Faculteit Wetenschappen
Departement Informatica

Offline Approaches to Recommendation with Online Success

Proefschrift

voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen: Informatica
aan de Universiteit Antwerpen
te verdedigen door

Olivier Jeunen

Promotor: prof. dr. Bart Goethals

Antwerpen, 2021

Offline Approaches to Recommendation with Online Success
Nederlandse titel: *Offline Methoden voor Aanbevelingen met Online Succes*

The important thing is not to stop questioning. Curiosity has its own reason for existence. One cannot help but be in awe when he contemplates the mysteries of eternity, of life, of the marvellous structure of reality. It is enough if one tries merely to comprehend a little of this mystery each day.

— Albert Einstein

Acknowledgements

The past four years have had an immense impact on me, on a professional as well as a personal level. I have learned so much, and maybe the most important insight is the fact that I still know so little. This is humbling, but incredibly exciting at the same time! This thesis is by no means the work of me alone, and there are many people to whom I owe a debt of gratitude for supporting me.

I'll never forget the first words my advisor, prof. dr. Bart Goethals, told me on my first day as a PhD student: *"The most important thing is that you keep on learning new things and enjoying yourself every single day... The rest will follow."* Granted, this wasn't the most action-oriented communication, and I spent many long days wondering what it exactly was that I was supposed to be doing, but days without epiphanies or laughter were rare. Slowly but surely we figured out what kinds of problems I found interesting, and before long we started thinking of solutions. Bart, I want to thank you for introducing me to the world of academic research, for your mentorship, and for always encouraging me to pursue whatever I wanted to.

The Adrem Data Lab has been one of the most stimulating working environments I could imagine, and it owes it all to amazing colleagues: Boris, Daphne, Ewoenam, Floris, Hassaan, Jan, Joeri, Joey, Len, Lien, Koen, Maarten, Marco, Mozhgan, Nikolaj, Robin, Sam, Sandy, Stephen, Toon, Kris and everyone at biomina. From heated political lunch discussions to New Year's receptions, dinners, drinks and journal club sessions: I have enjoyed every second of it, and am grateful to have spent this time with you! Beyond Adrem, I've been lucky to collaborate with and learn from many more people who are much smarter than me, and it undoubtedly improved the quality of my research. Koen and everyone at Froomle, David, Flavian, Dmytro, Alexandre, Martin, Amine, Otmane and everyone at Criteo, Daniel, Alexandra, Abbas and everyone at Facebook, Ciarán, Rishabh, Mounia and everyone at Spotify, ...
It has been a true privilege to get to work with you. You've all enabled the work in this thesis, directly or indirectly, and I'm indebted to every one of you for it.

Another special mention goes to the members of my doctoral jury: Toon Calders, Maarten de Rijke, Floris Geerts, Thorsten Joachims and Mounia Lalmas. Thank you for the effort you invested in thoroughly reviewing this thesis, for your insightful feedback and advice. I am honoured to have you all as part of my thesis committee.

Aside from all of this professional luck, I'm blessed with phenomenal friends and family. My parents, grandparents and entire family have always encouraged me to pursue my dreams, and I'll be eternally grateful for it. To my parents: thank you for everything. Your pride motivates me tremendously, even if I know it's unconditional. To my sister Julie and her husband Sam: the fact that I'm unsure whether you belong in the "*Family*" or "*Best Friends*" section, should speak for itself. I'm both grateful

for and amazed at your ability to let me blabber on and on about work and research. Thank you for always being there to listen to me.

The very first person to set me off on this scientific journey has to be Nonkel Loe. You've instilled a sense of curiosity in me that has shaped me for years to come, still. I will never forget Wednesday afternoons filled with quite literal *"huis-tuin-en-keuken"* scientific experiments that took over your entire home, or the passion and calm with which you explained to me the most fascinating insights. Nine-year-old me was too embarrassed to admit I never *fully* understood what you were talking about, but I now understand that asking the right questions is sometimes more important than knowing the answers. Thank you for everything.

Pursuing a doctorate is a surefire way to blur the boundaries between personal and professional life, and I more often than not brought my research and work home with me. To all of my friends: Simon, Jony, Maarten, Bram, Viktor, Danny, Eli, the entire "*Choc-groep*" and everyone else who has ever endured my relentless incoherent rambling, shared a drink for my paper acceptances or shared a drink for my paper rejections – you are all amazing people and I definitely could not have done this without you. Merci!

Last but certainly not least: I want to thank the love of my life, Tine. I'm under no illusion that sharing your life with me would be straightforward, but somehow you do manage to make it look effortless. Dealing with unpredictable working hours, many conference trips overseas, and months of foreign research visits before asking you to move to another country with me is decisively no small ask, and I am infinitely grateful for your love and support. No challenge is too great when you are by my side.

Thank you all!

Olivier Jeunen
Antwerpen, 2021

Contents

Publications

Journals

- **Olivier Jeunen**, Jan Van Balen and Bart Goethals. Embarrassingly Shallow Auto-Encoders for Dynamic Collaborative Filtering. Under revision for *User Modeling and User-Adapted Interaction (UMUAI) Journal Special Issue on Dynamic Recommender Systems & User Models (DyRSUM)*, 36 pages, 2021.

Conferences

- **Olivier Jeunen**. Revisiting Offline Evaluation for Implicit-Feedback Recommender Systems. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*, pages 596–600, 2019.
- **Olivier Jeunen**, Koen Verstrepen and Bart Goethals. Efficient Similarity Comptuation for Collaborative Filtering in Dynamic Environments. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*, pages 251–259, 2019.
- **Olivier Jeunen**, David Rohde, Flavian Vasile and Martin Bompaire. Joint Policy-Value Learning for Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20)*, pages 1223–1233, 2020.
- **Olivier Jeunen**, Jan Van Balen and Bart Goethals. Closed-Form Models for Collaborative Filtering with Side-Information. In *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys '20)*, pages 651–656, 2020.
- **Olivier Jeunen** and Bart Goethals. Pessimistic Reward Modelling for Off-Policy Learning in Recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems (RecSys '21)*, 17 pages, 2021.
- **Olivier Jeunen** and Bart Goethals. Top-K Contextual Bandits with Equity of Exposure. In *Proceedings of the 15th ACM Conference on Recommender Systems (RecSys '21)*, 16 pages, 2021.

Workshops, Demos & Tutorials

- **Olivier Jeunen**, Koen Verstrepen and Bart Goethals. Fair Offline Evaluation Methodologies for Implicit-Feedback Recommender Systems with MNAR Data. In *Proceedings of the ACM RecSys Workshop on Offline Evaluation for Recommender Systems (REVEAL '18)*, 9 pages, 2018.

- **Olivier Jeunen** and Bart Goethals. Predicting Sequential User Behaviour with Session-based Recurrent Neural Networks. In *Proceedings of the ACM WSDM Cup Workshop (WSDM Cup '19)*, 4 pages, 2019.

- Sandy Moens, **Olivier Jeunen** and Bart Goethals. Interactive Evaluation of Recommender Systems with SNIPER: An Episode Mining Approach. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*, pages 538–539, 2019.

- **Olivier Jeunen**, David Rohde and Flavian Vasile. On the Value of Bandit Feedback for Offline Recommender System Evaluation. In *Proceedings of the ACM RecSys Workshop on Reinforcement and Robust Estimators for Recommendation (REVEAL '19)*, 3 pages, 2019.

- **Olivier Jeunen**, Dmytro Mykhaylov, David Rohde, Flavian Vasile, Alexandre Gilotte and Martin Bompaire. Learning from Bandit Feedback: An Overview of the State-of-the-art In *Proceedings of the ACM RecSys Workshop on Reinforcement and Robust Estimators for Recommendation (REVEAL '19)*, 3 pages, 2019.

- Dmytro Mykhaylov, David Rohde, Flavian Vasile, Martin Bompaire and **Olivier Jeunen**. Three Methods for Training on Bandit Feedback. In *Proceedings of the NeurIPS Workshop on Machine Learning and Causal Inference for Improved Decision Making (CausalML '19)*, 5 pages, 2019.

- Flavian Vasile, David Rohde, **Olivier Jeunen** and Amine Benhalloum. A Gentle Introduction to Recommendation as Counterfactual Policy Learning. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '20)*, pages 392–393, 2020.

- **Olivier Jeunen** and Bart Goethals. An Empirical Evaluation of Doubly Robust Learning for Recommendation. In *Proceedings of the ACM RecSys Workshop on Bandit and Reinforcement Learning from User Interactions (REVEAL '20)*, 3 pages, 2020.

- Flavian Vasile, David Rohde, **Olivier Jeunen**, Amine Benhalloum and Otmane Sakhi. Recommender Systems through the Lens of Decision Theory. In *Proceedings of the 30th World Wide Web Conference (WWW '21)*, 4 pages, 2021.

List of Figures

List of Tables

CHAPTER 1

Introduction

Over three decades ago, the "Information Revolution" began [175]. Sparked by exponential advances in computing technology and fuelled by the rise of the internet, access to information on the web quickly became widespread. In addition to that, the rate at which new information in the form of data is being generated has now grown to be higher than it has ever been before. Today, backed up by massive advances in machine learning, it is safe to say that the revolution is still going strong.

Access to a wealth of information is one thing, but being able to effectively and efficiently explore or filter information quickly becomes crucial. And so, not long after the start of the revolution, the problem of "Information Overload" rapidly gained in importance and research interest [18]. Indeed, what good is an encyclopedia if it is entirely unstructured and missing its typical back-of-the-book index?

The field of Information Retrieval (IR) aims to solve these problems, with web search engines as their most visible embodiment that we now all use on a daily basis. These systems take in queries like "presidential elections 2020", "is a Ph.D. worth it?" or "can doctoral juries be bribed?" and produce an ordering of web pages as the result to that query. Search engines are by no means restricted to text, and diverging applications such as multimedia retrieval to search audio, image or video catalogues spawned over the years as well. Nowadays, all "Big Tech" companies have some search functionality on their platform that is often central to their business. Whether we are browsing retail products on Amazon, musical artists on Spotify or Apple Music, TV series on Netflix, friends' profiles on Facebook or websites via Google or Bing, IR applications are ubiquitous and touch upon many people's daily lives.

An interesting question to ask is then: what happens when we don't actually have an explicit query? What if I'm just browsing a retail store? What happens when I do not know which musical artist I want to listen to or which movie I want to watch? This is where the field of "Recommender Systems" comes into play.

1.1 A Brief History of Recommender Systems

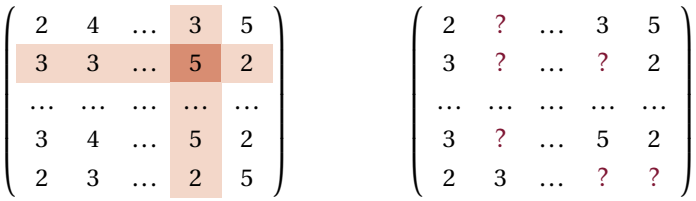
It may be unfair to introduce the field by summing up its present-day industrial applications, as we did above. Although they are indeed widespread and industrial recommenders are arguably the ones we interact with so often, the research field was originally not designed with business cases or profiteering in mind. As Resnick and Varian so eloquently put it [172]:

"It is often necessary to make choices without sufficient personal experience of the alternatives. In everyday life, we rely on recommendations from other people either by word of mouth, recommendation letters, movie and book reviews printed in news papers, [...]. Recommender systems assist and augment this natural social process."

This focus on the utility of the recommendation to the recipient is what sets it aside from related sub-fields like computational advertising [137], although many algorithmic approaches to either of these problems can be applied interchangeably.

One of the first recommendation systems was called "Tapestry", and it was motivated by the information overload that stemmed from an increasing use of electronic mail at the time. Modelling the recommendation algorithm to mimick the above-described social process, the authors coined the term "collaborative filtering" to denote that information distilled from *other* users' interactions would be used to figure out what we should recommend to *you* [50]. These days, collaborative filtering is still the most widely adopted paradigm behind modern approaches to recommendation [39].

Not much later, the "GroupLens" architecture was proposed, which framed the task of recommendation as that of predicting the *rating* that a user would give to an item [173]. If we have a dataset consisting of interval-scale ratings from users to items, we can represent it in a user-item rating matrix $\boldsymbol{R}$ where the value at $\boldsymbol{R}_{u,i}$ holds the observed rating for user u and item i. Figure 1.1(a) visualises this, with an example rating matrix $\boldsymbol{R}$ that highlights a 5 star rating for the item at index $n-1$, from user number 2. Generally speaking, we do not have access to a full user-item matrix. When the size of the item catalogue grows, this becomes increasingly cumbersome to obtain. Additionally, the goal of the recommendation algorithm is to figure out which items a certain user might rate highly *without* explicitly eliciting this information from the user. As such, the task at hand is to predict the missing ratings from an incomplete user-item matrix, as shown in Figure 1.1(b). A rating prediction model generates predicted ratings for ever user-item pair, and its output can be seen as a reconstruction of the rating matrix $\tilde{\boldsymbol{R}} \approx \boldsymbol{R}$. Competing models are then often evaluated on the Root Mean Squared Error (RMSE) between the true and approximated ratings for some held-out test set [199]. Because the reconstructed rating matrix $\tilde{\boldsymbol{R}}$ holds predictions for all the items for which we do not know the user's rating, we can use these to obtain a sorted recommendation list to show to the user. The assumption here is that higher predicted ratings imply improved recommendation quality. As this modelling approach requires us to explicitly prompt users to provide ratings on an interval scale, it is often referred to as the "explicit feedback" setting. This framework has been hugely successful, and dominated the field for many years. Perhaps the most famous example is that of the



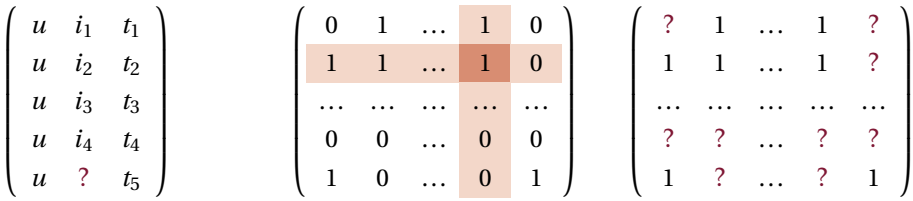
(a) Full user-item matrix. (b) Missing ratings.

Figure 1.1: An example of rating prediction. (a) We build a user-item matrix $\boldsymbol{R}$ that holds the rating for user u and item i at $\boldsymbol{R}_{u,i}$. (b) Typically, some ratings are missing, and the recommendation task is defined as the prediction of these missing ratings.

Netflix Prize competition, where the eponymous video streaming service pledged a USD 1 000 000 prize to the team that could achieve a minimal RMSE at the end of the competition [13, 11].

Nevertheless, obtaining a suitable dataset with explicit ratings can be a hurdle, as repetitively prompting users to rate items can be detrimental to the user experience. Furthermore, studies have shown that ratings obtained through such online rating systems do not always reflect users' true evaluations of the items at hand [250].

These observations combined with the reality that users' interactions on online platforms were being logged and stored, gave rise to the "implicit feedback" setting which then took over [158, 65]. By exploiting the information that is inherent to a user *interacting* with an item, we can avoid the need for explicit ratings. Indeed, we can reasonably assume that a user will mostly view retail product pages of items they are interested in, or movies and series that they enjoy. The implicit-feedback recommendation setting quickly gained in popularity, and is now far more widespread than its explicit counterpart. Even Netflix has moved on from their 5-star ratings to a simpler feedback mechanism, predominantly focusing on signal acquired from interaction data [51]. When purely focusing on deduplicated "views", "clicks", "purchases" et cetera, this setting is referred to as *binary, positive-only* [221]. This setting is visualised in Figure 1.2 where we build a user-item matrix based on a sequence of user-item interactions, and frame the task of recommendation as predicting missing interactions. Note that there are several differences between the item and rating prediction tasks, most notably that there's an absence of explicit negative information in the former (hence, positive-only). When a user-item interaction is missing from the dataset, we often do not know whether this means that the user is simply unaware of the item, or whether it is irrelevant to the user. This inability to distinguish between "missing" and "negative" feedback has motivated several modelling approaches such as Bayesian Personalised Ranking [169] and Weighted Regularised Matrix Factorisation [65, 158]. Furthermore, the data is typically Missing-Not-At-Random (MNAR), and this characteristic remains important to take into account when learning or evaluating recommendation models [197]. In contrast with the RMSE metric often used in rating prediction, the item prediction setup is typically evaluated with IR-inspired metrics such as precision, recall or Normalised Discounted Cumulative Gain (NDCG) [199, 20, 215].



(a) Interactions for u. (b) Full user-item matrix. (c) Missing interactions.

Figure 1.2: An example of item prediction. (a) We observe a sequence of items a user u has interacted with, and wish to predict which item completes the sequence. (b) We build a user-item matrix $\boldsymbol{X}$ that holds binarised interactions for user u and item i at $\boldsymbol{X}_{u,i}$. (c) Typically, many interactions are missing, and the recommendation task is defined as the prediction of which user-item interactions actually occur in the data.

Many approaches to recommendation from implicit-feedback data find their roots in the explicit-feedback setting. In what follows, we give a brief introduction to the most common families of approaches, and the rationale behind them.

Latent Factor Models

Latent factor models assume a low-rank generating process behind the user-item interaction data, and they explicitly model this by optimising low-rank user- and item-factor matrices $\boldsymbol{U} \in \mathbb{R}^{m \times k}, \boldsymbol{V} \in \mathbb{R}^{k \times n}$, if we assume to have m unique users, n unique items, and k latent factors [98]. Figure 1.3 visualises this type of approach. The intuitive idea is that the learnt latent factors will represent common concepts such as genres, and the values in $\boldsymbol{U}$ and $\boldsymbol{V}$ encode users' and items' affinities to those genres respectively. The recommendation score for a user u and an item i, is then computed as the dot-product between their low-rank embeddings, as shown by Equation 1.1.

$$\boldsymbol{X}_{u,i} \approx \boldsymbol{U}_{u,\cdot} \cdot \boldsymbol{V}_{\cdot,i} = \sum_{k} \boldsymbol{U}_{u,k} \cdot \boldsymbol{V}_{k,i} \tag{1.1}$$

Lower values of k restrict the model's capacity and might result in underfitting, whereas for larger values of k we can eventually exactly reconstruct the user-item matrix $\boldsymbol{X}$. Note that the latter is equally undesirable, as a model that correctly identifies all zeroes in Figure 1.2(b) does not provide any actionable information for personalisation. For these reasons, a regularisation term is often added to the optimisation problem at hand. Earlier methods learn the factorised matrices via singular value decomposition, and many extensions to this have been proposed over the years [181, 31, 40]. Latent factor models are one of the oldest classes around, and are often still surprisingly effective when compared to more complex alternatives [170, 171].

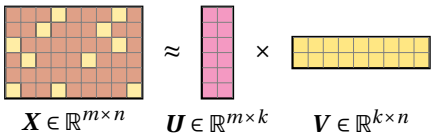


Figure 1.3: Latent factor models explicitly learn common latent factors that describe users and items, in an attempt to reconstruct the user-item matrix via a low-rank bottleneck. The transitivity that comes from the low-rank assumption helps to generalise and predict affinity scores between users and unseen items.

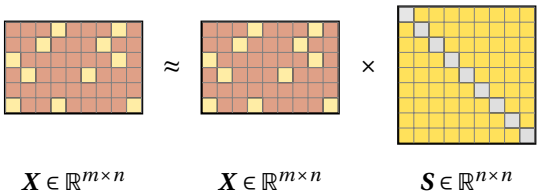


Figure 1.4: Item-based recommendation models learn an item-item similarity matrix to reconstruct values in the user-item matrix based on the values in other columns. The diagonal of the matrix can be restricted towards zero, as to avoid the trivial solution where $\boldsymbol{S}$ is the identity matrix.

Item-Based Models

Item-based models are – generally speaking – a type of full-rank model. The rationale is that we can predict the affinity from a user towards an item by considering the other items a user has already interacted with in the past. We effectively learn an item-item similarity matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, and use these similarities in a weighted sum with items in the users' history to predict relevance. This is shown in Figure 1.4 and Equation 1.2.

$$\boldsymbol{X}_{u,i} \approx \boldsymbol{X}_{u,\cdot} \cdot \boldsymbol{S}_{\cdot,i} = \sum_{j} \boldsymbol{X}_{u,j} \cdot \boldsymbol{S}_{j,i} \tag{1.2}$$

When the item-item matrix is constructed with an analytically computable similarity measure between the high-dimensional yet sparse columns of the user-item matrix $\boldsymbol{X}$ (such as cosine similarity, Jaccard index or conditional probabilities), this type of approach is often called "nearest neighbours"-based [34]. Although technically correct, this is not the most descriptive terminology. Indeed, latent factor models are also "nearest neighbours"-based, albeit in a projected low-dimensional space instead of the one represented directly by the user-item matrix $\boldsymbol{X}$. Many extensions to the item-based paradigm have been proposed [147, 30], and posing the computation of $\boldsymbol{S}$ as a simple linear optimisation problem is often sufficient to obtain highly competitive results [200]. When the item-item similarities are learnt directly, this model is referred to as full-rank. It is however not uncommon to model $\boldsymbol{S}$ itself as the factorisation of two lower-dimensional matrices [90].

Sequence-Aware Models and Beyond

The two families of approaches introduced above operate on the bare minimum of information: the presence of an interaction between user u and item i. For many practical use-cases, more information will be available as well. Consider the set-up we introduced in Figure 1.2, by generating a binarised user-item matrix from the sequence of interactions, we effectively threw away all temporal information. Naturally, this information can encode many implicit sequential relations among items, and advanced methods can exploit these relations for enhanced recommendation accuracy [167]. Many more examples can be thought of, where we additionally have information about item content, contextual information about item consumption, browsing sessions et cetera [2, 74, 124].

Contextual Bandits

The "contextual bandit" framework is a general machine learning paradigm where an *agent* observes a certain *context* and can perform an *action* [102]. We can replace these abstract concepts with a recommendation system that observes a user visiting the system, and then chooses which items to recommend to this user. As such, real-world recommendation systems can be referred to as contextual bandits. Nevertheless, general bandit algorithms and recommendation approaches tend to differ quite a bit. Bandit algorithms were introduced as a general framework for decision-making under uncertainty, and they typically update their model based on the outcome of their actions [23, 37]. Traditional approaches to recommendation, in contrast, focus on predicting the occurrence of interactions between users and items, and do not take into account the outcomes of the recommendations themselves. Several recent works have tried to bridge this gap [112, 114, 136, 140], but these fields have largely evolved independently of one another.

The recommender systems research space is broad and vast, and it is by no means our ambition to provide an extensive survey in the introductory Chapter to this thesis. Nevertheless, we have now introduced the reader to the tools necessary to contextualise the research contributions that make up the rest of this manuscript. The contributions presented in this thesis can be broadly categorised into three disjoint parts. In what follows we briefly introduce these sub-fields, and provide an outline of the chapters that follow. The common theme uniting our research contributions, is that we aim to bridge the gap between the (often static) recommendation problem that is prevalent in academic research, and the more complex dynamic systems that are typically faced by practitioners in industry.

1.2 Efficient and Incremental Computations

Recommendation models that are deployed into the real world are far from static. They need to be updated periodically as new data comes in to account for shifting user preferences and item popularities. This incurs a computational cost that we would naturally like to reduce as much as possible. The first goal is to reduce the initial computation time for the recommendation model to a minimum, that is, make model training *efficient*. Second, we wish to reduce the time needed for

subsequent updates to the model. By targeting those model parameters that need to be updated instead of blindly recomputing the entire model, we obtain models that can be updated *incrementally*.

- In **Chapter 2**, we present *Dynamic Index*: a novel algorithm for efficient and exact similarity computation between sparse, high-dimensional vectors. The algorithm is tailored towards the implicit-feedback data setting that widely occurs in recommender systems, learns incrementally, and is easily parallellisable. As such, it is naturally suited for item-based collaborative filtering approaches that are deployed in dynamic environments, where updates need to be performed in real-time. We additionally explore the concept of item *recommendability*, and show that our method can exploit this naturally occurring concept efficiently and effectively.

- In **Chapter 3**, we present *Dynamic* EASE^R (DYN-EASE^R): a novel algorithm for incrementally updating ridge regression models as new data arrives, making use of the Dynamic Index algorithm to incrementally update the item co-occurrence matrix, and subsequently leverage well-known identities from linear algebra to incrementally compute its inverse. Our exact updating algorithm significantly improves the efficiency of the state-of-the-art recommendation approach EASE^R [200]. Moreover, we present approximate variants of DYN-EASE^R, providing a tuneable trade-off between the exactness of the model and the efficiency with which it can be computed. Approximate DYN-EASE^R can further improve the recommendation accuracy of its exact counterpart, by exploiting transitivity relations that arise in low-rank representations when the data is sparse.

1.3 Offline Evaluation Methodologies

Machine learning models are typically evaluated on offline datasets, where a random split divides the data into a *train* and *test* set, and a model that has learned from the samples in the training set is evaluated on its ability to predict (labels for) the samples in the test set [96]. This well-validated and established paradigm has found widespread adoption in the recommender systems community as well, and is often used to report empirical gains in recommendation accuracy in the literature. Nevertheless, such evaluation procedures generally conflate improved *prediction* capabilities with improved *recommendations*, due to the assumption made by the item prediction paradigm that they are equivalent.

In contrast, recommendation algorithms deployed in the real world have a distinct advantage: they get to actually show recommendations to users and observe whether the user interacts with them. Metrics based on click-through-rate, page dwell time, session length, subscription renewal, et cetera are then often used as proxies for user satisfaction, which still remains the paramount and overarching target. The downside here is that online experiments are generally much more expensive than offline procedures, and are often out of reach for academic researchers.

Current offline evaluation methodologies are notoriously uncorrelated with online success metrics, and the reasons why are often only superficially understood. Understanding *when* and *why* offline evaluation results diverge from online success

metrics can be an important step to ensure that empirical gains in the literature translate to better recommendations being presented to users.

- In **Chapter 4**, we present results from an empirical study that compares various offline evaluation methodologies on real-world data collected on a retail website with a deployed recommendation algorithm. We present a novel evaluation procedure called *Sliding Window Evaluation* (SW-EVAL), which much more tightly adheres to the use-cases deployed recommendation algorithms encounter than random train-test splits. We show how taking the sequential nature of user-item interactions into account provides much more reliable offline estimates of performance, and show how alternative methods provide conflicting results. Finally, we show that the selection bias from the deployed recommendation algorithm significantly biases results towards recommendations that were shown by the logging policy, and underline that Missing-Not-At-Randomness (MNAR) in the data is crucial to model.

- In **Chapter 5**, we review the differences and commonalities between online and offline evaluation strategies, and present a research agenda to bring them together. Online evaluation methods are effective but inefficient, and offline alternatives are efficient yet ineffective. We highlight the importance of temporal evaluation to model sequentiality in the data, off-policy evaluation to de-bias the item selection bias that occurs from deployed recommenders, and argue in favour of exploiting information related to impressions and user inaction in offline metrics, to distinguish *missing* interactions from *negative* feedback.

1.4 Effective Learning from Bandit Feedback

The assumption behind the item prediction paradigm is that accurately predicting which interactions between users and items will occur *in the absence of the recommender*, makes for a good recommendation algorithm. This is rooted in the many observational datasets containing *organic* user-item interactions that are widely available and adopted by the research community, and has proven its worth for many years. The online paradigm, in contrast, does not focus on merely predicting which items a user will interact with. Crucially, there is an interactive component where we *show* a recommendation to the user, and we observe any subsequent interactions. This interventionist view departs from many classical approaches to recommendation, but is a fundamental component of any practical recommendation use-case.

If metrics related to online interventions are what we wish to optimise, it makes sense to use datasets containing logged recommendations and their outcomes. Such data is often called *bandit* feedback, as we only observe the outcome for actions taken by the contextual bandit that was deployed at data collection time. This contextual bandit is referred to as the *logging policy*, and it will often be biased towards performing actions that it deems successful (i.e. showing recommendations that are likely to lead to user satisfaction). This arm selection bias can severely skew the data if not taken into account properly, making effective learning from bandit feedback a highly non-trivial task.

Datasets consisting of bandit feedback are abundant in industrial recommendation applications. It is exactly the data that is being used to evaluate the online performance of recommendation algorithms, and it comes naturally that this data should be used to provide meaningful offline estimates of online performance. Nevertheless, these datasets are rarely publicly available, and counterfactual evaluation procedures using them often lack the statistical power to conclusively reject hypotheses [49]. There is a fundamental divide here between the bulk of academic research and the recommendation use-cases that arise in practice.

- In **Chapter 6**, we present an overview of the state-of-the-art in learning from bandit feedback, with an eye on the recommendation task. We present results from the first broad empirical study of counterfactual learning methods for recommendation, using reproducible simulation environments. We highlight how existing methods tend to fail due to stochastic and sparse rewards, and propose the use of a logarithmic lower bound on the traditional importance sampling estimator to mitigate these issues. Moreover, we show that the two contrasting families of *value*- and *policy*-based methods can be modelled with an identical parameterisation, which allows for a model that jointly optimises a hybrid objective. We show that this *Dual Bandit* approach achieves state-of-the-art performance in a wide range of scenarios, and that its gains over competing methods are most outspoken in the realistic and complex settings.
- In **Chapter 7**, we focus on improving the performance of reward models that are learned from bandit feedback. We present a general-purpose framework for pessimistic decision making under model uncertainty, and show how it can be used to obtain state-of-the-art performance in off-policy recommendation tasks. Our decision-making approach exploits Bayesian uncertainty estimates to know what the reward model does not know, and takes decisions with a maximal worst-case outcome. This form of principled scepticism leads to a significant and robust increase in online recommendation performance. We additionally show how our method limits post-decision disappointment, implying that it can also be used to accurately forecast model performance by practitioners.

Chapter 8 summarises the research contributions presented throughout this thesis, and those presented in additional related work by the author. We conclude by presenting a scope for future research to further bridge the gap between academia and industry in recommendation research.

Part I

Efficient and Incremental Computations

If I have seen further, it is by standing on the shoulders of Giants.

— Isaac Newton

CHAPTER 2

Efficient Similarity Computation for Collaborative Filtering in Dynamic Environments

The problem of computing all pairwise similarities in a large collection of vectors is a well-known and common data mining task. As the number and dimensionality of these vectors keeps increasing, however, currently existing approaches are often unable to meet the strict efficiency requirements imposed by the environments they need to perform in. Real-time neighbourhood-based collaborative filtering (CF) is one example of such an environment in which performance is critical.

In this work, we present a novel algorithm for efficient and exact similarity computation between sparse, high-dimensional vectors. Our approach exploits the sparsity that is inherent to implicit feedback datastreams, entailing significant gains compared to other methods. Furthermore, as our model learns incrementally, it is naturally suited for dynamic real-time CF environments. We propose a MapReduce-inspired parallellisation procedure along with our method, and show how even more speed-up can be achieved. Additionally, in many real-world systems, many items are actually not recommendable *at any given time, due to recency, stock, seasonality, or enforced business rules. We exploit this fact to further improve the computational efficiency of our approach. Experimental evaluation on both real-world and publicly available datasets shows that our approach scales up to millions of processed user-item interactions per second, and well advances the state-of-the-art.*[1]

[1] This chapter is based on work published in the *Proceedings of the 2019 ACM RecSys Conference* as "Efficient Similarity Computation for Collaborative Filtering in Dynamic Environments" by Olivier Jeunen, Koen Verstrepen and Bart Goethals [83].

2.1 Introduction

Many important recommender system use-cases are highly dynamic in nature: news, movie, music or retail recommenders all want to incorporate new behaviour into their models as quickly as possible. With new user-item interactions arriving at high rates, the need for dynamic models that can efficiently handle incremental updates in approximately real time becomes more and more apparent [92]. In the context of highly dynamic environments where items have limited lifetimes, this issue becomes even more pressing. News websites typically only want to recommend recent articles, and interactions with newly written articles need to be incorporated into the model as quickly as possible. Auction websites frequently deal with items that are only available for a few days and face the same concerns. Many more examples exist. Traditional Collaborative Filtering (CF) approaches fall short in this setting, as frequent model updates often become too time consuming. Typically, the entire CF model will be retrained at certain fixed points in time, after which the updated model is then deployed. For highly dynamic use-cases, the time between subsequent model updates should ideally be kept minimal, in order to allow information from new incoming user-item interactions to be incorporated into the recommendation process as soon as possible. However, as more and more data arrives, the iterative recomputation of the entire model becomes more and more costly as well, putting a hard upper limit on the frequency with which model updates can be performed. We see a fundamental divide here, and such a trade-off is unacceptable for many present-day applications. A clear need arises for CF models that can instantaneously process new transactions and incorporate them into the model in an incremental manner, while avoiding the periodical re-processing of old data.

In this paper, we present a novel exact algorithm to tackle the problem of efficient similarity computation for high-dimensional and fast changing sparse implicit feedback data streams. Such algorithms are at the basis of nearest-neighbour-based CF techniques, which have recently been shown to attain competitive results with more advanced state-of-the-art approaches, such as recurrent neural networks [73]. On top of this, they provide naturally explainable recommendations [220]. As a consequence, they remain a popular approach to recommendation. Currently existing alternative methods for efficient similarity computation often make use of approximations, sacrificing accuracy for efficiency [144, 66, 207]. Our algorithm, on the other hand, computes all *exact* item-item similarities. The algorithm learns incrementally, making it naturally suitable for real-time CF environments. We exploit the data's sparsity to avoid unnecessary iterative computations and propose the use of an inverted index to quickly identify affected pairs of items when updates arrive. Our approach is presented in a MapReduce-inspired formulation, demonstrating its scalability.

As the number of users and items in present-day real world systems quickly scales up to hundreds of thousands and millions, it often becomes undesirable or unnecessary to keep updated recommendation scores for all catalogued items in the database. Again, in the case of a news website recommendation engine, scores for old articles will be irrelevant as only *recent* items are allowed to be recommended. Or, in the case of a retail environment: recommending items that are currently out of stock is to be avoided. Media recommenders that deal with expiring licenses encounter the same issues. As such, for many different use-cases, the set of *recom-*

mendable items at a given time is a much smaller subset of the full item collection. This imbalance is exploited by our algorithm, as we compute and maintain recommendation scores only for those items that are recommendable. We show that incorporating this natural aspect into our algorithm has dramatic effects on system throughput.

To summarise, the main contributions of this paper are:

1. We introduce a novel algorithm, called "Dynamic Index", for efficiently computing all pairwise similarities in a collection of sparse high-dimensional vectors, which are typical for recommender systems.
2. Our approach learns incrementally, making it suitable for real-time environments.
3. We further exploit non-recommendable items to improve the computational efficiency of our method.
4. By presenting our algorithm in a MapReduce-inspired formulation, it is easily parallellised and scalable.
5. Experimental results on real-world data demonstrate the efficiency and performance of our methodology.[2]

2.2 Related Work

Nearest-neighbour or similarity join processing is not a new problem, and has been thoroughly investigated in the last 15 to 20 years. Most recent trends for speeding up computation tend to either focus on approximate solutions [144], distributed algorithms [248, 246] or incremental approaches [237, 243]. The first notable work in the latter area is the kNNJoin$^+$ algorithm [243], which uses the *iDistance* similarity measure [241, 242] and a *Sphere-tree* index to efficiently reduce the high-dimensional search to a single dimension. However, when updating two points i and j, the distance between these two points still needs to be re-evaluated in the high-dimensional space before the index can be updated to enable efficient nearest neighbour search. Moreover, this work was aimed at a dimensionality ranging from 20 to 50 and only 100 000 data points, whereas we focus on much larger but very sparse datasets consisting of millions of dimensions, as is typical for recommender systems.

Yang et al. propose a method called *HDR-tree* for incrementally updating nearest neighbour joins in the context of recommender systems [237], exploiting the distance-preserving properties of Principal Component Analysis (PCA). Their algorithm focuses on content-based filtering with a strict window size of recent items that they consider for recommendations, whereas our algorithm focuses on collaborative filtering with a much more flexible set of recommendable items that can change over time. Furthermore, they require a fixed set of users, which is too restrictive for the more typical setting we consider. In the context of CF algorithms for streaming scenarios, multiple online learning approaches for matrix factorization, learning-to-rank and neural network models have been presented as well [168, 60, 232, 231].

[2] Code available at: `https://github.com/olivierjeunen/dynamicindex`

Several incremental or online learning algorithms specifically for nearest-neighbour-based CF models have also been published in recent years. Liu et al. propose an incremental learning algorithm that includes temporal information in their novel similarity measure to tackle concept drift in users' preferences over time [119]. The work of Luo et al. focuses on reducing model storage complexity and increasing rating prediction accuracy by incrementally learning biases on top of similarities [125]. TencentRec is a framework implementing several well-known recommendation algorithms in a streaming environment to provide real-time recommendations [66]. Their variant prunes probable dissimilar items, leading to an approximate solution instead of an exact one. Another neighborhood-based approach is proposed by Subbian et al., where a probabilistic data structure is used to approximate item-item similarities and provide recommendations in a real-time manner [207]. Sreepada and Patra present a novel similarity measure that is incrementally learned more easily than other common similarity measures, called *item tendency* [196].

However, most of the above-mentioned methods [119, 125, 207, 196] rely on explicit-feedback data, which is vastly different than the implicit-feedback data use-case we tackle with this work in terms of similarity measure computation as well as general aspects of the dataset. Moreover, several of these methods [66, 207] use approximations to speed up computation time, at the cost of similarity- (and as a consequence recommendation-) accuracy. In this work, we focus on the task of *exact* nearest-neighbour and similarity computations from implicit-feedback data, without the use of any approximations or need of explicit rating data. In addition, with our approach, non-relevant items or users are not considered at computation time, which allows us to work directly on the high-dimensional space, as we can take maximal advantage of the highly sparse nature of the data. Finally, as our algorithm only needs a simple inverted index to efficiently identify affected pairs of items when updates arrive, we can formulate it in accordance with the MapReduce paradigm, ensuring scalability through parallel processing [33].

2.3 Background

Preliminaries

Let U be a set of m users and I a set of n items. Our work focuses on transactional data with implicit feedback. More specifically: we work with a set of user-item pairs $(u,i) \in U \times I$ denoting that user u has *consumed* item i, be it in the form of a product purchase, a movie streaming, a click on a news article or otherwise. We call such preference expressions *pageviews*, and represent them as a tuple (u,i,t_c), where t_c denotes the consumption time. The set of all pageviews up to, but not including time t is denoted by $\mathscr{P}_t$. We can represent these pageviews in the form of a sparse user-item matrix $\mathbf{P}_t \in \{0,1\}^{m \times n}$ for m unique users and n unique items. We omit the timestamp t when it is clear from context. Rows in this matrix are users represented by the items they have consumed, and vice versa for columns: $\mathbf{P}_{u,i} = 1$ if and only if user u has consumed item i and $\mathbf{P}_{u,i} = 0$ otherwise. When we represent an item i by the i-th column-vector of the matrix $\mathbf{P}_t$, we denote it as $\boldsymbol{i}$. The set of users that have consumed a specific item $i \in I$ is denoted as U_i. Vice versa, the set of items that a certain user $u \in U$ has consumed is denoted as I_u.

Algorithm 1 Naive Baseline

Input: A set of pageviews $|\mathscr{P}_t|$.
Output: An inverted index from items to users $\mathscr{K}$, a matrix of item similarities $\mathbf{S}$.

1: $\mathscr{K} \leftarrow \emptyset$, $\mathbf{S} \leftarrow \mathbf{I}$
2: **for** $(u, i, t_c) \in \mathscr{P}_t$ **do**
3: $\quad \mathscr{K}[i] = \mathscr{K}[i] \cup \{u\}$
4: **for** $i \in \mathscr{K}$ **do**
5: $\quad$ **for** $j \in \mathscr{K}$ **do**
6: $\quad\quad$ **if** $i < j$ **then**
7: $\quad\quad\quad \mathbf{S}_{i,j} \leftarrow \frac{|\mathscr{K}[i] \cap \mathscr{K}[j]|}{\sqrt{|\mathscr{K}[i]|} \cdot \sqrt{|\mathscr{K}[j]|}}$
8: **return** $\mathscr{K}, \mathbf{S}$

Between items $i, j \in I$, similarity is expressed as the well-known cosine similarity: $\cos(\boldsymbol{i}, \boldsymbol{j})$. The goal at hand is to efficiently and incrementally compute and store the similarity $\cos(\boldsymbol{i}, \boldsymbol{j})$ for every such item-pair. In a worst-case scenario, this would incur a memory overhead of $\frac{n \cdot (n-1)}{2}$ item similarities that need to be stored. In many real world datasets, the user-item matrix $\mathbf{P}$ is extremely sparse. For many implicit-feedback datasets, this can lead to sparsity in the item co-occurrence matrix $\mathbf{M}$. We denote the sparsity of any matrix by the function $\sigma(\cdot)$. Partially to alleviate spatial complexity issues, and partially to exploit this inherent sparseness and avoid unnecessary iterative computations on zero-values, we propose the use of sparse data-structures throughout the algorithms presented in the following sections. Finally, familiarity with item-based nearest neighbour collaborative filtering approaches is assumed [183].

Baseline Approaches

The naive approach to computing cosine similarities between pairs of items occurring in a given set of pageviews $\mathscr{P}_t$ is laid out in Algorithm 1. First, an inverted index from every item i to the set of users that have seen that item, U_i, is constructed. Subsequently, the algorithm iterates over said sets of users for every item-pair $i, j \in I$ and computes the sparse dot-product $\boldsymbol{i} \cdot \boldsymbol{j}$, which is equivalent to the intersection of their user-sets $|U_i \cap U_j|$. Because of the symmetric nature of our similarity measure, only half of the iterations lead to actual computations (line 6). Note that only the *size* of the intersection needs to be computed, and not the set intersection itself. Efficient algorithms with linear time complexity exist for this operation over sorted inverted indices. However, the naive baseline approach explicitly computes all $\frac{n \cdot (n-1)}{2}$ sparse vector dot-products, even when a significant amount of them are irrelevant. For many (sparse) real world datasets, this is extremely inefficient.

An improved baseline, specifically tuned to the setting of sparse data is presented in Algorithm 2. On top of the original item-to-user inverted index, we now construct a user-to-item inverted index as well. As a result, we can deconstruct the sparse vector dot-product, and iteratively count which item-pairs $i, j \in I$ also appear in I_u for every user $u \in U_i$. As $|U_i| \ll |U|$ in sparse datasets, this entails a significant efficiency advantage. Note that this baseline is less memory efficient than the naive baseline, as it needs a second inverted index to efficiently exploit the sparse nature

Algorithm 2 Sparse Baseline

Input: A set of pageviews $|\mathscr{P}_t|$.
Output: An inverted index from items to users $\mathscr{K}$, an inverted index from users to items $\mathscr{L}$, a matrix of item similarities $\mathbf{S}$.

```
1: K ← ∅, L ← ∅, S ← I
2: for (u, i, t_c) ∈ P_t do
3:    K[i] = K[i] ∪ {u}
4:    L[u] = L[u] ∪ {i}
5: for i ∈ K do
6:    for u ∈ K[i] do
7:       for j ∈ L[u] do
8:          if i < j then
9:             S_{i,j} += 1
10: for i, j ∈ S do
11:    if S_{i,j} > 0 then
12:       S_{i,j} /= sqrt(|K[i]|) · sqrt(|K[j]|)
13: return K, L, S
```

of the data. In both baseline algorithms, the square roots of the item-norms $\sqrt{|U_i|}$ can be pre-computed for improved efficiency.

2.4 Methodology

Recommendable Items

Traditionally, recommender systems are seen as functions that predict some relevance score specific to a user-item pair: $f_{\mathbf{P}} : U \times I \rightarrow [0,1]$. Here, the recommender system represented by the function f is dependent on the user-item matrix $\mathbf{P}$, hence the subscript. In real-world present-day systems, the number of users and items can quickly scale up to hundreds of thousands and even millions. It is clear that the model represented by the function $f_{\mathbf{P}}$ becomes much more complex to compute and will take up much more memory to store in the case of ever-growing user- and item-sets and the matrix $\mathbf{P}$. We identify two possible methods to alleviate this issue: either reduce the size of the training matrix $\mathbf{P}$, or reduce the complexity of $f_{\mathbf{P}}$ by putting restrictions on the set of items to compute recommendation scores for. Although the first option opens up interesting directions for future research in how datasets can be optimally summarised with minimal loss of information, we focus on the latter. We define our model as follows: $f_{\mathbf{P}} : U \times R \rightarrow [0,1]$, where $R \subseteq I$ denotes the set of *recommendable* items. This set can be highly dynamic, and depends on any number of factors such as recency, seasonality, stock and much more. Throughout the rest of this manuscript, R_t will represent the set of recommendable items at time t. When omitted, all items are considered recommendable ($R_t = I$).

Incremental Similarity Computation

Papagelis et al. present an incremental user-based CF method, focused on explicit feedback [160]. This work has later been adapted by Yang et al. to allow incremental

updates of item-based CF methods relying on explicit feedback [240]. Inspired by their work, our work focuses on incremental updates with implicit feedback. We split cosine similarity into three key components and incrementally update these components instead of recomputing the entire similarity after every update. Equation 2.1 shows the formula for computing the cosine similarity between two item vectors, where $\boldsymbol{i}_k$ represents whether user k has consumed item i.

$$\cos(\boldsymbol{i}, \boldsymbol{j}) = \frac{\boldsymbol{i} \cdot \boldsymbol{j}}{\|\boldsymbol{i}\|_2 \left\|\boldsymbol{j}\right\|_2} = \frac{\sum_{k=1}^{m} \boldsymbol{i}_k \boldsymbol{j}_k}{\sqrt{\sum_{k=1}^{m} \boldsymbol{i}_k^2} \sqrt{\sum_{k=1}^{m} \boldsymbol{j}_k^2}} \tag{2.1}$$

In the case of implicit feedback (0's and 1's) from transactional data and the use of sparse data-structures, this formulation can be rewritten as shown in Equation 2.2. Here, items i and j are no longer explicitly represented by vectors in a user-dimensional space, but rather as sets of users that have consumed these items. These sets can be easily computed from the aforementioned historical transaction data, as they are effectively a sparse column-wise representation of the binary preference matrix $\mathbf{P}$.

$$\cos(\boldsymbol{i}, \boldsymbol{j}) = \frac{|U_i \cap U_j|}{\sqrt{|U_i|} \cdot \sqrt{|U_j|}} \tag{2.2}$$

Thus, item similarities can be directly computed when the set-intersection between their respective user sets and their set sizes are known. We exploit this formulation to reduce the problem of incrementally updating item similarities to continuously updating $|U_i|$, $|U_j|$ and $|U_i \cap U_j|$ for every pair of items $i, j \in I$. We denote the vector containing all item-vectors' l_1-norms and the matrix containing all item-pair intersections at time t as follows:

$$\mathbf{N}_t \in \mathbb{N}^n : \mathbf{N}_{i,t} = |U_{i,t}|, \text{and}$$

$$\mathbf{M}_t \in \mathbb{N}^{n \times n} : \mathbf{M}_{i,j,t} = |U_{i,t} \cap U_{j,t}|.$$

The final formula for computing the similarity between two items i, j at time t then becomes the following: $\cos(\boldsymbol{i_t}, \boldsymbol{j_t}) = \frac{\mathbf{M}_{i,j,t}}{\sqrt{\mathbf{N}_{i,t}} \cdot \sqrt{\mathbf{N}_{j,t}}}$.

Since $\mathbf{M}$ is a symmetrical matrix, we can further improve performance by using appropriate data structures.

The Dynamic Index Algorithm

Suppose we have a set of recommendable items R_t at time t. Define $U_t \subseteq U$ as the set of all users u that have *ever* seen an item that is recommendable *at time t*:

$$U_t = \{u | \exists (u, i, s) \in \mathscr{P}_t \wedge i \in R_t\}.$$

Define $\mathscr{A}_t \subseteq \mathscr{P}_t$ as the set of all pageviews by users in that set:

$$\mathscr{A}_t = \{(u, i, s) \in \mathscr{P}_t | u \in U_t\}.$$

$\mathscr{A}_t$ now holds all pageviews that are relevant to the intersections $|U_i \cap U_j|$ where either i or j is a recommendable item. Naturally, when $R_t = I$, $\mathscr{A}_t = \mathscr{P}_t$. Using

Algorithm 3 Dynamic Index

Input: A set of pageviews $\mathscr{P}_t$, a set of recommendable items R_t.

Output: A matrix of item intersections $\mathbf{M}$, a vector of items' l_1-norms $\mathbf{N}$, an inverted index of users to rec. items $\mathscr{L}_r$, an inverted index of users to non-rec. items $\mathscr{L}_n$.

1: $\mathbf{M} \leftarrow \mathbf{0}, \mathbf{N} \leftarrow \mathbf{0}$
2: $\forall u \in U : \mathscr{L}_r[u] \leftarrow \emptyset, \mathscr{L}_n[u] \leftarrow \emptyset$
3: **for** $(u, i, s) \in \mathscr{P}_t$ **do**
4: **for** $j \in \mathscr{L}_r[u]$ **do**
5: $\mathbf{M}_{i,j}$ += 1
6: **if** $i \in R_t$ **then**
7: **for** $j \in \mathscr{L}_n[u]$ **do**
8: $\mathbf{M}_{i,j}$ += 1
9: $\mathscr{L}_r[u] = \mathscr{L}_r[u] \cup \{i\}$
10: $\mathbf{N}_i$ += 1
11: **else**
12: $\mathscr{L}_n[u] = \mathscr{L}_n[u] \cup \{i\}$
13: **return** $\mathbf{M}, \mathbf{N}, \mathscr{L}_r, \mathscr{L}_n$

Algorithm 3, we can compute the co-occurrence matrix $\mathbf{M}$, and thus all pair-wise similarities, efficiently. The algorithm dynamically builds two inverted indices for every user: one for all items recommendable at time t and one for all other items. The idea of dynamically indexing the data rather than doing this in a preprocessing step, is adopted from the work of Sarawagi and Kirpal [182]. This approach enables us to exploit the sparsity that is inherent to the data as we quickly identify those pairs of items that are of interest, i.e. (i, j) where 1. either i or $j \in R_t$, and 2. $|U_i \cap U_j| > 0$, while avoiding unnecessary computations on all other pairs of items. Note that this proposed algorithm is more space-efficient than the Sparse Baseline shown in Algorithm 2: $\mathscr{P}_t$ is indexed only once instead of twice.

As the inverted indices are dynamically built, the core algorithm consists of a single *for*-loop over the set of pageviews. Consequently, when a set of new user-item interactions $\Delta\mathscr{P}$ arrives, the model can be updated by executing lines 3-12 from Algorithm 3 on top of the already initialised model computed on the data $\mathscr{P}_t$. As $|\mathscr{P}|$ grows, this benefit becomes increasingly important. Figure 2.1 provides some visual intuition into this phenomenon.

As we have hinted at before, $\mathbf{M}$ is a symmetrical matrix. We avoid explicitly incrementing $\mathbf{M}_{j,i}$ when incrementing $\mathbf{M}_{i,j}$ since they will be represented as one number in an efficient implementation. Additionally, the dynamically constructed inverted indices $\mathscr{L}_r$ and $\mathscr{L}_n$ do not need to store the sets of items in an ordered manner, improving further on runtime efficiency.

