE indexed by τ is a NE at $\tau = \infty$. Interestingly, Theorem 4 shows that a sufficiently large but *finite* $\tau > 0$ is sufficient for existence of ϵ-(P)NE. The result is constructive, showing that the ϵ-PNE is parallel to the average consumer embedding.

Theorem 4. *For any $\epsilon > 0$ and $P_c \in \mathcal{P}(\mathbb{R}d)$ with compact support and $\mathbb{E}[c] \neq 0$, $\exists \tau_0 > 0$ s.t. $s_1 = \ldots = s_n = \overline{c}$ is an ϵ-PNE for all $\tau \geq \tau_0$. Moreover, for all $\tau \geq \tau_0$, the smallest ϵ_τ for which $\overline{c}$ is an ϵ_τ-PNE satisfies $\epsilon_\tau \leq \frac{\epsilon}{\tau}$. If also $\epsilon < \|\widehat{c}\|$, then the set of better-responses to $\overline{c}$*

$$\Psi(\overline{c}) \coloneqq \{v \in S^{d-1} \colon u_1(v, \overline{c}, \ldots, \overline{c}) \geq u_1(\overline{c}, \overline{c}, \ldots, \overline{c})\},$$

is a subset of $B^d_\delta(\overline{c}) = \{v \colon \|v - \overline{c}\| \leq \delta\}$ with $\delta = 2\epsilon/(\|\widehat{c}\| - \epsilon)$, and $\delta \to 0$ as $\tau \to \infty$.

This result shows that all ϵ-improvements concentrate near the consumer average ϵ-PNE as $\tau \to \infty$. Additionally, the "consumer symmetry" $\|\widehat{c}\| = \frac{1}{n}(1 - \frac{1}{n}) \|\mathbb{E}[c]\|$ determines how quickly $\delta \to 0$. When consumers are spread approximately symmetrically w.r.t. the origin, the degenerate equilibrium appears only for large τ. However, smaller τ are sufficient for more directionally concentrated P_c. A high number of producers also slows the concentration as the appeal of $u_i(\overline{c}, \ldots, \overline{c}) = \frac{1}{n}$ decreases with n. We conclude by stating a corollary based on the development in the last two sections.

Corollary 5. *There is a fixed $\epsilon_0 > 0$ and a demand distribution P_c which—depending on the chosen τ—induce zero, one, multiple, or infinitely many ϵ-NE for all $\epsilon \leq \epsilon_0$.*

Corollary 5 underscores the sensitivity of exposure games to the temperature parameter τ, with uniformly homogeneous content at one end (high τ), and potentially persistent oscillation behavior in competition when no NE exist (low τ). A higher $\tau > 0$ can be a result of *algorithmic exploration* [40, 45, 139], which is provably necessary for optimal performance in *static* environments [139]. In contrast, our results show that in environments with *strategic* actors, exploration may incentivize content which is uniform and broadly appealing rather than diverse.

This may contradict the intuition that more exploration should lead to greater content diversity due to the higher exposure of niche content. One way to understand this result is the tension between randomization and the ability of niche creators to reach their audience: producers may be discouraged from creating niche content when the algorithm is exploring too much (τ high), and encouraged to mercilessly seek and protect their own niche when the algorithm performs little exploration (τ low). Exploration effects are typically thought of as having negative impact on user experience through immediate reduction in quality of service as a result of suboptimal recommendations. However, the above results show secondary long-term effects.

Local Nash equilibria

In a local NE, no producer can benefit from changing s_i in some neighborhood within the embedding space. Sometimes motivated as a form of bounded rationality, LNE can often be found by local search algorithms [e.g., 162]. Since our motivation in studying exposure games is ultimately better system understanding and audits, we are particularly interested in these algorithmic benefits.

Practical first-order algorithms for identifying LNE operate analogously to gradient descent, implying they may terminate in *critical points* that are not LNE. Unlike NE, critical points always exist.

Proposition 4. *Every $\tau > 0$ exposure game with $\mathbb{E}[c] \neq 0$ has a critical point at $s_1 = \ldots = s_n = \bar{c}$.*

As we have seen, $s_1 = \ldots = s_n = \bar{c}$ may be an equilibrium (Proposition 3). To distinguish LNE from mere critical points, we use the Riemannian version of the second derivative test [32].

Definition 3. *A point s in strategy space satisfies the* second derivative test *if $\forall i \in [n]$ (1) the* Riemannian gradient $(I - s_i s_i^\top)\nabla_{s_i} u_i(s)$ *are zero, and (2) the* Riemannian Hessian

$$(I - s_i s_i^\top)\left[\nabla_{s_i}^2 u_i(s)\right](I - s_i s_i^\top) - \langle s_i, \nabla_{s_i} u_i(s)\rangle (I - s_i s_i^\top)\,,$$

is strictly negative definite in the subspace perpendicular to s_i.

This condition is sufficient but not necessary for a critical point to be an LNE. The LNE that do satisfy Definition 3 are termed *differentiable* Nash equilibria [12, 188]. The distinction can be understood as analogous to the flat minimum at zero of x^4 compared with the more well-behaved x^2.

5.3 Pre-deployment audit of strategic creator incentives

Beyond regularly retraining on new data, online platforms continuously roll out algorithm updates. While A/B testing can detect changes in user metrics, like satisfaction or churn, prior to the full-scale deployment [85, 97, 224, 240], assessing the impact on content producers is comparatively harder due to the longer delay between the roll-out and corresponding content adaptation. Furthermore, since producers cannot be easily assigned to distinct treatment groups without limiting their content to only a subset of consumers, modern A/B testing methods must eschew making causal statements about producer impact [109, 174, 227]. Undesirable results including promulgation of junk and abusive content then have to be addressed via *post-hoc* measures like content filtration and moderation.

A tool for *ex-ante* (pre-deployment) assessment of producer impact could thus limit the harm experienced by users, moderators, and other affected parties. We demonstrate how to utilize a creator behavior model for this purpose, using the exposure game as a concrete example. The incorporation of factorization-based algorithms in exposure games allows us to use real-world datasets and rating models. While exposure games have limitations as a behavior model, we believe our experiments provide a useful illustration of the insights the proposed audit can offer to platform developers.

Setup

We use the `MovieLens-100K` and `LastFM-360K` datasets [24, 90, 208], implement our code in Python [197] and rely on `numpy` [91], `scikit-surprise` [107], `pandas` [225], `matplotlib` [108], `jupyter` [128], `reclab` [134], and `JAX` [34] packages to fit probabilistic [PMF; 170] and non-negative [NMF; 142] matrix factorization. The models are trained to predict the user ratings (centered in the PMF case). To select regularization and learning rate, we performed a two-fold 90/10 split cross-validation separately on each dataset. The tuned recommenders were then fit on the full dataset, and the resulting user embeddings, $\{c_j\}_{j \in [m]} \subset \mathbb{R}d$, were used to construct the demand distribution $P_c = \frac{1}{m} \sum_j \delta_{c_j}$, and evaluate the recommendation probabilities $p_i(c)$. Details in Section 5.7.

The only algorithm we know for finding NE in hardmax exposure games has exponential worst-case complexity. We therefore focus on the softmax case. While search for general *mixed* NE is possible in special cases [22, 75, 122], we are not aware of any technique applicable to n-player exposure games. We therefore focus on first-order methods and *pure* ϵ-LNE (Section 5.2). We employ simple *gradient ascent combined with reparametrization*, where we set $s_i = \theta_i / \|\theta_i\|$ for each producer, and iteratively update $\theta_{i,t} = \theta_{i,t-1} + \alpha \nabla_{\theta_{i,t-1}} u_i(s_{i,t-1}, s_{\setminus i,t-1})$ for shared step size $\alpha > 0$, and

$$\nabla_{\theta_i} u_i(s) = \tfrac{1}{\tau \|\theta_i\|_2} (I - s_i s_i^\top) \mathbb{E}\left[p_i(c)(1 - p_i(c)) c\right] = \tfrac{1}{\|\theta_i\|_2} (I - s_i s_i^\top) \nabla_{s_i} u_i(s) \,.$$

Section 5.3 shows the update direction is proportional to the Riemannian gradient of $u_i(s)$ w.r.t. $s_i \in S^{d-1}$ (Section 5.2). We also experimented with the related Riemannian gradient ascent optimizer [32], but abandoned it after (predictably) observing little qualitative difference. We note that the local updates themselves define *better-response dynamics* linked to iterative minor content changes; investigation of their relation to real-world producer behavior is an interesting future direction.

We investigate dependence of the incentivized content on the following parameters: (i) *rating model* $\in$ {PMF, NMF}, (ii) *embedding dimension* $d \in \{3, 50\}$, and (iii) *temperature* $\log_{10} \tau \in \{-2, -1, 0\}$. We further vary the number of producers $n \in \{10, 100\}$ to examine scenarios with different producer to consumer ratios (user count is fixed to the full 943 for

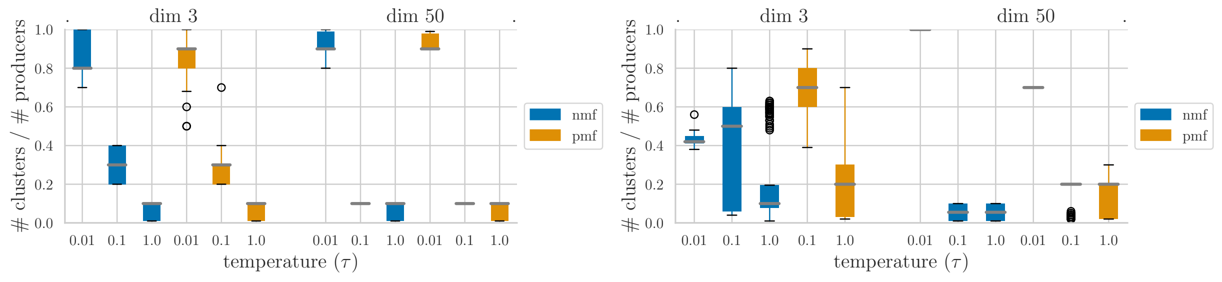


Figure 5.3: Clustering of strategic producers depends on the exploration level τ. As Theorem 4 predicts, large τ (e.g., more exploration) leads to higher concentration, i.e., creating content which appeals to *more* users. **Left:** MovieLens. **Right:** LastFM. See Section 5.3 for more discussion.

MovieLens, and 13,698 for LastFM). The above values were selected in a preliminary sweep as representative of the effects presented below. For every setting, we used five random seeds for initialization of the recommender (affects P_c), and for each ran the gradient ascent algorithm 10x to identify possible ϵ-LNE. We applied early stopping when ℓ^2-change in parameters between iterations dipped below $10^{-8} \cdot \sqrt{d}$; the number of iterations was set to 50K so convergence was achieved for *every* run. We only report runs where the second-order Riemannian test from Section 5.2 did not rule out an ϵ-LNE. Additional results, including those where the Riemannian test was conclusive, are in Section 5.7.

Results

Emergence of clusters with growing τ. Theorem 4 shows that producers concentrate around $\bar{c} = \mathbb{E}[c] / \|\mathbb{E}[c]\|$ for sufficiently high temperature τ. Fig. 5.3 corroborates the result on both MovieLens and LastFM, with the concentration happening already at $\tau = 1.0$ regardless of the embedding dimension d and producer count n. We also see that lower τ can lead to "local clustering" where only few producers converge onto the same strategy. We hypothesize that the *simultaneous* local updates of the consumers create "attractor zones" where close-by producers collapse onto each other; they will remain collapsed henceforth due to equality of their gradients (by symmetry). Theorem 4 does tell us collapse is to be expected for high τ, and it is possible that a local version of the result with more than one clusters is true for intermediate values of τ. This highlights how crucial the *algorithmic choice* of τ is for the induced incentives within our model.

