# Contextual Models for Sequential Recommendation

Zur Erlangung des akademischen Grades eines
Doctor rerum naturalium (Dr. rer. nat.)
genehmigte Dissertation von

**Maryam Tavakol, M.Sc.**

geboren am 28.12.1987 in Meimeh, Iran

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Prof. Dr. Ulf Brefeld

Tag der Einreichung: 21. Februar 2019
Tag der Prüfung: 25. April 2019

Darmstadt–D17

Contextual Models for Sequential Recommendation

genehmigte Dissertation von Maryam Tavakol
Technische Universität Darmstadt, Darmstadt–D17

Tag der Einreichung: 25. April 2019

# Acknowledgements

I would like to express my sincere gratitude to Ulf Brefeld for being unbelievably supportive from the first day I moved to Germany, for all the insightful discussions, critical yet priceless guidance, and providing valuable lessons on academic life beyond research. Thanks for all these years, I enjoyed working with you a lot. I am also profoundly grateful to Johannes Fürnkranz for all the scientific as well as administrative support, for the inspiring discussions, and all the help in writing this thesis. Thanks for making my PhD easier. Besides, I would like to thank the rest of my thesis committee who accepted to be part of the examination process.

I further appreciate the encouragement of all current and previous members of Ulf's group at Leuphana and Darmstadt, and also KE group of Darmstadt that I temporary stayed with them. Thanks to Uwe Dick for our interesting collaborations, and all the others that helped me on discussing ideas, technical problems, or proofreading my thesis. Special thanks to Sebastian Mair and Ahcène Boubekki whom together we shared an office, for all the scientific and non-scientific conversations, all the fun we have had, and offsite memories, thank you guys.

My deepest thanks go to my mom and dad for their unconditional support, without them, I couldn't start the whole journey, I can never thank you enough. I also would like to thank the rest of my family and my friends who were kindly supportive throughout my study and my life in general. Finally, to my dear husband Hamid, thank you for being there for me no matter what, for your enduring love, all the weekly commutes, and of course all the helps on my research.

# Abstract

Recommender systems aim to capture the interests of users in order to provide them with tailored recommendations for items or services they might like. User interests are often unique and depend on many unobservable factors including internal moods or external events. This phenomenon creates a broad range of tasks for recommendation systems that are difficult to address altogether. Nevertheless, analyzing the historical activities of users sheds light on the characteristic traits of individual behaviors in order to enable qualified recommendations.

In this thesis, we deal with the problem of comprehending the interests of users, searching for pertinent items, and ranking them to recommend the most relevant items to the users given different contexts and situations. We focus on recommendation problems in sequential scenarios, where a series of past events influences the future decisions of users. These events are either the developed preferences of users over a long span of time or highly influenced by the zeitgeist and common trends. We are among the first to model recommendation systems in a sequential fashion via exploiting the short-term interests of users in session-based scenarios.

We leverage reinforcement learning techniques to capture underlying short- and long-term user interests in the absence of explicit feedback and develop novel contextual approaches for sequential recommendation systems. These approaches are designed to efficiently learn models for different types of recommendation tasks and are extended to continuous and multi-agent settings. All the proposed methods are empirically studied on large-scale real-world scenarios ranging from e-commerce to sport and demonstrate excellent performance in comparison to baseline approaches.

**Keywords:** recommendation systems, personalization, reinforcement learning, short-term interests, contextual models

# Zusammenfassung

Empfehlungssysteme zielen darauf ab, die Interessen der Benutzer zu erfassen und maßgeschneiderte Empfehlungen zu geben. Die Interessen der Benutzer sind oft einzigartig und hängen von vielen unbeobachtbaren Faktoren ab, z.B. Stimmungen oder externen Ereignissen. Dieses Phänomen schafft ein breites Aufgabenspektrum für Empfehlungssysteme. Alle Aufgabe zusammen sind schwer zu lösen. Durch die Analyse historischer Aktivitäten der Benutzer werden die charakteristischen Merkmale einzelner Verhaltensweisen gefunden, um qualifizierte Empfehlungen zu ermöglichen.

In dieser Arbeit beschäftigen wir uns mit dem Problem, diese Interessen zu verstehen, nach sachdienlich Empfehlungen zu suchen und sie zu ordnen, um den Benutzern die Relevantesten, in verschiedenen Kontexten und Situationen, zu empfehlen. Wir konzentrieren uns auf Empfehlungsprobleme in sequentiellen Szenarien, in denen der Verlauf vergangener Ereignisse die zukünftigen Entscheidungen der Benutzer beeinflusst. Diese Ereignisse sind entweder die entwickelten Vorlieben der Nutzer über einen längeren Zeitraum oder stark vom Zeitgeist und den gängigen Trends beeinflusst. Wir gehören zu den Ersten, die Empfehlungssysteme sequentiell und unter Ausnutzung der kurzfristigen Umstände modellieren.

Wir nutzen Bestarkendes Lernen, um die zugrunde liegenden kurz- und langfristigen Benutzerinteressen ohne explizites Feedback zu erfassen und entwickeln Ansätze für sequentielle Empfehlungssysteme. Diese Ansätze sind darauf ausgelegt, Modelle für verschiedene Arten von Empfehlungsaufgaben effizient zu erlernen, und werden auf Stetig und multi-Agenten Probleme erweitert. Alle vorgeschlagenen Methoden werden empirisch von E-Commerce bis zu Sport untersucht und zeigen im Vergleich zu Baseline-Ansätzen eine hervorragende Leistung.

**Schlüsselwörter:** Empfehlungssysteme, Personalisierung, Bestarkendes Lernens, Kurzzeitinteressen, Kontextmodelle

# Contents

**List of Symbols** **iv**

**List of Figures** **v**

**List of Tables** **vii**

**1 Introduction** **1**

1.1 Sequential Recommendation . . . . . 2

1.2 Contributions . . . . . 3

1.3 Outline . . . . . 5

1.4 Previously Published Work . . . . . 5

**2 Literature Review** **8**

2.1 Recommendation Based on Explicit Feedback . . . . . 8

2.2 Scenarios with Implicit Feedback . . . . . 10

2.3 Context-Aware Methods . . . . . 10

2.4 Sequential Approaches . . . . . 11

2.5 Session-based Recommendation . . . . . 12

**3 Theoretical Background** **14**

3.1 The Recommendation Problem . . . . . 14

3.2 Reinforcement Learning . . . . . 16

3.2.1 Markov Decision Processes . . . . . 17

3.2.2 Algorithms for MDP Problems . . . . . 19

3.2.3 Multi-Armed Bandits . . . . . 21

3.2.4 Deep Reinforcement Learning . . . . . 21

**4 Detecting Topics of User Sessions** **23**
4.1 Motivation . . . 23
4.2 Related Work . . . 25
4.3 The MDP Framework . . . 26
4.3.1 Factorization . . . 28
4.3.2 Optimization via RL . . . 33
4.3.3 Topic Extraction and Recommendation . . . 34
4.4 Empirical Evaluation . . . 35
4.4.1 Topic Detection . . . 36
4.4.2 Topic-driven Recommendations . . . 41
4.4.3 Summary of Results . . . 43
4.5 Conclusion . . . 44

**5 MDP-based Itinerary Recommendation** **45**
5.1 Motivation . . . 45
5.2 Related Work . . . 47
5.3 Modeling User Itineraries . . . 48
5.3.1 Simplified Topic Model . . . 49
5.3.2 Multi-step POI Recommendation . . . 50
5.3.3 Online Personalization . . . 51
5.4 Empirical Study . . . 52
5.4.1 Data Extraction and Analysis . . . 52
5.4.2 Experimental Setup . . . 54
5.4.3 Results and Discussion . . . 56
5.5 Conclusion . . . 58

**6 Unified Contextual Bandit Models** **59**
6.1 Motivation . . . 59
6.2 General Optimization Problem . . . 61
6.2.1 Upper Confidence Bound . . . 63
6.2.2 Instantiations . . . 63
6.3 Unified Short- and Long-term Model . . . 65
6.3.1 The Objective Function . . . 66
6.3.2 Optimization . . . 68
6.4 Extensions . . . 71
6.4.1 Toward more Efficient Models . . . 71
6.4.2 Preference-based Bandits . . . 72

6.5 Empirical Study . . . 73
6.5.1 Overall Performance . . . 73
6.5.2 Cold Start Scenarios . . . 74
6.6 Conclusion . . . 76

**7 Preference-based Personalized Transaction Kernels 78**
7.1 Motivation . . . 78
7.2 Related Work . . . 79
7.3 Transaction Kernels . . . 81
7.4 Informed Sampling for Recommendation . . . 85
7.4.1 Structure of the Search Tree . . . 86
7.4.2 Algorithmic Modifications . . . 87
7.5 Empirical Study . . . 88
7.5.1 Performance of Preference Model . . . 88
7.5.2 Performance of Informed Sampling . . . 91
7.6 Conclusion . . . 94

**8 Extension to Continuous Scenarios 95**
8.1 Motivation . . . 95
8.2 Related Work . . . 97
8.3 Learning to Rate Actions . . . 98
8.3.1 Estimating the Action Model . . . 100
8.3.2 Learning Value Function $\bar{Q}$ . . . 101
8.3.3 Deep Model . . . 103
8.4 Empirical Study . . . 105
8.4.1 Baselines . . . 106
8.4.2 Movement Valuation . . . 107
8.4.3 Movement Model . . . 109
8.5 Conclusion . . . 110

**9 Conclusions 112**

**Bibliography 115**

**Abbreviations 133**

# List of Symbols

**Recommendation Problem:**

| | |
|---|---|
| $m$ | number of users |
| $\mathbb{U}$ | set of users |
| $n$ | number of items |
| $\mathbb{B}$ | set of items |
| $\mathcal{Z}_d$ | random variable encoding the $d$-th attribute of items |
| $\boldsymbol{x}_i$ | vectorized features of item $b_i \in \mathbb{B}$ |
| $\tau_{j,i}^t$ | transaction of user $u_j$ with item $b_i$ at time $t$ |
| $\mathcal{S}_j^t$ | session of user $u_j$ at time $t$ |
| $\boldsymbol{s}_t$ | feature vector of context |
| $T$ | total number of time steps in a sequential trials |
| $k$ | size of history (length of session) |
| $\mathcal{X}$ | set of different data sources |

**Markov Decision Processes:**

| | |
|---|---|
| $\mathbb{S}$ | set of states |
| $\mathbb{A}$ | set of actions |
| $A(s)$ | state dependent continuous action space |
| $\mathcal{P}$ | transition function |
| $\mathcal{R}$ | reward function |
| $\gamma$ | discount factor |
| $\pi$ | policy function |
| $V$ | state value function |
| $Q$ | state-action value function |
| $\mathcal{A}$ | advantage value of an action compared to alternative actions |

# List of Figures


3.1 The main reinforcement learning scenario. . . . . 16

4.1 An exemplary user session of an e-commerce platform which is viewed as a sequences of item attributes to determine the topic of the session. 26
4.2 (a): Transition model of a joint factored MDP. Every attribute value depends on the complete history of all previously viewed items. (b): There are no dependencies between different attributes. . . . . 29
4.3 Variance of topic detection within sessions (fMDP-approx). . . . . 40
4.4 Changing the size of topics for attribute category. . . . . 40
4.5 Impact of topic threshold on the accuracy of prediction (fMDP-approx). 41
4.6 Average ranks for the recommendation task (log-scale). . . . . 42

5.1 Fraction of geo-referenced images in our dataset from Flickr. . . . . 46
5.2 Flickr POIs in Munich, Paris and London. . . . . 53
5.3 Variation in partial path accuracy with regard to $K$ in accuracy@$K$. . 56
5.4 Partial path accuracy for two personalisation techniques. . . . . 56
5.5 Personalized recommendation vs. baselines (partial path accuracy). . 57
5.6 Personalized recommendation vs. baselines (exact path accuracy). . . 57

6.1 Normalized average rank for different data sizes. . . . . 74
6.2 Left: Normalized average rank for different ratios of new items to new users. Right: Execution time for different data sizes. . . . . 75
6.3 Normalized average rank for the data with new users and items. . . . 76

7.1 The structure of search tree in MCTS with five categorical layers for two products: a red straw hat for men and blue soccer shoes for women. 86
7.2 AUC values for SVMs with different kernels vs. baselines. . . . . 89
7.3 AUC values for (a) various click distribution of users, (b) different context popularity. . . . . 90
7.4 Influence of different hash sizes in terms of AUC. . . . . 91

7.5 Performance of MCTS w.r.t. different parameters. . . . . 92
7.6 Performance of MCTS compared to baseline methods. . . . . 92
7.7 Performance evaluation on four users and one product. User 4 is a new user without any training data. . . . . 93

8.1 Learned ratings of potential movements of the circled red player (green dots, the darker the higher the rating). The red circle denotes the true movement that the player is about to perform. . . . . 96
8.2 Input representation to the deep convolutional net. . . . . 103
8.3 Feature representation for an agent: activations of all layers at the position of the agent are concatenated. . . . . 104
8.4 Output nodes and loss. . . . . 105
8.5 Performance in terms of average AUC. . . . . 107
8.6 (a) Performance of movement valuation, and (b) action model (the movements take 3 seconds long into the future). . . . . 108
8.7 Likelihood for different numbers of components in the action model. . 109
8.8 Predicted movements with mixtures using $L = 5$ (left), $L = 15$ (center), and $L = 90$ (right). . . . . 110
8.9 The running agent on the right realizes a very different movement distribution than the slowly moving player on the left. . . . . 110

# List of Tables


4.1 Accuracies for the topic detection on a subset. . . . . 38
4.2 Accuracies for the topic detection using all data. . . . . 39
4.3 Aggregated Average Ranks of recommendation (in hundreds). . . . . 43

5.1 Place types for POIs across cities. . . . . 49
5.2 Variation of partial path accuracy for top-7 closest POIs w.r.t. user history. . . . . 55

7.1 Exemplary feature mappings. . . . . 84

8.1 Per-Layer definition of the convolutional model. . . . . 104
8.2 Players ratings. . . . . 108

# Chapter 1

# Introduction

Recommendation (or recommender) systems are among the most widely used online frameworks in recent years; these aim to serve user needs by providing relevant items to users (Resnick & Varian, 1997). The main examples of these systems can be found in e-commerce, entertainment, social networks, services like travel packages, and online content such as media, where each of them has its own characteristics.

Recommendation systems are very broad and deal with various challenges and problems. These problems range from searching for related items, e.g., movies, products, news articles, friends, etc., in a given context to maximize user satisfaction by providing personalized recommendations. Every user is unique in her preferences, and personalization increases the chances that she keeps interacting with the recommender system. Nevertheless, in addition to user engagement, alternative optimization criteria such as click-through rate, revenue, relevance score and so on are of interest for such systems. Some other challenges relate to the characteristics of the application at hand, such as the size of the user/item pool, its lifetime and dynamicity, scalability of the system, and whether additional contexts (e.g., user or/and item attributes) are available. Moreover, novelty and diversity of the recommendations are important factors to enhance user satisfaction and loyalty.

The primary objective of recommendation systems is to investigate the behavioral patterns and interests of users in order to provide tailored recommendations. One of the main elements in achieving this goal is the feedback that users provide in the recommendation process. User feedback plays a crucial role in learning and optimizing an eligible recommendation model, and falls into two main categories: explicit and implicit. Explicit feedback is the result of users directly expressing their preferences in the form of binary reactions such as "like" or "dislike," ordinal ratings (usually on a scale from 1 to 5), or even tags and textual comments about an item. Implicit feedback is collected without the user providing a direct opinion. Instead, it is inferred

from the user's behavior and interaction with the system, such as a click, adding an item to a shopping cart, or listening to a song. Although implicit feedback is noisy and possibly not very accurate, it does not require the user's direct involvement; ergo it is more convenient to collect and process. Apart from the type of feedback, the latency and volume of the provided sentiment are also important. Delayed reactions impose additional constraints on the optimization problem, and a low volume of feedback leads to sparsity issues.

All these constraints and challenges create complex tasks and problems to be addressed in various applications. Therefore, designing a good recommendation system is more of a work of art with detailed requirements. In this thesis, we contribute to the field of recommendation by studying several facets of recommendation systems with a focus on sequential and contextual scenarios.

## 1.1 Sequential Recommendation

User preferences come from latent and sometimes unknown patterns where samples of those patterns can be revealed at the time of a particular action, e.g., browsing, rating, leaving a comment, and so on. Analyzing the previous choices of a user facilitates understanding the motivation behind the user's behavior; thus, the past activities of users are the main source of predicting their future decisions. For example, if a user is frequently buying garments of a specific brand or color, it is more likely that the user keeps this habit for his further purchases.

We are among the first to view the recommendation process as a sequential problem and design a sequential framework for a real-world recommendation system. Intuitively, user choices are sequential in nature. Certain decisions depend on the previous ones; for instance, after buying a smartphone, one might purchase a phone case or earphones. From another point of view, in addition to the sequential essence of user choices, this concept highlights the importance of recent activities in recommendation systems. Short-term events or interest of users influence their decisions as well. In certain situations, short-term interests are even more vital than long-term preferences. Assume a news article recommendation engine that serves a set of users with various interests, e.g., sport, politics, technology, etc. A temporarily significant event, such as the soccer World Cup, could change the course of news viewing for most people with different interests.

On the other hand, the preferences of users develop over a long span of time: a teenager becomes an adult and might change her clothing style from casual to classy.

Besides, people tend to experience and discover new things from other resources, such as their social group, which affects their decisions. However, traditional recommendation systems usually encode the complete history of individuals as a bag of events that neither is sequential nor considers short-term preferences.

In this thesis, we intend to model the recommendation problem in a sequential fashion to understand the short- and long-term temporal patterns within the decision making process of users. Nevertheless, sequential learning tasks usually encounter the exploitation-exploration dilemma. A system is required to balance exploitation, which in our case means to recommend an item that led to the best results in the past, and exploration, to recommend miscellaneous content to help users discover and expand new interests whilst the model updates accordingly. Reinforcement Learning (RL) algorithms provide suitable mechanisms to trade-off exploitation versus exploration. Therefore, we propose approaches based on reinforcement learning to address sequential recommendation problems and incorporating short-term interests of users into personalized recommendations.

## 1.2 Contributions

The key research question in this thesis is how to model recommendation systems as sequential decision making processes, which naturally capture the behavioral patterns of users over short- as well as long-term spans of time. We design novel contextual models, mainly using reinforcement learning techniques, to unravel underlying short- and long-term user interests in the absence of explicit feedback. The main contributions of this thesis are as follows.

To the best of our knowledge, we are one of the first to present a sequential framework using Markov Decision Processes (MDPs) to model recommendation systems in short-term scenarios, i.e., user sessions. We demonstrate that the recent activities of users are important factors to clarify the true intention of users or the topic of the current session. We thus take a contextual session-based approach and further factorize the resulting MDPs over attributes of items to detect the user's goal (the topic) of a session. We show that an independence assumption on the attributes of items leads to a set of independent models that can be optimized efficiently. Our approach results in interpretable topics that can be effectively turned into recommendations. Empirical results on a real-world click log from a large e-commerce company exhibit highly accurate topic prediction rates. Translating our approach into a topic-driven recommendation system outperforms baseline competitors.

Furthermore, we use a simplified version of our MDP-based approach to the task of recommending personalized itineraries. We leverage social media, more explicitly photo uploads on Flickr and their tags, to reverse engineer historic user trips. Our solution is based on MDPs to detect the main category of the next Point of Interest (POI). This is a special case of our topic model without factorization. The tags attached to the photos provide the elements to generate possible configurations and prove crucial for contextualizing the proposed approach. We further personalize our model by exploiting the individual preferences from the long-term history of users. In the experiments, we observe that the predicted itineraries are more accurate than standard path-planning algorithms.

Following the success of RL-based methods in the session-based recommendation, we incorporate the long-term interests of users into the same model. We hence relax the Markov property of the MDP and introduce a unified contextual bandit framework for recommendation problems to capture both short- and long-term interests of users. The model is devised in the dual space and the derivation is consequentially carried out using Fenchel–Legendre conjugates of loss functions. It thus leverages to a wide range of tasks and settings. Moreover, we represent several extensions and special cases of our approach that result in a general and unified framework for different applications. The empirical study demonstrates significant performance in various settings and scenarios, particularly in cold start problems.

Additionally, we propose a combined short-term and personalized model by focusing on the pair-wise preference data. We first exploit theories from tensor products to capture the contextual transactions of a user in a joint feature space by means of tensor kernels. The representation is extended to all users via hash functions that allow effectively storing and retrieving personalized slices of data and context. These personalized transaction kernels are employed in Support Vector Machines (SVMs) to learn the underlying preference models, which are further used in Monte Carlo Tree Search (MCTS) for efficient sequential recommendations. Empirically, on a real-world transaction dataset, both the preference models as well as the search tree exhibit excellent performance with respect to baseline approaches, in particular in cases where only a small number of products can be sampled.

Finally, we extend the recommendation problems to continuous and multi-agent scenarios for sport analytics, using the example of soccer. We study sequences of trajectory data from players in soccer to evaluate their fine-grained movements, depending on different game situations. The movements of individual players are rated with respect to their potential for staging a successful attack. We propose a purely

data-driven approach to simultaneously learn a model of agent movements as well as their ratings via an agent-centric deep reinforcement learning framework. Our model allows for efficient learning and sampling of ratings in the continuous action space, which can be used to recommend optimal movements. We empirically observe using historic soccer data that the model accurately rates agent movements with respect to their relative contribution to the collective desired outcome, which serves as a proxy for assessing the quality of the recommendations.

## 1.3 Outline

The thesis is structured as follows. In Chapter 2, we provide an overview of related work on recommendation systems and further zoom into the challenges and approaches that are of interest in this thesis. Chapter 3 summarizes the main theoretical background of the methods used in this thesis and gives a formal problem setting. In Chapter 4, we present an MDP-based approach for modeling topics of user sessions that copes with short-term recommendation scenarios. We describe a simplified case of our approach for itinerary recommendations in Chapter 5 with a taste of personalization. Chapter 6 combines personalization with the short-term view in one model and yields a general optimization framework using a contextual bandit setting. We model the recommendation systems as preference learning frameworks based on personalized transaction kernels in Chapter 7. In Chapter 8, we model trajectory data from soccer players as a recommendation problem and present a deep RL approach to address the recommendation scenarios in a continuous space. Chapter 9 provides the conclusions of the thesis.

## 1.4 Previously Published Work

Some parts of this thesis have already been published in articles, and some of them emerged from collaborations with colleagues. The following list enumerates these articles and gives a brief summary of my respective contributions to the papers.

**Factored MDPs for Detecting the Topic of User Sessions** by Maryam Tavakol and Ulf Brefeld, *Proceedings of the ACM Conference on Recommender Systems*, 2014. The paper originates from my ideas to identify the important factors in the recent browsing history of users. I exploited my knowledge from RL to model the topic of user sessions via MDP and further derived a factorized model to address the problem

for large scale settings. In addition to developing the idea, I conducted the experiments and carried out all the implementations.

**A Unified Contextual Bandit Framework for Long- and Short-term Recommendations** by Maryam Tavakol and Ulf Brefeld, *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2017.
I proposed incorporating a personalized component into a contextual bandit model for short-term recommendation. I further devised a general optimization approach in the dual space using Fenchel–Legrende conjugates of loss functions and provided special cases and possible extensions. I programmed the empirical studies for different experimental setups and evaluated various scenarios.

**MDP-based Itinerary Recommendation using Geo-Tagged Social Media** by Radhika Gaonkar, Maryam Tavakol, and Ulf Brefeld, *International Symposium on Intelligent Data Analysis*, 2018.
I supervised Radhika Gaonkar on her internship, which turned into a bachelor thesis on recommending travel itineraries. We continued to collaborate on the project and further work led to an article. We developed the ideas together and she performed the experiments including collecting and processing the data from Flickr. The base implementation was the MDP framework from my first publication, and I helped her to adapt its theoretical and practical framework to the new problem setting.

**Personalized Transaction Kernels for Recommendation using MCTS** by Maryam Tavakol, Tobias Joppen, Ulf Brefeld, and Johannes Fürnkranz, *ACM Conference On User Modelling, Adaptation And Personalization (under review).*
Together with Tobias Joppen, I built a model in the preference-based settings that leverages MCTS for recommendation problems. I developed the personalized kernel functions for the first part of the paper and Tobias Joppen came up with a method to use MCTS in the feature space of products. I did the implementation and completed the experiments from the preference models, while Tobias Joppen conducted the empirical study of the search strategies.

**Rating Continuous Actions in Spatial Multi-Agent Problems** by Uwe Dick, Maryam Tavakol and Ulf Brefeld, *Machine Learning Journal (under review).*
Uwe Dick had the original idea of employing deep RL to analyze players trajectory data in soccer, and I had the idea of turning this into a recommendation system. We

developed the approach together to be able to sample and evaluate individual actions and perform recommendations. Uwe Dick implemented the deep learning architecture and trained the model via RL optimization, while I programmed the evaluation part for recommendation and added the baselines used in the paper.

# Chapter 2

# Literature Review

Recommender systems have become very popular in the past twenty years and have thus received considerable attention from the machine learning community. Machine learning techniques provide promising means with which to address recommendation tasks by modeling them as optimization problems while taking their corresponding constraints and limitations into account. A large number of methods have been proposed for making accurate recommendations in various settings and applications. In this chapter, we review important related works and present the major advancements and shortcomings of the state-of-the-art recommendation approaches.

## 2.1 Recommendation Based on Explicit Feedback

One of the starting points for research on recommendation systems goes back to using traditional data mining techniques, such as association rule mining (Agrawal et al., 1993), which generates all significant association rules between items in a database. Subsequently, Agrawal & Srikant (1994) proposed the Apriori algorithm for efficiently generating frequent-item sets, which are then used further for cart analysis in recommendation scenarios. However, the topic became more popular by introducing Collaborative Filtering (CF) approaches in mid-90s (Hill et al., 1995; Resnick et al., 1994). Recommendation strategies are originally categorized into content-based filtering and collaborative filtering strategies. Early content-based methods focus on the similarity of item features via cosine similarity (Mooney & Roy, 2000) or textual similarity, which uses the minimum description length of texts (Lang, 1995) to recommend items that are contextually analogous to the previously preferred items. Nevertheless, collaborative filtering approaches, which recommend items that users with similar preferences liked in the past, have been shown to be more successful (Resnick et al., 1994).

The core idea in CF methods is to construct a user-item matrix from the explicit ratings of users for different items. The generated matrix is used to first find the users who share a similar rating pattern, then to predict the missing ratings of one user from the ratings provided by those like-minded users. There are two types of collaborative filtering approaches: memory-based and model-based methods. The memory-based approaches utilize the ratings from the user-item matrix to compute the similarities between users and items. The neighborhood methods, which are memory-based, are used to predict the missing ratings and are either user-based (Sarwar et al., 2000a; Shardanand & Maes, 1995) or item-based (Karypis, 2001; Linden et al., 2003; Sarwar et al., 2001). Perugini et al. (2004) provide a survey on the traditional recommendation approaches using neighborhood techniques.

In contrast to neighborhood methods, model-based approaches employ the user-item ratings to train a model that later predicts the missing values (Billsus & Pazzani, 1998; Goldberg et al., 2001). A very popular model-based approach is matrix factorization, the winner of the Netflix prize for predicting movie ratings (Koren et al., 2009). Matrix factorization represents users and items in a lower-dimensional latent space. It provides personalized recommendations by decomposing the user-item matrix into two smaller ones, while learning the latent factors of users and items.

Lately, deep learning algorithms have demonstrated remarkable performance in different areas of machine learning, including recommendation systems. In the context of collaborative filtering, deep learning has been used in combination with matrix factorization as an optimization method (Xue et al., 2017) and also as standalone model-based technique. Covington et al. (2016) propose an approach for developing YouTube video recommendations by creating embedding of users and items via a deep architecture. The generated deep model is then employed for both candidate generation and ranking. In addition, Long Short Term Memory (LSTM) networks have been developed to capture long-term temporal changes in both users and items to predict unavailable ratings (Wu et al., 2017). Furthermore, time intervals between neighbor actions are modeled by time gates in Time-LSTM (Zhu et al., 2017). Zhang et al. (2017) provide a survey of deep learning and CF-based recommendation systems.

All of the above approaches focus on the recommendation scenarios with explicit feedback, and are mostly designed to capture the long-term preferences of users, particularly in collaborative filtering techniques. Nonetheless, in many applications, recommendation systems deal with implicit feedback, since explicit feedback is impossible to collect. Moreover, modeling the temporal changes of users and items as well as taking the short-term interests of users into account are important and challenging

problems in recommendation systems. In the remainder of this chapter, we will focus on the approaches that address these challenges.

## 2.2 Scenarios with Implicit Feedback

The first step toward recommending from implicit feedback is to adjust the collaborative filtering methods to deal with implicit behaviors. Hu et al. (2008) construct the user-item matrix from pre-specified weights instead of explicit user ratings. The weights come from the frequency of clicks of users, meaning that the more an item has been clicked, the more the user is interested in that item. The entries with no information (weight) receive a constant value, which leads to a complete matrix. An alternative least square algorithm is employed to factorize the matrix. Furthermore, Takács & Tikk (2012) alter the objective function from rating prediction to ranking prediction and theoretically derive a more efficient update rule for the objective function of the alternative least square algorithm.

In the one-class collaborative filtering approach proposed by Pan et al. (2008), the missing values are not discarded from the low-rank approximation as well. Instead, the method samples negative examples from the missing values using various sampling techniques to balance the optimization problem. Additionally, Yi et al. (2014) use the time spent on an item as another indicator of interest and propose a method for computing the dwelling time for recommendation purposes.

In addition to collaborative filtering-based approaches, probabilistic models are utilized for recommendation based on implicit feedback. Rendle et al. (2009) deploy a maximum aposteriori estimation for Bayesian personalized ranking. In other work (Das et al., 2007), probabilistic latent semantic indexing and the click history of the community are used for min-hash clustering of users for personalized Google news recommendations. Moreover, Sarwar et al. (2000b) present a method for reducing the matrix dimension via latent semantic indexing, which mainly addresses the data sparsity problem in collaborative filtering techniques.

## 2.3 Context-Aware Methods

Context-aware approaches have emerged for coping with cold start situations. In many recommendation scenarios, the sets of users and items are highly dynamic, and standard methods, which learn a model for existing ones, can not be used for new users and items. Rendle & Schmidt-Thieme (2008) propose an online updating approach

for matrix factorization models for new users and items and show that using the same gradient descent step in the learning process only for new ratings can be as good as a complete update. However, incorporating context into matrix factorization methods has proved to be a more successful approach to overcoming cold start problems.

Karatzoglou et al. (2010) introduce a hybrid recommendation method in which context is added as a new dimension to the existing 2-D matrix of collaborative filtering methods in order to build a tensor factorization model. Similarly, factorization machines (Rendle, 2010) use any kind of feature vectors and contexts to provide a general predictor for various tasks and settings. Rendle et al. (2011) additionally present a faster version of factorization machines in which the model equations are computed in linear time (in terms of the number of contexts and factors). Furthermore, a novel gradient-boosting factorization machine model has been designed to incorporate feature selection algorithms and factorization machines in a unified framework (Cheng et al., 2014). Moreover, Hidasi & Tikk (2012) present methods for more efficient tensor factorization.

In the class of probabilistic methods, Ansari et al. (2000) construct a hierarchical approach to employ information from user features, item features, expert evaluations and interactions between users and items in a Markov chain Monte Carlo method used for estimating the ratings. Agarwal & Chen (2009) propose a regression-based latent factor model as a two-stage hierarchical model for binary-relational data. The first part of the model employs maximum likelihood to estimate model parameters from the observations. The second part consists of a generative model for the latent factors. In another generative model (Maneeroj & Takasu, 2009), variational Bayesian expectation maximization algorithm is used to estimate the parameters and cope with multi-attributed records. Collaborative topic modeling, which was introduced by Wang & Blei (2011), learns the user topic by integrating Latent Dirichlet Allocation (LDA) (Blei et al., 2003) into probabilistic matrix factorization.

Cold start scenarios are addressed via SVM-based methods (Oku et al., 2006) and multi-armed bandits (Caron & Bhagat, 2013), as well. The handbook of *Context-Aware Recommender Systems* (Adomavicius & Tuzhilin, 2011) gives a more complete review of context-aware methods for recommendation systems.

## 2.4 Sequential Approaches

Sequential recommendation systems are poorly explored and insufficiently studied in real scenarios. Hence, we introduce contextual and sequential approaches, particularly

for capturing the short-term interests of users in large-scale applications (Tavakol & Brefeld, 2014) (see Chapter 4). Nevertheless, Zimdars et al. (2001) take the order of ratings into account by transforming input data into sequential form. Rendle et al. (2010) study first-order Markov chains with matrix factorization for basket recommendations, which results in a transition cube over sequential data and models the sequential behavior of users. Temporal effects are further utilized to predict the probability of a user purchasing a product at a particular time via variational Bayesian methods (Wang & Zhang, 2013). In their work, the joint probability of time and product is computed using a combination of proportional hazards and a logistic regression model.

Markov Decision Processes (MDP) are used frequently for sequential decision making under uncertainty (Puterman, 2014; Sutton & Barto, 1998). Shani et al. (2005) introduce MDPs for sequential recommendations, in which the state encodes the last $k$ items viewed by the user and a tabular reinforcement learning algorithm learns the value of state-action pairs. Additionally, sequential patterns in user activities are captured via a mixture of Markov models to cluster users according to their behavior because it is infeasible to have separate models for each user (Liu et al., 2007). A Q-learning-based approach for recommendation systems is presented by Taghipour et al. (2007). Moreover, Karatzoglou (2011) combines temporal and collaborative aspects of recommendation systems by minimizing regularized loss functions.

Furthermore, bandit-based approaches are used to automatically improve recommendations on the basis of user feedback and to conduct trade-offs between exploration and exploitation (Ten Hagen et al., 2003). Radlinski et al. (2008) deploy a multi-armed bandit for diverse recommendations of documents in ranking problems. Their method displays diverse items with regards to a given query containing at least one document which is relevant to the user. In addition, contextual bandits are introduced by Li et al. (2010) for recommendation systems which are extended by Chu et al. (2011), who propose a polynomial-time algorithm for bandits with a linear payoff function. The contextual bandits are further modeled using Gaussian processes for the top-K recommendation tasks (Vanchinathan et al., 2014).

## 2.5 Session-based Recommendation

Once again, we are among the pioneers in the field to exploit the short-term interests of users for session-based recommendations. In this thesis, we describe novel contextual approaches that are suited for short-term scenarios, as well. The existing literature

on session-based recommendation systems is fairly newer than our first paper and mostly based on deep learning.

Recurrent Neural Networks (RNNs) have been recently employed in sequential session-based recommendations. Devooght & Bersini (2017) suggest a categorical cross-entropy objective function to recurrent neural networks for a sparse session-based rating task. In other work, Song et al. (2016) have the system first learn an embedding of users and items and then model the temporal changes in the clicking behaviors of users. Hidasi et al. (2015) present a context-free approach for ranking every available item. In their method, gated recurrent units are used to model variable-length sequence data, which are then updated via in-sequence mini batches. An extension for improving RNN-based short-term recommendation consists of augmenting the data with the time spent on the items (Dallmann et al., 2017). In addition, Hidasi et al. (2016) investigate a contextual version of session-based recommendation, in which a parallel architecture combines item identifiers with their features to score every item (or a subset of them). In another contextual model, the likelihood of a sequence is conditioned on the context, and the next item is predicted using a contextual recurrent neural network (Smirnova & Vasile, 2017).

Session-based recommendation is combined further with personalization, in which the entire click history of every user is also taken into account (Wu et al., 2016b). In the work of Twardowski (2016), the embedding of users and items is obtained from the user-item matrix and applied in session-aware recommendation systems. Moreover, a hierarchical structure is proposed by Quadrana et al. (2017) which retains two gated recurrent units, one for the users and one for the sessions. In their framework, the personalized model is updated after every session of user and later initializes the subsequent session.

# Chapter 3

# Theoretical Background

The purpose of this chapter is to establish a sound foundation for this thesis and describe the general concepts employed in our theoretical contributions. Section 3.1 concentrates on recommendation systems and formally defines the recommendation problem for sequential settings. In Section 3.2, we look into reinforcement learning, which is the technique that our approaches are mainly based on. We further cover fundamental concepts and present commonly used notations, terminology, and algorithms for reinforcement learning problems.

## 3.1 The Recommendation Problem

This thesis deals with recommending items of interest to users while taking the sequential and implicit history of user activities into account. Therefore, we define the recommendation problem in the sequential setting, which is different from the setting imposed by the standard unordered and explicit matrix form. We formulate the recommendation process as an optimization problem which takes the sequence of the past activities of every user to predict the most relevant items for each user's next-step recommendation. We describe the problem setting and general notation used throughout the thesis.

