Toward Data Efficient Online Sequential Learning

Kaige Yang

[redacted]

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Electronic and Electrical Engineering

University College London

June 2022

I, Kaige Yang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Acknowledgements

Four years ago, I was very fortunate to have the opportunity to join the fantastic research team LASP led by Dr Laura Toni at University College London. I would like to take this opportunity to thank many people without whom my PhD journey would not have been so enjoyable and memorable.

First and foremost, I am grateful for the supervision and support of my principal supervisor Dr Laura Toni. She is an incredible mentor, always ready to provide support and valuable advice whenever I need help throughout my PhD journey. Without her help, it would not have been possible for me to complete this thesis. I really enjoy the great freedom she offered me in exploring new ideas, while always guiding me with insightful suggestions. Moreover, I am extremely grateful for the internship opportunities she provided me, which not only involved me in many interesting academic communities but also underpinned the foundations for my future career. I feel lucky to have had the opportunity to work with her and very much look forward to our future collaborations.

I also want to thank other advisors and mentors during my study. Prof. Christos Masouros was my first mentor when I was a bachelor's student, with whom I conducted my first academic research project. This experience gave me a first taste of what academic research is. He set a great role model for me to follow. I want to express thanks to Dr XiaoWen Dong, who was informally my supervisor and collaborator. I appreciate every discussion and brainstorming with him. His knowledge of graph learning is unparalleled. His questions, insights and perspectives were incredibly valuable. I have always looked up to him as a role model.

I would like to thank Dr ZhenJie Zhang for the internship opportunity at Ad-

vanced Digital Science Centre (ADSC) in Singapore, which was my first working experience. I also would like to thank Prof. Marta Mrak for the internship opportunity at BBC. It was an interesting experience working with her and other colleagues in the BBC R&D London team.

Thank you, more than I could express in words, to my parents and siblings for their unlimited love and support. Without their support, I cannot imagine that my 10-year overseas study experience would be possible. I am looking forward to starting new chapters of my life together with them as always but within a more physically reachable distance. Finally, standing at the end of my PhD study, I deeply believe this is not an end but a beginning and cannot wait to see what is next.

Abstract

"Can machines optimally take sequential decisions over time?". Since decades, researchers have been seeking an answer to this question, with the ultimate goal of unlocking the potential of artificial general intelligence (AGI) for a better and sustainable society. Many are the sectors that would be boosted by machines being able to take efficient sequential decisions over time. Let think at real-world applications such as personalized systems in entertainment (content systems) but also in healthcare (personalized therapy), smart cities (traffic control, flooding prevention), robots (control and planning), etc.. However, letting machines taking proper decisions in real-life is a highly challenging task. This is caused by the uncertainty behind such decisions (uncertainty on the actual reward, on the context, on the environment, etc.). A viable solution is to learn by experience (i.e., by trial and error), letting the machines uncover the uncertainty while taking decisions, and refining its strategy accordingly. However, such refinement is usually highly data-hungry (data-inefficiency), requiring a large amount of application specified data, leading to very slow learning processes – hence very slow convergence to optimal strategies (curse of dimensionality). Luckily, data is usually intrinsically structured. Identifying and exploiting such structure substantially improves the data-efficiency of sequential learning algorithms. This is the key hypothesis underpinning the research in this thesis, in which novel structural learning methodologies are proposed for decision-making strategies problems such as Recommendation System (RS), Multi-armed Bandit (MAB) and Reinforcement Learning (RL), with the ultimate goal of making the learning process more (data)-efficient. Specifically, we tackle such goal from the perspective of modelling the problem structure as graphs, embedding tools from

graph signal processing into decision learning theory.

As the first step, we study the application of graph-clustering techniques for RS, in which the curse of dimensionality is addressed by grouping data into clusters via graph-clustering techniques. Next, we exploit spectral graph structure for MAB problems, representing online learning problems. A key challenge is to learn sequentially the unknown bandit vector. Exploiting the smoothness-prior (i.e., bandit vector smooth on a given underpinning graph), we study theoretically the Laplacian-regularized estimator and provide both empirical evidences and theoretical analysis on the benefits of exploiting the graph structure in MABs. Then, we focus on the theoretical understanding of the Laplacian-regularized estimator. To this end, we derive a theoretical error upper bound on the estimator, which illustrates the impact of the alignment between the data and the graph structure as well as the graph spectrum on the estimation accuracy.

We then move to RL problems, focusing on the specific problem of learning a proper representation of the state-action (representation learning problem). Motivated by the fact that a good representation should be informative of the value function, we seek a learning algorithm able to preserve continuity between the value function and the representation space. Showing that state values are intrinsically correlated to the state transition dynamic structure and the diffusion of the reward on the MDP graph, we build a new loss function based on the newly defined diffusion distance and we propose a novel method to learn state representation with such desirable property.

In summary, in this thesis we address both theoretically and empirically important online sequential learning problems leveraging on the intrinsic data structure, showing the gain of the proposed solutions toward more data-efficient sequential learning strategies.

Impact Statement

Artificial Intelligence (AI) research dedicates to understanding and developing human intelligence which creates and shapes the fantastic world we live in. It is unarguable that AI would benefit humanity and the world at large. Reinforcement Learning (RL), an important branch of AI research, is about understanding and solving sequential decision making problems. Sequential decision making is ubiquitous in almost every domain in the real-world such as making a coffee, driving a car, traveling round the world, planning and accomplishing a scientific experiment, writing an academic paper, etc. Human's ability in sequential decision making is remarkable. RL dedicates to understanding and demystifying such ability.

Recently, RL has emerged as a powerful solution to many complex sequential decision making problems including video games, simulated robotic control and data centre energy allocation, etc. Despite numerous demonstrated successes, the wide application of RL to real-world problems is still limited. One of the main reasons is the low data efficiency of modern RL. Modern RL algorithms require a large amount of data to achieve a reasonable level of performance, which is usually in many orders of magnitude than human need. Unfortunately, in many real-world problems, collecting a large amount of data is expensive or even prohibited.

This thesis is an effort towards more data-efficient RL solutions. We improve the data efficiency from two important perspectives: exploration and representation learning. As a results, our contributions lead to more data-efficient RL algorithms in comparison with existing algorithms in the literature. We believe that the research contained in the thesis are applicable to many real-world problems. At large, this research contributes to improving the applicability of RL to data-limited domains.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sequential decision-making plays a pivotal role in our daily life. As humans, we constantly face a variety of tasks involving sequential decision-making strategies. Planning and adjusting the commute route from home to the working place according to the traffic conditions. Shopping in grocery based on products' availability and family need. Changing trajectory and speed while walking or driving based on others' behaviour. In these examples, each of us need to make a series of interrelated decisions, while interacting with the real-world, to then achieve a desired (possibly long-term) goal. Being able to solve such complex tasks is one of the hallmarks of human intelligence.

However, sequential decision-making is also ubiquitous in real world problems that should be solved by machines instead of humans. Smart cities with efficient traffic systems proactively managed to avoid congestion. Personalized healthcare with medical treatment procedures fine tuned to address each patient needs. Data centre cooling system adapting to the dynamic workload and outdoor temperatures. Being able to automatize such decisions would lead to significant societal and economical impact. Given the ability of humans as decision maker, demystifying and understanding such ability is one of the foundational driving force underlying the research of artificial intelligence.

Reinforcement Learning (RL) along with supervised learning and unsupervised learning form the main paradigms in machine learning research, as shown in Fig 1.1. Supervised learning learns under supervision trying to infer the relationship between

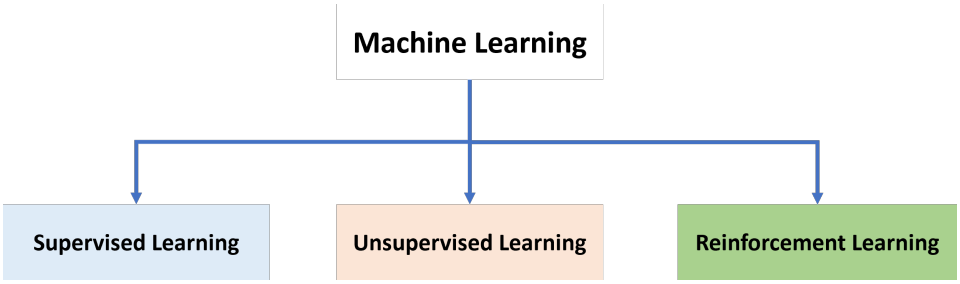


Figure 1.1: Machine Learning Paradigms

inputs and outputs, and apply the inferred relationship to unseen but similar inputs via generalization and extrapolation. Typical applications include prediction and classification. Unsupervised learning (i.e., learning without supervision) aims to mine the hidden structure of data with applications in clustering and dimension reduction. While both supervised learning and unsupervised learning paradigms can be useful for learning decision making strategies (could be useful for representation learning tasks in RL, for example), they can not solve the sequential decision problems alone. Hence, the need for a new learning paradigm, called Reinforcement Learning (RL). RL paradigm focuses on sequential decision making aiming to learn without supervision the mapping from situations (states) to optimal actions in order to maximized a long-term goal.

Specifically, within the RL framework, a decision-making agent interacts sequentially with an environment and aims to achieve a long-term goal (defined by the task). At each interaction, the agent takes an action upon its sensing about the current state of the environment and then receives a feedback signal from the environment. The environment transits to next state according to its dynamic and the action of agent. The agent seeks a sequence of actions, called policy, by trails and error to solve the task. RL concerns the learning of an optimal policy from interactions with the environment. The learning methodology of RL mimics the foundational way which is used by human to learn. Throughout our live, we constantly interact with the environment around us and adapt our actions based on our sensing to fulfil goals like grasping a cup of tea, making a meal and riding a bicycle, etc.

RL has a history of decades [1]. Recently, with the combination of deep learning [2] and powerful computational capability, (Deep) RL has gone through a tremendous

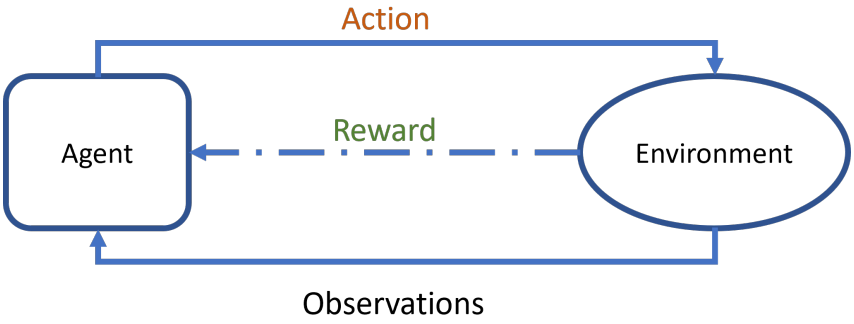


Figure 1.2: Depiction of Reinforcement Learning Framework

progress making remarkable successes in a wide range of complex applications. For example, Playing video games at the level of human players [3]. Outperforming the world champion in broad game Go [4]. Training walking robotics to learn locomotion skills [5]. Recommending videos to user from million of candidate videos [6].

Despite impressive successes, there are still many challenges preventing the wide use of RL in real-life applications [7]. A profound challenge is the low data efficiency, which refers to the amount of data required to achieve a certain level of performance. Modern deep RL algorithms require a large amount of interactions with the environment to achieve a reasonable level of performance, which is usually in many orders of magnitude than human need [8]. Unfortunately, collecting interaction data in many real-life tasks are costly or even prohibited. For example, in healthcare, the data collection, involving the interaction between medical treatments and patients, could be expensive and dangerous. In robotics, data collection, requiring intensive experiments, could be time consuming and with high cost in terms of labor and finance. Therefore, it is vital to develop data efficient algorithms able to achieve reasonable performance while drastically reducing the amount of required data.

In this thesis, we devote our investigation on data efficiency in RL and make several contributions allowing RL agents to learn faster (with less data) while minimizing the suboptimality gap of the learned policy.

1.1 Main Challenges

The remarkable successes achieved by RL algorithms has draw an exponentially increasing attentions in applying RL algorithms into real-life problem. However, many challenges remained unsolved preventing RL algorithms to be actually adopted in such real-life problems in a data-efficient and reliable manner. In the following, we highlight the key challenges underpinning the data inefficiency problem in RL.

Exploration-Exploitation Dilemma

The balance between exploration and exploitation is a foundation issue for solving RL problems. The agent is expected to find the optimal policy as soon as possible, *exploiting* the acquired knowledge. However, committing to the policies too early without sufficient *exploration* might lead to local/sub-optimal policies. In short, too little exploration provides insufficient information for policy improvement, while too much exploration delays the finding of the optimal policy. This issue is particularly serious in tasks with large action and state space, such as continuous control in robotic and recommendation system with millions items. Finding the right balance between exploration and exploitation critically determines the data efficiency of RL algorithms. i.e., how quick the optimal policy can be found.

Representation Learning

In RL, the agent first senses the environment based on observations and then acts according to it. In many real-world scenarios, the observations provided by the environment might contain redundant and distracting information which is irrelevant to the problem the agent aims to solve. For instance, in self-driving car applications, the shape of building and cloud in the background are irrelevant to the driving task. This implies that the agent needs to distil useful information from redundant/rich observations. This is known as representation learning and in RL is particularly challenging. Specifically, in the literature, there is a lack of theoretical understanding and a consensus on the notion of optimal representations for RL. A representation learned to reduce the value-function approximation error, might not lead to efficiently learn the optimal agent policy. Similarly, it is not clear which conditions an optimal

representation should respect to ensure for the agent to quickly learn the optimal policy. Because of all those problems, current representation learning strategies could lead to low data efficiency.

Generalization

Most RL research has been focused on solving a specific task where the environment is given, a prior defined and stationary. This leads the machine to overfit to such given environment, developing an impressive ability in taking decision but in a very narrow task. As a consequence, every time there is 1) a slight change of environment/task; 2) a noisy or out of distribution, the machine miserably fails in taking proper decisions. This is a very important limitation of current RL agent as in many practical tasks, the environment is dynamic and constantly changing. For instance, in financial portfolio management, the financial market (environment) is versatile and non-stationary. If those scenarios are treated as new ones every time, a new learning process is experienced, instead of relying on previously acquired knowledge. Hence, the well populated new line of research on generalization, robustness and open-ended learning in RL, to make the agent with a more general intelligence. The goal is for the RL agents to able to quickly adapt to new unseen situations and be robust with respect to variations of the environment. It is clear that achieving this goal would drastically impact the learning efficiency of the agents.

With the above challenges in mind and noticing the common denominator of data-inefficiency (as current limitation of RL), in this thesis, we made several contributions toward data-efficient RL with a particular focus on **exploration-exploitation dilemma** and **representation learning**.

1.2 Contributions

Through our investigations on exploration-exploitation dilemma and representation learning, this thesis makes a step toward more data-efficient RL agents.

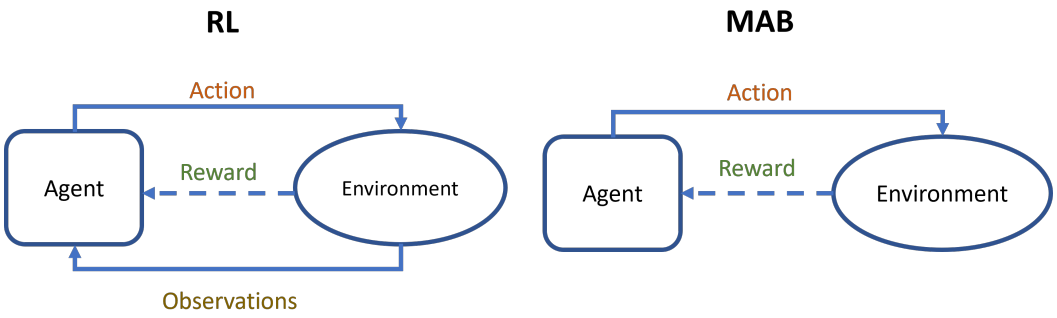


Figure 1.3: Depiction of Reinforcement Learning (RL) and Multi-armed Bandit (MAB)

Exploration-Exploitation Dilemma

Our efforts devoted to the exploration-exploitation dilemma fall under the framework of Multi-armed Bandit (MAB). MAB can be viewed as a special RL problem, as shown in Figure 1.3. In the RL framework, at decision opportunity, the agent observes the state, and selects an action among the feasible ones from the current state. Then, the agent receives a reward upon the selected action and the environment transits to next state, according to both the environment dynamics and the selected action. With the new state, the agent needs to select an action again and the process continues. In RL there is a clear dependency on the actions timeline since actions have an impact on the future evolution of the environment. In contrast, in MAB problems, the environment state does not change after an action is selected by the agent, or if this changes (as in the case of contextual bandit), this variation is action-independent and it is usually randomly or adversarial selected. Therefore, MAB problems disentangle the exploration-exploitation dilemma from more complex aspects of RL such as off-policy learning (collecting data on a behavioural policy different from the learned policy) and temporal evolution/transitions of states in the environment. The MAB problems have a long history [9] and been studied intensively across many disciplines such as statistics, operation research and mathematics [10]. During recent years, MAB has experienced a dramatic increase of interest due to its fundamental importance in RL and its own wide range of applications such as recommendation system [11], experimental design [12], auctions [13] and radio channel allocation [14]. Within the MAB framework, this thesis makes three main

contributions in addressing the MAB problem.

1. **Leveraging structural knowledge for multi-task MABs.** We consider multi-task linear bandit problems. Solving each bandit task independently would result in a total regret scaling linearly with the number of tasks. This makes the multi-task bandit problem unfeasible in a large-scale setting (large number of tasks). Specifically, Suppose the number of linear bandit task to solve is n, and bandit tasks are sampled uniformly across the time horizon T. Each bandit will appear roughly $T' = T/n$ times. If the agent solves each bandit independently (suppose `LinUCB` is used), the cumulative regret of each bandit can be upper bounded as $R(T') = \mathcal{O}(d\sqrt{T'})$. The total regret will be $R(T) = \mathcal{O}(nd\sqrt{T'})$, which scales linearly with the number of bandit n. Under the assumption that those tasks share similarities, we propose an algorithm that exploits prior knowledge regarding such relational structure among tasks. We model the task relatedness as graph structure and we demonstrate, empirically and theoretically, that such prior structural knowledge leads to better cumulative regret and we also highlight the dependency of the regret on the task relatedness.

2. **Theoretically understanding the impact of structure knowledge in learning frameworks.** We provide a theoretical analysis of the Laplacian-regularized estimator, well known estimator with a close form solution but with a lack of theoretical understanding. We fill-in this gap by providing an error upper bound that illustrates the impact of the alignment between the data and the graph structure as well as the graph spectrum on the estimation accuracy. We then show how this theoretical result plays a key role in the multi-task problem. However, we believe this contribution is of strong interest beyond MABs, as such Laplacian-regularized estimators have been applied to many learning problems and its statistical consistency properties have been largely overlooked in the literature.

3. **Balancing the exploration and exploitation in a data-driven fashion**: In some existing works, the uncertainty around the reward of each arm is char-

acterized by *upper confidence bound* [15], which is typically derived from concentration inequalities based on assumptions on the reward and noise distribution. The validity of these assumptions however is unknown in practice. We propose a differentiable linear bandit algorithm, which learns the confidence interval in a data-driven fashion, making it adaptive to the actual problem structure.

4. **Graph-based recommendation system**: As an application of MAB, we consider the problem of recommendation system (RS) in a large settings, i.e., when recommending to a large number of users. Each user is modelled as a bandit problem and items represent arms. Again, the users' similarity can be captured by an underpinning graph. To improve the efficiency of the recommender agent, we group users via spectral clustering technique, where RS serves user groups rather than individual users. As additional contribution, we built a real-users dataset from our collaboration with BBC and modelled user viewing behaviours on BBC programs as random walk process over an item (program) graph. The goal is to predict what program each user will visit (random walk) given past behaviours.

Representation Learning in RL

The second part of the thesis focuses on representation learning in RL. In RL tasks with large or continuous state space, one usually resorts to function approximation where state values are modelled as a function in terms of state *representations*. Such representations can be hand-crafted [16] [1] or learned through representation learning techniques. The data efficiency of RL algorithms heavily relies on its ability of learning compact representation, which serves to approximate value functions and enact new policies. In the literature, there is a lack of consensus on the concept of what is a good representation for RL. This thesis contributes toward improving the representation learning in RL from the perspective of value function approximation.

1. **RL agent behavioural similarity as diffusion model.** Intuitively, value functions are inherently induced by the underlying environment dynamics and

reward function. Such transitions, dynamics and reward values are usually driven by an underpinning structural model, which captures the behaviour of the RL agent. Leveraging such structure could help in inferring the value functions (hence better learning). Following this line of thought, we first investigate the intrinsic structure of value functions and identify a clear link between agent behaviour and diffusion model on graph. Then we identify a state distance metric upon which value functions are provable Lipschitz continuous. Such metric is key in behavioural representation learning in RL, which has been considered among the most promising strategies for data efficient RL.

2. **Developing of novel diffusion-based representation learning strategies for data-efficient RL agents.** To facilitate the representation learning, we first present a sample-based approach to learn the diffusion-distance from transition data. Then we propose a novel auxiliary loss which utilizes the learned distance to shape the state representation in such a way that value functions are Lipschitz continuous with respect to the learned representations. We hypothesise that the auxiliary loss can improve the learning efficiency, due to the metric-respect representation which facilities the generalization ability of value function approximation across states with similar representations.

1.3 Thesis Outline

This thesis consists of 8 chapters divided in parts. Subsequent to this introductory chapter, the rest of this document is organised as follows:

- **Chapter 2** provides the relevant background knowledge, problem formulations and existing methods related to Multi-armed Bandit and Reinforcement Learning. This chapter lays the foundation upon which this thesis is established.

- **Chapter 3** introduces our algorithm on multi-task linear bandit problem which exploits a prior knowledge on the relational structure among tasks through *graph Laplacian regularizer*. It also presents a finite-time analysis on the

proposed algorithm showing a significant improvement in terms of regret in comparison with existing algorithms, and demonstrate empirical advantages on both synthetic and real-world data.

- **Chapter 4** presents the theoretical analysis on Laplacian-regularized estimator, which illustrates the impact of the alignment between the data and the graph structure as well as the graph spectrum on the estimation accuracy. As expected, our analysis suggests that the smoother the data with respect to the graph and the denser the graph connected, the lower the estimation error.