Targeting of incentivized content by gender. The MovieLens dataset contains binarized women/men user gender information. In Fig. 5.4, we examine targeting of incentivized content on women and men. To do so, we employ aggregate statistics of *predicted* ratings. While predicted ratings may differ from actual user preferences, they do determine recommendations and thus *user experience*. Since effect of τ on rating models varies, we

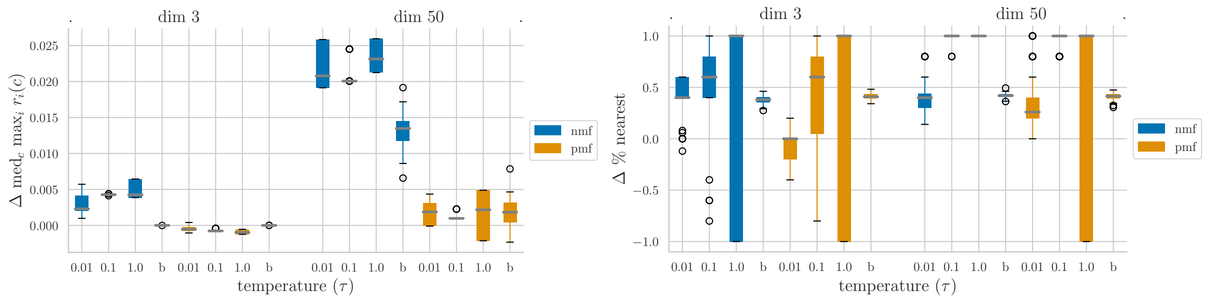


Figure 5.4: Targeting of incentivized content by gender on MovieLens. **Left:** Difference between $\text{median}_{c\in G}\{\max_{i\in[n]} \overline{r}_i(c)\}$ for men and women (group G), with $\overline{r}_i(c)$ the normalized rating (cosine similarity between c and the strategic s_i). Positive values imply bias towards men (higher median). Note the higher bias when $d = 50$ (more expressive algorithm); especially NMF incentivizes more biased content relative to the pre-adaptation baseline `'b'`. **Right:** Difference in proportions of s_i with best (normalized) rating by women/men. Positive imply bias towards men (more items best-rated by men). Bias again more pronounced at $d = 50$. See Section 5.3 for more discussion.

also include *baseline* values (labeled by `'b'`) computed using the original learned item embeddings (i.e., item locations before strategic adaptation). Since the baseline embeddings need not satisfy the unit norm constraint (see Definition 1), we measure *normalized* ratings $\overline{r}_i(c) := \frac{\langle c, s_i\rangle}{\|c\|\|s_i\|}$ to facilitate comparison. The normalization also alleviates the known issue of varying interpretation of ranking scales between users [154].

Both plots in Fig. 5.4 estimate a difference of group statistics $\Delta = \phi_{\text{men}} - \phi_{\text{women}}$, where $\phi_G = \text{median}_{c\in G}\{\max_{i\in[n]} \overline{r}_i(c)\}$ (left), and $\phi_G = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{\arg\max_c \overline{r}_i(c) \in G\}$ (right). The former is a user-centric metric measuring whether preferences of either group are more targeted by the new content. The latter statistic is producer-centric, measuring the proportion of producers who create content expected to be most liked by either women or men. In both cases, higher values signify content crafted towards male audience (users skew 71% to 29% male). Notably, higher embedding dimension results in higher bias, presumably due to the higher model expressivity, and thus option to create more targeted content. Interestingly, NMF consistently incentivizes more biased content.

Association between incentivized content and creator gender. Platform developers may want to know if some creators are being disadvantaged [194]. While solutions were proposed in the static case [e.g, 25, 232], understanding if the algorithm (de)incentivizes content by particular creator groups may limit future harm. In Fig. 5.5, we measure the difference between the *proportion of* (left) and the *median distance to* (right) *baseline* creator embeddings (learned by the recommender before strategic adaptation), within increasingly large neighborhoods of each strategic s_i. Since the baseline embeddings need not be unit norm, we use the cosine distance to define the neighborhoods.

Starting with the proportion (left), higher embedding dimension (more flexible model)

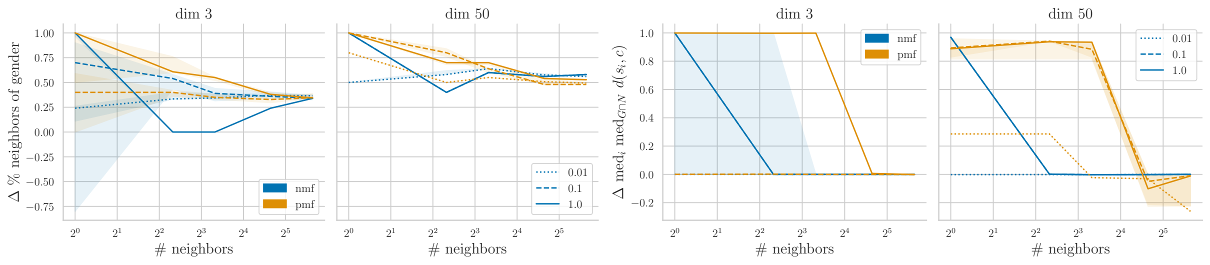


Figure 5.5: Incentivized content and creator gender on LastFM. Quantifying relative difficulty of strategic adaptation for female and male content creators, Uses baseline creator embeddings (and associated gender), and their cosine distance from strategic embeddings. **Left:** Difference between fractions of male and female creators in increasingly large neighborhood of each strategic item. Values above zero imply bias towards male producers. Higher embedding dimension (model expressivity) again results in larger bias. The bias also seems to be larger for higher τ and for the PMF rating model. **Right:** Difference between median cosine distance to female and male creators within increasingly large neighborhood of each strategic item. Values above zero imply bias towards male producers. Higher bias is again associated with higher embedding dimension and the PMF rating model, but the impact of temperature τ is less pronounced. See Section 5.3 for more discussion.

incentivizes content more typical of male artists. This may be related to the higher prevalence of men in LastFM, combined with training by average loss minimization. The gender imbalance also explains why the proportion (left) stabilizes at a positive value, whereas the median distance (right) reverts to zero, as the number of considered neighbors grows. The bias is also related to the choice of rating model, where especially PMF at high temperatures results in significant advantage for male artists.

5.4 Discussion

From social media and streaming to Google Search, many of us interact with recommender and information retrieval systems every day. While the core algorithms have been developed and analyzed years ago, the socio-economic context in which they operate received comparatively little attention in the academic literature. We make two main contributions: (a) we define *exposure games*, an incentive-based model of content creators' interactions with real-world algorithms including the popular matrix factorization and two-tower systems, and (b) we formulate a *a pre-deployment audit* which employs a creator behavioral model to identify misalignment between incentivized and desirable content.

Our main theoretical contributions focus on the existence and character of Nash equilibria in exposure games. We found that seemingly innocuous *algorithmic* choices like temperature τ, embedding dimension d, or a non-negativity constraint on embeddings can have serious impact on the induced incentives. For example, high τ incentivizes uniform broadly appealing content, whereas low τ motivates targeting smaller consumer groups. Since higher τ is often linked to exploration, which is necessary for optimal performance in a static setting [139], this result highlights the importance of considering the socio-economic context in algorithm development.

Our producer model has several limitations which we aim to address in the future, from assuming rationality, complete information, and full control, to taking the skill set of each producer to be equivalent, their utility to be linear in total exposure, and ignoring algorithmic diversification of recommendations. We also consider the attention pool as fixed and finite, neglecting the problematic reality of the modern attention economy, and the constant struggle of online platforms to increase the number of users and daily usage [26, 47, 238]. The empirical evaluation of our behavior model is hindered by the lack of academic access to the almost exclusively privately run platforms [86].

Due to their sizable influence on individuals, societies, and economy [165], information and recommender systems are of critical importance from an ethical and societal perspective. While we hope that a better understanding of the incentives these algorithms create will mitigate their negative social consequences, this also entails risks. Perhaps the most important is the possibility of employing an optimizer such as the one in Section 5.3 to game a real-world algorithm. This is especially relevant to the current debate about transparency [e.g., 193, 216, 217], and the proposal to (partially) open-source the Twitter code base [129]. Due to the aforementioned limitations, we also caution against treating the predictions of our incentive-based behavior model as definitive, especially given the significant complexity of many real-world algorithms and the environments in which they operate.

Going forward, we want to deepen our understanding of exposure games, and make pre-deployment audits a practical addition to the algorithm auditing toolbox. We hope this research enriches the debate about online platforms by a useful perspective for thinking about harms, (over)amplification, and design of algorithms with regard to the relevant incentives of the involved actors.

5.5 Proofs

Stand-alone statements

Theorem 3. *PNE need not exist in either the hardmax ($\tau = 0$) or softmax ($\tau > 0$) exposure games.*

Proof of Theorem 3. **(I) Hardmax:** Take $n = d = 2$ and $P_c = \frac{1}{3}\sum_{j=1}^{3} \delta_{c_j}$ where the angle between any $c_j \neq c_k$ is $\frac{2\pi}{3}$. Place s_1 anywhere on the circle. W.l.o.g. $c_1 = \arg\max_j \langle s_1, c_j \rangle$. Then there is v on the geodesic connecting c_2 and c_3 which has higher dot product with both c_2 and c_3 than s_1.

(II) Softmax: Let $n = d = 2$, and $P_c = \frac{1}{3}(2\delta_{e_1} + \delta_{e_2})$ where $e_1 = [1, 0]^\top$ and $e_2 = [0, 1]^\top$. By Proposition 3, we know that the only possible PNE is $s_1 = s_2 = \bar{c} \propto \mathbb{E}\,[c] = [2, 1]/3$, where both players enjoy $u_1(s) = u_2(s) = \frac{1}{2}$. Let $s_1' = (\bar{c} + \epsilon e_1)/\|\bar{c} + \epsilon e_1\|$ for some $\epsilon > 0$. As $\tau \to 0$, $u_1(s_1', \bar{c}) \to \frac{2}{3}$ by continuity. Hence $\exists \tau_0 > 0$ s.t. $s_1 = s_2 = \bar{c}$ is not a PNE for all $\tau < \tau_0$. $\square$

Theorem 4. *For any $\epsilon > 0$ and $P_c \in \mathcal{P}(\mathbb{R}d)$ with compact support and $\mathbb{E}[c] \neq 0$, $\exists \tau_0 > 0$ s.t. $s_1 = \ldots = s_n = \bar{c}$ is an ϵ-PNE for all $\tau \geq \tau_0$. Moreover, for all $\tau \geq \tau_0$, the smallest ϵ_τ for which $\bar{c}$ is an ϵ_τ-PNE satisfies $\epsilon_\tau \leq \frac{\epsilon}{\tau}$. If also $\epsilon < \|\widehat{c}\|$, then the set of better-responses to $\bar{c}$*

$$\Psi(\bar{c}) \coloneqq \{v \in S^{d-1} \colon u_1(v, \bar{c}, \ldots, \bar{c}) \geq u_1(\bar{c}, \bar{c}, \ldots, \bar{c})\} ,$$

is a subset of $B_\delta^d(\bar{c}) = \{v \colon \|v - \bar{c}\| \leq \delta\}$ with $\delta = 2\epsilon/(\|\widehat{c}\| - \epsilon)$, and $\delta \to 0$ as $\tau \to \infty$.

Proof of Theorem 4. We w.l.o.g. focus on the defection strategies for s_1. By the mean-value theorem

$$\Delta \coloneqq u_1(s_1, \bar{c}, \ldots, \bar{c}) - u_1(\bar{c}, \ldots, \bar{c}) = \langle g_1', s_1 - \bar{c} \rangle ,$$

where $g_1' = \nabla_{s_1'} u_1(s_1', \bar{c}, \ldots, \bar{c})$ for some s_1' on the *line* connecting s_1 and $\bar{c}$. While the rigorous argument below relies on a few technicalities, the main idea is simple: as $\tau \to \infty$, $\tau \cdot g_1' \to \widehat{c} = \frac{1}{n}(1 - \frac{1}{n})\mathbb{E}[c]$ *uniformly* over $s_1 \in S^{d-1}$ (Lemma 6), and thus $\tau \cdot \Delta \approx \langle \widehat{c}, s_1 - \bar{c} \rangle \leq \|\widehat{c}\|(1-1) = 0$.