From an existing model $\mathscr{M} = \{\mathbf{M}, \mathbf{N}, \mathscr{L}_r, \mathscr{L}_n\}$, we can compute all recommendable neighbours j of i with their respective cosine similarities as follows: $\cos(\boldsymbol{i}, \boldsymbol{j}) = \frac{|U_i \cap U_j|}{\sqrt{|U_i|} \cdot \sqrt{|U_j|}} = \frac{\mathbf{M}_{i,j}}{\sqrt{|U_i|} \cdot \sqrt{\mathbf{N}_j}}$. It should be noted that $|U_i|$ cannot simply be extracted from $\mathbf{N}$, since we have only computed these item norms from $\mathscr{A}_t \subseteq \mathscr{P}_t$. By definition, this vector of item norms will be up to date for recommendable items, but it might not be for non-recommendable items. However, retrieving $|U_i|$ from $\mathscr{P}_t$ is only needed when the actual cosine similarity is important and not just the internal ranking

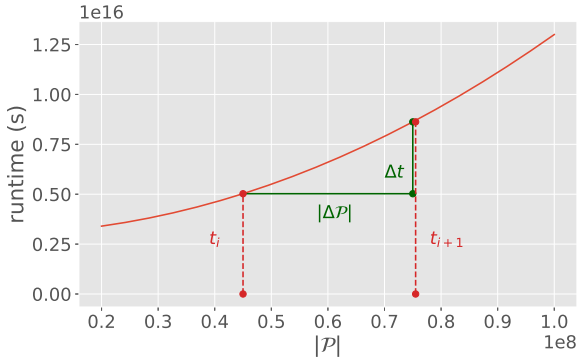


Figure 2.1: A visualisation of incremental computation, in comparison with the classical iterative variant. As more data becomes available, iterative models need to be retrained from scratch, with computation time t_{i+1}. In contrast, online or incremental methods, can update the existing model after $|\Delta\mathscr{P}|$ new user-item interactions occur, requiring only Δt time.

among neighbours. Since all similarities are divided by the constant factor $\sqrt{|U_i|}$, it is trivial to see that the internal ordering will not be impacted by this.

Parallellisation Procedure

From Algorithm 3, we can see a clear independence between the contribution of different users to the similarity of an item-pair. As $\boldsymbol{i} \cdot \boldsymbol{j}$ equals the number of users that have consumed both i and j, it is easy to see that a pageview (u, i, s) only has to be correlated with other items j seen by user u. This insight, albeit trivial, allows the computation of Algorithm 3 to be easily and efficiently parallellised following the MapReduce paradigm [33]: if the sets of users processed by every map-process are mutually disjoint, the reduce-process effectively consists of a summation of the different matrices $\mathbf{M}$ and vectors $\mathbf{N}$.

Let $\mathscr{M} = \{\mathbf{M}, \mathbf{N}, \mathscr{L}_r, \mathscr{L}_n\}$ be a model, as obtained through Algorithm 3. Figure 2.2 visualises the MapReduce-inspired parallellisation procedure we adopt in this work. With n available cores, Algorithm 3 generates n different models in parallel, as shown in the top row of Figure 2.2. As this step is embarrasingly parallel, this is the so-called Map-procedure. We then go on to recursively merge models in parallel, until we obtain one final model. This is visualised in the subsequent rows of Figure 2.2, and correlates with the Reduce-procedure. Algorithm 4 presents the process to correctly merge two models $\mathscr{M}$ and $\mathscr{M}'$. After i iterations of parallel reduce-processes have been completed, $\frac{n}{2(i-1)}$ models remain. Ergo, $\log_2(n)$ iterations of parallel reduce steps are required to obtain a single final model.

From Algorithm 4, it is clear to see that most of the complexity comes from correlating items that a given user has seen in model $\mathscr{M}'$ with items the same user has seen in $\mathscr{M}$. When parallellising the initial similarity computation, we therefore ensure that the data used for all map-processes and models $\{\mathscr{M}_0, \ldots, \mathscr{M}_n\}$ consists of entirely disjoint sets of users: $|U \cap U'| = \emptyset$. However, for incremental model updates,

Algorithm 4 Merging two models (*reduce*)

Input: $\mathbf{M}, \mathbf{M}', \mathbf{N}, \mathbf{N}', \mathscr{L}_r, \mathscr{L}'_r, \mathscr{L}_n, \mathscr{L}'_n$.
Output: $\mathbf{M}, \mathbf{N}, \mathscr{L}_r, \mathscr{L}_n$.

```
M += M'
N += N'
for u ∈ L'_r do
    for i ∈ L'_r[u] do
        for j ∈ L_r[u] do
            M_{i,j} += 1
        for j ∈ L_n[u] do
            M_{i,j} += 1
for u ∈ L'_n do
    for i ∈ L'_n[u] do
        for j ∈ L_r[u] do
            M_{i,j} += 1
∀u ∈ U : L_r[u] = L_r[u] ∪ L'_r[u]
∀u ∈ U : L_n[u] = L_n[u] ∪ L'_n[u]
return M, N, L_r, L_n
```

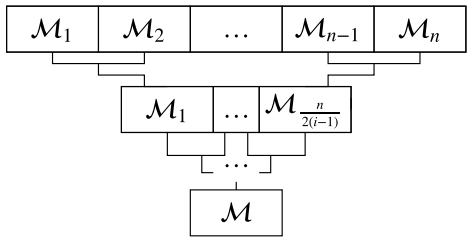


Figure 2.2: A visualisation of the MapReduce-inspired parallellisation procedure adopted in this work. Assuming n independent map-processes, n models $\{\mathscr{M}_1, \ldots, \mathscr{M}_n\}$ are obtained through Algorithm 3, and subsequently recursively merged through Algorithm 4. After i iterations of the reduce step, $\frac{n}{2(i-1)}$ models remain. Consequently, $\log_2(n)$ reduce iterations are required.

this is less straightforward: as the new model $\mathscr{M}'$ is trained on newly incoming interactions, we have no way of ensuring that the intersection between U and U' is kept minimal. As a consequence, the computational complexity of the reduce-step grows significantly, and with it the overhead of the parallellisation procedure.

Incremental Model Updates with Dynamic Recommendability

At time $t+1$, the model needs to be updated for a new set of recommendable items R_{t+1}. As the set of recommendable items changes, the set of users with interactions that are relevant to these items needs to be re-evaluated as well. We compute U_{t+1} analogous to the previous iteration: $U_{t+1} = \{u | (u, i, t_c) \in \mathscr{P}_{t+1} : i \in R_{t+1}\}$. $\mathscr{A}_{t+1}$ is initialised as the empty set $\emptyset$. Three different possibilities for every user u in

$U_t \cup U_{t+1}$ emerge:

Case $u \in U_t \setminus U_{t+1}$: The user u was relevant in the previous iteration, but no longer is. Since their inverted indices $\mathscr{L}_r[u]$ and $\mathscr{L}_n[u]$ will not be needed during this iteration, we remove them out of memory.

Case $u \in U_t \cap U_{t+1}$: The user u was relevant and still is. As all u's interactions up until time t were already incorporated in the model, we only need to take into account new interactions between time t and $t+1$:

$$\mathscr{A}_{t+1} = \mathscr{A}_{t+1} \cup \{(u,i,t_c) \in \mathscr{P}_{t+1} \setminus \mathscr{P}_t | u \in U_{t+1} \cap U_t\}.$$

Case $u \in U_{t+1} \setminus U_t$: The user u was not relevant during the previous iteration, but has become now. As the model has no record of any interactions by this user, we need to take into account their full history:

$$\mathscr{A}_{t+1} = \mathscr{A}_{t+1} \cup \{(u,i,t_c) \in \mathscr{P}_{t+1} | u \in U_{t+1} \setminus U_t\}.$$

At this point, an updated set of pageviews $\mathscr{A}_{t+1}$ to be incorporated into the model has been computed analogous to Algorithm 3. However, some precautions still need to be taken with relation to the recommendability of items over time. For every item i in $R_t \cup R_{t+1}$, three analogous cases to the ones outlined above occur:

Case $i \in R_t \setminus R_{t+1}$: The item i was recommendable in the previous iteration, but no longer is. We drop all entries in the matrix $\mathbf{M}_{i,j}$ where $j \notin R_{t+1}$. This is important to ensure consistency when the item i would later become recommendable again, otherwise increments might not start at 0. Additionally, we move item i from $\mathscr{L}_r[u]$ to $\mathscr{L}_n[u]$.

Case $i \in R_t \cap R_{t+1}$: The item i was recommendable and still is, nothing needs to be done here.

Case $i \in R_{t+1} \setminus R_t$: The item i was not recommendable during the previous iteration, but has become recommendable now. Since item i might have already been included in the index, we should compute possible intersections $\mathbf{M}_{i,j}$ that were not included in the matrix before. This is true for all users u who have seen item i before time t: $\{u | (u,l,t_c) \in \mathscr{P}_t : l = i\}$. For every non-recommendable item $j \in \mathscr{L}_n[u]$ seen by those users, we increment $\mathbf{M}_{i,j}$. Afterwards, item i has to be deleted from $\mathscr{L}_n[u]$ and inserted into $\mathscr{L}_r[u]$.

If recommendability of items is a monotonically decreasing function over time, one does not have to worry about these issues: $\{(u,l,t_c) \in \mathscr{P}_t : l = i\}$ will be the empty set for items $i \in R_{t+1} \setminus R_t$, since items that *become* recommendable are per definition new in this context. In, for example, a news recommendation setting this makes perfect sense: older articles should not be considered for recommendation. In a retail environment, however, this is not the case: recommendability will often depend on seasonality and current stock.

2.5 Experimental Results

Table 2.1 shows the characteristics of the datasets we used to experimentally validate the efficiency of our proposed approach. *Movielens* is the latest well-known Movielens dataset [57], *Netflix* refers to the full dataset that was used for the famous Netflix-Prize [13]. For both movie datasets, we converted explicit ratings to binary

Table 2.1: Experimental dataset characteristics. Datasets denoted by an asterisk (*) are binarised from explicit-feedback, to mimick the implicit-feedback setting.

	Movielens*	Netflix*	News	Outbrain
$\lvert\mathscr{P}\rvert$	20e6	100e6	96e6	200e6
$\lvert U\rvert$	138e3	480e3	5e6	113e6
$\lvert I\rvert$	27e3	18e3	297e3	1e6
$\overline{\lvert I_u\rvert}$	144.41	209.25	18.29	1.76
$\overline{\lvert U_i\rvert}$	747.84	5654.50	242.51	184.50
$\sigma(\mathbf{P})$	99.46%	98.82%	99.99%	99.99%
$\sigma(\mathbf{M})$	59.90%	0.22%	99.83%	99.98%
$\overline{\mathbf{S}_{i,j} : \mathbf{S}_{i,j} > 0}$	0.050	0.037	0.027	0.012

implicit feedback, entirely disregarding the actual ratings. *Outbrain* is a dataset containing logs from users and articles they read, published in a recent Kaggle competition [155]. We use a deduplicated version of the first 200 million logged user-item events in our experiments: in the case of recurring user-item pairs, we keep only the earliest entry. *News* is a proprietary real-world dataset consisting of roughly 96 million user-item pairs originating from article reads on the website of a large Belgian newspaper. Our algorithm, as well as the baseline methods, are implemented in C++ and compiled with all the available optimisation flags. Experiments ran on a single Intel Xeon processor. We aim to answer three research questions, respectively covered in the following sections:

RQ1 Is the proposed Dynamic Index algorithm more efficient than the state-of-the-art in computing similarity between pairs of high-dimensional sparse vectors?

RQ2 Is the proposed MapReduce-inspired parallellisation procedure effective in reducing the necessary computation time?

RQ3 What is the impact of restrictions on the set of recommendable items on the efficiency of the algorithm?

Efficiency of Dynamic Index (RQ1)

To validate the efficiency of our proposed algorithm, we report computation time for the sparse baseline and Dynamic Index, as shown in Figure 2.3. Both algorithms run on a single computational core. The naive baseline presented in Algorithm 1 is not included in these results, as it is orders of magnitude slower than the Sparse Baseline or Dynamic Index on every dataset we consider. We do not consider other algorithms in our comparison, as other proposed exact approaches in the literature were demonstrated only on dense datasets, covering a few hundred dimensions at most [241, 242, 243, 237]. Our method, aimed towards sparse datasets, can efficiently handle millions of dimensions. Additionally, to the best knowledge of the authors, no competing exact methods or implementations are available at the time of writing.

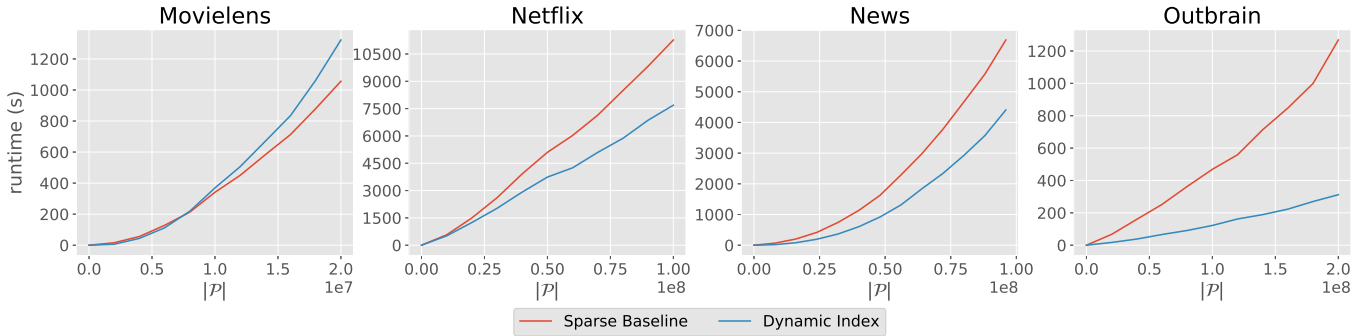


Figure 2.3: Computation time for the sparse baseline (Algorithm 2) and our proposed Dynamic Index algorithm (Algorithm 3) on the 4 different datasets laid out in Table 2.1. Both algorithms run on a single core, and all available items in the dataset are considered recommendable ($R_t = I$). We chronologically sorted the available user-item interactions and gradually increased the size of the training data passed to the algorithm (over the x-axis), in order to provide a realistic view of the benefit of online or incremental computation.

All datasets were chronologically sorted, and we gradually retrained every algorithm with more data, in order to provide a realistic view of the benefits of online or incremental computation. We can see that for all datasets but *Movielens*, our proposed algorithm significantly outperforms the sparse baseline. *Movielens* has the highest average non-zero similarity between any item-pair, which might make it more suited to Algorithm 2. However, as our method learns incrementally, the potential efficiency gains are much more tangible than merely shown by the area between the two lines in the plot. The improvement of Dynamic Index is most apparent for the largest dataset: our algorithm provides a speedup factor larger than four when all available user-item interactions are considered. Looking at the average number of pageviews processed by Algorithm 3 per second at every point in the plots in Figure 2.3, we observe throughputs ranging from $14\,500\frac{|\mathscr{P}|}{\text{s}}$ for the *Netflix* dataset, to more than $834\,000\frac{|\mathscr{P}|}{\text{s}}$ for *Outbrain*. These numbers effectively represent an upper bound on the number of new incoming pageviews per second the single-core streaming model could process in real-time, assuming a constant-rate influx. From Table 2.1 and the nature of Algorithm 3, we can deduce interesting observations about the efficiency of our approach. First, as the throughput is highest for those datasets with large $|I|$, it seems this is not an important factor. This may seem counter-intuitive at first, as more unique items will lead to more similarities that have to be computed. However, the sparsity of the co-occurrence matrix $\sigma(\mathbf{M})$ is more significant than its absolute dimensions, as we effectively leverage this by avoiding computations on zero-values. The second decisive factor is $|\overline{I_u}|$. As most of the complexity of the algorithm lies in iterating over inverted indices containing user histories, it should come as no surprise that shorter lists imply faster iterations.

Efficiency of Parallellisation Procedure (RQ2)

To validate the efficiency of our proposed parallellisation procedure, we report runtime results for the same experimental setting as laid out in Section 2.5, for a varying number of available cores. Results from this experiment are visualised in Figure 2.4. We see a clear benefit from parallellising the computation over multiple cores, over all datasets. For the *Netflix* and *News* datasets, using 8 cores provides a speedup larger than factor 4 compared to the single-core variant. The *Outbrain* dataset, which gains the least from the parallellisation scheme, was also the dataset on which the highest throughputs for the single-core algorithm were reported.

As mentioned in Section 2.4, the reduce-step for merging two models in Algorithm 4 is especially efficient when both models were generated from logged interactions by mutually disjoint sets of users. When this condition can not be guaranteed, it becomes significantly more complex. Therefore, when the batch-size $|\Delta\mathscr{P}|$ is small, the single-core variant proves to be more efficient at incremental updates than the parallellised version. However, for sufficiently large $|\Delta\mathscr{P}|$, the bulk of computation time needed to incrementally update the existing model will come from dynamically indexing the new data using Algorithm 3 to generate the new model $\mathscr{M}_{t+1}$, contrary to merging the old model $\mathscr{M}_t$ with $\mathscr{M}_{t+1}$ using Algorithm 4. In these cases, the multi-core variant proves itself to be advantageous. Moreover, in cases where the influx of new data is limited, periodically retraining the model in parallel or performing the incremental updates batch-wise might be more cost-efficient than performing incremental updates in a streaming fashion. Simplistically: if the entire

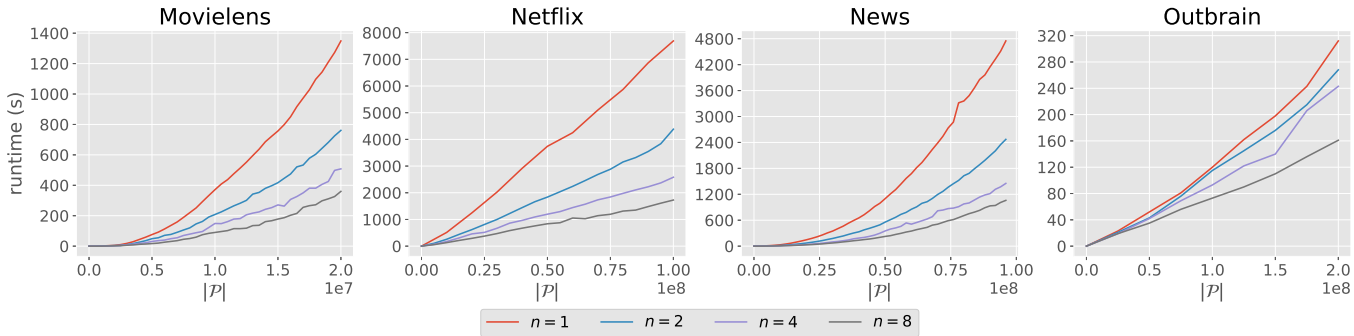


Figure 2.4: Computation time for our proposed algorithm on the 4 different datasets laid out in Table 2.1, parallellised over a varying number of computational cores ($n \in \{1, 2, 4, 8\}$). We chronologically sorted the available user-item interactions and gradually increased the size of the training data passed to the algorithm (over the x-axis). However, the model is iteratively retrained as more data becomes available. All items are considered recommendable ($R_t = I$).

model can be retrained in 20 minutes and an hour of new data can be processed in 1 minute, these options are respectively 3 and 60 times more cost-efficient than a 24/7 streaming solution.

Efficiency of Restricted Recommendability (RQ3)

Up until now, we have assumed no restrictions on the set of recommendable items. However, as we have argued before, we believe that this will often not hold in real-world applications. Whether based on recency, seasonality, available stock, business rules or any other reason, the set of items that actually should *not* be recommended can grow to be of significant size.

To demonstrate the effect that a varying set of recommendable items R_t can have on the Dynamic Index algorithm, we focus on the news recommendation application. We define δ as the recommendability threshold in this recency-focused setting: when a new item arrives, it remains recommendable for δ hours. After this period has passed, the item is no longer considered newsworthy and should no longer be recommended. Figure 2.5 shows runtime (top plot) and the number of recommendable items (bottom plot) when the model is retrained iteratively on more data, using the Dynamic Index algorithm with varying thresholds δ. Note that both y-axes are logarithmically scaled. We focus on the case where ample data is available, and show results for the last week in the *News* dataset, where the entire model is iteratively retrained on a growing set of user-item interactions.

Clear performance gains are observed when comparing the results from the restricted-recommendability variants to the unrestricted algorithm ($\delta = \infty$). First, absolute runtimes are decreased massively when focusing on a smaller, yet more relevant, set of items. For δ = 48h, the algorithm computes the exact similarity for all relevant item-pairs in < 10% of the time needed for $\delta = \infty$. The number of recommendable items, however, still exceeds 17 000, leaving plenty of room for personalisation. With δ = 24h, runtime reduces to < 5%, with more than 8 000 recommendable items. At $\delta = 6h$, these numbers turn to 1.6% of the original runtime, retaining an average of 2 100 recommendable items. The sinusoid pattern that emerges in the bottom plot for low values of δ is an artefact originating from the data: as fewer news articles are published at night, the number of recent items drops and rises periodically.

Second, looking at the slope of the runtime of the unrestricted variant compared to that of all restricted variants, we observe that the latter variants all suffer far less from ever-growing dataset sizes in terms of reduced efficiency. Last, the model size, number of recommendable items, and runtime are highly correlated with δ.

A reasonable question to ask might be how the restricted recommendability impacts the accuracy of the generated recommendations. We did not further explore this due to the following reasons: 1. When recommendability depends on recency, seasonality or available stock, these are often hard-imposed business rules. As a result, restricting recommendability is often not a choice in real-world settings. Our approach deals with this in a flexible way, and effectively exploits the imbalance for improved efficiency. 2. In offline experiments on logged feedback data, a multitude of biases is consistently present [80, 239, 52]. As users are often presented with only *recent* articles on news websites, offline experiments will heavily favour

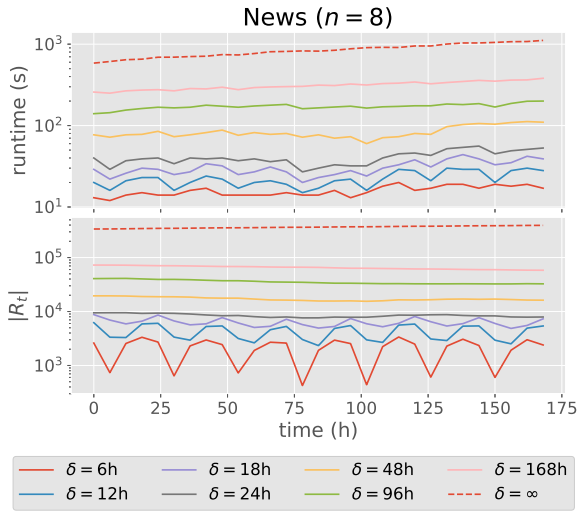


Figure 2.5: Computation time (top) and number of recommendable items (bottom) for varying recommendability thresholds in the *News* dataset ($n = 8$). δ denotes how long a new item remains recommendable after its first appearance, mimicking the real-world application of news recommendation where recency is critical. Note that both y-axes are logarithmically scaled.

recency-based approaches. On the other hand, presenting users with irrelevant and old news in an online experiment is also inappropriate for obvious reasons.

2.6 Conclusions

In this paper, we have motivated and discussed the need for highly dynamic collaborative filtering algorithms that are incrementally updateable in near real-time, to keep up with the highly dynamic environments these algorithms need to perform in. As a step towards this goal, we proposed a novel parallel approach to incrementally compute similarity among high-dimensional vectors, specifically tuned to the inherent sparsity of real-world datasets in a nearest-neighbour collaborative filtering recommender system setting. Our algorithm uses simple inverted indices to quickly identify relevant pairs of items when updates arrive, and as a consequence avoids further unnecessary computations. Moreover, we have formulated our method in accordance with the MapReduce paradigm, making it readily parallellisable and distributable. We have shown that our approach easily scales up to millions of pageviews, and is able to process industrial-sized datasets in a matter of minutes on non-specialised hardware. Attainable processing throughputs vary with configuration and data, but can easily range from tens of thousands to millions of pageviews per second. Our approach is highly scalable and flexible in terms of new users and items arriving over time. We introduced the concept of item recommendability and how it can be exploited to avoid wasting unnecessary computation time for the right use-cases. In our experiments, we effectively increased system throughput by a fac-

tor of up to 60 when considering a smaller, yet more relevant set of recommendable items.

As future work, we intend to further experimentally validate the efficiency of incremental updates to our model with non-monotonic recommendability constraints. In an attempt to further improve upon the scalability of CF systems, summarisation algorithms to compress a transactional dataset with minimal information loss, specifically in the context of recommender systems, would be an interesting direction for future research. Furthermore, we intend to look into other similarity functions to determine whether they can be decomposed and incrementally computed as well. As Jaccard Index, Pointwise Mutual Information and Pearson's correlation coefficient all depend on the co-occurrence matrix $\mathbf{M}$, we believe this to be an attainable extension of our work. Throughout this manuscript, we have focused on item-to-item nearest-neighbour collaborative filtering as the main application of our work. When changing the terminology from "users" and "items" to "terms" and "documents", we believe that our approach is applicable to more general information retrieval use-cases as well. Nevertheless, in these settings, extensions for non-binary data (by including a term-value pair in the inverted indices instead of just the term) would be appropriate. Naturally, when these inverted indices keep growing in size, compression techniques might be convenient to improve on space efficiency. However, most state-of-the-art compression techniques do not support incremental updates, and random access would be imperative [153, 166].

Reflections

This Chapter has focused on incremental similarity computation for item-based nearest-neighbour models, but has largely ignored the inseparable problem of nearest-neighbour querying based on the acquired similarities. The latter is largely an open problem, as indices for querying typically need to be rebuilt from scratch when new data arrives. Tackling these two problems jointly could give rise to fully incremental, end-to-end recommendation pipelines. Nevertheless, the contributions presented in this Chapter represent an important step in this direction.

Furthermore, this Chapter has dealt with analytically computable similarity metrics, such as cosine similarity (or those that consist of the same building blocks). Several important extensions have been made to this modelling approach, where the item-item matrix is trained to minimise some objective function on the data (e.g. SLIM [147] and EASE$^{\text{R}}$ [200]). As these extensions yield significantly improved recommendation accuracy, a natural question to ask is whether we can provide an incremental update procedure for these methods. This question provides the main motivation for the work presented in Chapter 3.

CHAPTER 3

Embarrassingly Shallow Auto-Encoders for Dynamic Collaborative Filtering

Recent work has shown that, despite their simplicity, item-based models optimised through ridge regression can attain highly competitive results on collaborative filtering tasks. As these models are analytically computable and thus forgo the need for often expensive iterative optimisation procedures, they have become an attractive choice for practitioners. Computing the closed-form ridge regression solution consists of inverting the Gramian item-item matrix, which is known to be a costly operation that scales poorly with the size of the item catalogue. Because of this bottleneck, the adoption of these methods is restricted to a specific set of problems where the number of items is modest. This can become especially problematic in real-world dynamical environments, where the model needs to keep up with incoming data to combat issues of cold start and concept drift.

In this work we propose Dynamic EASE$^{\text{R}}$*: an algorithm based on the Woodbury matrix identity that incrementally updates an existing regression model when new data arrives, either approximately or exact. By exploiting a widely accepted low-rank assumption for the user-item interaction data, this allows us to target those parts of the resulting model that need updating, and avoid a costly inversion of the entire item-item matrix with every update. We theoretically and empirically show that our newly proposed methods can entail significant efficiency gains in the right settings, broadening the scope of problems for which closed-form models are an appropriate choice.*[1]

[1] This chapter is based on work under submission to the *User Modeling and User-Adapted Interaction (UMUAI) Special Issue on Dynamic Recommender Systems and User Models (DyRSUM)* as "Embarrassingly Shallow Auto-Encoders for Dynamic Collaborative Filtering" by Olivier Jeunen, Jan Van Balen and Bart Goethals.

3.1 Introduction

Recommender systems are information retrieval applications that aim to mitigate the problem of "information overload", by matching users to certain *items* [18]. They have become ubiquitous on the world wide web, and have found applications in many different areas where these *items* can represent anything from news articles and musical artists to retail products and social media accounts. Most modern approaches to recommendation are based on some form of *collaborative filtering* [39], a family of methods that aim to model user preferences and learn them from a dataset of user behaviour. These methods have known widespread success over the years, and are the cornerstone of modern recommender systems research. As a consequence, the quest for more effective collaborative filtering algorithms is a very lively research area, where significant strides forward are being made every year. Many novel methods are based on deep and non-linear neural networks, and the expressiveness of this model class has made them ubiquitous in the field [116, 40, 189]. Recent work casts doubt on the reproducibility of evaluation strategies that are often adopted to empirically validate research findings [32, 170, 171], making it harder to conclude whether these complex model classes are what the field needs moving forward.

In a parallel line of research, the effectiveness of simpler linear models for the collaborative filtering task has been shown time and again [147, 109, 187, 201, 202, 203]. Most notably and recently, Embarrassingly Shallow Auto-Encoders (reversed: EASE$^{\text{R}}$) have been shown to yield highly competitive results with the state-of-the-art, whilst often being much easier to implement, and much more efficient to compute [200]. The closed-form solution that is available for ridge regression models is at the heart of these major advantages, as EASE$^{\text{R}}$ effectively optimises a regularised least-squares problem. Recently, EASE$^{\text{R}}$ has been extended to incorporate item metadata into two variants: CEASE$^{\text{R}}$ and ADD-EASE$^{\text{R}}$ [85]. These extensions improve the capabilities of closed-form linear models to deal with issues such as the "long tail" (very few items account for the large majority of interactions) and "cold start" (new items do not have any interactions) [186, 161, 190].

The main benefit of EASE$^{\text{R}}$ and its variants over competing approaches, is their computational efficiency. As the core algorithm consists of a single inversion of the Gramian item-item matrix, it is often many times more efficient to compute than models relying on iterative optimisation techniques. As reported in the original paper, the algorithm can be implemented in just a few lines of Python and is typically computed in the order of minutes on various often used publicly available benchmark datasets [200]. Nevertheless, matrix inversion is known to scale poorly for large matrices, and EASE$^{\text{R}}$'s reliance on it does inhibit its adoption in use-cases with large item catalogues. In such cases, methods that rely on gradient-based optimisation techniques are still preferable.

To add insult to injury, real-world systems rarely rely on a single trained model that is trained once and then deployed. To make this concrete: suppose we operate a hypothetical retail website, and we wish to send out an e-mail with a top-N list of personalised recommendations to our subscribed users every few days. Naturally, the model that generates these recommendation lists should evolve over time, preferably incorporating new user-item interactions that occurred over the past days. The importance of a dynamic model like that is threefold: 1. it will generate more

novel and diverse recommendations than its static counterpart [21], 2. it will be able to combat concept drift in the data (due to shifting item popularity or seasonality trends in preferences) [47], and 3. it will have a means to handle cold-start problems when either with new items or news users appear [186]. Many modern digital systems generate new data at increasingly fast rates, and this is no different for our hypothetical retail website. This is important to take into account when choosing a recommendation algorithm. Models that are already inefficient to compute initially, will only see these problems exacerbated when the predominant approach every few days is to recompute them iteratively on more and more data. This puts a theoretical limit on how often we can update the model, and incurs a computational cost that we would like to reduce. Instead, it would be much more preferable to have models that can be updated with new information when it arrives, but do not require a full retraining of untouched parameters for every new batch of data that comes in. This is not an easy feat, and the field of "online recommender systems" that are able to handle model updates more elegantly has seen much interest in recent years [225]. More generally, the problem of "lifelong" or "continual" learning in the machine learning field deals with similar issues [24].

In this work, we present a novel algorithm to incrementally update the state-of-the-art item-based linear model EASE^R, which is naturally extended to include recent variants that exploit side-information: CEASE^R and ADD-EASE^R. EASE^R consists of two major computation steps: (1) the generation of the Gramian item-item matrix, and (2) the inversion of this matrix that yields the solution to the regression problem.

We propose Dynamic EASE^R (DYN-EASE^R), consisting of incremental update rules for these two steps that leverage the recently proposed Dynamic Index algorithm [83] and the well-known Woodbury matrix identity [55] respectively. As such, DYN-EASE^R provides a way to efficiently update an existing EASE^R-like model without the need of recomputing the entire regression model from scratch with every data update.

A theoretical analysis of the proposed algorithm shows that the highest efficiency gains can be expected when the rank of the update to the Gramian is low, an assumption that has been widely adopted in the recommender systems literature before [98]. We show how this quantity can be bounded using simple summary statistics from the new batch of data, and support our findings with empirical results. Further experiments confirm that DYN-EASE^R is able to significantly cut down on computation time compared to iteratively retrained EASE^R, in a variety of recommendation domains. Finally, we show how we can update the model with low-rank approximations when the new batch of data itself is not low-rank; providing a tuneable trade-off between the exactness of the solution and the efficiency with which it can be kept up-to-date. Empirical observations show how this approximate variant of DYN-EASE^R still yields highly competitive recommendation performance, with greatly improved update speed. As a result, our work broadens the space of recommendation problems to which the state-of-the-art linear model EASE^R can efficiently be applied. To foster the reproducibility of our work, all source code for the experiments in Section 3.4 is publicly available at github.com/olivierjeunen/dynamic-easer/.

This Chapter builds upon and combines parts of our previously published work on incremental models [83], and our preliminary work on closed-form models that exploit side-information [85].

The rest of this Chapter is structured as follows: Section 3.2 introduces our use-case, with mathematical notation and relevant related work; Section 3.3 introduces

DYN-EASE^{R} and presents a theoretical analysis of its inner workings; Section 3.4 presents empirical observations from a wide range of experiments and shows where DYN-EASE^{R} can provide meaningful improvements, findings that are in line with what the theory suggests. This work is concluded in Section 3.5, where we additionally present a scope for future research.

3.2 Background and Related Work

We first formalise our use-case, and present relevant mathematical notation used throughout the rest of this work. We are interested in the "binary, positive-only" implicit feedback setting [221], where we have access to a dataset consisting of preference indications from users in $\mathcal{U}$ over items in $\mathcal{I}$ at time $t \in \mathbb{N}$, assumed from a set of interaction data $\mathcal{P} \subseteq \mathcal{U} \times \mathcal{I} \times \mathbb{N}$. Ignoring temporal information, these preferences can be represented in a binary user-item matrix $\boldsymbol{X} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, where $\boldsymbol{X}_{u,i} = 1$ if we have a click, view, purchase,... for user u and item i in $\mathcal{P}$, and $\boldsymbol{X}_{u,i} = 0$ otherwise. With $\mathcal{P}_t$, we denote the set of all interactions up to time t: $\{(u,i,t') \in \mathcal{P} | t' < t\}$. Consequently, $\boldsymbol{X}_t$ is the user-item matrix constructed from the set of interactions $\mathcal{P}_t$. We will refer to the set of all items seen by user u as $\mathcal{I}_u \subseteq \mathcal{I}$, and vice versa $\mathcal{U}_i \subseteq \mathcal{U}$ for an item i. The Gramian of the user-item matrix is defined as $\boldsymbol{G} := \boldsymbol{X}^\top \boldsymbol{X}$; it is an item-item matrix that holds the co-occurrence count for items i and j at index $\boldsymbol{G}_{i,j}$. The goal at hand for a recommendation algorithm is to predict which zeroes in the user-item matrix $\boldsymbol{X}$ actually *shouldn't* be zeroes, and thus imply that the item would in some way "fit" the user's tastes and consequently make for a good item to be shown as a recommendation.

In some cases, additional information about items can be available. Such "side-information" or "metadata" often comes in the form of discrete *tags*, which can for example be a release year, genre or director for a movie, an artist or genre for a song, a writer for a book, or many more. Incorporating item metadata in the modelling process can help mitigate cold-start and long-tail issues, where the preference information for a given item is limited [186, 161]. We will refer to the set of all such tags as the *vocabulary* $\mathcal{V}$. In a similar fashion to the user-item matrix $\boldsymbol{X}$, a tag-item matrix $\boldsymbol{T} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{I}|}$ is constructed. Note that this matrix is real-valued, as it will often contain pre-computed values such as tf-idf weights instead of binary indicators.

In what follows, we present a brief introduction to item-based recommendation models, most notably ITEM-KNN [183], SLIM [147] and EASE^{R} [200]. We then additionally introduce CEASE^{R} and ADD-EASE^{R} as extensions of EASE^{R} that incorporate item side-information whilst retaining a closed-form solution [85], as these are most relevant to the dynamic EASE^{R} algorithm we will present in Section 3.3. This section is concluded with an overview of related work in the field of incremental collaborative filtering approaches.

Item-based Models, SLIM & EASE^{R}

Item-based collaborative filtering models tackle the recommendation task by defining a conceptual similarity matrix $\boldsymbol{S} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$. The score given to a potential recommendation is then computed as the sum of similarities between items in the user's

history and the item at hand:

$$\text{score}(u,i) = \sum_{j \in \mathcal{I}_u} \boldsymbol{S}_{j,i} = (\boldsymbol{X}_{u,\cdot}\boldsymbol{S})_i \tag{3.1}$$

Here, $\boldsymbol{X}_{\boldsymbol{u},\cdot}$ denotes the u^{th} row of $\boldsymbol{X}$. Note that computing recommendation scores for all training users and all items simply consists of computing the matrix multiplication $\boldsymbol{XS}$, an operation that is made more efficient when the matrix $\boldsymbol{S}$ is restricted to be sparse. Scores for items i already present in the user history $\mathcal{I}_u$ are often ignored, and the remaining items are ranked and presented in a top-N recommendation list or slate to the user. Early seminal works would define the similarity matrix $\boldsymbol{S}$ as all pairwise cosine similarities among items in the high-dimensional but sparse user-item matrix $\boldsymbol{X}$ [183]. This has then been extended to include slightly more advanced notions of similarity such as Pearson's correlation or conditional probabilities [34]. Recent work has introduced the "Dynamic Index" algorithm to incrementally compute the Gramian of $\boldsymbol{X}$, additionally showing that several conventional similarity metrics such as cosine similarity or Jaccard index can be readily computed from $\boldsymbol{G}$ when it is up-to-date [83].

Methods for actually *learning* an optimal item-item similarity matrix have been proposed for the task of rating prediction [97], as well as for pairwise learning from implicit feedback [169]. Ning and Karypis were the first to propose to learn a sparse weight matrix $\boldsymbol{S}$ through a pointwise optimisation procedure, aptly dubbing their approach the Sparse LInear Method (SLIM) [147]. SLIM optimises a least-squares regression model with elastic net regularisation, constrained to positive weights:

$$\boldsymbol{S}^* = \underset{\boldsymbol{S}}{\arg\min} \|\boldsymbol{X} - \boldsymbol{XS}\|_F^2 + \lambda_1 \|\boldsymbol{S}\|_1^2 + \lambda_2 \|\boldsymbol{S}\|_F^2, \quad \text{subject to } \text{diag}(\boldsymbol{S}) = 0 \text{ and } \boldsymbol{S} \geq 0. \tag{3.2}$$

The restriction of the diagonal to zero avoids the trivial solution where $\boldsymbol{S} = \boldsymbol{I}$. Many extensions of SLIM have been proposed in recent years, and it has become a widely used method for the collaborative filtering task [148, 109, 29, 187, 30, 200, 202, 203, 26]. In practice, the SLIM optimisation problem is often decomposed into $|\mathcal{I}|$ independent problems (one per target item). Although these can then be solved in an embarrassingly parallel fashion, this renders the approach intractable for very large item catalogues. Indeed, as they aim to solve $|\mathcal{I}|$ regression problems, their computational complexity is in the order of $\mathcal{O}(|\mathcal{I}|(|\mathcal{I}|-1)^{2.373})$, assuming they exploit the recent advances in efficient matrix multiplication and inversion [104, 7]. The computational cost of the original SLIM approach is a known impediment for its adoption in certain use-cases; related work has reported that hyper-parameter tuning took several weeks on even medium-sized datasets [116].[2]

Steck studied whether the restrictions of SLIM to only allow *positive* item-item weights and their l_1-regularisation-induced sparsity were necessary for the resulting model to remain competitive, and concluded that this was not always the case [200, 203]. The resulting Tikhonov-regularised least-squares problem can then be formalised as:

$$\boldsymbol{S}^* = \underset{\boldsymbol{S}}{\arg\min} \|\boldsymbol{X} - \boldsymbol{XS}\|_F^2 + \lambda \|\boldsymbol{S}\|_F^2, \text{ subject to } \text{diag}(\boldsymbol{S}) = 0. \tag{3.3}$$

[2] It should be noted that the authors have since released a more performant coordinate-descent-based implementation of their method [149].

The main advantage of simplifying the optimisation problem at hand, is that the well-known closed form solutions for Ordinary Least Squares (OLS) and ridge regression can now be adopted. Including the zero-diagonal constraint via Lagrange multipliers yields the Embarrassingly Shallow Auto-Encoder (EASE$^{\text{R}}$) model:

$$\hat{\boldsymbol{S}} = \boldsymbol{I} - \hat{\boldsymbol{P}} \cdot \text{diagMat}(\mathbf{1} \oslash \text{diag}(\hat{\boldsymbol{P}})), \text{ where } \hat{\boldsymbol{P}} := (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1}. \tag{3.4}$$

As this model consists of a single regression problem to be solved and thus a single matrix inversion to be computed, its complexity is orders of magnitude smaller than that of the original SLIM variants: $\mathcal{O}(|\mathcal{I}|^{2.373})$. EASE$^{\text{R}}$ no longer yields a sparse matrix, possibly making Equation 3.1 much less efficient to compute. Nevertheless, the author reported that there was only a marginal performance impact when simply sparsifying the learnt matrix by zero-ing out weights based on their absolute values up until the desired sparsity level. As an additional advantage, EASE$^{\text{R}}$ has only a single regularisation strength hyper-parameter to tune compared to the two needed for SLIM's elastic net regularisation. We refer the interested reader to [200, 201] for a full derivation of the model and additional information.

Another recent extension of the SLIM paradigm proposes to use Block-Diagonal-Regularisation (BDR) to obtain a block-aware item similarity model [26]. The block-diagonal structure in the learnt matrix inherently represents clusters among items. As inter-block similarities are penalised, BDR has a sparsity-inducing effect that positively impacts the efficiency of the recommendation-generating process. Because the block-aware model presented by [26] no longer has an analytically computable solution readily available, further comparison with their method is out of scope for the purposes of this work. The item-based paradigm and its closed-form instantiations have also recently been adapted for bandit-based recommendation use-cases[78].

Item-Based Models with Side-Information

The EASE$^{\text{R}}$ definition can be further extended to incorporate side-information in either a "collective" (CEASE$^{\text{R}}$) or "additive" (ADD-EASE$^{\text{R}}$) manner [85]. The first method, inspired by collective SLIM [148], intuitively treats discrete tags equivalent to how users are treated, and re-weights their contribution to the solution of the regression problem by the diagonal weight-matrix $\boldsymbol{W} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{V}|)\times(|\mathcal{U}|+|\mathcal{V}|)}$:

$$\boldsymbol{S}^* = \arg\min_{\boldsymbol{S}} \left\| \sqrt{\boldsymbol{W}}(\boldsymbol{X}' - \boldsymbol{X}'\boldsymbol{S}) \right\|_F^2 + \lambda \|\boldsymbol{S}\|_F^2,$$
$$\text{subject to } \text{diag}(\boldsymbol{S}) = 0, \text{ where } \boldsymbol{X}' = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{T} \end{bmatrix}. \tag{3.5}$$

The closed-form solution is then given by Equation 3.6, where $\oslash$ denotes element-wise division, diag$(\cdot)$ extracts the diagonal from a matrix, diagMat$(\cdot)$ generates a square diagonal matrix from a vector, and $\mathbf{1}$ is a vector of ones.

$$\hat{\boldsymbol{S}} = \boldsymbol{I} - \hat{\boldsymbol{P}} \cdot \text{diagMat}(\mathbf{1} \oslash \text{diag}(\hat{\boldsymbol{P}})), \text{ where } \hat{\boldsymbol{P}} := (\boldsymbol{X}'^\top \boldsymbol{W} \boldsymbol{X}' + \lambda \boldsymbol{I})^{-1} \tag{3.6}$$

The second method, ADD-EASE$^{\text{R}}$, treats the regression problem on the user-item matrix $\boldsymbol{X}$ and the one on the tag-item matrix $\boldsymbol{T}$ as two fully independent problems

to solve in parallel; combining the two resulting item-item weight matrices $\boldsymbol{S_X}$ and $\boldsymbol{S_T}$ in an additive fashion later down the line.

$$\begin{aligned}\boldsymbol{S}^* = \alpha \operatorname*{argmin}_{\boldsymbol{S_X}} \left(\left\| \sqrt{\boldsymbol{W_X}}(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{S_X}) \right\|_F^2 + \lambda_{\boldsymbol{X}} \|\boldsymbol{S_X}\|_F^2 \right) \\ +(1-\alpha) \operatorname*{argmin}_{\boldsymbol{S_T}} \left(\left\| \sqrt{\boldsymbol{W_T}}(\boldsymbol{T} - \boldsymbol{T}\boldsymbol{S_T}) \right\|_F^2 + \lambda_{\boldsymbol{T}} \|\boldsymbol{S_T}\|_F^2 \right), \\ \text{subject to } \operatorname{diag}(\boldsymbol{S_X}) = \operatorname{diag}(\boldsymbol{S_T}) = 0.\end{aligned} \tag{3.7}$$

ADD-EASE$^{\text{R}}$ doubles the amount of parameters used by EASE$^{\text{R}}$ and CEASE$^{\text{R}}$, increasing its degrees of freedom at learning time at the cost of having to solve two regression problems instead of one. Note, however, that these are fully independent and can be computed in parallel. Equation 3.8 shows the analytical formulas to obtain the two independent models, and combine them with a blending parameter $0 \leq \alpha \leq 1$.

$$\begin{aligned}\hat{\boldsymbol{S}}_{\boldsymbol{X}} &= \boldsymbol{I} - \hat{\boldsymbol{P}}_{\boldsymbol{X}} \cdot \operatorname{diagMat}(\mathbf{1} \oslash \operatorname{diag}(\hat{\boldsymbol{P}}_{\boldsymbol{X}})), \text{ where } \hat{\boldsymbol{P}}_{\boldsymbol{X}} := (\boldsymbol{X}^\top \boldsymbol{W_X} \boldsymbol{X} + \lambda_{\boldsymbol{X}} \boldsymbol{I})^{-1} \\ \hat{\boldsymbol{S}}_{\boldsymbol{T}} &= \boldsymbol{I} - \hat{\boldsymbol{P}}_{\boldsymbol{T}} \cdot \operatorname{diagMat}(\mathbf{1} \oslash \operatorname{diag}(\hat{\boldsymbol{P}}_{\boldsymbol{T}})), \text{ where } \hat{\boldsymbol{P}}_{\boldsymbol{T}} := (\boldsymbol{T}^\top \boldsymbol{W_T} \boldsymbol{T} + \lambda_{\boldsymbol{T}} \boldsymbol{I})^{-1} \\ \hat{\boldsymbol{S}} &= \alpha \hat{\boldsymbol{S}}_{\boldsymbol{X}} + (1-\alpha) \hat{\boldsymbol{S}}_{\boldsymbol{T}}\end{aligned} \tag{3.8}$$

The computational complexity of CEASE$^{\text{R}}$ and ADD-EASE$^{\text{R}}$ remains in the order of $\mathcal{O}(|\mathcal{I}|^{2.373})$, which is equivalent to the original EASE$^{\text{R}}$ approach. As such, these methods allow item side-information to be included into the model without a significant added cost in terms of computational complexity. The main reason for this, is that we adapt the entries in the Gramian $\boldsymbol{G}$, but do not alter its dimensions.

Incremental Collaborative Filtering

Collaborative filtering techniques that can be incrementally updated when new data arrives are a lively research area in itself. Vinagre et al. propose incremental Stochastic Gradient Descent (SGD) as a way to dynamically update matrix factorisation models based on positive-only implicit feedback [222]. Their methodology has first been extended to include negative feedback [224], and then to a co-factorisation model that is more complex than traditional matrix factorisation, but also leads to superior recommendation accuracy [8]. He et al. propose an incremental optimisation procedure based on Alternating Least Squares (ALS), and also show how it can be applied to efficiently and effectively update matrix factorisation models [60]. More recently, Ferreira et al. propose a method that personalises learning rates on a user-basis, reporting further improvements. In contrast, our work focuses on item-based similarity models that come with closed-form solutions, as these have been shown to be highly competitive with the state-of-the-art in many collaborative filtering use-cases.

Instead of just incorporating new data into the model, Matuszyk et al. propose to *forget* older data that has become obsolete, reporting significantly improved performance for collaborative filtering approaches [133]. The dynamic EASE$^{\text{R}}$ method we propose in Section 3.3 fits perfectly into this paradigm, as it can incorporate new data just as easily as it can forget irrelevant information in a targeted manner. This type of de-cremental learning has the additional advantage of being able to

avoid complete retraining in privacy-sensitive application areas, where specific user histories need to be removed from the model upon request.

Neural Auto-Encoders

The Auto-Encoder paradigm of which EASE^R is a specific instantiation, has gained much popularity in recent years. The Mult-VAE method proposed by Liang et al. [116], consists of a variational auto-encoder with a multinomial likelihood, and has been a strong baseline for several years [32]. Khawar et al. propose an architecture that first learns a grouping of items and leverages this structure when learning the auto-encoder, reporting significant gains over the original Mult-VAE method [93]. As these methods rely on gradient-based optimisation of often highly non-convex objective functions, they rely on software packages with automatic differentiation capabilities, and typically require significant computational resources, in the form of several hours of training on machines equipped with GPUs. The methods we consider in this work are computed in the order of minutes on CPUs, and we do not include neural approaches in our comparison for this reason. Furthermore, among others, the work of Steck [200] and Dacrema et al. [32] have repeatedly shown that linear item-based models can attain highly competitive recommendation accuracy compared to neural alternatives.

3.3 Methodology and Contributions

We have given a brief history of item-based collaborative filtering models, and have discussed why EASE^R and its variants are computationally often more efficient than their counterparts based on SLIM. For very large item catalogues, however, its more than quadratic computational complexity in the number of items still becomes a very tangible issue. Because of this, the demand for an algorithm that can efficiently update EASE^R-like models when new data arrives, is still very real, and a necessity for these methods to obtain widespread adoption in practice. Recent work proposes the "Dynamic Index" algorithm as a way to incrementally update item similarities in neighbourhood-based models that adopt cosine similarity [83]. A crucial building block of this metric and the algorithm is the efficient and incremental computation of the Gramian matrix $\boldsymbol{G} = \boldsymbol{X}^\top \boldsymbol{X}$. By storing $\boldsymbol{G}$ in low-overhead sparse data-structures such as inverted indices, they minimise memory overhead whilst still allowing for an amortised constant lookup time when querying $\mathcal{I}_u$, which is a requirement for incremental updates. From Equations 3.4, 3.6 and 3.8, it is clear to see that EASE^R and its variants are dependent on this Gram-matrix as well. In fact, it is the *only* building block needed to be able to compute the resulting item-item weight matrix $\hat{\boldsymbol{S}}$. As such, we adopt parts of the Dynamic Index algorithm proposed by Jeunen et al. to first efficiently compute and then incrementally update the Gramian matrix $\boldsymbol{G}$. Once we have an up-to-date matrix $\boldsymbol{G}$, we need to compute its inverse to obtain $\hat{\boldsymbol{P}}$ and the eventual model $\hat{\boldsymbol{S}}$ from that. The matrix inversion to go from $\boldsymbol{G}$ to $\hat{\boldsymbol{P}}$ is the workhorse behind EASE^R that takes up the large majority of the computation time, as this step corresponds to solving the regression problem formulated in Equation 3.3. Iterative re-computation of this matrix inverse every time we wish to incorporate new data into the model, is thus to be avoided if it can be.