- **Chapter 5** presents a differentiable linear bandit algorithm that utilizes *upper confidence bound* to balance exploration and exploitation trade-off. Such confidence bound is learned in a data-driven fashion by introducing a gradient estimator. This chapter also contains theoretical analysis and empirical demonstrations on the performance of the proposed algorithm.

- **Chapter 6** introduces a new approach for representation learning in RL based on the intrinsic structure of value functions. It also provides theoretical guarantee on value function approximation and demonstrates empirically that the proposed approach gives arise to informative and structural representations, and improves the control performance in comparison with baselines.

- **Chapter 7** presents an empirical work where we utilize graph-based clustering technique (spectral clustering) to solve the multi-task linear bandit problem with an application in recommender system (RS). Specifically, each user represents a bandit problem. Users on social network are grouped together through spectral clustering, and then the RS serves user groups rather than individual users.

- **Chapter 8** summarises the work presented in this thesis and highlights the importance of the proposed methods and tools. In addition, the chapter outlines future research directions.

- **Appendices** include proofs of theoretical results provided in the above chapters as well as supplementary experimental results.

Due to IP-copyright, this thesis cannot include the main contribution on real-users recommend systems. However, the goal is to publicly release both the dataset built in collaboration with BBC as well as designed algorithms.

1.4 Publications

The research presented in this document has resulted the following publications, listed in chronological order:

- *Proceedings*

  1. **K. Yang** and L. Toni, “Graph-based recommendation systems”, IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2018
  2. **K. Yang**, X. Dong, L. Toni, “Laplacian-regularized graph bandits: Algorithms and theoretical analysis", International Conference on Artificial Intelligence and Statistics (AISTATS), 2020.

- *In Submission*

  1. **K. Yang**, and L. Toni, “Differentiable linear bandit algorithm”, arXiv preprint arXiv:2006.03000, 2020.
  2. **K. Yang**, X. Dong, L. Toni, “Laplacian-Regularized Estimator: Analysis and Application on Contextual Bandits”, in submission IEEE Transaction on Signal and Information Processing over Networks (arXiv preprint arXiv:1902.03720). 2022.
  3. **K. Yang**, S. Madjiheurem, L. Toni, “Learn Diffusion-Induced State Representation”, in submission to ICLR 2023 (arXiv preprint arXiv:2101.02230).

- *Open-source dataste and graph-based RS algorithms*

1. **K. Yang**, L. Toni, D. Walker, M. Mrak, "A Time-dependent real-users dataset for iPLayer recommendation", gitHub (to be released soon jointly with a challenge publication, now IP-protected on BBC side).

- *PhD Scholarship*

1. UCL Overseas Research Scholarship (UCL-ORS).

- *Internships*

1. July-Nov, 2021, machine learning applied scientist at **Amazon Web Services** (China).
2. Jan-June 2021, research intern on recommendation systems at **BBC R&D London** (UK).
3. Sep-Nov 2019, visiting researcher on theoretical analysis of bandit problems at SequeL team at **Inria** (Lille, France).

Chapter 2

Background

This chapter introduces the relevant background knowledge related to sequential decision making strategies. To ensure a good understanding of the body of this thesis, we first review the background knowledge of multi-armed bandit problems (Section 2.1). Next, we provide the fundamental concepts and algorithms in Reinforcement Learning. Finally, we introduce the representation learning in Reinforcement Learning (Section 2.2).

2.1 Multi-armed Bandit (MAB)

MAB provides a framework to formalize the problem of sequential decision making under uncertainty. Within the MAB framework, an agent repeatedly interacts with an environment over a time horizon T, with T being a positive integer number. The environment consists of a set of alternatives choices made available to the agent, often referred as K-arms of the bandit. Each arm leads to a reward, *unknown* to the agent, who needs to learn, by trial and error, which is the best arm to select (i.e., the ones with the higher reward). At each time step $t \in [T]$, the agent is required to select one arm and receives a bandit feedback. Over the time horizon T, the agent generates a sequence of arm selections, called policy, and corresponding sequence of rewards. The goal of the agent is to find the policy such that maximizes the cumulative reward.

When making the arm selection, at each time step, the agent encounters an trade-off between exploration and exploitation: selecting the highest-rewarded arm based on current information (exploitation) or selecting unknown arms to gain more

information (exploration). Selecting unknown arms (exploration) provides more information at the possible expense of immediate reward. Selecting the current highest-rewarded arm might miss the chance of finding arms with higher rewards. Finding the right balance between exploration and exploitation is the key of solving MAB problems.

MAB problems were first introduced, in almost a century ago, for medical experiment trails [9] and formally investigated in [17] and [18]. In recent decades, due to its essential role in Reinforcement Learning and its own wide range of applications such as ads recommendation [11], auction [13] and resource allocation [14], MAB problems have experienced a surge of attention from various disciplines resulting an enormous body of work in the literature. Many books exist on various branches of MAB problems, such as [19, 20, 21, 10]. Interested readers are referred to such books for deep studies.

Based on assumptions about time horizon, the number of arm, reward distributions and feedback mechanism, MAB problems can be divided into many categories such as stochastic bandit, adversary bandit, partial-feedback, bandit-feedback, finite arm, infinite arm, known/unknown horizon, etc [10]. This thesis focuses on the stochastic bandit problem with finite arms, the basic form for MAB problem, which is formally defined in the next section.

2.1.1 Stochastic Bandit

In the stochastic bandit problem, an agent interacts with an environment repeatedly over a time horizon T. The environment consists of a finite set of arms $\mathscr{A}$ where $|\mathscr{A}| = K$ denotes the number of arms. Each arm $a \in \mathscr{A}$ is associated with an unknown but fixed reward distribution $\mathscr{P}(a)$. At each time step $t \in T$, the agent selects one arm $a_t \in \mathscr{A}$ from the arm set and receives a reward y_t which is sampled independently from the corresponding reward distribution $y_t \sim \mathscr{P}(a_t)$. The agent aims to maximize the cumulative reward collected over the time horizon, defined as $Y_T = \sum_{t=1}^{T} y_t$.

The arm selection strategy used by the agent is called policy, denoted as π. A commonly adopted measure of the policy performance is the regret, which is defined

as the gap between the cumulative reward of the policy π and that of a benchmark policy. One type of benchmark policy is the optimal policy, denoted as π^*, which always selects the arm a^* with the highest expected reward y^* throughout the time horizon. More formally, the regret of policy π is defined as

$$R_T(\pi) = T \cdot y^* - \mathbb{E}\left[\sum_{t=1}^{T} y_t\right] \tag{2.1}$$

The expectation is taken over the randomness of reward realization and the arm selection process. Intuitively, the regret measures how much the agent loses following policy π instead of always selecting the optimal arm. Minimizing the regret is equivalent to maximizing the cumulative reward.

When analysing the policy performance, we mainly concern about the dependence of regret $R_T(\pi)$ over the time horizon T. A policy π is said to be no-regret when the regret is sublinear with respect to the time horizon. Formally,

$$\lim_{T\to\infty} \frac{R_T(\pi)}{T} \to 0 \tag{2.2}$$

The sublinear regret implies that the agent concentrates on the optimal arm as the time horizon approaches to infinity (no more regret incurs). In [18] and [19], authors showed the optimal scalability of regret R_T over the time horizon T is $\Omega(\log T)$. Several algorithms [22] [23] [24] were proposed satisfying the optimal regret lower bound. The subtle difference in terms of algorithm design and regret analysis can be found in respect papers and books [20] [10]. In the following, we describe two foundational algorithms for stochastic bandit problem: `UCB1` and `Thompson Sampling`.

UCB1

As discussed before, the core of MAB problems is to balance exploration and exploitation trade-off. To deal with this trade-off, [25] proposed the `UCB1` algorithm, based on a principle called *Optimism in Face of Uncertainty*. The key intuition is that the agent selects an arm that either has high reward or large uncertainty. Specifically,

due to the uncertainty of the environment, i.e., noise, the expected reward of each arm can not be estimated accurately, especially at the beginning of the decision process. Therefore, at each time step t, based on information obtained so far, the reward of each arm can be estimated by its empirical average $\hat{y}_t(a) = \sum_{a_t=a} y_t(a)/n_t(a)$ where $n_t(a)$ is the number of times in which arm a has been chosen up to time t. Beyond the estimated reward, there is a need to quantify the uncertainty on the estimation, i.e, a confidence interval, which can be constructed using the Hoeffding inequality.

$$\mathbb{P}\{|\hat{y}_t(a) - y(a)| \leq \beta_t(a)\} \geq 1 - \frac{2}{T^2} \text{ where } \beta_t = \sqrt{\frac{2\log(T)}{n_t(a)}} \tag{2.3}$$

This results in a confidence interval of the reward of each arm at round t,

$$\hat{y}_t(a) - \sqrt{\frac{2\log(T)}{n_t(a)}} \leq y(a) \leq \hat{y}_t(a) + \sqrt{\frac{2\log(T)}{n_t(a)}} \tag{2.4}$$

Thus, the upper confidence bound is defined as

$$UCB_t(a) = \hat{y}_t(a) + \sqrt{\frac{2\log(T)}{n_t(a)}} \tag{2.5}$$

At each time step t, the agent chooses the arm with the maximum upper confidence bound, i.e., $a_t = \max_{a \in \mathscr{A}} UCB_t(a)$. This naturally favors the exploration and exploitation trade-off. In fact, $UCB_t(a)$ assumes large value in two settings:

- The first is a large $\hat{y}_t(a)$ which means a appears to be a high rewarded arm based on current information.
- The second is a large $\sqrt{2\log(T)/n_t(a)}$ which means a large uncertainty of the estimated reward due to a small $n_t(a)$.

Both are good reasons to choose arm a_t leading to a trade-off between exploration $\sqrt{2\log(T)/n_t(a)}$ and exploitation $\hat{y}_t(a)$. Striking the trade-off leads to good regret.

Theorem 1. *[25] Suppose K arms and T rounds, UCB1 achieves regret*

$$R_T \leq \mathscr{O}(\sqrt{KT\log(T)}) \tag{2.6}$$

Algorithm 1: UCB1

Input : $\mathscr{A}$: Arm set, T: Time horizon

1. Choose each arm once.
2. **for** $t \in [1,T]$ **do**
   Choose arm
   $a_t = \arg\max_{a\in\mathscr{A}} UCB_t(a) = \arg\max_{a\in\mathscr{A}} \left(\hat{y}_t(a) + \sqrt{\frac{2\log(T)}{n_t(a)}}\right)$
   end
3. Receive y_t, update $\hat{y}_t(a)$ and $n_t(a)$.

Thompson Sampling

An alternative to *Optimism in Face of Uncertainty* is *Probabilistic Randomization*. Thompson Sampling [26] models the reward of each arm via a posterior distribution and takes samples from this distribution. The arm with largest sampled reward is selected. The underlying idea is that if the posterior distribution is less concentrated, samples with high fluctuation lead to a high level of exploration (a low level of exploitation). On the other hand, a highly concentrated distribution leads to a low level of exploration (a high level of exploitation).

More specifically, Thompson Sampling initializes the reward distribution per arm to be $\mathbb{P}_0^a$, it updates this distribution over time, denoted by $\mathbb{P}_t^a$ based on all the rewards observed up to t. At each round t, a sample is drawn from the posterior distribution $\hat{y}_t(a) \sim \mathbb{P}_t^a$ of each arm a, the arm with the largest sampled reward is selected. Formally,

$$a_t = \arg\max_{a\in\mathscr{A}} \hat{y}_t(a) \tag{2.7}$$

Theorem 2. *[26] Suppose K arms with T rounds,* Thompson Sampling *achieves regret*

$$R_T \leq \mathscr{O}(\sqrt{KT\log(T)}) \tag{2.8}$$

In the above, we present the upper bounds of algorithms' regret which measure their worst-case performance. To justify that an algorithm is optimal or close to

Algorithm 2: `Thompson Sampling`

Input : $\mathscr{A}$: Arm set, T: Time Horizon

for $t \in [1,T]$ **do**

1. For each arm a, sample $\hat{y}_t(a)$ from distribution $\mathbb{P}_t^a$.
2. Choose arm $a_t = \arg\max_{a\in\mathscr{A}} \hat{y}_t(a)$.
3. Receive y_t, update distribution $\mathbb{P}_t^a$.

end

optimal, it is necessary to compare the upper bound with a lower bound. In the literature, there exits two types of lower bound: worst-case (minimax) and instance-dependent lower bound.

Theorem 3. *(Worst-case regret lower bound) [10] Fixed time horizon T and the number of arms K. For any bandit algorithm, there exists a problem instance such that*

$$R(T) \geq \Omega(\sqrt{KT}) \tag{2.9}$$

This lower bound implies the upper bounds of `UCB1` and `Thompson Sampling`, stated above, are essentially close to the optimality except the $\log(T)$ term. To merge the $\log(T)$ gap, many effort have been devoted. In the next remark, we briefly review the progress been made.

Remark 1. *A variant of `UCB1`, called `MOSS` [27], removes the log term entirely by refining the arm index as shown in Algorithm 3.*

$$R(T) \leq \mathscr{O}(\sqrt{KT}). \tag{2.10}$$

A phased elimination [28] algorithm achieves the optimal regret rate, $R(T) \leq \mathscr{O}(\sqrt{KT\log(K)})$ which is an improvement when $K \leq T$.

The regret bound of `Thompson Sampling`, stated in Theorem 2, is obtained by [29], which is then improved by [30], removing the log factor $R(T) \leq \mathscr{O}(\sqrt{KT})$.

Theorem 4. *(instance-dependent regret lower bound) For stochastic bandit instance, let T be the time horizon. The instance-dependent lower bound is in the following*

Algorithm 3: MOSS [27]

Input : $\mathscr{A}$, T

for $t \in [1,T]$ **do**

1. Choose each arm once.
2. Subsequently choose

$$a_t = \arg\max \hat{y}_t(a) + \sqrt{\frac{4}{n_t(a)} \log^+(\frac{T}{Kn_t(a)})} \tag{2.11}$$

where $\log^+(x) = \log\max\{1,x\}$.

end

Algorithm 4: Phased Elimination [28]

Input : $\mathscr{A}$, sequence $(m_l)_l$

$\mathscr{A}_1 = 1,2,3,...K$

for $l = 1,2,3,...$ **do**

1. Choose each arm $a \in \mathscr{A}_l$ exactly m_l times.
2. Calculate $\hat{y}_l(a)$ the empirical average reward for arm a from this phase.
3. Update the active arm set

$$\mathscr{A}_{l+1} = \{a : y_l(a) + 2^{-l} \geq \max_{a \in \mathscr{A}_l} \hat{y}_l(a)\} \tag{2.12}$$

end

form:

$$R(T) = \Omega(c \log T) \tag{2.13}$$

where c is a constant dependent on the bandit problem instance rather than T.

Remark 2. *Both* UCB1 *and* Phased Elimination *satisfy the following regret upper bound [10].*

$$R(T) \leq \mathscr{O}(C \log T) \tag{2.14}$$

where $C = \sum_{a:y(a)\leq y(a^*)} \frac{1}{y(a^*)-y(a)}$

The instance-dependent regret upper bound of Thompson Sampling *is*

stated below [10]:

$$R(T) \leq \mathcal{O}(C_{ts} \log T) \tag{2.15}$$

where $C_{ts} = \sum_{a:y(a)\leq y(a^*)} \frac{y(a^*)-y(a)}{KL(y(a),y(a^*))}$

Comparing the instance-dependent bound and upper bounds, we see that these algorithms are near-optimal up to a constant factor.

2.1.2 Contextual Linear Bandit

In the previous described stochastic bandit setting, the reward (or distribution) is assumed to be independent between arms. The received reward upon selecting an arm is the only information available to the agent. However, practical applications are the seldom the case. For example, in news recommendation scenarios, a recommender system is usually informed with side information regarding users and news features (or meta-data). The preference (reward) of users on news (arms) can be predicted to some extent based on the side information. Such side information, called context, provides a hint on the reward of arms. In this section, we introduce the *Contextual Bandit* problem, a framework modelling MAB problems in which contextual information is available.

In contextual bandit, at each time step $t \in [T]$, before selecting the arm, the agent is informed a context $C_t \in \mathscr{C}$ generated by the environment, with $\mathscr{C}$ being the set of contexts. Both $\mathscr{C}$ and the generate process of C_t from $\mathscr{C}$ is unknown to the agent and independent to agent's arm selection. Given this context C_t, the agent selects an arm a_t from the arm set $\mathscr{A}$ and receives a reward y_t which is dependent on both C_t and a_t. Formally,

$$y_t = f(C_t, a_t) + \eta_t \tag{2.16}$$

where $f : \mathscr{C} \times \mathscr{A} \to \mathbb{R}$ is the unknown reward function and η_t is a noise term.

Denote the best arm, at each time step t, as $a_t^* = \arg\max_{a\in\mathscr{A}} f(C_t, a)$. Note that the best arm is time-dependent due to its dependency of context. Denote the agent's policy as π. The policy regret is defined as

$$R_T(\pi) = \mathbb{E}\left[\sum_{t=1}^{T} f(C_t, a_t^*) - \sum_{t=1}^{T} y_t\right] \tag{2.17}$$

The expectation is taken over the randomness induced by context generate process, reward noise and arm selection process. Note that we make no restriction on the form of the reward function $f(\cdot)$. Under the generic setting, [20] showed the worst-case regret over any contextual bandit, with $|\mathscr{C}| = M$ contexts and K arms, is at least $\Omega(\sqrt{TMK})$. The $\sqrt{MK}$ dependency over contexts number and arm number might be infeasible in practical applications. For example, in production recommendation for online shopping site which might have million users (M) and hundreds of thousands of products (K). To make the problem more tractable, some assumptions can be made to pose structures on the reward function.

Lipschitz contextual bandit [31] [32] [33], as an example, assumes the reward function is Lipchitz continuous with respect to the contexts and arms. Formally,

$$|f(C,a) - f(C',a')| \leq D_{\mathscr{C}}(C,C') + D_{\mathscr{A}}(a,a'),\ \forall (C,C') \in \mathscr{C} \text{ and } \forall (a,a') \in \mathscr{A} \tag{2.18}$$

where $D_{\mathscr{C}}(\cdot,\cdot)$ and $D_{\mathscr{A}}(\cdot,\cdot)$ are metrics in context space and arm space, respectively.

Another line of work poses linearity on the reward function [34] [35] [36] [37] [38] where the expected reward of each arm follows an unknown linear function. Usually, the agent has the access to a feature mapping $\phi : \mathscr{C} \times \mathscr{A} \to \mathbb{R}^d$, which maps a context-arm pair to a vector feature $\phi(C,a) \in \mathbb{R}^d$ with dimensionality d. The corresponding reward follows an unknown linear function in terms of the feature vector. Namely, $y = \phi(C,a)^T \theta$ where $\theta \in \mathbb{R}^d$ is the fixed but unknown parameter vector and η is a noise term.

There are several slightly different variants [36] [37] [38] of contextual linear bandit in the literature. In the following, we focus on a specific instance defined in [37], upon which our contributions are built.

At each time step $t \in [T]$, the agent chooses an arm $a_t \in \mathscr{A}$ from the arm set. Each arm is represented by a vector feature $\phi(a_t) \in \mathbb{R}^d$, called arm feature. Given the selected arm, the reward y_t is received which is assumed has the following form:

$$y_t = \phi(a_t)^T \theta + \eta_t \tag{2.19}$$

where $\theta \in \mathbb{R}^d$ is the unknown function parameter and η_t is the noise term. The noise term is assumed to be conditional $\sigma-$subgaussian, such that

$$\mathbb{E}\left[\lambda \eta_t | \mathscr{F}_t\right] \leq \exp \frac{\lambda^2 \sigma^2}{2}, \ \forall \lambda \in \mathbb{R} \tag{2.20}$$

where $\mathscr{F}_t = \sigma(a_1, y_1, a_2, y_2, ..., a_{t-1}, y_{t-1})$ summaries the information up to time step t. The $\sigma-$subgaussion means $\mathbb{E}[\eta_t | \mathscr{F}_t] = 0$ and $Var(\eta_t | \mathscr{F}_t] \leq \sigma^2$.

By denoting the optimal arm as $a^* = \arg\max_{a \in \mathscr{A}} \phi(a)^T \theta$, the regret of a policy is defined as

$$R_T(\pi) = \sum_{t=1}^{T} \theta^T \phi^* - \sum_{t=1}^{T} \theta^T \phi(a_t) \tag{2.21}$$

The goal of the agent is to find the policy such that minimizes the regret. In the following, we present two fundamental algorithms, `LinUCB` [35] and `LinTS` [26], which extend `UCB1` and `Thompson Sampling` to contextual linear bandit, respectively.