Lemma 6. $\lim_{\tau \to \infty} \sup_{s_1 \in S^{d-1}} \|\tau \cdot g_1' - \widehat{c}\| \to 0$.

Proof of Lemma 6. Since $\operatorname{supp}(P_c)$ is compact by assumption, and $\tau \cdot g_1' = \mathbb{E}[p_1(c)(1 - p_1(c))c]$, all we need is $p_1(c)(1 - p_1(c)) \to \frac{1}{n}(1 - \frac{1}{n})$ *pointwise* in c (dominated convergence), and *uniform* over $s_1' \in B^d = \{v \colon \|v\| \leq 1\}$ (mean-value theorem yields s_1' on the *line* connecting s_1 with $\bar{c}$). As

$$s_1' \mapsto \left| p_1(c)(1 - p_1(c)) - \tfrac{1}{n}(1 - \tfrac{1}{n}) \right| , \quad \text{with} \quad p_1(c) = \frac{\exp(\tau^{-1}\langle c, s_1' \rangle)}{\exp(\tau^{-1}\langle c, s_1' \rangle) + (n-1)\exp(\tau^{-1}\langle c, \bar{c} \rangle)} ,$$

is continuous, it is maximized by some $(s_1^\star, c^\star) \in B^d \times \operatorname{supp}(P_c)$ (Tychonoff and compactness of $\operatorname{supp}(P_c)$). By monotonicity, $(s_1^\star, c^\star)$ will be a maximizer $\forall \tau' \geq \tau$. Conclude by continuity. $\square$

For any given $\epsilon > 0$, Lemma 6 can be combined with

$$\tau \cdot \Delta \leq \langle \widehat{c}, s_1 - \bar{c} \rangle + \|\tau \cdot g_1' - \widehat{c}\| \|s_1 - \bar{c}\| ,$$

where $\langle \widehat{c}, s_1 - \bar{c} \rangle \leq 0$ for all $s_1 \in S^{d-1}$ by $\bar{c} = \widehat{c}/\|\widehat{c}\|$, to obtain $\Delta < \varepsilon$ for a sufficiently large τ. In particular, Lemma 6 yields a τ_0 such that $\|\tau_0 \cdot g_1' - \widehat{c}\| \leq \epsilon/2$, which ensures

$$\|\tau \cdot g_1' - \widehat{c}\| \|s_1 - \bar{c}\| \leq 2\|\tau \cdot g_1' - \widehat{c}\| \leq \epsilon ,$$

for all $\tau \geq \tau_0$. Hence $\bar{c}$ is at least an $\frac{\epsilon}{\tau}$-PNE for all $\tau \geq \tau_0$ (w.l.o.g. $\tau_0 \geq 1$).

The above can be used to obtain a bound on $\delta := \|s_1 - \overline{c}\|$ for $s_1 \in \Psi(\overline{c})$. Using orthogonality

$$\begin{aligned}\Delta &= \langle (I - \overline{c}\overline{c}^\top) g_1', s_1 - \overline{c} \rangle + \langle \overline{c}, g_1' \rangle \langle \overline{c}, s_1 - \overline{c} \rangle \\ &\leq \tau^{-1} \|s_1 - \overline{c}\| \left[\left\| (I - \overline{c}\overline{c}^\top) \tau \cdot g_1' \right\| - \tfrac{1}{2} \langle \overline{c}, \tau \cdot g_1' \rangle \, \|s_1 - \overline{c}\| \right] ,\end{aligned}$$

by the triangle inequality, and $\langle \overline{c}, s_1 - \overline{c} \rangle = \frac{1}{2}(2\langle \overline{c}, s_1 \rangle - 2) = -\frac{1}{2}\|s_1 - \overline{c}\|^2$ by $s_1, \overline{c} \in S^{d-1}$. Again by orthogonality, $\epsilon^2 > \|\tau \cdot g_1' - \widehat{c}\|^2 = \|(I - \overline{c}\overline{c}^\top)\tau \cdot g_1'\| + |\langle \overline{c}, \tau \cdot g_1' \rangle - \|\widehat{c}\||^2$, which implies

$$\tau \cdot \Delta < \delta \left[\epsilon - \tfrac{\delta}{2}(\|\widehat{c}\| - \epsilon) \right] .$$

The r.h.s. is positive only when $0 < \delta < 2\epsilon / (\|\widehat{c}\| - \epsilon)$. Since ϵ in Section 5.5 is only used as an upper bound on $\|\tau \cdot g_1' - \widehat{c}\|$, and Lemma 6 tells us this norm converges to zero, $\delta \to 0$ as $\tau \to \infty$. □

Proposition 3. *A PNE always exists in $n = d = 2$ non-negative* hardmax *games, but may not without non-negativity or when $d > 2$. For $n = 2$ non-negative* softmax *games with $\widehat{c} := \frac{1}{n}(1 - \frac{1}{n})\mathbb{E}[c] \neq 0$, the only possible PNE is $s_1 = s_2 = \overline{c}$ with $\overline{c} := \frac{\widehat{c}}{\|\widehat{c}\|}$ (independently of d), but a PNE may not exist. When $n > 2$, non-negative softmax games can have a PNE other than $s_1 = \cdots = s_n = \overline{c}$.*

Proof of Proposition 3. **(I) Hardmax:** For **existence when** $n = d = 2$, let θ_c be the angle of c from (w.l.o.g.) $[1, 0]$, and let $A \subset C$ denote the set of angles such that for every $\theta_m \in A$, $\mathbb{P}(\theta_c \leq \theta_m) = \frac{1}{2}$ and $\mathbb{P}(\theta \geq \theta_m) = \frac{1}{2}$, with $\mathbb{P}$ implied by the underlying P_c. Then any $(s_1, s_2) \in A \times A$ is a PNE.

For **non-existence when** $d > 2$, consider $d = 3$ and $P_c = \frac{1}{3}\sum_{j=1}^{3} \delta_{c_j}$ where c_j are the three canonical basis vectors. Disregards of s_1 location, there will be a point on the great circle connecting the two most distant points from s_1 (break ties arbitrarily) which is closer to both of the two.

For **non-existence without non-negativity** in $d = 2$, see the hardmax part of the Theorem 3 proof.

(II) Softmax: In **the** $n = 2$ **case**, a necessary condition for $s = (s_1, s_2)$ to be a PNE is that the Riemannian gradients of the utility—$(I - s_i s_i^\top) g_i$ with $g_i = \nabla_{s_i} u_i(s)$—are zero. Since $\nabla_{s_i} u_i(s) = \tau^{-1}\mathbb{E}[p_i(c)(1 - p_i(c))c]$, g_i belongs to the first orthant by the definition of the non-negative game, and it is not zero (for $\tau > 0$, all probabilities lie in $(0, 1)$, and c is *not* a.s. zero since we assumed $\mathbb{E}[c] \neq 0$). Hence the Riemannian gradients can only be zero if $s_i \propto g_i$, and in particular $s_i = g_i / \|g_i\|_2$ because this is the direction which makes dot products with all vectors in the first orthant positive.

Crucially, $g_1 = g_2$ in 2-player games due to the symmetry of $p_1(c)(1 - p_1(c)) = p_1(c)p_2(c) = p_2(c)(1 - p_2(c))$. Therefore at a PNE, $s_1 = s_2$ in which case $p_i(c) = \frac{1}{2}$ for all c, and thus $g_i(s) \propto \mathbb{E}[c]$, implying $s_1 = s_2 = \overline{c}$ is the only possible PNE. To show it may not be a PNE, consider $P_c = \frac{1}{3}(2\delta_{c_1} + \delta_{c_2})$ for arbitrary non-zero $c_1 \neq c_2$ in the first orthant. Then $\overline{c} \propto 2c_1 + c_2$ with $u_1(\overline{c}, \overline{c}) = u_2(\overline{c}, \overline{c}) = 1/2$. Fixing $s_1 = c_1 / \|c_1\|_2$ and taking $\tau \downarrow 0$, we get

$u_1(s_1, \bar{c}) \to 2/3$, and thus there exists a $\tau > 0$ for which $s_1 = c_1/\|c_1\|_2$ is a strict improvement over $s_1 = \bar{c}$ when $s_2 = \bar{c}$.

For **the** $n > 2$ **case**, we focus on a two-dimensional $n = 4$ game with $P_c = \frac{1}{2}(\delta_{c_1} + \delta c_2)$ with $c_1 = [1, 0]^\top$ and and $c_2 = [0, 1]^\top$ (the two canonical basis vectors). In particular, we investigate existence NE of the form $s_1 = s_2$ and $s_3 = s_4$. Since $d = 2$, the strategies are restricted to S^1, we can use polar coordinates to parameterize $s_i = \varphi(\theta_i) := [\cos(\theta_i), \sin(\theta_i)]^\top$.

The maximum utility of each player in a NE is $1/4$, since otherwise each of the two players in the less profitable location (say $s_1 = s_2$) could jump to the other location ($s_3 = s_4$), gaining utility $\geq 1/4$. This narrows down our search to configurations where every player has utility *equal to* $1/4$, which is only the case when $\theta_1 = \theta_2 = \theta$ and $\theta_3 = \theta_4 = \frac{\pi}{2} - \theta$ for some $\theta \in [0, \frac{\pi}{4}] =: K$. We can thus define

$$Q := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

and look for which values of $\theta \in K$ does (w.l.o.g.) $\nabla_{\theta_1} u_1(s) = \langle g_1, Qs_1 \rangle$ equal zero.

Lemma 7. *For a sufficiently small $\tau > 0$, $f \colon \theta \mapsto \langle g_1(s), Qs_1 \rangle$ is* strictly *convex on K.*

Proof of Lemma 7. It is sufficient to prove that $f'' > 0$ on K. For this, observe $f'(\theta) = \|Qs_1\|_{H_1}^2 - \langle g_1, s_1 \rangle$ where $H_1 := \nabla_{s_1}^2 u_1(s)$, and $f''(\theta) = \|Qs_1\|_{\nabla_{\theta_1} H_1}^2 - \langle Qs_1, 3H_1 s_1 + g_1 \rangle \geq \|Qs_1\|_{\nabla_{\theta_1} H_1}^2 - 3\|H_1\|_2 - \|g_1\|_2$, where by construction

$$g_1 = \frac{1}{2\tau} \begin{pmatrix} p_1(c_1)(1 - p_1(c_1)) \\ p_1(c_2)(1 - p_1(c_2)) \end{pmatrix},$$

$$H_1 = \frac{1}{2\tau^2} \begin{pmatrix} (1 - 2p_1(c_1))p_1(c_1)(1 - p_1(c_1)) & 0 \\ 0 & (1 - 2p_1(c_2))p_1(c_2)(1 - p_1(c_2)) \end{pmatrix}$$

$$\nabla_{\theta_1} H_1 = \frac{1}{2\tau^3} \begin{pmatrix} (1 - 6p_1(c_1)(1 - p_1(c_1))p_1(c_1)(1 - p_1(c_1)) & 0 \\ 0 & (1 - 6p_1(c_2)(1 - p_1(c_2))p_1(c_2)(1 - p_1(c_2)) \end{pmatrix},$$

Hence $\|Qs_1\|_{\nabla_{\theta_1} H_1}^2 \sim \tau^{-3}$, $\|H_1\|_2 \sim \tau^{-2}$, and $\|g_1\|_2 \sim \tau^{-1}$, implying that for τ low enough, the positive term $\|Qs_1\|_{\nabla_{\theta_1} H_1}^2$ dominates (using that all expressions share the term $p_1(c)(1-p_1(c))$, and thus after dividing and observing $\tau \to 0$ gives $p_1(c)$ close to either one or zero, we get that all the terms scale as $p_1(c)(1 - p_1(c))/\tau^k$ for the appropriate $k \in \{1, 2, 3\}$). □

Lemma 7 implies there are at most two NE (f is strictly convex, and $f(\theta) = 0$ is a necessary condition). At $\theta = \frac{\pi}{4}$, $s_1 = s_2 = s_3 = s_4 = \bar{c}$ by definition, which we know is a critical point, so $f(\frac{\pi}{4}) = 0$. Since $g_1 \propto \mathbb{E}[p_1(c)(1 - p_1(c))c] \neq 0$ for any $\tau > 0$ (c cannot be a.s. 0 by $\mathbb{E}[c] \neq 0$ and the non-negativity assumption), $f(0) = \langle g_1, Qs_1 \rangle > 0$ ($s_1 = \varphi(0) = e_1 = [1, 0]^\top$). The other possible root of f thus could only be in the interior $(0, \frac{\pi}{4})$ of K. Since f is negative when moving towards $e_1 = [1, 0]^\top$ increases utility, we can see for example $f(\frac{\pi}{8}) < 0$ if τ is sufficiently small. Hence there exists $\tau > 0$ and $\theta_\tau^\star \in (0, \frac{\pi}{8})$ s.t. $f(\theta_\tau^\star) = 0$ by the mean value theorem.