The sequential recommendation problem is defined for a set of $n$ available items, $\mathbb{B} = \{b_1, b_2, \ldots, b_n\}$, and $m$ users, $\mathbb{U} = \{u_1, u_2, \ldots, u_m\}$. We use subscript $i$ to address items and subscript $j$ for users when encountering both entities together throughout the thesis. Consider that items are completely characterized by their attributes, i.e., every item $b_i \in \mathbb{B}$ is described by a set of attributes given by a feature vector $\boldsymbol{x}_i \in \mathbb{R}^{d_x \times 1}$ of size $d_x$. The users, on the other hand, lack attributes in our setting and are therefore denoted by their unique identifier, $u_j \in \mathbb{N}$.

The history of the users' activities contains their previous actions, such as explicit ratings, clicks, purchases, adding items to wish lists, and so forth, performed on different items in discrete time steps and are called *transactions*. From a total of $T$ transactions, let $T_j$ be the total number of transactions completed by user $u_j$. Analogously, $T_i$ is the number of transactions conducted on item $b_i$. A transaction $\tau_{j,i}^t = (u_j, b_i, t)$ is therefore a triple of user, item, and time, which indicates that user $u_j$ had an interaction with item $b_i$ at time step $t$. Further, we specify user sessions: a session of size $k$ of user $u_j$ at time $t$ is determined by a sequence of the user's $k$ previous transactions in the time interval,

$$\mathcal{S}_j^{t;k} = [\tau_{j,i}^{t'}], \text{ where } i \in \{1, \ldots, n\} \text{ and } t' = [t-k, \ldots, t-2, t-1].$$

Consequently, the history of all of the transactions carried out by user $u_j$ up to time $t$ (excluding $t$) is displayed by $\mathcal{S}_j^t$, where $k = T_j$. Moreover, additional information at time $t$ as well as recent events that influence the user's decision provide the so-called *context* for the model. Examples of context include the content of the last click, demographic characteristics of users, the topic of an ongoing session, weather conditions, the search query, and so on. We conceptually consider the context in our models as equivalent to the state in MDPs (see Section 3.2.1), shown by $s_t$ indicating the state or context at time $t$. Nevertheless, in some chapters we use a more specific definition for the context depending on the application at hand.

Therefore, given the transaction history of all of the users $\{\mathcal{S}_1^t, \ldots, \mathcal{S}_m^t\}$, the feature vector of items $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, and the actual context $s_t$, the goal is to model the sequential recommendation process as a *learning to rank* problem using offline training data. The reader should bear in mind that recommendation problems are ideally learned and evaluated in online scenarios, such as A/B testing. However, due to practical constraints, we are unable to conduct such experiments for our research. The learning task is hence defined as finding a function $f : \mathbb{U} \times \mathbb{B} \times T \to \mathbb{R}$ which provides a relevance score of every item to each user at time $t$,

$$y_{j,i}^t = f(j, i, t | \mathcal{S}_j^t, \boldsymbol{x}_i, s_t).$$

The obtained scores are sorted further in descending order to create a ranked list, which is used to optimize the following objective function

$$\min_f \quad \forall j, t \quad \mathcal{L}\Big(sort\big(y_{j,1}^t, \ldots, y_{j,n}^t\big), l \in \tau_{j,l}^t\Big),$$

where $\mathcal{L}$ computes the loss of the ranking prediction (depending on the ranking measure) given the scores from $f$. Note that the scores obtained from $f$ could be additionally translated into a probability distribution to estimate the probability of the

next item. In the next chapters, we describe several approaches that address the whole or part of the above optimization problem.

## 3.2 Reinforcement Learning

Reinforcement Learning (RL) is a learning approach that is conducted via trial and error while interacting with an environment in which there is no explicit teacher supervising (Sutton & Barto, 1998). The essential concepts in RL are the **agent**, the **action**, the **environment**, the **state**, and the (delayed) **reward**. Figure 3.1 shows the overall procedure used in RL problems. The agent interacts with the environment and learns how to behave in order to maximize its accumulated reward. In every state of the environment, the agent performs an action while presuming that the action leads to a positive feedback. The performed action receives a reward from the environment, which is either immediate or delayed. The objective is that the agent fulfills enough trials such that later the agent will be able to estimate the consequence of its actions in every state in order to maximize the total reward obtained.

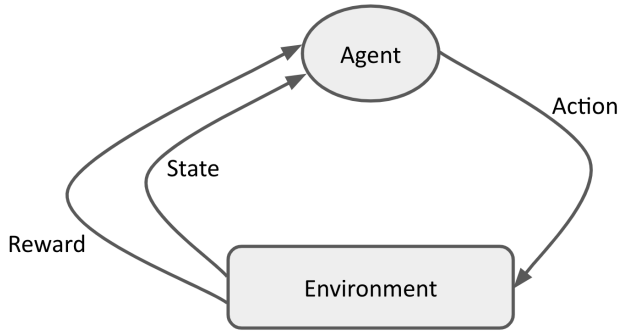



Figure 3.1: The main reinforcement learning scenario.

A popular example of an RL setting is the game of chess. The arrangement of pieces on the game board forms a state of the environment. The agent plays to win the game. Hence, the delayed reward is revealed only at the end of the game and is characterized as a win, loss, or draw. At each step, the player moves a piece, which leads to a change in the state with no immediate outcome. Nevertheless, the reward at the terminal state is used to evaluate the actions of the player during the game. For instance, after executing a sequence of moves which result in losing the game, the agent would subsequently try other actions in similar situations in order to change the outcome. After sufficient number of playing games, the player would be optimally able to estimate the eventual outcome of each action in every state.

Reinforcement learning is a framework for sequential decision making problems which implicitly trades-off exploration versus exploitation. Additionally, the learning process is neither supervised nor unsupervised, and the agent only updates its internal model from the partial feedback that it receives upon performing an action. Therefore, the problem setting of RL provides a perfect basis for the sequential recommendation problem described in Section 3.1. In the remainder of this section, we introduce the technical background of reinforcement learning approaches.

### 3.2.1 Markov Decision Processes

Reinforcement learning problems are mathematically described using Markov Decision Processes (MDPs) (Sutton & Barto, 1998). MDPs (Bertsekas & Shreve, 2004; Puterman, 2014) provide a strong framework for solving sequential stochastic decision problems for which the Markov property holds. The Markov property refers to memorylessness, meaning that the next state only depends on the current state and action and is independent of any other previous state in the history.

An MDP is defined via a five-tuple $(\mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma)$ representing the set of states, the set of actions, the transition function, the reward function, and the discount factor over $T$ trails (or time steps), which are specified as follows.

- **States:** A state $s_t \in \mathbb{S}$ consists of all relevant information about the environment at time $t$ as perceived by the agent for future decisions.

- **Actions:** $\mathbb{A}$ is the set of possible actions that the agent can take at every state to interact with the environment. An action $a_t \in \mathbb{A}$ changes the state of the environment from $s_t$ to $s_{t+1}$. Actions usually have stochastic effects due to the nondeterministic properties of the world.

- **Transition function:** The transition function $\mathcal{P} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$ models the stochasticity of the environment by characterizing a probability distribution over the next state given the current state and action, $\mathcal{P}(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$, where $s, s' \in \mathbb{S}$ and $a \in \mathbb{A}$.

- **Reward function:** The reward function, $\mathcal{R} : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$, determines the expected immediate reward associated with taking a particular action, $\mathcal{R}(s, a) = \mathbb{E}\{r_t | s_t = s, a_t = a\}$.

- **Discount factor:** The discount factor $0 \leq \gamma < 1$ discounts the worth of a future (delayed) reward for the current state.

The main goal of an MDP is to find a policy that maximizes the total expected reward (Sutton & Barto, 1998). One way to learn the optimal policy is to compute the value of each state and select the actions that transit to the states with the highest values. A **state value function** $V : \mathbb{S} \to \mathbb{R}$ specifies how good the states are with respect to a given policy and is defined as the expected return starting from the current state $s$

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left\{r_t | s_t = s\right\} = \mathbb{E}_{\pi}\left\{\sum_{k=0}^{T} \gamma^k r_{t+k} | s_t = s\right\},$$

where $\pi : \mathbb{S} \to \mathbb{A}$ is the policy of the agent in terms of choosing the next action in each state. Furthermore, the **state-action value function** $Q : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is defined as the expected return from state $s$ after taking action $a$ following the policy $\pi$

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left\{r_t | s_t = s, a_t = a\right\} = \mathbb{E}_{\pi}\left\{\sum_{k=0}^{T} \gamma^k r_{t+k} | s_t = s, a_t = a\right\},$$

which gives the values of different actions in a state. Both state value and state-action value functions can be estimated from experiences. Note that learning only the $V$ function is a value prediction problem, while action values are used in control problems (Szepesvári, 2010).

An intriguing property of value functions in reinforcement learning is a particular recursive relationship between consecutive states. This property is demonstrated using the Bellman equation

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \left[\mathcal{R}(s, a) + \gamma \sum_{s'} \mathcal{P}(s, a, s') V^{\pi}(s')\right], \tag{3.1}$$

which shows that the value of a state is obtained from the discounted values of possible next states. The goal of an MDP thus becomes to solve the Bellman equation in order to find the optimal policy $\pi^*$ which maximizes the value function such that $V^*(s) = \max_{\pi} V^{\pi}(s)$. Similarly, the optimal state-action value function is $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$. Plugging the optimal policy into the Bellman equation expresses that the value of a state under optimal policy is equivalent to the expected return of the best action

$$V^*(s) = \max_{a} \left[\mathcal{R}(s, a) + \gamma \sum_{s'} \mathcal{P}(s, a, s') V^*(s')\right].$$

Similarly, the Bellman optimality equation for $Q^*$ gives

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s'} \mathcal{P}(s, a, s') \max_{a'} Q^*(s', a').$$

### 3.2.2 Algorithms for MDP Problems

There are three major classes of methods for learning an MDP problem: model-based, model-free, and a combination of both (Szepesvári, 2010). The model-based methods can be applied when the model of the environment, consisting of the transition and reward functions, is available. Therefore, the model can be used in the Bellman equation, and the MDP solves via dynamic programming. In contrast, model-free approaches sample experiences directly from the environment using Monte Carlo methods. Moreover, temporal difference learning benefits from both model-based and sampling approaches. In this section, we describe each of them briefly.

**Dynamic Programming.** Dynamic programming algorithms compute the optimal policy of an MDP given the complete model of the environment, $\mathcal{P}$ and $\mathcal{R}$, for a policy $\pi$ (see Equation (3.1)). The procedure consists of two steps: policy evaluation and policy improvement. In the first step, the algorithm estimates the value function for an initial policy (policy evaluation), which is then updated based on the values obtained in the next step (policy improvement). The term "*d*ynamic programming" refers to the cached values of $V(s')$ from the previous step which are used to predict the values in the next iteration

$$V_{\iota+1}(s) = \sum_{a} \pi(s,a) \left[ \mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{P}(s,a,s') V_{\iota}(s') \right],$$

where $\iota$ is the iteration index. In the next step, the values obtained from the last iteration are used to find a policy which chooses the action that leads to a state with the highest value. Therefore, the algorithm follows a greedy strategy with respect to the value function of the original policy, therefore, it is called "policy improvement."

**Policy iteration** and **value iteration** are the most common algorithms for dynamic programming. For each repetition of the policy iteration algorithm, the values computed with respect to a given policy $\pi$ are used to generate a better policy, and the value function in the next iteration is obtained from the new policy and so on. In value iteration, the value function is updated directly from the best previous values without maintaining an intermediate policy. Hence, the update rule for value iteration becomes

$$V_{\iota+1}(s) = \max_{a} \left[ \mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{P}(s,a,s') V_{\iota}(s') \right]. \tag{3.2}$$

**Monte Carlo Methods.** Unlike model-based approaches, model-free methods require no prior knowledge of the environment, just sampled sequences of states, actions, and rewards from actual or simulated interactions with an environment. Therefore, Monte Carlo methods solve the underlying MDP by computing the expected returns from episodes of sampled experiences. However, the sampling policy can be different from the improving policy, which leads to two different Monte Carlo controls. If the method used consists of sampling and updating the same policy, it is called **on-policy**: the policy needs to be non-greedy to ensure enough exploration in the sampling process. In contrast, for **off-policy** methods, the policy used for sampling is different from the improving policy, and the sampling policy deals with exploration.

**Temporal Difference Learning.** Temporal Difference (TD) learning is a combination of Monte Carlo sampling and dynamic programming that learns from raw samples using Monte Carlo ideas while bootstrapping the current estimated values from dynamic programming. The general case is defined by a parameter $\lambda$ that balances the two components and is called TD($\lambda$). We introduce the simplest instance, or TD(0), which uses one sample and the estimated value of the next state. The temporal difference error (TD error) is therefore determined as the difference of the current value of a state and its prediction which later is employed to update the value function

$$V(s_t) \leftarrow V(s_t) + \xi[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)],$$

where $\xi$ is the learning rate. In the above equation, the predicted value $V(s_t)$ is $r_{t+1} + \gamma V(s_{t+1})$ in which the first term is the reward of a drawn sample and the second term is the value of the next state.

Two of the most popular instances of TD control are SARSA (Sutton & Barto, 1998) and Q-Learning (Watkins & Dayan, 1992), which learn the state-action values via on-policy and off-policy methods, respectively. The updating rule in SARSA is as follows

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \xi[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \tag{3.3}$$

and for Q-Learning, it is

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \xi[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \tag{3.4}$$

### 3.2.3 Multi-Armed Bandits

Multi-armed bandit methods aim to estimate the distribution of outcomes obtained from multiple machines (arms) in a stochastic sequential decision making process (Auer et al., 2002; Bubeck & Cesa-Bianchi, 2012). The standard $k$-armed bandit originates from the context-free sequential analysis problems of Lai & Robbins (1985), where, at every step of arm selection, the agent needs to follow a strategy leading to the highest reward. The agent interacts with an unknown environment in which the arms have their own reward distributions, thus receiving a reward signal for each trial and getting closer to the average estimation of the payoffs for various arms.

There are several policies for arm selection which aim to optimally trade-off exploration versus exploitation. Simple methods, such as following greedy or $\epsilon$-greedy policies, are not suitable for the exploration-exploitation dilemma, as the former only exploits and the latter forever explores. An improved version is provided by the softmax approach, which limits the exploration for low-value arms. Another well-known method for addressing the exploration-exploitation dilemma in a Bayesian way is Thompson Sampling (Thompson, 1933). Auer (2003) proposes a successful policy called UCB (Upper Confidence Bound), in which the exploration is proportional to the confidence bound of the estimation and has been shown to result in a tight-bounded regret.

Furthermore, multi-armed bandits are utilized in contextual settings (Langford & Zhang, 2008; Li et al., 2010; Tang et al., 2014) in which the arm selection strategy depends on the context. Contextual bandits do not deal with transitions between states or delayed rewards and are hence considered to be single-state RL problems.

### 3.2.4 Deep Reinforcement Learning

Recent advances in deep learning have had a significant influence on the reinforcement learning domain, as well. Deep learning methods enable RL approaches to scale up for problems which were previously intractable, e.g., scenarios with high-dimensional state and action spaces. Consequently, deep reinforcement learning is successfully employed in games such as Atari, Go, and chess to defeat professional human players (Mnih et al., 2015; Silver et al., 2016, 2017).

Deep learning refers to the techniques that build multi-layered neural networks to approximate functions relating inputs to outputs (LeCun et al., 2015). The learning procedure in these networks aims at finding weights (or network parameters) by adjusting those weights iteratively via back propagating the error of prediction of an

objective function. In RL settings, the objective function used in deep networks approximates the components of reinforcement learning: the value function, policy, and model of the environment. Here, we provide the loss function for temporal difference learning with $\lambda = 0$, i.e., TD(0). The mean squared loss of the Q-values for SARSA obtained using a deep network with parameters $\Psi$ is as follows

$$\mathcal{L}(\Psi) = \mathbb{E}_{r_{t+1}, s_{t+1}, a_{t+1}} \left[ \left( r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}; \Psi) - Q(s_t, a_t; \Psi) \right)^2 \right] \quad \text{(cmp. Equation (3.3))}.$$

The same holds for Q-Learning

$$\mathcal{L}(\Psi) = \mathbb{E}_{r_{t+1}, s_{t+1}} \left[ \left( r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \Psi) - Q(s_t, a_t; \Psi) \right)^2 \right] \quad \text{(cmp. Equation (3.4))}.$$

# Chapter 4

# Detecting Topics of User Sessions

In this chapter, we introduce our first approach to modeling recommendation systems in sequential settings. We take into account the recent activities of users to identify their actual interests and present a session-based recommendation framework. The motivation for exploiting the latest events in the recommendation problems are described in the first section. Section 4.2 summarizes the related work and main methods for topic detection in sequential prediction problems, including recommender systems. Section 4.3 presents the problem setting and a running example, which we will use throughout the chapter, followed by the main technical contribution. We demonstrate the empirical results in Section 4.4, and Section 4.5 concludes.

## 4.1 Motivation

Recommender systems aim to capture the interests of users by providing tailored recommendations. Guessing the intentions of users is not only fundamental for the overall user experience but is linked directly to revenue. User interests are however often unique and driven by unobservable internal (e.g., mood, spontaneous inspiration) as well as external (e.g., weather, location) processes (Hassan et al., 2010). Capturing user intent is therefore one of the most challenging problems in many retrieval and recommendation tasks. The context of a user is often seen as a proxy for unobserved processes (White et al., 2010). Recall that context may be provided by previously visited pages, viewed items, or user profiles, and is often studied together with personalization (Bennett et al., 2012; Ma et al., 2007). Analyzing the latest browsing history of user sessions narrows down the search space of the recommendation process to the actual intention of users or the topic of the session.

We focus on recommendation systems in which user feedback is recorded implicitly, for instance, through clicks on result pages of search engines or on lists of

recommended items. The implicit feedback can be used to train autonomous recommender systems, as the noisy and incomplete batch of user responses provides a partial labeling of the data. Note that these partial labels do not suffice for purely supervised approaches, as the outcome of recommending alternative items is undefined. On the other hand, the task does not fits a purely unsupervised setting either, as the valuable (partial) ground-truth would be discarded. The abstract problem setting matches that of reinforcement learning-style approaches, where uncertainty about the value of actions (e.g., recommending an item) is minimized by a trade-off between exploration and exploitation (Li et al., 2010; Shani et al., 2005). Unlike matrix factorization-based methods, reinforcement learning approaches are naturally sequential models with intrinsic Markov assumptions that allow for capturing the context of a user by representing sequences of previously clicked items explicitly (see Chapter 3, as well).

We study factored Markov Decision Processes (fMDPs) (Boutilier et al., 1999) to detect topics of user sessions, where the topics are generally multivariate, i.e., the topics are described by a set of different attributes which demonstrate various factors of users interests. We take a sequential approach and leverage ideas introduced by Zimdars et al. (2001) and Shani et al. (2005) to characterize sessions in terms of the history of viewed items. However, solving the resulting fMDPs in a straightforward manner is infeasible due to the exponentially increasing state spaces. Besides, the structure of the value function does not necessarily retain the structure of the process after factorization (Koller, 1999). Hence, many approaches to approximating value functions have been proposed (Boutilier et al., 2000; Guestrin et al., 2003).

Commencing with a standard fMDP on the history of viewed items, our main contributions in this chapter are as follows. We show that an independence assumption on the attributes of items allows the fMDP to be represented equivalently by an ensemble of independent MDPs. Compared to the initial fMDP, the resulting state space is orders of magnitudes smaller, and the ensemble can be optimized efficiently. In addition, we propose a robust approximation following ideas from Shani et al. (2005) to improve the predictive accuracy in the presence of data sparsity and large-scale applications. We show that the learned value functions result in interpretable topics which can be effectively turned into recommendations.

## 4.2 Related Work

There are many applications that take contextual variables of users into account including query refinement (Sadikov et al., 2010), re-ranking for web search (Giannopoulos et al., 2011; Xiang et al., 2010), market segmentation (Haider et al., 2012), and latent variable models for spoken language understanding (Celikyilmaz et al., 2011). However, an alternative approach to capturing the intent of users are topic models (Blei et al., 2003). Topic models can be seen as generative probabilistic semantics that have been proposed for information retrieval (Wang & Blei, 2011) as well as recommender systems (Chatzis, 2012; Purushotham et al., 2012).

Topic detection is a broad field in machine learning, particularly for processing documents. Topics of static data collections such as text corpora are traditionally identified using Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and variations thereof. The evolution of topics in data streams is for instance detected by modeling time (Wang & McCallum, 2006) or by introducing additional dependencies (Barbieri et al., 2013). Other approaches, such as dynamic topic models by Blei & Lafferty (2006) and online LDA from AlSumait et al. (2008), study segmented data streams. The general idea is to turn topics of previous segments into priors for the actual time slice. A drawback of these approaches is that the topics remain constant across segments; effectively the same topics are re-identified and there is no mechanism to discard outdated topics or to introduce new ones.

Barbieri et al. (2013) extend LDA to a first-order Markov model that determines topics of interest for collaborative recommendations. They propose a personalized model based on user click histories where topics are identified for every user in the system. Empirically, the sequential approach performs comparable to a non-sequential baseline in terms of perplexity. Wang & Blei (2011) study LDA with collaborative filtering and matrix factorization. They deploy topic models to assess content similarities in the reduced space of topics. Similarly, Chatzis (2012) proposes to combine collaborative filtering with Indian buffet processes for movie recommendations. The three approaches nevertheless aim at capturing long-term interests of users and an application to short-term goals of a session is not straight forward.

By contrast, Wang & Zhang (2013) propose a session-aware recommender system that aims to capture the general intention of users in terms of three predefined and abstract categories: repurchase, variety-seeking, and buying new products. Note that the topics in these works (Barbieri et al., 2013; Chatzis, 2012; Wang & Blei, 2011; Wang & Zhang, 2013) are computed prior to the recommendation and can thus

be considered static. Nevertheless, we design a dynamic topic detection approach using Markov decision processes (Puterman, 2014; Sutton & Barto, 1998) which are frequently used for sequential decision making under uncertainty. The list of previous studies relating MDP to sequential recommendation is summarized in Chapter 2. In this chapter, we take advantages of factored MDPs that are introduced by Boutilier et al. (1999) for scalability and efficiency purposes.

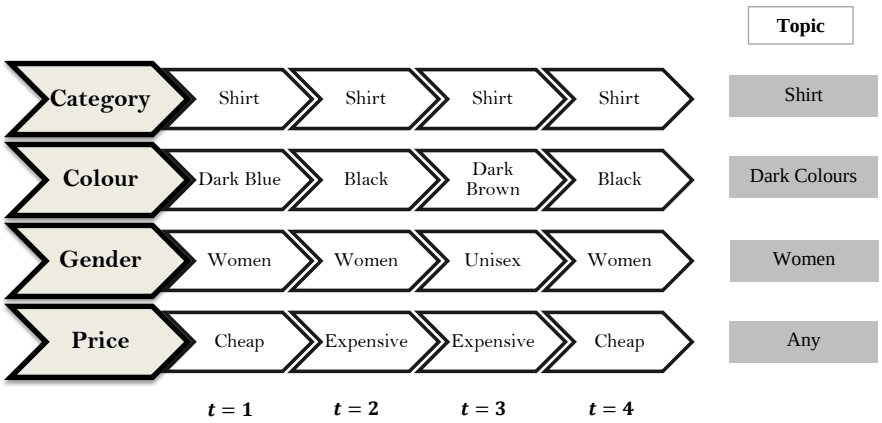



Figure 4.1: An exemplary user session of an e-commerce platform which is viewed as a sequences of item attributes to determine the topic of the session.

## 4.3 The MDP Framework

Short-term interests of users provide valuable additional information to improve the quality of recommendations. These interests may rise from unexpected events (got an MP3 player, need headphones), spontaneous moods and ideas (split-up with partner, need action movie for distraction), or external sources (it is freezing outside, need winter coat). Figure 4.1 visualizes the scenario using an e-commerce example. The figure shows an exemplary user session where a user views a series of garments. The first item is a "`cheap dark-blue shirt`" followed by an "`expensive black shirt`" and so on. That is, the garments are fully described by their attributes. Instead of addressing the items and their attributes jointly, we treat their features as independent. We thus focus on sequences [`shirt, shirt, shirt, shirt`] for the attribute **category** or [`dark-blue, black, dark-brown, black`] for **color**, respectively. Every such sequence gives us an expectation about the value of the next item. For instance, the former (constant) sequence is likely extended by another `shirt` while the latter gives rise to a dark colored item. The attribute **price** in Figure 4.1 constitutes a

special case as the sequence does not allow to confine the possible values. Hence, `any` of the attribute values is possible or, in other words, the attribute **price** is not important for the current user session.

Given the session in Figure 4.1, the user's goal is to find "`dark colored shirts for women of any price level`". We call the corresponding distribution of attribute values the *topic* of the session. For the attribute **color**, we expect dark colors to be very likely while light colors are associated with small probabilities close to zero. Formally, a topic with $d$ variables is defined as follows.

**Definition 1.** *Let $\mathcal{S}_j^t$ be a (possibly ongoing) session of user $u_j$ and $\{\mathcal{Z}_1, \ldots, \mathcal{Z}_d\}$ be a set of random variables encoding attributes of items. The topic of a session $\mathcal{S}_j^t$ is defined as the probability distribution of attributes of the next item given by $P(\mathcal{Z}_1 = z_1, \ldots, \mathcal{Z}_d = z_d | \mathcal{S}_j^t)$.*

Once the topic of a session has been detected, a recommender system could leverage the estimations to recommend items that lie in the very topic of the session. Note that the topic of a session is independent of other sessions of that user as well as its lifetime. Therefore, the topic is well suited to adapt to short-term interests of different users in a non-personalized setting. We thus discard the index $j$ in the remainder of this chapter. We aim to accurately identify the topic of user sessions using factored Markov Decision Processes (fMDPs) and define user sessions as sequences of viewed (clicked) items as defined in Section 3.1.

Formally, we are given a set of $n$ items $\mathbb{B}$, which are described by a set of $d$ random variables $\mathbb{B} = (\mathcal{Z}_1, \ldots, \mathcal{Z}_d)$, where each $\mathcal{Z}_l$ encodes an attribute (e.g., color, category) and takes on values in a discrete and finite set $\mathrm{dom}(\mathcal{Z}_l)$. Every item $b_i \in \mathbb{B}$ is therefore defined by a set of $d$ attributes $b_i = (z_1^i, \ldots, z_d^i)$ with $z_l^i \in \mathrm{dom}(\mathcal{Z}_l)$ for $1 \leq l \leq d$. Items are completely characterized by their attributes, that is, the existence or probability of an item is equivalent to the existence or probability of its combination of attributes. Note that in this chapter, we deal with the categorical features of items, and obtaining the vectorized features $\boldsymbol{x}_i$ requires some conversion from $(z_1^i, \ldots, z_d^i)$ such as one-hot encoding that we get back to that in Chapter 6. Hence, assuming for instance that item $b_i$ has attributes $b_i = (z_1^i, \ldots, z_d^i)$ it holds

$$P(b_i) = P(\mathcal{Z}_1 = z_1^i, \ldots, \mathcal{Z}_d = z_d^i). \tag{4.1}$$

We omit the superscript $i$ in the remainder to not clutter the notation unnecessarily.

An MDP is specified by a five-tuple $(\mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma)$ as represented in Section 3.2.1. In a straight forward sequential MDP for contextual item recommendation, the set

of states $\mathbb{S}$ is defined as the Kleene closure[1] of the set of items, that is $\mathbb{S} = \mathbb{B}^*$. The set $\mathbb{S}$ thus contains all possible sequences of items and every element $s \in \mathbb{S}$ can be identified with a (possibly unfinished) user session in terms of the viewed items, and gives the definition of context for this chapter. Therefore, an action $b \in \mathbb{B}$ corresponds to recommending a particular item from $\mathbb{B}$, so that we may identify $\mathbb{A} = \mathbb{B}$ and consequentially for every $b \in \mathbb{B}$ there exists an $a \in \mathbb{A}$ such that $b = a$ and vice versa.

The described MDP is trivially infeasible due to the infinite number of states $\mathbb{S}$. Though in practice not all possible sequences will actually be observed and an additionally incorporated Markov assumption may further reduce the state space, the model remains intractable even for small and medium-sized ranges of items.

### 4.3.1 Factorization

We therefore take a different approach and define the MDP over the set of attributes $\{\mathcal{Z}_1, \ldots, \mathcal{Z}_d\}$ instead of the items $\mathbb{B}$. Due to Equation (4.1), we obtain an equivalent factored MDP or fMDP where the set of states is given by the Kleene closure $\mathbb{S} = \{\mathcal{Z}_1, \ldots, \mathcal{Z}_d\}^*$. An element $s_l \in \mathbb{S}_l$ corresponds to a sequence of realizations of the $l$-th attribute. Consequentially, the factorization also impacts the set of actions which is now given by $\mathbb{A} = \mathbb{A}_1 \times \ldots \times \mathbb{A}_d$ with $\mathbb{A}_l = \mathrm{dom}(\mathcal{Z}_l)$ for all $l \in \{1, \ldots, d\}$. We use $a_l \in \mathbb{A}_l$ and $z_l \in \mathcal{Z}_l$ interchangeably in the remainder for convenience.

The reward after taking action $a \in \mathbb{A}$ (i.e., recommending the corresponding item $b$) in state $s$ is given by the reward function $\mathcal{R}(s, a)$. Positive rewards indicate a click on a recommended item in which case the recommendation was successful and has been accepted by the user. The transition function $\mathcal{P}(s, a, s')$ estimates the probability of entering state $s'$ after recommending action $a$ in state $s$. Note that $s$ serves as a prefix of $s'$ which is given by $s' = s \circ b$, where $b$ is the clicked item by the user and $\circ$ is the operator that appends two sequences, e.g., $p \circ q = pq$. The same holds for the factored representation.

The length of the actual state $s$ is continuously increased by appending clicked items, exactly one at a time. Thus, instead of addressing the complex $\mathcal{P}(s, a, s')$, transition probabilities $\mathcal{P}(s, a, b)$ are used as an equivalent proxy due to their simpler structure. The quantity $\mathcal{P}(s, a, b)$ is the transition probability of clicking on item $b$ when in state $s$ and recommending item $a$. The transition probabilities can be represented as a two-layer acyclic graph that connects the attributes of the previously

[1] A unary operation on sets (or string); it is described as possible combinations that can be created by concatenating elements of a base set.

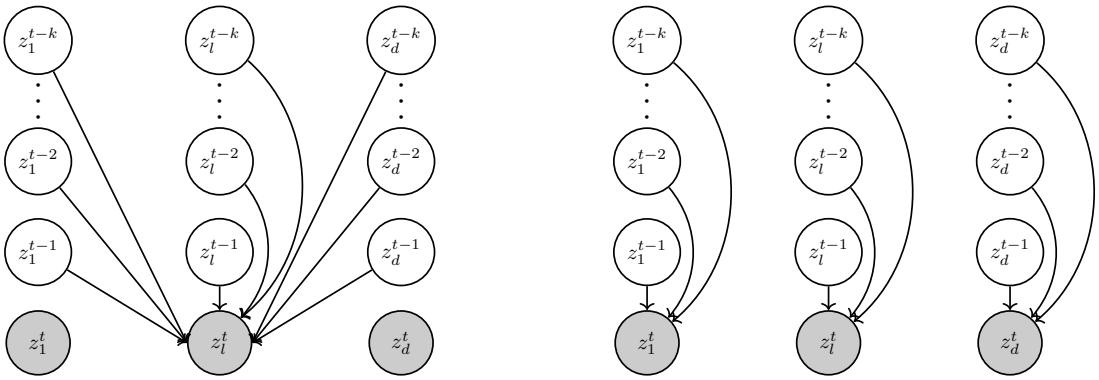



Figure 4.2: (a): Transition model of a joint factored MDP. Every attribute value depends on the complete history of all previously viewed items. (b): There are no dependencies between different attributes.

viewed items in $s$ with the attributes of the item to be clicked denoted by $b$. Theoretically, the joint transition probability can be efficiently computed by factorizing conditional probabilities, e.g.,

$$P(z'_1, \ldots, z'_d | s, a) = \prod_{l=1}^{d} P(z'_l | \text{ parents}(z'_l), a),$$

where parents$(z'_l)$ determines the parents of the node $z'_l$ in the underlying graphical model. Nevertheless, the state space of the factored MDP grows exponentially in terms of the number of attributes as well as the length of the session. This renders practical application infeasible as the exact estimation of the optimal policy is not feasible due to the curse of dimensionality (Guestrin et al., 2003). Thus, we have not won anything yet in terms of feasibility but successfully rephrased the model over attribute sequences of the viewed items. Figure 4.2 (a) displays the structure of the factored model where all the instances of previous attributes have part in determining a single variable in the next state.

Directly addressing the joint probability $P(z_1, \ldots, z_d | s, a)$ requires a state space that is intractable even for small and medium-sized warehouses. We therefore treat the attributes of the items as independent and approximate the intractable joint by a product of independent decisions,

$$P(z_1, \ldots, z_d | s, a) \approx \prod_{l=1}^{d} P(z_l | s_l, a_l),$$

which leads to factorized transition probabilities $\mathcal{P}(s_l, a_l, z_l)$. Figure 4.2 (b) shows that the independence assumption impacts the resulting transition model, and sequences of the same attribute form independent components that can be optimized efficiently. The idea is to split the fMDP into an ensemble of $d$ disjoint and independent MDPs, one for each attribute. The $l$-th MDP therefore focuses on only the $l$-th attribute and recommends a realization $a_l$ of $\mathcal{Z}_l$ based on the sequence of attributes $s_l$ of the previously viewed items. In Figure 4.1 for instance, guessing that the next item will be another `shirt` can trivially be done in the absence of all other attributes. A similar argument holds for expecting a `dark-color` or a garment for `women`.

The following theorem shows that an fMDP with independent chains of random variables admits an equivalent representation as an ensemble of $d$ independent MDPs. In order to propagate single receiving reward through all factors, we consequentially assume additive factorized rewards $\mathcal{R}(s, a) = \sum_{l=1}^{d} \mathcal{R}_l(s_l, a_l)$.

**Theorem 1.** *An fMDP with a set of $d$ independent components $\mathcal{Z} = \{\mathcal{Z}_1, \ldots, \mathcal{Z}_d\}$ allows an equivalent representation as an ensemble of $d$ independent MDPs, one for each component $\mathcal{Z}_l$, where $1 \leq l \leq d$. Let $V^*(s)$ be the optimal value for state $s = (z_1, \ldots, z_d)$ in the joint fMDP and $V^*(s_l)$ be the optimal value for attribute $s_l = z_l$ in the $l$-th factored MDP, then for all $1 \leq l \leq d$. It holds that*

$$V^*(s) = \sum_{l=1}^{d} V^*(s_l).$$

*Proof.* The standard update rule of value iteration is given by

$$V_{\iota+1}(s) = \max_a \left[ \mathcal{R}(s, a) + \gamma \sum_{s'} \mathcal{P}(s, a, s') V_\iota(s') \right] \quad \text{(cmp. Equation (3.2))}.$$

We approximate the maximum in the above equation by a softmax function,

$$\max(\{v_l\}_{l=1}^{d}; \rho) = \frac{1}{\rho} \log \sum_{l=1}^{d} \exp(\rho v_l),$$

to preserve linearity. The parameter $\rho$ controls the degree of approximating the maximum, that is, for $\rho \to \infty$ we obtain the exact maximum. Replacing the maximum by the softmax function yields

$$V_{\iota+1}(s) = \frac{1}{\rho} \log \sum_a \exp \left[ \rho \mathcal{R}(s, a) + \gamma \rho \sum_{s'} \mathcal{P}(s, a, s') V_\iota(s'). \right] \tag{4.2}$$

We show the claim by induction for value iteration. For $\iota = 1$, the value of the $l$-th component of the ensemble is

$$V_1(s_l) = \frac{1}{\rho} \log \sum_{a_l} \exp\ [\rho \mathcal{R}_l(s_l, a_l)]\,,$$

and on the other hand, the joint fMDP is obtained by

$$\begin{aligned} V_1(s) &= \frac{1}{\rho} \log \sum_{a} \exp\Big(\rho \mathcal{R}(s,a)\Big) \\ &= \frac{1}{\rho} \log \sum_{a} \exp\Big(\rho \sum_{l} \mathcal{R}_l(s_l, a_l)\Big) \\ &= \frac{1}{\rho} \log \sum_{a_1} \Big( \sum_{a_2} \ldots \big( \sum_{a_d} \prod_{l} \exp[\rho \mathcal{R}_l(s_l, a_l)]\big)\Big). \end{aligned}$$

By drawing the unrelated terms out of the sum, the innermost summation can be rewritten as

$$\begin{aligned} &\sum_{a_d} \Big( \exp[\rho \mathcal{R}_1(s_1, a_1)] \times \cdots \times \exp[\rho \mathcal{R}_d(s_d, a_d)]\Big) \\ &\quad = \exp[\rho \mathcal{R}_1(s_1, a_1)] \times \cdots \times \exp[\rho \mathcal{R}_{d-1}(s_{d-1}, a_{d-1})] \times \sum_{a_d} \exp[\rho \mathcal{R}_d(s_d, a_d)]. \end{aligned}$$

Continuing for the other summations gives

$$\begin{aligned} V_1(s) &= \frac{1}{\rho} \log \Big( \prod_{l} \sum_{a_l} \exp[\rho \mathcal{R}_l(s_l, a_l)]\Big) \\ &= \frac{1}{\rho} \sum_{l} \log \sum_{a_l} \exp[\rho \mathcal{R}_l(s_l, a_l)] \\ &= \sum_{l} V_1(s_l), \end{aligned}$$

which demonstrates the claim for $\iota = 1$.