LinUCB

Similarly to `UCB1`, `LinUCB` [35] is built on the *Optimism in Face of Uncertainty* principle. To take into account the linear model, the confidence interval in Eq. (2.4) is replaced by a confidence set of θ which contains the unknown ground-truth θ with high probability $1-\delta$. This set is defined as

$$\mathbf{C}_t = \{\theta : |\hat{\theta}_t - \theta| \leq \beta_t\} \tag{2.22}$$

with

$$\beta_t = \sqrt{\alpha} S + \sqrt{2 \log\left(\frac{1}{\delta}\right) + d \log\left(1 + \frac{TS}{d\alpha}\right)} \tag{2.23}$$

under the following assumptions: $x_a \in \mathbb{R}^d$ and $||x_a|| \leq S$ for $a \in \mathscr{A}$, $\hat{\theta}_t = V_t^{-1} B_t$ with $V_t = \alpha I + \sum_{s=1}^{t} x_{a_s} x_{a_s}^T$ and $B_t = \sum_{s=1}^{t} y_t x_{a_s}$ and $\delta > 0$.
Then, the confidence interval of reward can be expressed as

$$\hat{\theta}_t^T x_a - \beta_t(a) ||x_a||_{V_t^{-1}} \leq y_t(a) \leq \hat{\theta}_t^T x_a + \beta_t(a) ||x_a||_{V_t^{-1}} \tag{2.24}$$

Algorithm 5: LinUCB

Input : $\mathscr{A}$, T

for $t \in [1,T]$ **do**

1. Construct a confidence set $\mathbf{C}_t$ such that $\theta \in \mathbf{C}_t$ in high probability.
2. Choose arm $a_t = \arg\max_{a \in \mathscr{A}} UCB_t(a)$.
3. Receive y_t and update confidence set $\mathbf{C}_t$.

end

Then, the upper confidence bound is defined as

$$UCB_t(a) = \hat{\theta}_t^T x_a + \beta_t(a)||x_a||_{V_t^{-1}} \tag{2.25}$$

At time step t, the agent selects the arm with maximum $UCB_t(a)$ to balance the exploration and exploitation:

$$a_t = \arg\max_{a \in \mathscr{A}} UCB_t(a) \tag{2.26}$$

Theorem 5. *[35] Suppose $x_a \in \mathbb{R}^d$ and $||x_a|| \leq S$ for $a \in \mathscr{A}$. Let $\hat{\theta}_t = V_t^{-1} B_t$ with $V_t = \alpha I + \sum_{s=1}^t x_{a_s} x_{a_s}^T$ and $B_t = \sum_{s=1}^t y_t x_{a_s}$. For any $\delta > 0$, with probability at least δ, LinUCB achieves regret*

$$R_T = \mathscr{O}(d\sqrt{T}\log(TS)) \tag{2.27}$$

LinTS

Similar to Thompson Sampling, LinTS is based on *Probabilistic Randomization*. Given a prior on the reward distribution of each arm and likelihood function, the agent selects the arm with the highest expected reward, according to the posterior distribution. More precisely, LinTS [26] models the posterior distribution of each arm reward based on Gaussian prior and Gaussian likelihood function. The reward of each arm is modelled as a posterior distribution. More specifically, the reward of each arm is modelled as a Gaussian distribution $\mathscr{N}(\hat{y}_t(a), \sigma^2 V_t^{-1})$. The mean

Algorithm 6: LinTS

Input : $\mathscr{A}$, T

for $t \in [1,T]$ **do**

1. For each arm a, sample $\hat{y}_t(a)$ from distribution $\mathbb{P}_t^a$.
2. Choose arm $a_t = \arg\max_{a \in \mathscr{A}} \hat{y}_t(a)$.
3. Receive y_t and update distribution $\mathbb{P}_t^a$.

end

reward of each arm is estimated via least-square estimator as follows

$$\hat{y}_t(a) = \hat{\theta}_t^T x_a = (V_t^{-1} B_t)^T x_a \tag{2.28}$$

where $V_t = \sum_{s=1}^{t} x_{a_s} x_{a_s}^T$ and $B_t = \sum_{s=1}^{t} y_s x_{a_s}$.

Theorem 6. *[26] LinTS achieves regret*

$$R_T = \mathscr{O}(d\sqrt{Tlog(T)\log(T/d)}) \tag{2.29}$$

2.1.3 Multi-task Contextual Linear Bandit

In this section, we present the basic setting of Multi-task Contextual Linear Bandit (MLB) problem. One of our contribution is dedicated to solve this problem.

In MLB [39] [40], the agent is required to solve multiple contextual linear bandit tasks. In practical applications, this setting mimics the task faced by a recommender system, which recommends items to multiple users. Each user represents a bandit instance and all users (bandit instances) share a same set of candidate items (arm set). The bandit parameter vectors represent users' preference and items are arms characterised by arm features (contexts). The reward corresponds to user's preference to the recommended item.

Formally, the agent is given an arm set $\mathscr{A}$, each arm $a \in \mathscr{A}$ is associated with a feature $\phi(a) \in \mathbb{R}^d$. The environment consists of a set of bandit problem $\mathscr{B}$, where each bandit instance $b \in \mathscr{B}$ is characterized by a fixed by unknown parameter vector

$\theta_b \in \mathbb{R}^d$. Over a time horizon T, the agent is required to solve bandit problems sequential. At each time step t, a bandit problem b_t is randomly sampled from $\mathscr{B}$. The agent selects an arm $a_t \in \mathscr{A}$ for this bandit and receives a reward whose expected value is a linear combination of the arm feature and bandit parameter. Namely,

$$y_t = \theta_{b_t}^T \phi(a_t) + \eta_t \tag{2.30}$$

where the noise term η_t is modelled as a conditional σ-subgaussian variable.

The agent aims to find a policy π that minimizes the cumulative regret over the time horizon.

$$R_T(\pi) = \sum_{t=1}^{T} \theta_{b_t}^T \phi(a_{b_t}^*) - \sum_{t=1}^{T} \theta_{b_t}^T \phi(a_t) \tag{2.31}$$

where $a_{b_t}^* = \arg\max_{a \in \mathscr{A}} \theta_{b_t}^T \phi(a)$ is the optimal arm with respect the bandit instance $b_t \in \mathscr{B}$.

Suppose the number of contextual linear bandit instance to solve is n and bandit instances are sampled uniformly across the time horizon T. Each bandit will appear roughly $T' = T/n$ times. If the agent solves each bandit independently, the cumulative regret of each bandit can be upper bounded as $R(T') = \mathscr{O}(d\sqrt{T'})$. The total regret will be $R(T) = \mathscr{O}(nd\sqrt{T'})$, which scales linearly with the number of bandit n. To mitigate this, we can define a notion of *similarity* between tasks, which can be exploited by transferring the knowledge of one task to other tasks. One of our contribution – *Laplacian-regularized Bandits* in Chapter 4 – focuses on how to exploit the task similarity prior knowledge and how much it can reduce the cumulative regret. In essence, we propose a term $\Psi \in (0,1)$ measuring the similarity among bandit instances. We show that the total regret scales $R(T) = \mathscr{O}(m\Psi d\sqrt{T'})$ and prove that if bandit instances are closely related (similar), the term $\Psi \to 0$ leading to a lower regret.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a learning framework formalizing the sequential decision making problems. As shown in Fig 2.1, RL consists of two components:

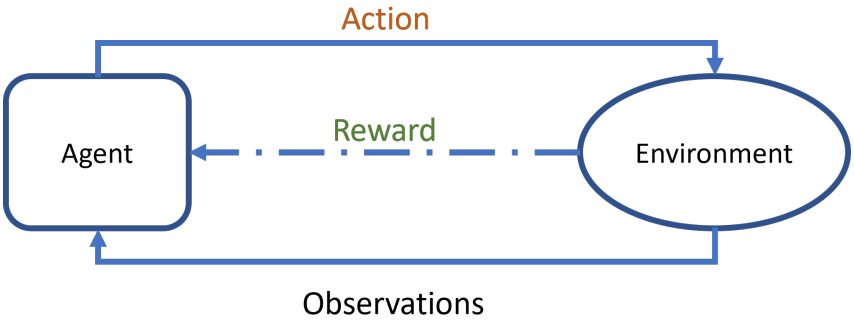


Figure 2.1: Depiction of Reinforcement Learning Framework.

the agent and the environment. The agent learns how to take decisions through sequential interactions with the environment. Within each interaction, the agent senses the environment state through observations and then takes an action. The environment transits to a new state due to the action and internal dynamic. The agent receives a feedback signal (reward) from the environment. Then, the next interaction continues. The agent aims to maximize the cumulative reward over interactions. This way of learning mimics the foundational way of which we human learn.

In this thesis, we consider the standard RL setting in which the agent-environmnet interaction is modelled as a Markov Decision Process (MDP). An MDP is described as a five-tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$, where $\mathscr{S}$ is a finite set of states, $\mathscr{A}$ is a finite set of actions, $\mathscr{P}$ is the transition function with $\mathscr{P}(s'|s,a)$ denoting the distribution of next state s' given action a is taken in state s. R is the reward function where the expected reward obtained if action a is taken in state s is denoted as $R(s,a)$; and $\gamma \in [0,1)$ is a discount factor.

The agent interacts with the environment in the following way: at each time step $t \in [T]$, the agent observes the state of the environment s_t and then it takes an action $a_t \in \mathscr{A}$. The environment transits to next state according the transition dynamic $s_{t+1} \sim \mathscr{P}(\cdot|s_t,a_t)$. The agent receives a reward $r_t = R(s_t,a_t)$. The sequence of interactions recorded up to time step t is called *trajectory*:

$$\mathscr{T}_t = (s_0, a_0, r_0; s_1, a_1, r_1; ...; s_t, a_t, r_t) \tag{2.32}$$

A policy π is a mapping from state to action space: $\pi : \mathscr{S} \to \Delta\mathscr{A}$, which is the agent's action selection strategy. A *stochastic* policy: $\pi : \mathscr{S} \to \mathscr{P}(\mathscr{A})$ is a mapping from state to the distribution of actions. A *deterministic* policy $\pi : \mathscr{S} \to \mathscr{A}$ is a mapping from state to a specific action.

The state value $V_\pi(s)$ under a policy π is defined as the expected discounted cumulative reward received by executing π starting from state s.

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_t \sim \pi(\cdot|s_t)\right] \tag{2.33}$$

The expectation is taken over the randomness of trajectory which is caused by the state transition and stochasticity of policy.

Similarly, the state-action value function $Q_\pi(s,a)$, under policy π, is the expected discounted cumulative reward received by taking action a in state s and following π afterward, namely

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a, a_t \sim \pi(\cdot|s_t)\right]. \tag{2.34}$$

Value functions satisfy the *Bellman Expectation Equation* [1]:

$$V_\pi(s) = \mathbb{E}_\pi\Big[R(s_t) + \gamma V_\pi(s_{t+1}) | s_t = s\Big] \tag{2.35}$$

$$Q_\pi(s,a) = \mathbb{E}_\pi\Big[R(s_t, a_t) + \gamma Q_\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a\Big] \tag{2.36}$$

and there exists a relation between $V_\pi(s)$ and $Q_\pi(s,a)$:

$$V_\pi(s) = \mathbb{E}_{a\in\pi(\cdot|s)} Q_\pi(s,a) = \sum_{a\in\mathscr{A}} \pi(a|s) Q_\pi(s,a). \tag{2.37}$$

The goal of the agent is to find the optimal policy π^* that maximizes the value function of each state:

$$\pi^* = \arg\max_{\pi\in\Pi} V_\pi(s) \quad \forall s \in \mathscr{S} \tag{2.38}$$

where Π is the set of stationary policies.

The theorem stated below guarantees the optimal policy π_* always exits for a given MDP.

Theorem 7. *[41] For any Markov Decision Process (MDP), there exists at least one optimal policy π^* whose value is better than or equal to all other policies*

$$V_{\pi^*} = \sup_{\pi \in \Pi} V_\pi(s), \quad \forall s \in \mathscr{S} \tag{2.39}$$

$$Q_{\pi^*} = \sup_{\pi \in \Pi} Q_\pi(s,a), \quad \forall (s,a) \in \mathscr{S} \times \mathscr{A} \tag{2.40}$$

Where V_{π^*} and Q_{π^*} are value functions of the optimal policy.

The value functions, V_{π^*} and Q_{π^*}, satisfy the *Bellman Optimality Equation*:

$$V_{\pi^*}(s) = \mathbb{E}_\pi \Big[R(s) + \gamma V_{\pi^*}(s_{t+1}) | s_t = s \Big] \tag{2.41}$$

$$Q_{\pi^*}(s,a) = \mathbb{E}_\pi \Big[R(s,a) + \gamma \max_{a \in \mathscr{A}} Q_{\pi^*}(s_{t+1},a) | s_t = s, a_t = a \Big] \tag{2.42}$$

There exists a close relationship between $V_{\pi^*}(s)$ and $Q_{\pi^*}(s,a)$:

$$V_{\pi^*}(s) = \max_{a \in \mathscr{A}} Q_{\pi^*}(s,a) \tag{2.43}$$

In most RL problems, we either concern finding the optimal policy (control) or evaluating the value of a predefined policy (prediction).

2.2.1 Prediction

In prediction (also called policy evaluation) problem, we aim to compute the value functions, V_π and Q_π, of a prescribed policy π. Recall the *Bellman Expectation Equations* Eq. (2.35), which provide a recursive relation between consecutive states/state-actions, and can be leveraged for policy evaluation.

$$V_\pi(s) = \mathbb{E}_\pi \Big[R(s_t) + \gamma V_\pi(s_{t+1}) | s_t = s \Big] \tag{2.44}$$

$$Q_\pi(s,a) = \mathbb{E}_\pi \Big[R(s_t,a_t) + \gamma Q_\pi(s_{t+1},a_{t+1}) | s_t = s, a_t = a \Big] \tag{2.45}$$

Algorithm 7: Iterative Policy Evaluation [1]

Input : An MDP $\mathscr{M} : (\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$; a fixed policy π
Initialization: $k = 0$;
$V^{(k)}(s) = 0,\ \forall s \in \mathscr{S}$;
$Q^{(k)}(s,a) = 0,\ \forall s,a \in \mathscr{S} \times \mathscr{A}$;
for $k \in [1, \infty)$ **do**

1. update state value for each $s \in \mathscr{S}$ following:

$$V^{(k+1)}(s) \leftarrow \sum_{a \in \mathscr{A}} \pi(a|s) \left[r(s,a) + \sum_{s'} \mathscr{P}(s'|s) V^{(k)}(s') \right] \tag{2.48}$$

2. or update value of each state-action pair $(s,a) \in \mathscr{S} \times \mathscr{A}$:

$$Q^{(k+1)}(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a) \sum_{a' \mathscr{A}} \pi(a'|s') Q^{(k)}(s',a') \tag{2.49}$$

end
Output : $\lim_{k \to \infty} V^{(k)} \to V_\pi$; $\lim_{k \to \infty} Q^{(k)} \to Q_\pi$

Based on the *Bellman Expectation Equation*, we can introduce the Bellman expectation operator $T^\pi : \mathscr{V} \to \mathscr{V}$ where $\mathscr{V}$ is the real-valued functions with support $\mathscr{S}$.

$$(T^\pi f)(s) = \sum_{a \in \mathscr{A}} \pi(s,a) \left[r(s) + \gamma \sum_{s'} \mathscr{P}(s'|s,a) f(s') \right],\ \forall f \in \mathscr{V} \tag{2.46}$$

Similarly, we can define the Bellman operator for Q_π.

$$(T^\pi f)(s,a) = r(s,a) + \gamma \sum_{s'} \mathscr{P}(s'|s,a) \sum_{a' \in \mathscr{A}} \pi(a'|s') f(s',a'),\ \forall f \in \mathscr{Q} \tag{2.47}$$

where $\mathscr{Q}$ is the set of real-valued function with support state-action space $\mathscr{S} \times \mathscr{A}$.

It was proven [42] that there is an unique fixed point of the operators which are the value functions V_π and Q_π, respectively. This suggests that the value function can be computed by iterative solving methods. Specifically, we turn the *Bellman Expectation Operators* into an iterative updated rules. A specified algorithm is shown in Algorithm 7.

Algorithm 8: Policy Iteration [1]

Input : An MDP $\mathscr{M} : (\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$
Initialization: $k = 0$;
$Q^{(k)}(s,a) = 0,\ \forall s,a \in \mathscr{S} \times \mathscr{A}$;
$\pi^{(k)}(s) \in \mathscr{A}$ randomly.

while $Q^{(k+1)} \neq Q^{(k)}$ **do**

1. Policy Evaluation: update value of each state-action pair $(s,a) \in \mathscr{S} \times \mathscr{A}$.

$$Q^{(k+1)}(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a) \sum_{a' \mathscr{A}} \pi^{(k)}(a'|s') Q^{(k)}(s',a') \tag{2.50}$$

2. Policy Improvement: update policy for each state $s \in \mathscr{S}$.

$$\pi^{(k+1)}(s) = \arg\max_{a \in \mathscr{A}} Q^{(k)}(s,a),\ \forall s \in \mathscr{S} \tag{2.51}$$

end
Output : $\lim_{k\to\infty} Q^{(k)} \to Q_{\pi^*}$ and $\lim_{k\to\infty} \pi^k \to \pi^*$.

2.2.2 Control

Now, we move our attention to the control problem: how does the agent learn the optimal policy for an MDP? To achieve this, we first tackle the challenge of how to improve the policy given an arbitrary policy π and the associated value function Q_π. Theorem 8 answers the question saying that the greedy policy with respect to Q_π is guaranteed to be a better policy.

Theorem 8. *(Greedy Policy Improvement) [1] Given a MDP $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$ and a policy π and value function Q_π. Define a policy π' which is obtained by acting greedily with respect to π. Formally,*

$$\pi'(s) = \arg\max_{a \in \mathscr{A}} Q^\pi(s,a),\ \forall s \in \mathscr{S}. \tag{2.52}$$

We have the guarantee that π' is an improvement over π. Namely,

$$Q_{\pi'}(s,a) \geq Q_\pi(s,a),\ \forall (s,a) \in \mathscr{S} \times \mathscr{A}. \tag{2.53}$$

Theorem 8 suggests that if we know the optimal value function Q^*, we can

Algorithm 9: Q-Learning [43]

Input : An MDP $\mathscr{M} : (\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$;
A behaviour policy π;
Learning rate: α;

Initialization: $\hat{Q}(s,a)$, $\forall (s,a) \in \mathscr{S} \times \mathscr{A}$ arbitrarily.

for *each episode* **do**
  for *for each step t within episode* **do**
    1. Take an action $a_t \sim \pi(s_t)$.
    2. Receive a reward r_t and next state s_{t+1}.
    3. Compute TD error $\delta_{t+1} = r_t + \gamma \max_{a \in \mathscr{A}} \hat{Q}(s_t, a) - \hat{Q}(s_t, a_t)$.
    4. Update $\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha(r_t + \gamma \max_{a \in \mathscr{A}} \hat{Q}(s_t, a) - \hat{Q}(s_t, a_t))$.
  end
end

Output : $\hat{Q}(s,a)$ $\forall (s,a) \in \mathscr{S} \times \mathscr{A}$

obtain an optimal policy π^* by acting greedily according to Q^*:

$$\pi^*(s) = \arg\max_{a \in \mathscr{A}} Q^*(s,a), \ \forall s \in \mathscr{S}. \tag{2.54}$$

The next question is how to compute the optimal value function. We can follow the method called **Policy Iteration** which contains two steps: (1) *Policy evaluation*: evaluate the current policy $Q_\pi(s,a)$; (2) *Policy improvement*: improve the policy to π' by acting greedily according to Q_π. Algorithm 8 shows a complete algorithm of policy iteration.

In the above discussions, we assume a full knowledge of the MDP, in particular, the transition probability $\mathscr{P}$ and reward function R. However, this is rarely the case in real-world problems. In standard RL settings, the agent has no access to such knowledge. Instead, the agent has to interact with the environment and generate experiences (trajectories) to solve prediction or control problems. In the next section, we present some foundational methods for solving RL problems only based on interaction experiences.

Q-Learning

In this section, we describe an algorithm (Q-Learning [43] [44]), which aims to estimate the optimal action-value function Q^* from the collected trajectories[1]. The intuition behind is that if the estimated action-value function $\hat{Q}$ is close to Q^*, acting greedily over $\hat{Q}$ will generate a policy close to optimal. Q-Learning can be viewed as a stochastic and approximated version of Policy Iteration (Algorithm 8). Q-Learning leverages on the *Bellman Optimility Operator* as an iterative update rule, which is guaranteed to converge to the optimal value function when the time horizon tends to infinity. Specifically, for a given MDP $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$, Q-Learning maintains an estimation of action-value $\hat{Q}(s,a)$ of each state-action pair $(s,a) \in \mathscr{S} \times \mathscr{A}$. The update rule is defined as:

$$\hat{Q}(s_t,a_t) \leftarrow \hat{Q}(s_t,a_t) + \alpha[r_t + \gamma \max_{a \in \mathscr{A}} \hat{Q}(s_t,a) - \hat{Q}(s_t,a_t)] \tag{2.55}$$

where $\alpha \in \mathbb{R}$ is the learning rate and $\delta_{t+1} = r_t + \gamma \max_{a \in \mathscr{A}} \hat{Q}(s_{t+1},a) - \hat{Q}(s_t,a_t)$ is the learning target, called TD-error. The full algorithm of Q-Learning is shown in Algorithm 10.

Q-Learning aims to minimize the TD error over all state-action pairs. At convergence, one has $\delta_{t+1}(s,a) = 0,\ \forall (s,a) \in \mathscr{S} \times \mathscr{A}$, which is equivalent to say that $\hat{Q}(s_t,a_t) = r + \gamma \max_{a \in \mathscr{A}} \hat{Q}(s_{t+1},a)$. This implies that the estimated action-value function $\hat{Q}$ satisfies the *Bellman Optimality Operator*. As the operator has an unique fixed point Q^*, it implies $\hat{Q} = Q^*$. More details on the behaviour policy π, learning rate α and the proof of convergence can be found in [42].

2.2.3 Function Approximation

In previous sections, we assume that we can enumerate the state-action pairs $(s,a) \in \mathscr{S} \times \mathscr{A}$ and maintain/compute the value of each state-action pair explicitly. In case of large or continuous state/action space, it is infeasible to maintain a lookup table and

[1] Other works aim at learning the MDP transitions (i.e., the model) [45] or directly the policy [46].

typically resort to approximate the value functions through function approximation.

$$V_\pi(s) = f_\theta(s) \;\; or \;\; Q_\pi(s,a) = f_\theta(s,a) \tag{2.56}$$

where θ is a set of parameters that parametrizes unknown functions.

We aim at learning θ based on the interaction experiences of the agent, such that f_θ approximates the value function V_π or Q_π as accurate as possible. A commonly used approach is to parameterize the value functions as a linear function of representation:

$$V(s) \approx \phi(s)^T\theta, \;\; or \;\; Q(s,a) \approx \phi(s,a)^T\theta \tag{2.57}$$

where $\phi(\cdot) \in \mathbb{R}^d$ is representations.

In the following, we discuss several algorithms that aim at solving the RL control problem using function approximation and following the **Policy Iteration** procedure.