So far we have established that $s_1 = s_2 = \varphi(\theta_\tau^\star)$, $s_3 = s_4 = \varphi(\frac{\pi}{2} - \theta_\tau^\star)$ is a *local* NE for the corresponding small τ. By symmetry, it is sufficient to check if there is a defection strategy for s_1. Any defection to $\theta_1 \in (\theta^\star_\tau, \frac{\pi}{2} - \theta^\star_\tau]$ will result in $p_1(c) < \frac{1}{4}$ for both $c = c_1, c_2$, and thus worse utility. Defection to $(\frac{\pi}{2} - \theta^\star_\tau, \frac{\pi}{2}]$ will not yield utility greater than defection to $[0, \theta^\star_\tau)$ since $s_3 = s_4 = \varphi(\frac{\pi}{2} - \theta^\star_\tau)$, so it is sufficient to focus on $\theta_1 \in [0, \theta^\star_\tau)$. Here

$$\begin{aligned}\nabla_{\theta_1} u_1(s) &= \langle g_1, Q s_1 \rangle \\ &\propto p_1(c_2)(1 - p_1(c_2))\cos(\theta_1) - p_1(c_1)(1 - p_1(c_1))\sin(\theta_1) \\ &\geq p_1(c_1)(1 - p_1(c_1))[\cos(\theta_1) - \sin(\theta_1)]\,,\end{aligned}$$

since $p_1(c_1)$ grows quicker than $p_1(c_2)$ decays. By construction, $\theta^\star_\tau < \frac{\pi}{4}$, and we know $\cos(\theta) - \sin(\theta) > 0$ for $\theta \in [0, \frac{\pi}{4})$. In other words, the utility of s_1 is strictly increasing on $\theta_1 \in [0, \theta^\star_\tau)$, i.e., none of the corresponding $s_1 = \varphi(\theta_1)$ is an improvement. Hence $s_1 = s_2 = \varphi(\theta^\star_\tau)$, $s_3 = s_4 = \varphi(\frac{\pi}{2} - \theta^\star_\tau)$ is a NE. (The construction is illustrated in Fig. 5.6.) $\square$

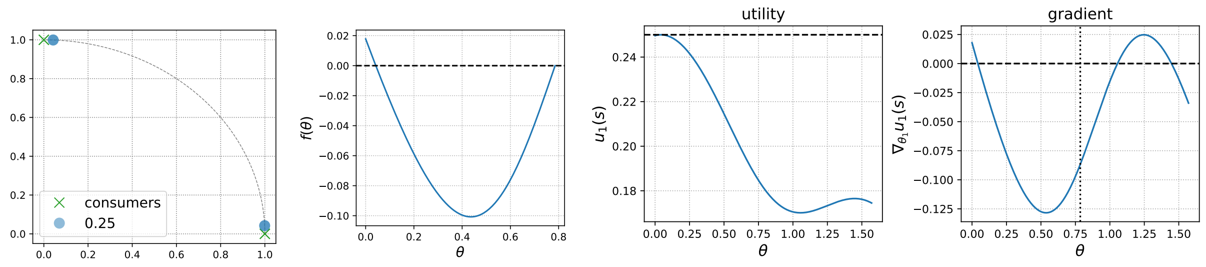


Figure 5.6: $n > 2$ softmax case from the proof of Proposition 3. Left: Symmetric PNE location (here for $\tau = \frac{1}{4}$). **Middle left:** Plot of $f(\theta) = \langle g_1(s), Q s_1 \rangle$ with $s_1 = s_2 = \varphi(\theta)$ and $s_3 = s_4 = \varphi(\frac{\pi}{2} - \theta)$. **Right:** Plot of utility and its gradient for all possible defection strategies $s_1 = \varphi(\theta_1)$ with s_2, s_3, s_4 kept put in the positions shown on the right. Vertical line shows $\frac{\pi}{4}$ (right end of K).

Inline statements

Lemma 8. *The distribution from the part (I) of the proof of Theorem 3—$d = 2$, $P_c = \frac{1}{3}\sum_{j=1}^{3} \delta_{c_j}$ with $c_j \neq c_k$ $\frac{2\pi}{3}$ apart—admits a* mixed *NE $P_1 = P_2 = P_c$ at $\tau = 0$.*

Proof of Lemma 8. By symmetry, $u_i(P_c, P_c) = \frac{1}{2}$, $\forall i$. Since for any $s_1 \in \operatorname{supp}(P_c) = \{c_1, c_2, c_3\}$

$$\mathbb{E}_{c, s_2}\left[u_1(s_1, s_2) \mid s_1\right] = \tfrac{1}{3}[u_1(s_1, c_1) + u_1(s_1, c_2) + u_1(s_1, c_3)] = \tfrac{1}{3}[1 \cdot \tfrac{1}{2} + 2 \cdot \tfrac{1}{3}(1 + \tfrac{1}{2} + 0)] = \tfrac{1}{2}\,,$$

all we need is to show that $\mathbb{E}_{c, s_2}\left[u_1(s_1, s_2) \mid s_1\right] \leq \frac{1}{2}$ for any $s_1 \notin \operatorname{supp}(P_c)$. W.l.o.g. assume s_1 lies on the geodesic connecting c_1 and c_3 (i.e., on the arc opposite of c_2). Such an s_1 is closer to c_1 and c_3 then c_2 ($u_1(s_1, c_2) = \frac{2}{3}$), but is further from c_1 and c_2 (resp. c_3 and c_2) than c_1 (resp. c_3). Hence

$$\mathbb{E}_{c,s_2}\left[u_1(s_1, s_2) \mid s_1\right] = \tfrac{1}{3}[u_1(s_1, c_1) + u_1(s_1, c_2) + u_1(s_1, c_3)] = \tfrac{1}{3}[1 \cdot \tfrac{2}{3} + 2 \cdot \tfrac{1}{3}] = \tfrac{4}{9}\,.$$

Since $\frac{4}{9} < \frac{1}{2}$, s_1 has no incentive to move any of its mass away from $\operatorname{supp}(P_c)$. $\square$

5.6 Hardmax games

In this section we present two different algorithms for finding mixed Nash equilibria in two-player hardmax games. We note that the set of allowable mixed strategies must be restricted in some way since certain distributions with support on the unit-sphere S^{d-1} require infinite storage. Hence, our first algorithm finds a mixed NE for a discretized strategy space, while our second algorithm considers settings where P_c is discrete and finds a mixed NE with support over a finite number of pure strategies in the original non-discretized space, assuming such a mixed NE exists.

We caution that both of these algorithms can only find mixed NE for small exposure games due to their poor scaling properties. We list them here to highlight the difficulty of solving hardmax games when compared to the softmax setting and to serve as inspiration for future research into more efficient algorithms.

Discretized Games

We first consider the setting where both players may only choose mixed strategies with support over a finite subset $A = \{s^{(1)}, s^{(2)}, \ldots, s^{(m)}\} \subset S^{d-1}$ of pure strategies. This setting includes embeddings that are represented using floating point numbers although A will be very large. In this case the mixed strategy of the players can be expressed as an m-dimensional probability vector π_i with $\pi_{ij} = P_i\left(s^{(j)}\right)$. Since there are a finite set of pure strategies a mixed NE is guaranteed to exist [175]. Furthermore since this is a two-player constant-sum game we can find a mixed NE by solving the following linear program [60]

$$\begin{array}{ll} \underset{\alpha}{\text{maximize}} & \alpha \\ \text{subject to} & Ux \geq \alpha \mathbf{1} \\ & \mathbf{1}^\top x = 1 \\ & x_i \geq 0,\ i = 1, \ldots, m, \end{array}$$

where $U_{ij} = u_1\left(s^{(i)}, s^{(j)}\right)$. Then the strategies where $\pi_1 = \pi_2 = x$, correspond to a mixed NE. While such a problem is simple to formulate and solve, the number of possible strategies will grow rapidly as d increases for most discretization schemes. For example, we might create a uniform grid of k points over each spherical coordinate, in which case we will have $m = k^{d-1}$ pure strategies to consider.

Finite Support

Next, we consider the setting where both players choose mixed strategies with support over at most m pure strategies and the support of P_c is over l points, $supp(P_c) = \{c_1, c_2, \dots, c_l\}$. Unlike the discretized case however, the players may choose any pure strategy that lies on S^{d-1}. We begin by outlining a method that, given a mixed strategy P, finds all pure strategies D that dominate it: $\mathbb{E}_{s_1 \sim P}[u_1(s_1, s_2) - u_2(s_1, s_2)] < 0$ for all $s_2 \in D$. Where we assume Player 1 provides the mixed strategy without loss of generality by symmetry. We will then use this method as a subroutine to find mixed nash equilibria.

Lemma 9. *(P_1, P_2) is a mixed NE if and only if $\mathbb{E}_{s_1 \sim P_1}[u_1(s_1, s)] \geq \frac{1}{2}$ and $\mathbb{E}_{s_2 \sim P_2}[u_2(s, s_2)] \geq \frac{1}{2}$ for all pure strategies $s \in S^{d-1}$.*

Proof of Lemma 9. Assume (P_1, P_2) is a mixed NE, then by definition $\mathbb{E}_{(s_1,s_2)\sim(P_1,P)}[u_1(s_1, s_2)] \geq \frac{1}{2}$ for all mixed strategies $P \in \mathcal{P}(S^{d-1})$, since each pure strategy is also a mixed strategy it follows that $\mathbb{E}_{s_1 \sim (P_1, s)}[u_1(s_1, s)] \geq \frac{1}{2}$ for all $s \in S^{d-1}$. Similarly for Player 2.

Now assume we have two mixed strategies (P_1, P_2) such that $\mathbb{E}_{s_1 \sim P_1}[u_1(s_1, s)] \geq \frac{1}{2}$ and $\mathbb{E}_{s_2 \sim P_2}[u_2(s, s_2)] \geq \frac{1}{2}$ for all $s \in S^{d-1}$. Given a mixed strategy $P \in \mathcal{P}(S^{d-1})$ it follows that

$$\begin{aligned}
\mathbb{E}_{(s_1,s_2)\sim(P_1,P)}[u_1(s_1, s_2)] &= \mathbb{E}_{s_2 \sim P}[\mathbb{E}_{s_1 \sim P_1}[u_1(s_1, s_2)]] \\
&= \int_{S^{d-1}} \mathbb{E}_{s_1 \sim P_1}[u_1(s_1, s_2)] dP(s_2) \\
&\geq \int_{S^{d-1}} \frac{1}{2} dP(s_2) = \frac{1}{2}.
\end{aligned}$$

Similarly for Player 2. □

Lemma 9 allows us to only consider pure strategies when checking if strategies are mixed NE.