Now assume that $V_\iota(s) = \sum_{l=1}^{d} V_\iota(s_l)$ holds for all $s \in \mathbb{S}$. Using Equation (4.2) for the joint fMDP, the second summand in the exponent is simplified by

$$\begin{aligned} \sum_{s'} \mathcal{P}(s,a,s') V_\iota(s') &= \sum_{s'} \mathcal{P}(s_1, a_1, s_1') \ldots \mathcal{P}(s_d, a_d, s_d') V_\iota(s') \\ &= \sum_{s'} \mathcal{P}(s_1, a_1, s_1') \ldots \mathcal{P}(s_d, a_d, s_d') \left[V_\iota(s_1') + \ldots + V_\iota(s_d')\right], \end{aligned}$$

where the latter gives rise to the following telescope sum

$$\begin{aligned} \sum_{s_1'} \mathcal{P}(s_1, a_1, s_1') \Big[ \sum_{s_2'} \mathcal{P}(s_2, a_2, s_2') \Big[ \times \ldots \\ \ldots \times \Big[ \sum_{s_d'} \mathcal{P}(s_d, a_d, s_d') \left[V_\iota(s_1') + \ldots + V_\iota(s_d')\right] \Big]\Big]\Big]. \end{aligned}$$

The innermost summation over the new state $s'_d$ yields

$$V_\iota(s'_1) \sum_{s'_d} \mathcal{P}(s_d, a_d, s'_d) + \ldots + \sum_{s'_d} \mathcal{P}(s_d, a_d, s'_d) V_\iota(s'_d),$$

and since $\sum_{s'_d} \mathcal{P}(s_d, a_d, s'_d) = 1$ we obtain

$$V_\iota(s'_1) + \ldots + V_\iota(s'_{d-1}) + \sum_{s'_d} \mathcal{P}(s_d, a_d, s'_d) V_\iota(s'_d).$$

Drawing out the remaining terms from unrelated summations and putting things together yields

$$V_{\iota+1}(s) = \frac{1}{\rho} \log \sum_a \exp \left[ \rho \sum_l \left( \mathcal{R}_l(s_l, a_l) + \gamma \sum_{s'_l} \mathcal{P}(s_l, a_l, s'_l) V_\iota(s'_l) \right) \right],$$

and therefore

$$V_{\iota+1}(s) \quad = \quad \frac{1}{\rho} \log \sum_{a_1} \left[ \cdots \left[ \sum_{a_d} \prod_l \exp[\rho(\mathcal{R}_l(s_l, a_l) \; + \; \gamma \sum_{s'_l} \mathcal{P}(s_l, a_l, s'_l) V_\iota(s'_l))] \right] \right].$$

Reordering terms analogously to the case $\iota = 1$ shows the claim. □

Theorem 1 shows that any high-dimensional fMDP with independent attributes can be equivalently expressed by several independent MDPs. Exploiting the independence between the attributes leads to an ensemble consisting of an MDP for every component. The resulting state spaces are independent sequences over a single attribute given by the Kleene closure $s_l = \{\mathrm{dom}(\mathcal{Z}_l)\}^*$ for all components $l$. Note that a study by Koller (1999) shows, that the value function of fMDPs does not in general retain the structure of the process. Our theorem proves that a structured value function is generally obtainable for fMDPs with independent components.

Still, a major drawback of the model is the dependence on the whole session, that is, every viewed item impacts all subsequent actions. We therefore take a $k$-th order Markov assumption to represent only the $k$ most recently viewed items explicitly which stands also for the length of a session. The set of states of the $l$-th MDP is effectively reduced to $s_l = (\mathrm{dom}(\mathcal{Z}_l))^k$. The Markov assumption discards long-range dependencies and lead, together with the previous independence assumption, to an efficient and compact representation of the ensemble as shown in Figure 4.2 (b).

### 4.3.2 Optimization via RL

The obtained independent fMDP can be optimized independently and in parallel using standard reinforcement learning techniques such as value iteration (see Section 3.2.2). This algorithm learns the state-value function, $V(s)$, using the model of the environment; the reward function $\mathcal{R}(s,a)$ and transition function $\mathcal{P}(s,a,s')$, and converges to an optimal solution in a discounted finite MDP (Sutton & Barto, 1998).

The set of states in the $l$-th fMDP is described by a $k$-sequence of realizations of the $l$-th attribute $\mathcal{Z}_l$, and is given by $s_l = (z_l^{t-k}, \ldots, z_l^t)$. The task of the agent is thus to predict the value of action $a_l \in \mathrm{dom}(\mathcal{Z}_l)$ in the actual state $s_l$. The transition function $\mathcal{P}$ encodes the probability of observing the subsequent state $s'_l = (z_l^{t-k+1}, \ldots, z_l^{t+1})$ and the reward function $\mathcal{R}_l$ provides feedback for recommending $a_l$ in $s_l$. Value iteration uses the following update rule for value determination,

$$V_{\iota+1}(s_l) = \max_{a_l} \Big[\mathcal{R}_l(s_l,a_l) + \gamma \sum_{s'_l} \mathcal{P}(s_l,a_l,s'_l) V_\iota(s'_l)\Big].$$

When the value function converges to the optimal $V^*$, state-action values $Q(s_l,a_l)$ can be derived from

$$Q(s_l,a_l) = \mathcal{R}(s_l,a_l) + \gamma \sum_{s'_l} \mathcal{P}(s_l,a_l,s'_l) V^*(s'_l), \tag{4.3}$$

where $Q(s_l,a_l)$ measures the quality of recommending $a_l$ in state $s_l$. Realizations with high $Q$-values are likely to be observed in the next page view while small $Q$-values indicate very unlikely observations. We use the terms $Q(s_l,a_l)$ and $Q(s_l,z_l)$ interchangeably in the remainder.

Note that RL techniques often perform poorly in large-scale problems due to slow convergence rates. Adapting the model to data is therefore carried out in two steps: offline and online. First, an initial model is learned by value iteration where transition and reward functions are adapted to historic data by maximum likelihood. The trained model is then deployed in an online scenario where it is gradually updated according to the user feedback to improve estimations. In practice, value iteration can be repeated periodically (e.g., once in a week) to keep the system up to date.

Nevertheless, in practical applications, the available data is often too sparse to allow for an accurate estimation of the transition probabilities. In addition, keeping the whole set of transition probabilities is infeasible for large-scale applications due to memory requirements. We thus propose an efficient approximation of our model based on the ideas from Shani et al. (2005).

The main idea is hence to estimate the probability $P(b'|s)$ of item $b'$ to be clicked next, irrespectively of the action. The transition $\mathcal{P}(s, a, b')$ can be approximately reconstructed from $P(b'|s)$ as follows. Recall that action $a$ is identical to an item $b \in \mathbb{B}$. There are three possible outcomes of taking action $b = z$ when in state $s$:

- The user accepts the recommendation $b$ with probability $P(s \circ b|s, b)$.
- The user rejects $b$ and clicks instead on item $b'$ with probability $P(s \circ b'|s, b)$.
- The session terminates with probability $P(\emptyset|s, b)$.

Consider the former two events. The task is to estimate $P(s \circ b|s, b)$ and $P(s \circ b'|s, b)$ as a surrogate for the entire transition function. Note that in the latter, a click on $b'$ is independent of the recommended item $b$.

We assume that the probability of clicking on a recommended item is intuitively larger than the probability of choosing the item in the absence of a recommendation, that is, $P(s \circ b|s, b) \geq P(s \circ b|s)$ (Shani et al., 2005). Analogously, the probability of clicking on item $b'$ in the absence of any recommendation is higher than for clicking on $b'$ when the recommended item is actually $b \neq b'$, that is, $P(s \circ b'|s, b) \leq P(s \circ b'|s)$. Therefore, by choosing appropriate constants $\delta_+ > 1$ and $0 < \delta_- < 1$, the desired quantities are approximated by

$$\mathcal{P}(s, b, b) \approx \delta_+ P(b|s) \quad \text{and} \quad \mathcal{P}(s, b, b') \approx \delta_- P(b'|s), \tag{4.4}$$

subject to $\mathcal{P}(s, b, b) + \sum_{b' \neq b} \mathcal{P}(s, b, b') + \mathcal{P}(s, b, \emptyset) = 1$, which is obtained by normalizing values for each $s$ and $b$.

### 4.3.3 Topic Extraction and Recommendation

Once the approximate or exact $Q$-values, $Q(s_l, z_l)$, for all sequences $s_l$ and realizations $z_l$ are computed, they can be used to extract the topic of the session as follows. The value $Q(s_l, z_l)$ is proportional to the probability that the user clicks on an item with attribute $z_l$ given the sequence of realizations $s_l$. In other words, realizations with high $Q$-values are more likely to be observed next and thus constitute a part of the topic of $s_l$. By contrast, for uniformly distributed $Q$-values, e.g., $Q(s_l, z_l) \approx Q(s_l, z'_l)$ for all $z_l, z'_l \in \mathcal{Z}_l$, the topic contains the whole domain $\mathrm{dom}(\mathcal{Z}_l)$, indicating that the $l$-th attribute does not contribute to the topic. As a consequence, any realization of that attribute may be observed next. Intermediate $Q$-values are ranked according

to their difference to the maximum $Q$-value, such that the expected realizations of attribute $l$ are computed by the min-max normalization

$$q(\mathcal{Z}_l = z_l|s_l) = \frac{Q(s_l, z_l) - \min_{s'_l}[Q(s_l, s'_l)]}{\max_{s'_l}[Q(s_l, s'_l)] - \min_{s'_l}[Q(s_l, s'_l)]},$$

for all $l \in \{1, \ldots, d\}$. The independent results are then multiplicatively combined to approximate the desired quantity of the joint topic

$$q(z_1, \ldots, z_d|s) \propto \prod_{l=1}^{d} q(z_l|s_l).$$

Consequently, we are able to turn our approach into a recommendation system. In contrast to the topic extraction, we use a softmax instead of the min-max normalization to translate $Q$-values into probabilities,

$$P(\mathcal{Z}_l = z_l|s_l) = \frac{\exp\{Q(s_l, z_l)\}}{\sum_{z'_l} \exp\{Q(s_l, z'_l\}}. \tag{4.5}$$

The softmax yields a probability distribution over the state space of every attribute. The use of the exponential function penalizes even small differences and thus acts like a probabilistic winner-takes-all. Note that in practice, recommendations have to be computed very efficiently under rigid time constraints. Having a clear set of winners helps to speed-up the computation by continuously filtering out items at early stages that cannot make it into the top-$K$ to save time for more promising candidates.

Given the estimates in Equation (4.5), the score for item $b$ with attribute combination $\{z_1, \ldots, z_d\}$ is simply given by the product of the corresponding probabilities, or alternatively, by the sum of the corresponding log-probabilities, that is,

$$\text{score}(b; s) = \prod_{l=1}^{d} P(\mathcal{Z}_l = z_l|s_l) \propto \sum_{l=1}^{d} \log P(\mathcal{Z}_l = z_l|s_l). \tag{4.6}$$

The scores impose a ranking on the items and the top-scoring products can be recommended in the next step.

## 4.4 Empirical Evaluation

In this section, we evaluate our topic detection approach on a real-world dataset from Zalando[2], a large European online fashion retailer. The available click log of users are anonymized of any information of users and products. The data distribution is

[2] `www.zalando.com`

modified so that no conclusions on customer data or business figures of the company can be drawn. The dataset contains $1,721,483$ user sessions consisting of $24,353,852$ clicks (transactions) in total. Sessions are split after 25 minutes idle time and the sessions in average comprise 14 clicks. Every click is associated with a time stamp, the attributes of the viewed item, user identifier, and the recommended items. We focus on attributes **color**, **gender**, **category**, and **price**. There are 62 different colors, 16 genders (including types of accessories), 61 categories, and 16 discrete levels of price in the log, which create the domain of each variable.

We first measure the performance of predicting the topic of user sessions in two sets of experiments: small-scale and large-scale. The derived topics are then utilized for recommendation purpose for further performance evaluation.

### 4.4.1 Topic Detection

Measuring the performance of topic detection methods using real-world data is difficult as topics are not observed variables but contained only implicitly in the data. We therefore test the topic prediction against the attribute values of the next clicked item. We translate the distribution in Equation (4.5) into a discrete set of attribute values. A simple thresholding approach discards unlikely realizations and returns a set $\mathbb{T}_l$ for every attribute $l \in \{1, \ldots, d\}$ given a session (state) $s = (s_1, \ldots, s_d)$,

$$\mathbb{T}_l(s_l) = \{z_l | z_l \in \mathrm{dom}(\mathcal{Z}_l) \wedge q(\mathcal{Z}_l = z_l | s_l) > \eta\},$$

where $\eta$ is a user defined constant. Large values of $\eta$ thin out the topic and focus on highly probable attribute values; thus, empty topics may be the consequence. On the other hand, small values of $\eta$ weaken the interpretability and usability of the resulting topics unnecessarily that may contain many unlikely realizations. In the first set of experiments, we use $\eta = \frac{1}{2}$ and study variations of the parameter afterwards. The joint topic $\mathbb{T}(s)$ is then given as the union over all attributes by

$$\mathbb{T}(s) = \bigcup_{l=1}^{d} \mathbb{T}_l(s_l).$$

The set $\mathbb{T}$ specifies the attribute values that are within the topic of the session.

We evaluate the accuracy of the extracted topics for every attribute as well as for the joint topic using indicator functions $\mathbb{1}[v]$, yielding one if the argument $v$ is true and zero otherwise. Let $\mathbb{T}_l(s_l)$ be the estimated topic of an ongoing session for $l$-th variable, and $z'_l$ be the corresponding realization of the next clicked item. The

topic prediction is correct if $\mathbb{1}[z'_l \in \mathbb{T}_l(s_l)]$. The joint topic is then evaluated by concatenating the individual results with an and-operator,

$$ACC(\mathbb{T}(s); z') = \bigwedge_{l=1}^{d} \mathbb{1}[z'_l \in \mathbb{T}_l(s_l)],$$

which indicates that a topic detection is accurate if all the variables of the next item are correctly predicted. Note that high accuracies in individual attributes do not necessarily indicate a good joint performance as the all attribute values need to be contained in the topic. We average the accuracy of all the visited states to compute the total accuracy on the test data.

We therefore compare the proposed ensemble approach with its approximation where the transition probabilities are estimated from probabilities without recommendation. In the former, the transition function $\mathcal{P}(s, a, b)$ is directly estimated from the training data, while in the latter, is approximated from $P(b|s)$ as in Equation (4.4). We address the two versions, **fMDP-exact** and **fMDP-approx**, respectively.

As a baseline, we deploy a simple Markov process (**MP**) that uses estimates of $P(b|s)$ directly instead of $Q(s, b)$ for the computation of the topic. Thus, its probabilities are proportional to the number of times that item $b$ has been clicked in state $s$, and is estimated by maximum likelihood. Additionally, we include Latent Dirichlet Allocation (**LDA**) (Blei et al., 2003) as another baseline. To this end, every session is treated as a document where the attributes of the viewed items are considered the (unordered) words of the document. The set of words is hence defined by $\bigcup_{l=1}^{d} \mathrm{dom}(\mathcal{Z}_l)$ and contains 155 distinct words. We apply the method by Blei et al. (2003) for both estimation and inference of topic proportions as well as word distribution per topic. At testing time, LDA determines the topic mixture of the ongoing session and computes the probability distribution of attributes according to the mixture. Thresholding procedure is identical for all methods.

For the first set of experiments, we only use a subset of the data for evaluation to compare the two proposed methods, since the exact variant cannot be evaluated on all available data due to memory issues. In the corresponding subset, there are $34,343$ user sessions consisting of $722,179$ clicks in total with the average of 21 clicks per session. We split 70% of the resulting sessions for training, 20% as holdout, and 10% as test sessions according to the temporal nature of the data. Optimal parameters for fMDP-approx and LDA are found by model selection using training and holdout sets only. The values are given by $\delta_+ = 2$, $\delta_- = 0.001$ for fMDP-approx and $\alpha_{LDA} = 0.1$ and 100 topics for LDA, respectively. Rewards are positive for clicks on recommended

Table 4.1: Accuracies for the topic detection on a subset.

| | $k$ | **joint** | **color** | **gender** | **category** | **price** |
|---|---|---|---|---|---|---|
| MP | 4 | 33.69 | 49.78 | 92.24 | 78.52 | 63.96 |
| | 3 | 37.70 | 52.98 | 92.31 | 79.50 | 65.06 |
| | 2 | 37.65 | 52.15 | 92.22 | 79.68 | 64.24 |
| | 1 | 28.06 | 44.31 | 91.85 | 79.01 | 56.28 |
| fMDP-exact | 4 | 67.53 | 85.61 | 95.00 | 90.70 | 78.68 |
| | 3 | 69.56 | 93.94 | 95.21 | 93.36 | 72.01 |
| | 2 | 40.62 | 45.96 | 95.30 | 94.90 | 78.39 |
| | 1 | 16.47 | 28.37 | **95.31** | **95.28** | 46.55 |
| fMDP-approx | 4 | 75.33 | 81.92 | 94.65 | 90.05 | 92.38 |
| | 3 | 89.52 | 92.95 | 94.83 | 92.81 | 94.48 |
| | 2 | 93.69 | 95.12 | 94.97 | 94.45 | **95.00** |
| | 1 | **94.14** | **95.25** | 94.98 | 94.82 | 94.97 |
| LDA | - | 01.65 | 11.76 | 85.89 | 52.80 | 21.14 |

items as well as adding to cart, and sale actions. Removing items from the cart is penalized with negative rewards, all other actions realize a reward of zero.

Table 4.1 shows the average accuracies of the best models for Markov assumptions of order $k \in \{1, 2, 3, 4\}$ as well as LDA. The exact ensemble fMDP-exact performs poorly for short histories but improves significantly for larger $k$. We credit this finding to the necessity of taking chains of consecutive clicks into account. Although the individual predictions on attribute levels are promising, the joint topic is not well captured in this setting. Furthermore, the predictive accuracy is constantly below 70%, which is because of sparsity issues on the small data samples, that is, a great deal of different attribute combinations are observed but their frequency is not high enough to properly explore the corresponding actions. By contrast, the approximate mode (fMPD-approx) performs much better for short histories and detects the correct topic in 94% of the cases for $k = 1$. The performance decreases for longer chains. The observed effect originates from the approximation itself. The data sample is not sufficiently large for reliably approximating longer histories. We will address this issue in the next experiment again.

On the other hand, LDA characterizes the next click by dominant attributes of the ongoing session without considering any sequentiality in the process. The results illustrate that users tend to click on items with so far unseen attribute values, particularly for price and color. However, apart from fMDP-approx, the joint topics are mostly inaccurate and do not reflect the performance for individual attributes.

Table 4.2: Accuracies for the topic detection using all data.

| | k | **joint** | **color** | **gender** | **category** | **price** |
|---|---|---|---|---|---|---|
| MP | 4 | 39.56 | 53.50 | 89.70 | 77.93 | 71.25 |
| | 3 | 39.53 | 52.83 | 89.70 | 78.09 | 71.04 |
| | 2 | 38.37 | 50.78 | 89.57 | 77.94 | 71.09 |
| | 1 | 30.82 | 42.37 | 89.15 | 77.29 | 70.02 |
| fMDP-approx | 4 | 88.30 | 91.09 | **92.61** | 90.88 | 92.19 |
| | 3 | 91.13 | 92.73 | 92.45 | 92.04 | **92.56** |
| | 2 | 91.48 | 92.82 | 92.46 | 92.37 | 92.49 |
| | 1 | **91.53** | **92.85** | 92.40 | **92.39** | 92.55 |
| LDA | - | 02.84 | 12.31 | 81.18 | 51.22 | 41.71 |

The outcomes of MP show that simply counting frequencies of subsequent events is not sufficient for achieving state of the art performances.

In the second experiments, we focus on the approximate ensemble and repeat the previous experiment on the whole click log rendering a large-scale setup. We split all available data into consecutive training (70%), holdout (20%), and test (10%) sets while preserving the temporal nature of the data. Table 4.2 shows the results for MP and fMDP-approx for histories of size $k \in \{1, 2, 3, 4\}$ as well as LDA. All three methods exploit the abundance of data and improve their performance. However, the overall joint performance of LDA is still far from a real-world deployment and even for MP still stays constantly below 40%. The fMDP-approx clearly outperforms the baselines and yields impressive joint accuracies of about 90% for all $k$. The additional data trades-off the approximation issues observed in the previous experiment for larger $k$ at the expense of smaller $k$. The performance decreases slightly for $k = 4$.

In addition, we analyze the behavior of fMDP-approx for various scenarios using all data. First, we study the variation of the detected topics in the course of the sessions using the experimental setup for larger scale data. We measure the difference of subsequent topics $\mathbb{T}(s)$ and $\mathbb{T}(s')$ by their Jaccard distance $J$ given by

$$J\left(\mathbb{T}(s), \mathbb{T}(s')\right) = \frac{|\mathbb{T}(s) \cup \mathbb{T}(s')| - |\mathbb{T}(s) \cap \mathbb{T}(s')|}{|\mathbb{T}(s) \cup \mathbb{T}(s')|}.$$

A large distance indicates rapid changes in neighboring topics and either refers to a badly adapted model or to undetermined users who are just browsing instead of following a specific goal. By contrast, small distances indicate that users are very predictable and only search for very particular items without digressing.

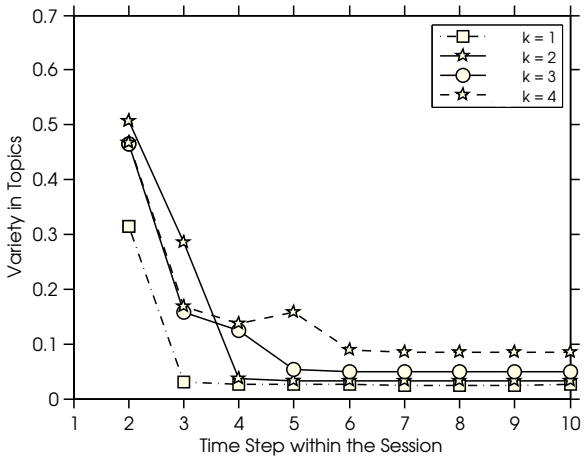



Figure 4.3: Variance of topic detection within sessions (fMDP-approx).

Figure 4.3 depicts the session on the $x$-axis and the variation of neighboring topics in terms of their Jaccard distance on the $y$-axis averaged over all sessions. Unsurprisingly, for all histories $1 \leq k \leq 4$, the variation decreases rapidly after a few clicks. The more clicks a user performs, the more feedback is provided to the system and the main topic can be exploited by the model. Except for histories of length $k = 4$, all models converge quickly to only a few variations. That is, only very few attribute values are replaced between time steps. For longer histories $k = 4$, we observe more variations which is also reflected by lower overall accuracies in Table 4.2. In other words, the model is not as well adapted and needs to correct its predictions at a higher frequency than its competitors.

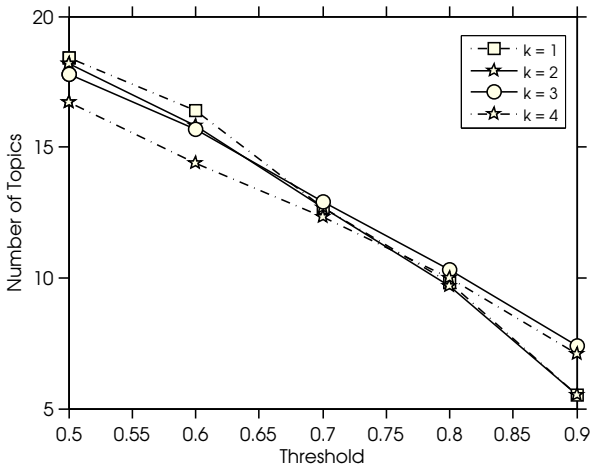



Figure 4.4: Changing the size of topics for attribute category.

In the next experiment, we focus on the attribute **category** in the topic. Fig-

ure 4.4 shows the impact of the topic threshold $\eta$ on the average size of the topics. Increasing the threshold effectively thins-out the topics on average. Nevertheless, changing the topic threshold $\eta$ also impacts the accuracy that we study in the last experiment. Figure 4.5 illustrates that tighter topics may fail to capture the user's intent and we observe decreasing accuracies for larger values of $\eta$. As a consequence, the actual value of $\eta$ trades-off the specificity of topics and the accuracy of the topic prediction.

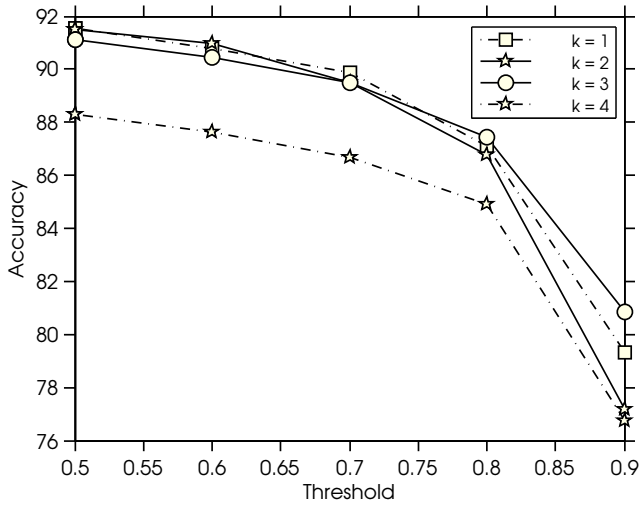



Figure 4.5: Impact of topic threshold on the accuracy of prediction (fMDP-approx).

### 4.4.2 Topic-driven Recommendations

We now demonstrate the effectiveness of the topic detection by translating the identified topics into recommendations according to Equation (4.6). We repeat the experimental setup of the large-scale dataset and compare the approximate ensemble **fMDP-approx** with a collaborative filtering method using matrix factorization (**CF**) and a combination of topic models and collaborative filtering presented by Wang & Blei (2011) that we address as **CF+TM**. We set $\alpha_{LDA} = 0.1$, number of topics = 100, and the number of factors in matrix factorization = 200 by model selection. To evaluate these two baselines, items are ranked according to the previous clicks of the actual user as given by the user-item matrix. Note the conceptual difference of our fMDP and the collaborative filtering approaches. While the former takes a session-based approach and thus aims at short-term interests, the latter two are user-specific and could be considered global models for personalized recommendation. Additionally, we incorporate three simple baselines: ranking items randomly (**Rand**), ranking

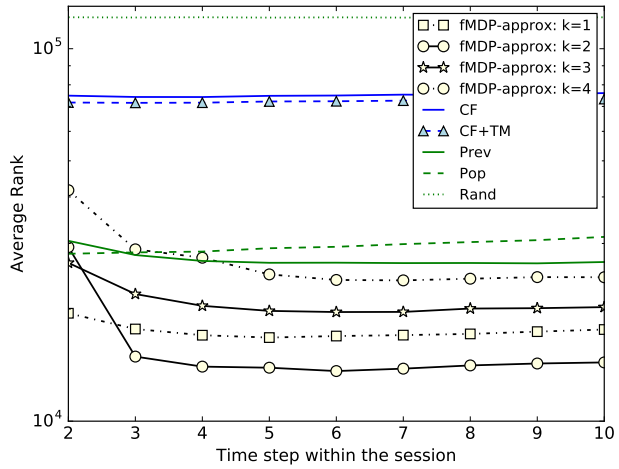



Figure 4.6: Average ranks for the recommendation task (log-scale).

them according to their similarity to the previously viewed item so that items with the same attributes are ranked on top (**Prev**), and ranking items based on their popularity among different users (**Pop**).

We also wanted to include the MDP-based method from Shani et al. (2005) as another competitor to better distinguish the performance of our fMDP from related approaches. However, the approach as described in Section 4.3, is defined in terms of the items and turns out infeasible even for the small sample that we used in the beginning of this section. There are more than 80,000 items in the small sample, leading to a minimum memory requirement of about 52GB for maintaining two tables of size $80,000 \times 80,000$, one for transitions and the other for the Q-values (for $k = 1$) using only four byte representations. The dataset employed in this section contains more than 240,000 items. We therefore leave out this comparison.

Figure 4.6 exhibits the performance of the recommendations in terms of average rank in the course of the sessions. Note that the average rank is a variant of average relative position criterion introduced by Pilászy et al. (2010). As the baselines are static recommender systems that do not exploit the sequential nature of the data, their performance is more or less constant in the length of the session; small fluctuations disappear in the figure due to the log-scaled y-axis. Adding the topic model only slightly improves the collaborative filtering method, and both CF-based approaches perform worse than popularity-based and repeating the previous topics. The sequential fMDP-approx exploits the temporal nature of the data and, similarly to Figure 4.3, adapts quickly to the topic of the session. The best method realizes a second-order Markov assumption. The figure could be extended to the right to

include longer sessions but the information gain is rather small as the performance of all methods does not change significantly.

Moreover, Table 4.3 demonstrates the aggregated results by averaging the performances over the length of the session. The baselines perform worse than fMDP-approx for all values of $k$, and among them, Prev and Pop again outperform the collaborative based methods. The best method for histories of length two realizes average ranks of about 15,000. Although the absolute number appears quite high, recall that there are more than 240,000 items in the dataset. On average, clicked items are among the top 7% of the ranking for $k = 2$.

Table 4.3: Aggregated Average Ranks of recommendation (in hundreds).

| fMDP-approx, $k =$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | CF+TM | CF | Prev | Pop | Rand |
| 174 | **158** | 209 | 270 | 723 | 749 | 272 | 295 | 1213 |

### 4.4.3 Summary of Results

We briefly discuss the obtained results from the empirical study. Tables 4.1 and 4.2 exhibit differences in the predictability of the attributes. Unsurprisingly, **gender** is always predicted with high accuracy as it is unlikely that users switch often between genders within a session. The predicted accuracies for the attribute **category** are also high which indicates that many users do have a clear need for a particular item or at least show interest in a certain type of items. In contrast to **gender** and **category**, attributes **color** and **price** prove more difficult. Apparently, users are somewhat flexible about prices and colors. Nevertheless, we observe highly accurate predictions for these attributes for the approximate ensemble fMDP-approx.

Note that the choice of $k$ depends on the application at hand. Our results demonstrate that the performance of the exact model (fMDP-exact) increases with larger $k$ (see also Tables 4.1). However, the larger the history, the longer it may take to adapt to a change in the topic; for instance because the user has not found what she was searching for or is distracted by a completely different item that is also displayed on the page. In practice, the fMDPs could be reset after cart or purchase operations by the user. The approximate fMDPs nonetheless perform better for short histories although the effect becomes smaller for larger training sets. We credit this finding to difficulties in the approximation caused by sparsity in the data distribution.

Furthermore, since the internal representation of the factored MDPs is a graphical model, augmenting additional variables to capture the context of the user is straight forward. A promising candidate seems to be the time the user spends on the page before clicking. Very short stays could be an indicator for dissatisfaction, possibly followed by a change in topic while longer stays may give rise to a careful examination of the item at hand and a possible cart operation.

Finally, recall the conceptual differences of the fMDP-based recommendation and the collaborative filtering baselines. While the former takes a session-based approach to capture short-term interests of users, the latter is user-centric and implements the notion of personalization which focuses on long-term interests. Thus, the two strategies can be considered orthogonal. In the following chapters, we propose approaches to combine session-based recommendation with personalized strategies to obtain the best of the two worlds.

## 4.5 Conclusion

In this chapter, we introduced a novel viewpoint for modeling recommendation systems as sequential decision making processes. We demonstrated that the sequences of the recent activities of users are important factors in identifying the actual intentions of users or the topic of an ongoing session for the next recommendation.

Therefore, we presented a sequential session-based approach for detecting the intention of user sessions on the web. We phrased the problem as a topic detection task in terms of item attributes and proposed to solve the task using factored MDPs. We argued that a straightforward application is infeasible and devised an efficient formulation by assuming independence over the underlying attributes. We showed that factored MDPs with independent components admit an equivalent representation as an ensemble of independent MDPs with structured value functions.

Additionally, we represented an approximation of the ensemble and evaluated both methods on a large click log from a real-world application at an enterprise-level scale. In extensive experiments, we observed topic detection accuracies of about 90%. We further showed that the topics can be utilized effectively to recommend items of interest with high accuracy. Translating our approach into a topic-driven recommendation system outperformed collaborative baseline methods and simple straw men in terms of average rank.

# Chapter 5

# MDP-based Itinerary Recommendation

In the previous chapter, we presented a generic approach using factored MDPs to model contextual topics of user sessions. The topics, in general, consist of several variables; however, some applications deal with simpler scenarios. In this chapter, we study a special non-factorized case of our proposed topic model from Chapter 4 in another interesting use case. The chapter is structured as follows. The first section motivates our approach for the task of itinerary recommendation using social media, and Section 5.2 describes the related work. The main approach is outlined in Section 5.3 followed by our empirical study including the methodology used for data preprocessing in Section 5.4. Section 5.5 concludes.

## 5.1 Motivation

Planning vacations is a complex decision problem. Many variables, such as the place(s) to visit, how many days to stay, the time spent at each location, and the overall travel budget, need to be controlled and arranged by the user. Automatically recommending travel itineraries would thus be a remedy for quickly converging on an individual trip that is tailored to the user's interests.

While on a trip, users frequently share their experiences on social media platforms e.g., by uploading photos of specific locations and times of day. Their uploaded data serves as an asset when it comes to gathering information on their journey. Therefore, online content has become an effective resource for travelers (Google, 2014). In addition to an increasing number of travel blogs, verticals providing reviews and recommendations of places, restaurants and hotels prove to be useful tools for planning trips and nights out. Nevertheless, the amount of content is increasing rapidly, and

it is becoming more and more difficult to find relevant information. Furthermore, services and resources do not cover a wide range of aspects exhaustively; instead, they often focus on narrow scopes to remain clearly segregated from other content providers. Thus, users who seek different types of information need to query various sites and aggregate the pieces of information themselves, which requires a significant amount of time and effort.

Meanwhile, the rise of digital photography through the widespread use of mobile devices and digital cameras has resulted in a great number of photos being shared online. Further, the growing acceptance of social platforms leads users to exposing more of their personal data online. Uploaded photos are mostly tagged by users with informational snippets and keywords which share the location, emotions, and so on with others. Popular online photo sharing sites collect millions of such photographs. A remarkable way of understanding itineraries is to study the photo streams of tourists from tourist zones.

In this chapter, we showcase how freely-available, user-tagged information on social media can be aggregated to recover tourist trajectories in cities. Our analysis is based on online photo streams of users that reflect (a possibly incomplete) sequence of locations visited during a trip, and we assume that these sequences indicate the overall trip satisfaction of the users. We thus turn Flickr[1], a popular photo-sharing site, into useful resource with which to reconstruct users' trips. Exploiting the assumption that photos of interesting places are uploaded more frequently than photos of uninteresting ones (Van Canneyt et al., 2011), Flickr proves useful in generating candidate lists of points of interests for any city. Moreover, many photos already come with geographic, temporal, and/or semantic annotations. Photos annotated with geo-coordinates can be accurately placed on a map, and, if the user also provides semantic tags, the content can be indexed and further processed by natural language processing techniques. However, Figure 5.1 shows that only a minority of

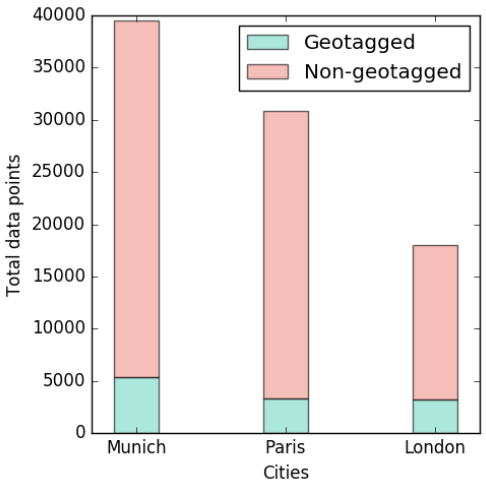



Figure 5.1: Fraction of geo-referenced images in our dataset from Flickr.

[1] www.flickr.com

photos from three European metropolises possess such descriptors, while the bulk of data needs an elaborate preprocessing.