Algorithm 5 DYN-GRAM

Input: $\mathscr{P}_{\Delta}$, $\mathscr{L}$
Output: $\boldsymbol{G}_{\Delta}$, $\mathscr{L}$
1: $\boldsymbol{G}_{\Delta} = 0$
2: **for** $(u, i) \in \mathscr{P}_{\Delta}$ **do**
3: **for** $j \in \mathscr{L}[u]$ **do**
4: $\boldsymbol{G}_{\Delta,i,j}$ += 1
5: $\boldsymbol{G}_{\Delta,j,i}$ += 1
6: $\boldsymbol{G}_{\Delta,i,i}$ += 1
7: $\mathscr{L}[u] = \mathscr{L}[u] \cup \{i\}$
8: **return** $\boldsymbol{G}_{\Delta}, \mathscr{L}$

Low-Rank Model Updates with the Woodbury Matrix Identity

Equation 3.9 shows the Woodbury matrix identity, which posits that the inverse of a rank-k correction of some $n \times n$ matrix $\boldsymbol{A}$ can be computed by performing a rank-k correction on the inverse of that matrix $\boldsymbol{A}$ [55].

$$(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1} \tag{3.9}$$

So, given $\boldsymbol{A}^{-1}$, $\boldsymbol{U}$, $\boldsymbol{C}$ and $\boldsymbol{V}$, there is no need to re-compute the inversion on the update of $\boldsymbol{A}$, but it is sufficient to multiply a few matrices and compute the inverse of $(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}) \in \mathbb{R}^{k \times k}$. Naturally, for large n and $k \ll n$, the efficiency gains will be most significant. Note that the inversion of $\boldsymbol{C}$ is trivial and consists of just k operations, as $\boldsymbol{C}$ is a diagonal matrix.

In our setting, suppose we have an up-to-date model at a certain time t with $\boldsymbol{X}_t$, $\boldsymbol{G}_t$, $\hat{\boldsymbol{P}}_t$ and $\hat{\boldsymbol{S}}_t$. At a given time $t+1$, suppose we have an updated user-item matrix $\boldsymbol{X}_{t+1}$, but we wish to compute $\boldsymbol{G}_{t+1}$, $\hat{\boldsymbol{P}}_{t+1}$ and the resulting $\hat{\boldsymbol{S}}_{t+1}$ as efficiently as possible. As we mentioned before, computing $\boldsymbol{G}_{t+1}$ incrementally can be achieved easily and efficiently by adopting parts of the Dynamic Index algorithm. In fact, because of the incremental nature of the algorithm, we can easily just store the *difference* in the Gramian matrix instead of its entirety: $\boldsymbol{G}_{\Delta} = \boldsymbol{G}_{t+1} - \boldsymbol{G}_t = \boldsymbol{X}_{t+1}^{\top}\boldsymbol{X}_{t+1} - \boldsymbol{X}_t^{\top}\boldsymbol{X}_t$. Given a set of user-item interactions $\mathscr{P}_{\Delta} \subset \mathscr{U} \times \mathscr{I}$ to include into the model and an inverted index $\mathscr{L}_t$ mapping users u to their histories $\mathscr{I}_u$, Algorithm 5 shows how to achieve this. Note that the indices holding $\mathscr{I}_u$ are just a sparse representation of the user-item matrix $\boldsymbol{X}$ and don't require any additional memory consumption. Furthermore, Algorithm 5 is easily parallellisable through the same MapReduce-like paradigm adopted by [83]. Naturally, an efficient implementation will exploit the symmetry of the Gramian $\boldsymbol{G}_{\Delta}$ to decrease memory consumption as well as the number of floating point operations needed at every update.

Now, having computed $\boldsymbol{G}_{\Delta}$, we can rewrite what we need as follows:

$$\hat{\boldsymbol{P}}_{t+1} = (\boldsymbol{G}_{t+1} + \lambda\boldsymbol{I})^{-1} = (\boldsymbol{G}_t + \lambda\boldsymbol{I} + \boldsymbol{G}_{\Delta})^{-1}. \tag{3.10}$$

The form on the right-hand side already begins to resemble Woodbury's formula in Equation 3.9. All that's left is to decompose $\boldsymbol{G}_{\Delta} \in \mathbb{R}^{n \times n}$ into matrices $\boldsymbol{U} \in \mathbb{R}^{n \times k}$,

$\boldsymbol{C} \in \mathbb{R}^{k \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{k \times n}$. As $\boldsymbol{G}_{\boldsymbol{\Delta}}$ is the difference of two real symmetric matrices $\boldsymbol{G}_{t+1}$ and $\boldsymbol{G}_t$, it will always be a real symmetric matrix as well. This means that the eigenvectors of $\boldsymbol{G}_{\boldsymbol{\Delta}}$ can be chosen to be orthogonal to each other: $\boldsymbol{Q}^\intercal \equiv \boldsymbol{Q}^{-1}$. Consequently, an eigendecomposition always exists, where k is the rank of $\boldsymbol{G}_{\boldsymbol{\Delta}}$:

$$\begin{aligned} \boldsymbol{G}_{\boldsymbol{\Delta}} &= \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \\ &= \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\intercal \\ &= \sum_{i=1}^{k} \boldsymbol{\Lambda}_{ii} \boldsymbol{Q}_{\cdot,i} \boldsymbol{Q}_{\cdot,i}^\intercal. \end{aligned} \tag{3.11}$$

As such, we can plug Equation 3.11 containing the eigendecomposition of $\boldsymbol{G}_{\boldsymbol{\Delta}}$ into Equations 3.9 and 3.10 to obtain our final update rule in Equation 3.12:

$$\begin{aligned} \hat{\boldsymbol{P}}_{t+1} &= (\boldsymbol{G}_t + \lambda \boldsymbol{I} + \boldsymbol{G}_{\boldsymbol{\Delta}})^{-1} \\ &= (\boldsymbol{G}_t + \lambda \boldsymbol{I} + \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\intercal)^{-1} \\ &= \hat{\boldsymbol{P}}_t - \hat{\boldsymbol{P}}_t \boldsymbol{Q}(\boldsymbol{\Lambda}^{-1} + \boldsymbol{Q}^\intercal \hat{\boldsymbol{P}}_t \boldsymbol{Q})^{-1} \boldsymbol{Q}^\intercal \hat{\boldsymbol{P}}_t. \end{aligned} \tag{3.12}$$

The full DYN-EASE$^{\text{R}}$ procedure is presented in Algorithm 6. If the updates to the Gramian matrix are low-rank, this procedure will be much more computationally efficient than re-computing the inverse of the entire Gramian matrix from scratch, as we will shown in the following subsection. The assumption that the data-generating process behind user-item interactions is generally low-rank, has been exploited far and wide in the recommender systems literature [98].

It is interesting to note that EASE$^{\text{R}}$ does not follow the low-rank assumption that motivates the popular family of latent factor models for collaborative filtering. Indeed, EASE$^{\text{R}}$ is a full-rank model, combatting overfitting with Gaussian priors on its parameters rather than reducing the dimensionality of the problem. The low-rank assumption we adopt here is on the update to the Gramian $\boldsymbol{G}_{\boldsymbol{\Delta}}$, instead of the full Gramian $\boldsymbol{G}$. As we will show further on, both theoretically and empirically, this assumption holds in a variety of settings.

The fact that $\boldsymbol{G}_{\boldsymbol{\Delta}}$ is symmetric and will often be very sparse in nature can be exploited when computing the eigendecomposition on line 3 of Algorithm 6, as we will show in the following section. Many modern software packages for scientific computing implement very efficient procedures specifically for such cases (e.g. SciPy [227]). Note that alternative algorithms to factorise $\boldsymbol{G}_{\boldsymbol{\Delta}}$ into lower-dimensional matrices exist, often relying on randomised sampling procedures [131, 56]. These algorithms are reportedly more efficient to compute than the traditional eigendecomposition, but often not geared specifically towards the high-dimensional yet sparse use-case we tackle in this work, or not equipped to exploit the symmetric structure that is typical for the Gramian. As they compute two dense matrices of $\boldsymbol{Q}$'s dimensions – their improvement in computation time comes with the cost of increased memory consumption. Furthermore, these methods are often focused on approximate matrix reconstructions whereas we are interested in an exact decomposition of the update to the Gramian. As the eigen-decomposition fulfils our needs, the study of alternative factorisation methods falls out of the scope of the present work.

Algorithm 6 Exact DYN-EASE$^{\text{R}}$

Input: $\hat{\boldsymbol{P}}_t, \mathscr{P}_{\boldsymbol{\Delta}}, \mathscr{L}_t$
Output: $\hat{\boldsymbol{P}}_{t+1}, \mathscr{L}_t + 1$.
1: $\boldsymbol{G}_{\boldsymbol{\Delta}}, \mathscr{L}_{t+1}$ = DYN-GRAM($\mathscr{P}_{\boldsymbol{\Delta}}, \mathscr{L}_t$) // (**Algorithm 5**)
2: k = estimate-rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$) // [117, 214]
3: $\Lambda, \boldsymbol{Q}$ = eigen-decomposition($\boldsymbol{G}_{\boldsymbol{\Delta}}, k$)
4: $\hat{\boldsymbol{P}}_{t+1} = \hat{\boldsymbol{P}}_t - \hat{\boldsymbol{P}}_t \boldsymbol{Q}(\boldsymbol{\Lambda}^{-1} + \boldsymbol{Q}^\intercal \hat{\boldsymbol{P}}_t \boldsymbol{Q})^{-1} \boldsymbol{Q}^\intercal \hat{\boldsymbol{P}}_t$
5: **return** $\hat{\boldsymbol{P}}_{t+1}$

Throughout this section, we have focused on DYN-EASE$^{\text{R}}$ as a general extension of EASE$^{\text{R}}$. Naturally, our approach is trivially extended to include CEASE$^{\text{R}}$, ADD-EASE$^{\text{R}}$ or a weight matrix $\boldsymbol{W}$ different from the identity matrix $\boldsymbol{I}$ as well, as these variants only change the input to Algorithms 5 and 6, but bear no impact on the procedures themselves.

Computational Complexity Analysis of Eigen-Decomposition

The computational complexity of EASE$^{\text{R}}$ is determined by the inversion of the Gramian, whereas the complexity of DYN-EASE$^{\text{R}}$ is dictated by that of the eigen-decomposition of the update to the Gramian. The computational complexity of matrix inversion, as well as that of solving the eigen-problem of a matrix, can be reduced to that of matrix multiplication [159, 104]. Given a square matrix of size $n \times n$, this is generally thought of as an $\mathscr{O}(n^3)$ problem. Nevertheless, specialised methods that provide improved bounds on the exponent exist, the most recent one being $\mathscr{O}(n^{2.37286})$ by [7].

In practice, it is easily seen that more efficient algorithms can be applied to specific cases instead of the general approach. Indeed, the inversion of a diagonal matrix consists of just n operations, and algorithms to multiply sparse matrices are often much more efficient than their dense counterparts. In what follows, we provide a brief theoretical analysis of the complexity of DYN-EASE$^{\text{R}}$, giving rise to an improved estimate for its computational complexity in practical settings. This bound explains the efficiency improvements of DYN-EASE$^{\text{R}}$ over EASE$^{\text{R}}$, and recovers the equivalence of eigen-decomposition to matrix inversion in the general case.

A first important thing to note is that the Gramian is symmetric, and so is $\boldsymbol{G}_{\boldsymbol{\Delta}}$. This allows us to use the iterative method proposed by [101] to compute its eigen-vectors and -values.[3] The core algorithm proposed by Lanczos consists of k steps – one per non-zero eigen-value-vector pair – which in turn consist of several vector and matrix manipulations. We refer the interested reader to an excellent analysis of the Lanczos algorithm provided by [157], showing *how* it works and *why* it converges. The computational complexity of every step in the method is determined by that of a matrix-vector product between the input $\boldsymbol{G}_{\boldsymbol{\Delta}}$ and an $|\mathscr{I}|$-dimensional vector. In the general case, such an operation is $\mathscr{O}(|\mathscr{I}|^2)$. In our specific case, however, $\boldsymbol{G}_{\boldsymbol{\Delta}}$ is

[3] In our experiments we use an efficient SciPy implementation of a variant called the Implicitly Restarted Lanczos Method [107, 227]; the analysis remains valid.

often of an extremely sparse nature. This allows us to describe the complexity of the product as $\mathcal{O}(m \cdot |\mathcal{I}|)$, where m is the average number of non-zero values in every column of $\boldsymbol{G}_{\boldsymbol{\Delta}}$. Repeating these steps for every non-zero eigen-value-vector pair yields a final computational complexity of $\mathcal{O}(k \cdot m \cdot |\mathcal{I}|)$. When we wish to do a full-rank update on a dense matrix (i.e. $k = m = |\mathcal{I}|$), this recovers the computational complexity of general matrix inversion: $\mathcal{O}(|\mathcal{I}|^3)$. In the cases where either the rank of the update is low ($k \ll |\mathcal{I}|$) or the update to the Gramian is highly sparse ($m \ll |\mathcal{I}|$), the eigen-decomposition will be most efficient and as a consequence, the performance benefits of DYN-EASE$^{\text{R}}$ over EASE$^{\text{R}}$ will be most apparent too. Note that although low-rankness and sparsity will often come in pairs in the practical settings we deal with, this does not have to be the case in general. As a counter-example: the identity matrix $\boldsymbol{I}$ is highly sparse yet full-rank.

Efficient Estimation and Upper Bounding of the Update's Rank

In order to compute the eigen-decomposition on line 3 of Algorithm 6, the numerical rank of $\boldsymbol{G}_{\boldsymbol{\Delta}}$ would need to be known a priori. Furthermore, as we have shown, the efficiency of the update procedure is highly dependent on the assumption that this rank is much smaller than the dimensionality of the Gram-matrix itself: $k \ll |\mathcal{I}|$. It is known that matrix ranks can be estimated efficiently through the use of randomised methods [117, 214]; when dealing with sparse and symmetric matrices, these methods tend to attain extremely efficient performance.[4] Being able to estimate rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$) of course does not guarantee that this quantity will be low. In practice, however, we notice that it is often the case. We can see that the rank of the update $\boldsymbol{G}_{\boldsymbol{\Delta}}$ depends on (1) the number of unique users in the update $\mathcal{P}_{\boldsymbol{\Delta}}$, denoted by $|\mathcal{U}_{\boldsymbol{\Delta}}|$, and (2) the average number of items in the *entire* history of these users: $|\mathcal{I}_{\mathcal{U}_{\boldsymbol{\Delta}}}|$. This can be intuitively seen from the fact that an index i, j in the Gramian matrix represents the number of co-occurrences between the items i and j in the dataset. As such, a new user-item interaction $(u, i) \in \mathcal{P}_{\boldsymbol{\Delta}}$ affects $\boldsymbol{G}_{i,j}, \forall\, j \in \mathcal{I}_u$.

Now, let $\boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot]}$ be the user-item matrix containing all (including historical) user-item interactions from only the users that appear in the update. This means we can rewrite the updated Gramian matrix as follows:

$$\begin{aligned}\boldsymbol{G}_{t+1} &= \boldsymbol{G}_t - \boldsymbol{X}_t^\top \boldsymbol{X}_t + \boldsymbol{X}_{t+1}^\top \boldsymbol{X}_{t+1} \\ &= \boldsymbol{G}_t - \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t}^\top \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t} + \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t+1}^\top \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t+1}.\end{aligned}$$

The update then becomes: $\boldsymbol{G}_{\boldsymbol{\Delta}} = \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t+1}^\top \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t+1} - \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t}^\top \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot],t}$.

Lemma 1. *Given a $|\mathcal{U}| \times |\mathcal{I}|$ user-item matrix $\boldsymbol{X}$, its Gramian matrix $\boldsymbol{G}$, and updates to $\boldsymbol{X}$; the rank of the update of the Gramian matrix $\boldsymbol{G}_{\boldsymbol{\Delta}}$ can be upper bounded by two times the number of unique, non-zero rows in $\boldsymbol{X}_{\boldsymbol{\Delta}}$:* $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}}) \leq 2|\mathcal{U}_{\boldsymbol{\Delta}}|$.

Proof. As the rank of a matrix is defined as its number of linearly independent row or column vectors, a (possibly loose) upper bound for rank($\boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot]}$) is given by its number of non-zero rows $|\mathcal{U}_{\boldsymbol{\Delta}}|$. Consequently, the rank of the Gramian matrix has the same bound: $\text{rank}(\boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot]}^\top \boldsymbol{X}_{[\mathcal{U}_{\boldsymbol{\Delta}},\cdot]}) \leq |\mathcal{U}_{\boldsymbol{\Delta}}|$. It is well known that the rank of the

[4] In the SciPy package for Python, an implementation of the randomised method presented by Liberty et al. can be found under scipy.linalg.interpolative.estimate_rank [117, 227].

Algorithm 7 Approximate DYN-EASE$^{\text{R}}$

Input: $\hat{\boldsymbol{P}}_t$, $\mathscr{P}_{\boldsymbol{\Delta}}$, $\mathscr{L}_t$, k
Output: $\hat{\boldsymbol{P}}_{t+1}$, $\mathscr{L}_t + 1$.
1: $\boldsymbol{G}_{\boldsymbol{\Delta}}, \mathscr{L}_{t+1}$ = DYN-GRAM($\mathscr{P}_{\boldsymbol{\Delta}}, \mathscr{L}_t$) // (**Algorithm 5**)
2: $\Lambda, \boldsymbol{Q}$ = truncated-eigen-decomposition($\boldsymbol{G}_{\boldsymbol{\Delta}}, k$)
3: $\hat{\boldsymbol{P}}_{t+1} = \hat{\boldsymbol{P}}_t - \hat{\boldsymbol{P}}_t \boldsymbol{Q}(\Lambda^{-1} + \boldsymbol{Q}^\top \hat{\boldsymbol{P}}_t \boldsymbol{Q})^{-1} \boldsymbol{Q}^\top \hat{\boldsymbol{P}}_t$
4: **return** $\hat{\boldsymbol{P}}_{t+1}$

sum of two matrices is less than or equal to the sum of the ranks of the individual matrices. Bringing those together, we have that $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}}) \leq 2|\mathscr{U}_{\boldsymbol{\Delta}}|$. □

This upper bound on $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}})$ holds for any update to $\boldsymbol{X}$. When users in the update are disjoint of those in $\boldsymbol{X}_t$, the bound can be tightened to $|\mathscr{U}_{\boldsymbol{\Delta}}|$. For general-purpose use cases, it is not be feasible to ensure that users in the update do not appear with partial histories in previous iterations of the model. For specific applications such as session-based recommendation, however, it is common practice to train models on the session-item matrix, which satisfies this assumption by definition [124].

Lemma 2. *Given a $|\mathscr{U}| \times |\mathscr{I}|$ user-item matrix $\boldsymbol{X}$, its Gramian matrix $\boldsymbol{G}$, and updates to $\boldsymbol{X}$ that only consist of adding new rows or altering previously zero-rows; the rank of the update of the Gramian matrix $\boldsymbol{G}_{\boldsymbol{\Delta}}$ can be upper bounded by the number of rows being added or altered:* $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}}) \leq |\mathscr{U}_{\boldsymbol{\Delta}}|$.

Proof. When the update only pertains to new users, this ensures that $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}}) = \text{rank}(\boldsymbol{X}_{\boldsymbol{\Delta}})$. Because $\text{rank}(\boldsymbol{X}_{\boldsymbol{\Delta}})$ is bounded by $|\mathscr{U}_{\boldsymbol{\Delta}}|$ per definition, so is $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}})$: $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}}) \leq |\mathscr{U}_{\boldsymbol{\Delta}}|$. □

We have provided bounds for $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}})$ by focusing on the number of users that have contributed interactions in the new batch of data that we wish to include into the model. Analogously, in some settings, it might be easier to bound the number of unique items that are being interacted with. In a news recommendation setting, for example, a new batch of data might consist of only a very limited number of items (in the order of hundreds) being read by a much higher number of users (hundreds of thousands). In this case, we can straightforwardly extend Lemmas 1 and 2 to bound the rank by the number of independent *columns* in $\boldsymbol{X}$ as opposed to its *rows*. The further reasoning and results follow trivially, bounding $\text{rank}(\boldsymbol{G}_{\boldsymbol{\Delta}})$ by $2|\mathscr{I}_{\boldsymbol{\Delta}}|$ and $|\mathscr{I}_{\boldsymbol{\Delta}}|$ respectively. Whereas the original EASE$^{\text{R}}$ approach and the need to iteratively retrain would make it a poor choice for applications with possibly vast item catalogues but smaller *active* item catalogues, such as catalogues of news articles, the presented upper bounds theoretically show why DYN-EASE$^{\text{R}}$ can provide an efficient updating mechanism.

Approximate DYN-EASE^R Updates via Truncated Eigen-Decomposition

Naturally, the rank of the update will not always be low in general recommendation use-cases. The easiest counter-example to think of is the case where we wish to include k user-item interactions that pertain to k new and unique users as well as k unique items. This will lead to a diagonal-like structure of $\boldsymbol{X}_{\boldsymbol{\Delta}}$ and $\text{rank}(\boldsymbol{X}_{\boldsymbol{\Delta}}) = k$, which is problematic for large values of k. However, it is also not hard to see that incorporating such a batch of data into our model will not affect any of our personalisation capabilities. Indeed, as EASE^R exploits signal from item co-occurrences, data where no item co-occurrences are present is practically useless, even though it is full-rank. Although this is a contrived example, it serves to illustrate that the rank of the update is not necessarily synonymous with its informational value.

In these cases, we can still resort to updating our model $\hat{\boldsymbol{P}}$ with a low-rank approximation of $\boldsymbol{G}_{\boldsymbol{\Delta}}$ without hurting the performance of the updated model. Instead of computing the rank and a full eigen-decomposition of the Gramian as shown in Algorithm 6, we can choose the rank k at which we wish to truncate, and update $\hat{\boldsymbol{P}}$ with a low-rank approximation $\tilde{\boldsymbol{G}}_{\boldsymbol{\Delta}}$ instead of the real thing. The resulting algorithm is shown in Algorithm 7, and it provides a tunable trade-off between the exactness of the acquired solution and the efficiency of incremental updates.

Interestingly, this type of approximate update is closely related to yet another extension of the SLIM paradigm: Factored Item Similarity Models (FISM) [90]. In FISM, the similarity matrix $\boldsymbol{S}$ is modelled as the product of two lower-dimensional latent factor matrices. The resulting low-rank model is shown to be increasingly effective as the sparsity in the user-item interactions it learns from increases, highlighting that this type of approximation does not necessarily imply a decrease in recommendation accuracy. In approximate DYN-EASE^R, we do not directly model the similarity matrix $\boldsymbol{S}$ as factorised, but we update $\boldsymbol{S}$ with a factorised version of the update to the Gramian $\boldsymbol{G}_{\boldsymbol{\Delta}}$. Factorised models such as FISM or approximate DYN-EASE^R also bear resemblance to models that are often used in natural language processing applications. Indeed, the well-known WORD2VEC algorithm to train word embeddings implicitly learns to factorise a matrix holding the (shifted positive) pointwise mutual information between word-context pairs [141, 110].

Although our motivations for approximate DYN-EASE^R are rooted in improving the computational cost of exact DYN-EASE^R, the advantages of transitivity that come from adopting low-rank representations can significantly impact recommendation performance as well. Imagine items $a, b, c \in \mathcal{I}$ where (a, b) and (b, c) co-occur in the training data of user histories, but (a, c) does not. Full-rank EASE^R cannot infer a correlation between a and c in such a setting, whereas low-rank models can learn a latent factor that unifies a, b and c. This explains the advantage that low-rank models have in sparse data environments. For further insights on the advantages, differences and analogies between full-rank and low-rank models, we refer the interested reader to the work of Van Balen and Goethals [216].

As we are factorising $\boldsymbol{G}_{\boldsymbol{\Delta}}$ by its truncated eigen-decomposition, we are guaranteed to end up with the optimal rank-k approximation with respect to the mean squared error between $\tilde{\boldsymbol{G}}_{\boldsymbol{\Delta}}$ and $\boldsymbol{G}_{\boldsymbol{\Delta}}$. Naturally, with the highly sparse nature of $\boldsymbol{G}_{\boldsymbol{\Delta}}$, this optimal approximation will focus on reconstructing entries with large values, and rows or columns with many non-zero values. This corresponds to focusing on the items that occur most often in the new incoming batch of user-item interactions

Name	nnz(**X**)	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	$\overline{\lvert\mathcal{U}_i\rvert}$	$\overline{\lvert\mathcal{I}_u\rvert}$	Timespan (δ)
MovieLens-25M (ML-25M)	16M	162k	30k	524	96	25 years
YooChoose	10M	1.3M	28k	359	8	6 months
RetailRocket	593k	115k	49k	12	5	4 months
Adressa	39M	1.4M	54k	725	28	3 months
Microsoft News (MIND)	16M	696k	62k	266	24	5 days
SMDI	738k	10k	7k	41	31	4 months

Table 3.1: Datasets we adopt throughout the experiments presented in this work, along with their source and summary statistics that describe the user-item interactions and their sparsity. nnz(·) denotes the number of non-zero entries.

$\mathscr{P}_\Delta$. Because of this, we can expect approximate DYN-EASE$^{\text{R}}$ to favour recently popular items, which can give an additional performance boost in the right application areas. Nevertheless, an in-depth discussion or validation of the efficacy of factorised EASE$^{\text{R}}$-like models falls outside the scope of this work, as we focus on the efficiency with which the model can be updated. If the cut-off rank k is lower than the true rank of the update, approximate DYN-EASE$^{\text{R}}$ guarantees an improvement in terms of the computational complexity of the update procedure.

3.4 Experimental Results and Discussion

The goal of this section is to validate that the methods we proposed in earlier sections of this manuscript work as expected, and to investigate whether expectations grounded in theory can be substantiated with empirical observations. Concretely, the research questions we wish to answer are the following:

RQ1 Can exact DYN-EASE$^{\text{R}}$ provide more efficient model updates in comparison with iteratively retrained EASE$^{\text{R}}$?

RQ2 Can our theoretical analysis on the correlation between rank($\boldsymbol{G}_\Delta$) and the runtime of DYN-EASE$^{\text{R}}$ set realistic expectations in practice?

RQ3 Do the phenomena we describe for bounding rank($\boldsymbol{G}_\Delta$) occur in real-world session-based or news recommendation datasets?

RQ4 Can approximate DYN-EASE$^{\text{R}}$ provide a sensible trade-off between recommendation efficiency and effectiveness?

Table 3.1 shows the publicly available datasets we use throughout our experiments in an attempt to provide empirical answers to the above-mentioned research questions. The well-known MovieLens dataset [57] consists of explicit ratings (on a 1–5 scale) that users have given to movies, along with the time of rating. We drop ratings lower than 3.5 and treat the remainder as binary preference expressions.

Additionally, we only keep users and items that appear at least 3 times throughout the dataset. This type of pre-processing is common, and ensures we are left with positive preference expressions that carry enough signal for effective personalisation [116, 9]. We take the newest and largest variant of the dataset as our starting point: MovieLens-25M.

Many recommender systems applications are based on shorter browsing sessions rather than full user histories that might span years [124]. As laid out in Section 3.3, these set-ups can be especially amenable to our approach, as the adoption of these shorter sessions instead of longer user histories naturally decreases the rank of the update to the Gramian. We adopt two well-known datasets for session-based recommender systems: the YooChoose dataset, released in the context of the 2015 ACM RecSys Challenge [12]; and the RetailRocket dataset [91]. These datasets consist of implicit feedback (clicks) from users on retail products, and we compute the 3-core for users and items in the same manner we did for MovieLens-25M, after removing repeated user-item interactions. To validate our intuitions regarding DYN-EASE$^{\text{R}}$ and the rank of the Gramian in news recommendation setups, we use the Adressa and Microsoft News datasets (MIND) [53, 235]. These datasets contain implicit feedback inferred from browsing behaviour on news websites; we pre-process them anaologously to the other datasets.

Some datasets have prohibitively large item catalogues for EASE$^{\text{R}}$ to compute the inverse Gramian at once. However, the large majority of items are often at the extreme end of the so-called "long tail", only being interacted with once or twice. We prune these items to keep the EASE$^{\text{R}}$ computation feasible but still highlight the advantages of DYN-EASE$^{\text{R}}$.

Note that these pruning operations on rare items significantly cut down computation time for EASE$^{\text{R}}$ (directly dependent on $|\mathcal{I}|$), but do not pose an unfair advantage for DYN-EASE$^{\text{R}}$. Items that appear just once in the dataset blow up the size of the Gramian, but do not significantly impact the rank of the Gramian updates. Indeed, in these situations we get that $k \ll |\mathcal{I}|$, and the computational advantages of DYN-EASE$^{\text{R}}$ over EASE$^{\text{R}}$ become even more pronounced. We adopt such pruning as it is common practice and keeps the computational needs for reproducing our experiments reasonable. The reason we do not further explore other commonly known datasets such as the Million Song Dataset (MSD) [15], is that these do not include logged timestamps that indicate *when* the user-item interactions occurred. Because of this, they are unsuited for evaluating a realistic scenario where models are incrementally retrained over time.

The final dataset we adopt is the SuperMarket Dataset with Implicit feedback (SMDI) introduced by Viniski et al. [226]. Because this dataset has a comparatively small item catalogue, the computation time for all EASE$^{\text{R}}$ variants is in the order of seconds and largely dominated by variance and system overhead. We adopt the SMDI dataset to study the recommendation performance of approximate DYN-EASE$^{\text{R}}$, as it exhibits a distribution shift that is largely absent in the other datasets we consider.

To foster the reproducibility of our work, all source code for the experiments we have conducted is publicly available at github.com/olivierjeunen/dynamic-easer/, under an open-source license. All code is written in Python 3.7 using SciPy [227]. Reported runtimes are wall-time as measured using an Intel Xeon processor with 14 cores. The rest of this section is structured to follow the research questions laid out

above.

Efficiency of exact DYN-EASE^R (RQ1)

To verify the gains in runtime from exact DYN-EASE^R over iteratively retrained EASE^R, we chronologically split the user-item interactions based on a fixed timestamp t, yielding all user-item interactions up to t, and all those after t. The Microsoft News Dataset comes with user's reading histories and clicks on shown recommendations, but the former type of interaction does not include timestamps. Because of this, we treat these historical interactions as "early data" included in the original EASE^R computation, and incorporate the timed clicks chronologically into DYN-EASE^R in the procedure described below.

We train an EASE^R model on the early batch, and report the runtime in seconds needed for this computation. This operation is repeated over 5 runs, and we report a 95% Gaussian confidence interval. As new incoming user-item interactions do not affect the dimension of the Gramian matrix that needs to be inverted, the runtime needed to compute EASE^R remains fairly constant when adding new user-interactions.

Over the newer batch of data, we employ a non-overlapping sliding window technique that chronologically generates batches of data to be included in the existing model via our proposed exact DYN-EASE^R procedure. The size of this window δ is varied to study the effects on the runtime of DYN-EASE^R. Larger values of δ imply larger update batch sizes, which will often lead to an increase in rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$). Naturally, when δ becomes too large, a point is reached where the overhead induced by our incremental updating method becomes prohibitively large, and it becomes favourable to fully retrain the EASE^R model. Sensible values of δ come with a restriction: when the runtime of the model update is larger than δ, this would indicate that the procedure cannot keep up with incoming data in real-time. We do not encounter this issue for any of the values of δ explored in our experiments - suggesting that DYN-EASE^R can be a good fit for various configurations.

Figure 3.1 visualises the resulting runtimes from the procedure laid out above, on all five considered datasets. The time for the sliding window increases over the x-axis, and runtime for varying values of δ is shown on the y-axis. The explored values of δ differ based on the dataset and use-case: for the 25-year spanning MovieLens dataset, daily updates might be sufficient; for the 3-month spanning news recommendation dataset Adressa, more frequent 5-minute updates might be more appropriate, to keep up with the highly dynamic nature of the environment.

We included values of δ that push the runtime for DYN-EASE^R up to that of EASE^R to highlight the limitations of our approach. Provided that the computing power and infrastructure is available, however, δ can be decreased to bring DYN-EASE^R's runtime into the desirable range. Note that this limitation on δ is general for online learning approaches from user-item interactions, and not specific to the methods we propose in this work.

From the runtime results, we can observe that our proposed method entails significant performance improvements compared to iterative model retraining, for a wide range of settings. Over all datasets, we observe a clear trend towards lower runtimes for shorter sliding windows and more frequent updates, as is expected from our theoretical results.

As the MovieLens-25M dataset spans several decades, the amount of new user-item interactions to be incorporated on a daily basis remains modest. Exploring lower values of δ would not provide any additional insights into the performance of DYN-EASE$^{\text{R}}$ because of this. As a consequence, we obtain a clean separation between the runtime for DYN-EASE$^{\text{R}}$ on batches of different length.

The remaining four datasets represent session- and news-based recommendation environments, which are known to be much more fast-paced and dynamic. Because we focus on smaller sliding window lengths δ here, we clearly see daily seasonal patterns emerging. Indeed, DYN-EASE$^{\text{R}}$ runtime peaks coincide with peaks in website traffic. As the rank of the update is typically correlated with the number of user-item interactions in the update, this phenomenon is to be expected. It highlights that DYN-EASE$^{\text{R}}$ is able to effectively target those model parameters that need updating, and does not spend unnecessary computing cycles on unchanged parts of the model. Note that δ does not need to be a fixed constant in real-world applications. An effective use of computing power might decrease and increase δ during traffic peaks and valleys respectively.

Correlating the rank of the update with the runtime of DYN-EASE$^{\text{R}}$ (RQ2)

The runtime of the incremental updates shown in Figure 3.1 is visualised against the rank of the updates in Figure 3.2. We clearly observe a strong correlation between the rank of the update to the Gramian and the runtime of DYN-EASE$^{\text{R}}$, with a trend that is consistent over varying values of δ.

We fit a polynomial of the form $f(x) = a \cdot x^b + c$ on a randomly sampled subset of 90% of measurements, and assess its performance in predicting the runtime for DYN-EASE$^{\text{R}}$ based on rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$) on the remaining 10% of the measurements. Table 3.2 shows the optimal parameters, the number of samples (runtime measurements) and the Root Mean Squared Error (RMSE) on the test sample for every dataset. Figure 3.2 qualitatively shows that we are able to to predict the runtime for DYN-EASE$^{\text{R}}$ updates with reasonable accuracy when we know the rank of the update. Combined with the bounds on this quantity laid out in Section 3.3, we can use this to set an expected upper bound for the computation time of our incremental updates through DYN-EASE$^{\text{R}}$. Table 3.2 quantitatively shows the magnitude of the errors, reassuring our qualitatively obtained insights. Note that whereas the *absolute* RMSE increases with the datasets with larger item catalogues, the *relative* error of the model remains fairly constant. Indeed, a mean error of 5 seconds on a prediction of 10 seconds is not equivalent to being 5 seconds off when the order of magnitude is 1000 seconds. These empirical observations together with the theoretical analysis presented in Section 3.3 highlight the efficiency and favourable scalability of the proposed DYN-EASE$^{\text{R}}$ procedure.

Analysing bounds for the rank of the update (RQ3)

Figure 3.3 shows the rank of the incremental updates from Figure 3.1 compared to summary statistics for the batches of user-item interactions. This visualisation shows the effectiveness of the upper bounds laid out in Section 3.3 in order to assess their utility and provide a better understanding of the underlying dynamics for every dataset.

Dataset	RMSE	N	a	b	c
MovieLens-25M (ML-25M)	2.34	1 457	3.59e-4	1.83	6.28
YooChoose	2.01	4 200	5.78e-4	1.79	9.01
RetailRocket	2.11	1 302	1.43e-3	1.72	18.81
Adressa	5.88	2 580	1.03e-3	1.75	26.35
Microsoft News (MIND)	8.77	3 000	5.98e-3	1.52	28.32

Table 3.2: Resulting polynomial model to predict runtime from rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$), along with the Root Mean Squared Error (RMSE) it attains and the number of observations it was fitted on. We observe that the models attain good performance in terms of RMSE, indicating that they can set realistic expectations for DYN-EASE$^{\text{R}}$ runtime. Furthermore, the exponent b in the model is lower than quadratic, indicating good scaling properties for DYN-EASE$^{\text{R}}$ with respect to rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$).

We observe that both for general purpose MovieLens-25M and the session-based datasets, the user-focused bound performs reasonably well in approximating the rank of the update to the Gramian. This is in line with our theoretical expectations, and confirms that the number of unique users in any given batch of user-item interactions are the main driving factor for rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$). We further see that the upper bound becomes looser as the number of unique users grows. This as well is expected behaviour, as it becomes less likely for new users' behaviour to be linearly independent of other users in the batch as the batch size grows. As mentioned in 3.3, the upper bound of $2|\mathscr{U}|$ could be tightened to $|\mathscr{U}|$ if we did not perform a hard split on time but rather divided user sessions into a "finished" and "ongoing" set. This phenomenon occurs naturally for the YooChoose dataset, where we clearly see that the $2|\mathscr{U}|$ bound is much looser. Note that the tight bounds for Movielens might change if this dataset would include timestamps for item consumption rather than rating, as the majority of users might watch a smaller set of current series or movies. Such a recency bias would decrease the *active* item catalogue, favouring the item-based bounds.

In contrast with the user-focused datasets, the bound on the number of unique items is much tighter for the news datasets, providing an almost perfect approximation in many cases. This confirms our intuition that the rank of the update in these settings is fully determined by the number active items in the catalogue and virtually independent of the number of users or interactions in a given batch. This in turn makes these environments especially amenable to our DYN-EASE$^{\text{R}}$ approach.

The number of unique users or items in a batch can give rise to reasonably tight upper bounds on the rank of the update in realistic scenarios, using real-world datasets. The absolute number of user-item interactions $|\mathscr{P}|$ provides another (impractical) bound on the rank of the update; indeed, in a worst-case scenario, every user-item interaction would pertain to a unique user and a unique item. We include the visualisation of the relation between rank($\boldsymbol{G}_{\boldsymbol{\Delta}}$) and $|\mathscr{P}|$ to intuitively show that our proposed approach scales favourably with respect to the size of the data, a property that is most appreciated in highly dynamic environments with ever-growing dataset sizes.

Efficiency and effectiveness of approximate DYN-EASE^R (RQ4)

Finally, we wish to validate the efficiency and efficacy of approximate updates to EASE^R-like models. Specifically, we wish to understand the trade-off between runtime and recall for models that are iteratively updated as new data comes in. We report experimental results for runtime and recommendation accuracy for both the Adressa and SMDI datasets. This experiment is not repeated on the other datasets, as they do not favour this type of experimental evaluation procedure. MovieLens-25M spans a too long time period, and we observe insignificant effects of model retraining on recommendation accuracy. YooChoose and RetailRocket focus on shorter user sessions, which are also unfavourable for SW-EVAL to reach statistically significant conclusions. Lastly, the Microsoft News Dataset contains bandit feedback, which is different to the organic user-item interactions we tackle in this work. This was not an issue when evaluating models' computational cost, but is prohibitive to properly evaluate recommendation accuracy in a common manner.

To illustrate the advantages of approximate DYN-EASE^R, we make use of the Sliding Window Evaluation (SW-EVAL) technique [80, 75]. We train a model on all user-item interactions that have occurred up to time t. For a fixed sliding window width δ_{update}, we periodically update the model with new incoming data, both for the exact and approximate DYN-EASE^R variants. A concurrent sliding window with width δ_{eval} dictates the evaluation period where every competing model is evaluated on its ability to predict with which items users interacted with next. This experimental procedure is formalised in Algorithm 8. We set $\delta_{\text{update}} = 60\text{min}$ and $\delta_{\text{eval}} = 120\text{min}$ for the Adressa dataset and evaluate over the final 24 hours, and $\delta_{\text{update}} = 6\text{h}$ and $\delta_{\text{eval}} = 3\text{d}$ for the final 120 days of the SMDI dataset. This difference in order of magnitude is to keep the overall runtime of the experiments reasonable, and to ensure statistically significant results from sufficiently large evaluation sample sizes.

Computation time for approximate DYN-EASE^R

Figure 3.4 shows computation time for exact DYN-EASE^R, as well as several approximate model variants with varying cut-off ranks k. In terms of runtime improvements, we observe very favourable results for approximate updates. As is expected, the computational cost of DYN-EASE^R's updates can largely be attributed to the computation of all eigen-pairs, and limiting the rank has a significant impact on the efficiency of said updates. At cut-off rank $k = 250$, the computational cost for the updates is decreased by a factor 3 or 65%.

As we have mentioned above, the computation time for all EASE^R variants on the SMDI dataset is in the order of seconds and largely dominated by variance and system overhead. As a result, runtime results on this dataset do not provide significant insights, and we do not report them.

Recommendation accuracy for approximate DYN-EASE^R

Figure 3.5 visualises Recall@K for $K \in \{1, 5, 10, 20\}$ over time on the Adressa and SMDI datasets. The SMDI dataset has large variance on the number of active users over time, which heavily influences the statistical significance of some of the evaluation results, as they are based on insufficient samples. We do not include evaluation results where the evaluation set consisted of less than 100 users. The denominator

Dataset	Algorithm	k	Runtime (s)	Δ %	R@1 (%)	Δ %	R@5 (%)	Δ %	R@10 (%)	Δ %	R@20 (%)	Δ %
Adressa	EASE^R (*stale*)		–	–	0.30	-25.6%	0.67	-33.0%	1.07	-34.0%	1.73	-36.0%
	DYN-EASE^R (*exact*)		118s	0%	0.40	0%	1.00	0%	1.64	0%	2.72	0%
	DYN-EASE^R (*approx.*)	1000	116s	-2%	0.4	0%	1.00	0%	1.64	0%	2.72	0%
		500	68s	-43%	0.40	0%	1.00	0%	1.64	0%	2.73	0%
		250	45s	-62%	0.40	0%	1.00	0%	1.64	0%	2.73	0%
		50	29s	-75%	0.39	-3.9%	0.99	-0.8%	1.62	-0.9%	2.71	-0.7%
		5	28s	-76%	0.56	+38.9%	1.21	+21.0%	1.87	+14.0%	2.96	+7.7%
		1	27s	-77%	0.67	+67.0%	1.30	+30.0%	1.90	+16.0%	2.87	+5.3%
SMDI	EASE^R (*stale*)		–	–	9.57	-6.8%	8.52	-16.0%	9.87	-16.0%	12.73	-18.0%
	DYN-EASE^R (*exact*)		6s	–	10.27	0%	10.13	0%	11.68	0%	15.53	0%
	DYN-EASE^R (*approx.*)	100	–	–	10.20	-0.7%	10.14	+0.1%	11.64	-0.4%	15.51	-0.1%
		50	–	–	9.96	-3.0%	10.01	-1.0%	11.56	-1.0%	15.42	-0.7%
		25	–	–	9.79	-4.6%	9.90	-2.3%	11.46	-1.9%	15.25	-1.8%
		5	–	–	10.28	+0.1%	10.01	-1.2%	11.52	-1.4%	15.28	-1.4%
		1	–	–	10.39	+1.2%	10.21	+0.8%	11.60	-0.7%	15.19	-2.2%

Table 3.3: Runtime and recommendation accuracy measurements on the Adressa and SMDI datasets, for the experimental procedure laid out in Algorithm 8. The R@K column shows measurements for Recall@K as described in the text. We compare exact DYN-EASE^R with approximate variants at varying cut-off ranks k, and observe favourable trade-offs.

for our Recall measure is $\min(K, |\mathscr{I}_u^{\text{test}}|)$ instead of the number of held-out items $|\mathscr{I}_u^{\text{test}}|$, to ensure that a perfect value of 1 can theoretically be attained at all cut-offs k.

Additional to several approximate model variants, we include recommendation accuracy results for a model that is not updated over time, yielding a lower bound on the accuracy we can expect from approximate updates. On the Adressa dataset, we observe that such a model quickly deteriorates over time. This is to be expected, as the Adressa dataset and news recommendation application are heavily biased towards recent items and interactions. This bias is less clear on the SMDI dataset at lower cut-off ranks K, but clearly manifests itself for Recall@20 near the end of the evaluation period.

For both datasets, we observe that the accuracy of approximate DYN-EASE$^{\text{R}}$ variants for high values of k, is statistically insignificantly different from exact DYN-EASE$^{\text{R}}$. This highlights that the N-fold improvement in terms of computational cost can come with a negligible impact on recommendation accuracy, showing the advantages of approximate computations. For Adressa, we observe a statistically significant improvement over exact DYN-EASE$^{\text{R}}$ for low values of k. This can be attributed to the reasons laid out in Section 3.3, as the low-rank approximation handles sparsity, transitivity, and favours recently popular items. These model characteristics are highly favourable in news recommendation settings – but might have smaller influence on supermarket data. Nevertheless, the results highlight that many efficient low-rank updates can yield highly competitive models compared to more costly full-rank updates.

All runtime and recommendation accuracy measurements are aggregated in Table 3.3, providing further insights on the trade-off between runtime and recommendation accuracy for approximate DYN-EASE$^{\text{R}}$. We denote the Recall@K measure as R@K for improved spacing. On the SMDI dataset, the differences in recommendation accuracy among exactly or approximately update model variants were not found to be statistically significant at the $p = 0.05$ level. The differences between the stale EASE$^{\text{R}}$ and DYN-EASE$^{\text{R}}$ models are significant.

The size of the Adressa dataset yields more statistical power, and both the differences between stale EASE$^{\text{R}}$ and DYN-EASE$^{\text{R}}$ and those between exact DYN-EASE$^{\text{R}}$ and approximate DYN-EASE$^{\text{R}}$ with $k \in \{1, 5\}$ were found to be statistically significant.

3.5 Conclusions

Linear item-based models are an attractive choice for many collaborative filtering tasks due to their conceptual simplicity, interpretability, and recommendation accuracy. Recent work has shown that the analytical solution that is available for ridge regression can significantly improve the scalability of such methods, with a state-of-the-art algorithm called EASE$^{\text{R}}$ [200]. EASE$^{\text{R}}$ consists of a single matrix inversion of the Gramian. As its computational complexity does not rely on the number of users or even the number of user-item interactions in the training set, it is particularly well suited to use-cases with many users or interactions, with the sole constraint that the size of the item catalogue is limited.

When deployed in real-world applications, models often need to be periodically recomputed to incorporate new data and account for newly available items and

shifting user preferences, as well as general concept drift. Iteratively retraining an EASE^R-like model from scratch puts additional strain on such real-world applications, putting a hard upper limit on the frequency of model updates that can be attained, and possibly driving up computational costs. This especially limits the application of EASE^R in domains where item recency is an important factor deciding on item relevance – such as in retail or news recommendation.

In this work, we propose a novel and exact updating algorithm for embarrassingly shallow auto-encoders that combines parts of the Dynamic Index algorithm [83] and the Woodbury matrix identity [55]: Dynamic EASE^R (DYN-EASE^R). We have provided a thorough theoretical analysis of our proposed approach, highlighting in which cases it can provide a considerable advantage over iteratively retrained EASE^R, and in which cases it does not. These theoretical insights are corroborated by empirical insights from extensive experiments, showing that DYN-EASE^R is well suited for efficient and effective online collaborative filtering in various real-world applications.

DYN-EASE^R exploits the sparse and symmetric structure of the Gramian to efficiently compute the eigen-decomposition of the Gramian update. When the rank of the update is large, however, this operation can still become prohibitively expensive. To mitigate this problem, we have additionally proposed an approximate DYN-EASE^R variant that uses a low-rank approximation of the Gramian update as opposed to its exact decomposition. Empirical results highlight further efficiency improvements at a small cost for recommendation accuracy. Our work broadens the scope of problems for which item-based models based on ridge regression are an appropriate choice in practice. To foster the reproducibility of our work, the source code for all our experiments is publicly available under an open-source license at github.com/olivierjeunen/dynamic-easer.

A promising area for future work is to further improve DYN-EASE^R's computational efficiency by looking at alternative (approximate) matrix decompositions that exploit efficient random sampling [56, 131], as the bottleneck of our current approach lies in the computation of the exact eigen-decomposition of the update to the Gramian. Furthermore, we would like to explore applications of our efficient incremental updating scheme to more general multi-label regression tasks beyond the collaborative filtering use-case we tackle in this work.

Reflections

This Chapter has provided an elegant solution for incremental updating of closed-form regression models for collaborative filtering. Although effective and efficient, this model class only represents a fraction of those available. In many real-world applications, the size of the item catalogue will be prohibitively large for item-based models to be an appropriate choice. Keeping this in mind, the impact of our proposed dynamic updating algorithm might remain fairly limited.

Thoroughly analysing and characterising under which circumstances either the closed-form solution, or gradient-descent-based alternatives would be more efficient, is an interesting line of future work that could crystallise *when* closed-form models are an appropriate choice.

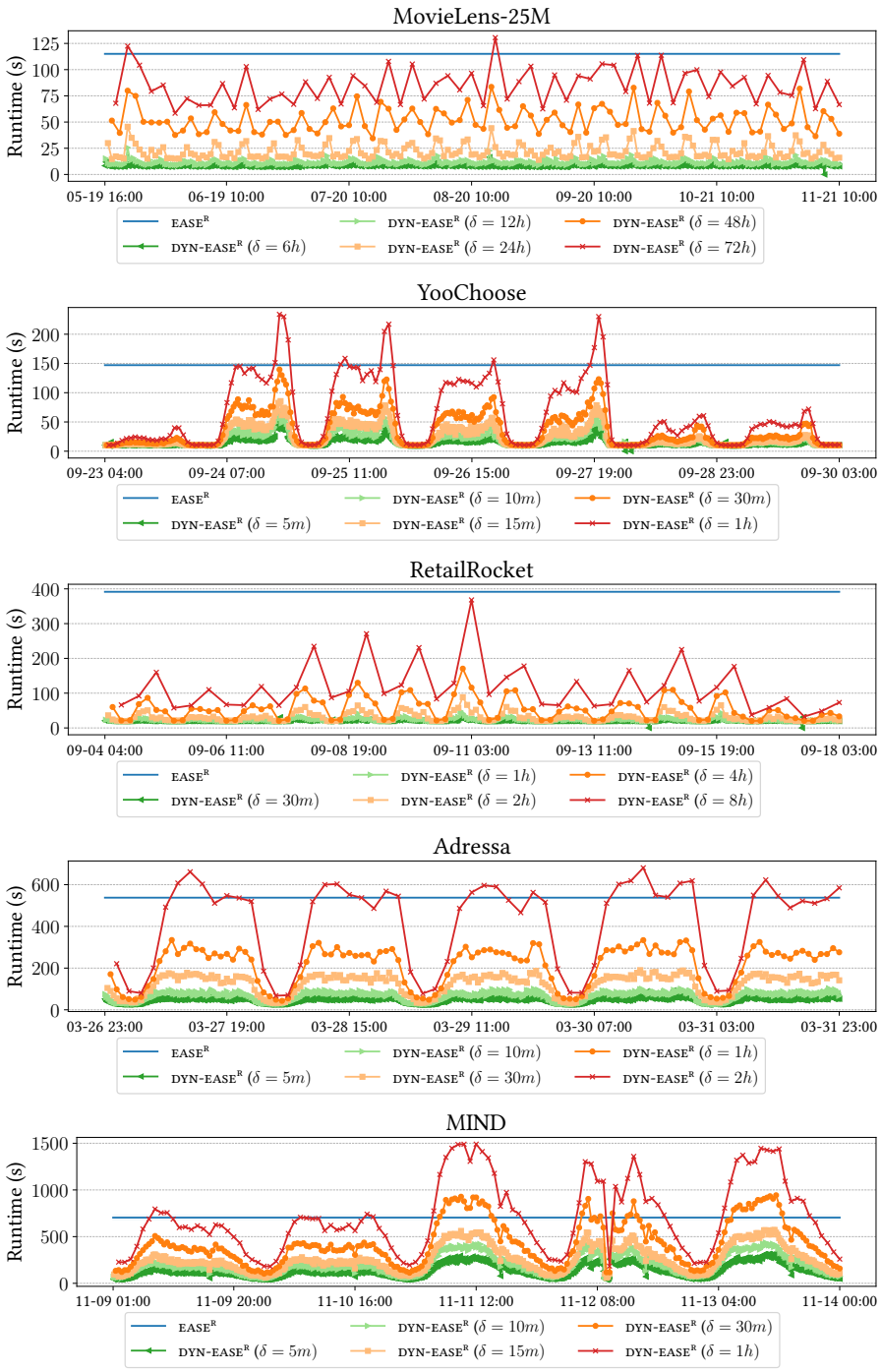


Figure 3.1: Runtime results for DYN-EASE^R updated with different intervals (sliding window width δ), as compared to iteratively retrained EASE^R over the final N days of the datasets.