Fitted Q-Iteration

Fitted Q-Iteration (FQI) [47] is an extension of Q-Learning to function approximation. In FQI, the agent is access to a set of interaction experiences $\mathscr{D}$, called sample set. Each element of $\mathscr{D}$ is a tuple of interactions (s_t, a_t, r_t, s_{t+1}). FQI aims to learn the optimal action-value Q^* which is parameterized by a function f_θ. With each sample experience, FQI updates the parameter θ by minimizing an objective defined as below:

$$\delta_t = (f_\theta(s_t,a_t) - (r_t + \gamma \max_{a\in\mathscr{A}} f_\theta(s_{t+1},a))^2 \tag{2.58}$$

DQN

Deep Q-Net (DQN), introduced in [3], employs Neural-Network as a powerful approximator to approximate the optimal action-value function Q^*. Similar to FQI, the optimal value function is modeled as a parameterized function $f_\theta \approx Q^*$. In DQN,

Algorithm 10: Fitted Q-Iteration [47]

Input : An MDP $\mathscr{M}$: $(\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$;
A sample set $\mathscr{D}$

Initialization: $f_\theta(\cdot,\cdot)$

while *Not Converge* **do**

1. Take an experience tuple (s, a, r, s')
2. Compute value estimation $f_\theta(s,a)$.
3. Compute value target $r + \gamma \max_{a\in\mathscr{A}} f_\theta(s',a)$
4. Update the parameter θ via minimizing the objective

$$\left(f_\theta(s,a) - (r_t + \gamma \max_{a\in\mathscr{A}} f_\theta(s',a)\right)^2 \tag{2.59}$$

end

Output : $f_\theta(\cdot,\cdot)$

θ is the weights of neural-network which is updated according to

$$\theta' \leftarrow \theta + \alpha\left(r_t + \gamma \max_{a\in\mathscr{A}} f_{\bar{\theta}}(s_t,a) - f_\theta(s_t,a_t)\right) \tag{2.60}$$

where $f_{\bar{\theta}}$ is output by a target network (with weights $\bar{\theta}$). The target network is updated periodically and less frequent than f_θ and it is used to improve the stability of training. DQN maintains a replay buffer $\mathscr{D}$ to contain interaction experiences. The buffer has a fixed size where old experiences are gradually replaced by new coming experiences. At each update steps, a mini-batch of experiences are randomly sampled from $\mathscr{D}$, and then networks are updated following Eq. (2.60).

Since its introduction, DQN has sparked a large amount of work combining RL algorithms with deep neural networks and demonstrating impressive ability in solving complex tasks.

Actor-Critic

In the previous section, we rely on value functions to obtain the optimal policy (acting greedily). An alternative approach is to parameterize the policy directly $\pi_\theta(a|s)$ and optimize the parameters with respect to a performance objective $J(\pi_\theta)$. Such approaches are called policy-gradient algorithms [1].

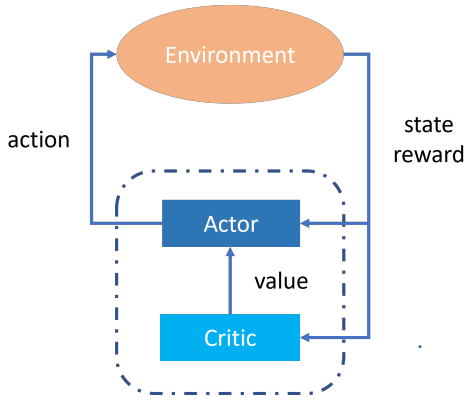


Figure 2.2: The Actor-Critic Architecture

The performance objective $J(\pi_\theta)$ is defined to measure the goodness of the policy π_θ. Various definitions exist in the literature for such objective function [1]. For example, in episodic RL, the agent starts from an initial state s_0 at the beginning of each episode and $J(\pi_\theta)$ is defined as the return of s_0:

$$J(\pi_\theta) = V_{\pi_\theta}(s_0) \tag{2.61}$$

Given $J(\pi_\theta)$, finding the optimal policy is an optimization problem: searching the θ that maximizes $J(\pi_\theta)$. This can be solved by gradient-ascent searching for local optimum points. For this, the gradient of θ with respect to the objective $J(\pi_\theta)$ is needed:

$$\Delta\theta = \alpha \bigtriangledown_\theta J(\theta) \tag{2.62}$$

where $\bigtriangledown_\theta J(\theta)$ is the policy gradient, α is learning rate.

In the following, we present how to compute the policy gradient $\bigtriangledown_\theta J(\theta)$. Assume the policy π_θ is differentiable and denote the gradient as $\bigtriangledown_\theta \pi_\theta(s,a)$. The likelihood ratio is defined as

$$\bigtriangledown_\theta \pi_\theta(s,a) = \pi_\theta(s,a)\frac{\bigtriangledown_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} = \pi_\theta \bigtriangledown_\theta \log \pi_\theta(s,a) \tag{2.63}$$

where $\bigtriangledown_\theta \log \pi_\theta(s,a)$ is called score function.

Theorem 9. ***(Policy Gradient Theorem)****[48] For any differentiable policy* $\pi_\theta(s,a)$,

Algorithm 11: Actor-Critic [49]

Input : An MDP $\mathscr{M} : (\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$;
Learning rates α, β;
Initialization: $\pi_\theta(\cdot)$ and $f_w(\cdot,\cdot)$.

for *each episode* **do**
  for *each step t* **do**
    1. take action $a_t \sim \pi_\theta(s_t)$ and receive r_t and s_{t+1}.
    2. sample action $a_{t+1} \sim \pi_\theta(s_{t+1})$
    3. compute value function error: $\delta_t = r_t + \gamma f_w(s_{t+1}, a_{t+1}) - f_w(s_t, a_t)$.
    4. update f_w:$w \leftarrow w + \alpha \delta_t \bigtriangledown_w f_w(s_t, a_t)$.
    5. update π_θ: $\theta \leftarrow \theta + \beta f_w(s_t, a_t) \bigtriangledown_\theta \log \pi_\theta(s_t)$.
  end
end
Output : $f_w(\cdot,\cdot)$ and $\pi_\theta(\cdot)$

for the policy objective function defined above, the policy gradient is

$$\bigtriangledown_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[Q_{\pi_\theta}(s,a) \bigtriangledown_\theta \log \pi_\theta(s,a) \right]. \tag{2.64}$$

Theorem 9 leads to the following update rule for θ for each transition (s_t, a_t, s_{t+1}),

$$\theta' \leftarrow \theta + \alpha Q_{\pi_\theta}(s_t, a_t) \bigtriangledown_\theta \log \pi_\theta(s_t, a_t). \tag{2.65}$$

However, there is an issue that is the knowledge of $Q_{\pi_\theta}(s_t, a_t)$. As discussed in the previous sections, we can use function approximation to estimate the action value function $f_w(s_t, a_t) \approx Q_{\pi_\theta}(s_t, a_t)$. This approach is called `Actor-Critic` in the literature.

`Actor-Critic` (Algorithm 11) is an instantiate algorithm of *Generalized Policy Iteration*. Recall that *Generalized Policy Iteration* involves two alternative steps: policy evaluation and policy improvement. In `Actor-Critic`, the critic $f_w(\cdot,\cdot)$ aims to estimate the value function of behavioural policy, while the actor $\pi_\theta(\cdot)$ is responsible to generate an improved policy given the estimated value function from critic. Fig 2.2 shows a typical architecture of `Actor-Critic` algorithm.

2.2.4 Representation Learning

In most real-life RL tasks, the state-action space is too large to allow a tabular representation for the value of each state-action pair, thus one must resort to function approximation. A function approximator, e.g., Neural-Network, takes as input the observation of the environment and outputs the estimated value or policy. Under the assumption of MDP, the observation represents the current state of the environment upon which the agent is required to make decisions. However, many RL tasks present raw, high-dimensional observations to agents (e.g., video games), which usually contain redundant and distracting information. Such observations posit a challenge for the agent as it needs to first extract relevant information and then acts accordingly. For example, in autonomous vehicle applications, the agent makes decisions upon observations of outdoor environment, which might contain irrelevant and distracting information such as cloud shape, colour of trees and details of buildings.

Therefore, the data efficiency of RL algorithms (agent) heavily relies on its ability in learning compact representations, which serves to approximate the value function and extract a new policy. To achieve this, representation learning techniques have been leveraged as a key component in modern RL algorithms. Representation learning aims to summarize the observation into a small/compact vectored representation, which are more suitable of solving the RL task, as shown in Fig 2.3.

Denote the observation of state as $o(s) \in \mathbb{R}^k,\ \forall s \in \mathscr{S}$ which could lie in a high-dimensional observation space $\mathscr{O}$ (e.g., pixels). We assume there exists a low-dimensional space Φ containing all the relevant information required to solve the RL task, in which the state representation is denoted as $\phi(s) \in \mathbb{R}^d$ where $d << k$. Under the function approximation, value functions are modelled as function over state/state-action representations. Formally,

$$V(s) \approx f_\theta(\phi(s)),\ \ or\ \ Q(s,a) \approx f_\theta(\phi(s,a)) \tag{2.66}$$

where θ are learnable weights, $\phi(\cdot) \in \mathbb{R}^d$ is representation. In deep reinforcement learning (e.g. DQN [3]), ϕ and θ are jointly learned via neural network where ϕ can be viewed as the output of the penultimate layer and θ is the weights of the final

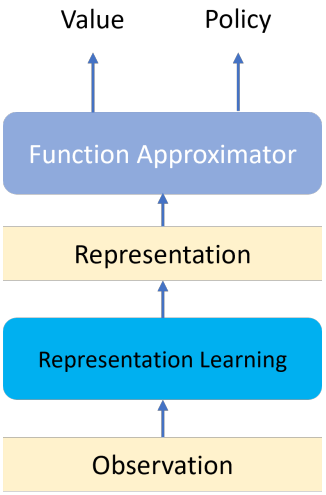


Figure 2.3: The role of representation learning in RL.

layer.

Representation Learning concerns learning the mapping ε from the observation space to the representation space. Namely, $\varepsilon : \mathcal{O} \rightarrow \Phi$. The value function approximations can be rewritten as

$$V(s) \approx f_\theta(\varepsilon(o(s))), \quad or \quad Q(s,a) \approx f_\theta(\varepsilon(o(s,a))) \tag{2.67}$$

where $\phi(s) = \varepsilon(o(s))$ and $\phi(s,a) = \varepsilon(o(s,a))$.

Due to the importance of representation learning in RL, many effort have been devoted resulting in a variety of approaches. Some work argue that representations should be expressive for all possible value functions [50] or value functions induced by the agent's policies [51]. Others believe that representations should be expressive for reward function [52], environment dynamic [53] [54] or state reconstruction [55]. The field of representation learning in RL is experiencing a fast growth. Most of work in RL representation learning take a form of *Auxiliary tasks*. The agent, besides the primary RL task (learning an optimal policy), is required to learn other aspects of the task. Most works demonstrate empirically the benefits of *Auxiliary tasks* on the performance of agents. While there is a limited theoretical understanding on the effectiveness of auxiliary tasks [56], a common view is that auxiliary tasks could improve the learning efficiency by shaping the representation in more semantically

aware ways [52]. Despite enormous amount of works existing in the literature, there is still a lack of consensus on the notion of what is a good representation for RL. In this thesis (Chapter 7), we contribute in improving the representation learning for RL by proposing a theoretical justified approach and argue that a good representation should capture the intrinsic structure of value functions.

In this first part of the thesis, we present the main relevant background knowledge about MAB and RL problems. This poses the basis for establishing the important concepts and for better understanding the contributions done in this PhD thesis.

Chapter 3

Laplacian-Regularized Graph Bandits: Algorithms and Theoretical Analysis

In this part of the thesis, we focus on stochastic linear bandit problems with multiple users, where the relationship between users is captured by an underlying graph and user preferences are represented as smooth signals on the graph. We introduce a novel bandit algorithm where the smoothness prior is imposed via the random-walk graph Laplacian, which leads to a single-user cumulative regret scaling as $\tilde{\mathcal{O}}(\Psi d\sqrt{T'})$ with single-user time horizon T', feature dimensionality d, and the scalar parameter $\Psi \in (0,1)$ that depends on the graph connectivity. This is an improvement over $\tilde{\mathcal{O}}(d\sqrt{T'})$ in `LinUCB` [35], where user relationship is not taken into account. In terms of network regret (sum of cumulative regret over n users), the proposed algorithm leads to a scaling as $\tilde{\mathcal{O}}(\Psi nd\sqrt{T'})$, which is an improvement over $\tilde{\mathcal{O}}(nd\sqrt{T'})$ in the baseline algorithm `Gob.Lin` [57]. To improve scalability, we further propose a simplified algorithm with a linear computational complexity with respect to the number of users, while maintaining the same regret. Finally, we present a finite-time analysis on the proposed algorithms, and demonstrate their advantage in comparison with state-of-the-art graph-based bandit algorithms on both synthetic and real-world data.

3.1 Introduction

In the classical multi-armed bandit (MAB) problem, an agent takes sequential actions, choosing one arm out of the K available ones and it receives an instantaneous payoff upon the chosen arm. The goal is to learn a policy such that maximizes the cumulative payoff over a course of T rounds [17]. MAB problems formalize a trade-off between exploration and exploitation, and a particular solution is imposing the principle of optimism in face of uncertainty. Specifically, the agent assigns each arm with an index called the upper confidence bound (UCB) that with high probability is an overestimate of the unknown payoff, and then selects the arm with the highest index.

Many variants of the basic MAB problem have been intensively studied, motivated by real-world applications such as ads placement and recommender systems. In the stochastic linear bandit [15], at each round, the agent receives a hint before taking the decision. Specifically, before choosing an arm, the agent is informed with a feature vector $x \in \mathbb{R}^d$ associated with each arm, referred to as the 'context'. The payoff associated with each arm is modelled as a linear function of x and an unknown coefficient vector $\theta \in \mathbb{R}^d$ perturbed by a noise term η, i.e, $y = x^T\theta + \eta$. This problem has been well understood in the literature and many studies have already proposed asymptotically optimal algorithms [15, 34, 26, 58, 59]. The problem is less understood in the case of multiple users, as opposed to a single user, where we assume a central agent selects arms for multiple users in a sequential fashion. In the chapter, we are interested in the setting where there are n users to be served by the recommender system. In essence, the agent faces a set of n bandit instance with each one characterized by the unknown coefficients $\theta_i, i \in \{1,2,...,n\}$. At each round, one user out of $[n]$ is selected uniformly at random. The agent needs to select one arm from $\mathscr{A}$ for the user and receives an instantaneous payoff associated with the selected arm and the user. The overall goal is to minimize the cumulative regret (or equivalently, maximize the cumulative payoffs), which is the summation of instantaneous regret over a finite time horizon T.

Naively implementing bandit algorithms (e.g., `LinUCB`) on each user independently will result in a cumulative regret scaling linearly with the number of users n.

Specifically, denote the single-user time horizon as $T' = T/n$, following LinUCB, the regret is $\mathcal{O}(d\sqrt{T'})$ for a single user. Therefore, the total regret will be $\mathcal{O}(nd\sqrt{T'})$, which scales linearly with respect to the number of users n. This is infeasible in case of a large number of users. In many cases, however, the users are related in some way, and this can be represented by a network (or graph) that encapsulates important additional source of information, such as similarities among users in terms of their preferences (user feature vectors). Intuitively, exploiting such relationship can mitigate the scalability problem. The key setup is therefore to construct a graph where each node represents a user and the edges identify the affinity between users. In real-world applications, such a graph can be a social network of users. This idea leads to a series of work on the so-called graph-based bandit problem [57, 60, 61, 62].

Despite previous effort, several important limitations remain to be addressed. First, the graph Laplacian matrix is commonly used in graph-based algorithms, but the justification of its usage remains insufficient (and as to which version of the graph Laplacian leads to optimal policies). Consequently, the advantage of graph-based bandit is largely shown empirically in previous works, without rigorous theoretical analysis. Furthermore, scalability remains a serious limitation of such algorithms. Involving user graph into bandit algorithms typically results in a computational complexity that scales with the number of users, which is infeasible in case of large number of users. In this chapter, we address the above limitations with the following main contributions:

- We propose a bandit algorithm GraphUCB based on the random-walk graph Laplacian, and show its theoretical advantages over other graph Laplacian matrices in reducing cumulative regret. We demonstrate empirically that GraphUCB outperforms state-of-the-art graph-based bandit algorithms in terms of cumulative regret.
- As a key ingredient of the proposed algorithm, we derive a novel UCB representing the single-user bound while embedding the graph structure, which reduces the size of the confidence set, in turn leading to lower regret;
- To improve scalability, we further propose a simplified algorithm

`GraphUCB-Local` whose complexity scales linearly with respect to the number of users, yet still holding the same regret upper bound as `GraphUCB`;

- Finally, we derive a finite-time analysis on both algorithms and show a lower regret upper bound than other state-of-the-art graph-based bandit algorithms.

3.2 Related work

Graph-based bandit algorithms can be roughly categorized as: *i*) topology-based bandits, where the graph topology itself is exploited to improve learning performance, and *ii*) spectral bandits, where the user feature vectors θ are modelled as signals defined on the underlying graph, whose characteristics are then exploited in the graph spectral domain via tools provided by graph signal processing [63] to assist learning.

In topology-based bandits, the key intuition is to achieve dimensionality reduction in the user space by exploiting the graph topology. Specifically, users can be clustered based on the graph topology and a per-cluster feature vector can be learned, substantially reducing the dimensionality of the problem as opposed to the case in which one vector is learned per user. For example, [60] clusters users based on the connected components of the user graph, and [64] generalizes it to consider both the user graph and item graph. On the other hand, [65] makes use of community detection techniques on graphs to find user clusters. More broadly, in the spirit of dimensionality reduction, even without constructing an explicit user graph, the work in [66] proposes a distributed clustering algorithm while [67] applies *k-means* clustering to the user features. Despite the differences in the proposed techniques, these studies share two common drawbacks: 1) the learning performance depends on the clustering algorithm being used, which tends to be expensive for large-scale graphs; 2) learning a per-cluster (and not per-user) feature vector means ignoring the subtle difference between users within the same cluster. In short, clustering can reduce the dimensionality of the user space, but it does not necessarily preserve key users' characteristics. To achieve both goals simultaneously, there is a need for a proper mathematical framework able to incorporate the user relationship into

learning in a more direct way.

On the spectral bandit side, the strong assumption that users can be grouped into clusters is relaxed; users are assumed to be similar with their neighbors in the graph and such similarity is reflected by the weight of graph edges. [68] employs a graph Laplacian-regularized estimator, which promotes similar feature vectors for users connected in the graph. In their setting, however, each arm is selected by all users jointly. This work results in a network regret scaling with $\tilde{\mathcal{O}}(dn\sqrt{T'})$. [62] casts the same estimator as GMRF (Gaussian Markov Random Filed) and proposes a Thompson sampling algorithm, leading to a much simpler algorithmic implementation without a UCB evaluation. However, the regret bound remains $\tilde{\mathcal{O}}(dn\sqrt{T'})$. To the best of our knowledge, an efficient algorithm able to address the multi-user MAB problem with a sub-linear regret bound is still missing.

A proper bound and mathematical derivation of spectral MAB are provided in [69], which represents the payoffs of arms as smooth signals on a graph with the arms being the nodes. Specifically, the arm features x are modeled as eigenvectors of the graph Laplacian and the sparsity of such eigenvectors is exploited to reduce the dimensionality of x, to a so-called 'effective dimension' term $\tilde{d}$. This work shows an improved regret bound $\tilde{\mathcal{O}}(\tilde{d}\sqrt{T'})$ where $\tilde{d}$ is significant less than d in `LinUCB`[37]. While interesting, the proposed solution applies to the single-user with high-dimensional arm set. Whereas, in our setting, the dimensionality issue is caused by the large number of users, leading to a completely different mathematical problem.

Among these works on graph bandit, the one that is most similar to our work in terms of problem definition and proposed solution is [57]. In [57], the graph is exploited such that each user shares instantaneous payoff with neighbors, which is promoted by a Laplacian-regularized estimator. This implicitly imposes smoothness among the feature vectors of users, resulting in the estimate of similar feature vectors for users connected by edges with strong weights in the graph.

In this chapter, we propose the `GraphUCB` algorithm that builds on and improves `Gob.Lin` in a number of important ways:

- `Gob.Lin` employs the combinatorial Laplacian as a regularizer, whereas our algorithm `GraphUCB` makes use of the random-walk graph Laplacian. We prove theoretically that the combinatorial Laplacian results in a cumulative regret scaling with the number of users, which could be large. However, random-walk graph Laplacian overcomes this serious drawback and yields a sub-linear regret with the number of users.

- We propose a new UCB that leads to a cumulative regret scaling with $\tilde{\mathcal{O}}(\Psi nd\sqrt{T'})$ where $\Psi \in (0,1)$.

- The computational complexity of `Gob.Lin` is quadratic with respect to the number of users. Our simplified algorithm `GraphUCB-Local` scales linearly with the number of users, and at the same time enjoys the same regret bound as `GraphUCB`. This significantly improves the scalability of the proposed graph bandit algorithm.

3.3 Problem Formulation

We consider a linear bandit problem n users. We denote by $\mathscr{U}$ the user set with cardinality $|\mathscr{U}| = n$. Each user is described by a parameter vector $\theta \in \mathbb{R}^d$, with d being the dimension of both vectors. The affinity between users is encoded by an undirected and weighted graph $\mathscr{G} = (V,E)$, where $V = \{1,2,..,n\}$ represents the node set for n users and E represents the edge set. The graph $\mathscr{G}$ is known a priori and identified by its adjacency matrix $W \in \mathbb{R}^{n\times n}$, where $W_{ij} = W_{ji}$ captures the affinity between θ_i and θ_j. The combinatorial Laplacian of $\mathscr{G}$ is defined as $L = D - W$, where D is a diagonal matrix with $D_{ii} = \sum_{i=1}^{n} W_{ii}$. The symmetric normalized Laplacian is defined as $\tilde{L} = D^{-1/2}LD^{-1/2}$. In addition, the random-walk graph Laplacian is defined as $\mathscr{L} = D^{-1}L$.

In our setting, the unknown user features $\Theta = [\theta_1, \theta_2, ..., \theta_n]^T \in \mathbb{R}^{n\times d}$ are assumed to be smooth over $\mathscr{G}$. The smoothness of Θ over graph $\mathscr{G}$ can then be quantified using the Laplacian quadratic form of any of the three Laplacian defined above. In this chapter, we choose the random-walk graph Laplacian $\mathscr{L}$ because of its two unique properties $\mathscr{L}_{ii} = 1$ and $\sum_{j\neq i} \mathscr{L}_{ij} = -1$. The benefit of these properties

will be clear after the introduction of our proposed bandit algorithm. Mathematically, the Laplacian quadratic form based on $\mathscr{L}$ is (see Appendix A.8 for the derivation)

$$tr(\Theta^T \mathscr{L} \Theta) = \frac{1}{4} \sum_{k=1}^{d} \sum_{i \sim j} \left(\frac{W_{ij}}{D_{ii}} + \frac{W_{ji}}{D_{jj}} \right) \left(\Theta_{ik} - \Theta_{jk} \right)^2 \tag{3.1}$$

where Θ_{ik} is the (i,k)-th element of Θ. The more the graph $\mathscr{G}$ reflects the similarity between users correctly, the smaller the quadratic term $tr(\Theta^T \mathscr{L} \Theta)$. Specifically, $tr(\Theta^T \mathscr{L} \Theta)$ is small when Θ_{ik} and Θ_{jk} are similar given a large weight $\frac{W_{ij}}{D_{ii}} + \frac{W_{ji}}{D_{jj}}$.