Technically, touristic trips are considered to be sequential processes, where, at each stage of travel, a user chooses her next destination from a list of tourist points in the city. Additionally, the data provides implicit feedback on the user's preference for a place through the photographs she uploads on Flickr. We hence ground our solution in Markov Decision Processes (MDPs) that capture the sequential nature of itineraries. The tags attached to the photos provide the factors with which to generate possible configurations and prove crucial in contextualizing the proposed approach.

We employ a non-factorized version of the approach proposed in Chapter 4 in the task of itinerary recommendation. The topic of an itinerary is characterized via the place categories of candidate POI (Points of Interest) which render a univariate topic model problem. Therefore, the task of itinerary recommendation becomes a special case of the problem introduced in the previous chapter. We leverage social media to reverse engineer historic user itineraries and determine the main categories of interest for users in a tourist city. Additionally, we propose an ad-hoc personalization technique in order to incorporate the long-term preferences of users into the framework. As a result, the user receives a recommendation for a place corresponding to the place category obtained from the topic model, one which is the most relevant to her, both in terms of distance and her personal interests.

## 5.2 Related Work

Previously, "tourist guide" systems have been developed to work with data collected from GPS[2] devices (Loecher & Jebara, 2009; Wolf et al., 2001). However, recent studies attempt to infer meaningful information from user-generated content on social media. One of the earlier works in associating Flickr photos to physical locations is introduced by Crandall et al. (2009), who apply their techniques to extract landmarks at various granularity levels that correspond to a geo-spatial hierarchy. Cao et al. (2009) present a method that uses both logistic regression and kernel canonical correlation to enrich semantic information and location information based on image content. The tags assigned to Flickr photographs are further employed to extract place names, coordinates, and categories as well as popularity values (Popescu et al.,

[2]Global Positioning System

2008; Quack et al., 2008; Rattenbury et al., 2007). Computing co-occurrences between textual tags (Baba et al., 2010) and scene similarity between images (Hays & Efros, 2008) are other approaches to assign location to non-geotagged target photos.

The growing surge of travel data on social media platforms has resulted in many recent works on touristic place recommendations (Berjani & Strufe, 2011). Jiang et al. (2015) enhance collaborative filtering recommendations with topic models that consider different types of user preferences to exploit textual metadata associated with geo-tagged pictures on Flickr. Zhang & Wang (2015) propose an extension to collaborative retrieval model for POI recommendation, taking temporal information and social relations into account. Additionally, Foursquare[3] data is used to build a probabilistic generative framework that recommends tours based on user's preference, peer circle, travel transitions and popularity of venues (Rakesh et al., 2017). Others restrict their work to the geo-tagged points on Flickr to find shortest routes with the highest satisfaction for groups that have constraints of time, distance, and start and end destinations (Lim, 2016; Quercia et al., 2014).

While all these models capture many different aspects of a tourist movement, they fail to address sequentiality in travel itineraries. Nevertheless, probabilistic approaches are used in recommending the next POI and are either based on location services (Noulas et al., 2012; Sang et al., 2012) or social networks (Feng et al., 2015). Monreale et al. (2009) design a T-pattern decision tree to classify the trajectory patterns and Muntean et al. (2015) rank POIs using gradient boosted regression trees and ranking SVMs. Furthermore, Markov models are applied to GPS data (Ashbrook & Starner, 2003) and in combination with user preferences (Kurashima et al., 2010) to model travel behaviors. Zhang et al. (2015) go one step further to prune the search space and recommend sequential POIs considering their time constraints.

## 5.3 Modeling User Itineraries

In this section, we characterize our framework for recommending user itineraries based on Markov decision processes inspired from the topic model of the previous chapter. In this setting, the users upload photos from a set of different cities. Each city $c$ contains a set of $n_c$ points of interest which is represented by $L_c$. The photos are described by a set of attributes containing the timestamp of capture, the latitude and longitude of photo location, the title, the textual tag, and further description attached to the picture.

[3]https://foursquare.com

Additionally, the places are assigned with various categories which are encoded in a random variable $\mathcal{Z}$. However, unlike our topic model, instead of taking one value from the finite set of $dom(\mathcal{Z})$, a location can belong to several categories. Table 5.1 illustrates some POI examples with their categories. Let $\boldsymbol{z}^p \in \mathbb{R}^{|\operatorname{dom}(\mathcal{Z})|}$ be the category vector of a place $p$, indicating which categories are assigned to that place, i.e., for the $l$-th category, $z_l^p = 1$ if $p$ belongs to that category, and is zero otherwise. Hence, a session is described by a sequence of visited POIs and the topic of the formed itinerary is the probability distribution over $\operatorname{dom}(\mathcal{Z})$. Subsequently, the goal is to recommend an itinerary $I^\kappa = (p_1, \ldots, p_\kappa)$ of length $\kappa$ for each user in order to maximize her overall trip satisfaction, ensuring that she hits maximum number of places closer to her current location and to her liking.

Table 5.1: Place types for POIs across cities.

| **POI** | **Place Type** |
|---|---|
| London Eye | point of interest, establishment |
| St. Paul's Cathedral London | church, place of worship |
| BMW museum | museum, point of interest |
| Eiffel Tower | point of interest, establishment |
| Louvre Museum | point of interest, establishment |
| Olympic Park Munich | park, point of interest |

### 5.3.1 Simplified Topic Model

We benefit from a non-factorized version of the MDP formulation from Chapter 4 to specify the univariate topic model of itineraries. In this setting, the *states* represent the history of user travels, where a state $s \in \mathbb{S}$ is given by the sequence of size $k$ places the user has visited, $s = (p_1, \ldots, p_k)$. An *action* $a \in \operatorname{dom}(\mathcal{Z})$ is the set of all POI categories available in the city where the user is visiting. The *transition* function $\mathcal{P}(s, a, s')$ models the probability of going to another place given the current location and the recommended place category, where $s' = (p_2, \ldots, p_k, p_{k+1})$. For each state that the user enters on taking a particular action, she gets an immediate reward signal from the *reward* function $\mathcal{R}(s, a)$. A higher reward is granted when the transition is present in the sequence of places observed for all the users for that city.

The described problem is equivalent to one of the instances of the factored MDP in the previous chapter, therefore, the state-value functions are obtained in a similar fashion. First, a maximum-likelihood method is used to estimate the transition function based on the user travel data, which is given by

$$\mathcal{P}(s,a,s') = \frac{\text{freq}(s'|s,a)}{\sum_{s''}\text{freq}(s''|s,a)}, \quad \text{s.t.} \sum_{s''}\mathcal{P}(s,a,s'') = 1.$$

The freq function yields the frequency of occurrences of the observed sequences in the data. This conveys that the probability that a user, whose past POI visits cover the places $\{p_1,...,p_k\}$, will cover a new place $p_{k+1}$ after choosing some action $a$, where $z_a^{p_{k+1}} = 1$. The non-zero transitions take place when $(s,s')$ appears in the dataset and $a$ is a place category of $s'$, and is zero otherwise.

The immediate reward after taking action $a$ in state $s$ is given by the reward function $\mathcal{R}(s,a)$, which indicates whether the recommended action is beneficial for the traveler in the short term. The reward is simply inferred by the number of occurrence of state-action sequences in the training data

$$R(s,a) = \frac{\text{freq}(s,a)}{\text{freq}(s)}.$$

Consequently, the resulting MDP can be optimized using reinforcement learning methods such as value iteration to compute the value function $V(s)$. According to Section 4.3.2, the optimal state-value function $Q(s,a)$ is derived as follow

$$Q(s,a) = \mathcal{R}(s,a) + \gamma\sum_{s'}\mathcal{P}(s,a,s')V^*(s'). \quad \text{(cmp. Equation (4.3))}$$

The Q-values are proportional to the probability that the user visits a POI of place category $a$, given the sequence of visited locations in $s$. Hence, a higher value of $Q(s,a)$ indicates a higher likelihood of observing the transition from $s$ with action $a$.

### 5.3.2 Multi-step POI Recommendation

We use the *softmax* function to approximate the probability distribution over the place categories from the learned Q-values for each state

$$P(a|s) = \frac{\exp\{Q(s,a)\}}{\sum_{a'}\exp\{Q(s,a')\}}.$$

The action with the highest probability, $a^* = \arg\max_a P(a|s)$, is recommended at state $s$. However, the system is required to consider various places associated with

this category to recommend a specific place. We take the distance factor of POIs into account in order to predict the next place of interest. Considering all places corresponding to the optimal policy, the recommended place $p^*$ is the place closest in distance to the current state. Since each state consists of a sequence of places $(p_1, p_2, \ldots, p_k)$, the distance from the last place $p_k$ is evaluated. Therefore, the recommended POI is a place from $L_c$ that corresponds to the place category of the optimal action and minimizes the Euclidean distance from the last place $p_k$ in the current state

$$p^* = \arg \min_{\substack{p \in L_c \\ z^p_{a^*}=1}} \text{dist}(p_k, p),$$

where function dist computes the distance measure. Subsequently, by appending the determined place to the current state, the next POI is recommended from the same procedure until the path of size $\kappa$ is complete.

### 5.3.3 Online Personalization

We further aim to personalize the recommendation model by applying two techniques for inferring user preferences from her travel history: duration-based user interests as introduced by Lim et al. (2015) and frequency-based user interests.

**Duration-based User Preference.** Each location $p$ in the user travel history contains an arrival time $t^{arr}_p$ and a departure time $t^{dep}_p$. The duration-based user preference $\Omega_{dur}(u, a)$, for user $u \in \mathbb{U}$ and category $a$, is given by the fraction of time the user spent at each of the POIs from the category $a$ in her travel history,

$$\Omega_{dur}(u, a) = \sum_{\substack{p \in L_u \\ z^p_a=1}} (t^{dep}_p - t^{arr}_p),$$

where $L_u$ consists of all the locations visited by user $u$. These preferences are then normalized to the interval of $[0, 1]$ for each user, by converting the preference score to a probability distribution. As a result, the more time a user spends at a POI of a place category, the more likely it is that the user is interested in that specific category.

**Frequency-based User Preference.** In this method, the user preferences are inferred from the number of times a user has visited POIs of a certain category (Lim et al., 2015). The rationale is that the more times a user visits places of a certain

category, the more interested this user is in that category. The frequency-based user preference is given by

$$\Omega_{frq}(u, a) = \sum_{\substack{p \in L_u \\ z_a^p = 1}} \text{freq}(p),$$

which is further normalized between zero and one for each user.

The preference values obtained from either of both techniques, form a preference vector for each user, e.g., $\boldsymbol{\omega}_u = \{a_1 : 0.6, a_2 : 0.01, \cdots, a_{|dom(\mathcal{Z})|} : 0.3\}$, where the sum of the elements of the vector is one. We incorporate the individual preferences into our model at the time of recommending places for the optimal category $a^*$. We assign a score to each place proportional to the weighted sum of the distance and the preference associated with its other categories,

$$p^* = \arg\max_{\substack{p \in L_c \\ z_{a^*}^p = 1}} \left( (1 - \zeta) \cdot \frac{1}{\text{dist}(p_k, p)} + \zeta \cdot (\boldsymbol{\omega}_u \cdot \boldsymbol{z}^p) \right),$$

where $\zeta \in [0, 1]$ is the personalization coefficient. Consequentially, a place is recommended to a user which is closer in distance and also belongs to other categories that are preferred by the user.

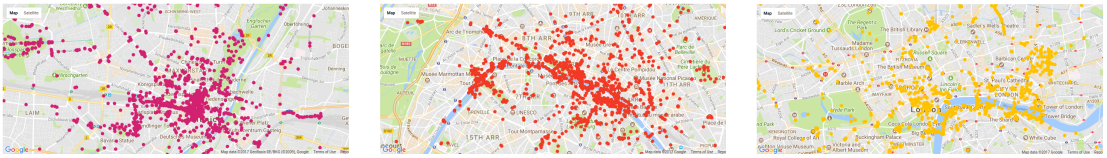
## 5.4 Empirical Study

In this section, we first describe the data pre-processing pipeline from acquisition of data to create itineraries of user trips. Subsequently, the performance evaluation of our approach is provided in details followed by the obtained results.

### 5.4.1 Data Extraction and Analysis

Our system focuses on extracting and discovering a large number of trips using geo-temporal data from Flickr. Flickr consists of over five billion photographs that many of them are time-stamped and additionally are annotated with semantic data, such as tags and titles associated with them. First, we collect 44051, 22970, 42104 photographs of three popular cities, *Munich*, *London*, and *Paris*, along with their meta-data, using the public API of Flickr. Figure 5.2 shows the spread of POIs across the three cities. However, as mentioned earlier, not all of this data is geo-referenced, and a significant portion of the photographs are without geo-coordinates.

Restricting ourselves to only the geo-referenced pictures would significantly decrease the coverage of our approach. Therefore, we utilize the meta-data associated



(a) Munich (b) Paris (c) London

Figure 5.2: Flickr POIs in Munich, Paris and London.

with photographs to infer their locations. The user provided textual tags often contain event and geospatial information, which could be used for inferring the location of non-geotagged data. We exploit the co-occurrence statistics of words in low-dimensional vector space by using the latent semantic analysis similarity (Dumais, 2004) between tags of a target non-geotagged photograph and each of the geo-tagged photographs. Using this, each non-geotagged photograph is assigned a location from the highest similarity score of its tag, provided it is above a certain similarity threshold. Nonetheless, most of the photographs have linguistically noisy tags or tags with no location information. Therefore data points without a place information in the tags are dropped for further analysis.

In order to maintain a high quality mapping from geo-coordinates of photos to place names, we query the free version of Google places API and obtain a list of POIs in each city. This is with the assumption that many touristic points are already available on Google Maps and are highly reliable. The place name and place category (e.g., museum, church, etc.) of the location coordinate is obtained by calling the API with *latitude*, *longitude*, *ranking criteria* and *search radius* around the location coordinates. We rank these results by their touristic prominence and look for places within 100m of the coordinates. Despite these techniques, a small fraction of the coordinates never get a place name from the Google places API. We thus assign their place name manually using the Google Maps interface.

Upon obtaining the POI names, we use the Flickr data to emulate tourist behavior. The first step is to remove all travel points falling outside the bounding box of a city. In addition, our method aims to recommend only single-day itineraries. But, most of the photograph sequences for each user contain data from many days and cannot directly be used for training the model. Therefore, the sequences of photographs for more than one day are split by their date time into single day sequences. Moreover, it is important to differentiate between a resident and a tourist in a city by checking the number of POIs covered by them. A resident would exhibit travel movements slower than a tourist. Not being constrained to cover maximum places in a single

day, residents usually cover only 1 or 2 tourist destinations. Hence, we discard travel paths consisting of less than 3 unique POIs. As a result, a total of 17904, 6000 and 9032 photographs are left for Munich, London and Paris, respectively.

### 5.4.2 Experimental Setup

We first analyze the path accuracy of our recommended itinerary by varying the amount of user history encoded in each state, that is the order of Markov chain. On obtaining the optimal length of user history, we further compare the performance of our approach across three cities against several baselines. Moreover, we compute accuracy@$K$ when the recommended places are among the top-$K$ closest places to the current POI. All the experiments are conducted first without the effects of personalization and then when user preferences are included.

Given the sequential travel paths of users, we use a time-series leave-one-out cross validation technique for tuning the parameters of our model. The dataset comprises of users covering POIs within a day as well as multiple days. For the users with multi-day itineraries, we keep the last day of travel sequence as the test set and the remaining as training set. For users that have only traveled on one day, the travel sequence is split into 60%-40%-of POIs into training and test sets. For empirical estimation of different parameters, a 10-fold cross validation system is employed on the training data, and the final system is validated on the test set.

A significant number of POIs in our dataset do not correspond to the transitions defined in the MDP due to the sparsity issue. We leverage this fact to improve the space and computation time in our experiments, by only keeping the non-zero transition probabilities of states for which a transition occurred in our training data. This significantly reduces the size of the transition matrix. At the same time, the computation of $p^*$ is carried out through parallel processing and reduces time by almost 75% on a quad-core processor (Shani et al., 2005). In addition, in order to compare the recommended itinerary $I^\kappa$ to the actual itinerary in the test set, we eliminate all the places from the test set that are not present in the training data.

For evaluating the recommended itineraries, we vary the length of the recommended path $\kappa$ within the range of one to six, $\kappa \in \{1, 2, 3, 4, 5, 6\}$, where a path of length 1 would contain two place locations $(p_1, p_2)$ and so on. The performance criteria evaluate how many of the recommended places are present in the test paths, while taking into account the order of POIs in those paths. We therefore, introduce two metrics to evaluate the performance of the path (itinerary) recommendation methods:

*exact path accuracy* and *partial path accuracy*. The exact path accuracy of total $N$ paths, $\{I_1, \ldots, I_N\}$, from the test set is given by

$$ACC_{exact} = \frac{1}{N} \sum_{l=1}^{N} \sum_{\kappa=1}^{|I_l|} \frac{h(I_l^\kappa)}{|I_l|},$$

where $h(I_l^\kappa) = 1$ if subpath of size $\kappa$ from path $I_l$ is predicted correctly from the model, and is zero otherwise. This accuracy is defined as the percentage of exact match of the recommended pair to the subpath of the above mentioned test paths, where each matching subpath is recorded as a positive hit. The overall accuracy is given by averaging the accuracy of all the $N$ paths.

Additionally, we compute the partial path accuracy which assigns a hit if at least one subpath of the test path matches the recommended pair

$$ACC_{partial} = \frac{1}{N} \sum_{l=1}^{N} \sum_{\kappa=1}^{|I_l|} \frac{\Lambda(I_l^\kappa)}{|I_l|}, \quad \Lambda(I_l^\kappa) = \begin{cases} |I_l|, & \exists \kappa \quad h(I_l^\kappa) \geq 1 \\ 0, & \text{otherwise.} \end{cases}$$

We compare our approach with standard graph search algorithms as baselines and the non-personalized MDP policy. We start from the simplest, Breadth First Search (**BFS**) and evaluate more sophisticated algorithms of **Dijkstra**, **heuristic** search and $A^*$. For Dijkstra and $A^*$, the edge cost is given by the distance between the locations. For each of the baseline algorithms, we look for paths starting from $p_1$ corresponding to the starting POI in the test set and iteratively choose a next place to visit, till the last POI $p_\kappa$ in the itinerary is found. The heuristic used in $A^*$ and heuristic search is the *Manhattan Distance* between the current place node and the goal node. Recommendation pairs are obtained from each of these paths by taking consecutive locations. For each baseline, the same validation as the MDP model against the test set is performed to get the exact and partial path accuracy scores.

Table 5.2: Variation of partial path accuracy for top-7 closest POIs w.r.t. user history.

| **Path Length** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1st order | 0.041 | 0.041 | 0.042 | 0.042 | 0.041 | 0.034 |
| 2nd order | **0.098** | **0.090** | **0.096** | **0.106** | **0.100** | **0.103** |
| 3rd order | 0.097 | **0.090** | 0.093 | 0.105 | 0.090 | 0.087 |
| 4th order | 0.089 | 0.084 | 0.083 | 0.094 | 0.077 | 0.060 |
| 5th order | 0.074 | 0.071 | 0.058 | 0.072 | 0.070 | 0.058 |

### 5.4.3 Results and Discussion

We first study the impact of path history or the order of Markov chain on the prediction accuracy. Note that history length is the number of visited POIs encoded in the state, while path length is the number of next consecutive POIs to recommend in the itinerary. A path length of one hence stands for step-by-step recommendation. Table 5.2 captures the relation of path history to the performance of the system. There is a significant improvement in performance as we change the path history from one to two. Nonetheless, the performance shows very less improvement as the path history is increased up to length five. This finding is primarily due to the fact that many of the travel sequences do not cover places more than three on a single day. Besides, we observe that, as the path history increases, the number of successors in the transition decreases (cmp. Table 4.1).

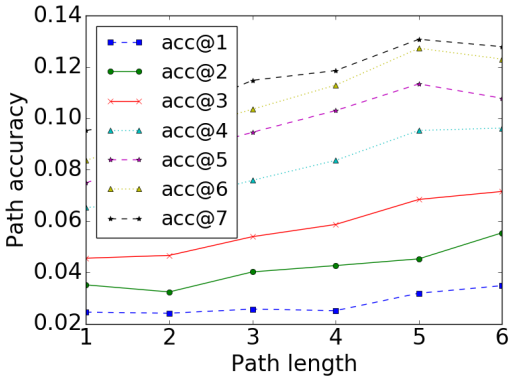



Figure 5.3: Variation in partial path accuracy with regard to $K$ in accuracy@$K$.

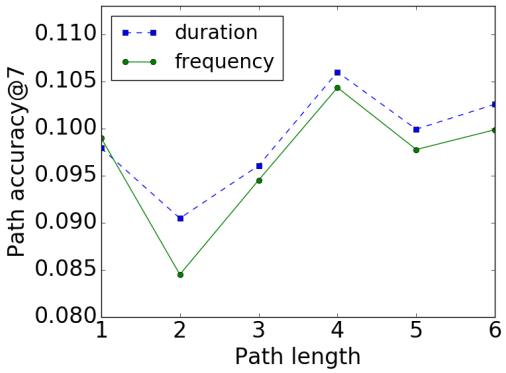
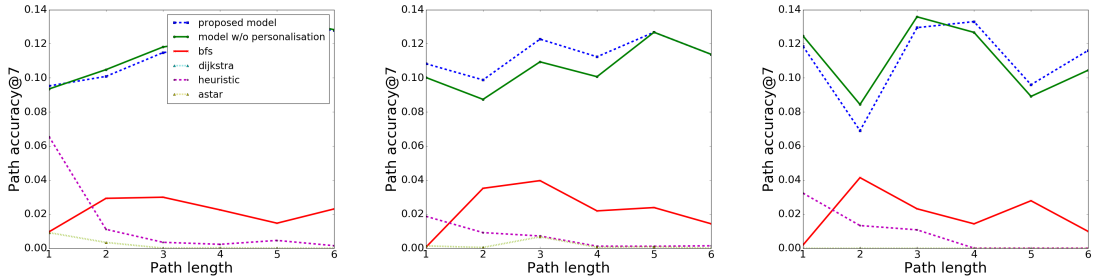



Figure 5.4: Partial path accuracy for two personalisation techniques.

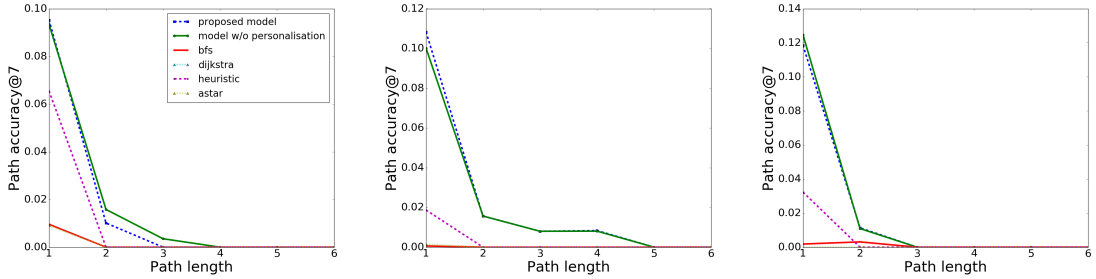
In addition, we demonstrate the increase in performance of our approach by recommending the $K$ closest places in Figure 5.3 for top-$K$ recommendation in Munich. The results are reported for partial path accuracy. As expected, increasing $K$ enhances the performance of the system since the size of candidate list grows. All top-$K$ places correspond to the optimal place category obtained from value iteration. Thus, the more flexible a traveler is to multiple recommendation options at the current POI, the higher the likelihood to recommend the best possible place entailing the user's travel preferences.

We further compare the performance of the two introduced personalization techniques, i.e., duration- and frequency-based user preferences as shown in Figure 5.4. The former consistently outperforms its counterpart and proves more accurate with




(a) Munich (b) Paris (c) London

Figure 5.5: Personalized recommendation vs. baselines (partial path accuracy).




(a) Munich (b) Paris (c) London

Figure 5.6: Personalized recommendation vs. baselines (exact path accuracy).

respect to real-life tours of users, compared to the frequency-based personalization. Additionally, the personalization factor $\zeta$ can be varied to balance the distance from the current state and the user personal interest. A value of $\zeta = 0.35$ yields the highest partial path accuracy during cross validation.

Moreover, Figures 5.5 and 5.6 illustrate the performance compared to baselines in terms of partial and exact path accuracy, respectively. Our proposed approach is plotted with and without personalization counterpart. The figures confirm that there is an average improvement of 10.5% of the MDP-based method over the path planning baselines, across the three cities. The effect of personalization over the non-personalized recommendations is still not very significant in our experiments. However, there is a slight improvement for the shorter tour recommendations in Paris. Munich and London show almost the same performance between the personalized and non-personalized versions for all the path lengths.

One of the reasons for the lower impact of personalisation is the lack of data points in the training set as well as per user. The personalized recommendation accuracy is the average partial path accuracy across all the users in the validation or test set. Most of Flickr users in our dataset have photographs for a single day, with the photo

album size of averaging out to less than eight POI pictures. On splitting the data into train and test set, enough pictures are not left for learning the user preferences in the training. Also, since the travel sequences are short, there are less chances of place categories observed in the training data repeating in the test set. Thus, the user preference learned in the training set does not relate well to the preferences extracted from the test set. Consequentially, the overall low accuracy is due to the limited quality data, sparsity of transitions, and minimal manual intervention in data processing. Nevertheless, the computationally inexpensive MDP-based personalized recommender system outperforms the standard path planning algorithms and serves as a promising technique for modeling user behavior in travel recommendation.

## 5.5 Conclusion

In this chapter, we presented an itinerary recommendation approach based on MDPs as a special case and modified version of our topic model from Chapter 4. Our approach took the sequential travel histories and preferences of users into account when recommending sequences of points of interest. We employed both photo-sharing sites (Flickr), as well as the large abundance of geographical information on web-mapping services, to extract supplementary knowledge for our system. As opposed to many of the existing systems proposed earlier, our model was not restricted to the geo-tagged pictures on Flickr but, instead, tracked tourist movements from the time-stamps extracted from the data, i.e., the recommended travel plans emulated the trip plans of tourists.

We compared our approach empirically to various path planning algorithms on data from three European tourist cities, i.e., Munich, Paris and London, and observed that the predicted itineraries were more accurate than those produced by the path-planning algorithms. We further showed that personalization is beneficial in certain scenarios, particularly when sufficient training data is available. Nevertheless, an effective recommendation requires a universal method with which to model personalized interests in the same short-term model. We cope with that problem in subsequent chapters.

# Chapter 6

# Unified Contextual Bandit Models

In the previous two chapters, we successfully phrased recommendation tasks as sequential decision making processes, allowing us to exploit the session-based or short-term interests of the users. Further, we introduced an ad-hoc technique for accommodating personalization at the end of Chapter 5. From now on, we focus on developing approaches which are able to learn both personalized preferences as well as short-term interests in the same model.

In this chapter, we propose a unified contextual bandit framework for recommendation problems which is able to capture both the short- and long-term interests of users. We introduce the problem and the related work in the first section. Section 6.2 derives a generalized optimization problem in the dual space, which is then followed by its instantiations for regression and classification scenarios. Section 6.3 contains our main contribution and presents the combination of short-term and long-term recommender systems within the unified framework, along with potential optimization methods. Possible extensions for our proposed approach are discussed in Section 6.4. We present empirical studies in Section 6.5, and Section 6.6 concludes.

## 6.1 Motivation

Recommendation systems are designed to serve user needs. While some needs arise on short notice due to weather changes, news articles, or advertisements, others manifest over a long time span and express general interests in, for example, cars, stock markets, or garments in favored colors. User needs are therefore driven by individual *long-term* as well as collective *short-term* interests, which the latter being highly influenced by the zeitgeist and common trends. On the other hand, both user and item sets are highly dynamic and change rapidly over time. Hence, a qualified rec-

ommendation system requires a universal method to model both views at the same time which also generalizes well for various situation and settings.

Recall that collaborative filtering-based methods (Hu et al., 2008; Koren et al., 2009) focus only on the personalized, long-term preferences of users, while others aim to model the topics of user sessions and focus on short-term interests (Barbieri et al., 2013; Gaonkar et al., 2018; Tavakol & Brefeld, 2014; Wang & Blei, 2011). However, context-aware approaches (Li et al., 2010) and their kernelized variants (Deshmukh et al., 2017; Valko et al., 2013; Vanchinathan et al., 2014), may be leveraged to respond to both aspects. Some recent works study context-aware bandits for personalization purposes. Collaborative contextual bandits are introduced by Wu et al. (2016a), where the context and payoffs are shared among the neighboring users to reduce learning complexity and overall regret. In addition, contextual bandits are used to learn the latent structure of users in probabilistic settings to cope with cold start scenarios (Tang et al., 2015; Zhou & Brunskill, 2016).

Nevertheless, these methods are usually tailored to solve very specific recommendation tasks and may not be applicable to different scenarios. Therefore, a more flexible and comprehensive approach is required to cope with the diverse facets of recommendation problems. In this chapter, we present a unified contextual bandit framework to capture the short- and long-term interests of users. The underlying model consists of contextual (the short-term) and individual user-based (the long-term) parts in order to determine the expected reward,

$$\mathbb{E}[r_t|b_i, u_j] = \underbrace{\boldsymbol{\theta}_i^\top \boldsymbol{s}_t + \ e_i}_{\text{short-term}} + \underbrace{\boldsymbol{\beta}_j^\top \boldsymbol{x}_i}_{\text{long-term}} \ .$$

In the above composition, the expected reward is computed from two distinct terms. The first component models the short-term behavior for a given context $\boldsymbol{s}_t$ at time $t$. The context in this chapter determines the recent trend or the topical interest of the current session. In the short-term part, the outcome of choosing any arm $b_i$ for the given context $\boldsymbol{s}_t$ is specified linearly and by its weight vector, $\boldsymbol{\theta}_i$. The long-term model, on the other hand, allows individual interests for user $u_j$ to be captured across item features, $\boldsymbol{x}_i$ (describing item $b_i$). We propose to combine the short-term and long-term recommendations into one unified model. Note that $e_i$ acts as constant term in the linear model for each arm.

The optimization is performed simultaneously for all of the arms so that the short-term part serves as a joint popularity-based predictor while the long-term part acts as an individual offset. The model is devised in the dual space, and all of

the derivations are consequentially carried out using Fenchel–Legendre conjugates of the loss functions, which renders our approach to serve as a framework for a wide range of loss functions and settings. We detail two instantiations for regression and classification scenarios and obtain LinUCB (Li et al., 2010) and LogUCB (Mahajan et al., 2012) for these special cases, respectively. The resulting general and unified framework allows contextual bandits to adapt quickly to different applications.

## 6.2 General Optimization Problem

Following the notations from Section 3.1, we focus on sequential recommendation systems for $m$ users, $\mathbb{U} = \{u_1, u_2, \cdots, u_m\}$, and $n$ items, $\mathbb{B} = \{b_1, b_2, \cdots, b_n\}$. Every item $b_i$ is characterized by a set of attributes given by a feature vector $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$. At each time step $t$, the goal of the system is to recommend items for the actual context of the ongoing session. In the problem setting of this chapter, the context $s_t$ is determined using a vectorized form, which is given by a feature vector $\boldsymbol{s}_t \in \mathbb{R}^{d_s}$. In this section, we show how to derive the general optimization framework for linear bandits in the dual space taking only the short-term component into consideration.

Assume that the learning procedure for every item (arm) consists of $T_i$ trials, and for every context $\boldsymbol{s}_t$ the reward $r_t$ is obtained. Therefore, $\{(\boldsymbol{s}_t, r_t)\}_{t=1}^{T_i}$ is the set of $T_i$ samples of contexts and their corresponding rewards. The reward $r_t$ reflects the user feedback with respect to the recommended items at time $t$; its domain depends on the application at hand: e.g., $r_t \in \{1, 0\}$ for click/no click signals. We thus design a contextual bandit framework with linear payoff function for arm $b_i$ as follows

$$H^{(i)}_{\boldsymbol{\theta}_i, e_i}(\boldsymbol{s}_t) = \boldsymbol{\theta}_i^\top \boldsymbol{s}_t + e_i,$$

where hypothesis $H$ predicts the expected payoff of the $i$-th arm, i.e., $\mathbb{E}[r_t|b_i]$, given the model parameters $\boldsymbol{\theta}_i$ and $e_i$. The bandit framework learns every hypothesis $H^{(i)}$ independently of the other arms. We therefore discard the index $i$ in the remainder of this section for ease of notation and address the problem for a single arm.

Given an arbitrary loss function $\mathcal{L}(\cdot, r_t)$, and using $\ell_2-$norm regularizer, the optimization problem can be stated as

$$\inf_{\boldsymbol{\theta}, e} \quad \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\theta}^\top \boldsymbol{s}_t + e, r_t) + \frac{\nu}{2} \|\boldsymbol{\theta}\|^2,$$

where $\nu > 0$ is the regularization parameter. We rewrite the objective function by incorporating $v_t$ as the shorthand for the predicted payoff, and using $C = \frac{1}{\nu T}$ gives

$$\inf_{\boldsymbol{\theta},\boldsymbol{v},e} \quad C\sum_{t=1}^{T}\mathcal{L}(v_t, r_t) + \frac{1}{2}\|\boldsymbol{\theta}\|^2 \quad s.t. \quad \forall t : \boldsymbol{\theta}^\top \boldsymbol{s}_t + e = v_t.$$

The equivalent unconstrained problem is derived by incorporating Lagrange multipliers, $\boldsymbol{\alpha} \in \mathbb{R}^T$,

$$\sup_{\boldsymbol{\alpha}} \quad \inf_{\boldsymbol{\theta},\boldsymbol{v},e} \quad C\sum_{t=1}^{T}\mathcal{L}(v_t, r_t) + \frac{1}{2}\|\boldsymbol{\theta}\|^2 - \sum_{t=1}^{T}\alpha_t(\boldsymbol{\theta}^\top \boldsymbol{s}_t + e - v_t).$$

Setting the partial derivatives of the objective function with respect to $e$ and $\boldsymbol{\theta}$ to zero, leads to the following conditions

$$\mathbf{1}^\top \boldsymbol{\alpha} = 0 \qquad \text{and} \qquad \boldsymbol{\theta} = \sum_{t=1}^{T}\alpha_t \boldsymbol{s}_t = S^\top \boldsymbol{\alpha},$$

where $S \in \mathbb{R}^{T \times d_s}$ is the design matrix given by the training data. Substituting the optimality conditions into the optimization function yields

$$\sup_{\boldsymbol{\alpha},\mathbf{1}^\top\boldsymbol{\alpha}=0} \quad \inf_{v_t} \quad C\sum_{t=1}^{T}\big(\mathcal{L}(v_t, r_t) + \frac{1}{C}\alpha_t v_t\big) - \frac{1}{2}\boldsymbol{\alpha}^\top SS^\top \boldsymbol{\alpha}.$$

Subsequently, we move the infimum inside the summation as it solely depends on the first term. Using $\inf_w f(w) = -\sup_w -f(w)$, we obtain

$$\sup_{\boldsymbol{\alpha},\mathbf{1}^\top\boldsymbol{\alpha}=0} \quad -C\sum_{t=1}^{T}\sup_{v_t}\big(-\frac{\alpha_t}{C}v_t - \mathcal{L}(v_t, r_t)\big) - \frac{1}{2}\boldsymbol{\alpha}^\top SS^\top \boldsymbol{\alpha}.$$

Recall that the Fenchel-Legendre conjugate of a function $f$ is defined as $f^*(\mathbf{w}) = \sup_{\boldsymbol{y}} \mathbf{w}^\top \boldsymbol{y} - f(\boldsymbol{y})$ (Boyd & Vandenberghe, 2004). Thus, the dual loss turns into

$$\mathcal{L}^*\big(-\frac{\alpha_t}{C}, r_t\big) = \sup_{v_t} -\frac{\alpha_t}{C}v_t - \mathcal{L}(v_t, r_t).$$

For a comprehensive list of dual losses see (Rifkin & Lippert, 2007). The generalized optimization problem in dual space therefore reduces to

$$\sup_{\boldsymbol{\alpha},\mathbf{1}^\top\boldsymbol{\alpha}=0} \quad -C\sum_{t=1}^{T}\mathcal{L}^*\big(-\frac{\alpha_t}{C}, r_t\big) - \frac{1}{2}\boldsymbol{\alpha}^\top SS^\top \boldsymbol{\alpha}. \tag{6.1}$$

### 6.2.1 Upper Confidence Bound

The challenge in multi-armed bandits is to balance exploration and exploitation in order to minimize the regret. Auer (2003) demonstrates that confidence bounds provide useful means to balance the two oppositional strategies. The idea is to use the predicted reward together with its confidence interval to reflect the uncertainty of the model given the actual context, and is called Upper Confidence Bound (UCB). Thus, gathering enough information to reduce the uncertainty in a multi-armed bandit is as important as maximizing the reward.