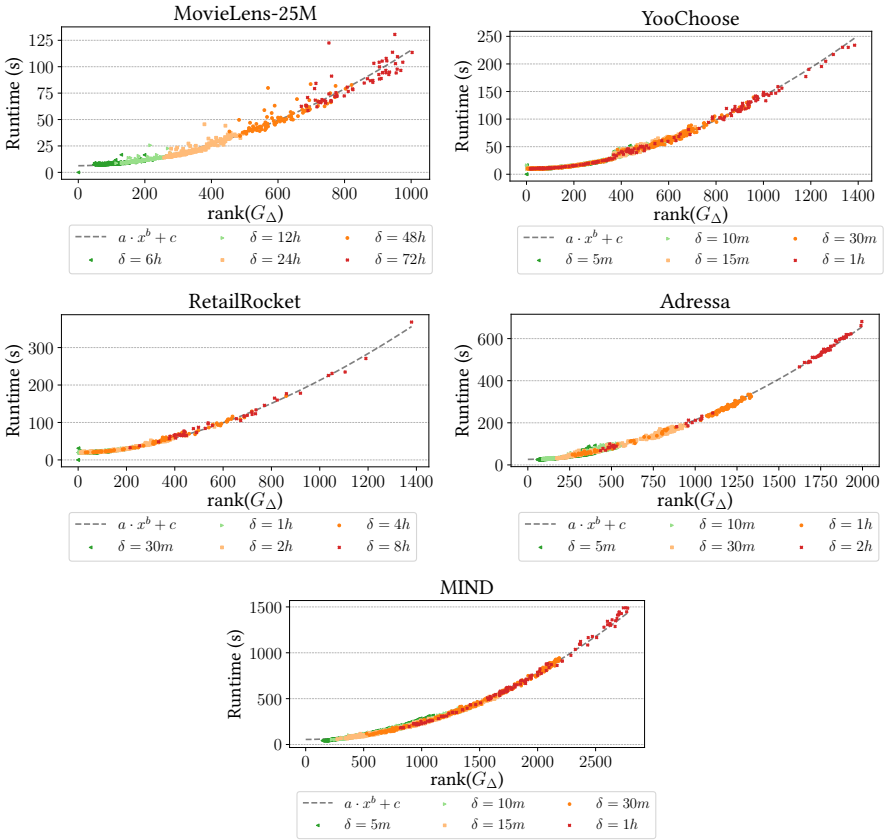


Figure 3.2: Runtime for incremental updates from Figure 3.1 plotted against the rank of the update to the Gramian matrix $\boldsymbol{G_\Delta}$. We observe a strong correlation between higher values of rank($\boldsymbol{G_\Delta}$) and runtime, as well as a correlation between δ and higher rank($\boldsymbol{G_\Delta}$). This result highlights that bounding rank($\boldsymbol{G_\Delta}$) can give realistic expectations for DYN-EASE^R efficiency in practice.

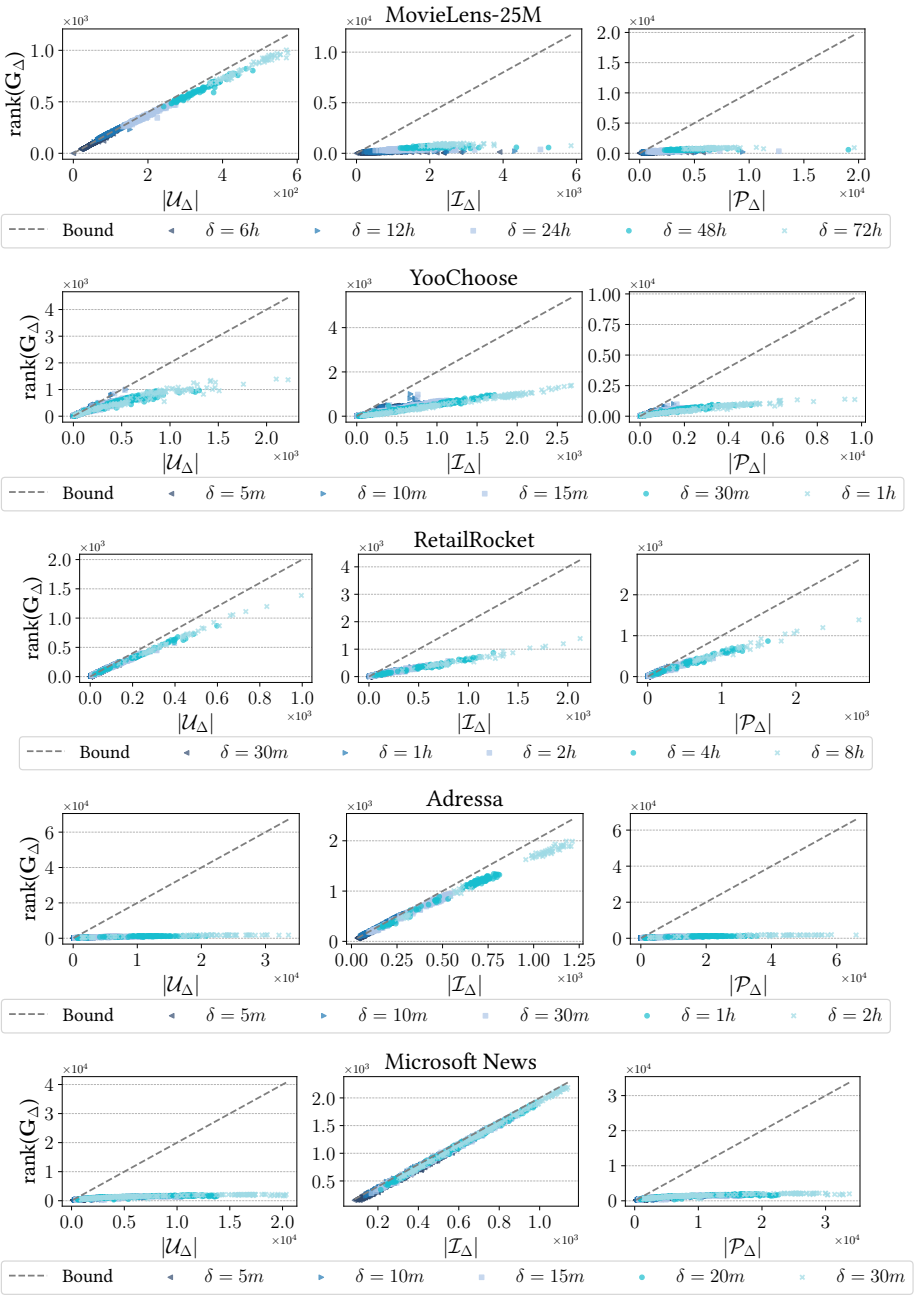


Figure 3.3: Upper bounds for the batches of incremental updates from Figure 3.1 plotted against the rank of the update to the Gramian matrix $\boldsymbol{G}_{\boldsymbol{\Delta}}$. We observe that different applications that imply different data characteristics bound the rank of the update in different ways.

Algorithm 8 Sliding Window Evaluation Procedure

Input: Pageviews $\mathscr{P}$, evaluation period timestamps (t, t_{max}), update intervals δ_{update}, evaluation sliding window width δ_{eval}, list K of cut-off ranks k to consider.

Output: Recommendation accuracy measurements $\mathscr{R}$.

$\boldsymbol{P}_t := \text{EASE}^{\text{R}}(\mathscr{P}_t)$
for $k \in K$ **do**
  $\boldsymbol{P}_{k,t} := \boldsymbol{P}_t$
$t' := t + \delta_{\text{update}}$
while $t' < t_{\text{max}}$ **do**
  $\boldsymbol{P}_{t'} := \text{EXACT DYN-EASE}^{\text{R}}(\boldsymbol{P}_{t'-\delta_{\text{update}}}, \mathscr{P}_{(t'-\delta_{\text{update}}, t')})$
  for $k \in K$ **do**
    $\boldsymbol{P}_{k,t'} := \text{APPROXIMATE DYN-EASE}^{\text{R}}(\boldsymbol{P}_{k,t'-\delta_{\text{update}}}, \mathscr{P}_{(t'-\delta_{\text{update}}, t')}, k)$
  if $(t' - t) \mod \delta_{\text{eval}} = 0$ **then**
    $\mathscr{R} \leftarrow \text{SW-EVAL}(\boldsymbol{P}_t, \boldsymbol{P}_{t'}, \boldsymbol{P}_{k,t'}, \mathscr{P}_{(t', t'+\delta_{\text{eval}})})$
  $t' := t' + \delta_{\text{update}}$
return $\mathscr{R}$

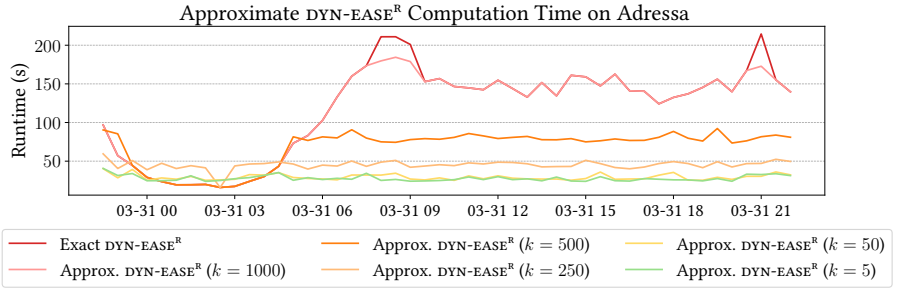


Figure 3.4: Runtime results for exact and approximate DYN-EASE$^{\text{R}}$ variants, with varying cut-off ranks k. We observe a quick and steady decline in computation time needed for lower values of k, which can be attributed to less computation time spent finding eigen-pairs.

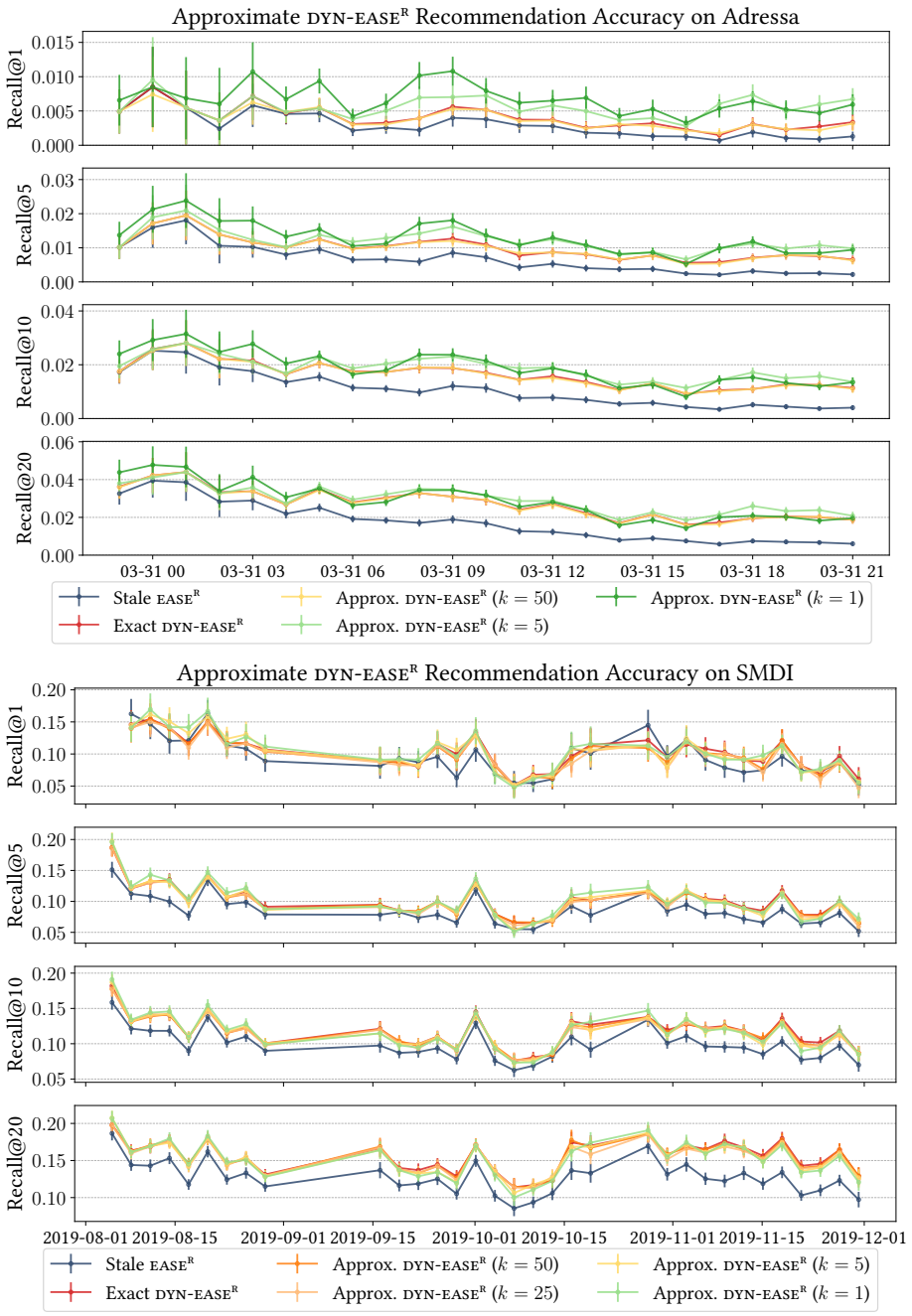


Figure 3.5: Recommendation accuracy results for exact and approximate DYN-EASE^R variants, with varying cut-off ranks k. We additionally report results for a stale EASE^R model that does not incorporate new user-item data over time.

Part II

Offline Evaluation Methods

Our comforting conviction that the world makes sense rests on a secure foundation: our almost unlimited ability to ignore our ignorance.

— Daniel Kahneman

CHAPTER 4

Fair Offline Evaluation with Missing-Not-At-Random Data

Recommender systems are traditionally evaluated using historical data, partitioned into a training- and test-set. The system is trained on the user-item interactions available in the training set and evaluated on its performance to predict which interactions are part of the test set. Leave-One-Out Cross-Validation (LOOCV) is a commonly recurring evaluation procedure, widely used to present novel algorithms as the state-of-the-art. However, the temporal aspect that is inherent to many recommender system use cases is entirely neglected with this technique, as well as potential biases in the data (i.e. interactions are Missing-Not-At-Random (MNAR)).

In this paper we propose and experimentally validate an alternative method to perform offline evaluation using real-world data from a live recommender system. Our novel approach adheres to the aspects that are inherent to web-based recommender systems in e.g. e-commerce much more tightly than LOOCV. Experimental results indicate that LOOCV is prone to overestimate model performance in general, underestimate the power of popularity-based baselines, and generally rank algorithms differently than our methodology. Furthermore, we study the impact of live recommendation algorithms in place during the time of data gathering on the offline evaluation of other algorithms on said data. We experimentally validate that such impact is indeed significant. Finally, we propose a scope for future research to model these MNAR biases and take them into account during training and evaluation to provide unbiased recommendations.[1]

[1] This chapter is based on work published in the *Proceedings of the 2018 REVEAL Workshop (co-located with the ACM RecSys Conference)* as "Fair Offline Evaluation Methodologies for Implicit-Feedback Recommender Systems with MNAR Data" by Olivier Jeunen, Koen Verstrepen and Bart Goethals [80].

4.1 Introduction

Over the past decades, personalisation has played an ever-growing role in how we consume content online. News websites, movie or music streaming services, retail stores, etc. can all greatly benefit from recommendation systems that accurately pair items with users and suggest them. Users can be guided towards the subset of catalogued items they are interested in, leading to more satisfying experiences and an increase in user engagement.

When recommender systems first gained traction, the field focused on the task of rating prediction. This assumes that a dataset with explicit feedback from users is available, which is often hard to collect. The goal was then to predict users' ratings for unseen items, with the rationale that items with higher predicted ratings make up better recommendations. In recent years, a shift has occurred towards item prediction from implicit feedback. These methods do not require explicit ratings by users, but rather take into account logged interactions between users and items to model inherent preferences and correlations. Throughout our work, we will focus on this task, as implicit feedback data is more prevalent in present day e-commerce.

Specific goals of recommender systems can vary greatly depending on their respective applications. Where some systems will be more focused on maximising user engagement in terms of time spent browsing a website, others might only focus on clicks or sales. User satisfaction, serendipity or diversity of the recommended items are only a few examples of many more possible objectives. We focus on maximising user engagement through clicks for the rest of this work, but our methodology is easily extended towards maximising sales. Analogously, this work focuses on, but is not limited to, collaborative filtering (CF) algorithms.

Traditionally, the performance of recommender systems (learning from implicit feedback) is evaluated on historical transactional data. As is often the case with classification problems and supervised learning in general, a portion of the data is split off and used as a test or validation set to assess algorithmic performance [96]. Leave-One-Out Cross-Validation (LOOCV) is a commonly recurring technique in the literature, where for every user one item-interaction is randomly selected to be part of the test set. The training set then consists of all remaining user-item pairs. Common metrics such as the *hit-rate-at-k* (HR@k) then compute the fraction of users for whom the removed item occurs in the top-k recommendations computed by the system, or the *normalised discounted cumulative gain* (NDCG@k) which evaluates the ranking of the removed item in the recommendation-list, or others. This process is repeated with different training-test splits, and performance metrics are subsequently aggregated over different runs in order to get a stable final result. Algorithms that can generate more promising metrics are then assumed to generate more revenue in an online setting than their competitors. This technique has been used widely and recently to present new algorithms as the state-of-the-art [169, 147, 30, 58, 150, 154].

Online evaluation methods such as live A/B-testing have been shown to paint a clearer and more honest image of an algorithm's performance in providing meaningful and interesting recommendations, but are generally more expensive and complex to realise [188]. During an A/B-test, the user-base is divided into groups. Every user in one of those groups is presented with recommendations generated by a different algorithm, specific to the group they belong to. Metrics such as the

click-through-rate (CTR) are then often used to compare which algorithm generates the most clicks, and thus is the most successful in generating revenue. Online evaluation is often favoured over its offline counterpart, as it directly correlates with the inherent goal of the system: to maximise user-interaction with the shown recommendations.

Crucial differences between live A/B-tests and LOOCV are two-fold: first there is a clear temporal dimension in many recommender systems use cases that is inherently taken into account in the first, but often neglected in the latter. Predicting past preferences based on future interactions is in many cases a considerably easier task than vice versa, and as a consequence their performances are not necessarily representative for each other.

Second, the goal of the evaluation technique is inherently dissimilar; where A/B-tests can present and evaluate a wide range of recommendations and evaluate how the user interacts with them, LOOCV is entirely restricted to predicting which items the user has already interacted with in the past. A top-k recommendation list might be clicked in an A/B-test because it sparks the user's interest, but it won't increase the hit-rate of LOOCV if the user has no recorded interactions with these items in the historical dataset. This distinction between negative and missing feedback is taken into account in the training phase of several well-known algorithms [65, 158, 169], but is much less studied in the context of offline evaluation.

Furthermore, if the available historical user-item interactions were collected with a live recommender system in place, this indicates the data are Missing-Not-At-Random (MNAR) [197]. We show that the algorithm presenting recommendations to the user significantly impacts offline evaluation results of different algorithms on the collected data. We adopt the terminology used by recent work in counterfactual estimation for recommender system evaluation and call such a live recommender system algorithm the *logging policy*. These counterfactual estimators have been shown to act as fast offline alternatives for more classical online methods such as A/B-testing, and are proven to be more correlated with online metrics than classical offline alternatives. As a consequence, improving these estimators has become a lively research direction in recent years [3, 49].

Related work has studied the correlation of the above-mentioned off- and online evaluation methods. Comparison research studies do exist for the specific fields of research paper recommendation [10], movie recommendation [177] and news recommendation [48], but none for the more general case. What these studies have in common however, is that they shed doubt on the assumption being made in many research papers that offline evaluations are good indicators of online performance. A discussion of how recommendation systems should be evaluated in an offline manner is presented by Herlocker et al. [62]. The authors identify a range of different goals the recommendation system and subsequently the evaluation method might need to be tuned to. A comparison of various evaluation metrics suggests that different metrics can be highly uncorrelated and make the evaluation procedure even less deterministic.

The contributions presented in this paper are the following:

1. We present a novel offline evaluation procedure that is more tightly coupled with the inherent goals of live recommender systems, and call it SW-EVAL.

2. We show that SW-EVAL generates very different results than LOOCV in terms of absolute metrics, ratios among algorithms and rankings among algorithms. We experimentally validate our findings on real-world data from a live e-commerce recommender system.

3. We show that the algorithm behind the live recommender system (or logging policy) induces a significant bias on collected data and as a consequence, severely influences offline evaluation results on said data.

The rest of this paper is structured as follows: we provide an overview of our alternative evaluation strategy in Section 4.2, and motivate our research questions. The data and algorithms we used for our experiments are presented and discussed in Section 4.3, along with their results. Our work is concluded in Section 4.4, where we finalise with a scope for future research.

4.2 Methodology

In what follows, we provide an overview of our methods. The following subsection focuses on preliminaries, after which we present our evaluation procedures and metrics. We then go on to motivate the research questions we aim to answer with this work, and how we achieve this.

Preliminaries

Throughout this paper, we assume to work with a set of historical transactional data, containing de-duplicated and timed logs of user-item interactions. U denotes the set of m unique users appearing in the dataset, and I the set of n unique items. A transaction is represented as a tuple $(u, i, t) \in U \times I \times \mathbb{R}^+$ where u is a user, i an item, and t a timestamp. $\mathscr{D}$ is the set of all available transactions. These transactions denote that user u has in some way consumed or interacted with item i at time t, be it in the form of a product purchase, a movie streaming, a click on a news article or otherwise. We represent these interactions in the form of a sparse user-item matrix $\mathbf{R} \in \{0,1\}^{m \times n}$, often called the rating or preference matrix. Rows in this matrix are users represented by the items they have consumed, and vice versa for columns: $\mathbf{R}_{u,i} = 1$ if and only if user u has consumed item i and $\mathbf{R}_{u,i} = 0$ otherwise. For a given item i, we define the set U_i as consisting of all users u that have consumed item i, that is $U_i = \{u \in U : \mathbf{R}_{u,i} = 1\}$, and I_u analogously for a given user u: $I_u = \{i \in I : \mathbf{R}_{u,i} = 1\}$. When we represent an item i or a user u as their respective column- or row-vectors in $\mathbf{R}$, we write them as $\boldsymbol{i}$ or $\boldsymbol{u}$. Time-intervals are characterised by subscripts: $\mathscr{D}_t$ is the set of all interactions up to but not including time t, $\mathbf{R}_{[t_x,t_y)}$ is the preference matrix containing all transactions $(u, i, t) \in \mathscr{D}$ where $t_x \leq t < t_y$. The timestamp of the latest transaction in $\mathscr{D}$ is denoted by $t_{\max}$.

Finally, as logged interactions in our setting originate from a system running live A/B-tests, we distinguish $\mathscr{D}^\pi$ as the set of transactions generated under logging policy π. In the trivial case where only one logging policy π is implemented, $\mathscr{D} = \mathscr{D}^\pi$. Note that sets of transactions generated under different logging policies typically contain disjoint sets of users, but the same sets of items.

Evaluation Procedure

As we have mentioned in Section 4.1, LOOCV is one of the most commonly recurring evaluation techniques in recommender system literature. For every user u, one item j is sampled uniformly at random from I_u to be used in the validation phase; all other remaining item-interactions $\{(u,i,t) \in \mathscr{D} : i \in I_u \backslash \{j\}\}$ are used to train the model and generate predictions. The model is then evaluated on its ability to predict which item was left out. This process is repeated several times with different random seeds, and results are then aggregated over runs. In this way the approach generates statistically stable results, covering every user in the dataset. We argue that what impedes this approach from being a good proxy for online behaviour, is that it completely ignores the chronological ordering of events. Not only does the model use future interactions to predict past interactions for a given user, future information about item correlations will be used as well to predict an interaction at a given earlier time.

As all transactions are timed, the fairest method to perform the train-test split would be a hard cut on a certain timestamp t: all interactions in $\mathscr{D}_t$ are used for training, and all interactions in $\mathscr{D}_{[t,t_{\max}]}$ can be used for testing. This process can be repeated, splitting on different times t and aggregating results in order to get statistically stable results over the full dataset. It is important to note that not all users are included in the evaluation at every split: to properly evaluate a user u, she should have recorded historical interactions to base recommendations on (we do not consider the extreme cold-start case [186]), and future interactions to predict and evaluate on. As this set of users might possibly be very small at certain points, we do not incorporate those where $|U_t \cap U_{[t,t_{\max}]}| < u_{min}$, with u_{min} a predefined threshold. If the number of users used for evaluation is too small, outliers will have an overly large impact on the overall view.

Furthermore, live recommender systems are not as static as they are made out to be in LOOCV. These systems are either incrementally trained or fully retrained regularly, with possibly multiple updates every hour. As a consequence, specific to the use case, it might not be best practice to evaluate a system on its ability to predict relevant items multiple weeks or even months in the future. In our proposed evaluation procedure, we divide the dataset into equidistant intervals of width t_Δ. At a given time t, we evaluate the model trained on $\mathscr{D}_t$ on its ability to generate relevant recommendations w.r.t. the interactions present in $\mathscr{D}_{[t,t+t_\Delta)}$. The granularity of t_Δ is relative to the use case. Where it might be very small for a news recommender (new items arrive at high rates and item popularity generally declines quickly over time, calling for fast and frequent updates), retail recommenders might call for wider intervals (daily or weekly, as sales are bound to seasonality).

Throughout the rest of this paper, we refer to our novel proposed procedure as k-fold Sliding Window Evaluation (SW-EVAL) where k indicates the number of intervals used in the validation step.

Our method corresponds to live A/B-testing in the sense that it follows a clear chronological ordering of events, which we argue is crucial to properly assess system performance. A major difference that remains is that of an over-representation of false negatives during the evaluation phase: where in a real-time setting a user might click on a certain recommendation when it would be given, that same recommendation will always be seen as non-relevant by offline evaluation procedures if there exists no historical interaction between said user and item. As this issue is very

non-trivial to solve, a careful choice of evaluation metrics that focus on rewarding true positives instead of penalising false positives is appropriate. We provide a brief overview of such metrics in the following subsection.

Evaluation Metric

As mentioned above, due to the inherent lack of user interaction in offline evaluation, we focus on metrics that reward true positives instead of those that penalise false positives. We will denote the set of known relevant items for a given user u as Rel_u, where her top-k recommendations are represented by $\text{Rec}_{u,k}$. Recall@k is then given by equation 4.1.

$$\text{Recall@}k = \frac{1}{m} \sum_{u \in U} \frac{|\text{Rel}_u \cap \text{Rec}_{u,k}|}{|\text{Rel}_u|} \tag{4.1}$$

With the leave-one-out scheme, Rel_u will always consist of exactly one item. In these cases, Recall@k is the same as HR@k: the fraction of users for whom the left-out item appears in the top-k recommendations.

In the more general case, Recall@k is the average fraction of retrieved relevant items in the top-k recommendations of all users. Note that when the number of relevant items is higher than k, it is impossible to achieve the perfect recall of 1. When k is set to the number of relevant items for every user $|\text{Rel}_u|$, equation 4.1 is called the R-precision, overcoming said issue.

LOOCV vs. SW-EVAL

The first hypothesis to tackle is whether LOOCV and SW-EVAL produce *comparable* results. We define *comparable* by three criteria in increasing order of importance:

Absolute metrics One of the main goals of recommender systems evaluation is to obtain a reliable estimate of the effectiveness of a recommendation algorithm. The order of magnitude of recommendation accuracy is therefore of critical importance. Business trade-offs that require accurate estimations of e.g. model cost vs. model performance cannot afford large errors here.

Ratios among algorithms As recommendation accuracy is often only one aspect of a broader evaluation, ratios among algorithms should be accurate as well. From an offline evaluation procedure, some model A might outperform some model B with a factor of 10. However, model A might also be 5 times more costly. If in practice, model A would only be twice as effective as model B, the inaccurate evaluation procedure leads to suboptimal decisions.

Rankings among algorithms In the case where model efficiency and cost are not taken into account, the ranking of competing algorithms that emerges from a certain evaluation procedure is still highly important. If simply the highest ranking model according to offline tests is deployed, it is imperative that the optimal model according to the offline evaluation procedure indeed reflects the best performer in

an online setting as well. Naturally, an offline evaluation procedure that correlates well with the system's actual online application is preferred.

As we motivated our evaluation procedure to be tightly coupled with the inherent characteristics of live recommender systems in e-commerce and other use cases, in the case of conflicting results on any of the above-mentioned aspects, we would place more trust in SW-EVAL than in LOOCV. To validate the effectiveness of SW-EVAL compared to LOOCV, we aim to study the parity between results of these offline evaluation procedures with those attained through live A/B-tests in future work.

Impact of Logging Policy

A second hypothesis we aim to investigate with this work is whether the logging policy π has significant impact when evaluating results on $\mathscr{D}^{\pi}$ or more generally on any $\mathscr{D}$ s.t. $\mathscr{D}^{\pi} \subseteq \mathscr{D}$. The work of Agarwal et al. provides a theoretical foundation for this problem in the more general case, where data from multiple diverging stochastic logging policies is naively combined [3]. However, to the best of our knowledge, in the context of recommender systems, no studies have conclusively proven or disproven that the impact of π is indeed highly significant with real-world data. If this is indeed the case, interesting directions for future research include modelling the bias π incurs, and deriving learning algorithms that mitigate this bias.

The impact of π can be defined by the same three aspects outlined in the previous subsection: whether it influences absolute values, ratios among algorithms, or rankings among algorithms. Intuitively, one would assume algorithms that closely correlate with the logging policy have an inherent advantage, as the disadvantages inferred by the lack of interaction in offline evaluation are annulled in this setting.

4.3 Experiments

The setting of our experiments is summarised in the following section. We give an overview of the recommendation algorithms we compared and go on by describing the dataset we used. Experimental results are presented and discussed in Subsections 4.3 and 4.3, following the same distinction as the research questions presented in Subsections 4.2 and 4.2 respectively.

Algorithms

Two simple baselines were used to compare algorithm performance against: a global popularity baseline (*POP*) and a sliding window popularity baseline (*POP-N*). The first sorts all items based on their number of occurrences in the full training set, and the latter sorts items based on the number of occurrences in the last N recorded interactions at the time of recommending. As the sliding window approach does not apply to the leave-one-out scheme (as it requires temporal information, which is not available), we do not include it in the LOOCV results. However, the approach proves surprisingly effective in our sliding window based evaluation method.

Apart from these baselines, we compared several well-known and widely used algorithms. The item k-nearest neighbour (*I-kNN*) algorithm computes the recommendation score $\tilde{\mathbf{R}}_{u,i}$ for a user u and an item i as a weighted sum of cosine

similarities between i and items $j \in I_u$, as shown in Equation 4.2 [183].

$$\tilde{\mathbf{R}}_{u,i} = \frac{1}{|I_u|} \sum_{j \in I_u} \cos(\boldsymbol{i}, \boldsymbol{j}) \tag{4.2}$$

As a full similarity self-join on the set of items I is both time- and space-consuming, only similarities between every item and its k nearest neighbours are computed and retained. This leads to a space complexity of $\mathcal{O}(kn)$ instead of $\mathcal{O}(n^2)$.

The user k-nearest neighbour algorithm (*U-kNN*) is a traditional and intuitive collaborative filtering algorithm that counts the occurrences of items that u has not yet consumed among the k nearest neighbours of u. Items that are more popular with similar users are then assumed to make up better recommendations [61].

Matrix factorization algorithms explicitly compute item- and user-factors in a fixed number of latent dimensions. The recommendation score for a user u and item i is then defined as the dot-product of their latent factors. We compute the factors using the well-known Singular Value Decomposition (SVD) algorithm [249].

As the purpose of this work is not to determine which algorithm generates optimal recommendations, we refrain from investigating more advanced or recent state-of-the-art algorithms. However, it should be noted that nearest-neighbour-based algorithms have recently still been shown to attain competitive performance with the state-of-the-art [220, 73].

If the top-k recommendation list $\text{Rec}_{u,k}$ generated by any algorithm contains items that were already in the history of u, we drop them from the list and expand it. As we work with de-duplicated interactions, re-targeting is out of the scope of this paper.

The baselines, U-kNN, I-kNN and SVD were implemented using Sci- and NumPy [227, 217]. Optimal hyper-parameters were obtained through an extensive grid search on LOOCV for optimal Recall@10 before experiments were conducted. For fair comparison, we did not recompute optimal hyper-parameters for the SW-EVAL setting or varying values of k, but retained the optimal ones for LOOCV and $k = 10$.

Dataset

Retail is a proprietary dataset obtained from the logs of a live recommender system serving a Belgian retail website, over the course of 4 months. During this period, 3 different algorithms generated recommendations in the fashion of an A/B-test. Approximately 25% of the recommendations were generated by a popularity baseline, 25% by an undisclosed algorithm, and 50% by *I-kNN*. We will respectively denote these logging policies by π_p, π_u and π_i. It is notable that the recommender system is subject to certain business rules, and top-k recommendation lists can therefore not be shown to the user as is. However, how and exactly which recommendations were effectively shown to the user is not important for the purposes of this work. Out of these 4 months, the last month acts as validation period for SW-EVAL. With 30 folds, this corresponds to daily updates and evaluations. Table 4.1 provides an overview of the dataset's size and properties.

Table 4.1: Characteristics of the dataset

	$\|\mathscr{D}\|$	$\|U\|$	$\|I\|$	Sparsity
Retail	734813	189343	10700	99.96%

LOOCV vs. SW-EVAL

In what follows we discuss the experimental results corresponding to the research question laid out in Section 4.2. To determine whether LOOCV and SW-EVAL produce comparable results as an offline evaluation procedure, we report mean Recall@k for varying k and relative performance to the best performer (in bold) for both 10-fold LOOCV and 30-fold SW-EVAL in Table 4.2. We set the threshold u_{min} to 100 for SW-EVAL, but generally multiple hundreds of users were included for evaluation at every fold. Results are visualised in Figure 4.1.

LOOCV results indicate that I-kNN is the clear best performer regardless of the value of k, exceeding the Recall of its competing algorithms with a factor of 2 and even beating the baseline algorithm with a factor of 10. However, SW-EVAL results paint an entirely different picture. First, the best performing algorithm has a mean Recall@10 that is a factor 5 smaller than reported by LOOCV. Second, we observe that the simple popularity baseline is able to attain up to 44% of the Recall@10 of the best performer, in stark contrast with the 11% reported by LOOCV. Furthermore, a simple sliding window extension boosts this further up to 71% and even 98% for $k = 5$. Third, where LOOCV concludes SVD and U-kNN to be virtually equal for $k = 10$, SW-EVAL clearly prefers SVD with 81% of the optimal performance instead of just 49%.

We observe that both for LOOCV and SW-EVAL, the gap between I-kNN and SVD closes as the number of generated recommendations grows. For the reported Recall@20, only a 3% difference remains between the two competing algorithms. The gap between U-kNN and I-kNN remains somewhat steady. A possible explanation for this is that the optimal amount of a user's neighbours to be taken into account when generating recommendations for $k = 10$ might be suboptimal for larger k, as the same candidate items might keep reappearing instead of novel recommendations. We see a similar but more stark effect for POP-N: as the optimal N obtained from the hyper-parameter optimisation procedure was rather low, the number of truly trending items might become less than k as k keeps growing. Extending POP-N to include a more advanced recency formula could certainly solve this issue.

It is clear that LOOCV and SW-EVAL do not yield comparable results for the given dataset. LOOCV is prone to overestimate model performance in general, vastly underestimate the effectiveness of popularity-based baselines, and generally rank algorithms very differently than SW-EVAL.

One might note that the number of relevant items for a given user in the validation set $|\text{Rel}_u|$ is important when computing the Recall@k: for LOOCV, $|\text{Rel}_u| = 1$. If it differs greatly for SW-EVAL, lower absolute numbers are to be expected. However, we also conducted experiments where only the first item a user interacts with in the validation interval is seen as relevant, effectively setting $|\text{Rel}_u| = 1$ as well. Results were very comparable, but omitted for brevity.

Table 4.2: Mean Recall@k for the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, for varying values of k. Δ% denotes the relative performance of an algorithm compared to the best performer (in bold). The sliding window baseline POP-N is not included for LOOCV as it lacks the use of temporal information.

	10-fold LOOCV								**30-fold SW-EVAL**									
k	**POP**	Δ%	**SVD**	Δ%	**U-kNN**	Δ%	**I-kNN**	Δ%	**POP**	Δ%	**POP-N**	Δ%	**SVD**	Δ%	**U-kNN**	Δ%	**I-kNN**	Δ%
5	0.019	-92%	0.098	-58%	0.116	-50%	**0.233**		0.016	-61%	0.039	-2%	0.028	-29%	0.022	-44%	**0.040**	
10	0.031	-89%	0.137	-51%	0.139	-51%	**0.282**		0.024	-56%	0.039	-29%	0.045	-19%	0.030	-45%	**0.055**	
15	0.041	-87%	0.165	-46%	0.151	-50%	**0.304**		0.032	-50%	0.039	-39%	0.058	-10%	0.034	-46%	**0.064**	
20	0.052	-84%	0.186	-41%	0.158	-50%	**0.317**		0.039	-45%	0.039	-45%	0.069	-3%	0.038	-47%	**0.071**	

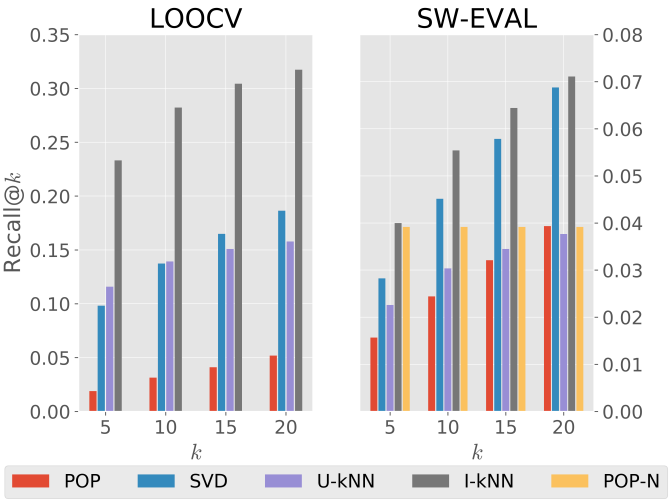


Figure 4.1: Mean Recall@k for the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, for varying values of k. Note that the SW-EVAL y-axis is scaled down by a factor of almost 5 in comparison with the LOOCV plot.

Impact of Logging Policy

In what follows we discuss the experimental results corresponding to the research question laid out in section 4.2. To determine whether the logging policy π has a significant impact on results from offline evaluation procedures on $\mathscr{D}^{\pi}$, we report mean Recall@k for varying k, and relative performance to the best performing algorithm for both 10-fold LOOCV and 30-fold SW-EVAL on $\mathscr{D}^{\pi_u}$, $\mathscr{D}^{\pi_p}$ and $\mathscr{D}^{\pi_i}$ respectively in Table 4.3. Even though the number of users covered by a subset $\mathscr{D}^{\pi}$ of $\mathscr{D}$ will be lower than the number of users covered by the full set of transactions $\mathscr{D}$, we set u_{min} to 100 as in our previous experiments. Equivalently to those previous experiments, this lower bound was never attained. Results are visualised in Figure 4.2, where the top and lower row of plots respectively correspond to LOOCV and SW-EVAL results. Every column represents results on a subset of logs corresponding to a given logging policy.

When considering LOOCV results, algorithm ranking generally does not seem to be impacted by varying π or k for this specific case. However, absolute measurements as well as ratios among competing algorithms clearly are. Under the undisclosed logging policy π_u and for $k = 10$, I-kNN only achieves roughly 65% of the performance it reaches under its own logging policy π_i. When we consider the absolute performance of U-kNN over all logging policies, we find it seems rather stable. However, the relative performance of U-kNN compared to I-kNN varies from 58% to 92% for its worst and best measurements respectively. While less pronounced, SVD exhibits similar behaviour. The popularity baseline seems moderately stable. Although less evident for π_u and π_p, a clear bias towards I-kNN is present in the data that was generated by showing I-kNN recommendations to users on the website, becoming more and more clear as k increases.

In the SW-EVAL results, the bias appears even more clear-cut. We distinctly observe that absolute values, the ratio amongst competing algorithms and the general ranking of algorithms is heavily influenced by π. For both the undisclosed algorithm π_u and the popularity policy π_p, the sliding window popularity baseline outperforms all competing algorithms for small values of k. As in the reported results from the previous section, SVD and I-kNN are able to thrive when generating more recommendations, whereas U-kNN and POP-N are much less able to do so. We suspect this is an artefact of our hyper-parameter optimisation procedure, since optimal parameters for $k = 5$ or $k = 20$ are bound to be different than those found for $k = 10$.

For the I-kNN policy π_i, a clear bias is present towards the I-kNN algorithm from the offline evaluation results. This is not surprising, as items that were effectively shown to a user have a much higher probability of being clicked than those that were not shown. Even though this clear bias is present, SVD attains up to 92% of I-kNN's performance for the largest value of k. This phenomenon is widely known as presentation bias [51], and can cause severe feedback loops if not handled properly. The effects of those feedback loops may be detrimental to the performance of a recommender system, as items in the long tail will never be considered fairly [161]. Recent related work has focused on retrieving users' intrinsic preferences when such feedback loops are present [193]. Metrics or frameworks that reward long-tail recommendations more than others are also a possible way of alleviating this issue [198, 1].

4.4 Conclusions

In this work, we have motivated the need for alternative offline evaluation procedures from LOOCV. We have identified that despite its clear flaws (ignoring all temporal information), LOOCV still remains an extremely popular technique to experimentally validate newly proposed algorithms as the state-of-the-art in recent recommender systems research [169, 147, 30, 58, 150, 154]. To this end, we have proposed a novel approach that much more tightly follows the important aspects of live recommender systems, such as their dynamic and temporal nature. We have experimentally validated on real-world data originating from the logs of a live recommender system that our approach yields very different results than LOOCV in terms of absolute metrics, ratios among competing algorithms, and mutual rankings of competing algorithms. Furthermore, LOOCV vastly underestimates the performance of advanced popularity-based approaches to recommendation.

Moreover, we motivated and discussed how live recommendation algorithms in place at the time of data collection can severely influence said data and produce a clear presentation bias, which is in turn prone to generate feedback loops. Offline evaluation results using traditional metrics on data gathered from a live recommender system are therefore heavily influenced by these biases, and lead to varying conclusions in terms of algorithm-specific performance.

Future Work

As most implicit feedback datasets for recommender systems originate from the logs of live systems, and more information about the logging policy is often unavail-

Table 4.3: Mean Recall@k for the different A/B-groups corresponding to logging policies in the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, for varying values of k. Δ% denotes the relative performance of an algorithm compared to the best performer (in bold). The sliding window baseline POP-N is not included for LOOCV as it lacks the use of temporal information.

		10-fold LOOCV								**30-fold SW-EVAL**									
	k	**POP**	Δ%	**SVD**	Δ%	**U-kNN**	Δ%	**I-kNN**	Δ%	**POP**	Δ%	**POP-N**	Δ%	**SVD**	Δ%	**U-kNN**	Δ%	**I-kNN**	Δ%
$\mathscr{D}^{\pi_u}$	5	0.017	-89%	0.092	-40%	0.141	-8%	**0.154**		0.016	-57%	**0.037**		0.026	-28%	0.025	-31%	0.023	-36%
	10	0.029	-84%	0.127	-32%	0.171	-8%	**0.185**		0.023	-43%	0.037	-8%	**0.040**		0.035	-13%	0.032	-21%
	15	0.039	-81%	0.152	-25%	0.188	-7%	**0.204**		0.031	-37%	0.037	-26%	**0.050**		0.040	-20%	0.037	-25%
	20	0.048	-78%	0.173	-19%	0.200	-7%	**0.214**		0.037	-36%	0.037	-36%	**0.058**		0.044	-25%	0.042	-29%
$\mathscr{D}^{\pi_p}$	5	0.021	-89%	0.097	-47%	0.139	-24%	**0.183**		0.016	-58%	**0.037**		0.029	-21%	0.230	-38%	0.032	-14%
	10	0.033	-85%	0.134	-41%	0.169	-25%	**0.225**		0.025	-47%	0.037	-23%	**0.048**		0.031	-34%	0.046	-4%
	15	0.045	-81%	0.159	-35%	0.186	-24%	**0.245**		0.033	-47%	0.037	-40%	**0.061**		0.038	-39%	0.053	-14%
	20	0.056	-78%	0.180	-30%	0.197	-23%	**0.257**		0.039	-45%	0.037	-48%	**0.071**		0.041	-42%	0.058	-18%
$\mathscr{D}^{\pi_i}$	5	0.018	-91%	0.102	-52%	0.137	-35%	**0.212**		0.016	-61%	0.038	-6%	0.027	-33%	0.027	-35%	**0.041**	
	10	0.031	-89%	0.142	-50%	0.165	-42%	**0.284**		0.024	-58%	0.038	-33%	0.045	-21%	0.034	-40%	**0.057**	
	15	0.042	-86%	0.170	-44%	0.180	-41%	**0.406**		0.032	-51%	0.038	-43%	0.057	-15%	0.039	-42%	**0.067**	
	20	0.051	-84%	0.192	-40%	0.189	-41%	**0.319**		0.040	-46%	0.038	-48%	0.068	-8%	0.042	-43%	**0.074**	

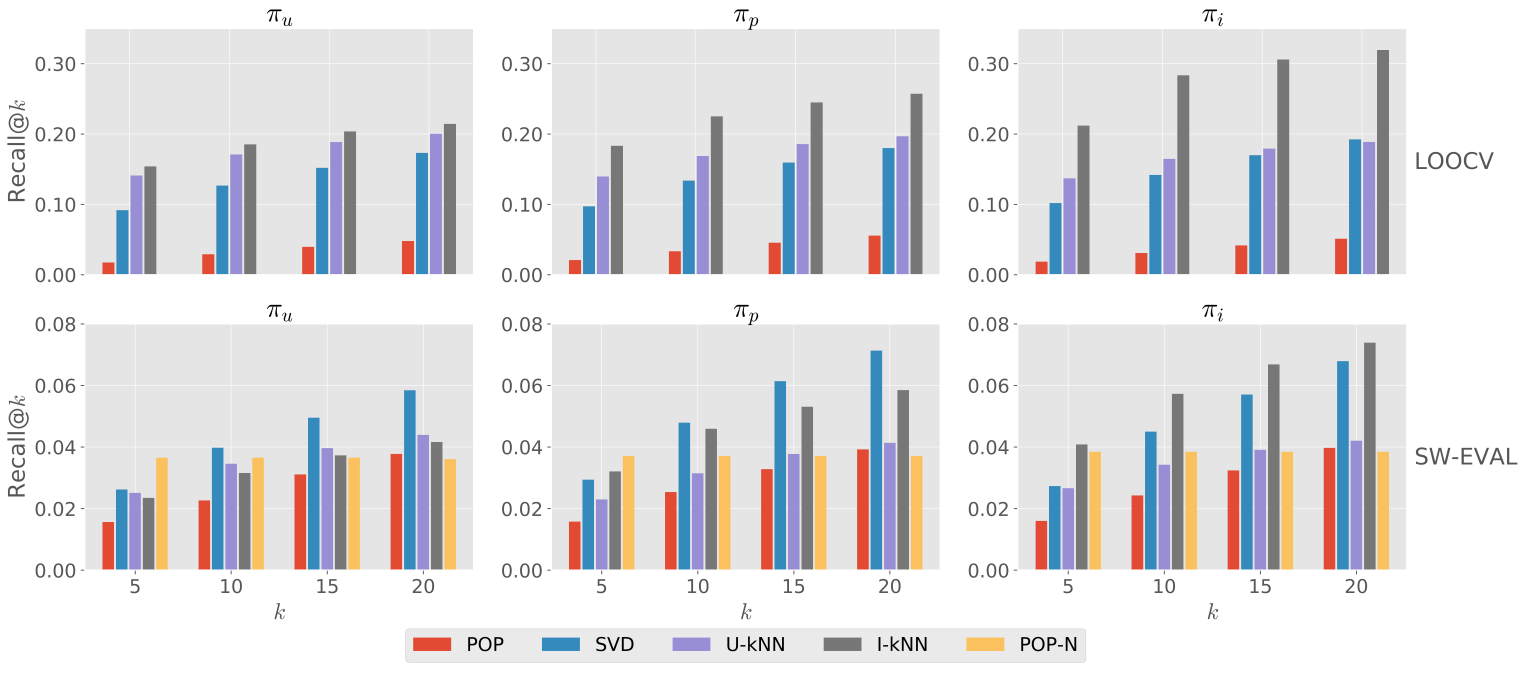


Figure 4.2: Mean Recall@k for the *Retail* dataset when using 10-fold LOOCV and 30-fold SW-EVAL respectively, subdivided by logging policy, for varying values of k. Note that the SW-EVAL y-axes are scaled down by a factor of almost 5 in comparison with the LOOCV plots.

able, the results presented throughout this paper reveal an important issue with current evaluation and learning procedures. We intend to further validate SW-EVAL by studying the impact of the window size t_Δ, and by examining the correlation between its results and those attained through large-scale online A/B-tests for different recommendation use cases. However, as the experimental results presented in Section 4.3 show, the choice of logging policy π induces a significant bias that heavily impacts offline evaluation results on $\mathscr{D}^\pi$. To this end, a clear need rises for bias-free learning and evaluation procedures.

The work of Steck [198] tackles the issue of popularity bias by proposing "Popularity-Stratified Recall" as an improved evaluation metric. The author goes on to present a model of item popularity, and an accompanying learning procedure that optimises a matrix factorization model for said novel metric. By adapting this "Popularity-Stratified Recall" and the accompanying recommender algorithm to a "Propensity-Stratified Recall" model, one might be able to alleviate the presentation bias that is inherent in most real-world datasets.

Recent work has tackled these biases by propensity-weighting in Learning-to-Rank specifically for information retrieval systems [87]. By extending such works to include biases that are inherent to recommender systems, we believe fairer offline learning and evaluation procedures for implicit feedback recommenders with MNAR data are achievable.

Reflections

This Chapter has highlighted several problems with prevalent offline evaluation procedures in the research literature, that still remain largely unsolved. In order to further validate the proposed sliding-window technique, a correlation study with results from online experiments would be imperative. Additionally, the selection bias that is a direct result from the logging policy provides a clear impediment for any offline evaluation technique to yield offline estimates that are predictive of online performance. This insight motivates the work in the final three Chapters of this thesis, where we want to quantify and mitigate the impact of the logging policy for offline *evaluation*, as well as offline *learning*.

CHAPTER 5

Revisiting Offline Evaluation for Implicit-Feedback Recommender Systems

Recommender systems are typically evaluated in an offline setting. A subset of the available user-item interactions is sampled to serve as test set, and some model trained on the remaining data points is then evaluated on its performance to predict which interactions were left out. Alternatively, in an online evaluation setting, multiple versions of the system are deployed and various metrics for those systems are recorded. Systems that score better on these metrics, are then typically preferred. Online evaluation is effective, but inefficient for a number of reasons. Offline evaluation is much more efficient, but current methodologies often fail to accurately predict online performance. In this work, we identify three ways to improve and extend current work on offline evaluation methodologies. More specifically, we believe there is much room for improvement in temporal evaluation, off-policy evaluation, and moving beyond using just clicks to evaluate performance.[1]

[1] This chapter is based on work published in the *Proceedings of the 2019 ACM RecSys Conference* as "Revisiting Offline Evaluation for Implicit-Feedback Recommender Systems" by Olivier Jeunen [75].