Equipped with the above notation, we now introduce the multi-user bandit problem. At each time step $t = 1, \ldots, T$, the agent receives a user index i_t, which is assumed to be sampled uniformly from user set $\mathscr{U}$, and a set of arm vectors $\mathscr{A}_t = \{x_{1,t}, x_{2,t}, ..., x_{|\mathscr{A}_t|,t}\} \in \mathbb{R}^d$. No assumptions are posed on the generation of arm set $\mathscr{A}_t$ and its size $|\mathscr{A}_t|$, which can be generated arbitrarily dependent on the past selections made by the agent. The agent needs to select one arm from $\mathscr{A}_t$ to user i_t. Upon this selection, the agent observes a payoff y_t, which is assumed to be generated by noisy version of linear function. Namely,

$$y_t = x_t^T \theta_{i_t} + \eta_t \tag{3.2}$$

where the noise η_t is assumed to be σ-sub-Gaussian for any t.

The agent is informed about the graph $\mathscr{G}$. The bandit parameters Θ is unknown and needs to be inferred. The goal of the agent is to learn a selection strategy that minimizes the cumulative regret with respect to an optimal strategy, which always selects the optimal arm for each user. Formally, after a time horizon T, the cumulative (pseudo) regret is defined as:

$$R_T = \sum_{t=1}^{T} \left((x_t^*)^T \theta_{i_t} - x_t^T \theta_{i_t} \right) \tag{3.3}$$

where x_t and x_t^* are the arm selected by the agent and the optimal strategy at t, respectively. Note that the optimal choice depends on t as well as on the user i_t. For notation convenience in the rest of the paper, at each time step t, we use i to generally

refer to the user appeared and x_t to represent the feature vector of the arm selected.

Remark 3. *In previous version of this Chapter, the arm set $\mathscr{A}$ is fixed with $|\mathscr{A}| = m$ arms. Each arm is represented by a feature vector $x \in \mathbb{R}^d$. This setting is different from that in Gob.Lin [57] where x is sampled from $\mathbb{R}^d$ and no assumptions are made on the arm set $\mathscr{A}$.*

In the current version, we adapt the same setting as Gob.Lin [57]. As we notice that our proposed algorithms do not exploit the setting of a fixed and finite arm set. The proposed algorithm is completely applicable to the setting defined in Gob.Lin [57].

3.4 Laplacian-Regularized Estimator

To estimate the user parameter Θ at time t, we make use of the Laplacian-regularized estimator:

$$\hat{\Theta}_t = \arg\min_{\Theta \in \mathbb{R}^{n \times d}} \sum_{i=1}^{n} \sum_{\tau \in \mathscr{T}_{i,t}} (x_\tau^T \theta_i - y_\tau)^2 + \alpha \, tr(\Theta^T \mathscr{L} \Theta) \tag{3.4}$$

where θ_i is the i-th row of Θ, $\mathscr{T}_{i,t}$ is the set of time steps at which user i is served up to time t. x_τ is the feature of arm selected by the learner, y_τ is the payoff from user i at time τ, and α is the regularization parameter. Eq. (3.4) is convex and can be solved via convex optimization techniques. Specifically, it has a closed form solution [70]:

$$vec(\hat{\Theta}_t) = (\Phi_t \Phi_t^T + \alpha \mathscr{L} \otimes I)^{-1} \Phi_t Y_t \tag{3.5}$$

where $\otimes$ is the Kronecker product, $vec(\hat{\Theta}_t) \in \mathbb{R}^{nd}$ is the concatenation of the columns of $\hat{\Theta}_t$, $I \in \mathbb{R}^{d \times d}$ is the identity matrix, and $Y_t = [y_1, y_2, ..., y_t]^T \in \mathbb{R}^t$ is the collection of all payoffs. Finally, $\Phi_t = [\phi_1, \phi_2, ..., \phi_t] \in \mathbb{R}^{nd \times t}$, where $\phi_t \in \mathbb{R}^{nd}$, is a long sparse vector indicating that the arm with feature x_t is selected for user i. Formally,

$$\phi_t^T = (\underbrace{0, ..., 0}_{(i-1) \times d \text{ times}}, x_t^T, \underbrace{0, ...0}_{(n-i) \times d \text{ times}}). \tag{3.6}$$

While Eq. (3.4) provides the estimate of feature vectors of all users at t, i.e., $\hat{\Theta}_t$, the agent is interested in the estimation of each single-user feature vector $\hat{\theta}_{i,t}$. Mathematically, $\hat{\theta}_{i,t}$ can be obtained by decoupling users in Eq. (3.5). This however is highly complex due to the inversion $(\Phi_t\Phi_t^T + \alpha\mathscr{L}\otimes I)^{-1}$, which the agent needs to perform at each time step (we recall that the Laplacian is high-dimensional). We notice that the close-form solution of $\hat{\theta}_{i,t}$ can be closely approximated via a Taylor expansion of $(\Phi_t\Phi_t^T + \alpha\mathscr{L}\otimes I)^{-1}$, as stated in Lemma 1 and further commented and tested empirically in Appendix A.1.

Lemma 1. *$\hat{\Theta}_t$ is obtained from Eq.* (3.5)*, let $\hat{\theta}_{i,t}$ be the i-th row of $\hat{\Theta}_t$ which is the estimate of θ_i. $\hat{\theta}_{i,t}$ can be approximated by*[1]*:*

$$\hat{\theta}_{i,t} \approx A_{i,t}^{-1}X_{i,t}Y_{i,t} - \alpha A_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}A_{j,t}^{-1}X_{j,t}Y_{j,t} \tag{3.7}$$

where $A_{i,t} = \sum_{\tau\in\mathscr{T}_{i,t}} x_\tau x_\tau^T \in \mathbb{R}^{d\times d}$ is the Gram matrix related to the choices made by user i, $\mathscr{T}_{i,t}$ is the set of time at which user i is served up to time t, and $\mathscr{L}_{ij}$ is the (i,j)-th element in $\mathscr{L}$. $X_{i,t} \in \mathbb{R}^{d\times|\mathscr{T}_{i,t}|}$ is the collection of features of arms that are selected for user i up to time t with $\{x_\tau\}, \tau\in\mathscr{T}_{i,t}$ as columns. $Y_{i,t} \in \mathbb{R}^{|\mathscr{T}_{i,t}|}$ is the collection of payoffs associated with user i up to time t, whose elements are $\{y_\tau\}, \tau\in\mathscr{T}_{i,t}$.

Proof. See Appendix A.1. □

In solving bandit problems, the agent needs to estimate $\hat{\theta}_{i,t}$ and the uncertainty of this estimate as well. Learning such uncertainty translates into learning a confidence set on $\hat{\theta}_{i,t}$.

3.4.1 Construction of Confidence Set

To balance exploration and exploration, we need to quantify the uncertainty over the estimation of $\hat{\theta}_{i,t}$. This is possible by defining a confidence set around $\hat{\theta}_{i,t}$ based on Mahalanobis distance using its precision matrix, as commonly adopted in bandit

[1] $A_{i,t}$(and $A_{j,t}$) is not full-rank when $|\mathscr{T}_{i,t}| < d$. To guarantee inversion, in practice we set $A_{i,t} = \sum_{\tau\in\mathscr{T}_{i,t}} x_\tau x_\tau^T + \lambda I_d$ with $\lambda = 0.01$.

literature [71]. Let $\Lambda_{i,t} \in \mathbb{R}^{d\times d}$ be the precision matrix of $\hat{\theta}_{i,t}$, the confidence set is formally defined as

$$\mathbf{C}_{i,t} = \{\theta_{i,t} : ||\hat{\theta}_{i,t} - \theta_i||_{\Lambda_{i,t}} \leq \beta_{i,t}\} \tag{3.8}$$

where $\beta_{i,t}$ is the upper bound of $||\hat{\theta}_{i,t} - \theta_i||_{\Lambda_{i,t}}$ which is what we are interested in for the bandit algorithm. With this goal in mind, we seek an expression for $\Lambda_{i,t}$. Let $\Lambda_t \in \mathbb{R}^{nd\times nd}$ denote the precision matrix of $vec(\hat{\Theta}_t) \in \mathbb{R}^{nd}$, where $\Lambda_{i,t} \in \mathbb{R}^{d\times d}$ is the i-th block matrix along the diagonal of Λ_t. Defining the precision matrix of $vec(\hat{\Theta}_t)$ as

$$\Lambda_t = M_t A_t^{-1} M_t \tag{3.9}$$

with $A_t = \Phi_t \Phi_t^T$, $\mathscr{L}_\otimes = \mathscr{L} \otimes I$, and $M_t = A_t + \alpha \mathscr{L}_\otimes$, we have

$$\Lambda_{i,t} = A_{i,t} + 2\alpha \mathscr{L}_{ii} I + \alpha^2 \sum_{j=1}^{n} \mathscr{L}_{ij}^2 A_{j,t}^{-1} \tag{3.10}$$

where $A_{i,t}$ and $A_{j,t}$ are defined in Lemma 1, and $\mathscr{L}_{ij}$ is the (i,j)-th element in $\mathscr{L}$. A detailed derivation of Eq. (3.10) is presented in Appendix A.2 and Appendix A.3. Given Eq. (3.10), we can upper bound the size of the confidence set, which provides the value of $\beta_{i,t}$.

Lemma 2. *Let $V_{i,t} = A_{i,t} + \alpha \mathscr{L}_{ii} I$, and $I \in \mathbb{R}^{d\times d}$ the identity matrix. Given a scalar $\delta \in [0,1]$, and by defining $\Delta_i = \sum_{j=1}^{n} \mathscr{L}_{ij}\theta_j$, the size of the confidence set defined in Eq. (3.8) is upper bounded with probability $1-\delta$ by $\beta_{i,t}$:*

$$\beta_{i,t} = \sigma \sqrt{2\log \frac{|V_{i,t}|^{1/2}}{\delta |\alpha I|^{1/2}}} + \sqrt{\alpha}||\Delta_i||_2 \tag{3.11}$$

Proof. See Appendix A.4 □

Remark 4. *Note that the random-walk graph Laplacian $\mathscr{L} = D^{-1}L$ has two properties: $\mathscr{L}_{ii} = 1$ and $\sum_{j\neq i} \mathscr{L}_{ij} = -1$. Then, The term $\Delta_i = \sum_{j=1}^{n} \mathscr{L}_{ij}\theta_j$ can be rewritten as follows:*

$$\Delta_i = \theta_i - \sum_{j\neq i} -\mathscr{L}_{ji}\theta_j \tag{3.12}$$

Denote $-\mathscr{L}_{ij}$ as w_{ij}, the second term $\sum_{j\neq i} -\mathscr{L}_{ji}\theta_j$ will be $\sum_{j\neq i} w_{ij}\theta_j$. As

$\sum_{j\neq i} -\mathscr{L}_{i,j} = 1$, *we have* $\sum_{j\neq i} w_{ij} = 1$. *Hence,* $\sum_{j\neq i} w_{ij}\theta_j$ *is the convex combination of* $\theta_j, j \neq i$. *i.e., the weighted average of neighbors.*

If we further denote $\tilde{\theta}_i = \sum_{j\neq i} w_{ij}\theta_j$. *The term* $\Delta_i = \theta_i - \tilde{\theta}_i$ *essentially measures the difference between* θ_i *and the weighted average of it neighbors* $\theta_j, j \neq i$ *where weights are corresponds to* $\mathscr{L}_{ij}$ *which encodes the similarity between* i *and* j. *In the following, we can bound the term* $||\Delta_i||$*:*

$$||\Delta_i|| = ||\theta_i - \sum_{j\neq} w_{ij}\theta_j|| \leq ||\theta_i|| + ||\sum_{j\neq} w_{ij}\theta_j|| \tag{3.13}$$

Assume $||\theta|| \leq 1$ *and thanks to* $\sum_{j\neq} w_{ij} = 1$ *(a.k.a,* $\sum_{j\neq i} \mathscr{L}_{ij} = -1$*) we have*

$$||\theta_i|| + ||\sum_{j\neq} w_{ij}\theta_j|| \leq 2 \tag{3.14}$$

Finally, notice that these two properties: $\mathscr{L}_{ii} = 1$ *and* $\sum_{j\neq i} \mathscr{L}_{ij} = -1$ *are not observed in either combinatorial Laplacian or Normalized Laplacian. The above upper bound of* $||\Delta_i||$ *can not be obtained for either combinatorial Laplacian or Normalized Laplacian..*

To show the effect of Δ_i on the confidence bound, we consider two extreme cases: *a)* an empty graph[2] where $\mathscr{L}_{ii} = 1$ and $\mathscr{L}_{ij} = 0$. In this case, $\Delta_i = \theta_i$, which recovers the confidence set used in `LinUCB` [37]; *b)* a fully connected graph with $W_{ij} = 1$ where $\mathscr{L}_{ii} = 1$, $\mathscr{L}_{ij} = \frac{1}{n-1}$ and $\theta_i = \theta_j$. In this case, $\Delta_i = \theta_i - \frac{n-1}{n-1}\theta_i = 0$ leading to a much lower bound than the one in *a)*. In between, Δ_i depends on the similarity between θ_i and its neighbors $\theta_j, j \neq i$. In general, the smoother the signal is on the graph, the lower the $||\Delta_i||_2$. This has been empirically shown in Fig 3.1(a), where we depict $||\Delta_i||_2$ as a function of the level of smoothness quantified by $tr(\Theta^T \mathscr{L} \Theta)$.

3.5 Algorithms

We now introduce the proposed `GraphUCB` bandit algorithm, sketched in Algorithm 12. `GraphUCB` leverages the Laplacian-regularized estimator Eq. (3.4) and

[2]For isolated node, we set $\mathscr{L}_{ii} = 1$.

Algorithm 12: GraphUCB

Input : $\alpha, T, \mathscr{L}, \delta$

Initialization : For any $i \in \{1,2,...,n\}$ $\hat{\theta}_{0,i} = 0 \in \mathbb{R}^d$, $\Lambda_{0,i} = 0 \in \mathbb{R}^{d\times d}$, $A_{0,i} = 0 \in \mathbb{R}^{d\times d}$, $\beta_{i,t} = 0$.

for $t \in [1,T]$ **do**

User index i is selected

1. $A_{i,t} \leftarrow A_{i,t-1} + x_{t-1}x_{t-1}^T$.
2. $A_{j,t} \leftarrow A_{j,t-1}, \forall j \neq i$.
3. Select x_t via Eq. (3.15) where $\beta_{i,t}$ is defined in Eq. (3.11)
4. Receive the payoff y_t.
5. Update $\hat{\Theta}_t$ via Eq. (3.4).
6. Update $\Lambda_{i,t}$ via Eq. (3.10).

end

the arm index defined in Eq. (3.15).

$$x_t = \arg\max_{x\in\mathscr{A}_t} \left(x^T \hat{\theta}_{i,t} + \beta_{i,t} ||x||_{\Lambda_{i,t}^{-1}} \right) \tag{3.15}$$

Formally, at each time t, a user index i is selected randomly from the user set $\mathscr{U}$. It selects the arm x_t from the arm set $\mathscr{A}_t$ following Eq. (3.15). Upon receiving the instantaneous payoff y_t, it updates the features of all users $\hat{\Theta}_t$ by solving Eq. (3.4) and $\Lambda_{i,t}$ according to Eq. (3.10). In practice, Δ_i is replaced by its empirical counterpart $\hat{\Delta}_i = \sum_{i=1}^n \mathscr{L}_{ij}\hat{\theta}_{i,t}$ where $\hat{\theta}_{i,t}$ is the i-th row of $\hat{\Theta}_t$.

One limitation of GraphUCB is its high computational complexity. Specifically, in solving Eq. (3.4), the running time is dominated by the inversion of $(\Phi_t\Phi_t^T + \alpha\mathscr{L}\otimes I)^{-1} \in \mathbb{R}^{nd\times nd}$, which could be impractical when the user number n is large. As we known, the proved lower bound on matrix inversion computational complexity is, given by [72], $\mathscr{O}((nd)^2\log(nd))$ for matrix with size $nd \times nd$.

We notice that only one user is selected at each time t. Thus, it suffices to only update $\hat{\theta}_{i,t}$ (i.e., a local rather than global update). Therefore, we propose to make use of Lemma 1 instead of Eq. (3.4) to update $\hat{\theta}_{i,t}$. This leads to a significant

reduction in terms of computational complexity.

Leveraging Lemma 1, GraphUCB-Local serves as a simplified version of GraphUCB. The only difference lies in the number of users updated at each round. GraphUCB updates all users $\hat{\Theta}_t$ via Eq. (3.4) (closed-form solution). By contrast, GraphUCB-Local updates one user $\hat{\theta}_{i,t}$. Pseudocode of GraphUCB-Local is presented in Appendix A.5.

3.6 Analysis

Before providing the regret analysis of the proposed algorithms, we first define

$$\Psi_{i,T} = \frac{\sum_{\tau \in \mathscr{T}_{i,T}} ||x_\tau||^2_{\Lambda^{-1}_{i,\tau}}}{\sum_{\tau \in \mathscr{T}_{i,T}} ||x_\tau||^2_{V^{-1}_{i,\tau}}} \tag{3.16}$$

where $\mathscr{T}_{i,T}$ is the set of time steps in which user i is served up to time T, $A_{i,\tau} = \sum_{\ell \in \mathscr{T}_{i,\tau}} x_\ell x_\ell^T$, $V_{i,\tau} = A_{i,\tau} + \alpha \mathscr{L}_{ii} I$ and $\Lambda_{i,\tau}$ is defined as in Eq. (3.10). Note that $||x_\tau||^2_{\Lambda^{-1}_{i,\tau}}$ and $||x_\tau||^2_{V^{-1}_{i,\tau}}$ quantify the uncertainty of predicted payoff $\hat{y}_\tau = \hat{\theta}^T_{i,\tau} x_\tau$ in the cases where the graph structured is exploited or ignored, respectively.

Lemma 3. *Let $\Psi_{i,T}$ be as defined in Eq.* (3.16) *and $||x_\tau||_2 \leq 1$ for any $\tau \leq T$, then*

$$\Psi_{i,T} \in (0,1)$$

and as $T \to \infty$, $\Psi_{i,T} \to 1$. This implies

$$\sum_{\tau \in \mathscr{T}_{i,T}} ||x_\tau||^2_{\Lambda^{-1}_{i,\tau}} \leq \sum_{\tau \in \mathscr{T}_{i,T}} ||x_\tau||^2_{V^{-1}_{i,\tau}}.$$

See proof in Appendix A.6.

This Lemma highlights the benefit of taking into account the graph structure, showing that the uncertainty of $\hat{y}_\tau$ is reduced when the graph structure is exploited. It also shows that this effect diminishes with time: In Fig. 3.1(b), we see that as more data are collected, the graph-based estimator approaches the estimator in which users parameters are estimated independently (when $\Psi_{i,T} \to 1$).

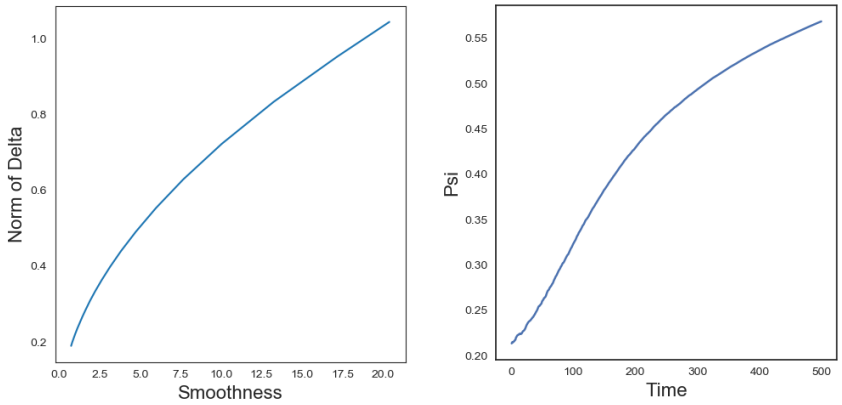


Figure 3.1: (a) $||\Delta_i||_2$ vs. smoothness ($tr(\Theta^T \mathscr{L} \Theta)$), (b) $\Psi_{i,T}$ vs. time.

3.6.1 Regret Upper Bound

We now present the cumulative regret upper bounds satisfied by both `GraphUCB` and `GraphUCB-Local`.

Theorem 10. *$\Psi_{i,T}$ is defined in Eq. (3.16), $\Lambda_{i,T}$ defined in Eq. (3.10) and $\Delta_i = \sum_{j=1}^{n} \mathscr{L}_{ij}\theta_j$. Without loss of generality, assume $||\theta_i||_2 \leq 1$ for any $i \in \{1,2,...,n\}$ and $||x_\tau||_2 \leq 1$ for any $\tau \leq T$. Then, for $\delta \in [0,1]$, for any user $i \in \{1,2,...,n\}$ the cumulative regret over time horizon T satisfies the following upper bound with probability $1-\delta$*

$$R_{i,T} = \sum_{\tau \in \mathscr{T}_{i,T}} r_\tau = \mathscr{O}\Bigg(\Big(\sqrt{d\log(|\mathscr{T}_{i,T}|)} + \sqrt{\alpha}||\Delta_i||_2\Big) \times \Psi_{i,T}\sqrt{d|\mathscr{T}_{i,T}|\log(|\mathscr{T}_{i,T}|)}\Bigg) \tag{3.17}$$

Assuming that users are served uniformly up to time horizon T, i.e., $|\mathscr{T}_{i,T}| = T/n$, the network regret (the total cumulative regret experienced by all users) satisfies the following upper bound with probability $1-\delta$:

$$\begin{aligned} R_T &= \sum_{i=1}^{n} R_{i,T} = \sum_{i=1}^{n} \tilde{\mathscr{O}}\Big(\Psi_{i,T} d\sqrt{T/n}\Big) \\ &= \tilde{\mathscr{O}}\Big(d\sqrt{Tn}\max_{i \in \mathscr{U}} \Psi_{i,T}\Big) \end{aligned} \tag{3.18}$$

Proof. See proof in Appendix A.7 □

Remark 5. *Both* GraphUCB *and* GraphUCB-Local *satisfy Theorem 10.*

The above remark is deduced from the observation that GraphUCB and GraphUCB-Local differ only in the estimation of $\hat{\theta}_{i,t}$. However, the regret bound is derived based on the ground-truth θ_i, which is the same in both algorithms.