In our contextual bandit, the expected payoff is approximated by a linear model with an arbitrary loss function where a general optimization approach is used to estimate the parameters. The uncertainty $U$ of the obtained value for each arm is therefore proportional to the standard deviation $\sigma$ of the expected payoff, $U = o\sigma$, where the variance $\sigma^2$ is estimated from training points in neighbouring contexts as well as the model parameters, and $o$ is a constant. The uncertainty is added as an upper bound to the prediction (the expected reward) to produce a confidence bound for the selection strategy across the arms. The computation of the confidence bound depends on the choice of the loss function. We illustrate the obtained bounds for two special cases in the remainder.

### 6.2.2 Instantiations

In this section, we represent two well-known optimization problems which can be recovered from Equation (6.1) by substituting the corresponding loss functions. The instantiations illustrate how a general platform simplifies comparing and analyzing various loss functions in different situations.

**Squared Loss.** The first instantiation deals with regression scenarios for real-valued payoffs, $r_t \in \mathbb{R}$. The squared loss function and its conjugate are given by

$$\mathcal{L}(v_t, r_t) = \frac{1}{2}(v_t - r_t)^2 \quad \text{and} \quad \mathcal{L}^*(w_t, r_t) = \frac{1}{2}w_t^2 + w_t r_t,$$

where the latter can be rewritten as

$$\mathcal{L}^*\big(-\frac{\alpha_t}{C}, r_t\big) = \frac{1}{2C^2}\alpha_t^2 - \frac{1}{C}\alpha_t r_t.$$

Incorporating the conjugate loss function into Equation (6.1) gives

$$\max_{\boldsymbol{\alpha}, \mathbf{1}^\top\boldsymbol{\alpha}=0} \quad -\frac{1}{2C}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} + \boldsymbol{\alpha}^\top\boldsymbol{r} - \frac{1}{2}\boldsymbol{\alpha}^\top SS^\top\boldsymbol{\alpha}, \tag{6.2}$$

where the supremum becomes a maximum as the loss function is continuous. On the other hand, the equivalent problem in the primal space corresponds to ridge regression where parameters are determined by optimizing the regularized sum of squared errors,

$$\min_{\boldsymbol{\theta},e} \quad \frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}(\boldsymbol{\theta}^\top \boldsymbol{s}_t + e - r_t)^2 + \frac{\nu}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}.$$

To obtain $\boldsymbol{\theta}$, we set its gradient to zero which yields

$$\boldsymbol{\theta} = -\frac{1}{\nu T}\sum_{t=1}^{T}(\boldsymbol{\theta}^\top \boldsymbol{s}_t + e - r_t)\boldsymbol{s}_t.$$

Furthermore, the relation $\alpha_t = -\frac{1}{\nu T}(\boldsymbol{\theta}^\top \boldsymbol{s}_t + e - r_t)$ holds and we have

$$\boldsymbol{\theta} = \sum_{t=1}^{T}\alpha_t \boldsymbol{s}_t = S^\top \boldsymbol{\alpha}.$$

For the threshold parameter $e$, we obtain the equation $\frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{\theta}^\top \boldsymbol{s}_t + e - r_t) = 0$, and thus arrive at the optimality conditions

$$-\nu\sum_{t=1}^{T}\alpha_t = 0 \quad \Rightarrow \quad \mathbf{1}^\top\boldsymbol{\alpha} = 0.$$

Expanding the terms in the summation and substituting the optimality conditions leads to the optimization problem

$$\min_{\boldsymbol{\alpha},\mathbf{1}^\top\boldsymbol{\alpha}=0} \quad C\big(\frac{1}{2}\boldsymbol{\alpha}^\top SS^\top SS^\top\boldsymbol{\alpha} - \boldsymbol{r}^\top SS^\top\boldsymbol{\alpha}\big) + \frac{1}{2}\boldsymbol{\alpha}^\top SS^\top\boldsymbol{\alpha},$$

where $C = \frac{1}{\nu T}$. By removing $SS^\top$ from all the terms and converting the minimization into a maximization, we have

$$\max_{\boldsymbol{\alpha},\mathbf{1}^\top\boldsymbol{\alpha}=0} \quad -\frac{1}{2}\boldsymbol{\alpha}^\top SS^\top\boldsymbol{\alpha} + \boldsymbol{r}^\top\boldsymbol{\alpha} - \frac{1}{2C}\boldsymbol{\alpha}^\top\boldsymbol{\alpha},$$

which precisely recovers Equation (6.2). In addition, the confidence bound for the linear bandit with square loss is given by (cmp. also (Li et al., 2010))

$$U = o\sqrt{\boldsymbol{s}_t^\top(S^\top S + \nu I)^{-1}\boldsymbol{s}_t}.$$

**Logistic Loss.** As the second instantiation, we derive the optimization problem for the logistic loss which in our setting is defined as

$$\mathcal{L}(v_t, r_t) = \log\left(1 + \exp(-v_t r_t)\right),$$

where $v_t \in \{0, 1\}$ The Fenchel-Legendre conjugate of this loss function is given by

$$\mathcal{L}^*(-\frac{\alpha_t}{r_t}, r_t) = \left(1 - \frac{\alpha_t}{Cr_t}\right) \log\left(1 - \frac{\alpha_t}{Cr_t}\right) + \frac{\alpha_t}{Cr_t} \log\left(\frac{\alpha_t}{Cr_t}\right),$$

and incorporating the latter into Equation (6.1) leads to Equation (6.3)

$$\max_{\boldsymbol{\alpha}, \mathbf{1}^\top \boldsymbol{\alpha} = 0} -C \sum_{t=1}^{T} \left[\left(1 - \frac{\alpha_t}{Cr_t}\right) \log\left(1 - \frac{\alpha_t}{Cr_t}\right) + \frac{\alpha_t}{Cr_t} \log\left(\frac{\alpha_t}{Cr_t}\right)\right] - \frac{1}{2} \boldsymbol{\alpha}^\top S S^\top \boldsymbol{\alpha}. \quad (6.3)$$

On the other hand, assume to optimize the logistic regression problem according to the dual formulation proposed in Keerthi et al. (2005), that gives

$$\min_{\hat{\boldsymbol{\alpha}}} \quad \frac{1}{2} \left\| \sum_{t=1}^{T} \hat{\alpha}_t r_t \boldsymbol{s}_t \right\|^2 + C \sum_{t=1}^{T} G(\frac{\hat{\alpha}_t}{C}), \quad s.t. \quad \sum_{t=1}^{T} \hat{\alpha}_t r_t = 0,$$

where $G(w) = w \log w + (1 - w) \log(1 - w)$. Setting $\alpha_t = \hat{\alpha}_t r_t$, and converting the minimization into a maximization recovers Equation (6.3).

The covariance of the parameters for the logistic regression problem is given by $\Sigma = S^T W S$, where $W$ is the diagonal matrix of $f(1 - f)$, and $f$ is computed by the sigmoid function, i.e., $f = \text{sigmoid}(S^\top \boldsymbol{\theta})$. Consequentially, the lower and upper confidence bounds are given by

$$U_{low} = \text{sigmoid}(\hat{r_t} - o\sqrt{\boldsymbol{s}_t^\top \Sigma^{-1} \boldsymbol{s}_t}), \quad U_{up} = \text{sigmoid}\left(\hat{r_t} + o\sqrt{\boldsymbol{s}_t^\top \Sigma^{-1} \boldsymbol{s}_t}\right),$$

respectively (Dybowski & Roberts, 2001), and $\hat{r_t}$ is the current estimate of outcome from the model. The confidence bound for the contextual bandits is therefore obtained from $U = U_{up} - U_{low}$. Mahajan et al. (2012) also introduce a variance approximation technique to obtain the confidence bound of the logistic loss for probit functions.

## 6.3 Unified Short- and Long-term Model

In our setting, personalized and user specific information cannot simply be incorporated into the bandits by another type of context. Instead, we suggest to incorporate a long-term model into the short-term approach of the previous section. As a result, we are able to model the behavior of users for the recommendation process as well.

The long-term component captures the interests of user $u_j$ for every arm $b_i$. We thus assume a separate set of parameters for the personalized part of the model, given by $\boldsymbol{\beta}_j \in \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m\}$, where $\boldsymbol{\beta}_j \in \mathbb{R}^{d_x}$ is the weight vector of user $u_j$ and $d_x$ is the size of the item feature vector. The long-term preferences of users are once more modeled by a linear relationship given by $\boldsymbol{\beta}_j^\top \boldsymbol{x}_i$. Consequently, the joint short- and long-term preferences are modeled as follows

$$H^{(i)}_{\boldsymbol{\theta}_i, \boldsymbol{\beta}_j, e_i}(\boldsymbol{s}_t, \boldsymbol{x}_i, u_j) = \boldsymbol{\theta}_i^\top \boldsymbol{s}_t + \boldsymbol{\beta}_j^\top \boldsymbol{x}_i + e_i.$$

### 6.3.1 The Objective Function

Similar to the optimization problem in Section 6.2, the parameters of the short-term model are still independent from every other item as well as the user parameters $\{\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_m\}$ are independent among themselves, but are shared across the arms and that makes the objective function to be connected for all the arms and users. In order to model the items and users in a sequential fashion with varying $t$, we use $\bar{\boldsymbol{\theta}}_t \in \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\}$ and $\bar{\boldsymbol{\beta}}_t \in \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m\}$ to indicate the corresponding item and user at time $t$, respectively. Analogously, $\bar{\boldsymbol{x}}_t$ and $\bar{e}_t$ denote the features and bias term of the current item under consideration. Therefore, the general optimization problem with arbitrary loss function, $\mathcal{L}(\cdot, r_t)$ becomes

$$\inf_{\substack{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_n \\ \boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_m \\ \boldsymbol{e}}} \quad \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\bar{\boldsymbol{\theta}}_t^\top \boldsymbol{s}_t + \bar{\boldsymbol{\beta}}_t^\top \bar{\boldsymbol{x}}_t + \bar{e}_t, r_t\right) + \frac{\nu}{2}\sum_{i=1}^{n}\|\boldsymbol{\theta}_i\|^2 + \frac{\hat{\mu}}{2}\sum_{j=1}^{m}\|\boldsymbol{\beta}_j\|^2$$

where $\nu$ and $\hat{\mu}$ are the regularization parameters for the item and user weights, respectively. Let $C = \frac{1}{\nu T}$, $\mu = \frac{\bar{\mu}}{\nu}$, and $\boldsymbol{v} = (\ldots, v_t, \ldots)^\top$, we have

$$\begin{aligned}\inf_{\substack{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_n \\ \boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_m \\ \boldsymbol{e},\boldsymbol{v}}} \quad & C\sum_{t=1}^{T}\mathcal{L}\left(v_t, r_t\right) + \frac{1}{2}\sum_{i=1}^{n}\|\boldsymbol{\theta}_i\|^2 + \frac{\mu}{2}\sum_{j=1}^{m}\|\boldsymbol{\beta}_j\|^2 \\ s.t. \quad \forall t: \quad & \bar{\boldsymbol{\theta}}_t^\top \boldsymbol{s}_t + \bar{\boldsymbol{\beta}}_t^\top \bar{\boldsymbol{x}}_t + \bar{e}_t = v_t,\end{aligned}$$

which results in the Lagrange function

$$\begin{aligned}\sup_{\boldsymbol{\alpha}} \quad \inf_{\substack{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_n \\ \boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_m \\ \boldsymbol{e},\boldsymbol{v}}} \quad & C\sum_{t=1}^{T}\mathcal{L}\left(v_t, r_t\right) + \frac{1}{2}\sum_{i=1}^{n}\|\boldsymbol{\theta}_i\|^2 + \frac{\mu}{2}\sum_{j=1}^{m}\|\boldsymbol{\beta}_j\|^2 \\ & - \sum_{t=1}^{T}\alpha_t\left(\bar{\boldsymbol{\theta}}_t^\top \boldsymbol{s}_t + \bar{\boldsymbol{\beta}}_t^\top \bar{\boldsymbol{x}}_t + \bar{e}_t - v_t\right).\end{aligned}$$

Recall that $(\bar{\boldsymbol{\theta}}_t, \bar{\boldsymbol{x}}_t) \in \{(\boldsymbol{\theta}_1, \boldsymbol{x}_1), \ldots, (\boldsymbol{\theta}_n, \boldsymbol{x}_n)\}$, $\bar{\boldsymbol{\beta}}_t \in \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m\}$, and $\bar{e}_t \in \{e_1, \ldots, e_n\}$. Thus, the derivatives with respect to $\boldsymbol{\theta}_i$ generate

$$\boldsymbol{\theta}_i = \sum_{\substack{t \\ \bar{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_i}} \alpha_t \boldsymbol{s}_t = \sum_{t=1}^{T} \delta_t^i \alpha_t \boldsymbol{s}_t = (S \odot \boldsymbol{\delta}^i)^\top \boldsymbol{\alpha}.$$

In the above equation, $\boldsymbol{\delta}^i \in \mathbb{R}^T$ is a binary vector with respect to item $b_i$ of the size of $T$ which is one when $\bar{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_i$, and zero otherwise. In addition, $S \in \mathbb{R}^{T \times d_s}$ is the design matrix of input vectors, and $\odot$ stands for the element-wise product (each element in

the vector multiplies by a row in the matrix). Similarly, we compute the derivatives with respect to $\boldsymbol{\beta}_j$,

$$\boldsymbol{\beta}_j = \frac{1}{\mu} \sum_{\substack{t \\ \bar{\boldsymbol{\beta}}_t = \boldsymbol{\beta}_j}} \alpha_t \bar{\boldsymbol{x}}_t = \frac{1}{\mu} \sum_{t=1}^{T} \phi_t^j \alpha_t \bar{\boldsymbol{x}}_t = \frac{1}{\mu} (\bar{X} \odot \boldsymbol{\phi}^j)^\top \boldsymbol{\alpha},$$

where again $\boldsymbol{\phi}^j \in \mathbb{R}^T$ is the indicator vector for the corresponding user and $\bar{X}$ is the design matrix for the items features collected over time. For instance, if item $b_i$ has been chosen twice at $t_1$ and $t_2$ within $T$, then $t_1$ and $t_2$ entries of $\bar{X}$ is equal to $\boldsymbol{x}_i$, $\bar{\boldsymbol{x}}_{t_1} = \bar{\boldsymbol{x}}_{t_2} = \boldsymbol{x}_i$. Additionally, the derivatives with respect to $e_i$ yields

$$\forall i \quad \sum_{t:\bar{e}_t = e_i} \alpha_t = 0 \quad \implies \sum_t \alpha_t = 0 \quad \implies \mathbf{1}^\top \boldsymbol{\alpha} = 0.$$

Substituting the obtained conditions in the original problem leads to

$$\begin{aligned} \sup_{\boldsymbol{\alpha}, \mathbf{1}^\top \boldsymbol{\alpha} = 0} \quad \inf_{\boldsymbol{v}} \quad & C \sum_{t=1}^{T} [\mathcal{L}(v_t, r_t) + \frac{1}{C} \alpha_t v_t] \\ & - \frac{1}{2} \sum_i \boldsymbol{\alpha}^\top (S \odot \boldsymbol{\delta}^i)(S \odot \boldsymbol{\delta}^i)^\top \boldsymbol{\alpha} - \frac{1}{2\mu} \sum_j \boldsymbol{\alpha}^\top (\bar{X} \odot \boldsymbol{\phi}^j)(\bar{X} \odot \boldsymbol{\phi}^j)^\top \boldsymbol{\alpha}, \end{aligned}$$

which can be written as

$$\begin{aligned} \sup_{\boldsymbol{\alpha}, \mathbf{1}^\top \boldsymbol{\alpha} = 0} \quad & - C \sum_{t=1}^{T} \sup_{v_t} (-\frac{\alpha_t}{C} v_t - \mathcal{L}(v_t, r_t)) \\ & - \frac{1}{2} \sum_i \boldsymbol{\alpha}^\top (S \odot \boldsymbol{\delta}^i)(S \odot \boldsymbol{\delta}^i)^\top \boldsymbol{\alpha} - \frac{1}{2\mu} \sum_j \boldsymbol{\alpha}^\top (\bar{X} \odot \boldsymbol{\phi}^j)(\bar{X} \odot \boldsymbol{\phi}^j)^\top \boldsymbol{\alpha}. \end{aligned}$$

Finally, by converting the first term to the conjugate of the loss function using Fenchel-Legendre conjugates, we obtain

$$\begin{aligned} \sup_{\boldsymbol{\alpha}, \mathbf{1}^\top \boldsymbol{\alpha} = 0} \quad & - C \sum_{t=1}^{T} \mathcal{L}^*(-\frac{\alpha_t}{C}, r_t) - \frac{1}{2} \sum_i \boldsymbol{\alpha}^\top (S \odot \boldsymbol{\delta}^i)(S \odot \boldsymbol{\delta}^i)^\top \boldsymbol{\alpha} \\ & - \frac{1}{2\mu} \sum_j \boldsymbol{\alpha}^\top (\bar{X} \odot \boldsymbol{\phi}^j)(\bar{X} \odot \boldsymbol{\phi}^j)^\top \boldsymbol{\alpha}. \end{aligned} \tag{6.4}$$

Hence, Equation (6.4) constitutes a generalized optimization problem for contextual bandits with arbitrary loss function. It contains the short-term model in Equation (6.1) as a special case when no personal long-term interests need to be captured.

### 6.3.2 Optimization

Equation (6.4) can be optimized with various optimization methods depending on the loss function as well as standard techniques such as gradient-based approaches. For real-time applications and online scenarios, model updates can be performed using (mini-) batches at regular intervals as well, for efficiency. The objective function needs to be maximized with respect to the dual parameters $\boldsymbol{\alpha}$ and is given by

$$\sup_{\boldsymbol{\alpha}, \mathbf{1}^\top \boldsymbol{\alpha}=0} \quad -C\mathbb{I}^\top \mathcal{L}^*(-\frac{\boldsymbol{\alpha}}{C}, \boldsymbol{r}) - \frac{1}{2}\sum_i \boldsymbol{\alpha}^\top (S \odot \boldsymbol{\delta}^i)(S \odot \boldsymbol{\delta}^i)^\top \boldsymbol{\alpha} - \frac{1}{2\mu}\sum_j \boldsymbol{\alpha}^\top (\bar{X} \odot \boldsymbol{\phi}^j)(\bar{X} \odot \boldsymbol{\phi}^j)^\top \boldsymbol{\alpha}.$$

The gradient with regard to $\boldsymbol{\alpha}$ is obtained by computing the derivatives

$$-C\frac{\partial \mathcal{L}^*(-\frac{\boldsymbol{\alpha}}{C}, \boldsymbol{r})}{\partial \boldsymbol{\alpha}} - \left[\sum_i (S \odot \boldsymbol{\delta}^i)(S \odot \boldsymbol{\delta}^i)^\top - \frac{1}{\mu}\sum_j (\bar{X} \odot \boldsymbol{\phi}^j)(\bar{X} \odot \boldsymbol{\phi}^j)^\top\right]\boldsymbol{\alpha} - \epsilon\mathbb{I} = 0,$$

where $\epsilon$ is another Lagrange multiplier and $\mathbb{I}$ is the identity matrix. However, the actual form of the gradient depends on the loss function and its conjugates $\mathcal{L}^*$ and further derivations are omitted accordingly. Note that instantiations often give rise to more sophisticated and efficient optimization techniques than the general form in Equation (6.4) allows, see also Section 6.2.2. Nevertheless, the sketched gradient-based approach will always work in case a general optimizer is needed, e.g., in cases where several loss functions should be tried out. Once the optimal parameters $\boldsymbol{\alpha}^*$ have been found, they can be used to compute the primal parameters

$$\boldsymbol{\theta}_i = (S \odot \boldsymbol{\delta}^i)^\top \boldsymbol{\alpha}^*, \quad \boldsymbol{\beta}_j = \frac{1}{\mu}(\bar{X} \odot \boldsymbol{\phi}^j)^\top \boldsymbol{\alpha}^*.$$

Note that $e_i$ is learned as an augmented variable into $\boldsymbol{\theta}_i$, but is not regularized. Additionally, kernels $K_S$ and $K_{\bar{X}}$ could be deployed in the dual representation to allow for non-linear transformations and convolutions in the feature space.

Once the required parameters are found, the payoff estimates are used together with the respective confidence interval $U$ of the arms, in every time step, to choose the arm with the maximum upper confidence value according to

$$b^*_{t;u_j} = \arg\max_{b_i \in \mathbb{B}} \quad \boldsymbol{\theta}_i^\top \boldsymbol{s}_t + \boldsymbol{\beta}_j^\top \boldsymbol{x}_i + e_i + U_{i,t},$$

that $b^*_{t;u_j}$ is the optimal arm selected from the contextual bandit model given user $u_j$ at time $t$. In the remainder of this section, we describe the learning algorithm for the special cases of loss functions.

**Learning with Squared Loss.** In this part, we present the optimization algorithm for a special case of unified contextual bandit framework with squared loss. As it is mentioned in Section 6.2.2, the conjugate of squared loss is given by

$$\mathcal{L}^*(-\frac{\alpha_t}{C}, r_t) = \frac{1}{2C^2}\alpha_t^2 - \frac{1}{C}\alpha_t r_t,$$

which by substituting in Equation (6.4) leads to the following objective

$$\begin{aligned}\max_{\boldsymbol{\alpha}, \mathbf{1}^\top\boldsymbol{\alpha}=0} \quad & -\frac{1}{2C}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} + \boldsymbol{r}^\top\boldsymbol{\alpha} - \frac{1}{2}\sum_i \boldsymbol{\alpha}^\top(S\odot\boldsymbol{\delta}^i)(S\odot\boldsymbol{\delta}^i)^\top\boldsymbol{\alpha} \\ & -\frac{1}{2\mu}\sum_j \boldsymbol{\alpha}^\top(\bar{X}\odot\boldsymbol{\phi}^j)(\bar{X}\odot\boldsymbol{\phi}^j)^\top\boldsymbol{\alpha}.\end{aligned}$$

The summation $\sum_i (S\odot\boldsymbol{\delta}^i)(S\odot\boldsymbol{\delta}^i)^\top$ is equivalent to $(\sum_i \boldsymbol{\delta}^i\otimes\boldsymbol{\delta}^{i\top})\odot SS^\top$, where $\otimes$ stands for the vector outer product. Considering the same equivalency for the last term as well, we rewrite the equation as follows

$$\begin{aligned}\max_{\boldsymbol{\alpha}, \mathbf{1}^\top\boldsymbol{\alpha}=0} \quad & -\frac{1}{2C}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} + \boldsymbol{r}^\top\boldsymbol{\alpha} \\ & -\frac{1}{2}\boldsymbol{\alpha}^\top\left[(\sum_i \boldsymbol{\delta}^i\otimes\boldsymbol{\delta}^{i\top})\odot SS^\top + \frac{1}{\mu}(\sum_j \boldsymbol{\phi}^j\otimes\boldsymbol{\phi}^{j\top})\odot\bar{X}\bar{X}^\top\right]\boldsymbol{\alpha}.\end{aligned}$$

By using *min* instead of the *max* operation, setting $\boldsymbol{M} = \frac{1}{C}\mathbb{I} + (\sum_i \boldsymbol{\delta}^i\otimes\boldsymbol{\delta}^{i\top})\odot SS^\top + \frac{1}{\mu}(\sum_j \boldsymbol{\phi}_j\otimes\boldsymbol{\phi}^{j\top})\odot\bar{X}\bar{X}^\top$, and $\boldsymbol{q} = -\boldsymbol{r}$, the problem becomes a standard quadratic optimization with a constraint,

$$\min \quad \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{M}\boldsymbol{\alpha} + \boldsymbol{q}^\top\boldsymbol{\alpha} \quad \text{s.t.} \quad \boldsymbol{\alpha}, \mathbf{1}^\top\boldsymbol{\alpha} = 0. \tag{6.5}$$

Algorithm 1 summarizes the procedure of optimizing for the squared loss. In each iteration, the algorithm computes the UCB value of all arms for the observed user, and chooses the arm with the highest value in line 20. The required parameters for the quadratic optimization are updated within line 21 to 28 which leads to optimizing $\boldsymbol{\alpha}$. The obtained vector is used to update the model parameters. Note that the objective function is optimized for all the parameters, therefore, it affects them all and not just one user and one item. In this algorithm, we further assume that the covariance matrices of item and user parameters are independent from each other. Hence, we discard the correlation between them and obtain the variance by summing them as $\boldsymbol{x}_i^\top(A_j^u)^{-1}\boldsymbol{x}_i + \boldsymbol{s}_t^\top(A_i^b)^{-1}\boldsymbol{s}_t$ (line 17) in order to compute the confidence bound.

**Algorithm 1** Short- and long-term regression UCB

Inputs: $o$, $C$, and $\mu$
Initialize $S \leftarrow \emptyset_{0\times d_s}$, $\quad \bar{X} \leftarrow \emptyset_{0\times d_x}$, $\quad \boldsymbol{r} \leftarrow \emptyset$
**for** $t = 1, 2, ..., T$ **do**
    Observe the user $u_j$ which arrives at time $t$ and context $\boldsymbol{s}_t \in \mathbb{R}^{d_s\times 1}$
    **if** $u_j$ is new **then**
        $A_j^u \leftarrow \mathbb{I}_{d_x} \cdot \mu$
        $\boldsymbol{\beta}_j \leftarrow \mathbf{0}_{d_x\times 1}$
        $\boldsymbol{\phi}^j \leftarrow \mathbf{0}_{t\times 1}$
    **end if**
    **for all** $b_i \in \mathbb{B}$ **do**
        Observe the features of arm $\boldsymbol{x}_i \in \mathbb{R}^{d_x\times 1}$
        **if** $b_i$ is new **then**
            $A_i^b \leftarrow \mathbb{I}_{d_s}$
            $\boldsymbol{\theta}_i \leftarrow \mathbf{0}_{d_s\times 1}$
            $\boldsymbol{\delta}^i \leftarrow \mathbf{0}_{t\times 1}$
        **end if**
        $\text{var}_t^i = \boldsymbol{x}_i^\top (A_j^u)^{-1} \boldsymbol{x}_i + \boldsymbol{s}_t^\top (A_i^b)^{-1} \boldsymbol{s}_t$
        $p_t^i = \boldsymbol{\theta}_i^\top \boldsymbol{s}_t + \boldsymbol{\beta}_j^\top \boldsymbol{x}_i + o\sqrt{\text{var}_t^i}$
    **end for**
    Choose arm $b^* = \arg\max_i p_t^i$ with ties broken randomly, and observe payoff $r_t$
    $A_i^b = A_i^b + \boldsymbol{s}_t \boldsymbol{s}_t^\top$
    $A_j^u = A_j^u + \boldsymbol{x}_{b^*} \boldsymbol{x}_{b^*}^\top$
    $S \leftarrow [S; \boldsymbol{s}_t^\top]$ ▷ (Append vertically)
    $\bar{X} \leftarrow [\bar{X}; \boldsymbol{x}_{b^*}^\top]$ ▷ (Append vertically)
    $\mathbf{r} \leftarrow [\mathbf{r}, r_t]$
    **for all** $b_i \in \mathbb{B}$ and $u_j \in \mathbb{U}$ **do**
        Update $\boldsymbol{\delta}^i$ and $\boldsymbol{\phi}^j$
    **end for**
    Obtain $\boldsymbol{\alpha}$ by optimizing Equation (6.5)
    **for all** $b_i \in \mathbb{B}$ and $u_j \in \mathbb{U}$ **do**
        $\boldsymbol{\theta}_i = (S \odot \boldsymbol{\delta}^i)^\top \boldsymbol{\alpha}$
        $\boldsymbol{\beta}_j = (\bar{X} \odot \boldsymbol{\phi}^j)^\top \boldsymbol{\alpha}$
    **end for**
**end for**

**Learning with Logistic Loss** Another special case of our unified framework is to apply the logistic loss for the optimization process. As we introduced in Section 6.2.2, the conjugate of the logistic loss is given by

$$\mathcal{L}^*(-\frac{\alpha_t}{r_t}, r_t) = (1 - \frac{\alpha_t}{Cr_t})\log(1 - \frac{\alpha_t}{Cr_t}) + \frac{\alpha_t}{Cr_t}\log(\frac{\alpha_t}{Cr_t}).$$

Employing the above conjugate into the Equation (6.4) leads to

$$\begin{aligned}\min_{\boldsymbol{\alpha}, \mathbf{1}^\top\boldsymbol{\alpha}=0} \quad & C\sum_{t=1}^{T}\left[(1 - \frac{\alpha_t}{Cr_t})\log(1 - \frac{\alpha_t}{Cr_t}) + \frac{\alpha_t}{Cr_t}\log(\frac{\alpha_t}{Cr_t})\right] \\ & + \frac{1}{2}\boldsymbol{\alpha}^\top[(\sum_i \boldsymbol{\delta}^i \otimes \boldsymbol{\delta}^{i\top}) \odot SS^\top + \frac{1}{\mu}(\sum_j \boldsymbol{\phi}^j \otimes \boldsymbol{\phi}^{j\top}) \odot \bar{X}\bar{X}^\top]\boldsymbol{\alpha}.\end{aligned}$$

The procedure for learning the model is similar to Algorithm 1, nevertheless, the objective function in line 29 needs to be optimized differently, and also computing $\text{var}^i$ in line 17 differs. For the latter case, the covariance matrix is computed for both set of parameters, $\Sigma^b = S^\top W^b S$ and $\Sigma^u = \bar{X}^\top W^u \bar{X}$, respectively. Therefore, $\boldsymbol{x}_i^\top(\Sigma_j^u)^{-1}\boldsymbol{x}_i + \boldsymbol{s}_t^\top(\Sigma_i^b)^{-1}\boldsymbol{s}_t$ is used as the variance in computing the lower and upper confidence bounds (see Section 6.2.2). Note that gradient-based methods are still applicable in these optimization problems.

## 6.4 Extensions

In this section, we discuss some potential alternatives of our proposed approach which are suitable and sometimes more effective for certain circumstances.

### 6.4.1 Toward more Efficient Models

The proposed unified model in Section 6.3 combines the contextual item model with the user preferences in one framework. The model is therefore more than the vanilla bandit-based approaches that only model one of those. However, the model contains many parameters and the optimization part becomes more and more complex as the system size (both the number of items and users) grows. We further suggest to simplify the approach in two different directions: relaxing the item model or discarding the personalized term. Hence, we introduce four simplified cases of the combined approach as follows.

1. **Short-Term:** To model the payoff function only for the items, no personalization (aka. LinUCB (Li et al., 2010)): $\mathbb{E}[r_t|b_i] = \boldsymbol{\theta}_i^\top \boldsymbol{s}_t + e_i$.

2. **Short-Term+Average:** Considering an average term for all the items into the model, still no personalization (resembling HybridUCB (Li et al., 2010)): $\mathbb{E}[r_t|b_i] = \boldsymbol{\theta}_i^\top \boldsymbol{s}_t + \boldsymbol{\beta}_0^\top \boldsymbol{x}_i + e_i$.

3. **Long-Term:** Only a personalized model: $\mathbb{E}[r|b_i, u_j] = \boldsymbol{\beta}_j^\top \boldsymbol{x}_i + e_i$.

4. **Long-Term+Average:** Incorporating the average over users into the personalized model: $\mathbb{E}[r|b_i, u_j] = \boldsymbol{\beta}_j^\top \boldsymbol{x}_i + \boldsymbol{\beta}_0^\top \boldsymbol{x}_i + e_i$.

Note that $\boldsymbol{\beta}_0$ models the average interest of all the users in items features and the latter two cases do not depend on time. Moreover, the average part in the second and fourth cases depicts the item popularity and common trends in the recommendation systems. These cases are easily derivable from the equations in Section 6.3. We further examine the benefits of average models in Section 6.5.

### 6.4.2 Preference-based Bandits

One natural extension of our approach is to characterize the contextual bandit model for the preference-based setting. There are many systems with no available quantitative feedback, whereas the feedback is provided in terms of pairwise comparison between items. In such cases, the preferences are used in the learning process and the rankings are predicted directly from the model. In this section, we discuss how to phrase our bandit framework in a preference-based context. Nevertheless, in the next chapter, we introduce an approach for recommendation scenarios which is only specialized for preference-based and qualitative data.

We consider the contextual bandit problem in a way that the context is specified by the features of items to recommend. The model is thus defined by a single bandit which learns the preferences between items for all the users. Assume that $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ are the features of items $b_i$ and $b_k$, respectively, and we assign $\boldsymbol{x}_{i \succ k} := \boldsymbol{x}_i - \boldsymbol{x}_k$ to show the preference of item $b_i$ over $b_k$. The payoff is therefore determined as a linear model of the preference,

$$\mathbb{E}[r_{i \succ k}|u_j] = \boldsymbol{\beta}_j^\top \boldsymbol{x}_{i \succ k} + \boldsymbol{\beta}_0^\top \boldsymbol{x}_{i \succ k},$$

where $\boldsymbol{\beta}_0$ is again the weight vector for the average model, while $\boldsymbol{\beta}_j$ is the individual parameter for user $u_j$ and acts as a personal offset. The above equation is theoretically analogous to the fourth case in the previous part.

## 6.5 Empirical Study

The purpose of this section is to evaluate the performance of our combined contextual bandit approach compared to either short-term or long-term models. We further investigate the effectiveness of our approach for cold start problems.

In our experiments, we use the squared loss function as in Algorithm 1 and benefit from the CVXOPT[1] tool for the optimization process. The quality of recommendation is measured via normalized average rank, that is, for every test instance, a ranking of all items is inferred by the model. The position of the actually clicked item in the ranking is then normalized (divided by the total number of items) and averaged over all test samples. Error bars in the figures indicate standard error. The empirical study demonstrates that adding a long-term model describing the user preferences improves short-term recommendations. Additionally, we show that the simplified average models are beneficial in cold start scenarios.

The experiments are conducted on a real-world dataset from Zalando, which we introduced in Chapter 4, with anonymized click history of various users. The data is collected over time and bucketized into consecutive sessions. Each user interacts with the system in different sessions, and each session contains a sequence of products views. Products are described with categorical attributes, which among them we use **category**, **brand**, **color**, **gender**, **price level**, and **action** $\in$ {view, sale, cart, wish list, remove from cart, remove from wish list}. We apply a one-hot encoding of the categorical features and enrich the representation by three additional features: the"*item popularity*" for each item, and "*sale to view*" as well as "*view to action*" ratios per user. The augmented dataset encompasses users with at least five sessions, where all sessions with more than one click are considered a valid session.

### 6.5.1 Overall Performance

In the first experiment, we examine how the combined approach performs on datasets of different sizes compared to the short- and long-term models and a matrix factorization baseline (**MF**) (Koren et al., 2009). The parameters of the latter are optimized by model selection which results in 200 factors, and a regularization constant of 0.1. We thus generate several subsets of data by randomly sampling different numbers of users to obtain sets with about 1-15k user transactions. We split each set into training and test sets by reserving 80% of sessions for the former and assigning the rest to the test set. Note that there is no new user or new item in the test data.

[1] https://cvxopt.org

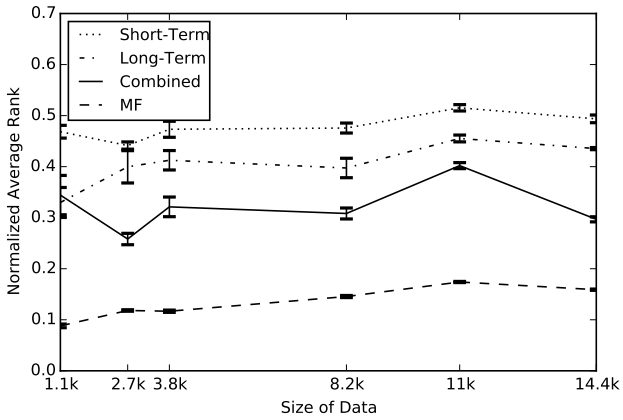



Figure 6.1: Normalized average rank for different data sizes.

The context in our setup is the feature vector of the previously viewed product which means $d_x = d_s$. Therefore, the first click of each session is discarded and kept as the context for the next click. The reward value for each action is either 1 for the correct arm or $-1$ otherwise. We consider a constant $o = 2.36$, and set the regularization parameters to $\nu = \mu = 1$. Figure 6.1 depicts the results for our approach as well as the short- and long-term models averaged over several runs.

The figure shows that the combined approach outperforms both the short- and long-term methods in terms of average rank (lower is better). The short-term approach performs worse than the other two, since the data is obtained by sampling users, and there are many more items than users. However, the size of data does not change the behavior of the tested methods significantly apart from the combined model that improves performance with increasing data sizes; an indicator for the necessity of experiments at even larger scales. The matrix factorization baseline performs best when all users and items are known.

### 6.5.2 Cold Start Scenarios

One of the main contributions of our proposed approach in the contextual setting is the ability to generalize over different items for individual users. This advantage suits well to cold start situations, where the content is highly dynamic and the item and user sets change frequently. First, we demonstrate the behavior of the combined approach when new users and items appear in the test set.