Now given a mixed strategy P with finite support $\text{supp}(P) = \{s^{(1)}, s^{(2)}, \dots, s^{(m)}\}$ we can find every subset of S^{d-1} that does not satisfy the condition in Lemma 9. Each $s^{(i)}$ partitions S^{d-1} into 3^l disjoint partitions $\mathcal{X}^{(i)} = \{X_1^{(i)}, X_2^{(i)}, \dots, X_{3^l}^{(i)}\}$, with $X_j^{(i)}$ satisfying

$$j_k = \begin{cases} 2 \ \texttt{if} \ \langle s^{(i)}, c_k \rangle > \langle s, c_k \rangle \\ 1 \ \texttt{if} \ \langle s^{(i)}, c_k \rangle = \langle s, c_k \rangle \\ 0 \ \texttt{if} \ \langle s^{(i)}, c_k \rangle < \langle s, c_k \rangle, \end{cases}$$

for all pure strategies $s \in X_j^{(i)}$, where j_k is the k-th digit in the ternary representation of j. We can further partition the space into 3^{lm} disjoint partitions $\mathcal{Y} = \{Y_1, Y_2, \dots, Y_{3^{lm}}\}$ with $Y_i = \bigcap_{j=1}^m X_{i_j}^{(j)}$ where i_j is the j-th digit of the 3^l-ary representation of i.

For every $Y \in \mathcal{Y}$ we have $\mathbb{E}_{s_1 \sim P}[u_1(s_1, s)] = \mathbb{E}_{s_1 \sim P}[u_1(s_1, s')]$ for all $s, s' \in Y$ by construction. Thus, we can find the set of all pure strategies D that dominate P by iterating over $\mathcal{Y}$, testing a single point in each partition, and taking unions:

$$Z = \left\{ Y \in \mathcal{Y} : s \in Y \implies \mathbb{E}_{s_1 \sim P}[u_1(s_1, s)] < \frac{1}{2} \right\}, \; D = \bigcup_{Y \in Z} Y.$$

It follows from Lemma 9 that (P, P) is a mixed NE if and only if D is empty.

Finally, we outline a method to find mixed NE. We first note that for every positive integer m, every pure strategy $s \in S^{d-1}$ defines a feasible set F_s of all mixed strategies with support over at most m pure strategies that are not dominated by s, that is:

$$F_s = \left\{ P = \sum_{i=1}^{m} \pi_i \delta_{s^{(i)}} : \sum_{i=1}^{m} \pi_i u_1(s^{(i)}, s) \geq \frac{1}{2} \right\},$$

where π is an m-dimensional probability vector. It follows from Lemma 4 that if P is mixed strategy with support over at most m points then (P, P) is a mixed NE if and only if $P \in \bigcap_{s \in S^{d-1}} F_s$. We can frame finding such a strategy P as an optimization problem

$$\begin{aligned} \underset{\mathbf{P} \subset \mathcal{P}}{\text{minimize}} \quad & |\mathbf{P}| \\ \text{subject to} \quad & \mathbf{P} \cap F_s \neq \emptyset, \; s \in S^{d-1}, \end{aligned}$$

where $\mathcal{P}$ is the set of all mixed strategies with support over at most m pure strategies. An optimal solution with more than one element in $\mathbf{P}$ indicates that there does not exist a mixed strategy with support over m points or fewer, whereas if $|\mathbf{P}| = 1$ then (P, P) is a mixed strategy where P is the singleton element in $\mathbf{P}$.

This is an instance of the *implicit hitting set problem*. Hence, we can use the algorithm proposed in Section 2.1 of Chandrasekaran et al. [42] to solve the above optimization problem. Their algorithm assumes an oracle that, given a proposed subset $\mathbf{P} \subseteq \mathcal{P}$ will return a subset F_s that is not hit $\mathbf{P} \cap F_s = \emptyset$ or will certify $\mathbf{P}$ as a feasible solution to the above optimization problem. We can easily achieve this by finding all dominating pure NE using our proposed method above for each $P \in \mathbf{P}$ and taking the intersection of the resulting sets. If the intersection is empty then $\mathbf{P}$ is a feasible solution, otherwise every element in the intersection represents a subset F_s that has not been hit by $\mathbf{P}$.

5.7 Experiments

Setup

The LastFM dataset was preprocessed by Shakespeare et al. [208]. Original larger scale sweep was executed with $n \in \{10, 25, 100, 500, 1500\}$, $d \in \{3, 10, 50, 100\}$, stepsize in $\{10^{-3}, 10^{-2}, 10^{-1}\}$, and $\tau \in \{10^{-2}, 10^{-1}, 0.25, 0.5, 1.0\}$. We only used 2 random seeds for the recommender, and 3 random seeds for our LNE-finding algorithm (i.e., 6 runs in total per configuration). For the reported results, stepsize sweep was restricted to$\{10^{-2}, 10^{-1}\}$; the number of steps was upper bounded by 50,000 (all runs have successfully converged to a fixed point as mentioned). While our code contains an option to `scale_lr_by_temperature` (see the `config.py` file in the provided code), which multiplies the stepsize by τ before its use, we did not use this option in the experiments.

The second-order Riemannian test (Definition 3) is implemented in `manifold.py`. Defining the tangent space projection $\Pi_i \coloneqq (I - s_i s_i^\top)$, we consider a candidate strategy profile $s \in (S^{d-1})^n$ as *violating* the second order test if any of the Riemannian gradients $\Pi_i \nabla_{s_i} u_i(s)$ had ℓ^2-norm higher than $10^{-5} \cdot \sqrt{d}$, *or* the Riemannian Hessian $\Pi_i [\nabla^2_{s_i} u_i(s)] \Pi_i - \langle s_i, \nabla_{s_i} u_i(s) \rangle \Pi_i$ had a strictly positive eigenvalue (no tolerance used here).

The final MovieLens and LastFM experiments were run on 72 AWS machines, each with 4 CPU cores, for 5 hours. Including preliminary and failed runs, we used over 50K CPU hours.

Additional plots

Section 5.7 contains plots where the second-order test confirmed and LNE. Section 5.7 then offers comparison to a third ranking algorithm: standard matrix factorization [MF; 132], i.e., PMF with additional bias terms. The bias terms effect interpretation of τ values, and we also ignore them when running the LNE-finding algorithm. This makes the comparison with PMF and NMF difficult, which is why we excluded MF from the main text. Results in Section 5.7 again contain runs where the second-order test did not rule out a LNE.

LNE confirmed by the second-order test

As mentioned, the plots shown in the main body of the paper are for runs where the second-order Riemannian test did not rule out that the found pure strategy profile is a LNE. Here we show exactly the same plots with only the runs where the test confirmed a LNE. The difference is that here we exclude the runs where the Riemannian Hessian had at least one zero eigenvalue associated with a direction perpendicular to s_i, for at least one $i \in [n]$. As you see below, this had little effect on the LastFM results, but has non-negligibly reduced the number of admitted runs for MovieLens.

Matrix factorization (PMF with biases) results

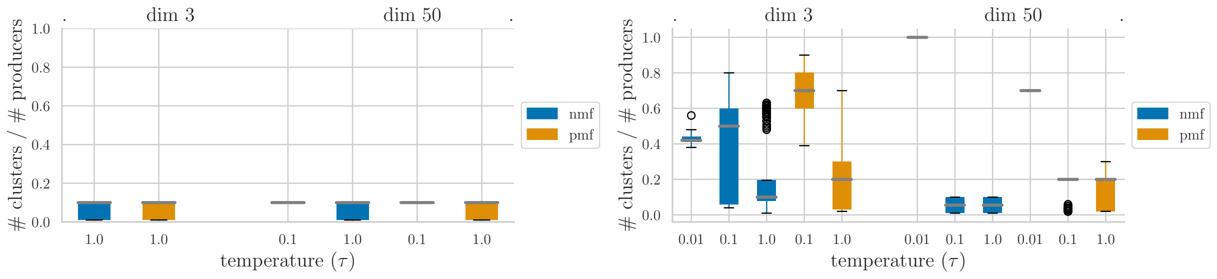


Figure 5.7: A counterpart to Fig. 5.3 with runs where LNE test was inconclusive excluded.

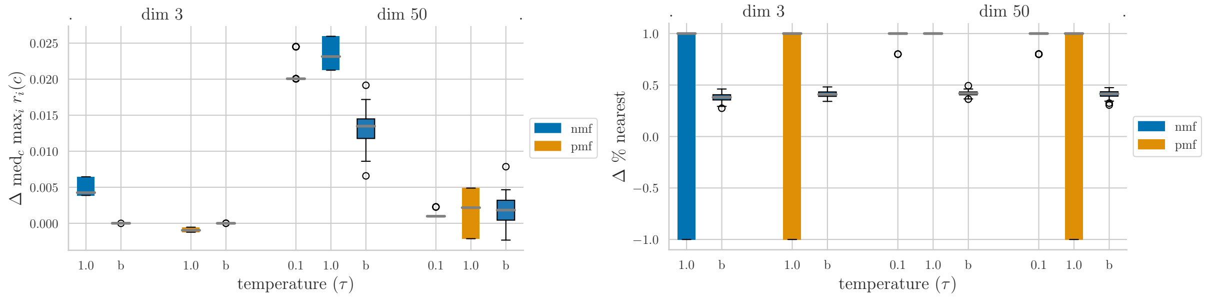


Figure 5.8: A counterpart to Fig. 5.4 with runs where LNE test was inconclusive excluded.

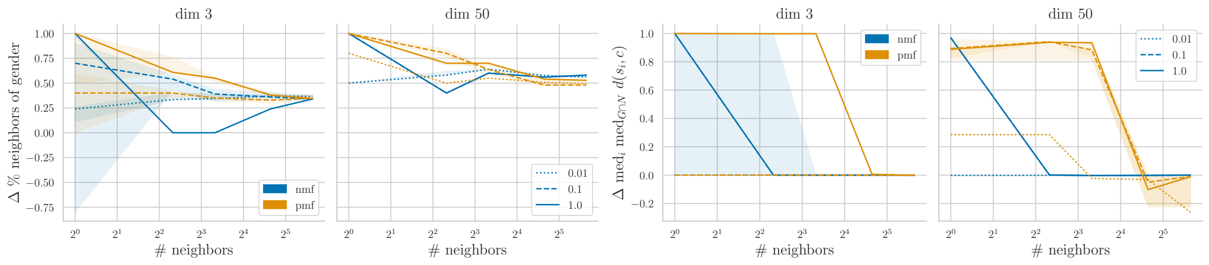


Figure 5.9: A counterpart to Fig. 5.5 with runs where LNE test was inconclusive excluded.

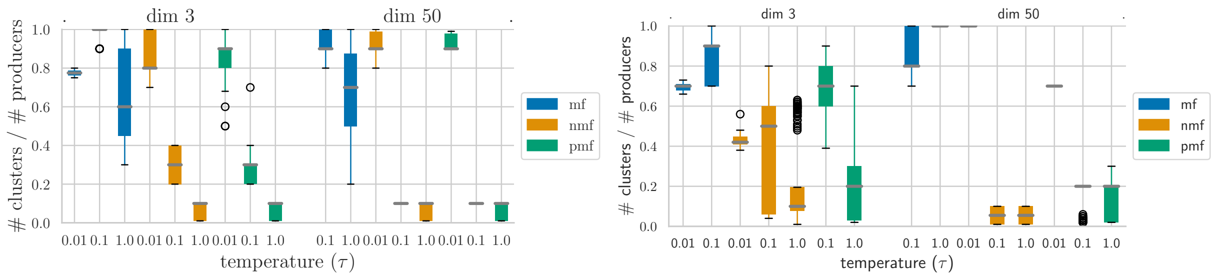


Figure 5.10: A counterpart to Fig. 5.3 with added MF results.

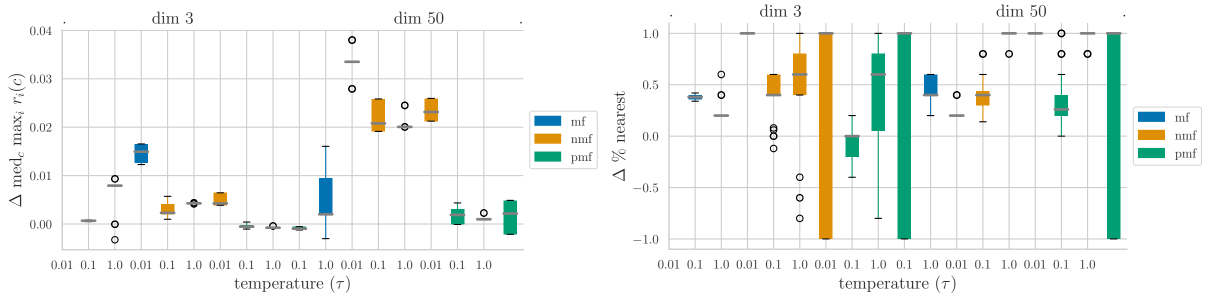


Figure 5.11: A counterpart to Fig. 5.4 with added MF results. Baselines omitted to reduce clutter.