3.6.2 Comparison with LinUCB and Gob.Lin

Under the same setting, the single-user regret upper bound of LinUCB [35] is

$$R_{i,T} = \mathcal{O}\Big((\sqrt{d\log(|\mathcal{T}_{i,T}|)} + \sqrt{\alpha}||\theta_i||_2) \times \sqrt{d|\mathcal{T}_{i,T}|\log(|\mathcal{T}_{i,T}|)} \Big) \tag{3.19}$$

Since $||\Delta_i||_2 \leq ||\theta_i||_2$ and $\Psi_{i,T} \in (0,1)$ (Lemma 2 and Lemma 3), GraphUCB (and GraphUCB-Local) leads to a lower regret −Eq. (3.17)− than LinUCB −Eq. (3.19).

For user i and item x, define the compound descriptor of the pair (i,x) as $\phi_i(x)^T \in \mathbb{R}^{nd}$:

$$\phi_i(x) = (0,...,0,x^T,0,...,0) \tag{3.20}$$

Let $\mathscr{L}$ be the Laplacian matrix of graph $\mathscr{G}$, define $A = I_n + \mathscr{L}$ and $A_\otimes = A \otimes I_d$ is the Kronecker product of matrix A and I_d. Next, define

$$\tilde{\phi}_i(x) = A_\otimes^{-1/2}\phi_i(x) \tag{3.21}$$

Then

$$M_T = \sum_{t=1}^{T} \tilde{\phi}_t(x_t)\tilde{\phi}_t(x_t)^T \tag{3.22}$$

The cumulative regret upper bound of Gob.Lin in [57] is shown as

$$R_T = 2\sqrt{T(\sigma^2 \ln \frac{|M_t|}{\delta} + L(\theta)) \ln |M_t|} \tag{3.23}$$

where

$$L(\theta) = \sum_{i=1}^{n} ||\theta_i||_2 + \sum_{(i,j)\in E} ||\theta_i - \theta_j||_2 \tag{3.24}$$

As stated in Gob.Lin [57], we have

$$\ln|M_T| \leq dn\ln(1+2T/(dn(n+1))) \tag{3.25}$$

Then,

$$R_T \leq \tilde{\mathcal{O}}(nd\sqrt{T}+\sqrt{L(\theta)Tnd}) \tag{3.26}$$

Next, we need to bound $L(\theta)$.
The author [57] claimed that this term $\sum_{(i,j)\in E}||\theta_i-\theta_j||_2$ is small under the working assumption that connected nodes share similar θ.

Let's take this assumption to the extreme case that $\theta_i=\theta_j$ if they are connected. Then, $||\theta_i-\theta_j||_2=0$ and assume $||\theta_i||_2\leq 1$. We have

$$L(\theta)\leq n \tag{3.27}$$

The regret can be bounded as

$$R_T \leq \tilde{\mathcal{O}}(nd\sqrt{T}+\sqrt{Tn^2d}) \leq \tilde{\mathcal{O}}(nd\sqrt{T}) \tag{3.28}$$

The cumulative regret achieved by GraphUCB in Eq. (3.18) is less than that in Eq. (3.28) by an order of $\tilde{\mathcal{O}}(\sqrt{n})$.

3.7 Experiment Results

We evaluate the proposed algorithms and compare them to LinUCB (no graph information exploited in the bandit), Gob.Lin (graph exploited in the features estimation) and CLUB (graph exploited to cluster users). All results reported are averaged across 20 runs. In all experiments, we set confidence probability parameter $\delta=0.01$, noise variance $\sigma=0.01$, and regularization parameter $\alpha=1$. For Gob.Lin, we use $\beta_{i,t}=\lambda\sqrt{\log(t+1)}$, and λ is set using the best value in range $[0,1]$. For CLUB, the edge deletion parameter α_2 is tuned to its best value.

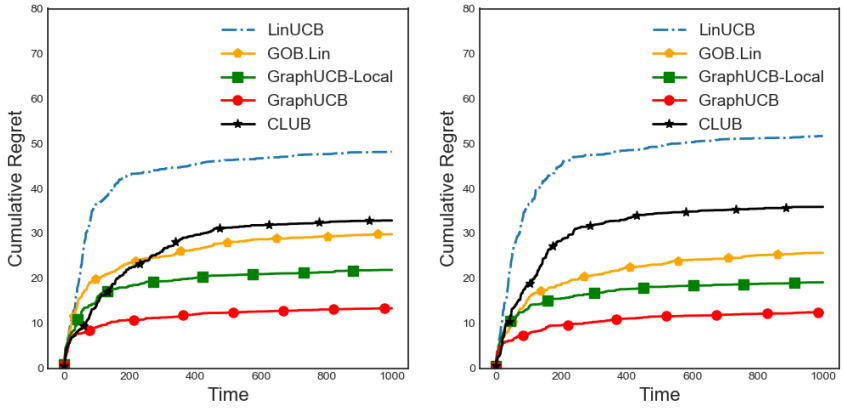


Figure 3.2: Cumulative regret vs. time for different type of graphs (ER and RBF) consistently generated with the same level of smoothness and sparsity between graphs.

3.7.1 Experiments on Synthetic Data

In the synthetic simulations, we first generate a graph $\mathscr{G}$ and then generate a smooth Θ via Eq. (3.29) which is proposed in [73]:

$$\Theta = \arg\min_{\Theta \in \mathbb{R}^{n \times d}} ||\Theta - \Theta_0||_F^2 + \gamma tr(\Theta^T \mathscr{L} \Theta) \tag{3.29}$$

where $\Theta_0 \in \mathbb{R}^{n \times d}$ is a randomly initialized matrix, and $\mathscr{L}$ is the random-walk graph Laplacian of $\mathscr{G}$. The second term in Eq. (3.29) promotes the smoothness of Θ: the larger the γ, the smoother the Θ over the graph[3]. In all experiments, $n = 20, d = 5$. To simulate $\mathscr{G}$, we follow two random graph models commonly used in the network science community: 1) Radial basis function (*RBF*) model, a weighted fully connected graph, with edge weights $W_{ij} = exp(-\rho||\theta_i - \theta_j||^2)$; 2) Erdős Rényi (ER) model, an unweighted graph, in which each edge is generated independently and randomly with probability p.

In Fig 3.2, we depict the cumulative per-user regret as a function of time for both RBF and ER graphs for both our proposed algorithms and competitors. The regret is averaged over all users and over all runs. Under all graph models, `GraphUCB` outperforms its competitors consistently with a large margin. Also `GraphUCB-Local` consistently outperform competitor algorithms, with however

[3]The regularization parameter γ in Eq. (3.29) is used to generate a smooth function in the synthetic settings, while the parameter α in Eq. (3.4) is used in the bandit algorithm to infer the smooth prior when estimating user features.

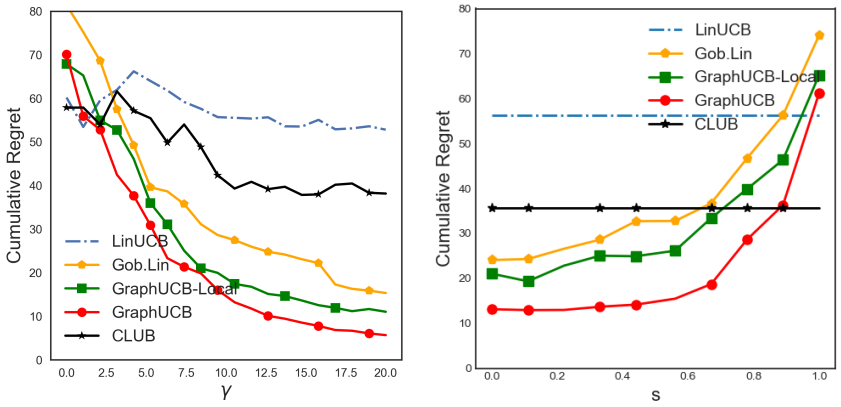


Figure 3.3: Cumulative regret for RBF graphs with different level of smoothness (a) and sparsity (b).

slightly degradated performance with respect to `GraphUCB`. This is due to the approximation introduced by Eq. (3.7). `Gob.Lin` is a close runner since it is also based on the Laplacian regularized estimator, but it performs worse than the proposed algorithms, as already explained in the previous section. `CLUB` performs relative poor since there is no clear clusters in the graph. Nevertheless, it still outperforms `LinUCB` by grouping users into clusters in the early stage which speeds up the learning process. It is worth noting that the two subfigures depict the same algorithm for two different graph models (RBF and ER) with the same level of smoothness and sparsity. The trend of the cumulative regret is the same, highlighting that the algorithm is not affected by the graph model. This behaviour is reinforced in Appendix A.10 where we provide further results.

We are now interested in evaluating the performance of the proposed algorithms against different graph topologies, by varying signal smoothness and sparsity of graph (edge density) as follows

Smoothness [γ]: We first generate a *RBF* graph. To control the smoothness, we vary $\gamma \in [0, 10]$.

Sparsity [s]: We first generate a *RBF* graph, then generate a smooth Θ via Eq. (3.29). To control the sparsity, we set a threshold $s \in [0, 1]$ on edge weights W_{ij} such that W_{ij} less than s are removed.

Fig 3.3, depicts the cumulative regret for different level of smoothness and

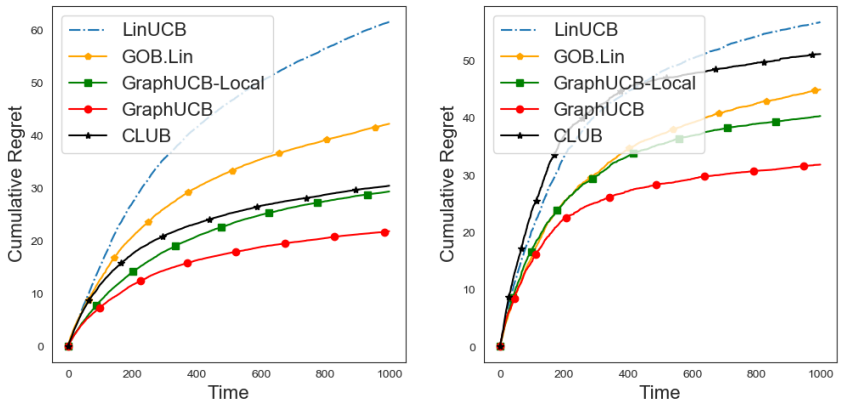


Figure 3.4: Performance on Real-World data.

sparsity of $\mathscr{G}$. `GraphUCB` and `GraphUCB-Local` show similar patterns (with `GraphUCB-Local` leading to higher regret due to the already commented approximation): *(i)* the smoother Θ the lower is regret, which is consistent with the Laplacian-regualrized estimator Eq. (3.4); *(ii)* denser graphs lead to lower regret since more connectivity provides more graph information which speeds up the learning process.

3.7.2 Experiments on Real-World Data

We then carry out experiments on two real-world datasets : **Movielens** [74] and **Netflix** [75]. We follow the data pre-processing steps in [69], described in details in Appendix A.10.

The cumulative regret over time is depicts in Fig 3.4 for both datasets. Both `GraphUCB` and `GraphUCB-Local` outperform baseline algorithms in all cases. Similarly to the synthetic experiments, `LinUCB` performs poorly, while `GOB` shows a regret behaviour more similar to the proposed algorithms. In the case of **Movielens**, `CLUB` outperforms `GOB`. A close inspection of the data reveals that ratings provided by all users are highly concentrated. It means most users like a few sets of movies. This is a good model for the clustering algorithm implemented in `CLUB`, hence the gain.

3.8 Conclusion

In this chapter, we propose a graph-based bandit algorithm `GraphUCB` and its scalable version `GraphUCB-Local`, both of which outperform the state-of-art bandit algorithms in terms of cumulative regret. On the theoretical side, we introduce a novel UCB embedding the graph structure in a natural way and show clearly that exploring the graph prior could reduce the cumulative regret. We demonstrate that the graph structure helps reduce the size of confidence set of the estimation of user features and the uncertainty of predicted payoff. As for future research directions, one possibility is to relax the assumption that the user graph is available and infer the graph from the data, ideally in a dynamic fashion.

Chapter 4

Error Analysis on Graph Laplacian Regularized Estimator

In the multi-task bandit problem introduced in the previous chapter, we utilize the graph Laplacian regularized estimator to estimate the unknown bandit coefficients. This type of estimator has recently received considerable attention due to its capability in incorporating underlying topological graph structure into the learning process. While the estimation problem can be solved efficiently by state-of-the-art optimization techniques, its statistical consistency properties have been largely overlooked. In this chapter, we develop a non-asymptotic bound of estimation error under the classical statistical setting, where sample size is larger than the ambient dimension of the latent variables. This bound illustrates theoretically the impact of the alignment between the data and the graph structure as well as the graph spectrum on the estimation accuracy. It also provides theoretical evidence of the advantage, in terms of convergence rate, of the graph Laplacian regularized estimator over classical ones (that ignore the graph structure) in case of a smoothness prior. Finally, we provide empirical results of the estimation error to corroborate the theoretical analysis.

4.1 Introduction

The aim of representation learning is usually to estimate the latent variables (i.e., the design matrix) Θ and the coefficient matrix X that explain the intrinsic characteristics

of the observations. This is usually solved by iterating between estimation of the design matrix and estimation of the coefficient matrix. In this work, we focus on the estimation of the design matrix Θ in $Y = \Theta X$, where the coefficient matrix X and the observations Y are assumed to be known, corresponding to one estimation step in representation learning. While there exists error analyses on ridge regression based estimator for the coefficient matrix, there has been little effort devoted to such analysis on the estimator for the design matrix. To provide a better understanding of the uncertainty of estimation in representation learning, it is therefore essential to derive a theoretical error analysis on the estimator for the design matrix, which is the goal of this chapter.

Estimating the design matrix given the observation and the coefficient matrix is a learning problem that appears naturally and frequently in applications across many fields such as image denoising, compress sensing, dictionary learning, and collaborative filtering (CF). In CF [76], the goal is to estimate users' preferences Θ through item features X based on users' responses Y (e.g., ratings). Dictionary learning [73] also shares a similar formulation, where Θ and X are referred to dictionary atoms and coding coefficient matrix. In this case, our model corresponds to the case when the coding coefficient matrix X is known. Another closely related application is image denoising [77], where Θ corresponds to a collection of basis functions (*e.g.*, wavelets, cosine waves) modeling the image signal as a linear combination of these basis functions with X being the coefficients. Due to the wide applicability of representation learning problems, the development of estimators for representation learning has recently received a substantial attention. A key type of estimators is regularized least squares estimators, which poses structural constraints on the unknown coefficient matrix X. For example, the Lasso formulation [78] poses a sparsity constraint on the number of non-zero entry in Θ. The work of [79] introduces nuclear/trace norm based regularizers aiming to find a low rank solution while fitting the data. Other examples include [80] and [81] that are based on various structural constraints. Compared to estimating the coefficient matrix, however, much less work has been devoted to the estimation of the design matrix, and in particular

error analysis associated with the uncertainty in such estimation.

In this chapter, we study the estimation of the design matrix in representation learning, where the estimator involves a graph based regularizer. Such a regularizer helps incorporating the underlying geometric structure of the data into the learning process. For example, in recommender systems [82], one might get access to users' social network. Incorporating such information may lead to a better understanding of users' preference (Θ), which may in turn improve the recommendation performance (Y). We focus on the graph Laplacian based regularizer, which has been widely adopted in the literature [83, 76, 84, 73] thanks to its mathematical regularity (*e.g.,* convexity and differentiability). In particular, the least squares estimation regularized by $tr(\Theta^T L\Theta)$ is a convex program, and Θ can be computed by the well-known Bartels-Stewart algorithm [85] or more efficient algorithms developed recently [76, 86]. Rather than focusing on solving methods for the estimation problem, our goal is rather to study its statistical consistency guarantee, and to gain insights about the impact of L on its convergence rate. We aim to provide a bound on the estimation error, $\triangle = \hat{\Theta} - \Theta^*$, defined as the difference between any estimation $\hat{\Theta}$ and the unknown ground-truth latent variables Θ^*. More formally, we derive a non-asymptotic upper bound on $||\triangle||_F = ||\hat{\Theta} - \Theta^*||_F$ that holds with high probability, where $||\cdot||_F$ is the Frobenius norm. To the best of our knowledge, this theoretical analysis is absent in the literature.

The main contributions of the chapter are as follows:

- we obtain an error bound for the graph Laplacian regularized estimator, which illustrates the effect of graph structure as well as its alignment with data on estimation accuracy;
- we show the impact of the graph spectrum (*i.e.,* eigenvalues of the graph Laplacian) on the estimation accuracy;
- we compare with the classical ridge estimator to prove theoretically advantages brought by incorporating graph structure into the estimator and we validate our claims empirically by simulations results.

In summary, we study analytically the estimation of the design matrix to understand the associated uncertainty. This is a key component to understand the effectiveness of representation learning algorithms, and to understand the effect of the topological structure $\mathscr{G}$ on the estimation uncertainty.

The reminder of the paper is organized as follows. Section 2 presents related work and in particular existing studies on the theoretical analysis of representation learning. Section 3 introduces the basic definition related to the graph Laplacian regularized estimator and formulates the estimation problem. Section 4 and 5 present the main results of error analysis on the estimator and the corresponding proofs, respectively. Section 6 shows empirical results and Section 7 summarizes the paper.

Notations: Let $A = [A_{ij}] \in \mathbb{R}^{n\times k}$ and $B = [B_{ij}] \in \mathbb{R}^{n\times k}$ be two $n \times m$ matrices. The scalar product $\langle A, B\rangle = tr(A^T B)$, where $tr(\cdot)$ is the trace operator. The Frobenius norm is defined as $||A||_F = (\sum_{ij} A_{ij}^2)^{\frac{1}{2}}$ and the infinity norm as $||A||_\infty = \max_{ij} |A_{ij}|$. The nuclear norm is defined as $||A||_* = tr(\sqrt{A^T A})$, with A^T being the transpose of A. Let $x = [x_1, ..., x_d]^T \in \mathbb{R}^d$ be a d-dimensional vector, where its L_1 and L_2 norms are defined as $||x||_1 = \sum_{i=1}^d |x_i|$ and $||x||_2 = (\sum_{i=1}^d x_i^2)^{\frac{1}{2}}$, respectively.

4.2 Related work

In many applications, data come with an underlying geometric structure, typically in the form of a graph, which should be considered in the learning process. There has been an increasing amount of interest in representation learning, where topological graph structures are embedded into estimators to promote desirable properties of the solution. For example, [87] introduces a measure of smoothness of the data with respect to a graph topology, in the form of the so-called Laplacian quadratic form $tr(\Theta^T L \Theta)$. Employing this term as a regularizer in the estimators thus finds a solution $\hat{\Theta}$ that is smooth on the graph. Alternatively, [88] introduces total variation and graph total variation estimation. Following works show empirically the effectiveness of such regularizers [89, 76, 90]. Other graph-based regularizers include edge Lasso [91], network Lasso [92], and graph trend filtering [93].

In this work, we study a graph Laplacian regularized least squares estimator

for the design matrix Θ. To the best of our knowledge, there is no prior work on the theoretical understanding of a consistency guarantee of this estimator, either in high-dimensional ($n < p$, or $n \ll p$) or in classical statistical setting ($n \geq p$). Nevertheless, there has been a few theoretical studies on the graph regularized estimators. For example, total variation regularized estimators [88] are proven to be similar to graph Laplacian regularized estimators [89] and promote piece-wise constant solutions, while the Laplacian regularized estimators lead to piece-wise smooth solutions. The work of [94] provides optimal rate analysis of the total variation regularized estimator. The work of [76] derives statistical consistency guarantees of the Laplacian regularized estimator in the application of collaborative filtering. A key difference from their paper is that, in our work, we consider the unknown coefficient matrix Θ as the graph signals.

Another difference is that, they derive a bound on the prediction error in the measurements $||\hat{Y} - Y^*||_F$, while we develop a bound on the estimation error in the design matrix $||\hat{\Theta} - \Theta^*||_F$. Finally, the work of [95] provides theoretical consistency guarantees on a graph regularized estimator in linear regression, which is formulated using a combination of graph Laplacian, total variation, and edge Lasso. Instead, we focus on such theoretical properties using only graph Laplacian based regularizer.

4.3 Graph Laplacian regularized estimator

In this section, we introduce our estimation problem in representation learning. We first provide some background on signals on graphs, and then introduce the graph Laplacian regularized estimator and the associated estimation problem.

4.3.1 Graph Laplacian and graph signal

Consider a weighted and undirected graph $\mathcal{G} = (V, E, W)$ of m vertices, where V is the finite set of vertices and E the finite set of edges, and $W = [W_{ij}] \in \mathbb{R}^{m \times m}$ denotes the weighted adjacency matrix. The entry W_{ij} represents the edge weight between vertex v_i and v_j. $W_{ij} = 0$ if v_i, v_j are not directly connected, and $W_{ij} > 0$ if connected. Moreover, $W_{ij} = W_{ji}$ for weighted undirected graph. The graph degree matrix is $D = [D_{ii}] \in \mathbb{R}^{m \times m}$, where $D_{ii} = \sum_j W_{ij}$ represents the degree of vertex v_i.

The combinatorial graph Laplacian L is defined to be $L = D - W$.

A graph signal is referred as a function $f : V \to \mathbb{R}^m$ that assigns a real value to each graph vertex. In this paper, we consider smooth signals over graphs. With the Laplacian matrix L, the smoothness of signal f over graph $\mathscr{G}$ can be quantified as a quadratic form of L [87]:

$$f^T L f = \frac{1}{2} \sum_{i \sim j} W_{ij} (f(i) - f(j))^2 \tag{4.1}$$

which is a weighted sum of the squared signal difference between connected vertices, where weights are corresponding edge weights.

4.3.2 The estimation problem

We now state the estimation problem to be addressed in this paper, which is the estimation of the design matrix Θ, in a linear model with a graph Laplacian regularizer. We first introduce the linear model, and then describe the graph Laplacian regularized estimator.