We create a subset of data from all the sessions of 100 randomly selected users. The data contains 1,295 sessions that gives an average of 13 sessions per user, and about 8,000 products. We split the data into training and test sets with different ratios

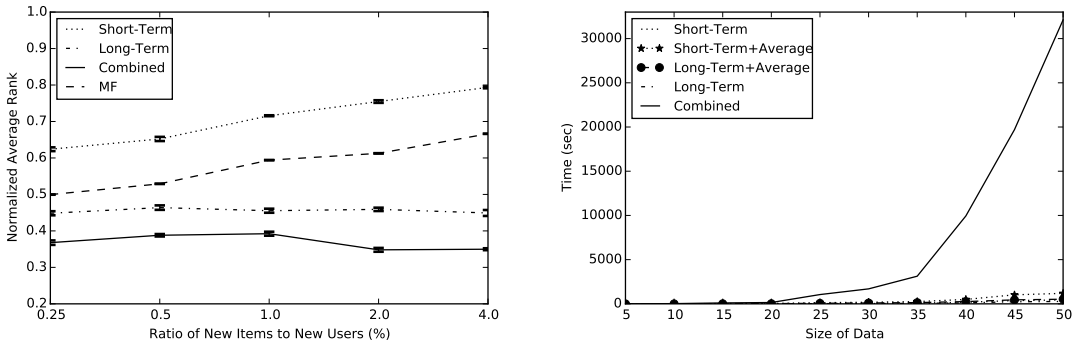
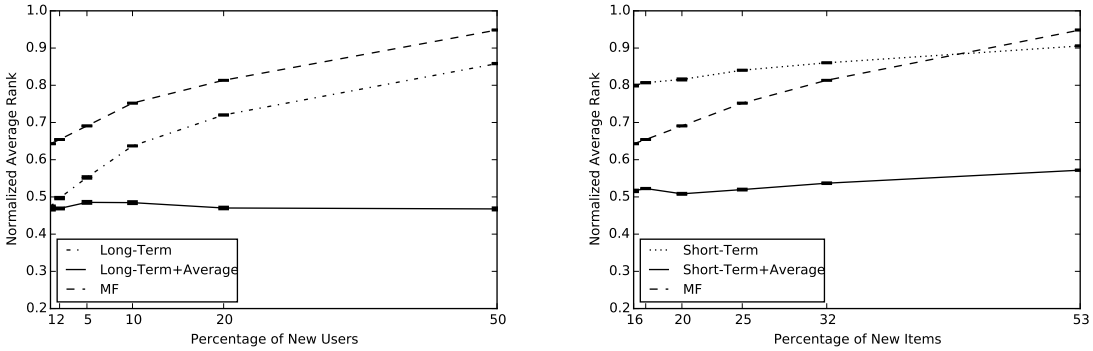



Figure 6.2: Left: Normalized average rank for different ratios of new items to new users. Right: Execution time for different data sizes.

for the percentage of new items and new users in the test data. To this purpose, we leave $f_u\%$ of the users and $f_b\%$ of the products to appear only in the test set such that it realizes a ratio of $\frac{f_b}{f_u}$ for the new items over new users. We train our combined approach as well as both short- and long-term models, where the context and reward setup is as in the previous section. Figure 6.2 (left) shows the behavior of different approaches; again results are averaged over ten runs.

The first impression from Figure 6.2 (left) suggests that although the combined method still outperforms the baselines, its performance declines a bit near the ratio of 1 when many new users and new items are available. Unsurprisingly, the short-term model performs better for scenarios with only a few new items. This holds vice versa also for the long-term model that performs better for scenarios with almost constant sets of users. The performance of matrix factorization degrades significantly in the new setting which confirms the robustness of our combined method in the real scenarios. Nonetheless, the robustness comes at the cost of run-time: the combined approach is computationally expensive because of the involved convex optimization. The run-time analysis in Figure 6.2 (right) displays the exponential growth in execution time of our combined approach in comparison to the other approaches discussed in Section 6.4.1.

In the next setting, we focus on the evaluation of adding average models to the short- and long-term approaches. We conduct the experiments on a medium-sized dataset to evaluate their performance. The dataset in this experiment contains all transactions of 500 random users. We split the data by modifying the percentage of new users and new items in the test set and analyze two cases. Figure 6.3 shows how adding the average term significantly improves the performance of both long-term and short-term models, respectively.




(a) Performance in terms of new users (b) Performance in terms of new items

Figure 6.3: Normalized average rank for the data with new users and items.

As in the previous experiment, Figure 6.3a depicts that the performance of the long-term model decreases for increasing numbers of new users. By contrast, extending the long-term model by an average model remedies this effect and the extended model is able to cope with the challenging scenario and even improves performance. Similar behavior is displayed in Figure 6.3b, where short-term model, augmented by an average model, eliminates the shortcomings of the short-term model in dealing with new items. By contrast, the collaborative filtering method fails to catch up and performs poorly in both scenarios. As a result, maintaining additional average models is an effective and efficient means in cold start situations. The experiments however also show that there is no one model that rules them all; instead, the model of choice depends clearly on the intrinsic dynamics of the applications.

## 6.6 Conclusion

In this chapter, we expanded our contributions to sequential recommendation to approaches that handle personalization, as well. We presented a unified model for short- and long-term recommendations in a contextual multi-armed bandit framework. The model incorporated both the information from the actual context as well as the long-term preferences of the users into a single contextual bandit. We transformed the optimization problem of our bandits theoretically into the dual space by considering a linear payoff model for the arms.

Addressing the problem in dual space led to a generalized optimization problem in which any arbitrary loss function can be used to reshape the payoff function according to the application at hand. As a result, applying contextual bandits for short- and long-term recommendations is considerably simplified. The experiments

show that adding an average model to short- and long-term models leads to robust methods which clearly outperform their vanilla peers in terms of normalized average rank. Moreover, applying some simplifications to the approach helped it to fit more efficiently into special scenarios and proved beneficial in cold start problems. Nevertheless, we once again conclude that choosing a suitable model depends highly on the characteristics of the underlying application.

# Chapter 7

# Preference-based Personalized Transaction Kernels

In the previous chapter, we proposed a sequential method based on contextual bandits to model both the personalized preferences of users and their short-term interests in the same framework. However, as discussed in Section 6.5.2, having an exclusive model per user comes at the cost of run-time, making such an approach unaffordable in many applications that deal with numerous users.

In this chapter, we present a more efficient approach with which to incorporate personalization into a more interpretable contextual setting. We introduce the problem for preference-based scenarios and motivate the undertaken approach in the first section. We then review the related work in Section 7.2. Section 7.3 describes the problem setting and our main contribution to the contextual and personalized preference model. An informed sequential search technique using MCTS is presented in Section 7.4. Section 7.5 reports on the empirical results, and Section 7.6 concludes.

## 7.1 Motivation

Understanding user behavior is essential in many recommendation tasks involving implicit feedback. Several approaches have been aimed at capturing the characteristic traits of users by analyzing data, ranging from atomic user actions, such as clicks and purchases, to their entire navigation patterns.

Nevertheless, for the vast majority of users, the available data is very limited. User clicks do not per se express an interest in an item; purchases are generally rare events, by definition; and a reliable analysis of navigation patterns requires regular (and possibly frequent) visits on the part of the same user. Maintaining an individual model for every user contains another caveat: in addition to retrieval and maintenance

costs, again only heavy hitters will really benefit from such an approach. Therefore, there is a great need for techniques that leverage all available data so that *every* user benefits, irrespective of their amount of data.

In this chapter, we explore pairwise preference data to study the behavior of users in online recommendation problems. We propose using qualitative feedback in the form of pairwise preferences as the lever since those preferences disclose the true interests of a user and have often served as a reliable source of information (Fürnkranz & Hüllermeier, 2010; Gemmis et al., 2010). However, preferences depend highly on the context of the user. We employ results from tensor theory to present a contextual transaction kernel which maps multiple data sources into a joint feature space (e.g., the user's click history, context, demographics, and so forth). The representation is extended to all users via hash functions which augment the data from other users into this space in such a way that user slices can be efficiently stored and retrieved. The kernel can be trivially extended to pairwise preference data so that a preference learning algorithm, such as $\text{SVM}^{\text{rank}}$ (Joachims, 2002), can be used to obtain personalized preference models.

Furthermore, we introduce an online search technique to recommend the most relevant items to the actual user in a given context. Therefore, we benefit from personalized preference models as a utility functions in order to conduct such informed sampling in product recommendation tasks. The utility function must be maximized in order to identify optimal item(s), i.e., those items which are most likely to be clicked on by the user in a given context according to the learned model. A naïve approach for computing the best item to recommend is an exhaustive search over all possible items and returning the item with the maximum value. An anytime version of this algorithm returns the item with the highest value among those that have been seen so far. To improve upon this setup, we propose a variant of the Monte Carlo Tree Search (MCTS) algorithm, which allows the system to focus quickly on items with favorable features. Our results demonstrate that the MCTS variant returns better recommendations for cases in which the number of sampled products is limited.

## 7.2 Related Work

Recall that recommendation systems leverage feedback either in form of explicit ratings or implicit behavior to retrieve items of interest for the users. An alternative viewpoint constitutes scenarios that are based on feedback in form of qualitative user preferences for predicting user interests (Gemmis et al., 2010). Preference learning

describes a family of learning problems where the target information is not necessarily given, but preferences between options are known; the task is thus to predict an ordering between these options (Fürnkranz & Hüllermeier, 2010). One can distinguish between object ranking problems (Kamishima et al., 2010), where the task is to order a set of featurized objects, and label ranking problems (Vembu & Gärtner, 2010), where the task is to order a set of labels in a featurized context. Both problems can be approached in different ways, such as by directly modeling the binary preference relation, or by inducing an underlying utility function.

We formalize our problem as an object ranking task, that we address by learning an underlying utility function. More precisely, we learn a personalized preference model using Support Vector Machines (SVMs), in a quite similar fashion to Joachims (2002), who effectively utilizes an SVM to learn a ranking function for click-through data, and Chapelle & Keerthi (2010), who present an efficient method to speed up the former algorithm. Furthermore, the use of SVMs facilitates to deal with non-linearity by using the kernel trick. Kernel methods have been successfully employed for top-K recommendation tasks using, for instance, Gaussian processes (Vanchinathan et al., 2014) or contextual bandits (Tavakol & Brefeld, 2017) (see also Chapter 6).

In this chapter, we benefit from tensor kernels to express the conjugation of different feature representations (or contexts) by tensor products. Tensor products (Dullemond & Peeters, 2010), both as an explicit feature mapping and kernel function, have been employed for feature selection in classification tasks (Cao et al., 2014, 2015; Smalter et al., 2009). Oyama & Manning (2004) propose a tensor kernel to conjugate features of example pairs for learning pairwise classifiers. Tensors are additionally used for relation extraction in unstructured natural language parsing (Zelenko et al., 2003). The idea of joint feature maps using tensor product is further utilized in recommendation, where Basilico & Hofmann (2004) present a collaborative-based kernel method over user-item pairs for rating prediction. Instead, we exert hash functions for learning user-specific models, and empirically show that our approach significantly outperforms their algorithm yet with a much faster computation.

Hashing functions are introduced by Shi et al. (2009) for sparse projections in multi-class classification tasks. The idea originates in an approach called *Count Sketch* (Charikar et al., 2002), a method to estimate item frequencies in data streams. Hashing is considered an effective way to reduce the dimensionality of the problem by reducing the number of bits in the hash function. Nonetheless, the induced mapping is no longer one-to-one and at some point the hashed representation becomes too

noisy to be useful. Weinberger et al. (2009) propose a hashing trick for large-scale multi-task learning, where all tasks are mapped into a joint hash space.

Together with tensor products, Pham & Pagh (2013) apply hashing as a random feature mapping to approximate polynomial kernels in large-scale problems with bounded error. They first represent the tensor product feature space as an equivalent representation of polynomial kernels, and propose an efficient way to randomly project the data into a lower dimension feature space without explicitly computing the tensor products. Subsequently, Wang et al. (2015) exploit randomized tensors to efficiently perform implicit tensor decomposition for latent topic modeling. We utilize hash kernels for personalization and tensor kernels for intermixing various feature sources to model user preferences.

Our learned preference model is further used in an MCTS framework to sample (near-) optimal products for online recommendation. MCTS is an anytime tree search algorithm for sequential decision making (Browne et al., 2012; Kocsis & Szepesvári, 2006) that became very popular due to the great success in the game playing domains such as Go (Silver et al., 2016). MCTS has been also employed in recommendation problems. Liebman et al. (2017) design an approach based on MCTS for playlist recommendation and develop alternative backup strategies to increase convergence speed. Moreover, Gaudel & Sebag (2010) deploy MCTS for efficient feature selection in recommendation tasks. Nevertheless, we aim at efficiently focusing on objects with desirable features but not a selection of features *per se*.

## 7.3 Transaction Kernels

We study transaction scenarios in which users, represented by their user identifier $u \in \mathbb{U}$, click on a product $b \in \mathbb{B}$. The context of the click encompasses all the available sources of information (e.g., the sequence of previous clicks, the day and time, etc.) which is captured by $\boldsymbol{s}_t$. In this chapter, we aim to understand why user $u$ in context $\boldsymbol{s}_t$ clicks on item $b$ and not on some other presented item $b'$ and to turn this understanding into a recommendation system that shows interesting new items to the users depending on their context.

Before we exploit user preferences, we assume that there is often more information available than the triplet of user identifier, item, and context. Some scenarios may provide additional data sources such as user profile data, shipping and billing addresses, additional information on items, user friendship graphs, or further demographics. We thus assume the existence of $h$ different data sources $\mathcal{X} = \{\mathbb{X}^1, \ldots, \mathbb{X}^h\}$,

where every source adds some pieces of information to the problem. We first consider the generalized problem of merging the $h$ sources into a personalized joint representation before we incorporate preferences and learn a utility function that can be used together with Monte Carlo tree search-based approaches.

**Tensor Kernels.** In order to capture the interlace properties of different data sources, $\{\mathbb{X}^1, \ldots, \mathbb{X}^h\}$, we define the mapping $\psi^t$ as the tensor product of their respective vector spaces. The tensor product of two vector spaces $Y$ and $Y'$ is again a vector space $Y \otimes Y'$ (Dullemond & Peeters, 2010). Let $\{y_1, y_2, \ldots, y_{d_1}\}$ and $\{y'_1, y'_2, \ldots, y'_{d_2}\}$ be the basis systems of $Y$ and $Y'$, respectively, their tensor product space is spanned by a basis that contains all pairs $(y_i, y'_j)$. For instance, if $\boldsymbol{y} = \{y_1, y_2, y_3\}$ and $\boldsymbol{y'} = \{y'_1, y'_2\}$, the tensor product $\boldsymbol{y} \otimes \boldsymbol{y'}$ is given by $\{y_1y'_1, y_1y'_2, y_2y'_1, y_2y'_2, y_3y'_1, y_3y'_2\}$. Applying this to our setting results in a mapping $\psi^t$ on $\mathbf{x} \in \mathcal{X}$ as follows

$$\psi^t(\mathbf{x}) = \psi^t(\mathbf{x}^1, \ldots, \mathbf{x}^h) = \mathbf{x}^1 \otimes \ldots \otimes \mathbf{x}^h. \tag{7.1}$$

Let $n_1, \ldots, n_h$ be the dimensions of the feature spaces. For all $\mathbf{x}, \mathbf{z} \in \mathcal{X}$ we derive

$$\begin{aligned}
\langle \psi^t(\mathbf{x}), \psi^t(\mathbf{z}) \rangle &= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \ldots \sum_{l=1}^{n_h} (\mathrm{x}_i^1 \mathrm{x}_j^2 \ldots \mathrm{x}_l^h)(\mathrm{z}_i^1 \mathrm{z}_j^2 \ldots \mathrm{z}_l^h) \\
&= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \ldots \sum_{l=1}^{n_h} \mathrm{x}_i^1 \mathrm{z}_i^1 \mathrm{x}_j^2 \mathrm{z}_j^2 \ldots \mathrm{x}_l^h \mathrm{z}_l^h \\
&= (\sum_{i=1}^{n_1} \mathrm{x}_i^1 \mathrm{z}_i^1)(\sum_{j=1}^{n_2} \mathrm{x}_j^2 \mathrm{z}_j^2) \ldots (\sum_{l=1}^{n_h} \mathrm{x}_l^h \mathrm{z}_l^h) \\
&= \langle \mathbf{x}^1, \mathbf{z}^1 \rangle \langle \mathbf{x}^2, \mathbf{z}^2 \rangle \ldots \langle \mathbf{x}^h, \mathbf{z}^h \rangle.
\end{aligned}$$

Thus, the *tensor kernel* $k^t$ is obtained by multiplying the corresponding inner products between the spaces

$$k^t(\mathbf{x}, \mathbf{z}) = \prod_{d=1}^{h} \langle \mathbf{x}^d, \mathbf{z}^d \rangle.$$

As a result, the tensor product features are equivalent to the product of kernels for all domains in case of a linear kernel (Smalter et al., 2009). Note that the proposed kernel possesses an explicit and interpretable representation given in Equation (7.1) that may be useful in large-scale tasks with small dimensionalities.

**Personalized Kernels.** As we mentioned earlier in this section, the user identifiers are also considered as another source of information. However, tensor products of such terms with other views act as normal inner products and ergo do not affect the learning process in a meaningful way. We thus propose a hashed feature mapping $\psi^p$ on top of the tensor product $\psi^t$ to remedy this limitation. Given two hash functions $g : \mathbb{N} \to \{1, \ldots, d_p\}$ and $\phi : \mathbb{N} \to \{-1, +1\}$ the hashed feature map $\psi^p$ is defined as

$$\psi^p_l(\mathbf{x}) = \sum_{j:g(j)=l} \phi(j)\mathrm{x}_j \quad \text{(Weinberger et al., 2009)},$$

where $d_p$ is the hash size and the binary hash function $\phi$ is used to remove the bias inherent in the hash kernel. Consequently, the obtained hashing function gives rise to the *personalized kernel* $k^p$

$$k^p(\mathbf{x}, \mathbf{z}) := \langle \psi^p(\psi^t(\mathbf{x})), \psi^p(\psi^t(\mathbf{z})) \rangle.$$

The presence of a user identifier automatically leads to a user-specific representation without the need to maintain an individual model for every user. Hence, the personalized kernel individually hashes all data sources into user slices and allows to control the dimensionality of the resulting feature space via the number of bits in the hash function. Moreover, the length of the hashed vector is preserved with high probability in the new space (Weinberger et al., 2009).

**Collective Kernels.** One of the major problems of personalized systems is to cope with cold start situations. Usually, many users in the system have no or too few transactions, which leads to inaccurate personalized models. Borrowing ideas from Chapter 4 and Weinberger et al. (2009), we propose an additional collective kernel that stores all user data in a single slice to account for users and contexts with limited data. Therefore, the *collective kernel* function $k^c$ simply discards the user identifiers from the tensor products and is thus given by

$$k^c(\mathbf{x}, \mathbf{z}) := \langle \psi^p(\psi^t(\mathbf{x} \setminus u)), \psi^p(\psi^t(\mathbf{z} \setminus u)) \rangle.$$

**Combining Personalized and Collective Kernels.** Furthermore, we propose to combine the personalized and the collective kernels into a single kernel function to have the best of the two worlds. Using that every user has their own individual model with all data created by that user and whenever that information is insufficient, the collective part of the kernel may help out. The combined feature map $\psi^{pc}$ is given by

$$\psi^{pc}(\mathbf{x}) = \psi^p\Big(\psi^t(\mathbf{x}) \cup \psi^t(\mathbf{x} \setminus u)\Big),$$

and leads to the *personalized and collective kernel* $k^{pc}$,

$$k^{pc}(\mathbf{x}, \mathbf{z}) := \langle \psi^{pc}(\mathbf{x}), \psi^{pc}(\mathbf{z}) \rangle. \tag{7.2}$$

As a result, three models are considered: a collective model, a personalized model, and a personalized+collective model. Note that the former learns the same parameters for all the users. To shed light on the characteristic traits of the proposed kernels, we showcase their feature spaces on the example of a user `u-742` with context $\mathbf{s}$ = `red`, `women`, `shoes`, `sneakers`, `Adidas` currently viewing an item $b$ = `white`, `women`, `shoes`, `sneakers`, `Nike`. Table 7.1 shows the resulting features for the collective, personalized, and personalized+collective feature maps. Note that we ignore the hash functions for a moment, which would map the resulting strings to some numbers.

Table 7.1: Exemplary feature mappings.

| Collective | Personalized | Personalized+Collective |
|---|---|---|
| `red::white` | | `red::white` |
| `red::women` | | `red::women` |
| `red::shoes` | | `red::shoes` |
| ... | | ... |
| `Adidas::sneakers` | | `Adidas::sneakers` |
| `Adidas::Nike` | | `Adidas::Nike` |
| | `u-742::red::white` | `u-742::red::white` |
| | `u-742::red::women` | `u-742::red::women` |
| | `u-742::red::shoes` | `u-742::red::shoes` |
| | ... | ... |
| | `u-742::Adidas::sneakers` | `u-742::Adidas::sneakers` |
| | `u-742::Adidas::Nike` | `u-742::Adidas::Nike` |

**Preference-based Transaction Kernels.** Finally, to leverage pairwise preference data for the recommendation scenarios, we consider user preferences of the form $\{b_t \succ b'_t \mid u_t, \boldsymbol{s}_t\}_{t=1}^{T}$, indicating that user $u_t$ prefers item $b_t$ over $b'_t$ in a given context $\boldsymbol{s}_t$ at time $t$. Every preference is thus translated into $\mathbf{x}_t = (b_t, u_t, \boldsymbol{s}_t)$ and $\mathbf{x}'_t = (b'_t, u_t, \boldsymbol{s}_t)$. Note that additionally available data sources are simply appended in the representation.

Using a linear model with parameters $\boldsymbol{w}$, we model the preference data such that $\boldsymbol{w}^\top \psi^{pc}(\mathbf{x}_t) \geq \boldsymbol{w}^\top \psi^{pc}(\mathbf{x}'_t)$, and due to linearity, we obtain

$$\boldsymbol{w}^\top \left(\psi^{pc}(\mathbf{x}_t) - \psi^{pc}(\mathbf{x}'_t)\right) \geq 0. \tag{7.3}$$

After certain transformations, the representer theorem (Schölkopf et al., 2001) allows to rewrite the primal parameters as

$$\boldsymbol{w} = \sum_{t'} \beta_{t'} \left( \psi^{pc}(\mathbf{x}_{t'}) - \psi^{pc}(\mathbf{x}'_{t'}) \right)$$

for dual variables $\beta_{t'}$. Plugging this result back into Equation (7.3) shows that all data-driven parts are of the form

$$\left\langle \psi^{pc}(\mathbf{x}_t) - \psi^{pc}(\mathbf{x}'_t), \psi^{pc}(\mathbf{x}_{t'}) - \psi^{pc}(\mathbf{x}'_{t'}) \right\rangle.$$

Subsequently, expanding the term gives

$$\langle \psi^{pc}(\mathbf{x}_t), \psi^{pc}(\mathbf{x}_{t'}) \rangle - \langle \psi^{pc}(\mathbf{x}_t), \psi^{pc}(\mathbf{x}'_{t'}) \rangle - \langle \psi^{pc}(\mathbf{x}'_t), \psi^{pc}(\mathbf{x}_{t'}) \rangle + \langle \psi^{pc}(\mathbf{x}'_t), \psi^{pc}(\mathbf{x}'_{t'}) \rangle,$$

and using Equation (7.2) leads to the desired *preference-based transaction kernel* that is given by

$$k^{ppc}(\mathbf{x}_t \succ \mathbf{x}'_t, \mathbf{x}_{t'} \succ \mathbf{x}'_{t'}) = k^{pc}(\mathbf{x}_t, \mathbf{x}_{t'}) - k^{pc}(\mathbf{x}_t, \mathbf{x}'_{t'}) - k^{pc}(\mathbf{x}'_t, \mathbf{x}_{t'}) + k^{pc}(\mathbf{x}'_t, \mathbf{x}'_{t'}).$$

Note that in the experiments, we also evaluate the other proposed kernels, $k^p$ and $k^c$, as well. The kernel can be plugged into a binary support vector machine or any other kernel machine. In case of the former, every preference encodes a positive example which renders the problem a binary ranking task. Note that thresholds cancel out in Equation (7.3); hence, the optimal hyperplane has to pass through the origin.

## 7.4 Informed Sampling for Recommendation

In order to recommend the most relevant item(s) to a user $u$ in a given context $\boldsymbol{s}_t$, we aim to maximize the described utility function over all products $b \in \mathbb{B}$. In many tasks with only a small set of items and a fast utility function, a complete enumeration, i.e., an evaluation of every possible product, might be feasible to find the best-rated item within $\mathcal{O}(|\mathbb{B}|)$. However, in real-world applications, the size of the item set is very large, and an efficient technique is required for online recommendations. In this section, we present a variant of MCTS for reducing the search space and efficiently finding the (near-) optimal candidates.

Therefore, we aim at minimizing the computational costs of evaluating the utilities of the items while still obtaining a reasonable approximation of the optimum. One way to minimize these costs is to limit the number of examples that can be considered for computing the maximum value. This implies a trade-off between the utility value of the returned optimal product and the number of items under consideration, i.e., we strive for finding (near-) optimal products within a bounded number of items. As a consequent, we incrementally build up a search tree.

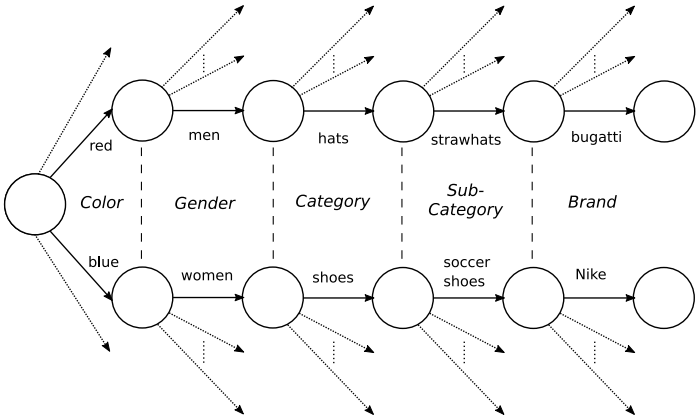



Figure 7.1: The structure of search tree in MCTS with five categorical layers for two products: a red straw hat for men and blue soccer shoes for women.

### 7.4.1 Structure of the Search Tree

Every product in our setting is characterized by a set of five categorical features: **color**, **gender**, **category**, **sub-category**, and **brand**. Figure 7.1 illustrates the search tree for an exemplary scenario. The tree is constructed in such a way that each layer of the tree corresponds to one categorical feature. The actions/arcs between two layers correspond to values of this feature, e.g., to assign the color of a product to `blue`. Hence, a trajectory starting from the root node chooses a value for the first feature, followed by a value for the second one, and so on. At the leaf nodes of the tree, all five features have been allotted to certain values so that a complete item description is available ensuring that each leaf node corresponds to a real product. For instance, choosing action `shoes` in depth 3 sets the third feature of the product to `shoes`, making `shirts` an illegal action at the next steps. The description is further used with the current user and context to query its utility value.

The tree is built up incrementally using a variant of Monte Carlo tree search. MCTS is an online tree search algorithm to explore the state space and find the optimal states for a given utility function. The key idea in MCTS is to incrementally construct an asymmetric partial search tree, guided by the estimates for the encountered actions (Kocsis & Szepesvári, 2006). The tree is expanded deeper in branches with most promising actions, so that less time is spent on evaluating less promising action sequences. In our setting, the trees are formed such that the nodes correspond to product features and the different combination of leaves are the products to recommend. Therefore, products with features that lead to higher utility value will be more likely to be sampled than products with uninteresting features. Note that MCTS is

an anytime algorithm, i.e., the algorithm can be stopped at any time and will provide the best result encountered up to that point.

The MCTS algorithm consists of four successive steps that are iterated for each new example (Browne et al., 2012). The *selection step* follows a path through the tree until a leaf is reached. The *expand step* adds a child of this node to the partial tree. Unless this leaf is already a terminal state in the search space (in our case a product with all features specified), Monte Carlo sampling is used to sample a random terminal state below the current leaf (the *rollout step*). The value of the terminal state, in our case the utility value of the sampled product, is then propagated back through all nodes up to the root node (the *backpropagation step*). These backed up values in the nodes are later used in the next selection step.

The next node to expand in the selection step of MCTS can be chosen in many different ways. Nevertheless, we benefit from upper confidence tree method (Kocsis & Szepesvári, 2006), which treats each node as a bandit problem. More precisely, in each node, it selects the action $a_t$ from $\mathbb{A}$, which maximizes the term

$$a_t = \arg\max_a \left( \bar{r}_a + 2 \cdot \upsilon \sqrt{\frac{2 \ln \bar{T}_a}{T_a}} \right), \tag{7.4}$$

where $\bar{r}_a$ is the average value propagated through the $a$-th node, $T_a$ is the number of times a value has been propagated through this node, while $\bar{T}_a$ is the number of times a value has been propagated through its parent node. In addition, $\upsilon$ is a parameter to trade off the two terms, which corresponds to exploitation (focusing on the best parts of the tree) and exploration (focusing on unexplored parts of the tree).

### 7.4.2 Algorithmic Modifications

MCTS aims at minimizing the regret, which means to maximize the sum of values gathered per iteration (Browne et al., 2012). Once a leaf node with a high value is found and added to the partial search tree, it is very likely that the algorithm will visit that node again to get a better estimation. For most applications of MCTS, this is a desired behavior. The tree will converge in such a way that the best action at the root node is the most frequently selected action.

However, a recently evaluated item does not necessarily need to be re-evaluated, since its value remains the same due to the deterministic setting of our problem. Hence, we remove the actions/arcs from the search tree if all products reachable from this edge have already been evaluated in a previous iteration. In this way, we ensure

that products are sampled at most once, while the search focuses on products that match the important properties in the higher levels of the tree. Therefore, in order to select the best product, we do not consider the most frequently visited actions, as the base MCTS algorithm would do. Instead, we keep track of the top encountered items with respect to the utility value. Although we only consider recommendations for the best product in this chapter, the framework can be easily extended to do top-$K$ recommendations.

## 7.5 Empirical Study

Once again, we conduct our experiments on a real-world dataset from Zalando, as described in Chapter 4 and 6. The dataset chosen for this chapter contains pairwise preferences of about ∼680k users on ∼16k items in ∼3.5m total transactions. A transaction occurs when a user prefers an item over another item, and any other click data is ignored. As mentioned earlier, every item is characterized by a set of five categorical features: **color**, **gender**, **category**, **sub-category**, and **brand**. The evaluation of our proposed approach is twofold: Firstly we study the performance of the personalized user preferences, and secondly, the efficiency of finding optimal items from the feature-based search tree is explored.

### 7.5.1 Performance of Preference Model

The pairwise preference model is obtained by training a support vector machine with the proposed transaction kernel. To assess the quality of our model, we run the SVM with different feature representations, i.e., kernel functions, and compare their performances. Note that the unmodified *string* features of products are used in the hash function as shown in Table 7.1 with a hash size of $2^{17}$. Moreover, SVM trade-off parameters are optimized by model selection techniques.

The simplest feature representation in our experiments uses only the five attributes of the products and discards user or context features. That is, $\psi(b_t, u_t, \boldsymbol{s}_t) = \boldsymbol{x}_t$, where $\boldsymbol{x}_t$ is the feature vector of item $b_t$ which are one-hot encoded and the dimensionality is about ∼1000. We refer to this representation as "**product only**". A second baseline concatenates all available features into a single vector. Since the user identifier is hardly an informative quantity, we replace it by clicking frequencies of item categories to form a feature vector for users. The frequencies are computed on historic data. The final representation is given by the vector $\psi(b_t, u_t, \boldsymbol{s}_t) = [\boldsymbol{x}_t; \mathit{freq}_{cat}(u_t); \boldsymbol{s}_t]$, which we refer to this representation as "**vectorized**". In addition, we include the collective

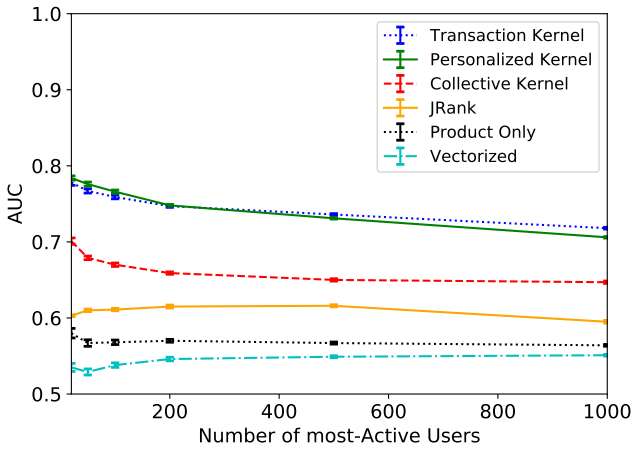
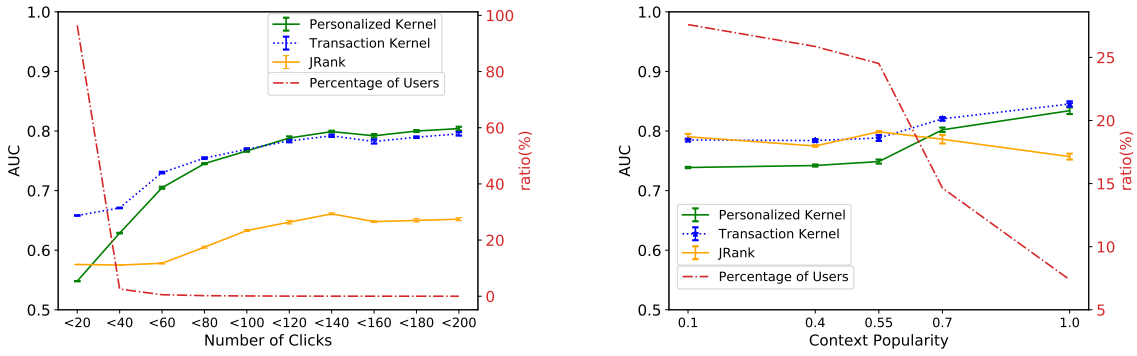



Figure 7.2: AUC values for SVMs with different kernels vs. baselines.

kernel and the personalized kernel as special cases of the transaction kernel in our experiments. We also compare the performances with **JRank** (Basilico & Hofmann, 2004), an online collaborative-based kernel perceptron that predicts ordinal ratings; this corresponds to learning a binary ranking in our scenario.

Figure 7.2 shows AUC (Area Under the ROC Curve) values obtained by a 10-fold cross validation for the different models. The users on the x-axis are ordered from left to right according to their click frequencies, i.e., the leftmost user has the highest number of clicks while the rightmost user ranks 1000th on that scale.

The results clearly demonstrate that the information encoded in the baselines "product only" and "vectorized"is not sufficient to accurately learn the preferences. On the other hand, the poor performance of JRank is caused by the sparsity of the data as well as cold start problem. JRank tries to remedy cold start issues by incorporating attributes of users and items into the model; however, compared to including the short-term context as in our model, there is only little to incorporate for JRank in this setting. Furthermore, the correlation kernel in JRank depends on collaborative information extracted from the user-item matrix which is too sparse in the problem at hand to capture accurate correlations between users as well as items. The collective kernel alone also does not perform well compared to the personalized and the transaction kernel. The reason for this lies in the choice of the users. Since all users expressed many preferences, they left enough data for the personalized kernel to capture their characteristic traits. However, at about rank 200, the performance of the transaction kernel increases over the personalized kernel as the collective kernel kicks in. The users on click rank 200 to 1000 clearly benefit, if only slightly, from the




(a) (b)

Figure 7.3: AUC values for (a) various click distribution of users, (b) different context popularity.

inclusion of the collective model into the transaction kernel as for them, the amount of available data alone is insufficient to train a highly accurate personalized approach.

To illustrate this effect, consider Figure 7.3 (a). The x-axis shows the different numbers of clicks of the users together with their distribution (dashed, red, y-axis on the right side). Simultaneously, the figure plots the AUC of the transaction and the personalized kernel (y-axis on left side). In terms of AUC, users who have about 120 clicks or more are better off with a purely personalized model that is only trained on their data alone. However, the red curve shows that these users are only a vanishing minority of all users. The distribution of clicks clearly follows an exponential law where the majority of users have only one or two dozens of clicks. For them, the transaction kernel leverages the collective kernel so that the majority benefits, even though they have only little data to share. On the other side of the scale, the heavy hitters do not lose too much in terms of AUC if the transaction kernel is used. This renders the proposed approach the best representation in this study. Note that the two left-most points of the figure involve data at large scales and the first cross validation fold for JRank took more than ≈30 days. We thus resort to showing only the results of this first fold; the respective standard errors are consequently missing on the left hand side of the figure.