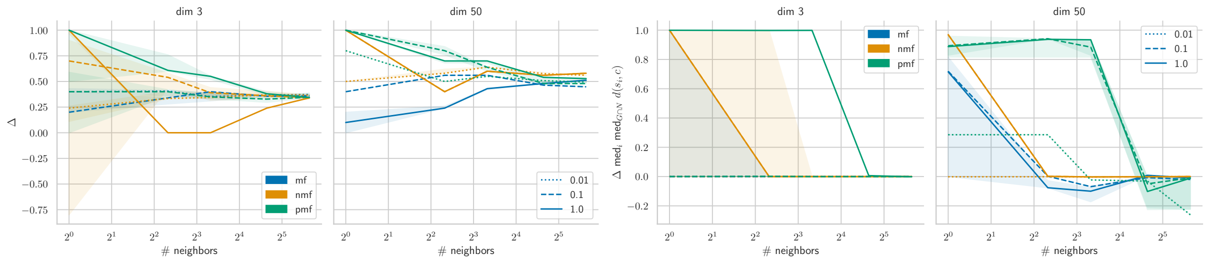


Figure 5.12: A counterpart to Fig. 5.5 with added MF results.

Bibliography

[1] Naoki Abe and Philip M Long. “Associative reinforcement learning using linear probabilistic concepts”. In: *ICML*. 1999.

[2] Gediminas Adomavicius, Jesse Bockstedt, Shawn Curley, and Jingjing Zhang. “Do recommender systems manipulate consumer preferences? A study of anchoring effects”. In: *Information Systems Research* (2013).

[3] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. “Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits”. In: *ICML*. 2014.

[4] Alekh Agarwal, Haipeng Luo, Behnam Neyshabur, and Robert E Schapire. “Corralling a band of bandit algorithms”. In: *COLT*. 2017.

[5] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. “Estimating Position Bias without Intrusive Interventions”. In: *WSDM*. 2019.

[6] Xavier Amatriain and Justin Basilico. *Netflix Recommendations: Beyond the 5 stars*. 2012.

[7] Dario Amodei and Danny Hernandez. *AI and Compute*. 2018.

[8] Nima Asadi and Jimmy Lin. “Fast candidate generation for two-phase document ranking: Postings list intersection with Bloom filters”. In: *CIKM*. 2012.

[9] Peter Auer. “Using confidence bounds for exploitation-exploration trade-offs”. In: *JMLR* (2002).

[10] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. “The non-stochastic multiarmed bandit problem”. In: *SICOMP* (2002).

[11] Babajide O Ayinde and Jacek M Zurada. “Deep learning of constrained autoencoders for enhanced understanding of data”. In: *TNNLS* (2017).

[12] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. “The mechanics of n-player differentiable games”. In: *ICML*. 2018.

[13] Hamsa Bastani, Mohsen Bayati, and Khashayar Khosravi. “Mostly exploration-free algorithms for contextual bandits”. In: *Management Science* (2021).

[14] Joeran Beel and Stefan Langer. “A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems”. In: *TPDL*. 2015.

[15] Robert Bell, Yehuda Koren, and Chris Volinsky. “Modeling relationships at multiple scales to improve accuracy of large recommender systems”. In: *SIGKDD*. 2007.

[16] Robert M Bell and Yehuda Koren. “Scalable collaborative filtering with jointly derived neighborhood interpolation weights”. In: *ICDM*. 2007.

[17] Ran Ben-Basat, Moshe Tennenholtz, and Oren Kurland. “A game theoretic analysis of the adversarial retrieval setting”. In: *Journal of Artificial Intelligence Research* (2017).

[18] Omer Ben-Porat, Gregory Goren, Itay Rosenberg, and Moshe Tennenholtz. “From recommendation systems to facility location games”. In: *AAAI*. 2019.

[19] Omer Ben-Porat, Itay Rosenberg, and Moshe Tennenholtz. “Content Provider Dynamics and Coordination in Recommendation Ecosystems”. In: *NeurIPS*. 2020.

[20] Omer Ben-Porat, Itay Rosenberg, and Moshe Tennenholtz. “Convergence of learning dynamics in information retrieval games”. In: *AAAI*. 2019.

[21] Omer Ben-Porat and Moshe Tennenholtz. “A game-theoretic approach to recommendation systems with strategic content providers”. In: *NeurIPS*. 2018.

[22] Michel Benaïm and Morris W Hirsch. “Learning Processes, Mixed Equilibria and Dynamical Systems Arising from Repeated Games”. In: *Games and Economic Behavior* (1997).

[23] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, and Chris Hesse. “Dota 2 with large scale deep reinforcement learning”. In: *arXiv* (2019).

[24] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. “The Million Song Dataset”. In: *ISMIR*. 2011.

[25] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H Chi, et al. “Fairness in recommendation ranking through pairwise comparisons”. In: *SIGKDD*. 2019.

[26] Vikram R Bhargava and Manuel Velasquez. “Ethics of the attention economy: The problem of social media addiction”. In: *Business Ethics Quarterly* (2021).

[27] Kush Bhatia, Kunal Dahiya, Himanshu Jain, Purushottam Kar, Anshul Mittal, Yashoteja Prabhu, and Manik Varma. *The extreme classification repository: Multi-label datasets and code*. 2016.

[28] Alberto Bietti, Alekh Agarwal, and John Langford. “A contextual bandit bake-off”. In: *arXiv* (2018).

[29] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *JMLR* (2003).

[30] Stephen Bonner and Flavian Vasile. “Causal embeddings for recommendation”. In: *RecSys*. 2018.

[31] Fedor Borisyuk, Krishnaram Kenthapadi, David Stein, and Bo Zhao. “CaSMoS: A framework for learning candidate selection models over structured queries and documents”. In: *SIGKDD*. 2016.

[32] Nicolas Boumal. *An Introduction to Optimization on Smooth Manifolds*. Cambridge University Press, 2022.

[33] Levi Boxell, Matthew Gentzkow, and Jesse M Shapiro. “Greater Internet use is not associated with faster growth in political polarization among US demographic groups”. In: *PNAS* (2017).

[34] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: composable transformations of Python+NumPy programs*. 2018.

[35] Leo Breiman. “Arcing Classifiers”. In: *Annals of Statistics* (1998).

[36] Leo Breiman. “Bagging Predictors”. In: *Machine Learning* (1996).

[37] Pedro Campos, Fernando Díez, and Iván Cantador. “Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols”. In: *User Modeling and User-Adapted Interaction* (2015).

[38] Erion Çano and Maurizio Morisio. “Hybrid recommender systems: A systematic literature review”. In: *Intelligent Data Analysis* (2017).

[39] Sarah Cen and Devavrat Shah. “Regulating algorithmic filtering on social media”. In: *NeurIPS* (2021).

[40] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. “Boltzmann exploration done right”. In: *NeurIPS* (2017).

[41] Soumen Chakrabarti, Alan Frieze, and Juan Vera. “The influence of search engines on preferential attachment”. In: *Internet Mathematics* (2006).

[42] Karthekeyan Chandrasekaran, Richard Karp, Erick Moreno-Centeno, and Santosh Vempala. “Algorithms for implicit hitting set problems”. In: *SODA*. 2011.

[43] Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. “How algorithmic confounding in recommendation systems increases homogeneity and decreases utility”. In: *RecSys*. 2018.

[44] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. “Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention”. In: *SIGIR*. 2017.

[45] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. “Top-k off-policy correction for a REINFORCE recommender system”. In: *WSDM*. 2019.

[46] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. “Contextual Bandits with Linear Payoff Functions”. In: *AISTATS*. 2011.

[47] Paul Covington, Jay Adams, and Emre Sargin. “Deep neural networks for YouTube recommendations”. In: *RecSys*. 2016.

[48] Koby Crammer and Claudio Gentile. “Multiclass classification with bandit feedback using adaptive regularization”. In: *Machine Learning* (2013).

[49] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. “An experimental comparison of click position-bias models”. In: *WSDM*. 2008.

[50] Mihaela Curmei, Sarah Dean, and Benjamin Recht. “Quantifying Availability and Discovery in Recommender Systems via Stochastic Reachability”. In: *ICML*. 2021.

[51] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. “Are we really making much progress? A worrying analysis of recent neural recommendation approaches”. In: *RecSys*. 2019.

[52] Pranav Dandekar, Ashish Goel, and David Lee. “Biased assimilation, homophily, and the dynamics of polarization”. In: *PNAS*. 2013.

[53] Varsha Dani, Thomas P Hayes, and Sham M Kakade. “Stochastic Linear Optimization under Bandit Feedback”. In: *COLT*. 2008.

[54] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2008.

[55] Sarah Dean, Sarah Rich, and Benjamin Recht. “Recommendations and user agency: the reachability of collaboratively-filtered information”. In: *FAccT*. 2020.

[56] Gerard Debreu. “A social equilibrium existence theorem”. In: *PNAS* (1952).

[57] Andrew Dennis. *Penguin 4.0: Necessary and positive improvement*. 2016.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL-HLT*. 2019.

[59] Virginie Do, Sam Corbett-Davies, Jamal Atif, and Nicolas Usunier. “Online certification of preference-based fairness for personalized recommender systems”. In: *arXiv* (2021).

[60] Robert Dorfman. “Application of the simplex method to a game theory problem”. In: *Activity Analysis of Production and Allocation*. 1951.

[61] Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. “Is a Good Representation Sufficient for Sample Efficient Reinforcement Learning?” In: *ICLR*. 2020.

[62] Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. “Efficient Optimal Learning for Contextual Bandits”. In: *UAI*. 2011.

[63] Miroslav Dudík, John Langford, and Lihong Li. “Doubly Robust Policy Evaluation and Learning”. In: *ICML*. 2011.

[64] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. “Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time”. In: *ACM WWW*. 2018.

[65] Marc Faddoul, Guillaume Chaslot, and Hany Farid. “A Longitudinal Analysis of YouTube’s Promotion of Conspiracy Videos”. In: *arXiv* (2020).

[66] Ky Fan. “Fixed-point and minimax theorems in locally convex topological linear spaces”. In: *PNAS* (1952).

[67] Sebastian Farquhar, Yarin Gal, and Tom Rainforth. “On statistical bias in active learning: How and when to fix it”. In: *ICLR*. 2021.

[68] William Fedus, Barret Zoph, and Noam Shazeer. “Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity”. In: *JMLR* (2022).

[69] Daniel Fleder and Kartik Hosanagar. “Blockbuster culture’s next rise or fall: The impact of recommender systems on sales diversity”. In: *Management Science* (2009).

[70] Dylan Foster, Alekh Agarwal, Miroslav Dudik, Haipeng Luo, and Robert Schapire. “Practical Contextual Bandits with Regression Oracles”. In: *ICML*. 2018.

[71] Dylan Foster, Claudio Gentile, Mehryar Mohri, and Julian Zimmert. “Adapting to Misspecification in Contextual Bandits”. In: *NeurIPS*. 2020.

[72] Dylan Foster and Alexander Rakhlin. “Beyond UCB: Optimal and Efficient Contextual Bandits with Regression Oracles”. In: *ICML*. 2020.

[73] Jerome H Friedman. “Greedy function approximation: A gradient boosting machine”. In: *Annals of Statistics* (2001).

[74] Jerome H Friedman. “Stochastic gradient boosting”. In: *Computational Statistics and Data Analysis* (2002).

[75] Drew Fudenberg and David M Kreps. “Learning mixed equilibria”. In: *Games and Economic Behavior* (1993).