We consider the problem of representation learning under a general noisy setting. The aim of this problem is to learn the design matrix $\Theta \in \mathbb{R}^{m \times k}$ that explains the observations $Y \in \mathbb{R}^{m \times n}$ with the coefficient matrix $X \in \mathbb{R}^{k \times n}$. Formally, we consider a linear model in the form of

$$Y = \Theta X + \Omega \tag{4.2}$$

where $Y \in \mathbb{R}^{m \times n}$ denotes the observation matrix, $X \in \mathbb{R}^{k \times n}$ is the coefficient matrix following the standard regularity assumption that columns are independent, and $\Theta \in \mathbb{R}^{m \times k}$ represents the design matrix. Let us denote the noise matrix by $\Omega = [\Omega_{ij}] \in \mathbb{R}^{m \times n}$ with entries $\Omega_{ij} \sim \mathscr{N}(0, \sigma^2)$ being Gaussian noise. We now consider an estimator arising frequently in the literature, which assumes Θ to be smooth with respect to an underlying graph structure $\mathscr{G}$.

To estimate Θ under the prior of smoothness on $\mathscr{G}$, we consider a graph Laplacian regularized estimator. Formally,

$$\hat{\Theta} = \arg \min_{\Theta \in \mathbb{R}^{m \times k}} \frac{1}{2n} ||Y - \Theta X||_F^2 + \alpha \, tr(\Theta^T L \Theta) \tag{4.3}$$

where $\alpha \geq 0$ is a regularization parameter. The regularization term $tr(\Theta^T L\Theta)$ is known to promote smoothness of Θ with respect to the underlying graph $\mathscr{G}$. From the perspective of statistical models, the regularizer in Eq. (4.3) assumes that Θ follows a degenerate multivariate Gaussian distribution, where the graph Laplacian L acts as the precision matrix. Such an estimator has been adopted in applications such as graph-structured matrix factorization [89, 83] and signal denoising [96].

It is instructive to compare Eq. (4.3) with a simpler estimator which applies the standard ridge estimator to the representation learning problem Eq. (4.2), solving

$$\hat{\Theta} = \arg\min_{\Theta \in \mathbb{R}^{m\times k}} \frac{1}{2n}||Y - \Theta X||_F^2 + \alpha\, tr(\Theta^T I_m \Theta) \tag{4.4}$$

where I_m is a $m \times m$ identity matrix. It is worth noting that the estimator Eq. (4.4) is a degenerative version of Eq. (4.3) when $L = I_m$, *i.e.,* when the graph structure is ignored. Therefore, the estimator in Eq. (4.4) is a desirable baseline for understanding the property of the graph Laplacian regularized estimator in Eq. (4.3). In the following sections, we provide a deep analysis on Eq. (4.3) as well as a comparison with Eq. (4.4).

4.4 Estimation error analysis

The central focus of this chapter is the error analysis on $\hat{\Theta}$, i.e., providing a bound on the estimation error $\triangle = \hat{\Theta} - \Theta^*$, where Θ^* is the unknown ground-truth latent variables matrix. To derive this bound, we generally follow the unified analysis framework of [97], properly adjusted to our problem. We first develop some key notations of the graph Laplacian regularized estimator in Eq. (4.3). Next, we describe an important ingredient of our main result: the *strong convexity* condition. Finally, we present our main results and their interpretation.

4.4.1 Key notations

The graph Laplacian regularized estimator Eq. (4.3) can be rewritten into the following form:

$$\hat{\Theta} = \arg\min_{\Theta \in \mathbb{R}^{m\times k}} \mathscr{L}(\Theta) + \alpha \mathscr{R}(\Theta) \tag{4.5}$$

where $\mathscr{L}(\Theta) = \frac{1}{2n}||Y - \Theta X|_F^2$ is the loss function assigning a cost to any $\Theta \in \mathbb{R}^{m\times k}$ given a pair of $\{X, Y\}$, and $\mathscr{R}(\Theta) = tr(\Theta^T L\Theta)$ denotes the Laplacian regularizer. The first order Taylor expansion of the loss function at Θ^* in the direction $\triangle = \hat{\Theta} - \Theta^*$ is expressed as

$$\mathscr{L}(\Theta^* + \triangle) = \mathscr{L}(\Theta^*) + \langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle \tag{4.6}$$

thus, the error of the first Taylor expansion $\delta\mathscr{L}(\Theta^*)$ is defined as

$$\delta\mathscr{L}(\Theta^*) = \mathscr{L}(\Theta^* + \triangle) - \mathscr{L}(\Theta^*) - \langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle \tag{4.7}$$

After some algebraic steps (proved in Appendix B.1), we can also observe that

$$\bigtriangledown \mathscr{L}(\Theta^*) = \frac{1}{n}\Omega X^T \tag{4.8}$$

$$\delta\mathscr{L}(\Theta^*) = \frac{1}{2n}||\triangle X||_F^2 \tag{4.9}$$

where $\Omega = \Theta^* X - Y$.

4.4.2 Strong convexity

We now pose a technical condition on the error of the Taylor expansion, $\delta\mathscr{L}(\Theta^*)$, which provides a desirable control of the error magnitude. This bound is based on the *strong convexity* condition [98], formally expressed as follows

$$\delta\mathscr{L}(\Theta^*) \geq \kappa||\triangle||_F^2, \quad \text{for} \triangle \text{ around } \Theta^* \tag{4.10}$$

where $\kappa > 0$ is a positive constant. Intuitively, Eq. (4.10) requires the loss function $\mathscr{L}$ is sharply curved around its optimal solution $\hat{\Theta}$ by setting a lower bound on its gradient. The necessity of this requirement can be interpreted as follows. Consider the difference loss $\mathscr{L}(\hat{\Theta}) - \mathscr{L}(\Theta^*)$, it is expected that small $\mathscr{L}(\hat{\Theta}) - \mathscr{L}(\Theta^*)$ indicates small $\triangle = \hat{\Theta} - \Theta^*$. However, this assumption is reasonable only when $\mathscr{L}$ sharply curves at $\hat{\Theta}$. For example, to illustrate this point, if $\mathscr{L}$ is relative flat curved, a large $\triangle = \hat{\Theta} - \Theta^*$ might also leads to small $\mathscr{L}(\hat{\Theta}) - \mathscr{L}(\Theta^*)$. Therefore, to avoid the curve of $\mathscr{L}$ is too flat, we pose a strong convexity Eq. (4.10) constraint on it ,

which provides a desirable control of the magnitude of the error $\triangle$ in turn.

Note that in [99] a similar notion named *restricted strong convexity* is introduced. The term “restricted" means a constraint on the set of $\triangle$, which is necessary in high-dimensional statistical inference, in settings in which the ambient dimension k is larger than the sample size n (*i.e,* $k \geq n$ *or* $k \gg n$). In such settings, the global strong convexity is not always ensured, thus it is necessary to restrict $\triangle$ into a set where the *strong convexity* holds (hence the restricted strong convexity). In contrast, in this paper we consider the standard setting, $n > k$, where it is natural to assume the loss function is strongly convex at a global scale under mild conditions. Another analogous condition known as “restricted eigenvalues” (RE) is introduced in [100].

4.4.3 Main results

Equipped with the above notations and assumptions, we are ready to state our main result: a deterministic upper bound on the estimation error of the Laplacian regularized estimator Eq. (4.3), which holds with high probability and it is defined in the following theorem.

Theorem 11. *Consider the linear model Eq.* (4.2)*, where the strong convexity condition Eq.* (4.10) *holds and the regularization parameter* $\alpha \geq 8\sigma\sqrt{D}\frac{\sqrt{m+k}}{mn}$ *with any constant* $D \geq 2$*. Imposing* $rank(\triangle) \leq r$*, then the optimal solution* $\hat{\Theta}$ *obtained by Eq.* (4.3) *satisfies the following error bound:*

$$||\hat{\Theta} - \Theta^*||_F \leq \frac{\alpha(\sqrt{r} + 2||L\Theta^*||_F)}{\kappa + \alpha\lambda_2} \tag{4.11}$$

with high probability. where λ_2 *is the second smallest eigenvalue of the graph Laplacian L.*

The sketched proof of Theorem 11 is presented in the next section, while detailed proof is postponed to Appendix B.2. In the following, we provide key interpretations of Theorem 11.

(a) Note that Theorem 11 is a non-asymptotic bound on the optimas of Eq. (4.3) given a fixed regularization parameter α. When applied to particular models,

the *strong convexity* condition and the assumption $\alpha \geq ||\bigtriangledown \mathscr{L}(\Theta^*)||_\infty$ are required to be satisfied.

(*b*) The term $||L\Theta^*||_F$ quantifies the alignment between Θ^* and graph information L. Suppose there exists a groundtruth L^* over which Θ^* is smooth. It can be verified that any deviation of L from L^* would lead to a larger value of $||L\Theta^*||_F$ than $||L^*\Theta^*||_F$. In other words, deviations of Θ^* from the smoothness assumption leads to a larger estimation error.

(*c*) The term λ_2 illustrates the impact of the density of graph on the estimation accuracy. By comparing the eigenvalue profiles of two graphs, it can be seen that λ_2 of a dense graph is typically larger than that of a sparse graph. Therefore, from Eq. (4.11), we can deduce that dense graph leads to lower error, if it indeed aligns with Θ^*, because that a dense graph indicates more correlated rows of Θ^*, with the help of L, the Laplacian regularized estimator is expected to result in more accurate estimation.

The following corollary provides a bound applied to Eq. (4.3), with the detailed proof provided in Appendix B.3.

Corollary 1. *Consider the linear model Eq.* (4.2)*, where the strong convexity condition Eq.* (4.10) *holds and the regularization parameter* $\alpha \geq 8\sigma\sqrt{D}\frac{\sqrt{m+k}}{mn}$ *with any constant* $D \geq 2$*. If the rank of* $\triangle = \hat{\Theta} - \Theta^*$ *is at most r. Then the optimal solution* $\hat{\Theta}$ *obtained by the ridge estimator Eq.* (4.4) *satisfies the following error bound:*

$$||\hat{\Theta} - \Theta^*||_F \leq \frac{\alpha(\sqrt{r} + 2||\Theta^*||_F)}{\kappa + \alpha} \tag{4.12}$$

with high probability.

Corollary 1 takes a simpler form than Theorem 11 since L is ignored and $\lambda_2(I_m) = 1$.

4.5 Proofs

In this section, we sketch the proofs of Theorem 11, while more detailed proof is provided in Appendix B.2. We first present some Lemmas used in the proof.

4.5.1 Assumption and Lemmas

Assumption 1. *Given the definition* $\triangle = \hat{\Theta} - \Theta^*$*, we assume* $\triangle$ *satisfies the following property*

$$\sum_{j=1}^{k}\sum_{i=1}^{m}\triangle_{ji}^2 \gg \frac{1}{m}\sum_{j=1}^{k}(\sum_{i=1}^{m}\triangle_{ji})^2 \tag{4.13}$$

See Appendix B.4 for detailed reasoning.

Lemma 4. *Let the eigenvalues of the graph Laplacian L be denoted by* $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_m$*. Suppose* $\triangle$ *satisfies* ***Assumption 1****, then* $tr(\triangle^T L \triangle)$ *satisfies the following lower bound*

$$tr(\triangle^T L \triangle) \geq \lambda_2 ||\triangle||_F^2 \tag{4.14}$$

The detailed proof is provided in next section.

Lemma 5. *[100]. Let entries of* $\Omega = [\Omega_{ij}]$ *be i.i.d.* $\mathcal{N}(0, \sigma^2)$ *random variables, where* $\Omega \in \mathbb{R}^{m\times n}$*.* $X \in \mathbb{R}^{k\times n}$ *follows standard statistical regularity with independent columns. Then, for any* $D \geq 2$*,*

$$\frac{1}{n}||\Omega X^T||_\infty \leq 8\sigma\sqrt{D}\frac{\sqrt{m+k}}{mn} \tag{4.15}$$

with probability at least $1 - 2exp(-(D - log5)(m+k))$*.*

Lemma 6. *[99]. Let* $X \in \mathbb{R}^{k\times n}$ *be a random matrix with i.i.d. columns sampled from a k-variate* $\mathcal{N}(0, \Sigma)$*. Then for* $n \geq k$*, we have*

$$\mathbb{P}[\sigma_{min}(\frac{1}{n}XX^T) \geq \frac{\sigma_{min}(\Sigma)}{9}, \sigma_{max}(\frac{1}{n}(XX^T) \leq 9\sigma_{max}(\Sigma)] \geq 1 - 4exp(-n/2) \tag{4.16}$$

4.5.2 Sketch proof of Theorem 11

Due to the optimality of $\hat{\Theta}$ for Eq. (4.5), we have

$$\mathscr{L}(\hat{\Theta}) + \alpha\mathscr{R}(\hat{\Theta}) \leq \mathscr{L}(\Theta^*) + \alpha\mathscr{R}(\Theta^*) \tag{4.17}$$

Substituting $\hat{\Theta} = \Theta^* + \triangle$ and $\mathscr{R}(\Theta) = tr(\Theta^T L \Theta)$ yields

$$\mathscr{L}(\Theta^* + \triangle) - \mathscr{L}(\Theta^*) + \alpha(2tr((\Theta^*)^T L \triangle) + tr(\triangle^T L \triangle)) \leq 0 \tag{4.18}$$

By the definition of scalar product and its property, we have

$$tr((\Theta^*)^T L \triangle) = \langle L\Theta^*, \triangle \rangle \geq -|\langle L\Theta^*, \triangle \rangle| \tag{4.19}$$

Combining this with Eq. (4.7) the definition of $\delta\mathscr{L}(\Theta^*)$ and $\langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle \geq -|\langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle|$, we know that

$$-|\langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle| + \delta\mathscr{L}(\Theta^*) + \alpha(-2|\langle L\Theta^*, \triangle \rangle| + tr(\triangle^T L \triangle)) \leq 0 \tag{4.20}$$

Applying the Hölder Inequality [101], we have $|\langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle| \leq || \bigtriangledown \mathscr{L}(\Theta^*)||_\infty || \triangle ||_*$ and $|\langle L\Theta^*, \triangle \rangle| \leq ||L\Theta^*||_F || \triangle ||_F$. Thus

$$\delta\mathscr{L}(\Theta^*, \triangle) + \alpha tr(\triangle^T L \triangle) \leq || \bigtriangledown \mathscr{L}(\triangle, \Theta^*)||_\infty || \triangle ||_* + 2\alpha ||L\Theta^*||_F || \triangle ||_F \tag{4.21}$$

Imposing the *strong convexity* condition $\delta\mathscr{L}(\Theta^*) \geq \kappa || \triangle ||_F^2$. Assume $\alpha \geq || \bigtriangledown \mathscr{L}(\Theta^*)||_\infty$. Note the fact that if $rank(\triangle) \leq r$, then $|| \triangle |_* \leq \sqrt{r} || \triangle ||_F$, we have

$$\kappa || \triangle ||_F^2 + \alpha tr(\triangle^T L \triangle) \leq \alpha\sqrt{r} || \triangle ||_F + 2\alpha ||L\Theta^*||_F || \triangle ||_F \tag{4.22}$$

The remaining is to lower bound $tr(\triangle^T L \triangle)$. Lemma 4 provides a proper lower bound on this.

Substituting Eq. (4.14) into Eq. (4.22) and dividing both sides with $|| \triangle ||_F$ yields

$$|| \triangle ||_F \leq \frac{\alpha(\sqrt{r} + 2||L\Theta^*||_F)}{\kappa + \alpha\lambda_2} \tag{4.23}$$

The remained issue is to choose valid values for the regularization parameter α and the positive constant κ such that bound Eq. (4.23) holds in high probability. For

the value of α, we follow Lemma 5, from [100], which provides a upper bound on $||\bigtriangledown \mathscr{L}(\Theta^*)||_\infty$. For the choice of κ, Lemma 6, obtained from [99], provides a lower bound on $\delta\mathscr{L}(\Theta^*)$. Interested readers are referred to proofs in their original work.

More specifically, to decide a proper choice of α, we need to upper bound $||\bigtriangledown \mathscr{L}(\theta^*)||_\infty$ since we assume $\alpha \geq ||\bigtriangledown \mathscr{L}(\theta^*)||_\infty$. Recall Eq. (4.8), we have $||\bigtriangledown \mathscr{L}(\Theta^*)||_\infty = \frac{1}{n}||\Omega X^T||_\infty$. From Lemma 5, it can be seen that the choice $\alpha \geq 8\sigma\sqrt{D}\frac{\sqrt{m+k}}{mn}$ is suffice to ensure $\alpha \geq ||\bigtriangledown \mathscr{L}(\Theta^*)||_\infty$ holds in high probability.

To establish the *strong convexity* condition defined in Eq. (4.10), it is required to build a lower bound on $\delta\mathscr{L}(\Theta^*) = \frac{1}{2n}||\bigtriangleup X||_F^2$. As can be seen, similar to [99],

$$\frac{1}{2n}||\bigtriangleup X||_F^2 \geq \frac{\sigma_{min}(XX^T)}{2n}||\bigtriangleup||_F^2 \tag{4.24}$$

where σ_{min} refers to the minimum singular value of the matrix XX^T. Lemma 6 introduces a lower bound on $\frac{\sigma_{min}(XX^T)}{n}$. From Lemma 6, we can see $\frac{\sigma_{min}(XX^T)}{2n} \geq \frac{\sigma_{min}(\Sigma)}{18}$ with probability $1-4exp(-n)$. Therefore, $\kappa = \frac{\sigma_{min}(\Sigma)}{18}$ could guarantee that the condition $\delta\mathscr{L}(\Theta^*) \geq \kappa||\bigtriangleup||_F^2$ holds with high probability.
With the above valid choice of α and κ, Theorem 11 holds with high probability.

4.5.3 Sketch proof of Lemma 4

Let the eigendecomposition of the graph Laplacian L is $L = Q\Lambda Q^T$. Define $u = Q^T\bigtriangleup$, where $u_j = Q^T\bigtriangleup_j$ and $\bigtriangleup_j$ denotes the j-th column of u and $\bigtriangleup$, respectively. It is straightforward to show that

$$tr(\bigtriangleup^T L\bigtriangleup) = \sum_{j=1}^{k}\bigtriangleup_j^T L\bigtriangleup_j = \sum_{j=1}^{k}\sum_{i=1}^{m}\lambda_i u_{ji}^2 \tag{4.25}$$

Where λ_i denotes the i-th eigenvalue of L, and u_{ji} denotes the i-th entry of the j-th column of u.
Given L is a symmetric positive semidefinite matrix, its eigenvalues are real and nonnegative. Moreover, we assume that the graph $\mathscr{G}$ is a connected component,

hence $\lambda_1 = 0$. If we denote its eigenvalues as $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_m$, we have

$$\sum_{i=1}^{m} \lambda_i u_{ji}^2 = \sum_{i=2}^{m} \lambda_i u_{ji}^2 \geq \sum_{i=2}^{m} \lambda_2 u_{ji}^2 = (\sum_{i=1}^{m} \lambda_2 u_{ji}^2) - \lambda_2 u_{j1}^2 = \lambda_2 || \triangle_j ||_2^2 - \lambda_2 u_{j1}^2 \quad (4.26)$$

The first inequality is due to $\lambda_1 = 0$. Substituting Eq. (4.26) into Eq. (4.25) yields

$$tr(\triangle^T L \triangle) \geq \sum_{j=1}^{k} (\lambda_2 || \triangle_j ||_2^2 - \lambda_2 u_{j1}^2) = \lambda_2 || \triangle ||_F^2 - \lambda_2 || Q_1^T \triangle ||_2^2 \quad (4.27)$$

Where $Q_1^T = [1/\sqrt{m}, 1/\sqrt{m}, ..., 1/\sqrt{m}]^T$ is the first eigenvector of L. Therefore, $||Q_1^T \triangle ||_2^2 = \sum_{j=1}^{k} \frac{1}{m} (\sum_{i=1}^{m} \triangle_{ji})^2$. Also note that $|| \triangle ||_F^2 = \sum_{j=1}^{k} \sum_{i=1}^{m} \triangle_{ji}^2$, So Eq. (4.27) turns to

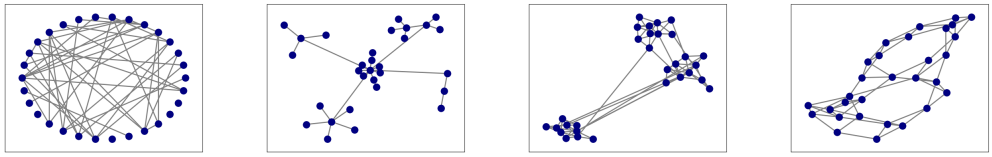
$$\begin{aligned} tr(\triangle^T L \triangle) &\geq \lambda_2 || \triangle ||_F^2 - \lambda_2 || Q_1^T \triangle ||_2^2 \\ &= \lambda_2 \sum_{j=1}^{k} \sum_{i=1}^{m} \triangle_{ji}^2 - \lambda_2 \sum_{j=1}^{k} \frac{1}{m} (\sum_{i=1}^{m} \triangle_{ji})^2 \end{aligned} \quad (4.28)$$

According to Assumption 1, at the right hand side of Eq. (4.28), the term $\lambda_2 \sum_{j=1}^{k} \frac{1}{m} (\sum_{1=1}^{m} \triangle_{ji})^2$ can be dropped. Hence,

$$tr(\triangle^T L \triangle) \geq \lambda_2 || \triangle ||_F^2 \quad (4.29)$$

4.6 Experimental validation

Experiment Setting We carry out experiments over various Laplacian and graph typologies. In the literature, there exists several definitions of Laplacian. Namely, Combinatorial Laplacian $L = D - W$, Normalized Laplacian $L = I - D^{1/2} W D^{1/2}$ and Random-Walk Laplacain $L = I - D^{-1} W$. On the topology side, we consider four commonly used graph topologies in the literature are shown in Fig 4.1. 1) *Erdős Rényi* (**ER**) model, an unweighted graph, in which each edge is generated independently and randomly with probability p; 2) *Barabási-Albert* (**BA**) model, an unweighted graph initialized with a connected graph with m nodes. Then, a new node is added to the graph sequentially with m edges connected to existing nodes



(a) ER **(b)** BA **(c)** SBM **(d)** WS

Figure 4.1: Graph Topologies

following the rule of preferential attachment where existing nodes with more edges has more probability to be connected by the new node; 3) *Stochastic Block Model* (**SBM**) is a generative model which tends to generate graph containing communities. Edges are denser within communities than between communities; 4) *Watts-Strogatz* (**WS**) model, an unweighted graph, which is a m-regular graph with edges randomly rewired with probability p.