5.1 Introduction

Traditionally, recommender systems research focused on *rating* prediction from *explicit* feedback. The best known example of this setting is probably the Netflix Prize competition, where researchers were challenged to predict which ratings users had given to certain movies, based on millions of other user-item-rating triplets [13]. User-item pairs that were predicted to have high ratings, were then assumed to make up good recommendations. In recent years, a shift has occurred towards *item* prediction from *implicit* feedback [158, 65, 169, 221]. These systems no longer rely on a set of explicitly generated ratings, but can learn to infer personal preferences from logged feedback such as click behaviour on a news website, listening behaviour on a music streaming service, video watches on video streaming websites, and many more. As logged feedback is much easier to obtain than explicit ratings, these systems are gaining more and more popularity. Netflix has even moved on from their star rating system, favouring binary preference expressions [51].

Implicit-feedback recommender systems can be evaluated either off- or on-line. In the offline setting, the data is split into a training and testing subset, as is often the case in classical supervised learning contexts. Models learn from the training set, and are evaluated on their ability to predict the samples that are part of the test set. Those models that perform best on some chosen metric, are then assumed to be the optimal performers in an online setting as well. Online evaluation methods often relate to some form of A/B-testing: multiple different models are deployed, and their performance is measured according to some Key Performance Indicator (KPI), such as click-through rate (CTR), sales revenue, dwell time, retention rate, and so on. The biggest advantage of online evaluation methods is that they are very effective: interaction between users and the systems is directly measured, and if done properly, online experiments provide a fair and unambiguous view of system performance. They are, however, much more expensive than offline alternatives for a number of reasons [188, 3, 49]. Because of this, offline evaluation methods that can accurately predict online performance remain imperative. However, multiple recent works show time and again that offline evaluation results from traditional procedures are often contradictory compared to the results of live A/B-tests [10, 48, 177]. The scientific aspirations of this research are to identify which aspects are at the root cause of the fact that current offline evaluation procedures are often ineffective, and alleviate these aspects during the training as well as the evaluation phase of live recommender system deployment. Specifically, we wish to research novel offline evaluation procedures that are much more tightly coupled with the inherent characteristics of these live recommender systems, such as dynamic deployment, user interaction, temporal information, and self-induced presentation bias in the data. The rest of this work broadly corresponds to the following research questions:

1. What is the importance and the role of temporal information in an offline evaluation stage? How do we handle the sequential order of events and the absolute time between interactions and predictions correctly?

2. What impact does a live recommender system, in place during data collection, have on the resulting logged feedback? How do we mitigate biases they induce?

3. Can we use more information about user (in-)activity, beyond just clicks, during the offline evaluation process? How do we exploit information about impressions, dwell time, scrolling,... to effectively distinguish between *missing* and *negative* feedback?

5.2 Temporal Evaluation

In its simplest form, supervised learning systems are evaluated on a hold-out test set. A subset of the available data is randomly sampled and held out when training a model. Then, the model is evaluated on its performance when predicting the labels for the unseen, held-out test set. k-fold cross validation and bootstrap sampling are the most well known and widely used methods [96]. As the recommender systems field emerged from the broader machine learning field, adaptations specific to the recommender systems use case were proposed. Leave-one-out cross-validation (LOOCV) is a commonly recurring scheme in the literature, where one item for every user is randomly sampled to be part of the test set. The training set then consists of all remaining user-item pairs. Every model generates a set of top-N recommendations, and those that can rank the missing sampled items highest in the set of recommendations are assumed to be the best performers in an online environment as well. This process is repeated with different random seeds and samples, and results are averaged in an attempt to reduce variance from different runs.

While this technique has been used widely and recently to present new models as the state of the art [147, 30, 115, 58, 150, 154, 238, 251, 31], it entirely disregards the sequential nature of user-item interactions. It should come as no surprise that positively rewarding the prediction of past interactions from future data leads to a distorted picture of algorithmic performance in an online environment. Because of this, temporal evaluation has recently gained traction as well [223, 80, 89]. Zhao et al. [252] use a temporal leave-one-out scheme that we will refer to as last-one-out: instead of randomly sampling an item for every user, the last item is left out for every user. By doing this, no information about future preferences of a user can be used by the model when generating recommendations for said user, avoiding look-ahead bias. However, as the model is still trained on future interactions from other users, time-constraints remain violated and biases remain inevitable. Li et al. [113] propose an evaluation protocol called *replay*, aimed towards contextual-bandit news recommenders. Their work is extended into StreamingRec, a recently introduced offline evaluation framework for news recommenders [89]. They focus on model recency and incremental updates, which are of vital importance in the news domain. Nevertheless, incremental learning is not trivial for many state-of-the-art algorithms, nor is it always necessary. Further work for broader recommendation domains still needs to be conducted.

A novel temporal evaluation technique was proposed by Jeunen et al. [80], using a sliding window technique to adhere to the chronological ordering of interactions in the data, and aggregating multiple measurements to provide a robust estimate (Sliding-Window Evaluation, or SW-EVAL). The authors show that taking the sequentiality of the data into account at evaluation time has a significant impact on evaluation results, in terms of (1) absolute values of evaluation metrics, (2) ra-

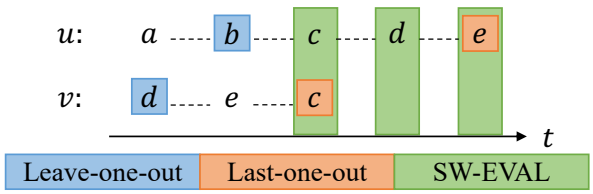


Figure 5.1: A visual representation of the differences between several offline evaluation procedures. $u, v \in \mathcal{U}$ are users, $a, b, c, d, e \in \mathcal{I}$ are items they have interacted with, and the x-axis represents time.

tios of evaluation metric values among competing algorithms, and (3) rankings of evaluation metric values among competing algorithms. Figure 5.1 provides some visual intuition into the differences of these methods. Here, $u, v \in \mathcal{U}$ are users and $a, b, c, d, e \in \mathcal{I}$ are items they have interacted with. The x-axis represents time. In this trivial example, the test windows used by SW-EVAL hold only one interaction per user. However, this parameter t_Δ can be tuned freely. We find that SW-EVAL has room for several extensions which could prove beneficial as future work. Currently, all user-item interactions that occur within the test window are taken into account with equal importance, and those that occur later are entirely ignored. Because of this, the impact of the window size t_Δ should be further investigated. Furthermore, a discounted importance function could be used to place higher weight on interactions that occur near the prediction time (i.e. the start of the test window), and lower weight on interactions that occur later in time. Additionally, until now, SW-EVAL has only been evaluated on a single dataset, with a modest selection of traditional baseline algorithms and the Recall@k metric. More thorough analysis needs to be done, by exploring the impact temporal evaluation has on clearly time-dependent use-cases (e.g. news recommendation) versus those that might seem less time-dependent on first look (e.g. movie recommendation). We wish to investigate the impact on various types of metrics as well, such as Mean Reciprocal Rank (MRR) or Normalised Discounted Cumulative Gain (NDCG). Finally, to validate the effectiveness of the evaluation procedure, a correlation study comparing results with those obtained from live A/B-tests needs to be conducted as well.

5.3 Debiasing Logged Feedback

Most recommender systems datasets are collected from logged user-item interactions on some website: be it e-commerce, news, media, and so on. Moreover, most of these websites have some form of live recommender system running that influences user behaviour. As user behaviour is logged *after* these users have been exposed to algorithmic recommendations, this generates severe biases in the resulting datasets [197, 22]. When these datasets are subsequently used to train models that are in turn deployed on the website, vicious feedback loops can occur [193]. The biases present in these datasets pose a significant challenge when the data is used to evaluate other competing algorithms in an offline manner. Under the presence of A/B tests, where multiple algorithms influence behaviour for different subsets of

the data [3], or for fast-changing environments that suffer from concept drift [70], these issues become even more poignant. This problem of predicting how a certain algorithm would have performed when we only have data that originates from a different algorithm, is a specific instance of the more general problem of counterfactual evaluation in the reinforcement learning literature. Importance weighting or inverse propensity scoring (IPS) is a well-known statistical technique, often used in these types of situations [19]. In this setting, a stochastic logging policy π_0 is assumed, that assigns a probability to an action, given some context: $\pi_0(i|x)$. Here, the action corresponds to recommending an item $i \in \mathcal{I}$, and the context is a feature vector that can be of arbitrary dimensionality d: i.e. $x \in \mathbb{R}^d$. When evaluating a new target policy π_t on data collected under π_0, the weights for samples (i, x) are then given by

$$w(i|x) = \frac{\pi_t(i|x)}{\pi_0(i|x)}. \tag{5.1}$$

It can be proven that, under mild assumptions, this weighting function results in an unbiased estimator. Its variance, however, can grow to be troublesome. When π_0 and π_t differ greatly, the ratio of their assigned probabilities will allow for some samples to be weighted disproportionately. Multiple techniques have been proposed to alleviate these issues, such as normalising or capping the weights [49].

However, for the case where π_0 is a deterministic policy (i.e. $\pi_0(i|x) \in \{0, 1\}$), these weighting techniques lose all practical value. Moreover, even when the policy is truly stochastic, the propensity scores for every recommendation-context pair (i, x) need to be known beforehand. Although this is somewhat trivial for the target policy, this is certainly not the case for general logging policies that might not be under our control[2]. Yang et al. [239] tackle this issue by modelling popularity bias as an exponential function to mimic the typical long-tail distribution. They then use this model to estimate propensity scores, and use those in turn to debias their evaluation procedure. The model is user- and context-independent, and assumes a single propensity score for every item i. Furthermore, they assume the probability of a user interacting with an item, is independent from whether that item has actually been recommended to and impressed upon the given user. Naturally, these assumptions do not hold in real-world situations. We identify two possible directions of future research:

1. By clustering users based on either meta-information (location, age-group, et cetera) or their historical sequence of logged items, the propensity of user-item pairs could more accurately be modelled in a cluster-local model. These models could then, stand-alone or in combination with a global model following the paradigm of [31], be used to improve upon their offline evaluation procedure, in combination with the SW-EVAL procedure.

2. The independence assumption between impressions and interactions is clearly oversimplified for real-world environments and data. For datasets that include impression information, this effect can be studied. When properly quantified, it can be used to improve upon the accuracy of the estimated propensities. Examples of such datasets include the Outbrain [155] and Plista [95] datasets.

[2] For the problem of computational advertising, a dataset containing logged propensities has been released by Lefortier et al. [106].

We discuss the benefit of using datasets enriched with this information in Section 5.4.

We believe more realistic propensity scores can be estimated through more nuanced modelling, which can in turn help in improving the effectiveness of IPS-based estimators in settings where propensities are unknown beforehand. In the case where not a single item, but a list of N items is recommended, Chen et al. [25] propose an adaptation of the IPS estimator. As propensity scores for the logging policy are also unknown beforehand in their setting, they propose to learn these alongside with the target policy via a recurrent neural network.

Preliminary results from SW-EVAL on logged feedback originating from different logging policies confirm that such biases are indeed present, and significantly impact offline evaluation results when not taken into account properly [80]. As future work, we wish to incorporate such a weighting procedure into the evaluation method, even when logging policy propensities are unknown in advance. Recent work used an adapted IPS-estimator to evaluate an algorithm for music playlist recommendation in an offline way [52]. They show that their offline evaluation results correlate with their online results, obtained through a series of live A/B tests. Multi-armed bandit models were used, with logged propensities. However, their work deals with an idealised environment, making it unclear whether the methodology can be readily applied other domains. Extending their work for generality, to include larger sets of available items, and towards ranking-based metrics is a promising direction for future research.

5.4 Beyond Just Clicks

Missing vs Negative Feedback

A well-known issue in implicit-feedback recommender system literature is the difficulty in distinguishing between *missing* and *negative* feedback: if a user-item pair (u, i) is missing from the dataset, does this mean that u disliked i, or was simply unaware of it? Many different algorithms have been proposed to address this problem, e.g. by sampling negatives or focusing only on positive preference expressions [158, 65, 169]. These works are limited, however, in that they only take a single type of feedback into account: clicks, sales, likes,... while typically much more interaction data is available to the entities providing the recommendations. The combination of clicks, add-to-cart actions and sales, page dwell times, *not* interacting with an impression, et cetera are all rich but largely neglected feedback signals. The work of Wan and McAuley [230] focuses on monotonic behaviour chains, where different levels of feedback are all jointly taken into account into the resulting model. We wish to study whether these different types of feedback can be used to distinguish between missing and negative feedback, in order to improve offline evaluation accuracy. If we know a user clicked on an item i, but later in the same session clicked on and bought an alternative item i', this might be interpreted as a negative feedback signal for the (u, i)-pair. Analogously, if i is shown as a recommendation to u without any resulting interaction, certain conclusions might be appropriate. After 1 impression, one can give the benefit of the doubt. After a large number of impressions, however, we might be able to infer a negative

relation between u and i. This issue has been tackled in the modelling stage of the recommender system, by effectively discounting the score for the item and pushing it downwards the top-N list as it is recommended again and again [105]. However, this information is mostly neglected when evaluating new models. In a broader sense, the problem of interpreting user *in*-action to better understand how users interact with live recommender systems in the movie domain was studied by Zhao et al. [253]. They identify various reasons why a user would *not* click on a given recommendation, and try to infer from context which of those reasons might explain a given sample. Further research for better understanding user interactions in various other domains would very much help with answering this research question in more general environments.

When the distinction between *missing* and *negative* feedback is facilitated, more involved objective functions such as Generalised Area under the Curve (GAUC) from the two-class collaborative filtering field [195, 163], can be exploited for use in the positive-only use-case [221]. Furthermore, multiple pairwise learning algorithms for recommender systems utilise the information that a given user has interacted with item i, but not with item j, to model preferences and similarities [169, 64]. These methods assume that the relation between u and j is less strong than u and i, and that u effectively prefers item i over j. Although this is a plausible assumption for datasets with large numbers of items, it cannot be assured. By sampling known, or assumed with high probability, *negative* items instead of *missing* items, we believe the performance of these existing algorithms can be significantly boosted.

Impression-data for Presentation Bias

Impression-data further has its use in the off-policy evaluation setting described in Section 5.3. Assume we have a dataset consisting of logged feedback from multiple loggers (e.g. collected from a live A/B test): $\mathscr{D} = \mathscr{D}_{\pi_0} \cup \ldots \cup \mathscr{D}_{\pi_n}$, where $\mathscr{D}_\pi$ refers to the subset of $\mathscr{D}$ that was generated under policy π, and $\mathscr{D}$ is a set of user-item-timestamp triplets. This is the same setting as tackled in the work of Agarwal et al. [3], where they propose two provably unbiased variants of IPS that limit the variance in these environments. However, it is assumed that all propensities are known beforehand, and the loggers are effectively stochastic multi-armed contextual bandits. When this is not the case, and the policies are possibly unknown, applying IPS-like methods gets troublesome. To this end, we propose a model-agnostic way of quantifying the biases among policies. By computing the overlap in their generated recommendations, we believe the presentation bias that is inherent to the data can be normalised and alleviated. We denote the set of impressions generated under the various logging policies by $\mathscr{R} = \mathscr{R}_{\pi_0} \cup \ldots \cup \mathscr{R}_{\pi_n}$. For every context-vector x in the logging data, a set of top-N recommendations is logged. As $\mathscr{D}$ and $\mathscr{R}$ usually originate from a live A/B-test, the context-vectors x will be disjoint for different logging policies. Suppose we want to evaluate a new policy π_t on $\mathscr{D}$; we can now compute $\mathscr{R}_{\pi_t}$ for every logged impression: the set of recommendations that *would* have been shown, had π_t been in effect. Through $\mathscr{R}_{\pi_i}$, we can quantify the similarity between a logging policy π_i and the target π_t. Although more sophisticated measures can be used, a simple first option is to use the Jaccard index to compute the intersection of the generated recommendations. Assuming $\mathscr{R}_{\pi_i}$ and $\mathscr{R}_{\pi_t}$ hold the same number of generated top-N recommendations, for an identical set of

context-vectors x, the similarity between these two policies then becomes

$$\text{sim}(\pi_i, \pi_t) = \frac{|\mathscr{R}_{\pi_i} \cap \mathscr{R}_{\pi_t}|}{|\mathscr{R}_{\pi_i}|}. \tag{5.2}$$

If the logging and target policy are identical, i.e. $\pi_i = \pi_t$, offline evaluation of π_t on $\mathscr{D}_{\pi_i}$ is effectively an *online* evaluation, as $\mathscr{R}_{\pi_t}$ does not hold hypothetical recommendations, but those that were actually shown to the user. On the other hand, if π_i and π_t generate entirely disjoint sets of recommendations, evaluating π_t on $\mathscr{D}_{\pi_i}$ will yield heavily biased results that disfavour π_t. Intuitively, the effect of this bias is correlated with $\text{sim}(\pi_i, \pi_t)$. A similarity of 0.5, indicates that 50% of the recommendations were actually shown at the time of data collection[3]. For a given evaluation function $f(\mathscr{D}, \mathscr{R}, \pi_t)$ that intends to evaluate the recommendations of π_t using $\mathscr{D}$, we propose a variant f', obtained by partitioning the data according to the logging policies and normalising according to their similarity with the target policy:

$$f'(\mathscr{D}, \mathscr{R}, \pi_t) = \frac{1}{n} \sum_{i=0}^{n} \frac{f(\mathscr{D}_{\pi_i}, \mathscr{R}_{\pi_i}, \pi_t)}{\text{sim}(\pi_i, \pi_t)}. \tag{5.3}$$

This formula can be decomposed even further, when normalising on a sample-by-sample basis instead of once for every logging policy. We intend to investigate this approach further from a theoretical basis, as well as empirically. Ideally, we wish to validate whether this approach can debias results obtained through off-policy evaluation, in situations where classical IPS weighting is not straightforward. This can be achieved by studying the correlation between results of our proposed approach, and those obtained through live A/B tests. Analogous to the IPS estimator presented in Equation 5.1, our proposed Equation 5.3 is prone to several of the same pitfalls. As $\text{sim}(\pi_i, \pi_t)$ approaches 0, these estimators will tend to over-compensate. It might prove beneficial, as is the case in IPS, to clip similarities to a given range or normalise them, with the goal of decreasing the variance. Since these approaches share the same intuitions, many of the proposed extensions to IPS can be studied with regards to their applicability in this setting [19, 49, 52, 70]. The work of Steck [198] on tackling popularity bias is also based on modelling bias, and normalising it in the evaluation phase. When our proposed approach is theoretically and experimentally fine-tuned and validated, interesting future work directions might open up on deriving novel learning and optimisation procedures that are unbiased as well. Furthermore, by studying the overlap between $\mathscr{D}$ and $\mathscr{R}$, we can disambiguate *organic* user behaviour (i.e. behaviour that was *not* instigated by a recommendation) and *influenced* user behaviour (i.e. behaviour that *was* instigated by a recommendation). We believe a better understanding of these different types of user behaviour will be useful to the research field as well.

5.5 Conclusions

In this paper, we have presented the key differences between on- and offline evaluation methodologies for implicit feedback recommender systems. We have motivated

[3] A possible extension would be to include the rank of the item in the recommendation list, instead of representing the recommendations as bags-of-items. In this case, Jaccard index could be replaced by e.g. Spearman's rank correlation coefficient.

the need for more effective offline evaluation strategies that are successful in predicting online performance. To this end, we have formulated and discussed three main research objectives: 1. temporal evaluation, 2. off-policy or counterfactual evaluation, and 3. the use of more involved interaction data to improve upon currently existing offline evaluation methods. For each of those areas, we summarised and discussed the state-of-the-art, identifying shortcomings and proposing promising areas for future work. In the near future, we wish to incorporate the above-mentioned extensions into our SW-EVAL approach [80].

Reflections

In this Chapter, we have laid out a research agenda for developing offline evaluation procedures that are predictive of online success metrics. Arguably, from the results presented in Chapter 4, we might infer that the issue of selection bias in logged feedback is the most pressing. If no selection bias is present, this means that the user-item interactions that are present in the dataset were not influenced by any deployed recommendation system. We call this *organic* feedback. On the other end of the spectrum, we can focus only on the user-item interactions that emerge from system recommendations. We call this *bandit* feedback.

Online success metrics are typically (but not exclusively) computed on bandit feedback, for which click-through-rates are the most prevalent example. Often adopted offline evaluation metrics in the research literature tend to (but not exclusively) assume organic feedback, for which examples include Recall and NDCG@K. The broader machine learning literature has leveraged ideas from counterfactual inference to compute purely *offline* estimators of *online* metrics, from bandit feedback. Computing such an estimator is an offline *evaluation* problem – finding the policy that maximises that same estimator is an offline *learning* problem. It now comes naturally that this unifying framework is a logical choice for designing offline methods that directly target online success, and this makes up the core motivation for the final Part of this thesis. Chapter 8 discusses methods that can jointly leverage *bandit* and *organic* feedback, a promising area for future work.

Part III

Effective Learning from Bandit Feedback

We must avoid false confidence bred from an ignorance of the probabilistic nature of the world, from a desire to see black and white where we should rightly see grey.

From a misplaced faith in certainty, the fact that to our minds, 99 percent, even 90 percent, basically means 100 percent – even though it does not. Not really.

— Immanuel Kant

CHAPTER 6

Joint Policy–Value Learning for Recommendation

Conventional approaches to recommendation often do not explicitly take into account information on previously shown recommendations and their recorded responses. One reason is that, since we do not know the outcome of actions the system did not take, learning directly from such logs is not a straightforward task. Several methods for off-policy or counterfactual learning have been proposed in recent years, but their efficacy for the recommendation task remains understudied. Due to the limitations of offline datasets and the lack of access of most academic researchers to online experiments, this is a non-trivial task. Simulation environments can provide a reproducible solution to this problem.

In this work, we conduct the first broad empirical study of counterfactual learning methods for recommendation, in a simulated environment. We consider various different policy-based methods that make use of the Inverse Propensity Score (IPS) to perform Counterfactual Risk Minimisation (CRM), as well as value-based methods based on Maximum Likelihood Estimation (MLE). We highlight how existing off-policy learning methods fail due to stochastic and sparse rewards, and show how a logarithmic variant of the traditional IPS estimator can solve these issues, whilst convexifying the objective and thus facilitating its optimisation. Additionally, under certain assumptions the value- and policy-based methods have an identical parameterisation, allowing us to propose a new model that combines both the MLE and CRM objectives. Extensive experiments show that this "Dual Bandit" approach achieves state-of-the-art performance in a wide range of scenarios, for varying logging policies, action spaces and training sample sizes.[1]

[1] This chapter is based on work published in the *Proceedings of the 2020 ACM SIGKDD Conference* as "Joint Policy-Value Learning for Recommendation" by Olivier Jeunen, David Rohde, Flavian Vasile and Martin Bompaire [84].

6.1 Introduction

Traditional approaches to recommendation are often based on some form of collaborative filtering on the user-item matrix containing *organic* user-item interactions [116, 200, 189]. These are generally user-item-timestamp triplets, indicating item purchases, clicks, views, et cetera. From rating- to next-item-prediction, such methods have known widespread success [199]. Generally, they are oblivious to whether actual recommendations were being shown to users in the data they learn from. In a parallel research direction, computational advertising applications often frame recommendation as an optimal *decision-making* problem, where the learning step aims to build an explicit reward model for all (user, recommendation)-pairs and the inference step chooses the best recommendation for the user given the learnt model. Existing work on modelling the probability of clicking on recommendations falls into this class, and publications on the subject traditionally originate from advertising research labs (see [137] for an overview). These approaches focus on *bandit* feedback: interactions between users and recommendations being shown. Therefore, this data is conditioned on the policy describing the existing recommender system.

For both existing frameworks, the majority of new recommendation algorithms presented in academic papers are evaluated on an offline dataset of logged user-item interactions, with results reported for some offline ranking metric. Recent work has shown repeatedly that offline evaluation results tend to diverge from online performance [48, 177, 75]. Additionally, existing offline evaluation results are often even contradictory over different runs and datasets, or extremely hard to reproduce in a robust manner [32, 99]. From a practical or industrial point of view, a need arises for offline evaluation methods that are robust, reproducible and closely related with the actual online objectives of the deployed recommender system.

The reinforcement learning literature has long dealt with similar issues. Based on logged data from a certain *policy* (recommender), we want to predict what the performance would have been if another policy had been deployed. Counterfactual estimators, often based on importance sampling [156], are at the heart of this type of evaluation. Recent work has shown that they can accurately reflect online performance in recommendation use-cases [49, 52].

We believe that the main reason why the value of bandit feedback is not further explored in most recommendation research, lies in the datasets we use: the vast majority of available offline datasets simply do not include information about the recommendations that were shown, and whether the user interacted with them. Naturally, we cannot learn from what we do not know. Some datasets containing logs of historical recommendations and their outcomes do exist, mostly for the specific task of click-through-rate (CTR) prediction. Nevertheless, they either do not include propensity scores produced through adding randomisation at recommendation time, which is often a requirement for unbiased learning and evaluation [19, 49], or the variance induced by the propensities prohibits effective learning and evaluation [106]. Recently, several simulation environments have been proposed for the recommendation setting, allowing online experiments such as A/B-tests to be simulated [176, 68]. They enable us to explore the use of bandit feedback for learning and evaluation of recommender systems in a reproducible manner, and have opened up promising new research directions. The value of such simulation frameworks has

steadily gained more attention in recent research [82, 146, 180].

Moving on from evaluation, recent work [19, 211, 212, 88, 81, 129] explores the problem of learning a new and optimal recommendation policy based on a dataset consisting of logged bandit feedback. However, real-world recommender systems tend to differ from the assumptions made in the existing work that studies the use of bandit feedback for the problem off-policy learning or Counterfactual Risk Minimization (CRM), in terms of the stochasticity and sparsity of rewards. Furthermore, the size of the action space in real-world scenarios makes the choice of problem formulations that result in convex objectives very attractive (due to the existence of mature fast large-scale optimisation algorithms, such as L-BFGS [103]). The technical contributions we present in this paper are the following:

(1) *Analysis of the Convex Policy Lower-bound for Stochastic Rewards.* We show where existing off-policy learning approaches fall short, especially due to the inherent stochasticity and sparsity of the reward process. We introduce a logarithmic variant of the traditional Inverse Propensity Scoring (IPS) estimator that allows us to map the objective to a convex problem, yielding a weighted multinomial log-likelihood that lower bounds the original. Recent related work has introduced a similar objective as a Policy Improvement Lower-bound (PIL) and a variance reduction technique [129]; we focus on its practicality in stochastic environments and empirically show how it can improve performance. As the logarithmic transformation convexifies the objective, it facilitates its optimisation.

(2) *Joint Policy-Value Optimisation.* By presenting value- and policy-based methods in a common framework, we show that they have an identical parameterisation under certain assumptions. This allows us to propose *Dual Bandit (DB)*, a hybrid learning objective that combines both Maximum Likelihood Estimation (MLE) and Counterfactual Risk Minimisation (CRM) without introducing additional parameters, effectively unifying the value- and policy-based families. We show how it effectively alleviates well-known problems such as propensity overfitting. Moreover, standard off-policy learning methods do not take into account negative evidence (i.e. non-clicked recommendations), which is solved through the value-based cross-entropy term. Finally, we show that the DB approach achieves state-of-the-art performance in a wide range of recommendation settings, for varying logging policies, training sample and action space sizes. The most interesting and realistic results deal with large action spaces, finite samples, and limited randomisation, which is exactly where the Dual Bandit shows its superiority.

(3) *Reproducible Simulation Study.* The empirical performance of counterfactual learning methods has mainly been studied in multi-class [211, 88, 129] and multi-label [212] classification environments where the bandit setting is simulated. Recent work that focuses on the recommendation use-case adopts a supervised-to-bandit conversion on existing datasets and custom simulated datasets that assume deterministic rewards [128], or shows empirical success through live experiments [25]. We conduct the first broad simulation study of both value-based methods based on MLE, and policy-based methods that rely on IPS to perform CRM in stochastic recommendation environments. In order to aid in the reproducibility of the research presented in our work, we adopt the RecoGym environment in our experiments [176]. All source code is available at `https://github.com/olivierjeunen/dual-bandit-kdd-2020`.

6.2 Background and Related Work

In what follows, we present an overview of the methods we consider in our comparison. As we focus on so-called *bandit feedback*, these methods make use of action-reward pairs: recommendations that were shown and whether they were interacted with. We discern two broad families: value- and policy-based methods. The first aims to model the reward a certain action will yield, relying on classical supervised learning approaches [63]. The latter directly models the actions that should be taken in order to maximise the total cumulative reward a policy will collect. This line of research is more closely related to the reinforcement learning (RL) field [208]. Value-based models and their variations are often referred to as Q-learning in the RL community.

We assume to have access to a dataset of logged bandit feedback $\mathscr{D}$, consisting of N tuples $(\mathbf{x}_i, a_i, p_i, c_i)$. This data has been collected under some stochastic logging policy π_0 that describes a probability distribution over actions (*recommendations*), conditioned on some context. Here, $\mathbf{x}_i \in \mathbb{R}^n$ describes the user state or context vector. Although this vector can be of arbitrary dimension, we will assume it to be a vector of length n containing counts of historical *organic* interactions with items for fair comparison and simplicity. $a_i \in \{1, 2, \ldots, n\}$ is a scalar identifier representing the action that was taken (i.e. the item that was shown when the system was presented with context $\mathbf{x}_i$), we denote the corresponding one-hot encoded vector as $\mathbf{a}_i$. The probability with which that action was taken by the logging policy is denoted by $p_i \equiv \pi_0(a_i|\mathbf{x}_i) \in [0, 1]$. The observed reward (whether the user interacted with the presented recommendation) is represented as $c_i \in \{0, 1\}$.

Value-based Approaches

The most straightforward method is to first perform statistical inference through Maximum Likelihood Estimation (MLE) and then do decision making in a separate step, bypassing the empirical/counterfactual risk minimisation principles. If the reward is a binary variable, then a logistic regression model is appropriate, as shown in Equation 6.1. This formulation can be naturally extended to include more advanced likelihood-based models such as deep neural networks.

$$P(C = 1|\mathbf{x}, a, \boldsymbol{\beta}) = \sigma((\mathbf{x} \otimes \mathbf{a})^\intercal [(] \boldsymbol{\beta}) + b) = \sigma(\mathbf{x}^\intercal \boldsymbol{\beta}_{\cdot,a} + b) \tag{6.1}$$

Here, $\boldsymbol{\beta} \in \mathbb{R}^{n \times n}$ are the model parameters where $\boldsymbol{\beta}_{\cdot,a}$ are those corresponding to predictions for action a. $\sigma(\cdot)$ is the logistic sigmoid, $\otimes$ is the Kronecker product and $[(]\cdot)$ vectorises the matrix into a column vector.[2] The intercept or bias-term is denoted by b. As it is a single constant scalar for all context-action pairs, it does not have any impact on the ranking of competing actions or the actual decision making process. Nevertheless, it positively impacts the quality of the fitted model [63]. We ensure fair comparison with the other approaches: all consist of exactly n^2 parameters that impact the resulting decision rule (excluding hyper-parameters, that is). Optimising the binary cross-entropy or negative log-likelihood of this model with respect to a historical dataset yields the objective shown in Equation 6.2.

[2] We implement the rightmost formula as it is equivalent to, but computationally significantly less expensive than explicitly computing the Kronecker product.

$$\begin{aligned}\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{N} c_i \ln\left(\sigma(\mathbf{x}_i^\intercal \boldsymbol{\beta}_{\cdot,a_i} + b)\right) \\ +(1-c_i)\ln\left(1-\sigma(\mathbf{x}_i^\intercal \boldsymbol{\beta}_{\cdot,a_i} + b)\right)\end{aligned} \tag{6.2}$$

Once the model parameters have been fitted, we can obtain a greedy decision for a given context $\mathbf{x}$ by performing the action a^* with the highest probability of leading to a positive reward. Naturally, this requires n model evaluations followed by an arg-max operation:

$$a^* = \arg\max_a P(C=1|\mathbf{x}, a, \boldsymbol{\beta}) = \arg\max_a \mathbf{x}^\intercal \boldsymbol{\beta}_{\cdot,a}. \tag{6.3}$$

We often fit simpler (for example, linear) models to capture more complex relationships. When doing this, the model *underfits* the data as it is unable to capture the true relationship. When a standard MLE approach is used, the error due to the underfitting will be minimised around common occurrences of $(\mathbf{x}, a)$. If the distribution of $(\mathbf{x}, a)$-pairs in the historical training data differs from those in the test set, this leads to a phenomenon widely known as *covariate shift* [191]. As we generally wish to learn a new policy that improves upon the logging policy, it will take different actions by definition, and covariate shift is inevitable.

One general solution to this issue is to make use of importance sampling [156], and reweight samples to adjust for the difference in the distribution of past actions (as per the logging policy) and future actions (which we will evaluate uniformly in this case, as that is exactly what we do with the argmax-operation over actions in Eq. 6.3). Practically, this is achieved by reweighting samples $(\mathbf{x}_i, a_i, c_i)$ in Equation 6.2 by the inverse propensity score of the logging policy during maximum likelihood estimation: $w_i = \frac{1}{\pi_0(a_i|\mathbf{x}_i)}$.

Value-based methods estimate the likely reward for each action. Due to the logging policy, the quality of this estimate can vary dramatically. The estimation uncertainty will be low for actions performed often by the logging policy, but poor otherwise. As the action is selected by selecting the maximum value, the policy may be disrupted badly by a single erroneously optimistic estimate; a phenomenon known as optimiser's curse [194]. In recommender systems, the action space can be large and the size of the training sample and amount of randomisation are often limited. Because of this, these problems are very tangible in a real-world setting.

Policy-based Approaches

Value-based methods model the probability of a click (*value*), given a context-action pair. Policy-based methods, on the other hand, bypass this step and directly map a context to a decision rule.

Contextual bandits are one example of such a method, directly modelling the probability of an action, conditioned on a context vector. This is shown in Equation 6.4, where $\boldsymbol{\theta} \in \mathbb{R}^{n \times n}$ are the model parameters.

$$P(A=a|\mathbf{x}, \boldsymbol{\theta}) = \pi_{\boldsymbol{\theta}}(a|\mathbf{x}) = \frac{\exp(\mathbf{x}^\intercal \boldsymbol{\theta}_{\cdot,a})}{\sum_{j=1}^{n} \exp(\mathbf{x}^\intercal \boldsymbol{\theta}_{\cdot,a_j})} = [\text{softmax}(\mathbf{x}^\intercal \boldsymbol{\theta})]_a \tag{6.4}$$

Consistent with the work of Bottou et al. [19] and Swaminathan and Joachims [211], this formulation focuses on learning policies that are parametrised as exponential models (i.e. going through a softmax). The goal at hand is to learn a policy that chooses the optimal action given a context $\mathbf{x}$, i.e. the policy that maximises the reward we would have gotten when $\pi_{\boldsymbol{\theta}}$ was deployed instead of the logging policy π_0. In our setting, this reward can be interpreted as the absolute number of clicks. Equation 6.5 formalises this counterfactual objective, which can be optimised directly. Essentially, contextual bandits try to replay decisions that worked in the training sample. From a learned stochastic policy $\pi_{\boldsymbol{\theta}}$, a deterministic decision rule can easily be deduced as shown in Equation 6.6.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} c_i \frac{\pi_{\boldsymbol{\theta}}(a_i|\mathbf{x_i})}{\pi_0(a_i|\mathbf{x_i})} \tag{6.5}$$

$$a^* = \arg\max_a P(A = a|\mathbf{x}, \boldsymbol{\theta}) = \arg\max_a \mathbf{x}^\top \boldsymbol{\theta}_{\cdot,a} \tag{6.6}$$

Note that if we let $\boldsymbol{\beta} \equiv \boldsymbol{\theta}$, the decision rule in Equation 6.3 is identical to the decision rule in Equation 6.6. Although the optimisation problems given in Equations 6.1 and 6.4 are quite different: maximum likelihood ignores the IPS score, and the contextual bandit ignores the non-clicks. The likelihood-based approach attempts to model the click for every action, whereas the contextual bandit simply attempts to identify the best action. This common parameterisation will motivate the "Dual Bandit" method we present later in this work, a principled approach to jointly optimise these two objectives.

Inverse Propensity Scoring (IPS) is a powerful technique that allows for counterfactual optimisation. When the target policy $\pi_{\boldsymbol{\theta}}$ and the logging policy π_0 diverge, however, IPS-based estimators tend to have very high variance and be unreliable for policy learning. Several extensions to the classical estimator have been proposed, trading variance for bias. Clipping the propensity ratios to a maximum value or self-normalising them are common practices. The most notable recent extensions to this formulation are POEM [211], BanditNet [88] and PIL-IML [129]. They respectively include additional terms for sample variance penalisation (SVP), self-normalisation (SNIPS) or imitation learning (IML). Conceptually, they all restrict the newly learned policy to not stray too far from the logging policy, as the uncertainty on rare actions brings along high variance on the performance. Originally introduced by Swaminathan and Joachims, we refer to these approaches as performing some form of Counterfactual Risk Minimisation (CRM).

Recent related work has linked IPS techniques to policy gradient methods such as REINFORCE [234]. These approaches aim to maximise the expected cumulative reward over a certain time horizon, and were extended to handle specific cases such as top-K recommendation [25] and two-stage recommendation pipelines [128]. For the binary, immediate reward use-case we tackle in this paper, maximising the expected reward via REINFORCE yields the same result (that is the same policy) as optimising the IPS objective given in Eq. 6.5. Ma et al. draw further connections between their objective and natural policy gradients; we refer the interested reader to Appendix E in their work [129]. Table 6.1 shows an overview of the discussed methods.

Family	**Method**	$P(C\|x,a)$	$P(A\|x)$	**Neg.**	**IPS**	**SVP**	**IML**	**Equivariant**	**Ref.**
Value learning	Likelihood	✓		✓					[63]
	IPS Likelihood	✓		✓	✓				[204]
Policy learning	Contextual Bandit		✓		✓				[19]
	POEM		✓		✓	✓			[211]
	BanditNet		✓	~	✓			✓	[88]
	PIL-IML		✓		✓		✓		[129]
Joint learning	Dual Bandit	✓	✓	✓	✓				This work.

Table 6.1: An overview of the methods we discuss in this paper, and how they relate to one another in terms of the target they optimise and whether they exploit negative samples, make use of Inverse Propensity Scoring (IPS), Sample Variance Penalisation (SVP), an Imitation Learning (IML) term, or whether the approach is invariant to translations of the reward.

Several additional counterfactual estimators have been proposed in the literature, such as the Direct Method (DM) and Doubly Robust (DR) [36], More Robust Doubly Robust (MRDR) [43], Continuous Adaptive Blending (CAB) [205] and others [228, 255]. These works either focus on evaluation instead of learning (as they are non-differentiable), or they include an additional regression model that estimates the reward $\delta(\mathbf{x}, a)$ for a given context-action pair. As such, they are out of scope for this study.[3] Additionally, contextual and multi-arm bandit approaches with online model updates have been thoroughly studied. Some notable approaches are [112, 23, 114]. In contrast with these methods, the models we discuss in this work are entirely *off-policy* and lack any interactive component.

As we will discuss in depth in the following section, recommender system logs are stochastic in nature. Probabilistic models propose a way of naturally handling uncertainty. When sophisticated priors are used, Bayesian methods can perform well in modelling complex relationships with small samples. For the related recommendation task of rating prediction, Bayesian approaches have recently been shown to robustly obtain state-of-the-art performance [181, 170]. In the case of top-N recommendation from organic user-item interaction data, recent variational methods consistently attain impressive results as well [116, 189, 32]. Sakhi et al. propose a probabilistic approach to combine both organic and bandit signals in order to improve the estimation of recommendation quality in a Bayesian latent factor model [180]. Further exploring such models and their applicability in off-policy recommendation settings is a promising avenue for future research.

6.3 Learning for Recommendation

The performance of most methods discussed in this work has been empirically validated for multi-class [211, 88, 129] or multi-label [212] classification environments, which simulate a "Batch-Learning from Bandit-Feedback" (BLBF) context. The policy performs an action (guesses a class or label), and observes the reward (the guess is either correct or incorrect). Other work has adopted these supervised-to-bandit conversions to mimic the recommendation task as well [128]. Assuming we have more than one item in the catalogue that is of interest to the user, the recommendation use-case is indeed most closely aligned with the multi-label setup. Several key differences remain, which we tackle throughout the rest of this section:

(1) *Stochastic vs Deterministic Rewards.* Previous work has always been evaluated in experimental set-ups where the reward is deterministic: if a model makes decision a when presented with context $\mathbf{x}$, the observed reward c will always be exactly the same. This (often implicit) assumption does not hold in real-world recommendation settings: users may click (or conversely, refuse to click) on shown recommendations for any number of reasons. As it is intractable for all factors on which the reward c is dependent to be included in the context-vector $\mathbf{x}$, the rewards conditional on the known context-features will always be stochastic.

(2) *Sparse Rewards and Low Treatment Effects.* CTR measurements in real-world systems can be notoriously low, skewing the estimates. When the training log $\mathscr{D}$ contains tuples for the same context-action pairs with different observed rewards,

[3] Note that the Dual Bandit method we propose later in this work also induces a regression model, but does this *without* introducing additional parameters.

however, the information embedded in the clicks is often more valuable than the non-clicks, although the latter will make up the vast majority of the logged samples. Furthermore, the difference between the empirical *best* and *second best* arms might not always be large. As such, solely focusing on the empirically best action is not an optimal strategy.

(3) *Large Action Spaces.* Recommender systems typically deal with vast item catalogues. Previous experimental validation of these methods has focused on setups where the number of *actions* (classes, labels, documents, ...) is at most a few dozen. Their practicality in very large action spaces remains understudied.

Logarithmic IPS for Stochastic Rewards

Rationally, we could assume the true reward to be drawn from some Bernoulli-distribution with parameter q relative to the relevance of the taken action; but relevant actions will not always lead to clicks, and a click does not necessarily imply that the performed action was the most relevant option. Estimating this parameter q is an indubitably harder task than the deterministic case, requiring larger training samples and leaving us vulnerable for common pitfalls such as the optimiser's curse [194].

The counterfactual objective function in Equation 6.5 describes an empirical IPS-estimate of the reward that $\pi_{\boldsymbol{\theta}}$ would incur based on data collected under π_0, given a historical dataset $\mathscr{D}$. We denote this estimator as $\hat{R}_{\text{IPS}}(\pi_{\boldsymbol{\theta}}, \mathscr{D})$, and drop parentheses and arguments when they are clear from context.

$$\hat{R}_{\text{IPS}}(\pi_{\boldsymbol{\theta}}, \mathscr{D}) = \frac{1}{N} \sum_{i=1}^{N} c_i \frac{\pi_{\boldsymbol{\theta}}(a_i|\mathbf{x}_i)}{\pi_0(a_i|\mathbf{x}_i)} \tag{6.7}$$

This objective leads to a "*winner takes it all*" scenario, where the optimal policy puts all its mass on the actions that obtained the highest empirical reward in the finite training sample – provided the capacity of the policy space allows for this to happen. We argue that this can lead to sub-optimal policies in the stochastic scenario, as positive samples for the empirically "*second best*" actions are simply ignored. This behaviour has recently been studied in top-K recommendation scenarios [25]; we focus on top-1 recommendation.

In supervised learning, it is common practice to optimise a surrogate loss function instead of a direct metric (hard classification error, for example). Through Jensen's inequality, we can derive a lower bound of the traditional IPS estimator ($\hat{R}_{\text{IPS}}$) that uses a logarithmic transform on the likelihood.[4] Intuitively, this is equivalent to optimising the log-likelihood of a multinomial logistic regression model where each observation has been weighted by $w_i = \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)}$. We refer to this logarithmic estimator as $\hat{R}_{\ln(\text{IPS})}$ and present it in Equation 6.8. Recent related work introduced this equivalently as a policy improvement lower-bound [129, 122]. These works primarily focus on model misspecification and multiple iterations of logging and learning, whereas we study the impact of the transformation in stochastic environments.[5]

[4] A derivation can be found in the reproducibility appendix.

[5] See [129] and its appendices to link the logarithmic estimator to existing variance reduction techniques such as capped IPS, SVP, SNIPS, and others.

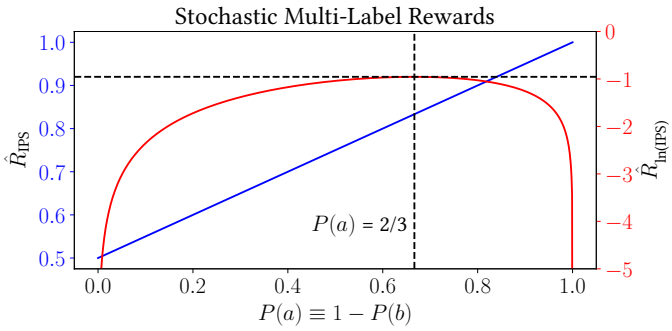


Figure 6.1: $\hat{R}_{\ln(\text{IPS})}$ penalises the learned policy for *missing* actions that led to positive rewards in the training sample. Maximising this empirical estimator for a toy example training sample with 2 clicks on action a and 1 click on action b leads to more proportional allocation compared to $\hat{R}_{\text{IPS}}$ (assuming a uniform π_0). For deterministic multi-class or multi-label settings, both estimators share optima.

$$\hat{R}_{\ln(\text{IPS})}(\pi_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} c_i \frac{\ln(\pi_{\boldsymbol{\theta}}(a_i|\mathbf{x}_i))}{\pi_0(a_i|\mathbf{x}_i)} \tag{6.8}$$

In combination with the exponential parameterisation in Equation 6.4, this logarithmic objective leads to a *proportional* allocation of probability mass. It can be readily plugged into existing policy learning methods described in Section 6.2. As it forces the model to include positive samples for all actions instead of the empirically best action only ($\lim_{P \to 0} \ln(P) = -\infty$), we expect and empirically observe that this leads to 1. less overfitting, and 2. more robust performance. Furthermore, computations in the log-space improve the numerical stability of the optimisation procedure. Negative log-likelihood or cross-entropy (as also used in Equation 6.2 and widely adopted in classification and deep learning) is commonly used in these settings, as it transforms the objective to be convex, and thus easy to optimise at large scale.

In policy gradient methods such as REINFORCE, a common way to optimise the objective consists of performing a gradient step involving the logarithm of the policy [209]. Also called the "log-trick" or "log derivative trick", this technique is by nature very different from ours. Indeed, the "log derivative trick" is a way to compute an unbiased estimate of the gradient of the expected reward under the new policy. Hence, using this trick does not change the optimised objective. In our case, the logarithmic transform *changes* the objective and moves the global optimum. Figure 6.1 visualises how this transforms the objective when the reward is stochastic, or the model misspecified.

Joint Policy-Value Optimisation

Policy-based approaches can suffer from so-called propensity overfitting, where the learning objective is trivially optimised by missing the observed data [212]. To see this, suppose we would transform Eq. 6.5 to a minimisation of non-clicks instead of

a maximisation of clicks. The result of this transformation would be that putting 0 probability mass on all observed $\pi_0(a|\mathbf{x})$ trivially minimises the objective, although it would clearly not lead to better generalisation capabilities. Indeed, learning to avoid actions does not imply learning which actions to take. These types of issues are generally avoided through the use of a SNIPS estimator, which includes a multiplicative control variate in the IPS estimator that heavily penalises this behaviour. BanditNet optimises a Lagrangian form of this estimator and achieved promising results in a deterministic multi-class bandit setting [88]. By introducing translations to the binary reward, it opens up opportunities for learning from *negative* samples; as can be seen from Eq. 6.5, probabilities for 0-reward actions ($c_i = 0$) have no direct impact on the objective. However, as the optimal Lagrange multiplier γ could be 0, this is not a guarantee; which is exactly what we observed in our empirical results.

Another way of limiting the issue of propensity overfitting is by introducing an IML term to the objective, as done in PIL-IML [129]. By penalising the objective with the Kullback-Leibler divergence between the learned and logging policies, it effectively favours those policies that *imitate* π_0. In most of their experimental setup, π_0 is a value-based linear model. Therefore, π_0 holds information on the non-clicked training samples and learning to imitate π_0 indirectly transfers this signal to the learned policy. It is clear to see that the quality of the policy learned through such an optimisation procedure is highly dependent on the quality of π_0, and that it might have adverse effects for highly skewed logging policies (e.g. an ϵ-greedy based approach). Note that this type of soft constraint where the learned policy should be "similar" to the logging policy is also induced by other common techniques, such as the variance regularisation method adopted in POEM [210].

As pointed out in Section 6.2, value- and policy-based methods can be formulated such that they have an identical parameterisation (Equations 6.6 and 6.3). This allows us to present a combined MLE-CRM objective that optimises a weighted average of the two. Due to the softmax-formulation from CRM, the model will be less prone to over-estimate under-explored actions. The logistic likelihood term can be seen as a regularisation term that ensures the dot-product between $\mathbf{x}$ and $\boldsymbol{\theta}_{\cdot,a}$ is low for un-clicked $(\mathbf{x}, a)$ samples in the training data, information that standard policy-based methods fail to exploit. We call this joint objective "Dual Bandit", as it provides a principled way to combine the best of the value- and policy-based worlds, whilst alleviating their individual weaknesses. Equation 6.9 shows the novel objective, where $0 \leq \alpha \leq 1$ is a hyperparameter that controls the influence of the log-likelihood on the final estimate. As such, both the contextual bandit and likelihood approaches can be seen as special cases of the dual bandit model, for $\alpha = 0$ and $\alpha = 1$ respectively. The contextual bandit aims to learn a probability distribution over n items, and its objective can be interpreted as the expected number of clicks under the new policy. The likelihood approach aims to learn whether an action led to a click, where the objective reflects whether the training sample supports the model parameters. Due to this disparity, α will act as a rescaling factor on top of a blending parameter. Keeping this in mind, higher values of α don't map linearly to higher importance of the likelihood approach compared with the contextual bandit.

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} (1-\alpha)\left(\sum_{i=1}^{N} c_i \frac{P(A=a_i|\mathbf{x}_i,\boldsymbol{\theta})}{\pi_0(a_i|\mathbf{x}_i)}\right) \\ &+\alpha\left(\sum_{i=1}^{N} c_i \ln\left(P(C=1|\mathbf{x}_i,a_i,\boldsymbol{\theta})\right) + (1-c_i)\ln\left(1-P(C=1|\mathbf{x}_i,a_i,\boldsymbol{\theta})\right)\right) \\ &= \arg\max_{\boldsymbol{\theta}} (1-\alpha)\left(\sum_{i=1}^{N} c_i \frac{[\text{softmax}(\mathbf{x}_i^\intercal\boldsymbol{\theta})]_{a_i}}{\pi_0(a_i|\mathbf{x}_i)}\right) \\ &+\alpha\left(\sum_{i=1}^{N} c_i \ln\left(\sigma(\mathbf{x}_i^\intercal\boldsymbol{\theta}_{\cdot,a_i}+b)\right) + (1-c_i)\ln\left(\sigma(\mathbf{x}_i^\intercal\boldsymbol{\theta}_{\cdot,a_i}+b)\right)\right)\end{aligned} \tag{6.9}$$

As minimising $\pi_{\boldsymbol{\theta}}(a|\mathbf{x})$ for all seen $(\mathbf{x},a)$-pairs will negatively impact the cross-entropy until it dominates the objective, this solves propensity overfitting as well. Unlike the NormPOEM-objective introduced by Swaminathan and Joachims [212], our objective decomposes into a sum over the observed samples. Because of this, it is perfectly suitable for optimisation through stochastic methods such as SGD and, as a consequence, applicable to large-scale problems and training deep neural networks. In this sense it is similar to BanditNet [88], but the Dual Bandit formulation additionally guarantees the inclusion of un-clicked samples in the model. Naturally, the IPS estimator in the first term can be replaced by the logarithmic IPS estimator introduced in Equation 6.8, or any other counterfactual estimator provided they are continuous and differentiable. SVP or IML terms could be naturally included in the objective, but our experiments show that the Dual Bandit is already highly competitive without them.