[76] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. “A survey on concept drift adaptation”. In: *ACM Computing Surveys* (2014).

[77] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. “Advances and challenges in conversational recommender systems: A survey”. In: *AI Open* (2021).

[78] Yingqiang Ge, Shuya Zhao, Honglu Zhou, Changhua Pei, Fei Sun, Wenwu Ou, and Yongfeng Zhang. “Understanding Echo Chambers in E-commerce Recommender Systems”. In: *SIGIR*. 2020.

[79] Claudio Gentile and Francesco Orabona. “On multilabel classification and ranking with bandit feedback”. In: *JMLR* (2014).

[80] Avishek Ghosh, Sayak Ray Chowdhury, and Aditya Gopalan. “Misspecified linear bandits”. In: *AAAI*. 2017.

[81] Irving L Glicksberg. “A further generalization of the Kakutani fixed point theorem, with application to Nash equilibrium points”. In: *Proceedings of the American Mathematical Society* (1952).

[82] Danny Goodwin. *A Complete Guide to the Google Panda Update: 2011-21*. https://www.searchenginejournal.com/google-algorithm-history/panda-update. Accessed: 2022-05-13. 2021.

[83] Prem Gopalan, Jake M Hofman, and David M Blei. “Scalable Recommendation with Hierarchical Poisson Factorization”. In: *UAI*. 2015.

[84] Prem K Gopalan, Laurent Charlin, and David Blei. “Content-based recommendations with poisson factorization”. In: *NeurIPS* (2014).

[85] Brett R Gordon, Florian Zettelmeyer, Neha Bhargava, and Dan Chapsky. “A comparison of approaches to advertising measurement: Evidence from big field experiments at Facebook”. In: *Marketing Science* (2019).

[86] Travis Greene, David Martens, and Galit Shmueli. “Barriers to academic data science research in the new realm of algorithmic behaviour modification by digital platforms”. In: *Nature Machine Intelligence* (2022).

[87] Keach Hagey and Jeff Horwitz. *Facebook Tried to Make Its Platform a Healthier Place. It Got Angrier Instead.* 2021.

[88] Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. “Strategic classification”. In: *ACM ITCS*. 2016.

[89] Muhammad Haroon, Anshuman Chhabra, Xin Liu, Prasant Mohapatra, Zubair Shafiq, and Magdalena Wojcieszak. “YouTube, The Great Radicalizer? Auditing and Mitigating Ideological Biases in YouTube Recommendations”. In: *arXiv* (2022).

[90] F Maxwell Harper and Joseph A Konstan. “The MovieLens datasets: History and context”. In: *TIIS* (2015).

[91] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. “Array programming with NumPy”. In: *Nature* (2020).

[92] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. “Neural collaborative filtering”. In: *WWW*. 2017.

[93] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. “Evaluating collaborative filtering recommender systems”. In: *TOIS* (2004).

[94] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. "Probabilistic matrix factorization with non-random missing data". In: *ICML*. 2014.

[95] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. *Neural networks for machine learning, (lecture 6): Overview of mini-batch gradient descent*. 2012.

[96] Tin Kam Ho. "Random decision forests". In: *ICDAR*. 1995.

[97] Henning Hohnhold, Deirdre O'Brien, and Diane Tang. "Focusing on the long-term: It's good for users and business". In: *CIKM*. 2015.

[98] Homa Hosseinmardi, Amir Ghasemian, Aaron Clauset, Markus Mobius, David M Rothschild, and Duncan J Watts. "Examining the consumption of radical content on YouTube". In: *PNAS* (2021).

[99] Homa Hosseinmardi, Amir Ghasemian, Aaron Clauset, David M Rothschild, Markus Mobius, and Duncan J Watts. "Evaluating the scale, growth, and origins of right-wing echo chambers on YouTube". In: *arXiv* (2020).

[100] Harold Hotelling. "Stability in competition". In: *The Economic Journal* (1929).

[101] Jiri Hron, Karl Krauth, Michael Jordan, and Niki Kilbertus. "On component interactions in two-stage recommender systems". In: *NeurIPS*. 2021.

[102] Jiri Hron, Karl Krauth, Michael I Jordan, and Niki Kilbertus. "Exploration in two-stage recommender systems". In: *REVEAL (ACM RecSys workshop)* (2020).

[103] Jiri Hron, Karl Krauth, Michael I Jordan, Niki Kilbertus, and Sarah Dean. "Modeling Content Creator Incentives on Algorithm-Curated Platforms". In: *arXiv* (2022).

[104] Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. "The disparate effects of strategic manipulation". In: *FAccT*. 2019.

[105] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets". In: *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE. 2008, pp. 263–272.

[106] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. "Learning deep structured semantic models for web search using clickthrough data". In: *CIKM*. 2013.

[107] Nicolas Hug. "Surprise: A Python library for recommender systems". In: *Journal of Open Source Software* (2020).

[108] John D Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* (2007).

[109] Ferenc Huszár, Sofia Ira Ktena, Conor O'Brien, Luca Belli, Andrew Schlaikjer, and Moritz Hardt. "Algorithmic amplification of politics on Twitter". In: *PNAS* (2022).

[110] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. "RecSim: A Configurable Simulation Platform for Recommender Systems". In: *arXiv* (2019).

[111] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. “SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets”. In: *IJCAI*. 2019.

[112] Aleksandar Ilic and Maja Kabiljo. *Recommending items to more than a billion people*. 2015.

[113] Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[114] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. “Adaptive mixtures of local experts”. In: *Neural Computation* (1991).

[115] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. “Degenerate feedback loops in recommender systems”. In: *AIES*. 2019.

[116] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. “Unbiased Learning-to-Rank with Biased Feedback”. In: *WSDM*. 2017.

[117] Michael I Jordan and Robert A Jacobs. “Hierarchical mixtures of experts and the EM algorithm”. In: *IJCNN*. 1993.

[118] Michael I Jordan and Robert A Jacobs. “Hierarchies of adaptive experts”. In: *NeurIPS*. 1992.

[119] Dimitris Kalimeris, Smriti Bhagat, Shankar Kalyanaraman, and Udi Weinsberg. “Preference Amplification in Recommender Systems”. In: *SIGKDD*. 2021.

[120] Marius Kaminskas and Derek Bridge. “Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems”. In: *TIIS* (2016).

[121] Wang-Cheng Kang and Julian McAuley. “Candidate generation with binary codes for large-scale Top-N recommendation”. In: *CIKM*. 2019.

[122] Yuri M Kaniovski and H Peyton Young. “Learning dynamics in games with stochastic perturbations”. In: *Games and Economic Behavior* (1995).

[123] Sampath Kannan, Jamie H Morgenstern, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. “A Smoothed Analysis of the Greedy Algorithm for the Linear Contextual Bandit Problem”. In: *NeurIPS*. 2018.

[124] Komal Kapoor, Karthik Subbian, Jaideep Srivastava, and Paul Schrater. “Just in time recommendations: Modeling the dynamics of boredom in activity streams”. In: *WSDM*. 2015.

[125] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. “Efficient Thompson Sampling for Online Matrix-Factorization Recommendation”. In: *NeurIPS*. 2015.

[126] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR*. 2015.

[127] Jon Kleinberg and Manish Raghavan. “How do classifiers induce agents to invest effort strategically?” In: *TEAC* (2020).

[128] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. “Jupyter Notebooks - a publishing format for reproducible computational workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. 2016.

[129] Will Knight. *Elon Musk’s Plan to Open Source the Twitter Algorithm Won’t Solve Anything*. https://www.wired.com/story/twitter-open-algorithm-problem/. Accessed: 2022-05-16. 2022.

[130] Yehuda Koren. “Collaborative filtering with temporal dynamics”. In: *SIGKDD*. 2009.

[131] Yehuda Koren. “Factorization meets the neighborhood: a multifaceted collaborative filtering model”. In: *SIGKDD*. 2008, pp. 426–434.

[132] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* (2009).

[133] Alejandro Bellogín Kouki and Alan Said. *Offline and online evaluation of recommendations*. World Scientific Publishing Company, 2019.

[134] Karl Krauth, Sarah Dean, Alex Zhao, Wenshuo Guo, Mihaela Curmei, Benjamin Recht, and Michael I Jordan. “Do Offline Metrics Predict Online Performance in Recommender Systems?” In: *arXiv* (2020).

[135] Karl Krauth, Yixin Wang, and Michael I Jordan. “Breaking Feedback Loops in Recommender Systems with Causal Inference”. In: *arXiv* (2022).

[136] Sanath K Krishnamurthy, Vitor Hadad, and Susan Athey. “Tractable contextual bandits beyond realizability”. In: *AISTATS*. 2021.

[137] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *NeurIPS* (2012).

[138] John Langford and Tong Zhang. “The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information”. In: *NeurIPS*. 2008.

[139] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[140] Tor Lattimore, Csaba Szepesvári, and Gellert Weisz. “Learning with good feature representations in bandits and in RL with a generative model”. In: *ICML*. 2020.

[141] Mark Ledwich and Anna Zaitsev. “Algorithmic extremism: Examining YouTube’s rabbit hole of radicalization”. In: *arXiv* (2019).

[142] Daniel D Lee and H Sebastian Seung. “Learning the parts of objects by non-negative matrix factorization”. In: *Nature* (1999).

[143] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. “LLORMA: Local low-rank matrix approximation”. In: *JMLR* (2016).

[144] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”. In: *arXiv* (2020).

[145] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. “Multi-interest network with dynamic routing for recommendation at Tmall”. In: *CIKM*. 2019.

[146] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. “A contextual-bandit approach to personalized news article recommendation”. In: *WWW*. 2010.

[147] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. “Modeling user exposure in recommendation”. In: *WWW*. 2016.

[148] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. “Variational autoencoders for collaborative filtering”. In: *WWW*. 2018.

[149] Nick Littlestone and Manfred K Warmuth. *The weighted majority algorithm*. UC Santa Cruz, Computer Research Laboratory, 1989.

[150] Lydia T Liu, Nikhil Garg, and Christian Borgs. “Strategic ranking”. In: *ICAS*. 2022.

[151] Lydia T Liu, Ashia Wilson, Nika Haghtalab, Adam Tauman Kalai, Christian Borgs, and Jennifer Chayes. “The disparate equilibria of algorithmic decision making when individuals invest rationally”. In: *FAccT*. 2020.

[152] Romain Lopez, Inderjit Dhillon, and Michael I Jordan. “Learning from eXtreme Bandit Feedback”. In: *AAAI*. 2021.

[153] Eli Lucherini, Matthew Sun, Amy Winecoff, and Arvind Narayanan. “T-RECS: A simulation tool to study the societal impact of recommender systems”. In: *arXiv* (2021).

[154] John G Lynch Jr, Dipankar Chakravarti, and Anusree Mitra. “Contrast effects in consumer judgments: Changes in mental representations or in the anchoring of rating scales?” In: *Journal of Consumer Research* (1991).

[155] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. “Off-policy Learning in Two-stage Recommender Systems”. In: *WWW*. 2020.

[156] Ashok Makkuva, Pramod Viswanath, Sreeram Kannan, and Sewoong Oh. “Breaking the gridlock in Mixture-of-Experts: Consistent and Efficient Algorithms”. In: *ICML*. 2019.

[157] Andrii Maksai, Florent Garcin, and Boi Faltings. “Predicting online performance of news recommender systems through richer evaluation metrics”. In: *RecSys*. 2015.

[158] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. “Feedback loop and bias amplification in recommender systems”. In: *CIKM*. 2020.

[159] Chris Marentis. *A Complete Guide To The Essentials Of Post-Hummingbird SEO*. 2014.

[160] Benjamin M Marlin and Richard S Zemel. "Collaborative prediction and ranking with non-random missing data". In: *RecSys*. 2009.

[161] Pawel Matuszyk, João Vinagre, Myra Spiliopoulou, Alípio Mário Jorge, and João Gama. "Forgetting methods for incremental matrix factorization in recommender systems". In: *SIGAPP*. 2015.