Additionally, Figure 7.3 (b) draws a similar picture for context popularity instead of user clicks. We choose a smaller subset of data for this experiment to be able to evaluate the baseline in a reasonable amount of time, which leads to an overall higher AUC in all the approaches. The figure confirms that the more popular the context, that is, the more often we have seen a particular context in the data, the better the performance of the transaction kernel. Furthermore, popular contexts

are rare, and again the majority of users create new and unseen contexts. However note that the transaction kernel clearly outperforms the personalized kernel for all contexts, irrespectively of their popularity. JRank does not rely on any context related information and is more or less unaffected by the context popularity.

We also investigate the effect of the size of the hashing function. Figure 7.4 shows the results for data from the 200 most active users. The more bits are used, the larger the resulting hashed space. Smaller numbers of bits lead to collisions that introduce noise into the representation as the mapping is no longer one-to-one but one-to-many.

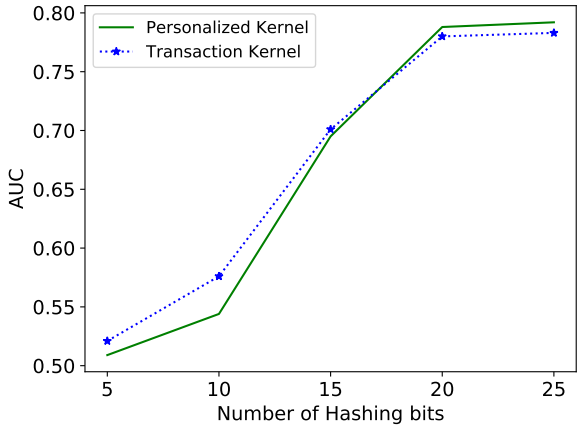



Figure 7.4: Influence of different hash sizes in terms of AUC.

### 7.5.2 Performance of Informed Sampling

In order to evaluate the effectiveness of our MCTS-based sampling, we compare the performance with other search strategies. We choose random subset exhaustion and greedy stochastic search as the baselines.

Random Subset Exhaustion (**RSE**) is a simple way to approximate the optimal element of a countable set without taking any structure into account. Given a fixed limit on the number of products that can be tested, it takes a random sample of the given size and exhaustively determines the best discovered item within this sample.

Greedy Stochastic Search (**GSS**) on the other hand explores the search tree shown in Figure 7.1 in a greedy fashion. As we cannot evaluate incomplete products, we first randomly initialize a product, and then explore a stochastic neighborhood of this product. This neighborhood is formed by sampling a fixed number of $M$ products, where half of them are randomly selected among those that differ only in brand (distance $d = 2$ edges in the tree of Figure 7.1), one quarter of the products differ

in brand and subcategory ($d = 4$), and, in general, $\frac{1}{d}$ percent of the products in the neighborhood are randomly selected among those with distance $d$. Subsequently, all products among the $M$ products in this neighborhood are evaluated and the search continues with the best one among those. The procedure repeats until all resources are exhausted, and the best encountered product is returned.

We run the experiments on several user-context pairs, where we randomly select three existing users and one new user without training data. For the context, i.e., the currently displayed product, we randomly select 50 items which leads to $4 \times 50 = 200$ different settings. The results of each context-user pair and parameter setting are averaged over 50 runs for MCTS and GSS. For random subset exhaustion, we show the average over all possible subsets, i.e., the expected value for exhaustively searching a randomly selected subset of that size.

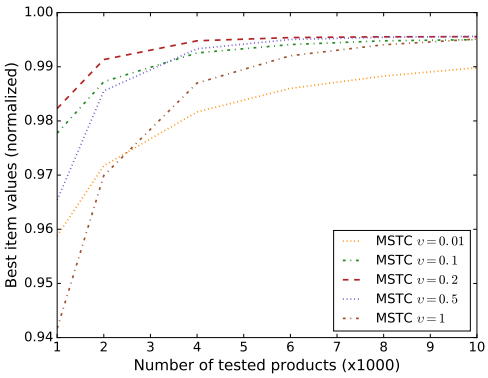



Figure 7.5: Performance of MCTS w.r.t. different parameters.

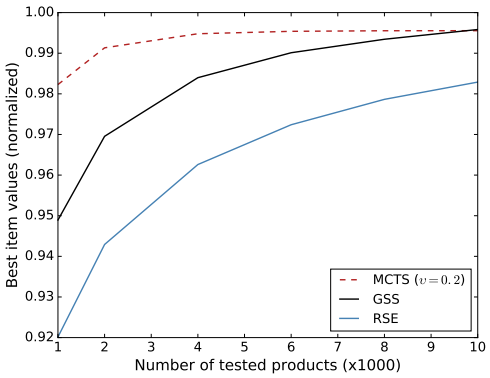



Figure 7.6: Performance of MCTS compared to baseline methods.

We first evaluate the performance of the proposed MCTS-based method for various values for the parameter $\upsilon$ that we choose from $\{0.01, 0.1, 0.2, 0.5, 1\}$. Figure 7.5 displays that the value of $\upsilon = 0.2$ achieves the best result for different numbers of tested products in our setup.

We further evaluate the performance of the MCTS approach compared to the two introduced baselines. Figure 7.6 shows the average values for the best found product for different numbers of items under evaluation. The results acknowledge that informed sampling via MCTS outperforms RSE for the same number of products and all maximum sample sizes. The advantage can be observed for all settings of the parameter $\upsilon$, although the magnitude of the advantage varies. In fact, even for the smallest considered sample size (1000), the best parameter setting of MCTS sampling ($\upsilon = 0.2$) finds an item that RSE finds over 10 times as many samples. The

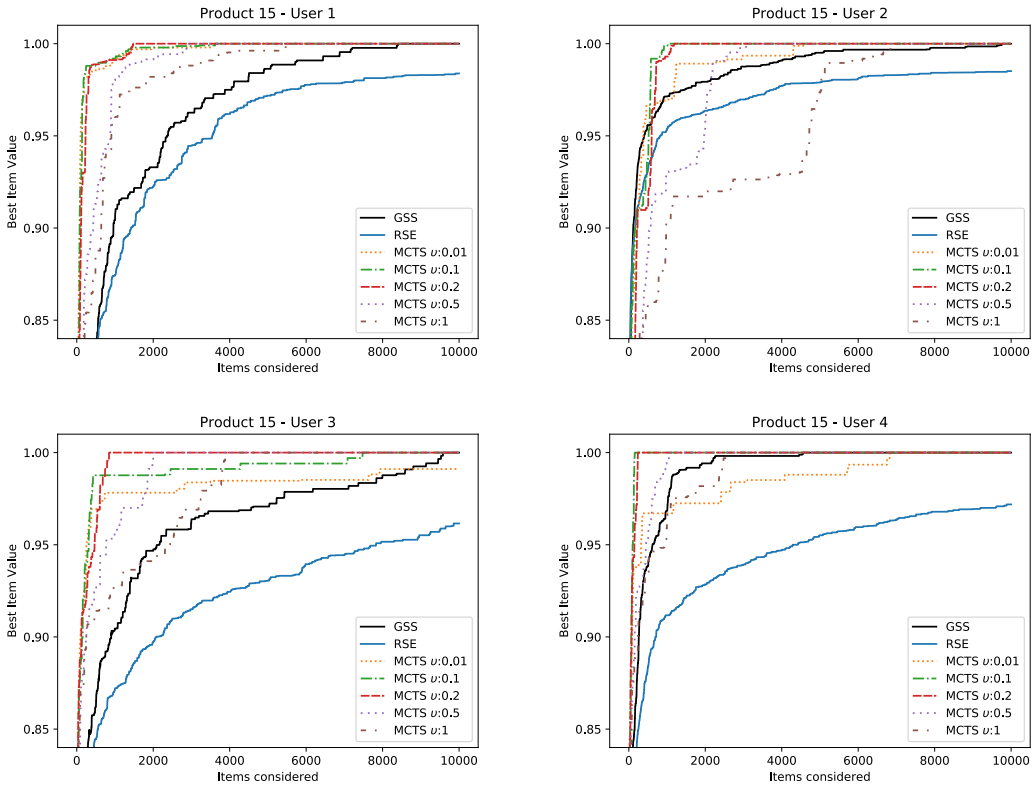



Figure 7.7: Performance evaluation on four users and one product. User 4 is a new user without any training data.

performance of GSS however lies between RSE and MCTS until 10,000 products, and then starts to slightly outperform MCTS with an average value of over 99.5%. The parameter value of $\upsilon = 0.2$ performs the best over all numbers of items considered for evaluation. It is not surprising that this value favors exploitation, since our algorithmic modifications of MCTS already enforces some exploration.

In Figure 7.7, we show a more detailed analysis for a single product (hand-picked as a representative case). The values of the best found item are shown for different users and different parameter settings. For all selected users, MCTS with $\upsilon = 0.2$ finds a better product within 2,000 products than RSE does within 10,000 products. There are extreme cases like for *User 4*, who is a new user without training data, where MCTS is able to find the best-rated item very quickly. In the case of *User 2*, nevertheless, a bad choice in MCTS for some values of parameter $\upsilon$ shows a worse performance than RSE, but this is a rare case (compare Figure 7.5). As Figure 7.5 shows, the value of parameter $\upsilon$ depends on the number of products considered for testing. Exploration (higher values of $\upsilon$) is more important for larger sets of products.

In summary, MCTS is able to find (near-) optimal products using a considerably

lower number of samples compared to GSS or RSE. Despite the fact that, at $10,000$ tested products, i.e., over 60% of all products, GSS has a better average performance than MCTS, both algorithms already achieve values over 99.5%. For real-time use cases it might be more interesting to reduce the number of tested products rather than reaching accuracy of over 99.5%. According to Figure 7.6, MCTS can reduce the number of querying the utility function by a factor of 10 with respect to RSE. However, this reduction in the number of tested products comes with higher computational costs. Therefore, the choice of sampling strategy depends on the problem at hand. Naturally, factors like evaluation time of the utility function, depth of the product tree, number of products, and desired item values are all important.

## 7.6 Conclusion

In this chapter, we presented an effective and efficient preference-based learning approach with which to model the personalized interests of users in contextual settings. We devised transaction kernels from pairwise preference data which combine theories from tensor products with hashing functions to capture individual, as well as collective, user preferences. The kernel functions were used in training a preference model via support vector machines to predict the utility of various products for a given user and context. Subsequently, we proposed a variant of the Monte Carlo tree search method for the efficient retrieval of (near-) optimal items for online recommendation purposes.

Empirically, on a real-world transaction dataset, both the preference models as well as the search tree exhibited excellent performances over baseline approaches, particularly in cases in which only a small number of products could be sampled. Our results showed that the personalized transaction kernels modeled user preferences accurately, and our informed search algorithm performed best with a considerably smaller number of sampling iterations compared to random subset exhaustion or greedy stochastic search.

# Chapter 8

# Extension to Continuous Scenarios

In all previous chapters, we focused on standard recommendation problems in which the system recommends from a finite set of available items $\mathbb{B}$. Additionally, contextual models were proposed to remedy cold start situations by generalizing over the item feature space when items are unavailable in the original set. Nevertheless, certain scenarios deal with uncountable and infinite sets, and these scenarios are not addressable using standard approaches.

In this chapter, we study an entirely new application for sequential recommendation problems. Consider a soccer match in which players carry out actions in teams in order to achieve a collective goal. The players' trajectories thus form series of movements that terminate in successful or unsuccessful sessions. We study the problem of recommending the next movement of individual players in the continuous space such that the collective movements lead to an eventual reward/success. In the first section, we introduce the problem on the example of soccer and motivate our proposed approach. Next, related work is presented in Section 8.2. The main contribution of this chapter is described in Section 8.3. Section 8.4 reports on the empirical results, and Section 8.5 concludes.

## 8.1 Motivation

Learning to assess individual behavior in an environment that contains several agents with potentially concurring goals has received a lot of attention recently (Foerster et al., 2018; Lowe et al., 2017), spurred, in part, by the great success of deep reinforcement learning models in single agent domains.

In this chapter, we focus on cooperative multi-agent problems in spatial environments by using soccer as an example. Figure 8.1 illustrates a snapshot of a professional soccer game in which individual players behave such that the collective is likely to

score. We aim to assign a rating to every agent's movements which reflects the quality of these movements with respect to the collective goal and leads to the identification of optimal behavior for recommendation purposes. Nonetheless, the movements in the continuous space are infeasible to fully evaluate and difficult to sample. To do so, we propose to learn the possible (and likely) movements of individual players in the unconstrained continuous space.

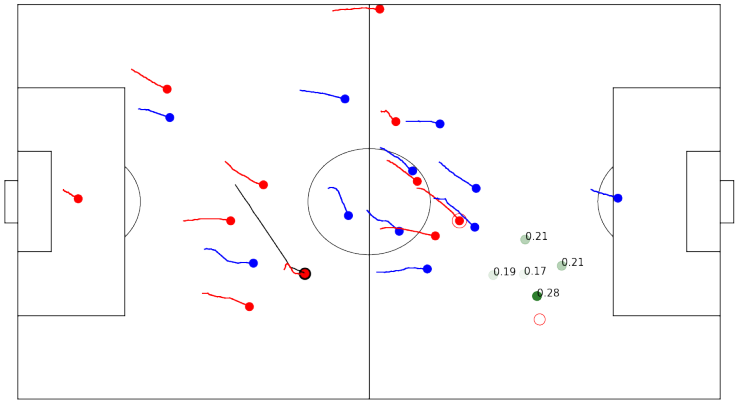



Figure 8.1: Learned ratings of potential movements of the circled red player (green dots, the darker the higher the rating). The red circle denotes the true movement that the player is about to perform.

We model the problem in a spatial multi-agent environment in which agents pursue a joint objective and benefit from multi-agent RL methods in terms of learning the values of the movements of the players. The value function, in one hand, facilitates the detection of good (near-optimal) actions in different game situations for use in, for instance assisting a coach during training sessions. On the other hand, we rate the movements of individual players with respect to their potential for staging a successful attack. These ratings are then utilized further to evaluate the performances of players during the game. Therefore, we translate the spatial multi-agent problem into a Markov Decision Process (MDP) in which agents perceive the positions and movements of other agents as a state representation which is possibly augmented by environmental variables such as the position of the ball. Given the state, every agent decides on a movement from a continuous action space that aims to support the team in receiving a joint reward.

Successful strategies that ground in simulation (Copete et al., 2015; Hausknecht & Stone, 2016) are, unfortunately, not applicable in our scenario, in which historic games are the only data source. On the other hand, credit assignments by experts are

tedious and only available for a few manually-selected scenes per game and are usually grounded in heuristics (Fernandez & Bornn, 2018; Link et al., 2016). To remedy the need for experts, we take a purely data-driven approach without external information and focus on learning from observation. On the example of soccer, we demonstrate that our complementary approach to expert-based ratings provides meaningful scores that highly correlate with the manually-created ones furnished by experts.

Therefore, we propose an agent-centric deep reinforcement learning approach with which to rate agent movements in spatio-temporal decision making problems. The real distribution governing each agent's movements in the continuous action space is unknown and can only be observed via examples. We thus introduce a mixture model that models the unknown distribution of actions, allowing for efficient learning, sampling and evaluation of movements when the examples are limited.

Given the mixture model, we devise a deep reinforcement learning-based approach to learn the values of the mixture components (the ratings of actions). Our approach allows us to efficiently evaluate the advantage values of the actual movements of the agents in any situation. Both policy and ratings of movements are learned simultaneously by jointly optimizing the likelihood and a variant of the Bellman error via convolutional neural networks. The deep model renders the input independent from the roles and positions of the agents, and the convolutions capture the entire environment from each agent's perspective. As a result, our purely data-driven approach allows for efficient learning and sampling of ratings in the continuous action space.

## 8.2 Related Work

Cooperative multi-agent scenarios where multiple agents act in a shared environment and pursue a joint objective are usually addressed with multi-agent RL techniques (Littman, 2001; Tan, 1993). These approaches are often attributed to either *team learning* aiming at models in joint action spaces or *concurrent learning* focusing on individual agents (Panait & Luke, 2005). Claus & Boutilier (1998) demonstrate that the convergence of cooperative concurrent learning could be nearly obtained as in a single-agent setting. Additionally, Omidshafiei et al. (2017) present a deep learning approach for partially observable multi-agent tasks, and Foerster et al. (2018) focus on learning decentralized policies while a central critic is optimizing independent behaviors. Moreover, deep RL has been successfully applied in game contexts (Lanctot et al., 2017; Mnih et al., 2015; Silver et al., 2016, 2017).

In relation to soccer, an actor-critic deep model for an abstraction of RoboCup has been proposed to learn the value of actions (Hausknecht & Stone, 2016). The action space contains four discrete actions with continuous parameters, one of which captures the magnitude and direction of movement for each agent and thus resembles our movement policies. Agents learn to play the game from scratch in a fully simulated setting, rendering the approach unsuited for problems like team sport that cannot (yet) be simulated.

Zhan et al. (2018) and Le et al. (2017b) employ deep imitation learning to model coordinated movements of agents in the problem of basketball and soccer, respectively. Similarly, Le et al. (2017a) use imitation learning to compare the behavior of soccer players to the average performance of either the player herself or a team average. Although the analysis is based on very interesting movement patterns, the average performance does not serve as a reliable evaluation of player movements as it is oblivious of situations and contexts. By contrast, our approach evaluates whether an agent's movement leads to maximizing the actual reward of an episode.

Player rating in soccer is a fairly new topic which aims at quantitatively ranking players instead of subjective measures. The related works consist of approaches based on defining game indexes (McHale et al., 2012), space generated and occupied by players (Fernandez & Bornn, 2018), and the location of completed passes (Brooks et al., 2016). We present a model to learn the value of players movements that gives a fine grained measure of players performance and further can be used for coarse grained strategy recommendation. Other computational approaches of analyzing soccer matches often rely on domain knowledge (Bialkowski et al., 2014; Lucey et al., 2014) or deploy frequency-related criteria to narrow down the candidate space (Haase & Brefeld, 2014). Scoring opportunities are for instance rated by Spearman (2018) and Link et al. (2016).

## 8.3 Learning to Rate Actions

In this section, we present an MDP-based approach to learn ratings of continuous actions which are used to predict the next optimal movement of the individual players in sequential settings. The idea is analogous to the session-based topic model of Chapter 4, with two main differences. First, instead of modeling the feature space of items by independently searching the space of their attributes, we model the problem in a joint continuous space. Second, we extend the framework to multi-agent domains which are qualified for team learning.

Recall from Chapter 3 that an MDP is represented by a five-tuple $(\mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma)$. In our multi-agent setting, state and action spaces are jointly defined for a set of $m$ users (agents). A state $s \in \mathbb{S}$ describes the positions of the agents, $\{f^1, ..., f^m\}$, where $f^j \in \mathbb{R}^2$ augmented by some other features such as their direction of movement or team identifier. Additionally, $s$ encodes the environmental features, e.g., the position and speed of the ball, as well as the location of the goals. The state-dependent action space is denoted by $A(s)$, where an action $a \in A(s)$ is defined by the set of movements $a = \{a^1, ..., a^m\}$ of the agents between two consecutive states $s$ and $s'$, resulting in a continuous action space. Furthermore, a stochastic policy $\pi(a|s)$ describes the distribution of actions $a$ that agents in state $s$ will decide for.

A soccer game is split into episodes of uninterrupted ball possession phases of a team, hence rendering episodes of variable length. A reward is granted at the last state transition of every episode. We assume a positive reward if the attacking team carries the ball into the dangerous zone of the opponent while retaining ball possession, and zero otherwise. However, note that any other definition of a positive reward may be used as well.

State transition probabilities, $\mathcal{P}(s, a, s')$, are partly deterministic since an agent's position is the sum of previous positions and the performed movements, $(f^j)' = f^j + a^j$. Nevertheless, the movement of the ball is a part of the environment and is modeled as a random movement. Therefore, the movement distribution is governed by a stochastic policy $\pi(a|s)$. Although the quantity is unknown, samples from $\pi(a|s)$ are contained in the episodes. We compute the state-movement (state-action) values using the Bellman equation of the form

$$Q(s, a) = \mathbb{E}_{s', a' \sim \pi}\Big[\mathcal{R}(s, a) + \gamma Q(s', a')\Big]. \tag{8.1}$$

The value function recursively computes the cumulative discounted reward until the end of an episode, where $\gamma \in (0, 1]$ is a discount factor. In the remainder, we use $r$ instead of $\mathcal{R}(s, a)$ for simplicity in the notation. Equation (8.1) gives the value of the collective movements in the current state, while we are interested in valuating the movements of individual agents $a^j$. Hence, we define

$$\bar{Q}(s, a^j) = \int_{\tilde{a}:\tilde{a}^j = a^j} \frac{\pi(\tilde{a}|s)}{\bar{\pi}(a^j|s)} Q(s, \tilde{a}) d\tilde{a} = \int_{\tilde{a}:\tilde{a}^j = a^j} P(\tilde{a}|s, a^j) Q(s, \tilde{a}) d\tilde{a}, \tag{8.2}$$

as the value of movement $a^j$ of agent $j$ in state $s$, where $\bar{\pi}(a^j|s) = \int_{\tilde{a}:\tilde{a}^j = a^j} \pi(\tilde{a}|s) d\tilde{a}$. In other words, the value of a single agent's movement is the expectation over all collective movements in which that agent performs that movement.

Finally, when rating an agent's behavior, we generally focus on the *relative* value of a movement compared to the values of alternative movements the agent could perform in the same state. The relative gain of an action is often called the *advantage* of that action and is defined as

$$\mathcal{A}(s, a^j) = \bar{Q}(s, a^j) - \int_{\tilde{a}^j} \pi(\tilde{a}^j|s)\bar{Q}(s, \tilde{a}^j)d\tilde{a}^j. \tag{8.3}$$

Note that the integral on the right hand side, that is, the state value function $V(s)$, is the same for all agents.

### 8.3.1 Estimating the Action Model

We first propose a method to model the continuous action space such that the policy $\pi(a^j|s)$ allows for efficient sampling of *possible* movements for $a^j$. Note that we use the terms action and movement interchangeably in this chapter. We model $\pi$ as a mixture of $L$ two-dimensional Gaussians

$$\pi(a^j|s) = \sum_{l=1}^{L} \hat{\pi}(l|s, j)\mathcal{N}(a^j|\mu^l(s, j), \sigma^l(s, j) \cdot \mathbb{I}), \tag{8.4}$$

where the two-dimensional function $\mu^l(s, j)$ computes the mean of the $l$-th Gaussian of agent $j$ in state $s$. Similarly, function $\sigma^l(s, j)$ maps state $s$ to the positive width parameter of the spherical covariance matrix $\sigma^l(s, j) \cdot \mathbb{I}$ of the $l$-th Gaussian of agent $j$, and $\mathbb{I}$ denotes the two-dimensional identity matrix.

We estimate the parameters of the Gaussian mixture model by minimizing the negative log-likelihood on the samples via deep learning. Therefore, the parameters $\Psi_{\hat{\pi}}$, $\Psi_{\mu^l}$, and $\Psi_{\sigma^l}$ indicate the weights of the underlying deep network for the respective $\hat{\pi}$, $\mu^l$, and $\sigma^l$. Bear in mind that these parameters are shared across agents, however the values of the mixture model are computed for each player $j$ individually given arguments $(s, j)$. As a result, the model allows to sample values efficiently at those $L$ means (see also Figure 8.1 for an example with $L = 5$). The structure of the deep model is defined in Section 8.3.3

In addition, the estimates of $P(s'|s)$ are required for the optimization problem. Let $s_M$ be the part of the state that is defined by movements of agents such that $s_M = \{f^j; \forall j\}$, and let $s_B$ denote the remaining part. Then we have

$$P(s'|s) = P(s'_B, s'_M|s) = P(s'_B|s'_M, s)P(s'_M|s).$$

Although $P(s'_B|s'_M, s)$ remains unknown, samples of that can be drawn from the data. Therefore, minimizing the negative log-likelihood of $P(s'|s)$ reduces to $P(s'_M|s)$. We

further assume that movements of agents are independent given the state and rewrite the state transition function in terms of Equation (8.4) as

$$P(s'_M|s) = \prod_j \pi(a^j|s).$$

Consequentially, the negative log-likelihood of the movement model, $\Omega_L$, is

$$\Omega_L(\Psi_{\hat{\pi}}, \Psi_{\mu^l}, \Psi_{\sigma^l}) = -\sum_{(s,a,s'),j} \log\left(\sum_l \hat{\pi}(l|s,j)\mathcal{N}(a^j|\mu^l(s,j), \sigma^l(s,j))\right). \tag{8.5}$$

Estimating the agent's movement model allows for learning the value function $\bar{Q}$ and the advantage function $\mathcal{A}$ (Equation (8.3)) which we show in the next section.

### 8.3.2 Learning Value Function $\bar{Q}$

A general approach to learning the individual value functions $\bar{Q}$ is to interpret the Bellman equation (Equation (8.1)) as an optimality criterion by minimizing the *temporal differences* as described in Section 3.2.2. However, we only have access to a finite amount of historic data consisting of episodes of state, action, and reward, $(s, a, r)$, thus, the objective becomes

$$\Omega_{\bar{Q}} = \sum_{(s,a,r,s')} \sum_j \left(\bar{Q}(s, a^j) - \left(r + \gamma \int_{\tilde{a}^j} \pi(\tilde{a}^j|s')\bar{Q}(s', \tilde{a}^j)d\tilde{a}^j\right)\right)^2, \tag{8.6}$$

where every $(s, a^j)$ is a sample of the policy $\pi$ and accordingly $\Omega_{\bar{Q}}$ is an objective for an on-policy policy evaluation from a fixed dataset.

A standard approach to learning $\bar{Q}$ is using a purely sample-based method which approximates the integral with the actual observed sample $(s', a'^j)$ that results in $\bar{Q}(s', a'^j)$. Nevertheless, the number of training examples is limited for optimizing such an objective. Besides, we are interested in sampling realistic movements that players can actually perform. For instance, while in theory the value of running toward the goal with a speed of 45km/h is very high, practically, no player will be able to achieve it. Therefore, in order to find the best possible movement of a player in a given state, the model should only be queried for possible actions. This problem arises since the continuous action space is essentially unconstrained and without additional information on the boundaries of player's physical capabilities. Even with this information, enforcing those bounds can be hard as investigated by Hausknecht & Stone (2016), who proposed to adjust gradients during learning. Another approach is to discretize the space into a predefined grid (Zheng et al., 2016).

Instead, we use the action model introduced in Section 8.3.1 that imposes no constraints on the space of movements a priori, but is able to learn the possible movements of players. We therefore present a model-based approach by devising parametric versions of $\bar{Q}(s, a^j)$ and $\pi(a^j|s)$ that are simultaneously adapted to data by a deep architecture.

Moreover, our parametric approach allows to replace the integral in Equation (8.6) by a computationally inexpensive sum which leads to efficient computation of $Q$, $\mathcal{A}$, and $V$. Recall that the discrete set of actions is defined by the means of the Gaussian mixture model of agent $j$ in state $s$. Let $\hat{Q} : (s, \mu^l(s, j)) \to \mathbb{R}$ be a value function that maps those state-movement pairs to a real number with parameters $\Psi_{\hat{Q}}$. The general value function $\bar{Q}$ computes the expected value of the movements of the mixture model

$$\bar{Q}(s, a^j) = \sum_l P\big(\mu^l(s, j)|s, a^j, \sigma^l(s, j)\big)\hat{Q}(s, \mu^l(s, j)),$$

where $P\big(\mu^l(s, j)|s, a^j, \sigma^l(s, j)\big)$ is the posterior for the movement being generated by the $l$-th component of the action model. The following proposition shows that the integral in Equation (8.6) can be substituted with a sum over the mixture components.

**Proposition 1.** *The integral in Equation (8.6) can be written as*

$$\int_{\tilde{a}^j} \pi(\tilde{a}^j|s')\bar{Q}(s', \tilde{a}^j)\ \mathrm{d}\tilde{a}^j = \sum_l \hat{\pi}(l|s', j)\hat{Q}(s', \mu^l(s', j)).$$

*Proof.* We have

$$\begin{aligned}
\int_{\tilde{a}^j} \pi(\tilde{a}^j|s')\bar{Q}(s', \tilde{a}^j)\ \mathrm{d}\tilde{a} &= \int_{\tilde{a}^j} \pi(\tilde{a}^j|s') \sum_l P(\mu^l(s', j)|s', \tilde{a}^j)\hat{Q}(s', \mu^l(s', j))\ \mathrm{d}\tilde{a}^j \\
&= \int_{\tilde{a}^j} \sum_l P(\mu^l(s', j), \tilde{a}^j|s')\hat{Q}(s', \mu^l(s', j))\ \mathrm{d}\tilde{a}^j \\
&= \sum_l \int_{\tilde{a}^j} \hat{\pi}(l|s', j)\mathcal{N}(\tilde{a}^j|\mu^l(s', j), \sigma^l(s', j))\hat{Q}(s', \mu^l(s', j))\ \mathrm{d}\tilde{a}^j \\
&= \sum_l \hat{\pi}(l|s', j)\hat{Q}(s', \mu^l(s', j)),
\end{aligned}$$

where $\int_{\tilde{a}^j} \mathcal{N}(\tilde{a}^j|\mu^l(s', j), \sigma^l(s', j))\ \mathrm{d}\tilde{a}^j = 1$. □

Incorporating this result into $\bar{Q}$ yields a simplified form of $\Omega_{\bar{Q}}$ that reduces the learning of $\bar{Q}$ to learning the parametric function $\hat{Q}$ which is given by

$$\begin{aligned}
\Omega_{\bar{Q}}(\Psi_{\hat{Q}}; \Psi_{\hat{\pi}}, \Psi_{\mu^l}, \Psi_{\sigma^l}) = \sum_{(s,a,r,s'),j} \Bigg[ &\sum_l P\big(\mu^l(s, j)|s, a^j, \sigma^l(s, j)\big)\hat{Q}(s, \mu^l(s, j)) \\
&- \left( r + \gamma \sum_{l'} \hat{\pi}(l'|s', j)\hat{Q}(s', \mu^{l'}(s', j)) \right) \Bigg]^2 .
\end{aligned} \tag{8.7}$$

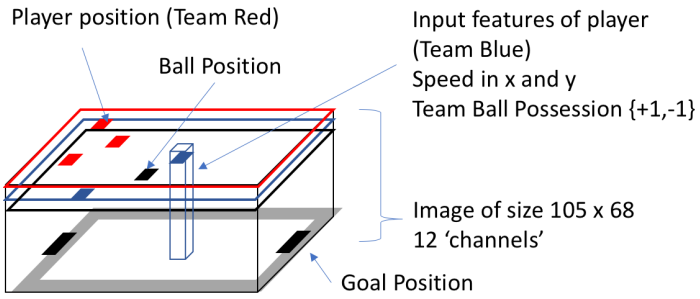



Figure 8.2: Input representation to the deep convolutional net.

Despite the fact that $\pi$ and $\bar{Q}$ could be learned in a pipe-lined approach one after another, we propose a deep model that allows to learn all parameters $\Psi = \Psi_{\hat{Q}} \cup \Psi_{\hat{\pi}} \cup \Psi_{\mu^l} \cup \Psi_{\sigma^l}$ within a single optimization problem

$$\min_{\Psi} \quad (\Omega_{\bar{Q}}(\Psi_{\hat{Q}}; \Psi_{\hat{\pi}}, \Psi_{\mu}, \Psi_{\sigma}) + \Omega_L(\Psi_{\hat{\pi}}, \Psi_{\mu}, \Psi_{\sigma})). \tag{8.8}$$

### 8.3.3 Deep Model

We benefit from a convolutional architecture which enables sharing large parts of the individual parameters, thus speeding up training as well as prediction. In this section, we present the deep convolutional model that is developed to learn both $\bar{Q}$ and $\pi$.

**Input:** Recall that the state consists of the position and direction of movements of all agents and the ball, augmented by additional information. The input to the net is thus represented by a three-dimensional 'image-like' tensor $\mathbf{I} \in \mathbb{R}^{105\times68\times12}$ shown in Figure 8.2. The tensor encodes the size of the pitch ($105 \times 68$) and provides 12 channels as follows: Channels 1-3 contain the positions of the agents (channels 1-2 for team red and team blue) and the ball (channel 3), respectively; tensor elements corresponding to the position of an agent or the ball are set to 1. Channels 4-9 contain the directions of movements and velocities of agents of team red (channels 4-5), team blue (channels 6-7), and the ball (channels 8-9). By doing so, input features that belong to a specific agent are given by the slice $(x, y, :)$ that is located at the agent's $(x, y)$ position[1]. Further channels encode ball possession (channel 10), the position of the goals (channel 11) as well as the boundaries of the pitch (channel 12). Note that channels 11 and 12 are constant and do not change over time.

**Architecture:** The introduced tensor is the input to a 6-layer deep convolutional net such that the receptive field of activations at the last layer spans over the

[1]The origin is the lower left of the pitch

Table 8.1: Per-Layer definition of the convolutional model.

| | kernel shape | #kernels | max. pooling |
|---|---|---|---|
| layer 1 | (5,5) | 32 | – |
| layer 2 | (5,5) | 64 | (2,2) |
| layer 3 | (5,5) | 64 | – |
| layer 4 | (5,5) | 128 | (2,2) |
| layer 5 | (5,5) | 128 | (2,2) |
| layer 6 | (5,5) | 128 | (2,2) |

whole pitch. The detailed definition of layers in the network is sketched in Table 8.1. Full feature representations of agents are computed by concatenating activations of convolutional layers as depicted in Figure 8.3. Hence, slices through all activation tensors are computed relative to the agent's position and then compiled into a single feature vector that consists of features of different levels of granularity and receptive fields, centered at the agent's position. An agent's feature representation connects to $5L$ outputs via a fully connected layer, where $L$ is the number of components in the Gaussian mixture model as defined in Equation (8.4) (cmp. Figure 8.4). Activations are linear for $\mu$ and $\hat{Q}$, whereas $\hat{\pi}$ uses an additional softmax function to ensure valid probability distributions, and $\sigma$ has an exponential activation function to ensure positiveness.

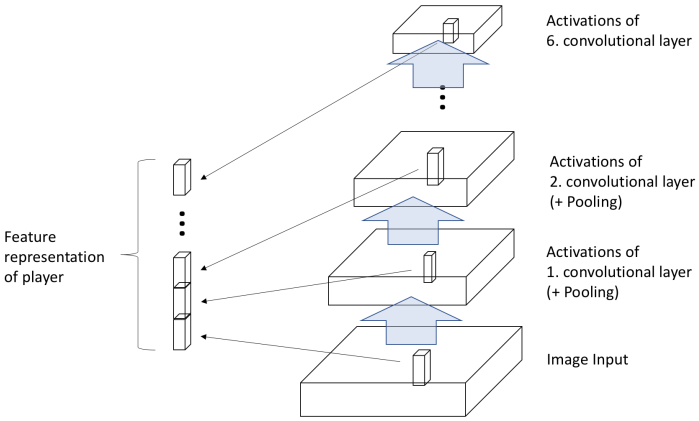



Figure 8.3: Feature representation for an agent: activations of all layers at the position of the agent are concatenated.

**Optimization:** Figure 8.4 shows that the feature representation of the agents serves as the input to the fully connected layer generating all outputs. The layout

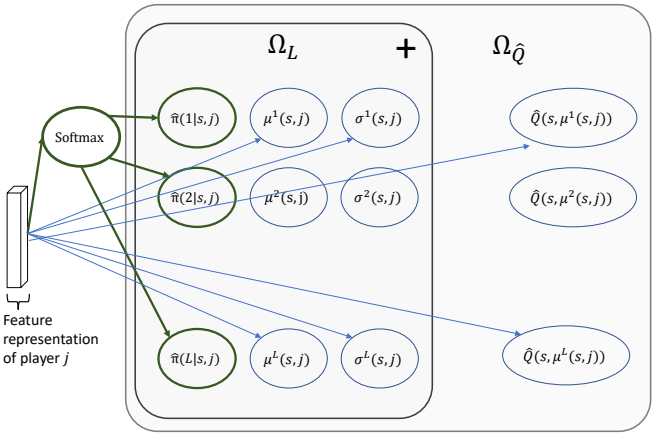



Figure 8.4: Output nodes and loss.

of the net imposes that a massive amount of the parameters of the convolutional net are shared between all agents and outputs. For a given input representation of a state $s$, the deep model outputs $\hat{\pi}(l|s,j)$, $\mu^l(s,j)$, $\sigma^l(s,j)$, and $\hat{Q}(s,\mu^l(s,j))$ for all agents $1 \leq j \leq m$ and mixture components $1 \leq l \leq L$. The parameters of the model are thus adjusted by minimizing the compositive objective function given in Equation (8.8), with standard optimizers for deep models such as stochastic gradient descent.

## 8.4 Empirical Study

We evaluate the performance of our model on trajectory data from the UEFA Women's Euro 2017 tournament. The dataset consists of six games of the Dutch team from the group stage to the final match. Each game comes as a sequence of $x$ and $y$-coordinates of all players as well as the ball sampled at 10 frames per second. Every frame is turned into an input tensor as described in Figure 8.2. We further compute the speed of players and the ball in both directions from the given positions.