6.4 Experimental Results

In what follows, we experimentally validate the efficacy of counterfactual learning approaches presented in previous and this work, with a focus on the task of recommendation. In order to evaluate these approaches effectively, we need a dataset containing logged feedback for context-action pairs, along with the logging propensity for the action performed. Related work has evaluated counterfactual learning methods on multi-class, multi-label or LTR tasks [211, 212, 88, 71], synthetically generating bandit feedback samples for a certain logging policy and existing datasets. What makes the recommendation task fundamentally different from the aforementioned settings, is that access to the true labels (i.e. how likely a user is to click on a given recommendation) becomes impractical, and effective offline evaluation thus much less straightforward. Recent work that focuses on the recommendation use-case adopts a supervised-to-bandit conversion on existing datasets, and custom simulated datasets that assume deterministic rewards [128], or shows empirical success through live experiments [25]. In order to aid in the reproducibility of the research presented in our work, we adopt the RecoGym simulation environment in our experiments [176]. RecoGym provides functionality to generate offline logs under a given logging policy (for training and/or evaluation), and allows for the opportunity to simulate online experiments such as A/B-tests. More information regarding the specific setup of our experiments can be found in the reproducibility appendix.

All the methods discussed throughout this work are optimised with the full-batch L-BFGS algorithm, in order to avoid the choice of optimiser to be a confounding factor. Although some of the objectives presented in this work are non-convex and non-smooth, the choice of L-BFGS is theoretically well-supported [244, 111] and has been empirically shown to yield good performance in previous work [103, 212]. We aim to answer the following research questions:

RQ1 How does the logarithmic IPS (or PIL) estimator $\hat{R}_{\ln(\text{IPS})}$ impact existing off-policy learning methods?

RQ2 How do the various methods presented in this paper compare in terms of performance in a recommendation setting?

RQ3 How sensitive is the performance of the learned models with respect to the quality of the initial logging policy π_0?

RQ4 How do the number of items n and the number of available samples N influence performance?

Logging policies

We now describe the logging policies that we employ to generate logged bandit feedback samples that serve as the training datasets to our methods.

Uniform. The uniform logging policy chooses its actions uniformly at random. Thus, every item's probability of being recommended is $\pi_{\text{uniform}}(a|\mathbf{x}) = \frac{1}{n}$, independent of the context. As a consequence, IPS reweighting does not have any impact, because all the weights would be identical. Data logged uniformly at random contains no biases, and learning from it is a considerably easier task than otherwise. Nevertheless, real-world data will usually not be logged under this type of policy, as complete randomisation can significantly impair user satisfaction. It is an idealised and unrealistic setting, but it provides interesting insights nonetheless.

Popularity-based. A simple yet effective baseline policy is to sample actions with probabilities proportionate to the occurrence frequency of the item in the user's historical *organic* interactions. In a toy setting with 3 products and a user state of $[3,1,0]$, this means we sample item 1 with probability $\frac{3}{4}$, item 2 with probability $\frac{1}{4}$, and we don't sample item 3. In general, for a user history $\mathbf{x}$, $\pi_{\text{pop}}(a|\mathbf{x}) = \frac{\mathbf{x}_a}{\sum_{i=1}^{n}\mathbf{x}_i}$. This policy does not have full support over the item catalogue, violating the assumptions that guarantee importance sampling to yield an unbiased estimate [156]. As a consequence, learning an effective policy from data logged under such a policy can become problematic (especially for value-based methods). This can be mitigated by adopting an ϵ-greedy scheme: with probability ϵ, take an action uniformly at random; with probability $1-\epsilon$, sample from the original probability distribution. Clearly, when $\epsilon = 0$, this reverts to the original policy. Although learning from data logged under a policy that does not have full support over the item catalogue loses some theoretical guarantees, we believe it to be closer to a realistic environment. In many real-world use-cases, various items may be non-recommendable, due to recency, stock, licensing, business rules, et cetera. Furthermore, as real-world item catalogues are usually vast, it is realistic to randomise over a smaller set of candidate items. To reflect these real-world constraints, we include both variants in our experiments.

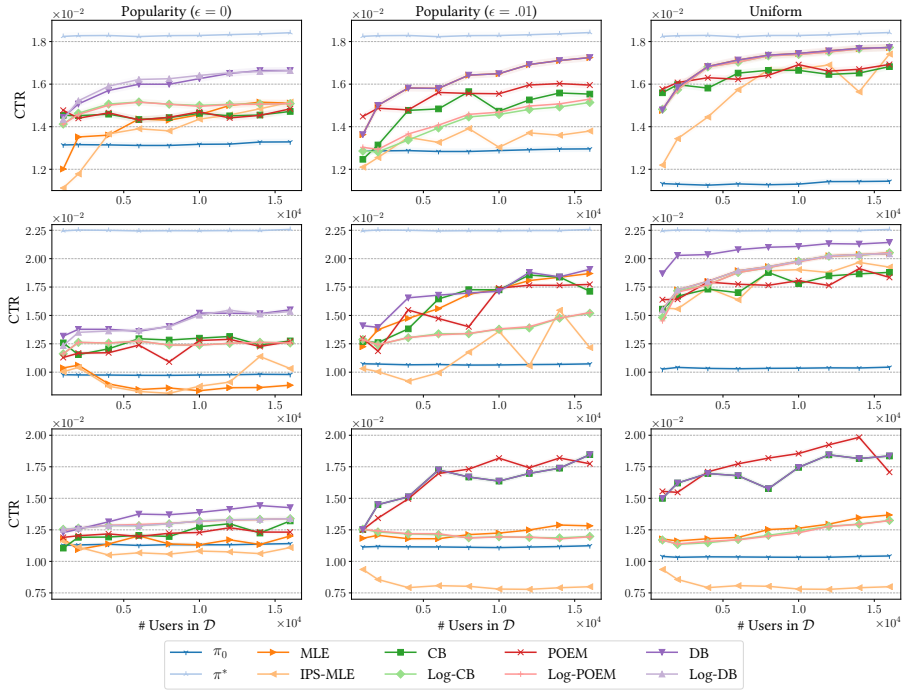


Figure 6.2: Experimental results from a range of A/B-tests for different settings in the RecoGym environment. Every column corresponds to a different logging policy, rows correspond to action spaces with $n \in \{10, 25, 100\}$. The size of the training sample is increased over the x-axis, the y-axis shows the average attained CTR over 5 runs, along with the 95% confidence interval.

Discussion

We simulate A/B-tests for varying amounts of training data logged under the different policies presented in the previous subsection, and report the approaches' attained CTR measurements in Figure 6.2. A more in-depth overview of our experimental setup can be found in the reproducibility appendix. The logging policy is denoted as π_0, an oracle policy that always performs the action with the highest probability of leading to a click is shown as π^*. While such a skyline is unattainable, it is interesting to analyse the *regret* of the competing methods. We do not show the horizontal skyline on the row corresponding to $n = 100$ as it skews the y-axis limit of the plots, but it occurs around 2.44%. All reported results are averaged out over multiple runs, and show the 95% confidence interval.

As discussed in Section 6.2, the "CB" objective is equivalent to a policy gradient method with a single-step horizon. "Log-CB" corresponds to the PIL-IML objective [129], without an accompanying IML term. As we don't suffer from propensity overfitting in this setup, and most logging policies are severely skewed, our hyperparameter tuning procedure showed the optimal weight for this term to be 0. This

was also the case for the Lagrange multiplier λ from BanditNet [88], reverting it back to the CB formulation. "Log-POEM" is POEM with the logarithmic estimator, "DB" and "Log-DB" are the Dual Bandit objective for the traditional and lower-bound estimators respectively. Every column represents a different logging policy. From left to right, these plots can be interpreted as going from the "most" to "least" realistic settings. Every row represents a differently sized action space, going up to 100. Larger action spaces are very relevant to the recommendation use-case, and we wish to study the generalisability of our results to them in future work. From a scalability perspective, this would require several adaptations to be made to the model formalisation in Section 6.2.

Effects of the Convex Policy Lower-bound (RQ1) We observe that the logarithmically transformed estimator positively influences results for both the popularity-based and the uniform logging policy. Moreover, we observed much more stable learning behaviour over various runs when using $\hat{R}_{\ln(\text{IPS})}$. As we increase the number of users in the training data, the performance of methods trained using $\hat{R}_{\ln(\text{IPS})}$ consistently improves. Methods using $\hat{R}_{\text{IPS}}$ showed far less consistent behaviour, as well as more variance across runs. This is consistent with the findings from Ma et al., as they primarily use the logarithm as a variance reduction technique. Aside from that, the logged estimator forces the model to take positive samples for *all* actions into account, leading to less overfitting on solely the *best* empirical actions.

While still improving over the logging policy and showing consistent behaviour, $\hat{R}_{\ln(\text{IPS})}$ hurts performance for the ϵ-greedy logging policy. Because it does not allow a single clicked sample to be missed, it is all the more sensitive to clicks on rare actions that might actually be suboptimal recommendations. Recent work on addressing click noise due to trust bias might provide a way of handling noisy training data in policy learning too [5].

Our experiments deal with the setting where the logging propensities $\pi_0(a|x)$ are known and exact. As a consequence, there isn't always a need to be conservative, which is what the convex lower bound does. Related work addresses settings where the logging propensities are unknown, learning an approximate $\hat{\pi_0}$ alongside their new policy [25]. In combination with highly stochastic rewards and small treatment effects, distinguishing the empirically optimal action becomes even more troublesome and noisy (see Fig. 6.1, for larger sample sizes, small variations on $\hat{\pi_0}$ may entirely flip the optimum). We expect the conservative logarithmic estimator to prove its worth even further in these settings.

Performance comparison (RQ2-4) Key observations from these results are the following: 1. In virtually all settings, the Dual Bandit approach achieves the best performance. Gains are the most tangible in cases where the logging policy does not randomise over the entire action space, as this is where classical value-based methods tend to fail. Policy-based methods can still improve upon the logging policy in these cases, but their performance does not seem to greatly improve with the size of the training sample. The Dual Bandit exhibits significant benefits over solely using value- or policy-based approaches. This suggests these families are complementary, and capture different relationships from the data. 2. As the logging policy randomises more uniformly, the performance of competing methods tends

to converge. A logging policy with support over the entire action space positively influences the performance of the value-based approaches, but the Dual Bandit either improves or reproduces their performance (for $\alpha = 1$). 3. In many cases, SVP as used in POEM has a positive impact on the traditional contextual bandit approach, but the regularisation strength λ is not straightforward to tune. The parameter is essentially unbounded and highly dependent on the variance in the data. An SVP term could straightforwardly be added to the DB objective. IPS reweighting for MLE disturbs the stability of the method, providing mixed results. In the majority of the cases, it causes a significant drop in performance. 4. Most of the compared methods gain significantly in performance when the size of the training sample increases. For sufficiently large enough samples, we expect all methods to slowly converge to the optimum. The most interesting and realistic results deal with large action spaces, finite samples, and limited randomisation, which is exactly where the Dual Bandit shows its superiority.

6.5 Conclusions

In this work, we have motivated the use of methods that exploit bandit feedback for recommendation tasks. Due to the limitations of offline datasets, we introduced simulation environments as an alternative and reproducible evaluation approach. We have presented an overview of the state-of-the-art in counterfactual learning, reviewing commonalities and key differences in existing algorithms. In doing so, we highlighted the fact that value- and policy-based approaches can be formulated with an identical parameterisation. This insight enabled us to propose a new combined MLE-CRM objective, aiming to unify both families. Various experiments underline the superiority of this Dual Bandit approach, excelling the most in the presence of finite samples and limited randomisation. Additionally, we discussed specific properties of the recommendation task such as stochastic and sparse rewards, small treatment effects, and large action spaces. To effectively deal with some of these issues, we introduced a logarithmic variant of the conventional IPS estimator and empirically show how it can further improve performance in the right environments but hurt in the wrong ones, connecting it to analogous findings in related work. Our findings represent the first general empirical study of the use of counterfactual techniques in a bandit-feedback recommendation scenario.

We believe that our work opens up many interesting directions for future research. First, we wish to include additional recent an advanced policy learning and evaluation methods in our comparison, such as DM, DR, MRDR, CAB and others. We specifically wish to further explore the relation between DR and our Dual Bandit approach. Second, the limitations of value-based approaches can be handled in other ways than we presented. Probabilistic models and Bayesian approaches that incorporate priors are a way forward in this aspect, as they can naturally handle the uncertainty that arises in recommendation scenarios. All the models discussed in this work require $\mathcal{O}(n^2)$ parameters. By using latent embeddings for the user-state, we could significantly reduce the parameter space to $\mathcal{O}(kn)$. This would enable much larger action spaces to be considered, and is very relevant to the recommendation use-case. As the quality of the learned models is then also dependent on the quality of the embeddings, we leave this for future work. Finally, we can extend the

value-based methods we compare in this work to higher-order models with non-linearities such as deep neural networks, and extend our analysis to include recent advances in counterfactual learning for top-K recommendations [25], two-stage recommendation pipelines [128], or multiple iterations of logging and learning, touching on the exploration-exploitation trade-off.

Reflections

The overview presented in this Chapter maps ideas and methods from the broader machine learning field to the specific problem of recommendation. In order to keep things focused – many simplifying assumptions were made that should be tackled in order to properly represent a real-world use-case. Dealing with *slate* instead of *single-item* recommendations, as well as more dynamic environments make up interesting avenues for future work that fit the spirit of this thesis. Moreover, although the "Dual Bandit" objective is empirically motivated and validated, a deeper theoretical perspective on the connections with doubly robust methods would be interesting and valuable.

6.6 Reproducibility Appendix

In what follows, we describe our experimental set-up in further detail. We make use of publicly available simulators to aid in the reproducibility of our work. Implementations of all methods are written in Python3.7, using PyTorch [162].[6]

Derivation of $\hat{R}_{\text{IPS}}$ lower bound

We derive the lower bound for the empirical IPS estimator as follows:

$$\begin{aligned}\hat{R}_{\text{IPS}}(\pi_{\boldsymbol{\theta}}, \mathscr{D}) &= \sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)} \pi_{\boldsymbol{\theta}}(a_i|\mathbf{x}_i) \\ &= \frac{1}{\sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)}} \sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)} \sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)} \pi_{\boldsymbol{\theta}}(a_i|\mathbf{x}_i) \\ &= \sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)} \sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)} \frac{1}{\sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)}} \pi_{\boldsymbol{\theta}}(a_i|\mathbf{x}_i)\end{aligned}$$

Subsequently taking the logarithm and applying Jensen's inequality (that says $\log(E(x)) \geq E(\log(x))$, we have that:

$$\begin{aligned}\log(\hat{R}_{\text{IPS}}(\pi_{\boldsymbol{\theta}}, \mathscr{D})) \geq &\log\Big(\sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)}\Big) \\ &+ \sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)} \frac{1}{\sum_{i=1}^{n} \frac{c_i}{\pi_0(a_i|\mathbf{x}_i)}} \log(\pi_{\boldsymbol{\theta}}(a_i|\mathbf{x}_i))\end{aligned}$$

The RecoGym Environment

We make use of the RecoGym simulation environment in our experiments [176].[7] An OpenAI Gym-inspired framework, it provides a standard, robust and reproducible way of evaluating recommendation approaches through simulation. It provides functionality to generate offline logs under a given logging policy (for training and/or counterfactual evaluation), and additionally allows for the opportunity to simulate online experiments such as A/B-tests [82]. Below, we provide an overview of the actual simulation framework behind RecoGym. Note that this is not our contribution, and all merit should go to the authors of the original paper [176]. We merely aim to provide a comprehensive overview of its inner workings for the interested reader, as the original paper only presents it from a higher level perspective.

Users and items are modelled via an underlying latent-factor model, as is widely accepted in the literature [98]. When the environment is set up, the system generates an embedding consisting of k latent factors for every item. These embeddings are represented in a real-valued matrix $\Gamma \in \mathbb{R}^{n \times k}$ and drawn from a multivariate Gaussian distribution centred around 0 with unit variance: $\Gamma \sim \mathcal{N}(0,1)$. A notion of item popularity is modelled as an additive bias per item: $\mu \in \mathbb{R}^n$, normally distributed with a configurable variance: σ_μ^2. Γ and μ directly impact how users organically interact

[6] Source code available at `https://github.com/olivierjeunen/dual-bandit-kdd-2020`.
[7] https://github.com/criteo-research/reco-gym

Parameter	k	σ_μ	σ_u	n
Value	5	3	0.1	{10,25,100}

Table 6.2: Parameter configurations for the RecoGym environment we used for our experiments.

with these items. The RecoGym authors argue that a given user's bandit behaviour for an item would be different, but similar to the organic behaviour between that user and item. The rationale being that users' natural browsing behaviour and reactions to recommendations are related, but not an exact one-to-one mapping. As such, bandit embeddings and popularities are obtained by performing a transformation f on the organic parameters: $B, \mu^{'} = f(\Gamma, \mu)$. We will not go further in-depth on how this transformation f is performed, and refer interested readers to the RecoGym source code for further information.

Users are described by vectors that conceptually reside in the same latent space as the items. That is, a user embedding $\omega \in \mathbb{R}^k$ is sampled from a multivariate Gaussian distribution with configurable variance: $\omega \sim \mathcal{N}(0, \sigma_u^2)$. Users are simulated sequentially and independently. The behaviour of a single user is modelled as a Markov chain, being either in an "organic" or "bandit" state. The organic state implies the user is currently browsing the item catalog, and generates organic user-item interactions. The bandit state on the other hand requires interventions from the agent, and generates the labeled training data we use for learning throughout this work. We use the default values for the state transition probabilities.

The next item a user views organically is sampled from a categorical distribution, where the individual probabilities for every item are proportional to how similar the user and the respective item's latent organic embeddings are. Formally, given u we draw

$$i \sim \text{Categorical}(\rho), \text{ where } \rho_i \propto \exp(\omega \Gamma_{i,\cdot}^{\intercal} + \mu_i).$$

When an action a is performed by an agent, the probability of it actually leading to a click is Bernoulli-distributed, with the probability of success being dependent on the similarity between the user and the recommendation's bandit embedding:

$$c \sim \text{Bernoulli}(p), \text{ where } p \propto \sigma(\omega B_{a,\cdot}^{\intercal} + \mu_a^{'}).$$

Naturally, consistently taking the action a^* with the highest probability of leading to a click leads to an optimal recommendation policy:

$$a^* = \arg\max_a \omega B_{a,\cdot}^{\intercal} + \mu_a^{'}.$$

This is exactly the skyline policy π^* shown in Figure 6.2. Naturally, none of these parameters are accessible to the learning algorithms we consider in our work: all they observe and learn from are the $(\mathbf{x}, a, p, c)$-samples as introduced in Section 6.2. Furthermore, the relation between the user history of organic interactions $\mathbf{x}$ and the probability of a click for a given action is non-linear, whereas all the approaches we study model them as such.

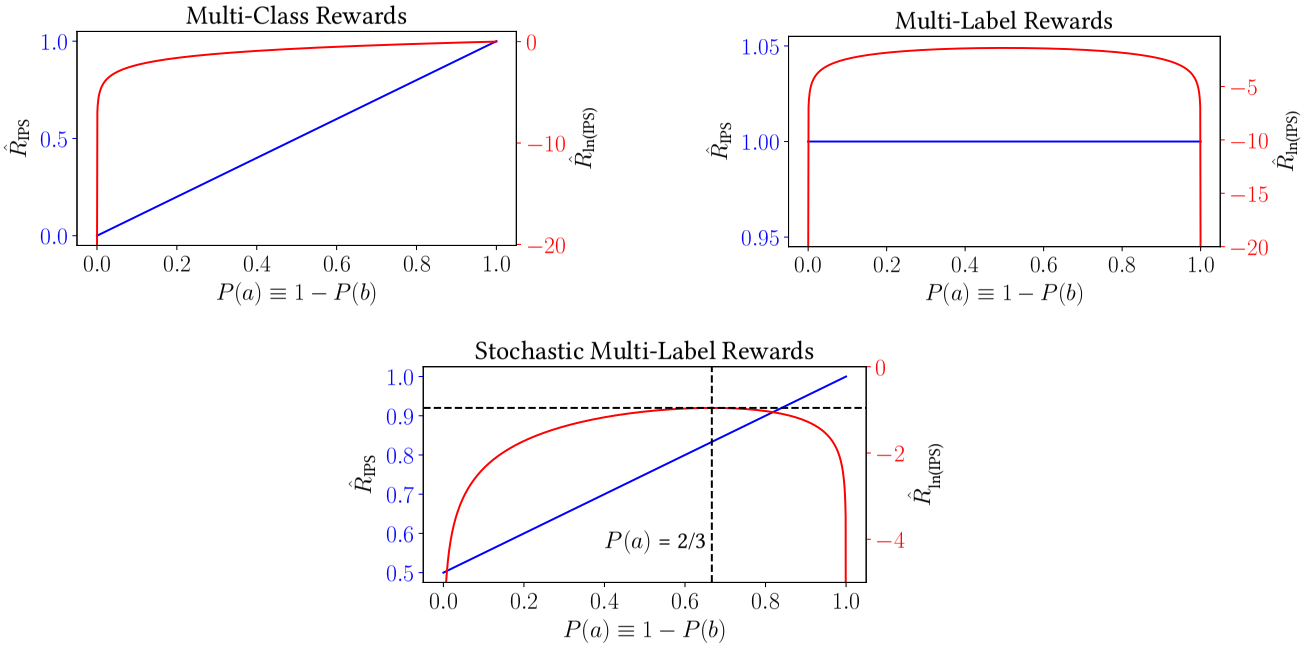


Figure 6.3: Behaviour of the objective functions corresponding to the $\hat{R}_{\text{IPS}}$ (blue, leftmost y-axis) and $\hat{R}_{\ln(\text{IPS})}$ (red, rightmost y-axis) estimators for the toy examples presented in Section 6.3. We see that both estimators agree on an optimal policy in the case of deterministic rewards, but this is no longer true in the case of a stochastic reward process.

Experimental Setup

Here, we list the details for the experimental setup used throughout Section 6.4. We vary the number of items $n \in \{10, 25, 100\}$. As we model users in a bag-of-word fashion and our models consist of n^2 parameters, this prohibits us to increase n significantly. Modelling users with latent embeddings is a solution for this, but falls outside the scope of this work. Our experimental observations are general and translate to larger action spaces, with the exception that effect sizes typically diminish. The number of unique users in the training sample varies in {1 000, 2 000, 4 000, 6 000, 8 000, 10 000 12 000, 14 000, 16 000}. These settings lead to anywhere 80 000 and 1 280 000 bandit feedback samples. The empirical CTR for the logging policy in the training data varies from 1.1% to 1.4%, yielding a wide spread between roughly 10 and 1 800 positive samples per item on average. Every contending method was trained until convergence and subsequently tested in a simulated A/B-test with 10 000 users. This process was repeated with 5 different random seeds, aggregating results to provide a robust CTR estimate with a tight confidence interval. We report the 95% confidence interval with error bars on the plot. Hyper-parameters were optimised through a grid-search where the training set consisted of 5 000 users, and validated through simulated A/B-tests with 10 000 users as outlined above, albeit with different random seeds as to ensure the training, validation and test users to be disjoint. Grid searches were analogously repeated and results averaged out over 5 runs to reduce the inherent noise of the simulation. Note that results from the grid-search with only 5 000 users for training might not yield the optimal hyper-parameters for smaller or larger training samples. Furthermore, optimising hyper-parameters through online experiments might not accurately reflect a real-world situation, but it ensures a fair comparison among algorithms. We varied the SVP-strength for POEM $\lambda \in \{.0, 0.05, .1, .25, .5, 1.0, 1.5, 2.0\}$ and DB's CRM-MLE balance $\alpha \in \{.0, .8, .85, .925, .95, .975, 1 - 1e2, 1 - 1e3, 1 - 1e4, 1.0\}$, and report results only for the optimal values. However, we observed much more stable results for varying α than for varying λ. $\alpha = 0$ corresponds to the (Log-)CB objective, $\alpha = 1$ to MLE. Analogously, $\lambda = 0$ corresponds to (Log-)CB, whereas $\lambda \to \infty$ goes to the logging policy. As we mentioned in Section 6.4, none of the weights ϵ for the IML-term used in PIL-IML improved performance. We varied the weight $\epsilon \in \{.0, .05, .1, .15, .20, .25, .50, 1.0\}$, and observed a steady decrease in attained CTR. The Lagrange multiplier γ for BanditNet was varied over {.0, .125, .25, .5, .75, .875, 1.0}, and was equally unable to improve results. Note that for $\gamma = 1.0$, the model only learns from the non-clicks and ignores the clicks. In this setting, propensity overfitting occurs. We observed that adding an additional IML term to BanditNet was then indeed able to prevent this, but the quality of the learned models remained subpar to those optimised for the original objectives.

Behaviour of Convex Policy Lower-Bound

Section 6.3 and Figure 6.1 discuss how the logarithmic IPS lower bound penalises policies that miss a single positive action in the training sample, leading to an allocation of probability mass that is proportional to the observed reward per action. Figure 6.3 shows how this only impacts the optimum of the objective function when rewards are stochastic. In the deterministic multi-class example, only class a is

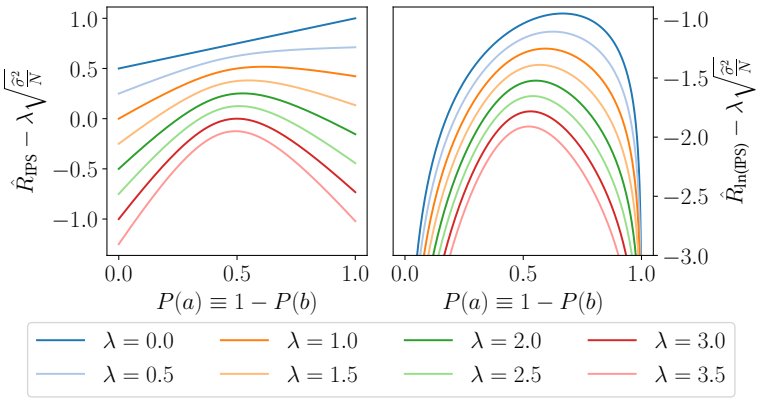


Figure 6.4: Impact of the SVP term on both the $\hat{R}_{\text{IPS}}$ (left) and $\hat{R}_{\text{ln(IPS)}}$ (right) estimators, for varying strengths λ and the stochastic example setup in Figure 6.1. The logarithmic estimator penalises degenerate policies, and SVP encourages imitation of the logging policy.

correct. In the deterministic multi-label example, labels a and b are both equally correct and $\hat{R}_{\text{IPS}}$ becomes indifferent. In the stochastic multi-label case, actions a and b are not equally likely to lead to a positive reward, and the optimum for $\hat{R}_{\text{ln(IPS)}}$ diverges from that of $\hat{R}_{\text{IPS}}$.

Figure 6.4 shows the impact of adding an additional SVP term to the optimisation objectives with varying strengths λ, for the stochastic multi-label setup. λ is the SVP strength, $\widehat{\sigma}^2$ an empirical estimate of the variance. Naturally, as $\lambda \to \infty$, the objective is increasingly dominated by the SVP term and the optimal policy moves towards the logging policy (the uniform distribution in this toy example). Intuitively, we can see that adding an SVP term to the original estimator $\hat{R}_{\text{IPS}}$ has similar effects as using the logarithmic variant: extreme values (0 and 1) are increasingly penalised, making the objective better behaved. Adding the SVP term to the logarithmic estimator $\hat{R}_{\text{ln(IPS)}}$ has less dramatic effects and does not change the shape of the objective function as much, as it was concave to begin with. It shifts the optimum towards imitating the logging policy ($P(a) = \pi_0(a)$).

CHAPTER 7

Pessimistic Reward Models for Off-Policy Learning in Recommendation

Methods for bandit learning from user interactions often require a model of the reward a certain context-action pair will yield – for example, the probability of a click on a recommendation. This common machine learning task is highly non-trivial, as the data-generating process for contexts and actions is often skewed by the recommender system itself. Indeed, when the deployed recommendation policy at data collection time does not pick its actions uniformly-at-random, this leads to a selection bias that can impede effective reward modelling. This in turn makes off-policy learning – the typical setup in industry – particularly challenging. In this work, we propose and validate a general pessimistic *reward modelling approach for off-policy learning in recommendation. Bayesian uncertainty estimates allow us to express scepticism about our own reward model, which can in turn be used to generate a conservative decision rule. We show how it alleviates a well-known decision making phenomenon known as the Optimiser's Curse, and draw parallels with existing work on pessimistic policy learning. Leveraging the available closed-form expressions for both the posterior mean and variance when a ridge regressor models the reward, we show how to apply pessimism effectively and efficiently to an off-policy recommendation use-case. Empirical observations in a wide range of environments show that being conservative in decision-making leads to a significant and robust increase in recommendation performance. The merits of our approach are most outspoken in realistic settings with limited logging randomisation, limited training samples, and larger action spaces.*[1]

[1] This chapter is based on work published in the *Proceedings of the 2021 ACM RecSys Conference* as "Pessimistic Reward Models for Off-Policy Learning in Recommendation" by Olivier Jeunen and Bart Goethals [78].

7.1 Introduction

Many modern web services deploy machine-learnt models on their websites to help steer traffic towards certain items. Retail websites try to predict which of their recommendations might lead to a sale, music streaming platforms suggest songs in your queue to optimise engagement metrics, search engines will often rank items in decreasing estimated probability of receiving a click, et cetera. These models are generally part of a 1. collect data, 2. train model, 3. deploy model loop, where models are iteratively retrained and earlier versions influence the training data that is used for future iterations. This correlation between the deployed model and the collected training data can impede effective learning if we are unable to somehow correct for the bias it creates. Recent work has shown how such "algorithmic confounding" leads to feedback loops when left untreated, which can be detrimental to the users, the platforms, and the models themselves [22, 130]. Traditional recommendation research assumes *organic* user-item interaction data to bypass any feedback loops that might occur due to deployed systems. In this work, we wish to learn directly from the logs of the deployed recommender system, casting the recommendation task in a bandit learning framework [25, 218].

Learning from biased data is not a novel problem, and many *unbiased* learning procedures have proven to be effective in counteracting *position*, *presentation*, *trust* and *selection* bias [87, 4, 25, 5, 151]. These methods typically make use of importance sampling or Inverse Propensity Score (IPS) weighting, in order to obtain an unbiased estimate of the counterfactual value-of-interest [156]. They aim to answer questions of the form: "*What click-through-rate would this new policy have obtained, if it were deployed instead of the old policy*"? The policy that maximises the answer to this question is the policy we want to deploy. Answering this question effectively and efficiently, however, is not an easy feat.

IPS is the cornerstone of counterfactual reasoning [19], but by no means a silver bullet. It is plagued by variance issues that are exacerbated at scale, often making it hard to deploy these systems reliably in the real world [49]. Furthermore, the randomisation requirements for IPS to remain unbiased are often unrealistic or simply unattainable. Recent work explores the effectiveness of counterfactual models in cases where IPS assumptions in the training data are violated, highlighting an interesting area for future research and a commonly-encountered yet understudied problem [178, 84].

An alternative family of approaches are so-called "value-based" models. These methods rely on an explicit model of the reward conditioned on a context-action pair – for example, the probability of a user clicking on a given recommendation when it is shown [137, 59]. When prompted, the model then simply takes the action that maximises the probability of a positive reward, given the presented context and the learnt model. Aside from the typical problems of model misspecification in supervised learning [132], another issue with value-based methods is that learning an accurate model of the reward is not straightforward when the collected training data is heavily influenced by the model that was deployed in production at the time. Methods that use IPS to re-weight the data as if it were unbiased exist [191], but their performance when deployed as recommendation policies is often disappointing in comparison with policy-based methods or even vanilla reward models [146, 84]. Furthermore, the logging policy is not always known before-hand, and even when we

do obtain unbiased value estimates we should expect the true obtained reward from acting on them to be disappointing with respect to the estimates – a phenomenon known as "the Optimiser's curse" [194].

In this paper, we focus on improving the recommendation performance of policies that rely on value-based models of expected reward. We propose and validate a general pessimistic reward modelling framework, with a focus on the task of off-policy learning in recommendation. Bayesian uncertainty estimates allow us to express scepticism about our own reward model, which can then in turn be used to generate conservative decision rules based on the resulting reward predictions – instead of the usual ones based on Maximum Likelihood (MLE) or Maximum A Posteriori (MAP) estimates. We show how closed-form expressions for both the posterior mean and variance can be leveraged to express pessimism when a ridge regressor models the reward, and how to apply them effectively and efficiently to an off-policy recommendation use-case. Our approach is agnostic to the logging policy, and does not require (a model of) propensity scores to quantify selection bias. As a result, we are not bound to the strict assumptions that make IPS work, and abide by statistical conjectures such as the likelihood principle [14]. Additionally, we show how our proposed framework lifts the Optimiser's Curse and effectively limits post-decision disappointment.

The empirical performance of counterfactual learning methods is often reported with a supervised-to-bandit conversion on existing multi-class or multi-label classification datasets [88, 129, 206]. As publicly available datasets with propensity information are scarce, this inhibits robust and reproducible evaluation of such methods on off-policy *recommendation* tasks. In line with recent work [84, 180, 81, 77], we adopt the RecoGym simulation environment in our experiments to yield reproducible results that are aligned with the specifics of real-world recommendation scenarios, such as stochastic rewards, limited randomisation and small effect sizes [176]. An added advantage of adopting such a simulation framework is the freedom gained to change environmental parameters and better understand how these changes affect the trade-offs between different methods.

Empirical observations for a wide range of configurations show that our proposed approach of pessimistic decision-making leads to a significant and robust increase in recommendation performance. The merits of our method are most outspoken in realistic settings where the amount of randomisation in the logging policy is limited, training sample sizes are small, and action spaces are large. All source code to reproduce the reported results is available at github.com/olivierjeunen/pessimism-recsys-2021. To summarise, the main contributions we present in this work are:

1. We propose the use of explicit pessimism in reward models for off-policy recommendation use-cases.

2. We introduce the decision-making phenomenon known as the Optimiser's Curse in the context of recommendation, and show how naive reward models suffer from it. In contrast, principled pessimism lifts the curse.

3. We show how to leverage closed-form estimates for the posterior mean and variance of a ridge regressor to express pessimism, and how to apply this effectively and efficiently to an off-policy recommendation use-case.

4. Empirical observations from reproducible simulation experiments highlight that explicit pessimism significantly and robustly improves online recommendation performance, compared to ML or MAP-based decision-making.

7.2 Background and Related Work

We are interested in modelling recommendation systems following the "Batch Learning from Bandit Feedback" (BLBF) paradigm [210]: a general machine learning setting that properly characterises the off-policy recommendation use-case as it widely occurs in practice. A recommender system is modelled as a stochastic policy π that samples its recommendations from a probability distribution over actions conditioned on contexts: $\mathsf{P}(A|C,\pi)$, often denoted $\pi(A|C)$. Note that π is modelled to be stochastic for generality, but that deterministic systems are implied when $\mathsf{P}(A|C,\pi)$ is a degenerate distribution. Contexts are drawn from some unknown marginal distribution $\mathsf{P}(C)$ and can represent a variety of information about the user visiting the system, such as their consumption history, the time of day and the device they are using. When talking about the feature vector for a specific context, we denote it as $\boldsymbol{c}$. Analogously, feature vectors for specific actions are represented as $\boldsymbol{a}$. The sets of all possible contexts and actions are $\mathscr{C}$ and $\mathscr{A}$, respectively. The combined feature representation of a context-action pair is $\boldsymbol{x} := \Phi(\boldsymbol{c},\boldsymbol{a})$, where Φ is a function that maps context- and action-features to a joint space. Note that this step – including interaction terms between contexts and actions – is necessary to allow for linear models to learn personalised treatments. Φ can be anything from a simple Kronecker product between one-hot-encoded contexts and actions [146], to a specialised neural network architecture that learns a shared embedding for multi-task learning [127, 254, 213]. In the off-policy or counterfactual setting, we have access to a dataset consisting of logged context-action pairs and their associated rewards: $\mathscr{D} := \{(c,a,r)\}$, where $c \sim \mathsf{P}(C)$, $a \sim \pi_0(a|c)$ and $r \sim \mathsf{P}(R|C,A)$. Here, r represents the reward that the system obtained from recommending a to c. In the general case this reward can be binary (e.g. clicks), real-valued (e.g. dwell time), or higher-dimensional to support multiple objectives (e.g. fairness and relevance) [138, 140]. The policy that was deployed at data collection time is called the logging policy (π_0). This type of setting is called "bandit feedback", as we only observe the reward of the actions chosen by the contextual bandit π_0. We place this paradigm at the focal point of our work, as it is the most closely aligned with the recommendation use-case that practitioners typically face in industry.

Learning to recommend from organic user-item interactions Most traditional approaches to recommendation do not make use of this type of experimental data tying recommendations to observed outcomes. Instead, they typically adopt observational datasets consisting of "organic" interactions between users and items, such as product views on retail websites. By framing the recommendation task as next-item prediction in such a setting, the goal of these systems is no longer that of learning optimal interventions. Maybe unsurprisingly, offline evaluation results in such environments are notoriously uncorrelated with online success metrics based on shown recommendations, making it harder to discern *true* progress with regard to online gains [48, 177, 75, 32]. Nevertheless, it is a very active research

area that yields many interesting publications and results every year. Recent trends are geared towards the use of Bayesian techniques that explicitly model uncertainty [116, 40, 189, 126], and linear item-based models that achieve state-of-the-art performance whilst being highly efficient to compute [147, 200, 85, 26, 28].

Off-policy learning from bandit feedback The bandit feedback setup described above finds its roots in the field of offline reinforcement learning (RL), with the additional simplifying assumption that past actions do not influence future states (more formally, the underlying Markov Decision Process consists of a single time-step) [108]. This type of learning setup is not specific to the recommendation task, and many learning methods are evaluated on simulated bandit feedback scenarios using general purpose multi-class or multi-label datasets. Approaches for off-policy learning optimise a parametric policy for some counterfactual estimate of the reward it would have obtained, if deployed.

The go-to technique that enables this type of counterfactual reasoning is importance sampling [156, 19]. Equation 7.1 shows how it obtains an empirical estimate for the value of a policy π, using data $\mathscr{D}$, and a model of the logging policy $\widehat{\pi_0}$. Many learning algorithms in this family aim to mitigate the increased variance that is a consequence of the IPS weights. Capping the probability ratio to a fixed value [69], self-normalising the weights [212, 88], imposing variance regularisation [134, 210], imitation learning [129] or distributional robustness [44, 192] on the learnt policy are commonly used tools to trade off the unbiasedness of IPS for improved variance properties in finite sample scenarios. Many of these techniques can be interpreted as a form of principled *pessimism*, where we would rather be conservative with the IPS weights than over-estimate the value of an action to a policy.

$$\widehat{V}_{\mathrm{IPS}}(\pi,\mathscr{D}) = \sum_{(c,a,r)\in\mathscr{D}} r \cdot \frac{\pi(a|c)}{\widehat{\pi_0}(a|c)} \tag{7.1}$$

$$\widehat{V}_{\mathrm{DM}}(\pi,\mathscr{D}) = \sum_{(c,a,r)\in\mathscr{D}} \sum_{a'\in\mathscr{A}} \pi(a'|c) \cdot \widehat{r}(a',c) \tag{7.2}$$

A conceptually simpler family of approaches are value-based methods, often referred to as Q-learning in the RL community, or the "Direct Method" (DM) in the bandit literature. Equation 7.2 shows how DM obtains an empirical estimate of policy π's value w.r.t. a dataset of logged bandit feedback $\mathscr{D}$. Value-based counterfactual estimators do not rely on a model of the logging policy, but rather learn a model of the reward an action will yield in a given context: $\widehat{r}(a,c) \approx \mathbb{E}[R|C=c, A=a]$. In practice, the available bandit feedback $\mathscr{D}$ is often split into disjoint training sets for the optimisation of the reward model and the resulting policy respectively. Nevertheless, it is easy to see that the optimal policy π^*_{DM} with respect to a given reward model places all its probability mass on the action with the highest estimated reward:

$$\pi^*_{\mathrm{DM}}(a|c) = \begin{cases} 1 & \text{if } a = \arg\max_{a'\in\mathscr{A}} \widehat{r}(a',c), \\ 0 & \text{otherwise.} \end{cases} \tag{7.3}$$

As a consequence, we can directly obtain a decision rule from the reward estimates and train the reward estimator on all available data [84]. Value-based methods

as laid out above are typically biased, but exhibit more favourable variance properties than IPS-based models. While policy-based methods for learning from bandit feedback need (a model of) the logging propensities [218, 25], this is not a constraint for the value-based family. When multiple logging policies are at play (e.g. during an A/B-test), this complicates the use of standard importance sampling techniques even further [3, 41].

A unifying family of doubly robust methods aims to marry these two types of approaches in an attempt to get the best of both worlds [36]. Recent advances in doubly robust learning typically optimise the trade-off between DM and IPS [205], optimise the reward model to minimise the overall variance of the estimator [43], or transform the IPS weights to minimise bounds on the expected error of the estimate [206]. Nevertheless, the performance of the reward model remains paramount for doubly robust approaches to attain competitive performance [81].

Off-policy learning for recommendation Methods that apply ideas from the bandit and RL literature to recommendation problems have seen increased research interest in recent years. Chen et al. extend a policy gradient-based method with a top-K IPS estimator and show significant gains from exploiting bandit feedback in online experiments [26]. In the top-1 use-case we consider with the additional independence assumption between current and future iterations, their method yields a policy that is analogous to one optimised for $\widehat{V}_{\text{IPS}}$ (Equation 7.1). This work has been extended to deal with two-stage recommender systems pipelines that are typically adopted to deal with large action spaces [128]. Xin et al. adopt a Q-learning perspective to deal with sequential recommendation tasks, exploiting both self-supervised (*organic*) and reinforcement (*bandit*) signals [236]. Analogously, Sakhi et al. propose a probabilistic latent model that combines organic and bandit signals in a Bayesian value-based manner [180]. The work of Jeunen et al. studies the performance of both value- and policy-based approaches when the organic data is only used to describe the context, proposing a joint policy-value approach that outperforms stand-alone methods without the need for an external reward model [84]. Their experimental set-up is the closest to the one we tackle in this work.

On-policy learning for recommendation Off-policy methods learn from data that was collected under a different policy. In contrast, on-policy methods learn from data that they themselves collect. In such cases, the well-known exploration-exploitation trade-off becomes important, as the policy needs to balance the immediate reward with the informational value of an action [114, 136]. Successful methods use variants of Thompson sampling [23, 135, 37] or confidence bounds [112]; recent work benchmarks a number of different exploration approaches to predict clicks on advertisements when the reward model is parameterised as a neural network [54]. Although the use-case we tackle in this work does not include any interactive component, we draw upon existing work in learning from on-policy bandit feedback to obtain improved, uncertainty-aware decision strategies in the off-policy setting.

Uncertainty estimation Both Thompson sampling and confidence-bound-based methods make use of a posterior distribution for the reward estimates, instead of the usual point estimate that is obtained from uncertainty-agnostic models. Principled

Bayesian methods can be used to obtain closed-form expressions for exact or approximate posteriors, but they are often restricted to specific model classes [23, 112]. The Bootstrap principle [38], its extensions [152] (originally proposed in the context of Q-learning), and Monte Carlo Dropout [46] can provide practical uncertainty estimates for general neural network models. The work of Guo et al. proposes a hybrid Bootstrap-Dropout approach, and validates the effectiveness of the obtained uncertainty estimates in an on-policy recommendation scenario [54]. Finally, other recent work shows promising results in inferring model uncertainty from neuron activation strength [27]. All these uncertainty estimation methods are complementary to the framework we propose in this paper.

7.3 Methodology and Contributions

The Optimiser's Curse in Recommendation

In what follows, we introduce the Optimiser's Curse [194] in the context of off-policy learning in recommendation scenarios. For illustratory purposes, we assume an immediate binary reward (e.g. a click) that follows a Bernoulli distribution with parameter p that is conditioned on the relevance of the given context-action pair. Nevertheless, the Optimiser's Curse is a general phenomenon that is by no means bound to these assumptions.

Suppose we have an action space $\mathscr{A}$ and for simplicity, but without loss of generality, assume that the probability of a positive reward is independent of the context. Now, every action $a_i \in \mathscr{A}$ has a *true* probability of leading to a click: $\mathsf{P}(R = 1|A = a_i) = p_i^*$. The goal of a reward model is to estimate these true Bernoulli-parameters p_i^*, yielding the estimated parameters $\widehat{r}(a_i) = \widehat{p}_i$. Widely used estimation methods include Maximum Likelihood (MLE) and Maximum A Posteriori (MAP) estimation. We aim to learn such a model based on a previously acquired log of training data, and assume that our obtained value estimates are conditionally unbiased in that $\forall i \in \{1,\ldots,|\mathscr{A}|\} : \mathbb{E}[\widehat{p}_i|p_1^*,\ldots,p_{|\mathscr{A}|}^*] = p_i^*$. Note that this assumption is already quite idealistic for many real world applications, and that it cannot be checked when we do not know the true parameters p^*. In practice, we can minimise the bias between the reward model and the empirical reward in the training sample.[2] Nevertheless, even in such an idealised setting, problems arise.

Once we have a reward model, we are ready to start showing recommendations to users. Analogous to Equation 7.3 we take the action with the highest estimated reward or Bernoulli-parameter, indexed by i^* :

$$a_{i^*} = \arg\max_{a_i \in \mathscr{A}} \widehat{r}(a_i). \tag{7.4}$$

After showing this recommendation to a user, we get to observe a sample from the true reward distribution: $r_{i^*} \sim \text{Bernoulli}(p_{i^*}^*)$. Now, the difference between the observed and estimated rewards $(r_{i^*} - \widehat{p}_{i^*})$ can be seen as the *post-decision surprise* we get from acting on the model $\widehat{r}$. Repeating this process and averaging the observed post-decision surprise yields the average expected surprise: $\mathbb{E}[p_{i^*}^* - \widehat{p}_{i^*}]$. The

[2] This type of "calibration" of the reward model with respect to the *empirical* reward distribution is often a requirement in computational advertising [137, 59], as a downstream bidding strategy then depends on the reward model.

Optimiser's Curse states that, even though the reward estimates are conditionally unbiased, this process leads to a *negative* expected surprise: $\mathbb{E}[p^*_{i^*} - \widehat{p}_{i^*}] \leq 0$, meaning that we incur *less* reward than predicted. This *disappointment* on average is not merely a result of the model itself (as it is unbiased), but rather a consequence of the decision making process that only considers the action with the highest estimated value $\widehat{p}_{i^*}$, leaving us especially vulnerable to actions with over-estimated rewards.

Smith and Winkler provide an excellent overview of this phenomenon, showing how it can be mitigated by adopting Bayesian methods with well-chosen priors [194]. They prove that, in the settings they consider, choosing actions based on MAP estimates alleviates any post-decision surprise *when these posteriors are unbiased*. This elegant theoretical finding is of limited practical use in our use-case. Indeed, we have no way to guarantee that the reward estimates we end up with are unbiased with respect to the true reward distribution parameters p^*. We can only check unbiasedness with respect to the empirically observed reward, which can be highly skewed due to the logging policy. Additionally, underfitting and model misspecification make this assumption of converging to the true parameters sound especially utopian [132]. To make matters worse, the training data $\mathscr{D}$ that is used to obtain the reward model $\widehat{r}$ is also highly dependent on this logging policy π_0, impeding effective reward modelling even before we take part in an ill-suited decision making process. Indeed, standard Empirical Risk Minimisation (ERM) focuses its efforts on context-action regions that are well-explored in the training data. This leaves us vulnerable when naively handling the resulting reward estimator $\widehat{r}$, because a single erroneously optimistic reward estimate can disturb the recommendation policy and decimate performance. The probability of this happening grows with the size of the action space and the level of "determinism" in the logging policy (more formally, decreasing entropy). Using (estimated) propensity scores to redistribute the errors in the model fit does not guarantee performance improvements in such cases [191, 146, 84].

Heteroscedasticity in Reward Estimates

Logging policies are typically not solely optimised for data collection. The currently deployed system will take actions with a higher estimated reward more often than it will take those with lower estimates. This skews the training data for future model iterations, which in turn leads to heteroscedasticity in the reward estimates. The most common frequentist approaches to reward modelling based on MLE – be it parameterised by simple linear models or deep neural networks – do not provide information about an estimated posterior distribution out-of-the-box. As a result, detecting pathological cases where gross over-estimation occurs is highly non-trivial. Well-chosen priors and the resulting MAP estimates can partially alleviate this, but are hard to validate and yield no guarantees.

As a simple example, consider a Beta-Bernoulli model with three actions [145, §7.2.1]. Rewards for action a_i are drawn as $r_i \sim \text{Bernoulli}(p_i)$, with $p_i \sim \text{Beta}(\alpha_0 + \alpha_i, \beta_0 + \beta_i)$. In this setup, α_i and β_i can be seen as the number of observed clicks and non-clicks for action a_i. For illustratory purposes, assume $\alpha_1 = \beta_1 = 1, \alpha_2 = 3, \beta_2 = 4, \alpha_3 = 33, \beta_3 = 60$. We assume a prior probability of receiving a click for the posterior predictive of 25%, so we set $\alpha_0 = 1, \beta_0 = 3$. Now, we can compute the ML and MAP estimates for these actions, and deduct the optimal policies. Figure 7.1

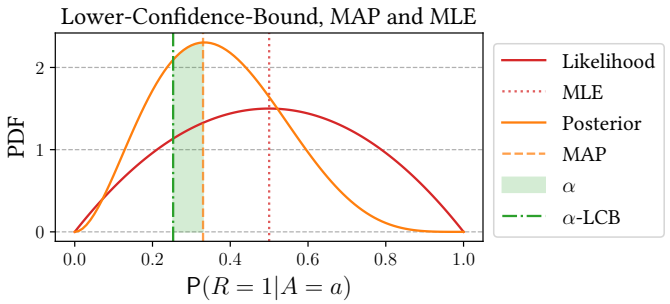


Figure 7.1: Illustration of the likelihood, prior and posterior estimates for the toy example in Section 7.3, showing different estimates for the reward distribution for action a_1.

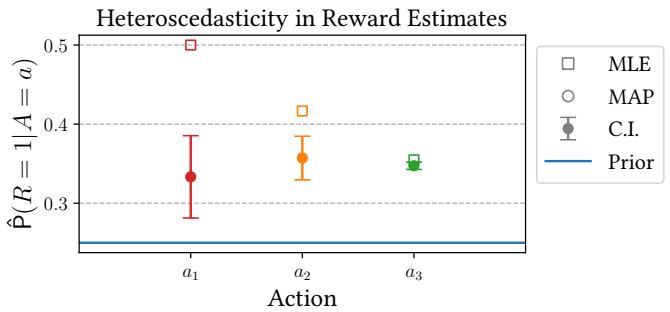


Figure 7.2: Illustration of the likelihood, prior and posterior estimates for the toy example in Section 7.3, showing different decision-making strategies for competing actions.

shows the resulting likelihood and posterior distributions for action a_1, Figure 7.2 shows that the MLE prefers action a_1, whereas the MAP estimate prefers a_2 (we will introduce the Lower-Confidence-Bound in Section 7.3). For well-explored context-action pairs, we see that the variance in the posterior predictive of the reward model is reduced, leading to a tighter credible interval. For under-explored context-action pairs, however, the error and variance grow to be quite substantial. The de facto decision making process of taking the action with the highest reward estimate, is then even more vulnerable to post-decision disappointment due to this type of heteroscedasticity, and thus prone to provide over-estimations of the *true* expected reward. There only needs to be a single action with a badly calibrated reward estimate for this situation to occur (due to the argmax in Eq. 7.4), and the probability of encountering such lesser explored actions will typically grow with the size of the action space (for realistic logging policies). If our posterior means are unbiased conditional on the value estimates, the results of Smith and Winkler show that the expected post-decision surprise will be zero. This is, however, an unreasonable assumption in complex real-world environments where correct model specification is often impossible, and the range of conjugate priors might not be expressive

enough to allow for this to happen. Furthermore, as we often deal with small effect sizes (i.e. $|p_i^* - p_j^*| < \epsilon$), even slight errors in the reward estimates can have significant impact on the actions taken by the resulting policy. If there is *some* probability of our recommendation policy taking a suboptimal action, the inequality bounding expected surprise even becomes strict: $\mathbb{E}[p_{i^*}^* - \widehat{p}_{i^*}] < 0$ [194].