[162] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. "On finding local Nash equilibria (and only local Nash equilibria) in zero-sum games". In: *arXiv* (2019).

[163] Julian McAuley and Jure Leskovec. "Hidden factors and hidden topics: Understanding rating dimensions with review text". In: *RecSys*. 2013.

[164] Wes McKinney. "Data structures for statistical computing in Python". In: *Python in Science*. 2010.

[165] Silvia Milano, Mariarosaria Taddeo, and Luciano Floridi. "Recommender systems and their ethical challenges". In: *AI & Society* (2020).

[166] John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. "The Effect of Natural Distribution Shift on Question Answering Models". In: *ICML* (2020).

[167] Smitha Milli, Luca Belli, and Moritz Hardt. "From optimizing engagement to measuring value". In: *FAccT*. 2021.

[168] Smitha Milli, John Miller, Anca D Dragan, and Moritz Hardt. "The social cost of strategic classification". In: *FAccT*. 2019.

[169] Martin Mladenov, Elliot Creager, Omer Ben-Porat, Kevin Swersky, Richard Zemel, and Craig Boutilier. "Optimizing long-term social welfare in recommender systems: A constrained matching approach". In: *ICML*. 2020.

[170] Andriy Mnih and Ruslan Salakhutdinov. "Probabilistic matrix factorization". In: *NeurIPS* (2007).

[171] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv* (2013).

[172] Adrien Mogenet, Tuan Anh Nguyen Pham, Masahiro Kazama, and Jialin Kong. "Predicting online performance of job recommender systems with offline evaluation". In: *RecSys*. 2019.

[173] Remi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. "Safe and Efficient Off-Policy Reinforcement Learning". In: *NeurIPS*. 2016.

[174] Preetam Nandy, Divya Venugopalan, Chun Lo, and Shaunak Chatterjee. "A/B Testing for Recommender Systems in a Two-sided Marketplace". In: *NeurIPS*. 2021.

[175] John F Nash Jr. "Equilibrium points in n-person games". In: *PNAS* (1950).

[176] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. "Exploring the filter bubble: the effect of using recommender systems on content diversity". In: *WWW*. 2014.

[177] Xia Ning and George Karypis. "Slim: Sparse linear methods for top-n recommender systems". In: *ICDM*. 2011.

[178] Weishen Pan, Sen Cui, Hongyi Wen, Kun Chen, Changshui Zhang, and Fei Wang. "Correcting the User Feedback-Loop Bias for Recommendation Systems". In: *arXiv* (2021).

[179] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *NeurIPS*. 2019.

[180] Akshita Patil, Jayesh Pamnani, and Dipti Pawade. "Comparative Study Of Google Search Engine Optimization Algorithms: Panda, Penguin and Hummingbird". In: *I2CT*. 2021.

[181] Judea Pearl. *Causality*. Cambridge University Press, 2009.

[182] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jak Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. "Scikit-learn: Machine Learning in Python". In: *JMLR* (2011).

[183] Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. "Performative prediction". In: *ICML*. 2020.

[184] Ariel D. Procaccia and Moshe Tennenholtz. "Approximate Mechanism Design without Money". In: *TEAC* (2013).

[185] Nimrod Raifer, Fiana Raiber, Moshe Tennenholtz, and Oren Kurland. "Information Retrieval Meets Game Theory: The Ranking Competition Between Documents' Authors". In: *SIGIR*. 2017.

[186] Inioluwa Deborah Raji, Emily M Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. "AI and the everything in the whole wide world benchmark". In: *NeurIPS*. 2021.

[187] Carl Edward Rasmussen and Zoubin Ghahramani. "Infinite Mixtures of Gaussian Process Experts". In: *NeurIPS*. 2001.

[188] Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. "On the characterization of local Nash equilibria in continuous games". In: *TACON* (2016).

[189] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. “Do imagenet classifiers generalize to imagenet?” In: *ICML*. 2019.

[190] Steffen Rendle. “Factorization machines”. In: *ICDM*. 2010.

[191] Steffen Rendle. “Factorization Machines with libFM”. In: *TIST* (2012).

[192] Steffen Rendle, Li Zhang, and Yehuda Koren. “On the difficulty of evaluating baselines: A study on recommender systems”. In: *arXiv* (2019).

[193] Bernhard Rieder and Jeanette Hofmann. “Towards platform observability”. In: *Internet Policy Review* (2020).

[194] Julian A Rodriguez. “LGBTQ Incorporated: YouTube and the Management of Diversity”. In: *Journal of Homosexuality* (2022).

[195] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. “Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising”. In: *arXiv* (2018).

[196] Marco Rossetti, Fabio Stella, and Markus Zanker. “Contrasting offline and online results when evaluating recommendation algorithms”. In: *RecSys*. 2016.

[197] Guido van Rossum and Fred L Drake. *Python 3 Reference Manual*. CreateSpace, 2009.

[198] Mark Rowland, Will Dabney, and Remi Munos. “Adaptive Trade-Offs in Off-Policy Learning”. In: *AISTATS*. 2020.

[199] Paat Rusmevichientong and John N Tsitsiklis. “Linearly parameterized bandits”. In: *Mathematics of Operations Research* (2010).

[200] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. In: *IJCV* (2015).

[201] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. “Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback”. In: *WSDM*. 2020.

[202] Matthew J Salganik, Peter Sheridan Dodds, and Duncan J Watts. “Experimental study of inequality and unpredictability in an artificial cultural market”. In: *science* (2006).

[203] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. “Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering”. In: *ICCIT*. 2002.

[204] Sven Schmit and Carlos Riquelme. “Human Interaction with Recommendation Systems”. In: *AISTATS*. 2018.

[205] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. “Recommendations as treatments: Debiasing learning and evaluation”. In: *ICML*. 2016.

[206] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. “Autorec: Autoencoders meet collaborative filtering”. In: *WWW*. 2015.

[207] Asim Shahzad, Deden Witarsyah Jacob, Nazri Mohd Nawi, Hairulnizam Mahdin, and Marheni Eka Saputri. “The new trend for search engine optimization, tools and techniques”. In: *IJEECS* (2020).

[208] Dougal Shakespeare, Lorenzo Porcaro, Emilia Gómez, and Carlos Castillo. “Exploring Artist Gender Bias in Music Recommendation”. In: *arXiv* (2020).

[209] Amit Sharma, Jake M Hofman, and Duncan J Watts. “Estimating the causal impact of recommendation systems from observational data”. In: *EC*. 2015.

[210] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer”. In: *ICLR*. 2017.

[211] David Simchi-Levi and Yunzong Xu. “Bypassing the monster: A faster and simpler optimal algorithm for contextual bandits under realizability”. In: *SSRN* (2020).

[212] Leo K Simon. “Games with discontinuous payoffs”. In: *The Review of Economic Studies* (1987).

[213] Ashudeep Singh and Thorsten Joachims. “Fairness of exposure in rankings”. In: *SIGKDD*. 2018.

[214] Adish Singla, Hamed Hassani, and Andreas Krause. “Learning to interact with learning agents”. In: *AAAI*. 2018.

[215] Ayan Sinha, David F Gleich, and Karthik Ramani. “Deconvolving feedback loops in recommender systems”. In: *NeurIPS* (2016).

[216] Rashmi Sinha and Kirsten Swearingen. “The role of transparency in recommender systems”. In: *SIGCHI*. 2002.

[217] Nasim Sonboli, Jessie J Smith, Florencia Cabral Berenfus, Robin Burke, and Casey Fiesler. “Fairness and transparency in recommendation: The users’ perspective”. In: *UMAP*. 2021.

[218] Harald Steck. “Embarrassingly shallow autoencoders for sparse data”. In: *WWW*. 2019.

[219] Alexander L Strehl, John Langford, Lihong Li, and Sham M Kakade. “Learning from Logged Implicit Exploration Data”. In: *NeurIPS*. 2010.

[220] Wenlong Sun, Sami Khenissi, Olfa Nasraoui, and Patrick Shafto. “Debiasing the human-recommender system feedback loop in collaborative filtering”. In: *WWW*. 2019.

[221] Adith Swaminathan and Thorsten Joachims. “Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization”. In: *JMLR* (2015).

[222] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. "Off-Policy Evaluation for Slate Recommendation". In: *NeurIPS*. 2017.

[223] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. "Scalable collaborative filtering approaches for large recommender systems". In: *JMLR* (2009).

[224] Diane Tang, Ashish Agarwal, Deirdre O'Brien, and Mike Meyer. "Overlapping experiment infrastructure: More, better, faster experimentation". In: *CIKM*. 2010.

[225] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[226] Philip Thomas and Emma Brunskill. "Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning". In: *ICML*. 2016.

[227] Viet Ha-Thuc, Avishek Dutta, Ren Mao, Matthew Wood, and Yunli Liu. "A counterfactual framework for seller-side a/b testing on marketplaces". In: *SIGIR*. 2020.

[228] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *IROS*. 2012.

[229] João Vinagre, Alípio Mário Jorge, and João Gama. "An overview on the exploitation of time in collaborative filtering". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge discovery* (2015).

[230] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J van der Walt, Matthew Brett, Joshua Wilson, K Jarrod Millman, Nikolay Mayorov, Andrew R J Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E A Quintero, Charles R Harris, Anne M Archibald, Antônio H Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* (2020).

[231] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. "Unifying user-based and item-based collaborative filtering approaches by similarity fusion". In: *SIGIR*. 2006.

[232] Xuezhi Wang, Nithum Thain, Anu Sinha, Flavien Prost, Ed H Chi, Jilin Chen, and Alex Beutel. "Practical Compositional Fairness: Understanding Fairness in Multi-Component Recommender Systems". In: *WSDM*. 2021.

[233] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. "A theoretical analysis of NDCG type ranking measures". In: *COLT*. 2013.

[234] Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. "Causal inference for recommender systems". In: *RecSys*. 2020.

[235] Yixin Wang, Dhanya Sridhar, and David M Blei. "Equal opportunity and affirmative action via counterfactual predictions". In: *arXiv* (2019).

[236] Romain Warlop, Alessandro Lazaric, and Jérémie Mary. "Fighting boredom in recommender systems with linear reinforcement learning". In: *NeurIPS*. 2018.

[237] Michael L Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* (2021).

[238] James Williams. *Stand out of our light: freedom and resistance in the attention economy*. Cambridge University Press, 2018.

[239] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. "Transformers: State-of-the-Art Natural Language Processing". In: *EMNLP*. 2020.

[240] Ya Xu, Nanyu Chen, Addrian Fernandez, Omar Sinno, and Anmol Bhasin. "From infrastructure to culture: A/B testing challenges in large scale social networks". In: *CIKM*. 2015.

[241] Zhi-Wei Xu. "Cloud-sea computing systems: Towards thousand-fold improvement in performance per watt for the coming zettabyte era". In: *Journal of Computer Science and Technology* (2014).

[242] Sirui Yao and Bert Huang. "Beyond parity: Fairness objectives for collaborative filtering". In: *NeurIPS*. 2017.

[243] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. "Sampling-bias-corrected neural modeling for large corpus item recommendations". In: *RecSys*. 2019.

[244] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. "Twenty Years of Mixture of Experts". In: *IEEE Transactions on Neural Networks and Learning Systems* (2012).

[245] Ruohan Zhan, Konstantina Christakopoulou, Ya Le, Jayden Ooi, Martin Mladenov, Alex Beutel, Craig Boutilier, Ed Chi, and Minmin Chen. "Towards Content Provider Aware Recommender Systems: A Simulation Study on the Interplay between User and Provider Utilities". In: *WWW*. 2021.

[246] Longkai Zhang, Houfeng Wang, and Xu Sun. "Coarse-grained candidate generation and fine-grained re-ranking for Chinese abbreviation prediction". In: *EMNLP*. 2014.

[247] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. "Recommending what video to watch next: a multitask ranking system". In: *RecSys*. 2019.

[248] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. "A neural autoregressive approach to collaborative filtering". In: *ICML*. 2016.

[249] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. "Learning tree-based deep model for recommender systems". In: *SIGKDD*. 2018.