We extract all episodes of open play, where one team continuously retains ball possession without the game being halted. Each such ball possession phase ends with the team controlling the ball either losing the ball or the play being stopped, or with that team performing a "*success action*". An episode is considered successful if the team carries the ball into an area that extends two meters outside the penalty box of the opponent. In case of a successful action, the episode is labeled with a positive reward, otherwise the reward is zero. We evaluate player movements with respect

to their potential of helping to stage a successful attack which serves as a proxy to determine optimal behaviors. We report on results of leave-one-game-out cross validations; that is, every result is the mean of a six-fold cross validation, where each fold contains all episodes of exactly one of the six games. Error bars in the figures indicate standard error.

### 8.4.1 Baselines

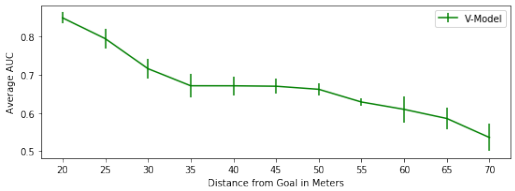
As mentioned in Section 8.3.2, a standard approach to learn the value function from data is to approximate the integral in Equation (8.6) with a single observed sample; we use this as a baseline. However, for rating players' movements, we are generally interested in the *advantage* function $\mathcal{A}$ of a movement to assess its quality. Subsequently, in order to compute the advantage values, we need an estimate of the state value function $V$ (cmp. Equation (8.3)).

We therefore consider three approaches to estimate $V$. The first two methods estimate the state value function from the learned $\bar{Q}$. Nonetheless, since the baseline lacks the discretized action model, we either uniformly sample actions from the space of possible movements[2], or use our action model to sample from the space of movements. These approaches lead to two different baselines which we call *Baseline with Random Sampling* and *Baseline with Model*, respectively. The third baseline directly estimates $V(s)$ using the same deep architecture as defined in Section 8.3.3. Instead of predicting values based on an agent's feature representation, a fully connected layer with a single output $V(s)$ is connected to the last convolutional layer. Thus, $V$ and $Q$ share all model parameters except those of the output nodes. $V$ is estimated by minimizing the expected TD error for states

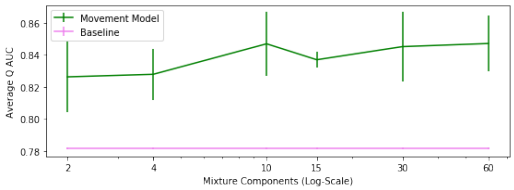
$$\min_V \sum_{(s,s',r)} \left(V(s) - (r + \gamma V(s'))\right)^2 .$$

To get a better understanding of the quality of the estimated $V$, Figure 8.5a shows the performance of state valuation in terms of average AUCs. We compute the value of a frame in a positively labeled episode and compare it with all frames of all negatively labeled episodes where the ball has the same distance to the opponent goal. We hence compute the probability that $V$ assigns higher scores to a frame from a successful episode than to one of a negative episode. Unsurprisingly, the closer the ball is to the opponent's dangerous zone, the better the AUC.

[2]We consider all movements with a speed below 30km/h possible.




(a) $V$ evaluation




(b) $Q$ evaluation

Figure 8.5: Performance in terms of average AUC.

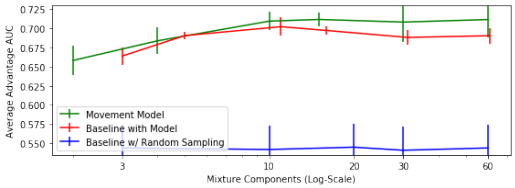
### 8.4.2 Movement Valuation

In this section, we evaluate the performance of our approach in terms of the ability to rate individual movements. An example of a rating for one player is shown in Figure 8.1 for $L = 5$ mixture components. The hypothesis is that the more the agents choose *good* actions, the more successful the team coordination and thus the higher the reward. Similarly in a real-world application, situations in which agents frequently choose (near-)optimal actions could be reinforced in the coaching. We validate the hypothesis as follows.

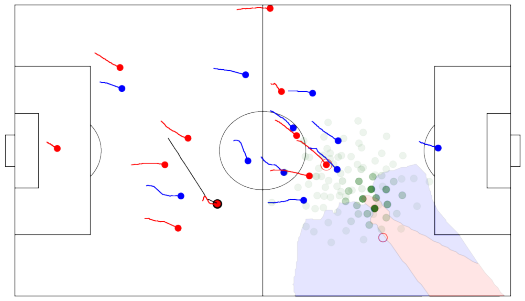
**Evaluation Measures:** We use Area Under the ROC curve (AUC) to evaluate both $Q$-values as well as advantage function $\mathcal{A}$. To evaluate $Q$, we compute the $Q$-values of the last completed movements of all players for each episode, where the duration of movements is 3 seconds. Therefore, the last completed action of a player starts 3 seconds before the end of an episode. The computed $Q$-values are averaged over all agents to compute a single value for every episode which is considered a prediction of the outcome of an episode. Hence, the higher the average $Q$, the higher the probability that the attack is successful. As a result, AUC values compare predictions with actual rewards of the episodes. We refer to this measure as *Q AUC*.

In order to evaluate $\mathcal{A}$, we compute the advantage function (Equation (8.3)) for every observed movement in an episode, and for all agents. The values are averaged over all agents and time steps to compute a single value for every episode. Analogous to *Q AUC*, these values are considered a prediction of the outcome of an episode, and are denoted as *Advantage AUC*. This measure demonstrates that the more qualified decisions the agents make during an episode, the higher the likelihood of success.

**Performance Results:** Figure 8.5b shows the average *Q AUC* for varying numbers of mixture components, with a prediction horizon of 3 seconds. The AUC values of our model slightly increase with the complexity of the model and significantly outperform the baseline in predicting $Q$-values. This result strongly indicates that using our model-based approach proves advantageous when learning $Q$ from limited data.




(a) Average Advantage AUCs



(b) Exemplary movement model prediction and occupation zones

Figure 8.6: (a) Performance of movement valuation, and (b) action model (the movements take 3 seconds long into the future).

On the other hand, Figure 8.6a plots the average *Advantage AUC* which also increases with the complexity of the model. The third baseline, that uses the learned $V$, is not shown in the plot due to giving a value of 0.489 in our experiments. The *Baseline with Random Sampling* performs poorly, nonetheless, the *Baseline with Model* that uses the same movement model as our model performs comparable to our approach. This highlights the benefit of learning a movement policy in order to asses the relative value of movements, where our model allows to sample efficiently. Even though our model performs only slightly better than the informed baseline, both approaches significantly outperform the baselines without movement models.

Table 8.2: Players ratings.

| SOG-based | | Our Model | | Newspaper | |
|---|---|---|---|---|---|
| 1:Sanden | 1.6 | 1:Sanden | 0.086 | 1:Martens | 8.5 |
| 2:Miedema | 1.5 | 2:Miedema | 0.032 | 2:Groenen | 8 |
| 3:Martens | 1.3 | 3:Donk | 0.005 | 3:Sanden | 7.5 |
| 4:Donk | 1.2 | 4:Groenen | 0.004 | 3:Miedema | 7.5 |
| 5:Groenen | 1.1 | 5:Martens | 0.003 | 4:Donk | 7 |
| 6:Spitse | 0.85 | 6:Dekker | 0.002 | 4:Dekker | 7 |
| 7:Van Es | 0.8 | 7:Gragt | -0.015 | 4:Gragt | 7 |
| 8:Dekker | 0.69 | 8:Spitse | -0.018 | 4:Spitse | 7 |
| 9:Gragt | 0.67 | 9:Van Es | -0.075 | 4:Van Es | 7 |

Moreover, the obtained advantage values of individual players are utilized to rate their performed movements. We average the advantage of the true movements of nine Dutch players (excluding the goal-keeper) who appeared in all six games over all time steps in the episode, and compute their performance via cross validation.

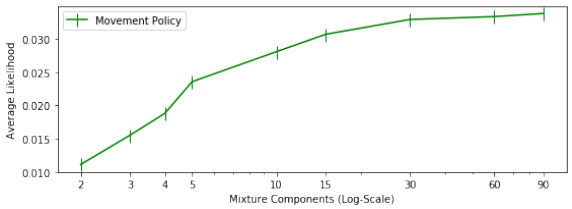



Figure 8.7: Likelihood for different numbers of components in the action model.

The models employ $L = 5$ mixture components and the movement duration is two seconds. We compare our ratings with expert rankings based on space occupation gain measure introduced by Fernandez & Bornn (2018) as well as scores from a newspaper[3]. Table 8.2 shows that our approach is in line with the two expert views.

### 8.4.3 Movement Model

An interesting application of the learned action model is to compute the possible whereabouts and their likelihood for all players at $t$ seconds in the future. The player with the highest likelihood wins a position. The resulting area won by an agent is sometimes called an agents dominant zone as she is likely to control this area (Brefeld et al., 2018; Taki & Hasegawa, 2000). Figure 8.6b shows the same game situation as in Figure 8.1. The green dots visualize $L = 90$ mixture components of the player in the red circle; darker green corresponding to higher likelihood. Prediction is performed for $t = 3$ seconds into the future. The red zone denotes the controlled zone of that player and the two blue zones correspond to the two blue defenders below and right of the player.

Additionally, we evaluate the performance of the action model with respect to the number of mixture components. Intuitively, the more mixture components, the better the approximation of the continuous space. Figure 8.7 confirms that and shows that the likelihood increases for further numbers of components, where the model predicts movements that span 3 seconds. See also Figure 8.8 for an exemplary situation with $L = \{5, 15, 90\}$ components. Darker green marks higher likelihood. Predictions are shown for the circled player and the real next position is depicted by an empty circle.

Furthermore, the movement policy nicely adapts to the context of the player as shown in Figure 8.9 which depicts two different exemplary situations of predicted movements by a learned model with $L = 90$ mixture components. The slowly moving agent on the left has a circular-shaped movement distribution as there is only little

[3]https://www.ad.nl/nederlands-voetbal/rapport-glanzende-cijfers-voor-de-oranje-leeuwinnen a986670a0/

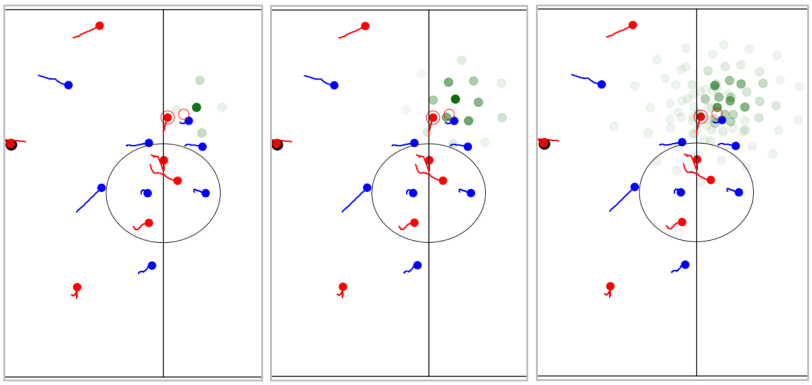


Figure 8.8: Predicted movements with mixtures using $L = 5$ (left), $L = 15$ (center), and $L = 90$ (right).

time needed for turning in any direction. The running agent on the right (marked by the red circle) realizes an ellipsoidal-shaped movement distribution.

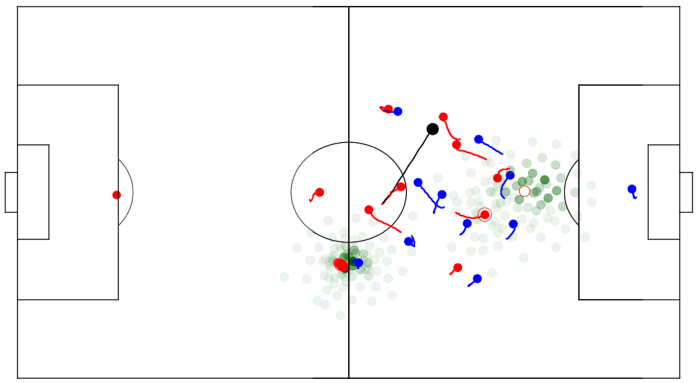


Figure 8.9: The running agent on the right realizes a very different movement distribution than the slowly moving player on the left.

## 8.5 Conclusion

In this chapter, we extended the sequential recommendation problems to continuous and multi-agent scenarios. We presented a purely data-driven approach to learning to rate agents' actions in collective spatial environments. Our method aimed at valuating fine-grained movements of agents in order to discover the optimal behaviors of agents in the continuous space.

Using soccer as the example, we took an MDP-based approach in which agents perceive positions and attributes of other players as the state representation and perform their movements from the continuous action space to pursue a joint goal. We

proposed an efficient deep architecture to simultaneously optimize an action model which models realistic movements and a value function that rates them. The choice of model facilitates efficient learning and sampling of actions. Empirically, we observed that our model predicted player movements accurately and successfully captured the contribution of individual movements to the collective goal. Moreover, our method led to sensible ratings comparable to the expert grading of professional soccer data.

# Chapter 9

# Conclusions

This thesis dealt with designing novel recommendation systems to serve user needs by providing tailored recommendations. More specifically, we focused on the sequential nature of the decision making process in recommendation problems to capture the interests of users over short- as well as long-term time periods. In addition, we concentrated on scenarios where those preferences are only inferable from implicit user behaviors rather than explicitly expressing their likings.

We successfully answered the central question of the thesis: how to model recommendation systems as sequential decision making processes, which naturally capture the behavioral patterns of users over short- as well as long-term spans of time. We presented novel approaches based on reinforcement learning to model recommendation systems in sequential settings while incorporating the contextual information of the users (when available), items, and the actual situation into the models. We empirically studied various applications with real-world and noisy datasets to evaluate the performance of our systems and showed that our contextual models outperform several baselines for different applications.

We first presented how to model recommendation systems in a sequential setting. We proposed an approach based on Markov Decision Processes (MDPs) to model the short-term interests of users in session-based recommendation scenarios, leaving the personalization out of the framework. We found out that the recent or short-term activities of users are important factors to identify the user intention or the topic of an ongoing session. Furthermore, factorizing the MDP framework over item attributes significantly helped our approach to scale up for the larger applications and settings. Extensive experiments on large-scale scenarios illustrated that our approach performs well in terms of efficiently detecting accurate topics of user sessions as well as recommending higher ranked items.

Furthermore, we showed that our sequential approach is applicable in other applications and domains. We utilized a simplified version of our MDP-based method and framed the task of recommending personalized itineraries as detecting the main category of the next point of interest. Hence, the problem became a special case of the topic model without multiple factors. We further personalized our model by exploiting the individual preferences of users from their long-term histories. Our empirical evaluation, on travel data collected from Flickr, exhibited promising performance compared to several baselines. However, we winded up concluding that this vanilla personalization technique is not sufficient for accurately modeling the individual preferences of users.

We thus incorporated the long-term personalized interests of users into the contextual models. We relaxed the Markov property of the MDP and introduced a contextual bandit approach, which models both short- and long-term preferences of users, simultaneously. We presented a general optimization method for our multi-armed bandit model in the dual space that operates for a wide range of tasks and settings. We further represented several extensions and special cases of our approach, where in the experiments, illustrated significant performance gain for various setups. Although the unified model was considerably beneficial for cold start problems, the computationally expensive run-time of the algorithm was a major drawback for large-scale scenarios.

To remedy the problem of long execution time that occurs in many personalization techniques, we proposed a much more efficient way for user-specific models using hash functions. We focused on the pairwise preference data to construct kernel functions that are employed in support vector machines to learn personalized preference models. The short-term contexts were captured via tensor products while long-term interests were modeled via individual hash functions. Using a bandit-based Monte Carlo tree search for recommending items showed to be suitable for efficient recommendations. Empirically, our approach performed effective and remarkably efficient compared to baseline methods. Although an end-to-end approach is considered to be more preferable, our pipe-lined method led to highly interpretable results, which renders it useful for various applications.

We were also able to extend the recommendation problems to continuous and multi-agent scenarios on the example of soccer. We studied sequences of trajectory data from players in soccer games to evaluate their fine-grained movements in different situations. We modeled the problem in a spatial multi-agent RL setting with an agent-centric deep neural network architecture. Both the descretized action model of

the players as well as their values were learned by the same deep model via temporal difference learning. Empirical studies on professional soccer games showed that our model accurately valuated agent movements with respect to their relative contribution to the collective desired outcome. However, our player-centric representation depended on their spatial position and did not model individual players.

In total, the thesis brings us to the conclusion that recommendation systems are theoretically and practically addressable using sequential approaches. We showed that zeitgeist and common trends highly influence user interests and are thus important in modeling personalized preferences. However, maintaining an individual model per user is costly and we need more efficient ways to cope with personalization. In addition, we found out that although a robust model seems to be the one that takes every aspect of the problem into account, there is no single model that rules them all. Different applications have diverse requirements and the model of choice depends clearly on the intrinsic dynamics of the applications. Still, there is abundant room for further improvements in recommendation systems, such as designing more scalable and more generic personalized approaches, which we leave for future works.

# References


G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pp. 217–253. Springer, 2011.

D. Agarwal and BC. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 19–28. ACM, 2009.

R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, volume 1215, pp. 487–499, 1994.

R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pp. 207–216. ACM, 1993.

L. AlSumait, D. Barbará, and C. Domeniconi. Online LDA: adaptive topic models for mining text streams with applications to topic detection and tracking. In *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008.

A. Ansari, S. Essegaier, and R. Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37(3):363–375, 2000.

D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

Y. Baba, F. Ishikawa, and S. Honiden. Extraction of places related to Flickr tags. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pp. 523–528. IOS Press, 2010.

N. Barbieri, G. Manco, E. Ritacco, M. Carnuccio, and A. Bevacqua. Probabilistic topic models for sequence data. *Machine Learning Journal*, 93(1):5–29, 2013.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine Learning*, pp. 9. ACM, 2004.

PN. Bennett, RW. White, W. Chu, ST. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short-and long-term behavior on search personalization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 185–194. ACM, 2012.

B. Berjani and T. Strufe. A recommendation system for spots in location-based online social networks. In *Proceedings of the 4th Workshop on Social Network Systems*, pp. 4. ACM, 2011.

D. P. Bertsekas and S. Shreve. *Stochastic optimal control: the discrete-time case*. Athena Scientific, 2004.

A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan, and I. Matthews. Large-scale analysis of soccer matches using spatiotemporal tracking data. In *Proceedings of the 14th IEEE International Conference on Data Mining*, pp. 725–730. IEEE, 2014.

D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*, volume 98, pp. 46–54, 1998.

D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11: 1–94, 1999.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, 2000.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

U. Brefeld, J. Lasek, and S. Mair. Probabilistic movement models and zones of control. *Machine Learning*, pp. 1–21, 2018.

J. Brooks, M. Kerr, and J. Guttag. Developing a data-driven player ranking in soccer using predictive model weights. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 49–55. ACM, 2016.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5 (1):1–122, 2012.

B. Cao, L. He, X. Kong, P. S. Yu, Z. Hao, and A. B. Ragin. Tensor-based multi-view feature selection with applications to brain diseases. In *Proceedings of the 14th IEEE International Conference on Data Mining*, pp. 40–49, 2014.

B. Cao, X. Kong, and P. S. Yu. A review of heterogeneous data mining for brain disorder identification. *Brain Informatics*, 2(4):211–233, 2015.

L. Cao, J. Yu, J. Luo, and T. S. Huang. Enhancing semantic and geographic annotation of web images via logistic canonical correlation regression. In *Proceedings of the 17th ACM International Conference on Multimedia*, pp. 125–134. ACM, 2009.

S. Caron and S. Bhagat. Mixing bandits: a recipe for improved cold-start recommendations in a social network. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, pp. 11. ACM, 2013.

A. Celikyilmaz, D. Hakkani-Tür, and G. Tür. Leveraging web query logs to learn user intent via Bayesian discrete latent variable model. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

O. Chapelle and S. Keerthi. Efficient algorithms for ranking with SVMs. *Information retrieval*, 13(3):201–215, 2010.

M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pp. 693–703. Springer, 2002.

S. P. Chatzis. A coupled Indian buffet process model for collaborative filtering. In *Proceedings of the 4th Asian Conference on Machine Learning*, 2012.

C. Cheng, F. Xia, T. Zhang, I. King, and M. R. Lyu. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pp. 265–272. ACM, 2014.

W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, 1998:746–752, 1998.

J. L. Copete, J. Suzuki, Q. Wei, R. Iwaki, N. Endo, H. Mori, Y. Nagai, and M. Asada. Estimation of players actions in soccer matches based on deep autoencoder. In *Proceedings of the 42nd Workshop on SIG Challenge*, pp. 7–12, 2015.

P. Covington, J. Adams, and E. Sargin. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 191–198. ACM, 2016.

D. J. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the world's photos. In *Proceedings of the 18th International Conference on World Wide Web*, pp. 761–770. ACM, 2009.

A. Dallmann, A. Grimm, C. Pölitz, D. Zoller, and A. Hotho. Improving session recommendation with recurrent neural networks by exploiting dwell time. *arXiv preprint arXiv:1706.10231*, 2017.

A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 271–280. ACM, 2007.

A. A. Deshmukh, U. Dogan, and C. Scott. Multi-task learning for contextual bandits. In *Advances in Neural Information Processing Systems*, pp. 4848–4856, 2017.

R. Devooght and H. Bersini. Long and short-term recommendations with recurrent neural networks. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pp. 13–21. ACM, 2017.

K. Dullemond and K. Peeters. *Introduction to Tensor calculus*. University of Heidelberg, 2010.

S. T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230, 2004.

R. Dybowski and S. Roberts. Confidence intervals and prediction intervals for feed-forward neural networks. *Clinical Applications of Artificial Neural Networks*, pp. 298–326, 2001.

S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan. Personalized ranking metric embedding for next new POI recommendation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pp. 2069–2075, 2015.

J. Fernandez and L. Bornn. Wide open spaces: a statistical technique for measuring space creation in professional soccer. In *Proceedings of the 12th Annual MIT Sloan Sports Analytics Conference*, 2018.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, 2018.

J. Fürnkranz and E. Hüllermeier (eds.). *Preference Learning*. Springer-Verlag, 2010.

R. Gaonkar, M. Tavakol, and U. Brefeld. MDP-based itinerary recommendation using geo-tagged social media. In *International Symposium on Intelligent Data Analysis*, pp. 111–123. Springer, 2018.

R. Gaudel and M. Sebag. Feature selection as a one-player game. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 359–366. Omnipress, 2010.

M. Gemmis, L. Iaquinta, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Learning preference models in recommender systems. In *Preference Learning*, pp. 387–407. Springer, 2010.

G. Giannopoulos, U. Brefeld, T. Dalamagas, and T. Sellis. Learning to rank user intent. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2011.

K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

Google. The 2014 traveler's road to decision. Technical report, June 2014.

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

J. Haase and U. Brefeld. Mining positional data streams. In *International Workshop on New Frontiers in Mining Complex Patterns*, pp. 102–116. Springer, 2014.

P. Haider, L. Chiarandini, and U. Brefeld. Discriminative clustering for market segmentation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

A. Hassan, R. Jones, and K.L. Klinkner. Beyond DCG: user behavior as a predictor of a successful search. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, 2010.

M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. In *Proceedings of the International Conference on Learning Representations*, 2016.

J. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2008.

B. Hidasi and D. Tikk. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 67–82. Springer, 2012.

B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 241–248. ACM, 2016.

W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pp. 263–272. IEEE, 2008.

S. Jiang, X. Qian, J. Shen, Y. Fu, and T. Mei. Author topic model-based collaborative filtering for personalized POI recommendations. *IEEE Transactions on Multimedia*, 17(6):907–918, 2015.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 133–142. ACM, 2002.

T. Kamishima, H. Kazawa, and S. Akaho. A survey and empirical comparison of object ranking methods. In *Preference Learning*, pp. 181–201. Springer-Verlag, 2010.

A. Karatzoglou. Collaborative temporal order modeling. In *Proceedings of the 5th ACM Conference on Recommender Systems*, 2011.

A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pp. 79–86. ACM, 2010.

G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pp. 247–254. ACM, 2001.

S. S. Keerthi, K. B. Duan, S. K. Shevade, and A. N. Poo. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61(1-3):151–165, 2005.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, pp. 282–293. Springer-Verlag, 2006.

R. Koller, D.and Parr. Computing factored value functions for policies in structured MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, volume 99, pp. 1332–1339, 1999.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 8:30–37, 2009.

T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 579–588. ACM, 2010.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, J. Perolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4193–4206, 2017.

K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*. Citeseer, 1995.

J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, pp. 817–824, 2008.

H. M. Le, P. Carr, Y. Yue, and P. Lucey. Data-driven ghosting using deep imitation learning. *Proceedings of the 11th Annual MIT Sloan Sports Analytics Conference*, 2017a.

H. M. Le, Y. Yue, P.r Carr, and P. Lucey. Coordinated multi-agent imitation learning. In *International Conference on Machine Learning*, pp. 1995–2003, 2017b.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on the World Wide Web*, pp. 661–670. ACM, 2010.

E. Liebman, P. Khandelwal, M. Saar-Tsechansky, and P. Stone. Designing better playlists with Monte Carlo tree search. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 4715–4720, 2017.

K. H. Lim. Recommending and planning trip itineraries for individual travellers and groups of tourists. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling*, pp. 115, 2016.

K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera. Personalized tour recommendation based on user interests and points of interest visit durations. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pp. 1778–1784, 2015.

G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, (1):76–80, 2003.

D. Link, S. Lang, and P. Seidenschwarz. Real time quantification of dangerousity in football using spatiotemporal tracking data. *PloS one*, 11(12):e0168768, 2016.

M. L. Littman. Value-function reinforcement learning in Markov games. *Cognitive Systems Research*, 2(1):55–66, 2001.

Y. Liu, X. Huang, and A. An. Personalized recommendation with adaptive mixture of markov models. *Journal of the American Society for Information Science and Technology*, 58(12):1851–1870, 2007.

M. Loecher and T. Jebara. CitySense: multiscale space time clustering of GPS points and trajectories. In *Proceedings of the Joint Statistical Meeting*, 2009.

R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6382–6393, 2017.

P. Lucey, A. Bialkowski, M. Monfort, P. Carr, and I. Matthews. Quality vs quantity: improved shot prediction in soccer using strategic features from spatiotemporal data. In *Proceedings of the 8th Annual MIT Sloan Sports Analytics Conference*, pp. 1–9, 2014.

Z. Ma, G. Pant, and O. R. L. Sheng. Interest-based personalized search. *ACM Transactions on Information Systems*, 25(1), 2007.

D. K. Mahajan, R. Rastogi, C. Tiwari, and A. Mitra. LogUCB: an explore-exploit algorithm for comments recommendation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pp. 6–15. ACM, 2012.

S. Maneeroj and A. Takasu. Hybrid recommender system using latent features. In *International Conference on Advanced Information Networking and Applications Workshops*, pp. 661–666. IEEE, 2009.

I. G. McHale, P. A. Scarf, and D. E. Folker. On the development of a soccer player performance rating system for the English premier league. *Interfaces*, 42(4):339–351, 2012.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A Rusu, J. Veness, M. G. Bellemare, M. Graves, A.and Riedmiller, A. K. Fidjeland, and G. Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 637–646. ACM, 2009.

R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the 5th ACM Conference on Digital Libraries*, pp. 195–204. ACM, 2000.

C. I. Muntean, F. M. Nardini, F. Silvestri, and R. Baraglia. On learning prediction models for tourists paths. *ACM Transactions on Intelligent Systems and Technology*, 7(1):8, 2015.

A. Noulas, S. Scellato, N. Lathia, and C. Mascolo. Mining user mobility features for next place prediction in location-based services. In *Proceedings of the 12th IEEE International Conference on Data Mining*, pp. 1038–1043. IEEE, 2012.

K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware SVM for context-dependent information recommendation. In *Proceedings of the 7th International Conference on Mobile Data Management*, pp. 109. IEEE Computer Society, 2006.

S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 2681–2690, 2017.

S. Oyama and C. Manning. Using feature conjunctions across examples for learning pairwise classifiers. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 322–333. Springer, 2004.

R. Pan, Y. Zhou, B. Cao, N. N Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pp. 502–511. IEEE, 2008.

L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-agent Systems*, 11(3):387–434, 2005.

S. Perugini, M. Gonçalves, and E. Fox. Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems*, 23(2):107–143, 2004.

N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 239–247. ACM, 2013.

I. Pilászy, D. Zibriczky, and D. Tikk. Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pp. 71–78. ACM, 2010.

A. Popescu, G. Grefenstette, and PA. Moëllic. Gazetiki: automatic construction of a geographical gazetteer. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2008.

S. Purushotham, Y. Liu, and C. Kuo. Collaborative topic regression with social matrix factorization for recommendation systems. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 691–698, 2012.

M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

T. Quack, B. Leibe, and L. Van Gool. World-scale mining of objects and events from community photo collections. In *Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval*, pp. 47–56. ACM, 2008.

M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the 11th ACM Conference on Recommender Systems*, pp. 130–137. ACM, 2017.

D. Quercia, R. Schifanella, and L. Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media*, pp. 116–125. ACM, 2014.

F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 784–791. ACM, 2008.

V. Rakesh, N. Jadhav, A. Kotov, and C. Reddy. Probabilistic social sequential model for tour recommendation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pp. 631–640. ACM, 2017.

T. Rattenbury, N. Good, and M. Naaman. Towards automatic extraction of event and place semantics from Flickr tags. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 103–110. ACM, 2007.

S. Rendle. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*, pp. 995–1000. IEEE, 2010.

S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pp. 251–258. ACM, 2008.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pp. 452–461. AUAI Press, 2009.

S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 811–820. ACM, 2010.

S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 635–644. ACM, 2011.

P. Resnick and H. Varian. Recommender systems. *Communications of the ACM*, 40 (3):56–58, 1997.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pp. 175–186. ACM, 1994.

R. M. Rifkin and R. A. Lippert. Value regularization and fenchel duality. *Journal of Machine Learning Research*, 8:441–479, 2007.

E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 841–850. ACM, 2010.

J. Sang, T. Mei, J. Sun, C. Xu, and S. Li. Probabilistic sequential POIs recommendation via check-in data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pp. 402–405. ACM, 2012.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pp. 158–167. ACM, 2000a.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, 2000b.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on the World Wide Web*, pp. 285–295. ACM, 2001.

B. Schölkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, pp. 416–426. Springer, 2001.

G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.

U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, A. Strehl, and S. Vishwanathan. Hash kernels. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 496–503, Clearwater Beach, Florida, USA, 2009.

D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, and T. Graepel. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

A. Smalter, J. Huan, and G. Lushington. Feature selection in the tensor product feature space. In *Proceedings of the 9th IEEE International Conference on Data Mining*, pp. 1004–1009. IEEE, 2009.

E. Smirnova and F. Vasile. Contextual sequence modeling for recommendation with recurrent neural networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, pp. 2–9. ACM, 2017.

Y. Song, A. Elkahky, and X. He. Multi-rate deep learning for temporal recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 909–912. ACM, 2016.

W. Spearman. Beyond expected goals. In *Proceedings of the 12th Annual MIT Sloan Sports Analytics Conference*, 2018.

R. Sutton and A. Barto. *Reinforcement learning: an introduction*, volume 1. MIT press Cambridge, 1998.

C. Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

N. Taghipour, A. Kardan, and S. Ghidary. Usage-based web recommendations: a reinforcement learning approach. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, pp. 113–120. ACM, 2007.

G. Takács and D. Tikk. Alternating least squares for personalized ranking. In *Proceedings of the 6th ACM Conference on Recommender Systems*, pp. 83–90. ACM, 2012.

T. Taki and J. Hasegawa. Visualization of dominant region in team games and its application to teamwork analysis. In *Proceedings of the Computer Graphics International Conference*, pp. 227–235. IEEE, 2000.

M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning*, pp. 330–337, 1993.

L. Tang, Y. Jiang, L. Li, and T. Li. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pp. 73–80. ACM, 2014.

L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 323–332. ACM, 2015.

M. Tavakol and U. Brefeld. Factored MDPs for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pp. 33–40. ACM, 2014.

M. Tavakol and U. Brefeld. A unified contextual bandit framework for long-and short-term recommendations. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 269–284. Springer, 2017.

S. Ten Hagen, M. Van Someren, and V. Hollink. Exploration/exploitation in adaptive recommender systems. *Proceedings of Eunite*, 2003.

W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

B. Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 273–276. ACM, 2016.

M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pp. 654–663. AUAI Press, 2013.

S. Van Canneyt, S. Schockaert, O. Van Laere, and B. Dhoedt. Time-dependent recommendation of tourist attractions using Flickr. In *Proceedings of the 23rd Benelux Conference on Artificial Intelligence*, pp. 255–262, 2011.

H. P Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pp. 225–232. ACM, 2014.

S. Vembu and T. Gärtner. Label ranking algorithms: a survey. In *Preference Learning*, pp. 45–64. Springer, 2010.

C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 448–456. ACM, 2011.

J. Wang and . Zhang, Y. Opportunity model for e-commerce recommendation: right product; right time. In *Proceedings of the 36th international ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 303–312. ACM, 2013.

X. Wang and A. McCallum. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

Y. Wang, H. Tung, A. Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pp. 991–999, 2015.

C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 1113–1120. ACM, 2009.

R. White, P. Bennett, and S. Dumais. predicting short-term interests using activity-based search context. In *Proceedings of the 19th ACM International on Conference on Information and Knowledge Management*. ACM, 2010.

J. Wolf, R. Guensler, and W. Bachman. Elimination of the travel diary: experiment to derive trip purpose from global positioning system travel data. *Transportation Research Record: Journal of the Transportation Research Board*, pp. 125–134, 2001.

C. Wu, A. Ahmed, A. Beutel, A. Smola, and H. Jing. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pp. 495–503. ACM, 2017.

Q. Wu, H. Wang, Q. Gu, and H. Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 529–538. ACM, 2016a.

S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu. Personal recommendation using deep recurrent neural networks in NetEase. In *IEEE 32nd International Conference on Data Engineering*, pp. 1218–1229. IEEE, 2016b.

B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, , and H. Li. Context-aware ranking in web search. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 451–458, 2010.

H. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 3203–3209, 2017.

X. Yi, L. Hong, E. Zhong, N. Nan Liu, and S. Rajan. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 113–120. ACM, 2014.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, pp. 1083–1106, 2003.

E. Zhan, S. Zheng, Y. Yue, and P. Lucey. Generative multi-agent behavioral cloning. *arXiv preprint arXiv:1803.07612*, 2018.

C. Zhang, H. Liang, K. Wang, and J. Sun. Personalized trip recommendation with POI availability and uncertain traveling time. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 911–920. ACM, 2015.

S. Zhang, L. Yao, and A. Sun. Deep learning based recommender system: a survey and new perspectives. *ACM Journal on Computing and Cultural Heritage*, 2017.

W. Zhang and J. Wang. Location and time aware social collaborative retrieval for new successive point-of-interest recommendation. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1221–1230. ACM, 2015.

S. Zheng, Y. Yue, and J. Hobbs. Generating long-term trajectories using deep hierarchical networks. In *Advances in Neural Information Processing Systems*, pp. 1543–1551. Curran Associates, Inc., 2016.

L. Zhou and E. Brunskill. Latent contextual bandits and their application to personalized recommendations for new users. In *Proceedings of the 35th International Joint Conference on Artificial Intelligence*, pp. 3646–3653. AAAI Press, 2016.

Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai. What to do next: modeling user behaviors by time-LSTM. In *Proceedings of the 26h International Joint Conference on Artificial Intelligence*, pp. 3602–3608, 2017.

A. Zimdars, D. Chickering, and C. Meek. Using temporal data for making recommendations. In *Proceedings of the 17TH Conference on Uncertainty in Artificial Intelligence*, pp. 580–588. Morgan Kaufmann Publishers Inc., 2001.

# Abbreviations

**AUC** Area Under the Curve. 89, 90, 106, 107

**CF** Collaborative Filtering. 8, 9, 42

**fMDP** factored Markov Decision Process. 24, 27, 28, 30–33, 41–44

**LDA** Latent Dirichlet Allocation. 11, 25, 37, 38

**MCTS** Monte-Carlo Tree Search. 4, 6, 78, 79, 81, 85–88, 91–94

**MDP** Markov Decision Process. 3–6, 12, 15, 17–20, 24, 26–30, 32, 33, 42, 44, 45, 47, 49, 50, 54, 55, 57, 58, 96, 98, 99, 110, 112, 113

**POI** Point of Interest. 4, 47–56, 58

**RL** Reinforcement Learning. 3–7, 16, 17, 21, 22, 33, 96, 97, 113

**SVM** Support Vector Machines. 4, 11, 48, 80, 88

**TD** Temporal Difference. 20, 22, 106

**UCB** Upper Confidence Bound. 21, 63, 69