The Optimiser's Curse can lead to a significant disparity between what we expect will happen based on the reward model, and what will actually happen when we act according to its estimates. Nevertheless, we can significantly improve recommendation performance by treating our reward estimates with a healthy dose of principled scepticism.

Pessimistic Decision-Making

Small effect sizes, bias and heteroscedasticity in the reward estimates are the main reasons why reward models typically perform more poorly than expected. The conceptually simplest way of mitigating this unevenly distributed variance is to mitigate selection bias altogether by adopting a uniformly random logging policy. However, showing recommendations to users independently of the estimated relevance of the action might not be in the best interest of the platform or the users, at least not from a business perspective. In what follows, we explore our decision-making options, borrowing ideas from the related on-policy bandit literature.

Traditional models generate point estimate predictions, which we can reasonably assume to be contained by the posterior shown in Figure 7.2. Possible actions are then ranked by $\widehat{r}(a_i)$, as these approaches cannot quantify differences between recommendations a_1, a_2 and a_3 in any other way. This is problematic due to all the reasons laid out above.

In an on-policy world, typical approaches make use of uncertainty estimates to balance the expected reward with the informational value of an action. Methods based on Thompson Sampling (TS) repeatedly sample reward estimates from an approximate posterior [23], and optimistic extensions to this paradigm are known to further improve performance [135]. Upper Confidence Bound (UCB) methods also follow the "optimism in the face of uncertainty" adage, explicitly taking the action with the highest posterior quantile instead of the MAP estimate [112]. For our example on Figure 7.2, this would lead to a ranking of $a_1 > a_2 > a_3$ (which coincides with ranking by the MLE). On-policy approaches are optimistic because it provably pays off; they get to observe the outcome of the chosen action and use this new data point to adjust reward estimates. Intuitively, this makes that reward estimates will never be overly optimistic for long, as the posterior will tend to converge to the lower, true p_i^* as more data comes in. This "self-correcting" property then naturally bounds metrics like regret in an online setting, and makes TS and UCB provably efficient. In an off-policy setting, we do not have the luxury to instantly learn from the outcome of our actions. All we have is a finite log of context-action-reward triplets, collected by a different policy, with which we will have to make do. It is clear that optimism will not help us in such cases, as we cannot reap the fruit of informational value that comes with it.

Optimism is not the way to go – but the naive decision-making procedure that purely focuses on the maximal reward estimates, is still likely to yield exactly those that were over-estimated. Even without explicitly encoding optimism, this still leads

to inflated expectations and subpar performance. We can offset this unwarranted over-estimation by treating our model predictions pessimistically. This is exactly what Smith and Winkler suggest when saying: "model the uncertainty in the value estimates explicitly and use Bayesian methods to interpret these value estimates" [194]. With a suitable prior distribution and unbiased posterior means, their suggested approach effectively encourages principled conservatism which provably limits disappointment. We have argued how their proposed solution breaks down in complex environments, and additionally note that advanced prior distributions tend to complicate the reward modelling procedure and can hurt scalability by surrendering conjugacy. Ranking the actions in Figure 7.2 according to their posterior means leads to a ranking of: $a_2 > a_3 > a_1$. Because of the vastly reduced variance from a_2 to a_3 and the small difference in their posterior means, we argue that a_3 should be the safe choice. One might argue that the MAP choosing a_2 is merely the result of an inappropriate prior, but small effect sizes combined with heteroscedasticity make this highly non-trivial to tune and validate properly. Optimising the prior as a hyper-parameter to achieve exactly zero post-decision disappointment is theoretically possible in controlled environments when Bayesian methods are used, but this is highly complex and intractable in real-world environments where we have approximate uncertainty estimates for general model classes like neural networks. Furthermore, as a simple bias term directly influences post-decision surprise without altering the actions that are being taken, it is clear that maximising the online performance of the deployed recommendation policy should still be the overarching objective compared to blindly limiting disappointment.

Instead of pursuing unbiasedness through appropriate priors, we propose to be even more sceptical of our own reward model, and to make decisions based on the maximal lower quantile of the posterior distribution. By adopting a Lower Confidence Bound (LCB)-driven decision-making strategy, we effectively penalise actions with high variance and pick the action with the best worst-case outcome. This is visualised as the $\alpha-$LCB in Figure 7.1. Following our toy example from Figure 7.2, this inverts the UCB and flips the MAP ranking to obtain $a_3 > a_2 > a_1$. Reward predictions based on posterior lower bounds are designed to be conservative and thus strictly lower than the estimated posterior means: $\widehat{p}_{i^*} > \widehat{p_{i^*}^{\text{LCB}}}$. As a consequence, it naturally follows that the post-decision disappointment from acting on these maximal lower bound predictions (actions j^*) will be strictly lower than if we had picked them according to their posterior mean predictions: $\mathbb{E}[p_{j^*}^* - \widehat{p}_{j^*}] < \mathbb{E}[p_{j^*}^* - \widehat{p_{j^*}^{\text{LCB}}}]$. Note that this result is quite loose and holds for any $\widehat{p_{j^*}^{\text{LCB}}} < \widehat{p}_{j^*}$; the posterior lower bounds still need to be constructed sensibly to improve the online performance of the resulting policy. As backed up by empirical observations from a wide range of experiments, our proposed pessimistic decision-making strategy leads to a significant and robust increase in recommendation performance. Naturally, the potential performance gains will be highest in those settings where traditional reward models fail: limited training sample sizes in large action spaces collected under highly skewed logging policies, as is often the case in real-world systems.

Pessimism in Policy Learning. The idea of scepticism, conservatism or pessimism is not novel in itself and lies at the heart of many advances in policy-based methods for off-policy learning as well, albeit often implicitly. One of the most widely used extensions to IPS weighting is that of capping the weights to a certain max-

imum value m [69, 49]. In doing so, we effectively choose to be sceptical about our reweighted rewards when things are too good to be true, replacing the probability ratio in Equation 7.1 with $\min\left(m, \frac{\pi(a|c)}{\widehat{\pi}_0(a|c)}\right)$. Capped IPS is known to improve the accuracy of the estimator and the performance of the resulting learnt policy, even when the logging propensities are known and exact. The use of such techniques is often justified by claiming an improved bias-variance trade-off, but the connections to over-estimation in reward models deserve mentioning. The same parallels can be drawn for several other policy learning tricks such as variance regularisation [134, 210], imitation learning [129], distributional robustness [44, 192] and estimator lower bounds [122, 84]. Several concurrent recent works provide a deeper understanding of the value of pessimism in more general offline RL scenarios, be it in policy- or Q-learning-based methods [94, 245, 120, 100]. We point the interested reader towards the work of Jin et al. for more theoretical underpinnings [86].

Closed-Form Lower-Confidence-Bounds with Bayesian Ridge Regression

By looking at the problem of learning an optimal recommendation policy through the lens of the "Direct Method", we effectively cast it as a classification or regression problem. As a consequence, the parameterisation of $\widehat{r}$ can take many forms. The pessimistic LCB method we propose in this work is generally applicable and not bound to any specific model class, with the exception that it relies on uncertainty estimates to generate sensible bounds. In what follows, we show how to obtain closed-form expressions for both the posterior mean and variance when a ridge regressor models the reward. The interpretability and efficiency of linear models makes them an attractive and common choice for practitioners that need to decide on a reward model [137, 59, 136, 146, 84]. An ongoing line of research in traditional approaches to recommendation has repeatedly shown the effectiveness of linear models in collaborative filtering tasks as well [147, 187, 200, 85, 26]. Other recent work reports empirical advantages of squared loss over cross-entropy loss [67], which could explain the effectiveness of item-based least-squares models like SLIM [147] and EASE^{R} [200], even when labels are binary. The model we propose here can be interpreted as a pessimistic, off-policy, bandit variant of the latter.

In line with the item-based paradigm, we model users based on their historical organic interactions with other items in the catalogue: $\boldsymbol{c} \in \mathbb{R}^{|\mathscr{A}|}$; additionally normalising samples according to their respective ℓ_1-norms to deal with varying-length user histories. Recommendations are represented as one-hot encoded vectors: $\boldsymbol{a} \in \{0,1\}^{|\mathscr{A}|}$. Action- and context-features are mapped to a joint space via a Kronecker product: $\boldsymbol{x} = \Phi(\boldsymbol{c}, \boldsymbol{a}) = \boldsymbol{c} \otimes \boldsymbol{a}$. When we denote the model parameters by $\boldsymbol{\theta} \in \mathbb{R}^{|\mathscr{A}|^2}$, a linear model estimates the reward as shown in Equation 7.5 (omitting a bias-term for brevity).

$$\hat{\mathrm{P}}(R = 1 | C = c, A = a, \boldsymbol{\theta}) = \boldsymbol{x}^\intercal \boldsymbol{\theta} = \boldsymbol{c}^\intercal \boldsymbol{\theta}_{|a} \tag{7.5}$$

Here, $\boldsymbol{\theta}_{|a}$ holds the parameters that are relevant for action a: the $|\mathscr{A}|$ parameters ending at index $i \cdot |\mathscr{A}|$ for actions $i \in \{1, \ldots, |\mathscr{A}|\}$. The final equation holds because we use a one-hot encoding for actions and a Kronecker product to link context and action features. This implementation trick makes computations significantly less expensive, as we now deal with vectors of size $|\mathscr{A}|$ instead of its square. If we define

$\boldsymbol{X} \in \mathbb{R}^{|\mathscr{D}| \times |\mathscr{A}|^2}$ as the design matrix holding joint context-action features for every sample in the training set $\mathscr{D}$, $\boldsymbol{y}$ as the vector of rewards to be predicted, and Θ the parameter space, we can formally define our optimisation problem as follows:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\arg\min} \left(\left\| \boldsymbol{X}^\intercal \boldsymbol{\theta} - \boldsymbol{y} \right\|_2^2 + \lambda \left\| \boldsymbol{\theta} \right\|_2^2 \right). \tag{7.6}$$

The Tikhonov-regularisation in Equation 7.6 is key to the Bayesian interpretation of this ridge regression problem. Indeed, it is known that this formulation is equivalent to imposing independent Gaussian priors with constant variance on the parameters, as well as on the errors in the rewards [145]:

$$\boldsymbol{\theta} \sim \mathscr{N}(0, \sigma_x^2), \qquad \boldsymbol{y} \sim \mathscr{N}(\boldsymbol{x}^\intercal \boldsymbol{\theta}, \sigma_y^2). \tag{7.7}$$

When $\lambda = \frac{\sigma_y^2}{\sigma_x^2}$, the solution to the ridge regression problem in Equation 7.6 is equivalent to the MAP estimate for $\boldsymbol{\theta}$. A key advantage to the efficiency of this procedure is that both the posterior mean and covariance can be computed with the analytical formulas presented in Equation 7.8:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}, \qquad \hat{\boldsymbol{\Sigma}} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1}. \tag{7.8}$$

The main bottleneck here is the inversion of the $|\mathscr{A}|^2 \times |\mathscr{A}|^2$ Gramian matrix, which can quickly grow to be cumbersome for larger action spaces. In similar spirit to the implementation trick in Equation 7.5, we can decompose this inversion into one per target action. This leads to $|\mathscr{A}|$ matrix inversions of size $|\mathscr{A}| \times |\mathscr{A}|$, and is possible because of the sparse, block-diagonal structure we acquire from the Kronecker product combined with one-hot encoded action vectors. We now end up with a model that is similar to the disjoint linear models used in the LinUCB procedure for on-policy bandit applications [112], although our prior variance ratio λ can be tuned whereas theirs is fixed (reducing to the MLE when $\lambda = 0$). Also in contrast with their approach, we will use the posterior mean and covariance to obtain a lower confidence bound reward estimate for a given context-action pair:

$$\widehat{\mathsf{P}}_{\text{LCB}}(R = 1 | C = c, A = a) = \boldsymbol{x}^\intercal \hat{\boldsymbol{\theta}} - \alpha \sqrt{\boldsymbol{x}^\intercal \hat{\boldsymbol{\Sigma}} \boldsymbol{x}} = \boldsymbol{c}^\intercal \hat{\boldsymbol{\theta}}_{|a} - \alpha \sqrt{\boldsymbol{c}^\intercal \hat{\boldsymbol{\Sigma}}_{|a} \boldsymbol{c}}. \tag{7.9}$$

Let $\hat{\boldsymbol{\Sigma}}_{|a}$ denote the sub-matrix of $\hat{\boldsymbol{\Sigma}}$ that is relevant to action a. That is, the $|\mathscr{A}|$ rows and columns ending at index $i \cdot |\mathscr{A}|$ for actions $i \in \{1, \ldots, |\mathscr{A}|\}$. From this formulation, it is clear that values in $\hat{\boldsymbol{\Sigma}}$ off of this block-diagonal will never be used, and thus never need to be computed. The hyperparameter α is related to the coverage of the approximate posterior induced by $\widehat{\mathsf{P}}_{\text{LCB}}$ [229]. Note that this hyperparameter is not specific to the ridge regression parameterisation, and will also occur when a nonlinear neural network models the reward and uncertainty estimates are obtained from the approximation techniques described in Section 7.2. Replacing the reward model in the direct method with this pessimistic alternative for the estimator based on the posterior mean ($\widehat{\mathsf{P}}_{\text{LCB}}$ vs $\hat{\mathsf{P}}$), yields the optimal deterministic LCB policy:

$$\pi^*_{\text{LCB}}(a|c) = \begin{cases} 1 & \text{if } a = \underset{a' \in \mathscr{A}}{\arg\max}\, \hat{r}_{\text{LCB}}(a', c), \\ 0 & \text{otherwise.} \end{cases} \tag{7.10}$$

7.4 Experimental Results

A key component of recommender systems is their interactive nature: evaluating recommendation policies on offline datasets is not a straightforward task, and conclusions drawn from offline results often contrast with the online metrics that we care about [48, 177, 75], which motivates casting recommendation as a bandit learning problem [82]. Recent work either shows empirical success with supervised-to-bandit conversions on organic user-item datasets [128], through live experiments [136, 25, 140], or by adopting open-source simulation environments [180, 84]. To aid in the reproducibility of our work, we make use of the RecoGym simulation environment [176]. RecoGym provides functionality to simulate organic user-item interactions (e.g. users viewing products on a retail website), as well as bandit interactions under a given logging policy (users clicking on shown recommendations). Publicly available datasets that contain both types of data (observational *and* experimental) are scarce, and still insufficient for reliable counterfactual evaluation. A considerable advantage of RecoGym is the opportunity to simulate online experiments such as A/B-tests, that can then be used to reliably estimate the online performance of an intervention policy. We refer the interested reader to the source code of the simulator[3] or the reproducibility appendix of [84] for an overview of the inner workings of the simulation environment. The source code to reproduce our experiments is publicly available at github.com/olivierjeunen/pessimism-recsys-2021. The research questions we wish to answer are the following:

RQ1 Can we find empirical evidence of the Optimiser's Curse in off-policy recommendation environments?

RQ2 Can our proposed LCB decision-making strategy effectively limit post-decision disappointment?

RQ3 Can we increase online performance with a recommendation policy using a reward model with LCB predictions?

RQ4 How are these methods influenced by the amount of randomisation in the logging policy?

RQ5 How are these methods influenced by the number of training samples and the size of the action space?

Logging Policies An important factor to take into account when learning from bandit feedback is the logging policy that was deployed at the time of data collection. Deterministic policies make bandit learning near impossible, whereas a uniformly random logging policy generates unbiased data, but is an idealised case in practice. Realistic logging policies will aim to show recommendations that they perceive to be relevant, whilst allowing other actions to be taken in an explorative manner. We adopt a simple but effective personalised popularity policy based on the organic user-item interactions that have preceded the impression opportunity. For a context c consisting of historical counts of organic interactions with items (as laid out

[3] github.com/criteo-research/reco-gym

in the parameterisation in Section 7.3), the logging policy π_{pop} samples actions proportionately to their organic occurrences. This policy is deficient, as it does not assign a non-zero probability mass to every possible action in every possible context [178]. Deficient logging policies violate the assumptions made by IPS to yield an unbiased reward estimate [156], which poses a significant hurdle for policy-based methods. Nevertheless, they are realistic to consider in real-world off-policy recommendation scenarios. This extreme form of selection bias impedes effective reward modelling as well, as we will show in the following section. Indeed, when a context-action pair has zero probability of occurring in the training sample, we *need* to resort to appropriate priors or conservative decision making. The deficiency of π_{pop} can be mitigated easily by adopting an ϵ-greedy exploration mechanism, where we resort to the uniform policy with probability $\epsilon \in [0,1]$. Naturally, this implies both π_{pop} and π_{uni} when ϵ is respectively 0 or 1. For arbitrarily small values of ϵ, π_0 is no longer deficient in theory, but extremely unlikely to explore the full context-action space within finite samples.

$$\pi_0(a|c) = \begin{cases} \pi_{\text{pop}}(a|c) & \text{with probability } 1-\epsilon, \\ \pi_{\text{uni}}(a|c) & \text{otherwise,} \end{cases}$$
$$\text{where } \pi_{\text{pop}}(a_i|c) = \frac{\boldsymbol{c}_i}{\sum_{j=1}^{|\mathscr{A}|} \boldsymbol{c}_j}, \text{ and } \pi_{\text{uni}}(a|c) = \frac{1}{|\mathscr{A}|}. \tag{7.11}$$

We vary $\epsilon \in \{0, 10^{-6}, 10^{-4}, 10^{-2}, 1\}$ in our experimental setup. Note that this type of logging policy is equivalent to the ones used in previous works [146, 81, 84, 77, 180], but that we explore a wider range of logging policy randomisation to highlight the effects on naïve reward modelling procedures.

Optimiser's Curse (RQ1-3)

To validate whether the theoretical concept of the Optimiser's Curse actually occurs when reward models are learned in off-policy recommendation settings, we adopt the following procedure: 1. Generate a dataset containing organic and bandit feedback, 2. train a reward model as described in Section 7.3 – optimising the regularisation strength λ to minimise Mean Squared Error (MSE) on a validation set of 20%, 3. simulate an A/B-test and log the difference between the reward estimates $\widehat{p}_{i^*}$ and the true reward probability $p^*_{i^*}$ for the actions selected by the competing decision strategies. We then vary the logging policy in (1), and repeat this process 5 times to ensure statistically robust and significant results. Every generated training set and every simulated A/B-test consists of 10 000 users, leading to approximately 800 000 bandit opportunities in the training set as well as 800 000 online impressions per evaluated policy. We report the average empirical disappointment ($\widehat{p_{i^*}} - p^*_{i^*}$) for both the standard decision-making strategy of taking the MAP action ($\alpha = 0$), and our pessimistic lower-confidence-bound strategy, varying the lower posterior quantile α. Note that the hyperparameter α plays an important role here, and that it can always be increased to achieve zero post-decision disappointment (and even lower). While this sets more realistic expectations for the performance of the reward model, this does not guarantee an improvement in the online metrics we care about. For this, we additionally report an estimate of the policy's attained click-through-rate

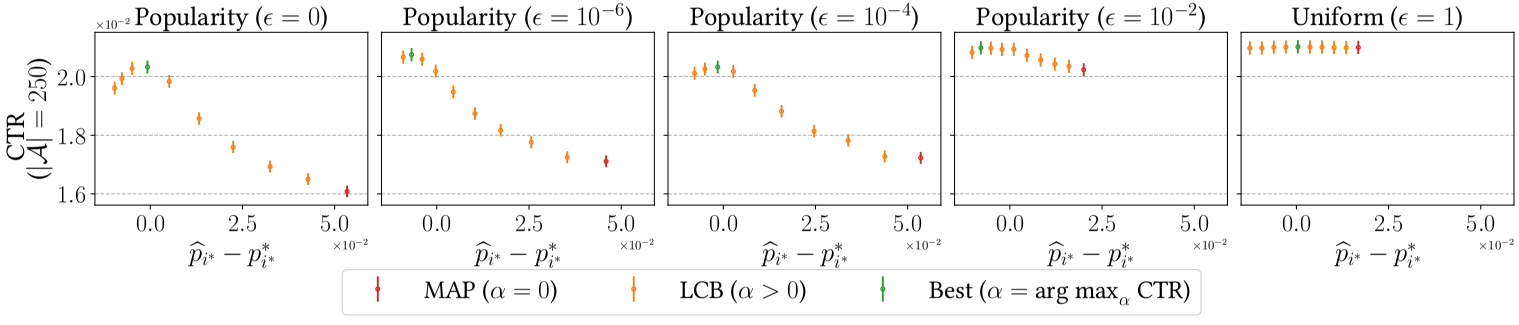


Figure 7.3: We evaluate varying degrees of pessimistic decision strategies ($\alpha \in \{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45\}$, corresponding with right- to leftmost measurements in the plots). The x-axis shows the resulting post-decision disappointment with the attained CTR on the y-axis (95% credible intervals). Every plot corresponds to a different amount of randomisation in the logging policy (increasing from left to right). We observe that LCB is effective in minimising post-decision disappointment, and that this is highly correlated with increasing online performance, most notably when the amount of logging randomisation is limited.

(CTR) in the A/B-test. Also note that this type of experimental procedure would not be feasible without the use of a simulation environment, as we usually don't have access to the true reward probability p_i^*. In such cases, we would need to resort to empirical averages based on the observed reward. Figure 7.3 visualises the results from these experiments.

Empirical Observations. First, we see clear empirical evidence of the Optimiser's Curse in action: when acting based on just the posterior mean, we encounter post-decision disappointment regardless of the logging policy. As our trained reward models are even slightly under-calibrated w.r.t. the empirical training sample (i.e. negative mean error), this result can seem counter-intuitive and is not straightforward to mitigate with a bias term tuned on offline data. Second, we observe that pessimistic decision-making based on predictive uncertainty consistently decreases disappointment, and that it can significantly increase the policy's attained CTR in A/B-tests. The optimal value of α with respect to online performance also brings the *absolute* surprise closer to zero, indicating that these values are closely related. α's interpretation relating to the coverage of the approximate posterior of $\hat{r}$ helps when tuning it [229]. Naturally, when the variance on the reward estimates is homoscedastic w.r.t. the actions, LCB does not affect the ordering of the reward estimates or the resulting policy. This explains why online performance is not significantly impacted when the logging policy is uniform, while post-decision disappointment can consistently be alleviated.

Performance Comparison (RQ3-5)

To further assess when our proposed pessimistic decision-making procedure can lead to an offline learnt policy with improved online performance, we train models on a range of datasets generated under different environmental conditions and report results from several simulated A/B-tests. The resulting CTR estimates with their 95% credible intervals are shown in Figure 7.4. Every row corresponds to a differently sized action space ($|\mathscr{A}| \in 10, 25, 50, 100, 250$), every column shows results for a different amount of randomisation in the logging policy. The amount of available training data for the reward model increases over the x-axis for every plot. We report CTR estimates for policies that act according to reward models based on ML or MAP estimates, and those that use lower confidence bounds with a tuned α. Additionally, we show the CTR attained by the logging policy π_0, and an unattainable skyline policy π^* that acts based on the true reward probabilities p^*. Every measurement shown in Figure 7.4 shows a 95% credible interval over 5 runs with 10 000 evaluation users, totalling 1 000 simulated A/B-tests with five competing policies each, or more than three billion impressions summed up. As our reward models are agnostic to the logging propensities, we do not include policy-based approaches that would require them (either purely based on IPS [19], hybrid [84] or doubly robust [36]). We do note that our results are directly comparable to those presented in [146, 84], and both our novel LCB method and MAP baseline show significant improvements over all their policy- and value-based competitors.

Empirical Observations. In line with our observations from Figure 7.3, we see that LCB decision-making yields a robust and significant improvement over naively acting on ML or MAP estimates. This result is consistent over varying training sample sizes, action spaces and logging policies, but most outspoken in cases where the

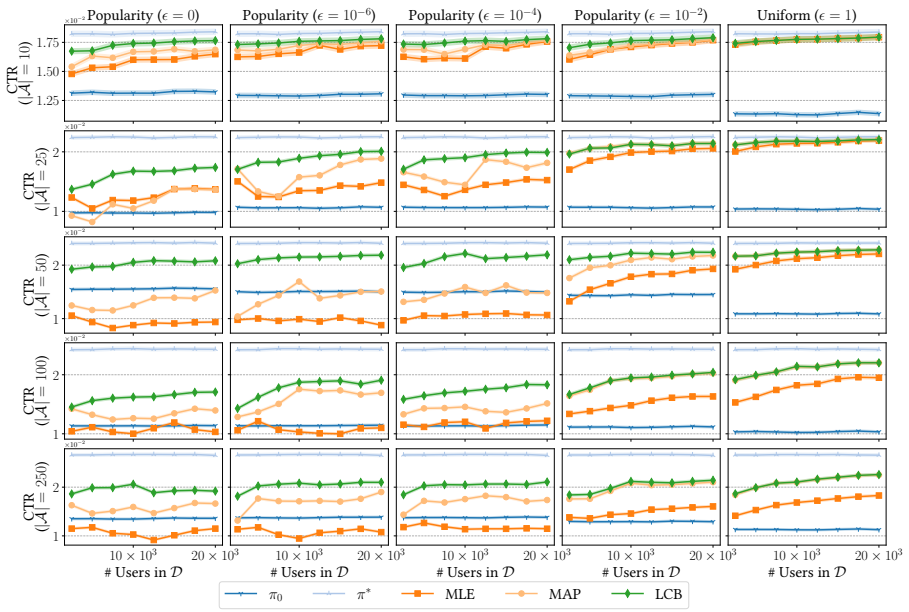


Figure 7.4: Experimental results for a range of simulated A/B-tests. The amount of training data is increased over the x-axis, the attained CTR is shown on the y-axis (shaded 95% credible interval). Every column corresponds to a different amount of randomisation in the logging policy (increasing from left to right); every row corresponds to a differently sized action space (increasing from top to bottom). We observe a significant increase in CTR for the pessimistic model, most apparent for smaller training samples, larger action spaces, and limited randomisation. The CTR improvements for LCB over MAP average to 16% over all measurements, and range up to 95%.

amount of randomisation and the number of available training samples are limited, and the action space is larger. As explicit randomisation and data collection can be expensive in practice, the environments where LCB excels are the ones that are most commonly encountered in real-world systems. Additionally, we observe more consistent and robust behaviour for policies that use LCB decisions compared to those that do not. This decreased variance in online performance can also be attributed to pessimistic decision making: because we no longer take our chances with high-uncertainty predictions, we fall back to more robust alternatives. We know what the reward model does not know, and this gained knowledge significantly benefits the interpretation of reward predictions, and the resulting decisions.

Limitations of our study design. Off-policy approaches for learning from bandit feedback are typically evaluated in set-ups where the size of the action space is a few dozen at most [210, 88, 206]. As a result, methods for counterfactual learning in recommendation are often evaluated in modestly sized action spaces too [84, 180, 146]. Therefore, the reported results are most relevant to personalisation use-cases where the number of alternatives is limited, such as personalising tiles or rows on a home-

page, recommending news articles from a set of recently published ones, or predicting clicks within a slate. The size of the item catalogue in general purpose recommendation scenarios can be in the hundreds of thousands, warranting further research into off-policy recommendation for very large action spaces [123]. In such environments, learning continuous item embeddings as opposed to the discrete representation we have adopted can provide a way forward. Moreover, the lack of publicly available datasets for the off-policy recommendation task can be prohibitive for reproducible empirical validation of newly proposed methods. The few alternatives that do exist [106, 179], still deal with comparatively small action spaces and need to resort to counterfactual evaluation procedures with high variance and limited statistical power (compared to simulated online experiments). Furthermore, a single dataset would be comparable to a single measurement in Figure 7.4, limiting the range of environmental parameters we can change to observe effects on the online performance for competing methods. Because of these reasons, we believe the RecoGym environment to be an appropriate choice for the experimental validation of our methods [176].

7.5 Conclusions and Future Work

In this work, we have advocated in favour of *pessimistic* reward modelling in bandit feedback settings. We have proposed a general framework for sceptic decision-making in off-policy recommendation use-cases, and have shown how to translate uncertainty estimates for ridge regressors into a conservative decision rule. Our proposed method lifts the Optimiser's Curse whilst achieving a significant and robust boost in recommendation performance for a variety of settings. In future work, we wish to investigate whether pessimistic reward predictions can lead to improved doubly robust learning [77], to study the generalisability of our results to larger action spaces, and to investigate the effects of scepticism on the informational value of data collected under such a policy.

Reflections

This work largely focuses on offline learning instead of offline evaluation. Nevertheless, as we have mentioned before in this thesis, one of the main motivations for the adoption of the bandit feedback paradigm is that *learning* and *evaluation* become two sides of the same coin. The insights with respect to the Optimiser's Curse presented throughout this Chapter suggest that offline learning is most effective when the offline estimator is free from over-estimated rewards, a connection that is worth studying more in-depth.

Similar to Chapter 6, it would be interesting to extend the current work to *slate* instead of *single-item* recommendations, and investigate whether the update mechanisms proposed in Chapters 2 and 3 would be applicable to the closed-form ridge regression solution that we leverage in this Chapter.

CHAPTER 8

Conclusions

This thesis has dealt with the broad field of recommender systems in its various flavours. We have presented several novel research contributions to the field, with seemingly very different focuses. This is undeniably true: whereas the first Chapters in this thesis focus on scalability and efficiency, the middle two shine a light on offline evaluation procedures and the final two cast the recommendation task as a bandit learning problem. At first sight, these parts may seem disconnected. Nevertheless, the overarching research questions that motivate and unite this thesis should now be clear:

Can we shift the offline paradigm towards methods that directly target success in online environments?

How do we ensure that academic research contributions translate to improved recommendations in practical settings?

Certainly, it can be argued that: 1. the importance of scalability, efficiency and models that are up-to-date is most palpable in real-world applications, 2. offline evaluation procedures are only effective when they provide reasonable estimates of online performance, and 3. the off-policy bandit setting is the predominant realisation of recommendation systems encountered by practitioners. We believe that it is of crucial importance for the field moving forward to keep an eye on the environments that our methods need to perform and thrive in, and to work towards the goals that are dictated by these environments. Only by doing this, can we ensure that our research progress will have *impact* in the real-world. In this concluding Chapter, we summarise our contributions and propose a scope for future research.

8.1 Main Contributions

- In **Chapter 2**, we have proposed the *Dynamic Index* algorithm for efficiently computing similarity between sparse, high-dimensional vectors. As the main computational cost for item-based nearest neighbours algorithms in collaborative filtering is exactly that, we applied our method to this use-case. Dynamic Index is an incremental algorithm by design, making it suitable for real-time recommendation scenarios. Additionally, we presented a MapReduce-inspired parallellisation procedure to efficiently distribute computations. Furthermore, we introduced the concept of item *recommendability*, and have shown how it can be exploited to further improve the computational efficiency of our method.

- In **Chapter 3**, we have introduced the *Dynamic* EASE^{R} method for efficiently updating an existing item-based ridge regression model for collaborative filtering. By exploiting parts of the Dynamic Index algorithm and subsequently applying the Woodbury matrix identity, we can incrementally update such models when new data arrives, instead of recomputing the entire model on the updated dataset. We showed how exact DYN-EASE^{R} significantly improves the efficiency with which EASE^{R}-like models can be kept up-to-date, and additionally proposed an approximate variant of the algorithm that improved the efficiency even further without a significant decrease in recommendation accuracy.

- In **Chapter 4**, we have presented *SW-EVAL*, a novel offline evaluation method for implicit-feedback recommender systems. SW-EVAL adheres much more tightly to the dynamic environments that deployed recommender systems need to perform in than traditional alternatives, and generates evaluation results that are more reliable with respect to online performance. Additionally, we presented an empirical study on the impact of live recommendation algorithms during data collection time on the collected data – and found that there can be a significant bias from the logging policy on this data. This Missing-Not-At-Random (MNAR) bias impedes effective offline evaluation, and it is paramount to take this into account. To this end, we proposed a scope for future research towards *fair* offline evaluation procedures in such settings.

- In **Chapter 5**, we have presented an overview of how recommendation algorithms are typically evaluated, contrasting the dynamic online environments that occur in practice with the typical static offline evaluation setups used in the literature. We have highlighted key differences between these paradigms, and proposed ways to improve. Specifically, we have presented a research agenda that focuses on temporal information, off-policy evaluation that models MNAR biases, and have advocated in favour of using information regarding to impressions and user inaction to facilitate the distinction between missing and negative feedback.

- In **Chapter 6**, we have presented an empirical study of counterfactual learning methods for recommendation. These methods make use of *bandit* feedback – logged recommendations and their outcomes – in contrast to the traditional

organic user-item interactions that are used in the majority of recommender systems research. We presented an overview of various existing algorithms for learning from bandit feedback, and showed how they can be applied to the off-policy recommendation task. We highlighted where existing methods tend to fail, and proposed a logarithmic lower bound on the traditional importance sampling estimator to account for stochastic rewards and low treatment effects combined with finite samples. Finally, we introduced a joint policy-value learning objective called the *Dual Bandit*, and showed how it can be used to obtain superior performance compared to stand-alone policy or value-based methods.

- In **Chapter 7**, we have shone a light on reward models learnt from bandit feedback in off-policy recommendation scenarios. We introduced the Optimiser's Curse – a general phenomenon that occurs in decision-making scenarios where only the highest estimated alternatives are considered. We have shown how it relates to the recommendation problem, and how it typically incurs post-decision disappointment on behalf of the recommendation algorithm. Furthermore, the traditional Bayesian solution of taking actions according to the mean of the posterior predictive distribution is insufficient in real-world scenarios where checking for unbiasedness is highly non-trivial. To this end, we have proposed a general-purpose conservative decision-making framework that takes actions based on a lower quantile of the posterior predictive, corresponding to a form of principled scepticism in decision making. We have shown how our uncertainty-aware reward modelling framework both lifts the Optimiser's Curse and allows for a significant boost in online recommendation accuracy.

8.2 Supplementary Contributions

The Chapters in this thesis and the contributions laid out above correspond to peer-reviewed manuscripts we published over the course of the past four years. Several additional research activities led to other peer-reviewed contributions to workshops and conferences in the form of research papers, demonstrations and tutorials. These contributions often (but not always) consist of supporting work in the context of the chapters of this thesis. In what follows, we briefly summarise them.

In Jeunen and Goethals [76], we propose a session-based recurrent neural network architecture to predict how users will interact with items in a session. We empirically validated our approach in the context of the 2019 ACM WSDM Cup on a dataset provided by Spotify, where the task was to predict whether a user would skip a song or listen to it in its entirety. Our proposed method achieved 5^{th} place.

In Moens et al. [142], we propose an episode mining approach to analysing recommender system data. We demonstrate how to use SNIPER, a tool for interactive pattern mining, to analyse and understand the behaviour of recommender systems for explorative offline evaluation purposes.

In Jeunen et al. [82], we explore the value of bandit feedback for the offline evaluation of recommender systems. We show how inverse propensity scoring estimators can provide reliable offline estimates of online performance metrics – in contrast with traditional IR-inspired metrics based on observational and organic

user-item interactions. This work follows our review of offline evaluation techniques in Chapters 4 and 5, and motivates the bandit view adopted in Chapters 6 and 7.

In Mykhaylov et al. [146], we present three methods for training on bandit feedback from the literature, summarising their histories and assumptions. We review the literature around this fundamental yet often overlooked choice in the literature.

In Jeunen et al. [81], we present an overview of the state-of-the-art in learning from bandit feedback – specifically focused on the task of off-policy learning in recommendation environments. This work provided the basis for the literature overview in Chapter 6.

In Vasile et al. [218], we cast the recommendation problem as counterfactual policy learning. We give a structured overview of the conceptual frameworks behind current state-of-the-art recommender systems, explain their underlying assumptions, resulting methods and their shortcomings. We go over the theoretical guarantees that counterfactual policy learning gives, and how it can address the shortcomings of the previous frameworks.

In Jeunen et al. [85], we study the applicability of item-based ridge regression models to implicit-feedback collaborative filtering when additional side-information or metadata about items is available. Two complementary extensions to the EASE^{R} paradigm are proposed, based on collective and additive models, and we show how they naturally retain analytically computable solutions.

In Jeunen and Goethals [77], we provide an empirical evaluation of doubly robust learning methods in the off-policy recommendation scenario. In line with previous work, our results highlight that the stochasticity of the logging policy is the main factor deciding between the superiority of value- or policy-based methods. In contrast with previous work, our results indicate that recommendation policies learned via standard doubly robust estimation can often be outperformed by either their standalone value- or policy-based component. We present a scope for future research to improve doubly robust estimation methods when the reward model performs poorly. This work motivated the uncertainty-aware decision-making framework presented in Chapter 7.

In Vasile et al. [219], we use the rich language of decision theory to present policy- and value-based methods to recommendation in a common framework. We offer side-by-side comparisons between these methods outlining their strengths and weaknesses, such as estimator variance, model misspecification, tractability and ease-of-use. By identifying the modes of failure for every class, we provide practical guidelines for future practitioners as to which method to apply in which types of environments. This tutorial provides a novel and unifying view on policy- and value-based methods for recommendation.

In Jeunen and Goethals [79], we study how the top-K contextual bandit problem relates to issues of disparate exposure, and how this disparity can be minimised. We propose a personalised exposure-aware arm selection algorithm that handles this relevance-fairness trade-off on a user-level. Our model-agnostic algorithm deals with arm selection instead of utility modelling, and can therefore be implemented on top of any existing bandit system with minimal changes. We conclude with a case study on carousel personalisation in music recommendation: empirical observations highlight the effectiveness of our proposed method and show that exposure disparity can be significantly reduced with a negligible impact on user utility.

8.3 Outlook and Future Work

Reproducibility in Bandit Learning for Recommendation

Reproducibility is the cornerstone of the scientific method. This especially rings true in a field that relies heavily on empirical validation for newly proposed methods. One of the main reasons why the field has predominantly focused on recommendation from organic user-item interactions, is that publicly available datasets facilitating empirical validation in this setting are plentiful. When we wish to move to the off-policy bandit learning paradigm, a need arises for large-scale publicly available datasets containing logs of recommendations and their outcomes. This alone might not be sufficient – as it allows for learning models to be optimised, but still forces us to resort to off-policy evaluation procedures for their validation. While these can be a significant step forward compared to metrics based on organic interactions [82], they come with limited statistical power due to their typical high variance [49, 106]. Furthermore – many publicly available datasets with bandit feedback do not include any information regarding the logging policy (e.g. [155]), inhibiting the use of counterfactual estimators entirely. Recent work has introduced the "Open Bandit Pipeline", alleviating some of these issues [179]. Further work on releasing publicly available real-world datasets that support effective counterfactual evaluation (e.g. via a heavily randomised logging policy) could significantly further the field.

Alternatively, validating empirical results that are obtained through simulation frameworks by correlating them with results obtained on real-world data could help better understand how generalisable observations stemming from such frameworks really are [176, 68]. This gap between simulated and real-world environments is a well-known problem in the reinforcement learning community, often referred to as the "Reality Gap" [247].

Jointly Leveraging Organic and Bandit Feedback

Our work so far has treated bandit and organic feedback in very different ways. In both Chapters 6 and 7, we have used organic user-item interactions to describe the user's history for contexts in which actions were taken by the logging policy. This decoupling is naive, as information about organic co-occurrences between items could very well indicate a positive correlation in the bandit space too. Recent work exploits both types of data to learn improved value-based recommendation models, either in a Bayesian latent factor model [180] or a self-supervised manner [236]. In future work, we wish to investigate how such mechanisms can be adapted for policy learning scenarios. This is especially useful when the amount of bandit interactions to learn from are limited, or to transfer the organic signal between bandit arms in cold-start situations [121].

Exploiting Natural Variations versus Forced Exploration

The core assumption behind inverse propensity score weighting is that all recommendations have a non-zero probability of being shown in any given context – a stringent assumption that has hindered the widespread adoption of such techniques in real-world environments. The work of Sachdeva et al. challenges this assumption,

leading to impactful insights on *why* and *when* randomisation is necessary [178]. Other related work proposes to use "intervention harvesting" to quantify position biases in search engines [6, 42]. The rationale behind this technique is to exploit natural variations in the data instead of relying on explicit randomisation. Further related work has shown how experimental groups from online A/B tests can be used as instrumental variables when estimating a causal graph [165]. Exploring how intervention harvesting could be used to alleviate selection bias in off-policy recommendation – both for learning and effective offline evaluation – is another exciting area for future research.

Mehrotra et al. study user responses to explicitly randomised recommendations – finding that these responses differ significantly among user groups and that they can be predicted [139]. This paves the way to smart, targeted and personalised randomisation, in contrast with the alternative of uniform randomisation.

Multi-Objective Optimisation for Offline Bandits

Recommender systems are seldom evaluated using a single metric, especially when deployed on digital platforms that need to service multiple stakeholders [138]. Therefore, recommendation algorithms often need to be optimised for multiple objectives at the same time, some of which might be conflicting. Several multi-objective optimisation approaches have been proposed for traditional organic [174], learning-to-rank in e-commerce [118] and online music recommendation [140] settings, but none that are specifically tailored towards the off-policy setting. The uncertainty-aware decision making strategy we proposed in Chapter 7 might prove especially applicable in such settings. This, too, provides an exciting avenue for future research.

Fairness as an Optimisation Criterion

Recommendation systems and information retrieval applications in general can have a significant impact on the world around them. It is often desirable for these systems to actively mitigate potentially harmful biases that might otherwise be perpetuated by the actions they take. One of the multiple objectives to be optimised can therefore be a notion of amortised fairness – often related to the "equity of attention" [16] or "equity of expected exposure" [35] principles. Recent work proposes a learning algorithm to simultaneously optimise fairness and relevance in dynamical learning-to-rank scenarios [143]. Learning algorithms targeted specifically at the off-policy setting are an open area, where much work remains to be done. To ensure fairness at data collection time, a form of *safe* exploration would be imperative [72]. For on-policy bandit settings, we propose an exposure-aware arm selection algorithm in [79].

Towards Recommendations with Causal Effect

In Chapter 6, we argued in favour of using bandit feedback instead of organic user-item interactions to estimate the value of recommendations, and to eventually optimise models for a counterfactual estimate of this value. In doing so, we effectively focus on the interventionist nature of the recommendation problem, but we turn a blind eye towards the organic user-item interactions that also occur on

the system. Accurately predicting that you will listen to a musical artist if I recommend them is one thing, but would you have listened to them had they not been recommended to you?

Answering this type of question requires an understanding of cause and effect, for which we need to resort to tools from the field of causal inference [164]. Bonner and Vasile propose to learn a model that predicts interactions both in the presence and absence of a recommendation – thus targeting the *uplift* or *treatment effect* of the recommendation algorithm [17]. Sato et al. propose an alternative method to optimise rankings for causal effect that uses a label transformation approach [184].

Other work focuses on alleviating confounding bias, which arises when the treatment assignment and outcome are not conditionally independent [185]. Wang et al. tackle confounding bias from observational data alone, first fitting an exposure model that is followed by a preference model that accounts for biased exposure [233].

There are still many exciting open research questions in this space – especially combining aspects of causal reasoning with the potential avenues for future work laid out above, such as exploiting natural variations, supporting multiple objectives, and replacing the relevance notion in the expected exposure principle with that of the treatment effect. In this way, we could account for the fact that not all impressions are equally valuable for various stakeholders.

Bibliography

[1] H. Abdollahpouri, R. Burke, and B. Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 42–46. ACM, 2017.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] A. Agarwal, S. Basu, T. Schnabel, and T. Joachims. Effective evaluation using logged bandit feedback from multiple loggers. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 687–696. ACM, 2017.

[4] A. Agarwal, K. Takatsu, I. Zaitsev, and T. Joachims. A general framework for counterfactual learning-to-rank. In *Proc. of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 5–14. ACM, 2019.

[5] A. Agarwal, X. Wang, C. Li, M. Bendersky, and M. Najork. Addressing trust bias for unbiased learning-to-rank. In *Proc. of the 2019 World Wide Web Conference*, WWW '19, pages 4–14. ACM, 2019.

[6] A. Agarwal, I. Zaitsev, X. Wang, C. Li, M. Najork, and T. Joachims. Estimating position bias without intrusive interventions. In *Proc. of the 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, page 474–482. ACM, 2019.

[7] J. Alman and V. Vassilevska W. A refined laser method and faster matrix multiplication. In *Proc. of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21. Society for Industrial and Applied Mathematics, 2021.

[8] S. C. Anyosa, J. Vinagre, and A. M. Jorge. Incremental matrix co-factorization for recommender systems with implicit feedback. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1413–1418. International World Wide Web Conferences Steering Committee, 2018.

[9] J. Beel and V. Brunel. Data pruning in recommender systems research: Best practice or malpractice? In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19. ACM, 2019.

[10] J. Beel, M. Genzmehr, S. Langer, A. Nürnberger, and B. Gipp. A Comparative Analysis of Offline and Online Evaluations and Discussion of Research Paper Recommender System Evaluation. In *Proc. of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, RepSys '13, pages 7–14, 2013.

[11] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007.

[12] D. Ben-Shimon, A. Tsikinovsky, M. Friedmann, B. Shapira, L. Rokach, and J. Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *Proc. of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 357–358. ACM, 2015.

[13] J. Bennett, S. Lanning, et al. The netflix prize. In *Proc. of the KDD cup and workshop*, volume 2007, page 35, 2007.

[14] J. O. Berger and R. L. Wolpert. *The Likelihood Principle*. IMS, 1988.

[15] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proc. of the 12th International Society for Music Information Retrieval Conference*, ISMIR '11, 2011.

[16] A. J. Biega, K. P. Gummadi, and G. Weikum. Equity of attention: Amortizing individual fairness in rankings. In *Proc. of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, page 405–414. ACM, 2018.

[17] S. Bonner and F. Vasile. Causal embeddings for recommendation. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 104–112. ACM, 2018.

[18] A. Borchers, J. Herlocker, J. Konstan, and J. Riedl. Ganging up on information overload. *Computer*, 31(4):106–108, April 1998.

[19] L. Bottou, J. Peters, J. Quiñonero-Candela, D. Charles, D. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14(1):3207–3260, 2013.

[20] R. Cañamares, P. Castells, and A. Moffat. Offline evaluation options for recommender systems. *Information Retrieval Journal*, 23(4):387–410, 2020.

[21] P. Castells, N. J. Hurley, and S. Vargas. *Novelty and Diversity in Recommender Systems*, pages 881–918. Springer US, 2015.

[22] A. Chaney, B. Stewart, and B. Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 224–232. ACM, 2018.

[23] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *Proc. of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2249–2257, 2011.

[24] B. Chen, Z.and Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[25] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proc. of the 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 456–464. ACM, 2019.

[26] Y. Chen, Y. Wang, X. Zhao, J. Zou, and M. de Rijke. Block-aware item similarity models for top-n recommendation. *ACM Trans. Inf. Syst.*, 38(4), September 2020.

[27] Z. Chen, Y. Wang, D. Lin, D. Z. Cheng, L. Hong, E. H. Chi, and C. Cui. Beyond point estimate: Inferring ensemble prediction variation from neuron activation strength in recommender systems. In *Proc. of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 76–84. ACM, 2021.

[28] M. Choi, J. Kim, J. Lee, H. Shim, and J. Lee. Session-aware linear item-item models for session-based recommendation. In *Proc. of the 2021 World Wide Web Conference*, WWW '21, 2021.

[29] E. Christakopoulou and G. Karypis. Hoslim: Higher-order sparse linear method for top-n recommender systems. In *Advances in Knowledge Discovery and Data Mining*, PAKDD '14, pages 38–49. Springer International Publishing, 2014.

[30] E. Christakopoulou and G. Karypis. Local item-item models for top-n recommendation. In *Proc. of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 67–74. ACM, 2016.

[31] E. Christakopoulou and G. Karypis. Local latent space models for top-n recommendation. In *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pages 1235–1243. ACM, 2018.

[32] M. F. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 101–109. ACM, 2019.

[33] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[34] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004.

[35] F. Diaz, B. Mitra, M. D. Ekstrand, A. J. Biega, and B. Carterette. Evaluating stochastic rankings with expected exposure. In *Proc. of the 29th ACM International Conference on Information and Knowledge Management*, CIKM '20, page 275–284. ACM, 2020.

[36] M. Dudík, J. Langford, and L. Li. Doubly robust policy evaluation and learning. In *Proc. of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1097–1104, 2011.

[37] B. Dumitrascu, K. Feng, and B. E. Engelhardt. Pg-ts: Improved thompson sampling for logistic contextual bandits. In *Advances in Neural Information Processing Systems 30*, NeurIPS'18, page 4629–4638, 2018.

[38] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[39] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, February 2011.

[40] E. Elahi, W. Wang, D. Ray, A. Fenton, and T. Jebara. Variational low rank multinomials for collaborative filtering with side-information. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 340–347. ACM, 2019.

[41] V. Elvira, L. Martino, D. Luengo, and M. F. Bugallo. Generalized multiple importance sampling. *Statist. Sci.*, 34(1):129–155, 02 2019.

[42] Z. Fang, A. Agarwal, and T. Joachims. Intervention harvesting for context-dependent examination-bias estimation. In *Proc. of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 825–834. ACM, 2019.

[43] M. Farajtabar, Y. Chow, and M. Ghavamzadeh. More robust doubly robust off-policy evaluation. In *Proc. of the 35th International Conference on Machine Learning*, volume 80 of *ICML'18*, pages 1447–1456. PMLR, 10–15 Jul 2018.

[44] L. Faury, U. Tanielian, F. Vasile, E. Smirnova, and E. Dohmatob. Distributionally robust counterfactual risk minimization. In *Proc. of the 34th AAAI Conference on Artificial Intelligence*, AAAI'20. AAAI Press, 2020.

[45] E. J. Ferreira, F. Enembreck, and J. P. Barddal. Adadrift: An adaptive learning technique for long-history stream-based recommender systems. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2593–2600, 2020.

[46] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of The 33rd International Conference on Machine Learning*, ICML '16, pages 1050–1059. PMLR, 2016.

[47] I. Gama, J.and Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), March 2014.

[48] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. Offline and Online Evaluation of News Recommender Systems at Swissinfo.Ch. In *Proc. of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 169–176, 2014.

[49] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé. Offline a/b testing for recommender systems. In *Proc. of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 198–206. ACM, 2018.

[50] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[51] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4):13:1–13:19, December 2015.

[52] A. Gruson, P. Chandar, C. Charbuillet, J. McInerney, S. Hansen, D. Tardieu, and B. Carterette. Offline evaluation to make decisions about playlist recommendation algorithms. In *Proc. of the 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 420–428. ACM, 2019.

[53] J. A. Gulla, L. Zhang, P. Liu, Ö. Özgöbek, and X. Su. The adressa dataset for news recommendation. In *Proc. of the International Conference on Web Intelligence*, WI '17, page 1042–1048. ACM, 2017.

[54] D. Guo, S. I. Ktena, P. K. Myana, F. Huszar, W. Shi, A. Tejani, M. Kneier, and S. Das. Deep bayesian bandits: Exploring in online personalized recommendations. In *Proc. of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 456–461. ACM, 2020.

[55] W. W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[56] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[57] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015.

[58] R. He, W. Kang, and J. McAuley. Translation-based recommendation. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 161–169. ACM, 2017.

[59] X. He, O. Pan, J.and Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela. Practical lessons from predicting clicks on ads at facebook. In *Proc. of the 8th International Workshop on Data Mining for Online Advertising*, ADKDD'14, page 1–9. ACM, 2014.

[60] X. He, H. Zhang, M. Kan, and T. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 549–558. ACM, 2016.

[61] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237. ACM, 1999.

[62] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[63] D. Hosmer Jr., S. Lemeshow, and R. Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[64] C. Hsieh, L. Yang, Y. Cui, T. Lin, S. Belongie, and D. Estrin. Collaborative metric learning. In *Proc. of the 26th International World Wide Web Conference*, WWW '17, pages 193–201. International World Wide Web Conferences Steering Committee, 2017.

[65] Y. Hu, Y. Koren, and C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *Proc. of the 8th IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Dec 2008.

[66] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu. Tencentrec: Real-time stream recommendation in practice. In *Proc. of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 227–238. ACM, 2015.

[67] L. Hui and M. Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In *Proc. of the 9th International Conference on Learning Representations*, ICLR '21, 2021.

[68] E. Ie, C. Hsu, M. Mladenov, V. Jain, S. Narvekar, J. Wang, R. Wu, and C. Boutilier. Recsim: A configurable simulation platform for recommender systems, 2019.

[69] E. L. Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2):295–311, 2008.

[70] R. Jagerman, I. Markov, and M. de Rijke. When people change their mind: Off-policy evaluation in non-stationary recommendation environments. In *Proc. of the 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 447–455. ACM, 2019.

[71] R. Jagerman, H. Oosterhuis, and M. de Rijke. To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions. In *Proc. of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 15–24. ACM, 2019.

[72] R. Jagerman, I. Markov, and M. De Rijke. Safe exploration for optimizing contextual bandits. *ACM Trans. Inf. Syst.*, 38(3), April 2020.

[73] D. Jannach and M. Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 306–310. ACM, 2017.

[74] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[75] O. Jeunen. Revisiting offline evaluation for implicit-feedback recommender systems. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 596–600. ACM, 2019.

[76] O. Jeunen and B. Goethals. Predicting sequential user behaviour with session-based recurrent neural networks. In *Proc. of the ACM WSDM Cup Workshop*, WSDM Cup '19, 2019.

[77] O. Jeunen and B. Goethals. An empirical evaluation of doubly robust learning for recommendation. In *Proc. of the ACM RecSys Workshop on Bandit Learning from User Interactions*, REVEAL '20, 2020.

[78] O. Jeunen and B. Goethals. Pessimistic reward models for off-policy learning in recommendation. In *Proc. of the 15th ACM Conference on Recommender Systems*, RecSys '21. ACM, 2021.

[79] O. Jeunen and B. Goethals. Top-k contextual bandits with equity of exposure. In *Proc. of the 15th ACM Conference on Recommender Systems*, RecSys '21. ACM, 2021.

[80] O. Jeunen, K. Verstrepen, and B. Goethals. Fair offline evaluation methodologies for implicit-feedback recommender systems with MNAR data. In *Proc. of the ACM RecSys Workshop on Offline Evaluation for Recommender Systems*, REVEAL '18, 2018.

[81] O. Jeunen, D. Mykhaylov, D. Rohde, F. Vasile, A. Gilotte, and M. Bompaire. Learning from bandit feedback: An overview of the state-of-the-art. In *Proc. of the ACM RecSys Workshop on Reinforcement Learning and Robust Estimators for Recommendation*, REVEAL '19, 2019.

[82] O. Jeunen, D. Rohde, and F. Vasile. On the value of bandit feedback for offline recommender system evaluation. In *Proc. of the ACM RecSys Workshop on Reinforcement Learning and Robust Estimators for Recommendation*, REVEAL '19, 2019.

[83] O. Jeunen, K. Verstrepen, and B. Goethals. Efficient similarity computation for collaborative filtering in dynamic environments. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 251–259. ACM, 2019.

[84] O. Jeunen, D. Rohde, F. Vasile, and M. Bompaire. Joint policy-value learning for recommendation. In *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 1223–1233. ACM, 2020.

[85] O. Jeunen, J. Van Balen, and B. Goethals. Closed-form models for collaborative filtering with side-information. In *Proc. of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 651–656. ACM, 2020.

[86] Y. Jin, Z. Yang, and Z. Wang. Is pessimism provably efficient for offline rl?, 2020.

[87] T. Joachims, A. Swaminathan, and T. Schnabel. Unbiased learning-to-rank with biased feedback. In *Proc. of the 10th ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 781–789. ACM, 2017.

[88] T. Joachims, A. Swaminathan, and M. de Rijke. Deep learning with logged bandit feedback. In *Proc. of the 6th International Conference on Learning Representations*, ICLR '18, 2018.

[89] M. Jugovac, D. Jannach, and M. Karimi. Streamingrec: A framework for benchmarking stream-based news recommenders. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 269–273. ACM, 2018.

[90] S. Kabbur, X. Ning, and G. Karypis. Fism: Factored item similarity models for top-n recommender systems. In *Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 659–667, 2013.

[91] Kaggle. RetailRocket Recommender System Dataset. https://www.kaggle.com/retailrocket/ecommerce-dataset, 2016.

[92] M. Karimi, D. Jannach, and M. Jugovac. News recommender systems - survey and roads ahead. *Information Processing & Management*, 54(6):1203 – 1227, 2018.

[93] F. Khawar, L. Poon, and N. L. Zhang. Learning the structure of auto-encoding recommenders. In *Proc. of The Web Conference 2020*, WWW '20, page 519–529. ACM, 2020.

[94] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS '20*, 2020.

[95] B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz. The plista dataset. In *Proc. of the International News Recommender Systems Workshop and Challenge*, NRS '13, pages 16–23. ACM, 2013.

[96] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. of the International Joint Conference on Artificial Intelligence*, volume 14 of *IJCAI '95*, pages 1137–1145, 1995.

[97] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 426–434. ACM, 2008.

[98] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[99] W. Krichene and S. Rendle. On sampled metrics for item recommendation. In *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 1748–1757. ACM, 2020.

[100] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS '20*, 2020.

[101] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45:255–282, 1950.

[102] T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[103] Q. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Ng. On optimization methods for deep learning. In *Proc. of the 28th International Conference on International Conference on Machine Learning*, ICML '11, pages 265–272, 2011.

[104] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, page 296–303. ACM, 2014.

[105] P. Lee, L. Lakshmanan, M. Tiwari, and S. Shah. Modeling impression discounting in large-scale recommender systems. In *Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1837–1846. ACM, 2014.

[106] D. Lefortier, A. Swaminathan, X. Gu, T. Joachims, and M. de Rijke. Large-scale validation of counterfactual learning methods: A test-bed. *arXiv preprint arXiv:1612.00367*, 2016.

[107] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[108] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[109] M. Levy and K. Jack. Efficient top-n recommendation by linear regression. In *RecSys Large Scale Recommender Systems Workshop*, 2013.

[110] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, volume 27 of *NIPS '14*, pages 2177–2185, 2014.

[111] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1-2):135–163, 2013.

[112] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. of the 19th International Conference on World Wide Web*, WWW '10, page 661–670. ACM, 2010.

[113] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. of the 4th ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 297–306. ACM, 2011.

[114] S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 539–548. ACM, 2016.

[115] X. Li and J. She. Collaborative variational autoencoder for recommender systems. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 305–314, 2017.

[116] D. Liang, R. G. Krishnan, M. D Hoffman, and T. Jebara. Variational autoencoders for collaborative filtering. In *Proc. of the World Wide Web Conference*, WWW '18, pages 689–698. ACM, 2018.

[117] E. Liberty, F. Woolfe, P. Martinsson, V. Rokhlin, and M. Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proc. of the National Academy of Sciences*, 104(51):20167–20172, 2007.

[118] X. Lin, H. Chen, C. Pei, F. Sun, X. Xiao, H. Sun, Y. Zhang, W. Ou, and P. Jiang. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 20–28. ACM, 2019.

[119] N. Liu, M. Zhao, E. Xiang, and Q. Yang. Online evolutionary collaborative filtering. In *Proc. of the 4th ACM Conference on Recommender Systems*, RecSys '10, pages 95–102. ACM, 2010.

[120] Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Provably good batch off-policy reinforcement learning without great exploration. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS '20*, 2020.

[121] B. London and T. Joachims. Offline policy evaluation with new arms. In *Offline Reinforcement Learning Workshop at Neural Information Processing Systems*, 2020.

[122] B. London and T. Sandler. Bayesian counterfactual risk minimization. In *Proc. of the 36th International Conference on Machine Learning*, volume 97 of *ICML '19*, pages 4125–4133. PMLR, 2019.

[123] R. Lopez, I. Dhillion, and M. I. Jordan. Learning from extreme bandit feedback. In *Proc. of the 35th AAAI Conference on Artificial Intelligence*, AAAI'21. AAAI Press, 2021.

[124] M. Ludewig and D. Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28(4):331–390, 2018.

[125] X. Luo, Y. Xia, Q. Zhu, and Y. Li. Boosting the k-nearest-neighborhood based incremental collaborative filtering. *Knowledge-Based Systems*, 53:90 – 99, 2013.

[126] C. Ma, L. Ma, Y. Zhang, R. Tang, X. Liu, and M. Coates. Probabilistic metric learning with adaptive margin for top-k recommendation. In *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 1036–1044. ACM, 2020.

[127] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, page 1930–1939. ACM, 2018.

[128] J. Ma, Z. Zhao, X. Yi, J. Yang, M. Chen, J. Tang, L. Hong, and E. H. Chi. Off-policy learning in two-stage recommender systems. In *Proc. of the International World Wide Web Conference*, WWW '20. ACM, 2020.

[129] Y. Ma, Y. Wang, and B. Narayanaswamy. Imitation-regularized offline learning. In *Proc. of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *AIStats '19*, pages 2956–2965. PMLR, 2019.

[130] M. Mansoury, H. Abdollahpouri, M. Pechenizkiy, B. Mobasher, and R. Burke. Feedback loop and bias amplification in recommender systems. In *Proc. of the 29th ACM International Conference on Information and Knowledge Management*, CIKM '20, page 2145–2148. ACM, 2020.

[131] P. G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30(1):47 – 68, 2011.

[132] A. Masegosa. Learning under model misspecification: Applications to variational and ensemble methods. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS '20*, pages 5479–5491, 2020.

[133] P. Matuszyk, J. Vinagre, M. Spiliopoulou, A. M. Jorge, and J. Gama. Forgetting techniques for stream-based matrix factorization in recommender systems. *Knowledge and Information Systems*, 55(2):275–304, 2018.

[134] A. Maurer and M. Pontil. Empirical bernstein bounds and sample variance penalization. *Stat.*, 1050:21, 2009.

[135] B. C. May, N. Korda, A. Lee, and D. S. Leslie. Optimistic bayesian sampling in contextual-bandit problems. *J. Mach. Learn. Res.*, 13(1):2069–2106, June 2012.

[136] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra. Explore, exploit, and explain: Personalizing explainable recommendations with bandits. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 31–39. ACM, 2018.

[137] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1222–1230. ACM, 2013.

[138] R. Mehrotra, J. McInerney, H. Bouchard, M. Lalmas, and F. Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proc. of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 2243–2251. ACM, 2018.

[139] R. Mehrotra, C. Shah, and B. Carterette. Investigating listeners' responses to divergent recommendations. In *Proc. of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 692–696. ACM, 2020.

[140] R. Mehrotra, N. Xue, and M. Lalmas. Bandit based optimization of multiple objectives on a music streaming platform. In *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 3224–3233. ACM, 2020.

[141] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26 of *NIPS '13*, pages 3111–3119, 2013.

[142] S. Moens, O. Jeunen, and B. Goethals. Interactive evaluation of recommender systems with sniper: An episode mining approach. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 538–539. ACM, 2019.

[143] M. Morik, A. Singh, J. Hong, and T. Joachims. Controlling fairness and bias in dynamic learning-to-rank. In *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 429–438. ACM, 2020.

[144] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

[145] K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2021.

[146] D. Mykhaylov, D. Rohde, F. Vasile, M. Bompaire, and O. Jeunen. Three methods for training on bandit feedback. In *Proc. of the NeurIPS Workshop on Causality and Machine Learning*, CausalML '19, 2019.

[147] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Proc. of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 497–506. IEEE Computer Society, 2011.

[148] X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In *Proc. of the 6th ACM Conference on Recommender Systems*, RecSys '12, page 155–162, 2012.

[149] X. Ning, A. N. Nikolakopoulos, Z. Shui, M. Sharma, and G. Karypis. SLIM Library for Recommender Systems, 2019. URL `https://github.com/KarypisLab/SLIM`.

[150] Y. Ning, Y. Shi, L. Hong, H. Rangwala, and N. Ramakrishnan. A gradient-based adaptive learning framework for efficient personal recommendation. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 23–31. ACM, 2017.

[151] H. Oosterhuis and M. de Rijke. Policy-aware unbiased learning to rank for top-k rankings. In *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 489–498. ACM, 2020.

[152] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, volume 29, pages 4026–4034, 2016.

[153] G. Ottaviano and R. Venturini. Partitioned elias-fano indexes. In *Proc. of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 273–282. ACM, 2014.

[154] R. Otunba, R. Rufai, and J. Lin. Mpr: Multi-objective pairwise ranking. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 170–178. ACM, 2017.

[155] Outbrain. Kaggle click prediction dataset. https://www.kaggle.com/c/outbrain-click-prediction/data, 2017.

[156] A. B. Owen. *Monte Carlo theory, methods and examples.* 2013.

[157] C. C. Paige. Accuracy and effectiveness of the lanczos algorithm for the symmetric eigenproblem. *Linear algebra and its applications*, 34:235–258, 1980.

[158] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proc. of the 8th IEEE International Conference on Data Mining*, ICDM '08, pages 502–511, Dec 2008.

[159] V. Y. Pan and Z. Q. Chen. The complexity of the matrix eigenproblem. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing*, STOC '99, page 507–516. ACM, 1999.

[160] M. Papagelis, I. Rousidis, D. Plexousakis, and E. Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. In *Foundations of Intelligent Systems*, pages 553–561. Springer, 2005.

[161] Y. Park and A. Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proc. of the 1st ACM Conference on Recommender Systems*, RecSys '08, page 11–18, 2008.

[162] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, NeurIPS '19, pages 8026–8037, 2019.

[163] B. Paudel, T. Haas, and A. Bernstein. Fewer flops at the top: Accuracy, diversity, and regularization in two-class collaborative filtering. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 215–223. ACM, 2017.

[164] J. Pearl. *Causality*. Cambridge university press, 2009.

[165] A. Peysakhovich and D. Eckles. Learning causal effects from many randomized experiments using regularized instrumental variables. In *Proc. of the 2018 World Wide Web Conference*, WWW '18, page 699–707. International World Wide Web Conferences Steering Committee, 2018.

[166] G. E. Pibiri, M. Petri, and A. Moffat. Fast dictionary-based compression for inverted indexes. In *Proc. of the 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 6–14. ACM, 2019.

[167] M. Quadrana, P. Cremonesi, and D. Jannach. Sequence-aware recommender systems. *ACM Comput. Surv.*, July 2018.

[168] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proc. of the 1st ACM Conference on Recommender Systems*, RecSys '08, pages 251–258. ACM, 2008.

[169] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461. AUAI Press, 2009.

[170] S. Rendle, L. Zhang, and Y. Koren. On the Difficulty of Evaluating Baselines: A Study on Recommender Systems, 2019.

[171] S. Rendle, W. Krichene, L. Zhang, and J. Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *Proc. of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 240–248, 2020.

[172] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3): 56–58, March 1997.

[173] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, page 175–186. ACM, 1994.

[174] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proc. of the 6th ACM Conference on Recommender Systems*, RecSys '12, page 19–26. ACM, 2012.

[175] D. S. Robertson. The information revolution. *Communication Research*, 17(2): 235–254, 1990.

[176] D. Rohde, S. Bonner, T. Dunlop, F. Vasile, and A. Karatzoglou. RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising. In *Proc. of the ACM RecSys Workshop on Offline Evaluation for Recommender Systems*, REVEAL '18, 2018.

[177] M. Rossetti, F. Stella, and M. Zanker. Contrasting Offline and Online Results when Evaluating Recommendation Algorithms. In *Proc. of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 31–34. ACM, 2016.

[178] N. Sachdeva, Y. Su, and T. Joachims. Off-policy bandits with deficient support. In *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 965–975. ACM, 2020.

[179] Y. Saito, S. Aihara, M. Matsutani, and Y. Narita. Large-scale open dataset, pipeline, and benchmark for bandit algorithms, 2020.

[180] O. Sakhi, S. Bonner, D. Rohde, and F. Vasile. Blob : A probabilistic model for recommendation that combines organic and bandit signals. In *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, pages 783–793. ACM, 2020.

[181] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proc. of the 25th International Conference on Machine Learning*, ICML '08, pages 880–887. ACM, 2008.

[182] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *Proc. of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 743–754. ACM, 2004.

[183] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295. ACM, 2001.

[184] M. Sato, J. Singh, S. Takemori, T. Sonoda, Q. Zhang, and T. Ohkuma. Uplift-based evaluation and optimization of recommenders. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 296–304. ACM, 2019.

[185] M. Sato, S. Takemori, J. Singh, and T. Ohkuma. Unbiased learning for the causal effect of recommendation. In *Proc. of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 378–387. ACM, 2020.

[186] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and Metrics for Cold-start Recommendations. In *Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260. ACM, 2002.

[187] S. Sedhain, A. K. Menon, S. Sanner, and D. Braziunas. On the effectiveness of linear models for one-class collaborative filtering. In *Proc. of the 30th AAAI Conference on Artificial Intelligence*, AAAI'16, page 229–235, 2016.

[188] G. Shani and A. Gunawardana. Evaluating recommendation systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[189] I. Shenbin, A. Alekseev, E. Tutubalina, V. Malykh, and S. I. Nikolenko. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In *Proc. of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 528–536, 2020.

[190] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1), May 2014.

[191] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227 – 244, 2000.

[192] N. Si, F. Zhang, Z. Zhou, and J. Blanchet. Distributionally robust policy evaluation and learning in offline contextual bandits. In *International Conference on Machine Learning*, ICML'20, 2020.

[193] A. Sinha, D. Gleich, and K Ramani. Deconvolving feedback loops in recommender systems. In *Advances in Neural Information Processing Systems 29*, NeurIPS '16, pages 3251–3259, 2016.

[194] J. E. Smith and R. L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

[195] D. Song and David A. Meyer. Recommending positive links in signed social networks by optimizing a generalized auc. In *Proc. of the 29th AAAI Conference on Artificial Intelligence*, AAAI'15, pages 290–296. AAAI Press, 2015.

[196] R. S. Sreepada and B. K. Patra. An incremental approach for collaborative filtering in streaming scenarios. In *Advances in Information Retrieval*, pages 632–637. Springer International Publishing, 2018.

[197] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 713–722. ACM, 2010.

[198] H. Steck. Item popularity and recommendation accuracy. In *Proc. of the 5th ACM Conference on Recommender Systems*, RecSys '11, pages 125–132. ACM, 2011.

[199] H. Steck. Evaluation of recommendations: Rating-prediction and ranking. In *Proc. of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 213–220. ACM, 2013.

[200] H. Steck. Embarrassingly shallow autoencoders for sparse data. In *Proc. of the World Wide Web Conference*, WWW '19, page 3251–3257, 2019.

[201] H. Steck. Collaborative filtering via high-dimensional regression. *CoRR*, abs/1904.13033, 2019.

[202] H. Steck. Markov random fields for collaborative filtering. In *Advances in Neural Information Processing Systems 32*, NeurIPS '19, pages 5473–5484, 2019.

[203] H. Steck, M. Dimakopoulou, N. Riabov, and T. Jebara. Admm slim: Sparse recommendations for many users. In *Proc. of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 555–563, 2020.

[204] A. Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, pages 3–28, 2009.

[205] Y. Su, L. Wang, M. Santacatterina, and T. Joachims. Cab: Continuous adaptive blending for policy evaluation and learning. In *International Conference on Machine Learning*, ICML'19, pages 6005–6014, 2019.

[206] Y. Su, M. Dimakopoulou, A. Krishnamurthy, and M. Dudik. Doubly robust off-policy evaluation with shrinkage. In *Proc. of the 37th International Conference on Machine Learning*, ICML '20, pages 9167–9176. PMLR, 2020.

[207] K. Subbian, C. Aggarwal, and K. Hegde. Recommendations for streaming data. In *Proc. of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2185–2190. ACM, 2016.

[208] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, volume 135. "MIT Press", 1998.

[209] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, NIPS'99, pages 1057–1063, 1999.

[210] A. Swaminathan and T. Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *Proc. of the 32nd International Conference on International Conference on Machine Learning*, ICML'15, pages 814–823. JMLR.org, 2015.

[211] A. Swaminathan and T. Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16(1):1731–1755, 2015.

[212] A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems*, NeurIPS '15, pages 3231–3239, 2015.

[213] H. Tang, J. Liu, M. Zhao, and X. Gong. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proc. of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 269–278. ACM, 2020.

[214] S. Ubaru and Y. Saad. Fast methods for estimating the numerical rank of large matrices. In *Proc. of the 33rd International Conference on Machine Learning - Volume 48*, ICML'16, page 468–477. JMLR.org, 2016.

[215] D. Valcarce, A. Bellogín, J. Parapar, and P. Castells. Assessing ranking metrics in top-n recommendation. *Information Retrieval Journal*, 23(4):411–448, 2020.

[216] J. Van Balen and B. Goethals. High-dimensional sparse embeddings for collaborative filtering. In *Proc. of the Web Conference 2021*, WWW '21, page 575–581. ACM, 2021.

[217] S. Van Der Walt, S. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13 (2):22–30, 2011.

[218] F. Vasile, D. Rohde, O. Jeunen, and A. Benhalloum. A gentle introduction to recommendation as counterfactual policy learning. In *Proc. of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20, page 392–393. ACM, 2020.

[219] F. Vasile, D. Rohde, O. Jeunen, A. Benhalloum, and O. Sakhi. Recommender systems through the lens of decision theory. In *Proc. of the 30th World Wide Web Conference ACM Conference*, WebConf '21. ACM, 2021.

[220] K. Verstrepen and B. Goethals. Unifying nearest neighbors collaborative filtering. In *Proc. of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 177–184. ACM, 2014.

[221] K. Verstrepen, K. Bhaduriy, B. Cule, and B. Goethals. Collaborative filtering for binary, positiveonly data. *SIGKDD Explor. Newsl.*, 19(1):1–21, September 2017.

[222] J. Vinagre, A. M. Jorge, and J. Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In *User Modeling, Adaptation, and Personalization*, pages 459–470, Cham, 2014. Springer International Publishing.

[223] J. Vinagre, A. Jorge, and J. Gama. Evaluation of recommender systems in streaming environments. *CoRR*, abs/1504.08175, 2015. URL `http://arxiv.org/abs/1504.08175`.

[224] J. Vinagre, A. M. Jorge, and J. Gama. Collaborative filtering with recency-based negative feedback. In *Proc. of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, page 963–965, 2015.

[225] J. Vinagre, A. M. Jorge, M. Al-Ghossein, and A. Bifet. *ORSUM - Workshop on Online Recommender Systems and User Modeling*, page 619–620. ACM, 2020.

[226] A. D. Viniski, J. P. Barddal, A. de Souza Britto Jr., F. Enembreck, and H. V. A. de Campos. A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start. *Expert Systems with Applications*, 176:114890, 2021.

[227] P. Virtanen, R. Gommers, T. E. Oliphant, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

[228] N. Vlassis, A. Bibaut, M. Dimakopoulou, and T. Jebara. On the design of estimators for bandit off-policy evaluation. In *Proc. of the 36th International Conference on Machine Learning*, volume 97 of *ICML'19*, pages 6468–6476. PMLR, 09–15 Jun 2019.

[229] T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 591–598. AUAI Press, 2009.

[230] M. Wan and J. McAuley. Item recommendation on monotonic behavior chains. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 86–94. ACM, 2018.

[231] Q. Wang, H. Yin, Z. Hu, D. Lian, H. Wang, and Z. Huang. Neural memory streaming recommender networks with adversarial training. In *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pages 2467–2475. ACM, 2018.

[232] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. Nguyen. Streaming ranking based recommender systems. In *Proc. of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, pages 525–534. ACM, 2018.

[233] Y. Wang, D. Liang, L. Charlin, and D. M. Blei. Causal inference for recommender systems. In *Proc. of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 426–431. ACM, 2020.

[234] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, May 1992.

[235] F. Wu, Y. Qiao, J. Chen, C. Wu, T. Qi, J. Lian, D. Liu, X. Xie, J. Gao, W. Wu, and M. Zhou. MIND: A large-scale dataset for news recommendation. In *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL'20, pages 3597–3606. ACL, 2020.

[236] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose. Self-supervised reinforcement learning for recommender systems. In *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 931–940. ACM, 2020.

[237] C. Yang, X. Yu, and Y. Liu. Continuous knn join processing for real-time recommendation. In *Proc. of the 14th IEEE International Conference on Data Mining*, ICDM '14, pages 640–649, 2014.

[238] J. Yang, C. Chen, C. Wang, and M. Tsai. Hop-rec: High-order proximity for implicit recommendation. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 140–144. ACM, 2018.

[239] L. Yang, Y. Cui, Yuan X., C. Wang, S. Belongie, and D. Estrin. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 279–287. ACM, 2018.

[240] X. Yang, Z. Zhang, and K. Wang. Scalable collaborative filtering using incremental update and local link prediction. In *Proc. of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2371–2374. ACM, 2012.

[241] C. Yu, B. Ooi, K. Tan, and H. Jagadish. Indexing the distance: An efficient method to knn processing. In *Proc. of the 27th International Conference on Very Large Databases*, VLDB '01, pages 421–430, 2001.

[242] C. Yu, B. Cui, S. Wang, and J. Su. Efficient index-based knn join processing for high-dimensional data. *Inf. Softw. Technol.*, 49(4):332–344, April 2007.

[243] C. Yu, R. Zhang, Y. Huang, and H. Xiong. High-dimensional knn joins with incremental updates. *GeoInformatica*, 14(1):55–82, 2009.

[244] J. Yu, S.V.N. Vishwanathan, S. Günter, and N. Schraudolph. A quasi-newton approach to nonsmooth convex optimization problems in machine learning. *Journal of Machine Learning Research*, 11(Mar):1145–1200, 2010.

[245] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS '20*, 2020.

[246] R. Zadeh and A. Goel. Dimension independent similarity computation. *The Journal of Machine Learning Research*, 14(1):1605–1626, 2013.

[247] J. C. Zagal, J. Ruiz del Solar, and P. Vallejos. Back to reality: Crossing the reality gap in evolutionary robotics. *IFAC Proceedings Volumes*, 37(8):834–839, 2004.

[248] C. Zhang, F. Li, and J. Jestes. Efficient parallel knn joins for large data in mapreduce. In *Proc. of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 38–49. ACM, 2012.

[249] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman. Using singular value decomposition approximation for collaborative filtering. In *Proc. of the 7th IEEE International Conference on E-Commerce Technology*, CEC '05, pages 257–264, July 2005.

[250] X. Zhang, J. Zhao, and J. C.S. Lui. Modeling the assimilation-contrast effects in online product rating systems: Debiasing and recommendations. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, page 98–106. ACM, 2017.

[251] Y. Zhang, H. Lu, W. Niu, and J. Caverlee. Quality-aware neural complementary item recommendation. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 77–85. ACM, 2018.

[252] Q. Zhao, J. Chen, M. Chen, S. Jain, A. Beutel, F. Belletti, and E. H. Chi. Categorical-attributes-based item classification for recommender systems. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 320–328. ACM, 2018.

[253] Q. Zhao, M. C. Willemsen, G. Adomavicius, F. M. Harper, and J. A. Konstan. Interpreting user inaction in recommender systems. In *Proc. of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 40–48. ACM, 2018.

[254] Z. Zhao, L. Hong, L. Wei, J. Chen, A. Nath, S. Andrews, A. Kumthekar, M. Sathiamoorthy, X. Yi, and E. H. Chi. Recommending what video to watch next: A multitask ranking system. In *Proc. of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 43–51. ACM, 2019.

[255] H. Zou, K. Kuang, B. Chen, P. Chen, and P. Cui. Focused context balancing for robust offline policy evaluation. In *Proc. of the 25th ACM Conference on Knowledge Discovery and Data Mining*, KDD '19, pages 696–704. ACM, 2019.

Summary

Recommender systems are information retrieval applications that provide users with algorithmic recommendations, in order to assist decision-making when sufficient knowledge about the various options is lacking. These systems have known widespread adoption in recent years and are extensively used by digital platforms to suggest restaurants, books, musical artists, retail products and even romantic partners – much like the recommendations you could provide for a friend.

Modern approaches to recommendation typically follow the *collaborative filtering* paradigm, making use of a large dataset of user behaviour to infer preferences, and to subsequently predict your preferences based on your historical behaviour. Impressive advances in machine learning in recent years have found their way to the recommendation field, and these systems are becoming ever more accurate when it comes to learning and predicting user preferences. There is, however, a gap between the recommendation use-case that is often posed in academic research and the use-case that practitioners typically face in industry.

First, the research literature typically deals with a single static dataset on which a model is trained once and then evaluated with respect to its prediction capabilities. In contrast, recommendation models in the real world are often part of a much more dynamic ecosystem where new data is constantly coming in and models need to be kept up-to-date to remain competitive. In such settings, a clear need arises for models that can be computed *efficiently* and *incrementally*.

Second, newly proposed methods in the research literature are often evaluated using offline procedures on datasets containing user-item interactions. While this is common practice in the broader machine learning field, recommendation datasets often lack *true* labels, and currently existing evaluation procedures conflate interactions with preference expressions and more dubiously, a lack of interaction with *negative* feedback. This is especially problematic when recommendations skew item exposure. Consequently, results obtained from offline procedures are notoriously uncorrelated with those obtained via the golden standard in industry of using online experiments such as randomised control trials – also known as A/B-tests.

Third, most approaches to recommendation learn from observational datasets consisting of user-item interactions. As they do not take any information regarding previously shown recommendations and their outcomes into account – they take a purely passive stance that focuses on prediction, which contrasts with the interventionist nature of the problem in practice. Indeed, we wish to show recommendations to users in hopes of encouraging additional interactions. As such data is being generated by every online platform with a recommendation component – it comes naturally that the value of such datasets needs to be explored.

Contributions

- In **Chapter 1**, we briefly review the history of the recommendation field, from rating prediction and the Netflix prize to the implicit-feedback setting that is more prevalent in modern research. We introduce the recommendation task as it often occurs in the literature and in practice, and present some classical families of approaches to solving it. We contextualise and motivate the research questions and contributions that make up the rest of this dissertation.

- In **Chapter 2**, we deal with the problem of efficiently computing all pairwise similarities for a set of sparse, high-dimensional vectors. As such vectors are typical for the data that is used in implicit-feedback recommender systems, general methods for efficiently computing these similarities are especially well-suited for item-based collaborative filtering approaches. We propose the *Dynamic Index* algorithm to compute all exact pairwise similarities between items in collaborative filtering scenarios, and show how it can effectively exploit vector sparsity to improve upon the efficiency of competing approaches. Our algorithm consists of a single pass over the data, making it naturally suited for incremental computations and streaming, real-time updates. We present our algorithm with a MapReduce-inspired parallellisation procedure that scales favourably with the number of available computational cores. Additionally, we introduce the concept of item "recommendability" – as many items in the catalogue might not be suitable recommendations at any given time due to limited stock, recency, licensing or any other reason. We incorporate this naturally occurring phenomenon into our method and show how it can further improve the computational efficiency of our proposed method.

- In **Chapter 3**, we focus on item-based models optimised through ridge regression for collaborative filtering tasks, as recent work has shown that they can achieve highly competitive results compared to more complex and less efficient alternative methods. The main reason for this improved efficiency, is that the solution to the regression problem has an analytically computable solution. As this computation involves the inversion of the Gramian item-item matrix, it is only suitable in situations where the number of items in the catalogue is modest. Indeed, matrix inversion scales cubically with the dimensions of the matrix, quickly becoming intractable as it grows. When the model needs to be recomputed iteratively to incorporate new incoming data, this pressure on computation time can become even more problematic. To this end, we propose *Dynamic* EASE^{R} (DYN-EASE^{R}), a novel algorithm that efficiently updates an existing ridge regression model when new data arrives. DYN-EASE^{R} adopts parts of the Dynamic Index algorithm to efficiently update the Gramian matrix, and then uses the Woodbury matrix identity to incrementally update its inverse. We additionally introduce an approximate variant that updates the inverse with a low-rank approximation of the model update, trading in the exactness of the solution for a gain in computational efficiency. We theoretically analyse our methods, and empirically show significant efficiency gains for collaborative filtering use-cases where the environment is highly dynamic.

- In **Chapter 4**, we look at the currently predominant offline evaluation procedures in the literature. Stemming from the machine learning field, recommendation models are typically trained on user-item interactions available in a training set and evaluated on their performance to predict which interactions are part of a test set, which was randomly sampled and held out from the full dataset. Two issues arise with this paradigm: 1. the sequential nature of user-item interactions is entirely neglected for the train-test split, so we effectively evaluate models based on their ability to predict the past from the future, and 2. deployed recommendation algorithms at data collection time are ignored, while they can significantly bias item exposure and lead to interactions that are Missing-Not-At-Random (MNAR).

  Results stemming from such evaluation procedures seldom align with those from online experiments – indicating that these issues have a significant impact on results. To alleviate (1), we propose an alternative evaluation approach that adheres much more tightly to the inherent characteristics of web-based recommender systems: *Sliding Window Evaluation* (SW-EVAL). Empirical observations indicate that SW-EVAL is less prone to over-estimate model performance, correctly identifies the power of popularity-based baselines, and generally ranks competing algorithms much differently than widely used alternative methods. Using real-world data from a Belgian retail website, we highlight that (2) severely influences results, and skews them in favour of those approaches that mimic the logging policy. We present a scope for future research to model these MNAR biases and take them into account during training and evaluation to attain unbiased performance estimates.

- In **Chapter 5**, we revisit currently existing offline evaluation procedures for implicit-feedback recommender systems. We briefly review offline and online procedures, highlighting where they tend to diverge. Online approaches are highly effective in measuring what we care about, but are often very expensive and thus inefficient. Offline approaches, in contrast, are much more efficient but often ineffective at predicting *true* online performance when deployed. In this Chapter, we study why they diverge, and how we can realign the offline objective to reflect online performance. We identify three ways to improve and extend current work on offline evaluation methodologies. First, as we also highlighted in Chapter 5, it is of crucial importance to correctly abide by sequential constraints in the data, and temporal evaluation procedures can provide ways forward in this regard. Second, to alleviate MNAR biases that skew collected data, we need to model the logging policy to be able to account for it. We draw parallels with the problem of off-policy evaluation in the reinforcement learning literature, and map these concepts to the recommendation problem. Third, we believe there is much to gain by using more data than just views and clicks to evaluate recommendation performance. Indeed, as we often have information regarding recommendations that were shown, together with user *inaction*, we can use it to distinguish between missing and negative feedback. We propose a research agenda focused on these three topics, with the goal of devising offline evaluation procedures that accurately reflect online performance.

- In **Chapter 6**, we explore how methods that learn from shown recommendations and their outcomes can be used for recommendation tasks. Since we do not know whether users would have clicked on items we did not show, learning directly from such data can be highly non-trivial. Several methods for so-called off-policy or counterfactual learning have been proposed for general machine learning tasks in recent years, but their efficacy as recommendation policies remains understudied. We present an overview of existing methods in the context of the recommendation problem, and conduct a broad empirical study to validate their worth. We show that existing off-policy learning methods tend to fail when rewards are stochastic and highly sparse, and we show how a logarithmic lower-bound on the traditional importance sampling estimator can alleviate these issues. Additionally, the logarithmic transformation convexifies the optimisation objective and facilitates convergence. We show how value- and policy-based families of approaches can be formulated with an identical parameterisation, and propose a novel model that jointly optimises both types of objectives. We show that this *Dual Bandit* approach is highly competitive with the state-of-the-art, especially in the realistic cases where the amount of logging randomisation and the size of the training sample are limited, while the action space is large.

- In **Chapter 7**, we turn towards reward models learned from bandit feedback – such as models that predict the probability of a click on a given recommendation. Learning such models accurately is no easy feat when the logging policy that chooses actions at data collection time is highly skewed. That is, it greedily shows recommendations it deems relevant, and only rarely explores alternatives. This is known as item selection bias, and it leads to heteroscedastic variance on the resulting reward estimates. We show how this can be detrimental to the recommendation performance of a recommendation policy relying on such a model of reward. Additionally, we show how this exacerbates a problem known as the "Optimiser's Curse", where the post-decision disappointment from acting on these reward estimates will be non-negative in many realistic cases. To counteract these problems, we propose a general uncertainty-aware decision-making framework that treats the predictions of the reward model with a healthy dose of scepticism. Bayesian uncertainty estimates can reveal what the reward model does not know, and we can use this gained knowledge to fall back to best worst-case decisions instead of the usual uncertainty-ignorant ones. Extensive experiments show how our proposed method can lead to significant and robust gains in recommendation performance.

- In **Chapter 8**, we conclude by summarising the research contributions presented throughout this dissertation and showing how they are connected to one another. We present a scope for future research to bridge the gap between recommendations in industry and academia – focusing on realistic extensions of the bandit learning paradigm that alleviate some of its rigid assumptions, so they ultimately provide a better reflection of the real world.

Samenvatting

Aanbevelingssystemen zijn toepassingen die gebruikers algoritmische aanbevelingen geven, met het doel om beslissingen te vergemakkelijken wanneer er een gebrek is aan voldoende kennis over de alternatieven. Deze systemen zijn de afgelopen jaren wijdverspreid geraakt en worden gebruikt door digitale platformen om restaurants, boeken, artiesten, producten en zelfs romantische partners voor te stellen – net zoals de aanbevelingen die u zou kunnen geven aan een vriend.

Moderne aanbevelings-methoden volgen doorgaans het "*collaborative filtering*" paradigma, waarbij gebruik wordt gemaakt van een grote dataset van gebruikersgedrag om voorkeuren af te leiden en vervolgens uw voorkeuren te voorspellen op basis van uw historisch gedrag. Een indrukwekkende vooruitgang in machinaal leren in de afgelopen jaren heeft zijn weg gevonden naar het aanbevelingsveld, en deze systemen worden steeds nauwkeuriger als het gaat om het leren en voorspellen van gebruikersvoorkeuren. Er is echter een kloof tussen de use-case van aanbevelingen die vaak wordt gesteld in academisch onderzoek en de use-case waarmee beoefenaars doorgaans in de industrie worden geconfronteerd.

Ten eerste behandelt de onderzoeksliteratuur typisch een enkele statische dataset waarop een model eenmalig wordt berekend en vervolgens geëvalueerd met betrekking tot zijn voorspellings-nauwkeurigheid. Aanbevelingsmodellen in de echte wereld maken daarentegen vaak deel uit van een dynamischer ecosysteem waar voortdurend nieuwe data binnenkomen en modellen up-to-date moeten worden gehouden om competitief te blijven. In dergelijke situaties ontstaat een duidelijke behoefte aan modellen die *efficiënt* en *incrementeel* kunnen worden berekend.

Ten tweede worden nieuw voorgestelde methoden in de onderzoeksliteratuur vaak geëvalueerd met behulp van offline evaluatie-procedures op datasets met interacties tussen gebruikers en items. Hoewel dit een gangbare praktijk is in het bredere onderzoeksveld m.b.t. machinaal leren, missen aanbevelingsdatasets vaak *echte* labels, en huidige evaluatie-procedures verwarren interacties met voorkeursuitdrukkingen en, erger nog, een gebrek aan interactie met negatieve feedback. Dit is vooral problematisch wanneer aanbevelingen de blootstelling van gebruikers aan verscheidene items kromtrekken. Resultaten verkregen uit offline procedures zijn notoir slecht gecorreleerd met die verkregen via de gouden standaard van online experimenten zoals gerandomiseerde controleproeven – beter gekend als A/B-testen.

Ten derde leren de meeste aanbevelings-methoden uit observationele datasets die bestaan uit interacties tussen gebruikers en items. Omdat ze geen rekening houden met eerder getoonde aanbevelingen en hun gevolgen, nemen ze een puur

passieve houding aan om te voorspellen, wat in contrast staat met het interventionistische karakter van het aanbevelings-probleem in de praktijk. We willen inderdaad aanbevelingen aan gebruikers tonen in de hoop om bijkomende interacties aan te moedigen. Aangezien dergelijke data worden gegenereerd door elk online platform met een aanbevelingscomponent, is het vanzelfsprekend dat de waarde van dergelijke datasets interessant is om te onderzoeken.

Bijdragen

- In **Hoofdstuk 1** bespreken we kort de geschiedenis van het onderzoeks-veld rond aanbevelings-systemen, van het voorspellen van *ratings* en de Netflix-prijs tot de impliciete feedback-setting die vaker voorkomt in modern onderzoek. We introduceren het aanbevelings-probleem zoals het vaak voorkomt in de literatuur en in de praktijk, en bespreken enkele klassieke families van methoden om het op te lossen. We contextualiseren en motiveren de onderzoeksvragen en bijdragen die de rest van dit proefschrift opmaken.

- In **Hoofdstuk 2** behandelen we het probleem van het efficiënt berekenen van alle paarsgewijze gelijkenissen voor een reeks ijle, hoog-dimensionale vectoren. Aangezien dergelijke vectoren typerend zijn voor de data die worden gebruikt in aanbevelingssystemen met impliciete feedback, zijn algemene methoden voor het efficiënt berekenen van deze gelijkenissen bijzonder geschikt voor op "item-based collaborative filtering" modellen. We stellen het *Dynamic Index* algoritme voor om alle exacte paarsgewijze gelijkenissen tussen items in collaborative filtering scenario's te berekenen, en we tonen hoe het effectief de ijlheid van de vectoren kan benutten om de efficiëntie van concurrerende methoden te verbeteren. Aangezien onze methode maar éénmalig over de data gaat, is ze geschikt voor incrementele berekeningen en streaming, real-time updates. We presenteren ons algoritme met een op MapReduce geïnspireerde parallellisatie-procedure die gunstig schaalt met het aantal beschikbare rekenkernen. Daarnaast introduceren we het concept van "recommendability": aangezien veel items in de catalogus op eender welk moment ongeschikt kunnen zijn om aan te bevelen vanwegede voorraad, recentheid, licenties of andere redenenen. We integreren dit natuurlijk voorkomende fenomeen in onze methode en laten zien hoe het de rekenefficiëntie van Dynamic Index nog verder kan verbeteren.

- In **Hoofdstuk 3** richten we ons op "item-based collaborative filtering" modellen die zijn geoptimaliseerd door middel van ridge-regessie, aangezien recent werk heeft aangetoond dat ze zeer competitieve resultaten kunnen behalen in vergelijking met meer complexe en minder efficiënte alternatieve methoden. De belangrijkste reden voor deze verbeterde efficiëntie is dat de oplossing voor het regressieprobleem een analytisch berekenbare oplossing heeft. Aangezien deze berekening de inversie van de Gramiaanse item-item matrix inhoudt, is deze alleen geschikt in situaties waarin het aantal items in de catalogus relatief laag is. Inderdaad, aangezien matrix-inversie kubiek schaalt met de dimensie van de matrix, wordt dit al gauw onhandelbaar in hoge dimensies. Wanneer het model iteratief opnieuw moet worden berekend om nieuwe inkomende

data te verwerken, kan deze druk op de rekentijd nog problematischer worden. Om dit probleen aan te pakken, stellen we *Dynamic* EASE^R (DYN-EASE^R) voor, een nieuw algoritme dat op efficiënte wijze een bestaand ridge-regressiemodel bijwerkt wanneer nieuwe data binnenkomen. Vervolgens gebruiken we de Woodbury-matrix-identiteit om de geïnverteerde matrix stapsgewijs bij te werken. We introduceren ook een benaderende variant die de inversie bijwerkt met een lage-rang benadering van de model-update, waarbij de nauwkeurigheid van de oplossing wordt ingeruild voor een winst in rekenefficiëntie. We analyseren onze methoden theoretisch en tonen empirisch een significante efficiëntiewinst aan wanneer we aan collaborative filtering doen in dynamische omgevingen.

- In **Hoofdstuk 4** kijken we naar de vaakst voorkomende offline evaluatieprocedures in de literatuur. Voortkomend uit het onderzoeksveld van machinaal leren, worden aanbevelings-modellen doorgaans getraind op interacties tussen gebruikers en items die beschikbaar zijn in een trainingsset. Ze worden dan geëvalueerd op hun prestaties om te voorspellen welke bijkomende interacties voortkomen in een testset, die willekeurig werd gesampled en uit de trainings-dataset werd gehouden. Bij dit paradigma doen zich twee problemen voor: 1. de sequentiële aard van interacties tussen gebruikers en items wordt volledig genegeerd bij de train-test-splitsing, dus evalueren we modellen in feite op basis van hun vermogen om het verleden te voorspellen op basis van de toekomst, en 2. aanbevelings-systemen die werkzaam zijn tijdens het data-verzamelings-proces worden genegeerd, terwijl ze de blootstelling van gebruikers aan items aanzienlijk kunnen beïnvloeden en leiden tot interacties die *Missing-Not-At-Random* (MNAR) zijn.

  Resultaten die voortvloeien uit dergelijke evaluatieprocedures komen zelden overeen met die van online experimenten – wat aangeeft dat deze problemen een aanzienlijke impact hebben op de resultaten. Om probleem (1) aan te pakken, stellen we een alternatieve evaluatie-procedure voor die veel nauwer aansluit bij de inherente omgevings-aspecten van aanbevelingssystemen op het wereld-wijde web: *Sliding Window Evaluation* (SW-EVAL). Empirische observaties tonen aan dat SW-EVAL minder vatbaar is voor overschatting van modelprestaties, de kracht van populariteits-gebaseerde baselines correct benadert, en in het algemeen concurrerende algoritmen anders rangschikt dan andere veelgebruikte methoden. Door gebruik te maken van data verzameld op een Belgische retailwebsite, tonen we dat (2) de resultaten ernstig beïnvloedt en ze bevooroordeelt ten gunste van systemen die de getoonde aanbevelingen nabootsen. We presenteren een plan van aanpak voor toekomstig onderzoek om deze MNAR-invloed te modelleren en er rekening mee te houden tijdens training en evaluatie.

- In **Hoofdstuk 5** herbekijken we de huidige bestaande offline evaluatieprocedures voor aanbevelingssystemen met impliciete feedback. We bespreken kort offline en online procedures, waarbij we aangeven waar de verschillen en gelijkenissen liggen. Online procedures zijn zeer effectief in het meten van wat we belangrijk vinden, maar zijn vaak erg duur en dus inefficiënt. Offline methoden zijn daarentegen veel efficiënter, maar vaak niet effectief in het

voorspellen van de prestaties in de echte wereld. In dit hoofdstuk bestuderen we waarom ze uiteenlopen en hoe we de offline doelstellingen kunnen afstemmen op online prestaties. We identificeren drie manieren om het huidige werk aan offline evaluatie-methoden te verbeteren en uit te breiden. Ten eerste, zoals we ook hebben benadrukt in Hoofdstuk 4, is het van cruciaal belang om rekening te houden met de sequentialiteit van interacties in de data, en temporele evaluatieprocedures kunnen in dit opzicht vooruitgang boeken. Ten tweede, om de MNAR-invloed op verzamelde data te verminderen, moeten we de getoonde aanbevelingen modelleren om er dan rekening mee te kunnen houden. We trekken parallelen met het probleem van *off-policy* evaluatie in de literatuur over *reinforcement learning*, en linken deze concepten met het aanbevelingsprobleem. Ten derde valt er veel te winnen door meer data te gebruiken dan alleen weergaven en *clicks* om de prestatie van aanbevelingen te evalueren. Inderdaad, aangezien we vaak informatie hebben over aanbevelingen die werden getoond, en ook de *inactie* van de gebruiker, kunnen we deze gebruiken om onderscheid te maken tussen ontbrekende en negatieve feedback. We stellen een onderzoeksagenda voor die gericht is op deze drie onderwerpen, met als doel om offline evaluatie-procedures te bekomen die online prestaties nauwkeurig kunnen weerspiegelen, met het doel om onbevooroordeelde performantie-schattingen te bekomen.

- In **Hoofdstuk 6** onderzoeken we hoe methoden die leren uit getoonde aanbevelingen en hun gevolgen, kunnen worden gebruikt voor aanbevelingstaken. Omdat we niet weten of gebruikers zouden hebben geklikt op items die we niet hebben getoond, kan het rechtstreeks leren van dergelijke data erg complex zijn. Verschillende methoden voor zogenaamd *off-policy* of contrafeitelijk leren zijn de afgelopen jaren voorgesteld voor algemene *machine learning*-taken, maar hun doeltreffendheid als aanbevelings-systeem blijft onderbelicht. We presenteren een overzicht van bestaande methoden in de context van aanbevelingen en voeren een brede empirische studie uit om ze te valideren. We tonen dat bestaande contrafeitelijke leer-methoden falen wanneer positieve uitkomsten van aanbevelingen stochastisch en schaars zijn, en we laten zien hoe een logaritmische ondergrens op de traditionele inverse-propensiteits-schatter deze problemen kan verlichten. Bovendien convexeert de logaritmische transformatie het optimalisatiedoel en vergemakkelijkt het daardoor convergentie. We laten zien hoe de *policy*- en *value*-gebaseerde families van methoden kunnen worden geformuleerd met een identieke parametrisering, en stellen een nieuw model voor dat gezamenlijk beide typen doelstellingen optimaliseert. We laten zien dat deze *Dual Bandit*-aanpak zeer competitief is met de state-of-the-art, vooral in de realistische gevallen waarin de hoeveelheid randomisatie in de getoonde aanbevelingen en de grootte van de trainings-set beperkt zijn, maar het aantal items groot.

- In **Hoofdstuk 7** richten we ons op beloningsmodellen die zijn geleerd uit *bandit* feedback - zoals modellen die de waarschijnlijkheid van een klik op een bepaalde aanbeveling voorspellen. Het nauwkeurig leren van dergelijke modellen is niet eenvoudig wanneer het systeem dat acties kiest op het moment van data-verzameling niet-uniform is. Dat wil zeggen, het toont gretig aanbevelingen die het relevant acht, en verkent slechts zelden alternatieven.

Dit staat bekend als *item selection bias*, en het leidt tot heteroscedastische variantie op de resulterende beloningsschattingen. We laten zien hoe dit nadelig kan zijn voor de aanbevelingsprestaties van een systeem dat steunt op een dergelijk beloningsmodel. Bovendien tonen we hoe dit een probleem verergert dat bekend staat als de *Optimiser's Curse*, waarbij de teleurstelling na de beslissing door het handelen op basis van deze beloningsschattingen in veel realistische gevallen niet-negatief zal zijn. Om deze problemen tegen te gaan, stellen we een algemeen onzekerheids-bewust besluitvormings-beleid voor dat de voorspellingen van het beloningsmodel behandelt met een gezonde dosis scepsis. Bayesiaanse onzekerheidsschattingen kunnen onthullen wat het beloningsmodel niet weet, en we kunnen deze opgedane kennis gebruiken om terug te vallen op de *best worst-case* beslissingen in plaats van de gebruikelijke aanbevelingen die blind zijn voor onzekerheden. Uitgebreide experimenten tonen hoe onze voorgestelde methode kan leiden tot aanzienlijke en robuuste verbeteringen in de prestatie van aanbevelingen.

- In **Hoofdstuk 8** besluiten we door de onderzoeksbijdragen die in dit proefschrift zijn gepresenteerd samen te vatten, en tonen we hoe ze met elkaar verbonden zijn. We presenteren onze visie op en een agenda voor toekomstig onderzoek om de kloof tussen aanbevelingen in de industrie en de academische wereld te overbruggen – gericht op realistische uitbreidingen van het *bandit* leerparadigma die sommige huidige rigide aannames verlichten, zodat ze uiteindelijk een betere weerspiegeling kunnen zijn van de echte wereld.