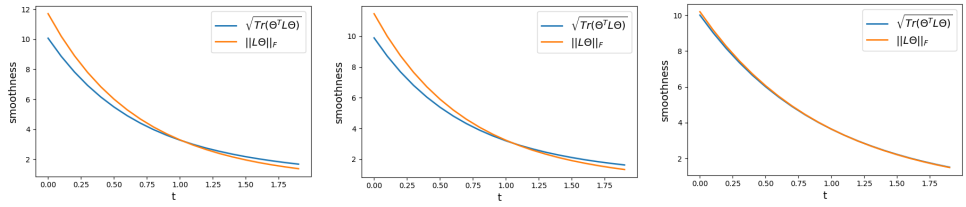
The impact of $||L\Theta||_F$, λ_2 and α

Theorem 11 contains three important terms $||L\Theta||_F$, λ_2 and α. In this section, we examine the impact of these terms, respectively. To conduct following experiments, we need a method to control the smoothness between Θ and the graph identified by L. To do this, we resort to Eq. (4.30)

$$\Theta = \exp(-tL)\Theta_0 \tag{4.30}$$

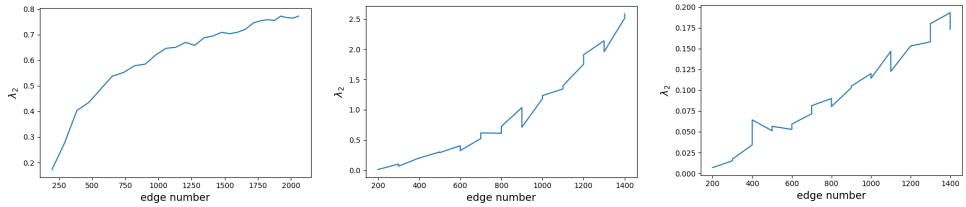
Given a fixed graph with Laplacian L, we first generate $\Theta_0 \in \mathbb{R}^{n\times d}$ a random matrix with normalized columns. Next, we obtain smooth Θ on the basis of Θ_0 and L through Eq. (4.30). The parameter $t \in (0,\infty)$ controls the level of smoothness. Larger t results in smoother Θ.

The term $||L\Theta||_F$, similar to Laplacain quadratic term $Tr(\Theta^T L\Theta)$, measures the smoothness between the graph L and Θ. In Fig 4.2, we show that the value of $||L\Theta||_F$ and $Tr(\Theta^T L\Theta)$ decreases as t increases under different Laplacians. It means that both $||L\Theta||_F$ and $Tr(\Theta^T L\Theta)$ measures the smoothness where lower value corresponding to higher smoothness between Θ and L. This is hold for all Lapalcians shown in Fig 4.2. Moreover, Theorem 11 indicates that the smaller $||L\Theta||_F$ the lower the estimation error (hence the more accurate the estimate). In Fig 4.4(a), we confirm



(a) $L = I - D^{-1}W$ **(b)** $L = D - W$ **(c)** $L = I - D^{1/2}WD^{1/2}$

Figure 4.2: $||L\Theta||_F$ and Smoothness with different Laplacian: Random Walk Laplacian $L = I - D^{-1}W$; Combinatorial Laplacian $L = D - W$; Normalized Laplacian $L = I - D^{1/2}WD^{1/2}$



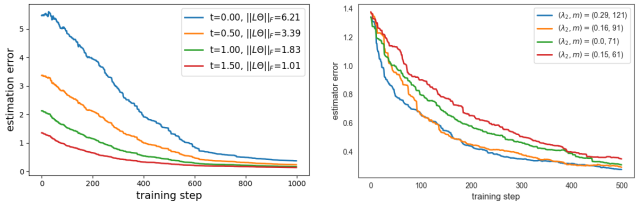
(a) $L = I - D^{-1}W$ **(b)** $L = D - W$ **(c)** $L = I - D^{1/2}WD^{1/2}$

Figure 4.3: λ_2 and graph connectivity (edge number): Random Walk Laplacian $L = I - D^{-1}W$; Combinatorial Laplacian $L = D - W$; Normalized Laplacian $L = I - D^{1/2}WD^{1/2}$

this by showing the error curves under different values of $||L\Theta||_F$ (different level of smoothness).

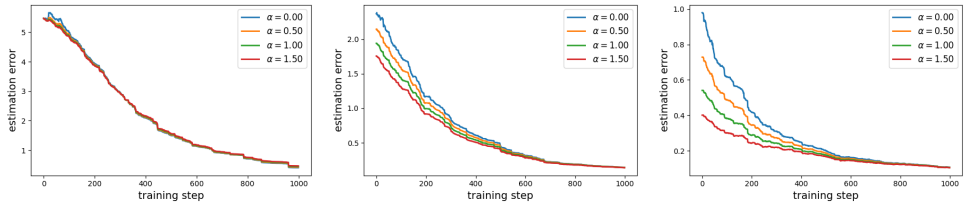
The term λ_2, the second smallest eigenvalue of L, is a measurement of the graph connectivity [102] where a denser connected graph typically has a higher value of λ_2. To confirm this, we first generate a fully connected graph and then remove edges gradually. Fig 4.3 confirms that larger value of λ_2 corresponds to denser connected graph (more edge numbers). This pattern is consistent across different Laplacian. In addition, Theorem 11 suggests that the larger λ_2 (denser connected) the lower estimation error. In Fig 4.4(b), we show learning curves under different value of λ_2 (various connectivity). It can be seen that in general, denser connected graph leads to lower estimation error (more accurate estimation).

The term α is a hyper-parameter, controlling the balance between fidelity term and Laplacian quadratic term. To test its impact, we fixed the graph L, Θ and the smoothness via t. For each level of smoothness t, we test the estimation error under different values of α. Fig 4.5 shows that given a smooth Θ (e.g., $t = 1$ or $t = 3$),



(a) **(b)**

Figure 4.4: The impact of $||L\Theta||_F$ and λ_2 on estimation error: m denotes the edge number, t is the parameter in Eq. (4.30) controlling the smoothness



(a) $t = 0$ **(b)** $t = 1$ **(c)** $t = 3$

Figure 4.5: The impact of α under different t (different smooth level)

larger value of α results in lower estimation error. However, Fig 4.5(c) shows that the Laplacian quadratic regularizer $Tr(\Theta^T L\Theta)$ has no impact on estimation error when Θ is no smooth with respect to L (when $t = 1$).

Results presented in Fig 4.4 and Fig 4.5 is built on combinatorial Laplacian $L = D - W$ and **ER** graph topology. Results of other Laplacian and topology have similar and consistent patterns.

4.7 Conclusion

In this chapter, we have analyzed the graph Laplacian regularized estimator, obtaining a non-asymptotic error bound on the Frobenius norm of estimation error. We derive the error bound in the main theorem, which provides a clear interpretation of the impact of both the graph structure and the smoothness prior on the estimation accuracy. Finally, we provide empirical evidence to confirm our theoretical analysis.

Chapter 5

Differentiable Linear Bandit

Previous chapters have investigated the bandit problems, mainly adopting UCB strategies. This is because Upper Confidence Bound (UCB) is among the most commonly used methods for linear Multi-armed bandit problems. However, while conceptually and computationally simple, this method highly relies on the evaluation of the confidence bounds, failing to strike the optimal exploration-exploitation if these bounds are not properly set. In the literature, confidence bounds are typically derived from concentration inequalities based on assumptions on the reward and noise distribution. The validity of these assumptions however is unknown in practice.

In this chapter, we aim at learning the confidence bound in a data-driven fashion, making it adaptive to the actual problem structure. Noting that existing UCB-based algorithms are non-differentiable with respect to confidence bound, we first propose a novel UCB-based differentiable linear bandit algorithm. Then, we introduce a gradient estimator, which allows us to learn the confidence bound via iterative gradient ascent. Moreover, we provide an upper bound on the cumulative regret of the proposed algorithm. Empirical results show that the learned confidence bound is significantly smaller than its theoretical upper bound and that the proposed algorithm outperforms baseline ones on both synthetic and real-world datasets.

5.1 Introduction

Multi-armed Bandit (MAB) [103] is an online decision-making problem, in which an agent selects arms sequentially and observes stochastic rewards as feedback. The

goal of the agent is to maximize the expected cumulative reward over trials. The expected reward of each arm is unknown *a priori* and it is learned from experience by the agent. As a consequence, the agent needs to balance the selection of arms to improve its knowledge (exploration) and the selection of the highest rewarding arm given the knowledge acquired till thus far (exploitation). This is formalized as the so-called exploration-exploitation trade-off. Bandit algorithms are designed to strike this trade-off. One class of MAB problems is the linear MAB [36], in which each arm is described by a feature vector and the expected reward follows a linear model over its feature vector and an unknown parameter vector. Each arm's feature vector is known *a priori* by the agent and it is considered as a hint on the arm reward. The learning problem boils down to the agent inferring the unknown parameter vector, based on the history (selected arms and received rewards) and then selecting arms accordingly.

In most existing works, confidence bounds are derived from concentration inequalities [37] [22] [104] given *a priori* assumptions on the reward distribution (e.g., sub-Gaussinaity). These bounds achieve strong minimax theoretical guarantees, outperforming competitor algorithms such as `LinTS` [26]. While these bounds are essential for a theoretical analysis, they do not necessarily translate into practice. In fact, these constructed confidence bounds are typically conservative *in practice*, as noted in [105] [106]. This is because concentration inequalities are usually built based on given reward distributions instead of the actual data (or problem structure). This results in non-adaptive and potentially wide confidence bounds which in turn leads to suboptimal performance in practice.

Alternatively, in this work we aim to learn the confidence bound in a data-driven fashion making it adaptive to the actual problem structure. Inspired by [107], we aim at having a parametrized and differentiable cumulative reward function with respect to the confidence bound, which can then be optimized. The key challenge is that existing UCB-typed algorithms are non-differentiable with respect to the confidence bound, mainly due to maximization of the UCB index (i.e., due to the presence of the arg max operator in the `OFUL` [37], `LinUCB` [36]). To address this, we propose

a novel differentiable UCB-typed linear bandit algorithm and introduce a gradient estimator which enables the confidence bound to be learned via gradient ascent.

Our proposed algorithm contains two core components. First, a novel UCB-based index not only summarizes the history of each arm but also differentiates arms to be suboptimal arms and non-suboptimal arms. Second, we utilize a softmax function to transform indexes into a probability distribution, where as a result of the novel index the probability for each suboptimal arm to be selected is arbitrary small. The key idea is that the exploration is conducted by selecting arms with large index, while the exploitation is achieved by soft-eliminating suboptimal arms (arbitrary small probability to be selected). The softmax function ensures the differentiability of the reward function, paving the way to learn confidence bound via gradient ascent. In addition, we provide a theoretical analysis on the cumulative regret upper bound.

In summary, our contributions can be listed as follows:

- We propose a novel UCB-typed linear bandit algorithm where the expected cumulative reward is a differentiable function of the confidence bound.
- We introduce a gradient estimator and show how the confidence bound can be learned via gradient ascent.
- Theoretically, we prove an upper bound of cumulative regret of the proposed algorithm.
- Empirically, we show the learned confidence bound is significantly smaller than its theoretical counterpart, leading to substantially lower cumulative regrets with respect to state-of-the-art baselines on both synthetic and real-world datasets.

Notation: $[K]$ denotes $\{1,2,...,K\}$. Arm is indexed by $i, j \in \mathscr{A}$. We use lower letter, e.g., x, to denote vector and upper letter. e.g., M, to denote matrix. For a positive definite matrix $M \in \mathbb{R}^{d\times d}$ and a vector $x \in \mathbb{R}^d$, we denote the weighted 2-norm by $||x||_M = \sqrt{x^T Mx}$. Each arm k is represented by the feature vector $x_k \in \mathbb{R}^d$. We denote by $\mathbb{P}$ and $\mathbb{E}$ the probability distribution and the expectation operator, respectively.

5.2 Related work

Our work is inspired by [107], which was the first attempt in addressing policy-gradient optimization of bandit policies via differentiable bandit algorithm. However, there are fundamental differences between [107] and our work. First, authors proposed a differential bandit framework for Bayesian MAB problem, which is not directly applicable to linear MAB problems. Second, the main goal of [107] is to learn the learning rate (coldness-parameter) of the softmax function, while our algorithm aims at learning the size of the confidence bound. Third, we propose algorithms for both offline and online settings, while [107] covered the offline setting only.

Another work focused on data-dependent UCB is [108]. Authors proposed an algorithm called `bootstrapedUCB`. In [108], the stochastic reward is assumed to be sub-Weibull random variable. Multiplier bootstrap was employed to approximate the reward distribution. The bootstrapped quantile acted as UCB to facilities exploration. Their algorithm was deployed on both MAB and linear MAB problems, while regret analysis covered MAB only. Similar to this work, other bootstrap techniques were employed [109] [110] [111]. Although aiming to the same goal (data-dependent UCB), these works are fundamentally different from our approach. Our algorithm is a differentiable bandit algorithm where we rely on gradient estimator to learn UCB. Their algorithm is non-differentiable, relying on the bootstrapped quantile of the assumed reward distribution to construct UCB.

Bootstrap techniques were used also for Thompson Sampling exploration in [105], in which author proposed the `BoostrapThompson` algoritihm for MAB. Bootstrap techniques were used to sample observations from historical and pseudo-observations to approximate the posterior distribution which was then used to encourage exploration. As an extension, [112] generalized this technique to Gaussian reward MAB, while [113] and [106] proposed an extension to contextual linear bandit, achieving the same regret bound of `LinTS` [26]. The problem they aimed to address was the computational infeasibility of inferring posterior distribution when reward follows nonlinear models. This departs from our goal, which is rather

learning the confidence bound from data.

Our work can be viewed as a subtle combination of EXP3 [103] and Phased Elimination[1] [71]. EXP3 was designed for MAB, where arms with higher empirical averaged reward are signed with larger probability by softmax function. The coldness-parameter of softmax function is a tuneable hype-parameter chosen by the user. In our work, we propose a novel scheme to set this parameter automatically in a data-driven fashion. Moreover, although Exp3 is a differentiable bandit algorithm, it is not an UCB-typed algorithm. Phased Elimination eliminates suboptimal arms based on the same index as ours and selects non-suboptimal arms uniformly (pure exploration). There are several fundamental differences between this approach and our work: *i)* the confidence bound in our work is learned from data and not from concentration inequalities; *ii)* Phased Elimination is a non-differentiable algorithm; *iii)* Phased Elimination achieves optimality in a worst case scenario (minmax regret) while our algorithm get an empirical gain being data dependent.

In summary, to the best of our knowledge, our work is the first differentiable UCB-typed linear bandit algorithm which enables confidence bound to be learned purely from data without relying on concentration inequalities and assumptions on the form of reward distribution.

5.3 Problem setting

We consider the stochastic linear bandit with an arm set $\mathscr{A}$ and a time horizon of T-rounds. The arm set contains K arms, i.e., $|\mathscr{A}| = K$, where K could be large. Each arm $i \in \mathscr{A}$ is associated with a known feature vector $x_i \in \mathbb{R}^d$. The expected reward of each arm $\mu_i = x_i^T \theta$ follows a linear relationship over x_i and an unknown parameter vector θ. Similarly to other works in the bandit literature, we assume that arm feature and parameter vector are bounded $||x||_2 \leq L$ and $||\theta||_2 \leq C$, where $L > 0$ and $C > 0$. At each decision opportunity $t \in [T]$, the learning agent selects one arm $i \in \mathscr{A}$ out of $\mathscr{A}$. Upon this selection, the agent observes the instantaneous reward $y_t \in [0,1]$, which is drawn independently from a distribution with unknown

[1] Algorithm: Phased elimination with G-optimal exploration page. 258 [71]

mean $\mu_i = x_i^T \theta$. The agent aims to maximize the expected cumulative reward over the time horizon T. Namely,

$$Y_T = \sum_{t=1}^{T} \mathbb{E}[y_t] \tag{5.1}$$

This is equivalent to minimize the expected cumulative regret which measures the difference between the expected cumulative reward if the optimal arm were always selected and the agent's expected cumulative reward. Denoting by $\mu_* = \max_{i \in \mathscr{A}} x_i^T \theta$ the expected reward of the optimal arm, we get

$$R_T = T\mu_* - \sum_{t=1}^{T} \mathbb{E}[y_t]\,. \tag{5.2}$$

Upper Confidence Bound (UCB) in linear bandit problems has been introduced in Chapter 1, we restate the key concept here for the ease of reading. The upper confidence bound algorithm, e.g., OFUL [37], is designed based on the *Optimism in Face of Uncertainty* principle. The key aspect is to construct a confidence bound of the estimated reward of each arm. Formally, at each round t, the confidence bound is defined as

$$|\hat{\mu}_{i,t} - \mu_i| \leq \beta ||x_i||_{V_t^{-1}}, \quad \forall i \in \mathscr{A} \tag{5.3}$$

where $\hat{\mu}_{i,t}$ is the estimate of the reward of arm i at round t and $V_t = \sum_{t=1}^{T} x_t x_t^T$ is the Gram matrix up to round t. Then, the agent selects the arm with the highest upper confidence bound as follows

$$i_t = \arg\max_{i \in \mathscr{A}} \hat{\mu}_{i,t} + \beta ||x_i||_{V_t^{-1}} \tag{5.4}$$

It is well known that the tighter the bound in Eq. (5.3), the better the balance between exploration and exploitation [59]. Most existing confidence bounds are established based on concentration inequalities. e.g., Hoeffding inequality [22], self-normalized [37], Azuma Inequality [71], Bernstein inequality [104]. As a specific example, under the assumption of the stochastic reward to be a R-sub-Gaussian variable, one of the state-of-the-art high probability upper bound of β, derived based on properties

Algorithm 13: SoftUCB

Input : β, $\mathscr{A}$, K, T, α.

Initialization : $V_0 = \alpha I \in \mathbb{R}^{d\times d}$, $b_0 = \mathbf{0} \in \mathbb{R}^d$, $\hat{\theta}_0 = \mathbf{0} \in \mathbb{R}^d$, $\gamma_0 = 0$.

for $t \in [1, T]$ **do**

1. Find $S_{i,t}, \forall i \in \mathscr{A}$ via Eq. (5.7) with β.
2. Find π_t via Eq. (5.8) with γ_{t-1}.
3. Select arm $i_t \in \mathscr{A}$ randomly following π_t and receive payoff y_t.
4. Update $V_t \leftarrow V_t + x_t x_t^T$, $b_t \leftarrow b_{t-1} + x_t y_t$ and $\hat{\theta}_t = V_t^{-1} b_t$.
5. Update γ_t via Lemma 8.

end

of self-normalized martingale, was given by [37]:

$$\beta \leq R\sqrt{2\log\left(\frac{1}{\delta}\right) + d\log\left(1+\frac{T}{d}\right)} + \sqrt{\alpha}C \tag{5.5}$$

where α is a regularizer parameter of least-square estimator, $1-\delta$ is the probability of which Eq. (5.3) holds and $||\theta||_2 \leq C$. The tightness of these bounds rely on the validity of assumptions on the reward distribution, which are unfortunately unknown in practice. Alternatively, we aim at learning the confidence bound, i.e., β, in a data-driven fashion without any *a priori* assumption on the unknown reward distribution except the linearity function of the mean reward, i.e., is $\mu_i = x_i^T \theta, \forall i \in \mathscr{A}$.

5.4 Algorithms

In this section, we first present a novel UCB-based index. Then, we provide a gradient estimator of the expected cumulative regret with respect to confidence bound. Next, we propose bandit algorithm. Finally, we prove a theoretical regret upper bound.

5.4.1 Differentiable Algorithm

Our proposed algorithm named SoftUCB is shown in Algorithm 13. SoftUCB contains two core components: an UCB-based index $S_{i,t}$ and an arm selection policy

π_t. Formally, for $i \in \mathscr{A}$, $\hat{\mu}_{i,t} = x_i^T \hat{\theta}_t$ where $\hat{\theta}_t = V_t^{-1} \sum_{s=1}^{t} x_s y_s$ is the least-square estimator and $V_t^{-1} = \sum_{s=1}^{t} x_s x_s^T$ is the Gram matrix up to round t. Let denote by $i_* = \arg\max_{i \in \mathscr{A}} \hat{\mu}_{i,t} - \beta ||x_i||_{V_t^{-1}}$ the arm with the largest lower confidence bound at round t. Let us also define

$$\phi_{i,t} = ||x_i||_{V_t^{-1}} + ||x_{i_*}||_{V_t^{-1}} \quad \text{and} \quad \hat{\Delta}_{i,t} = \hat{\mu}_{i_*,t} - \hat{\mu}_{i,t} \tag{5.6}$$

where β is the confidence bound defined in Eq. (5.3) and $\hat{\Delta}_{i,t}$ is the estimated reward gap between i_* and i. We introduce the UCB-based index $S_{i,t}$ defined as

$$S_{i,t} = \beta \phi_{i,t} - \hat{\Delta}_{i,t} \,. \tag{5.7}$$

Lemma 7. *If $S_{i,t} < 0$, arm i is a suboptimal arm, i.e., $\mu_* - \mu_i > 0$. If $S_{i,t} \geq S_{j,t} \geq 0$, then the upper confidence bound $\hat{\mu}_{i,t} + \beta ||x_i||_{V_t^{-1}} \geq \hat{\mu}_{j,t} + \beta ||x_j||_{V_t^{-1}}$.*

The proof is provided in Appendix C.1

The index $S_{i,t}$ has two key properties: *i)*, $S_{i,t}$ differentiates arms into suboptimal arms and non-suboptimal arms. Specifically, $S_{i,t} < 0$ identifies arms which are suboptimal and therefore could be eliminated (i.e., not selected by the agent); *ii)*, $S_{i,t} \geq S_{j,t} \geq 0$ implies that the upper confidence bound $\hat{\mu}_{i,t} + \beta |x_i||_{V_t^{-1}} \geq \hat{\mu}_{j,t} + \beta ||x_j||_{V_t^{-1}}$ and therefore arm i is more likely to be selected, in line with the *Optimism in Face of Uncertainty* principle.

We now describe the arm selection strategy. At each round $t \in [T]$, the probability for arm i to be selected is defined as

$$p_{i,t} = \frac{\exp(\gamma_t S_{i,t})}{\sum_{j=1}^{K} \exp(\gamma_t S_{j,t})} \tag{5.8}$$

where $\gamma_t > 0$ is the coldness-parameter controlling the concentration of the arm probability (policy).

Lemma 8. *At any round $t \in [T]$, for any $\delta \in (0,1)$, setting*

$$\gamma_t \geq \log(\frac{\delta |\mathscr{L}_t|}{1-\delta}) / \tilde{S}_{max,t} \tag{5.9}$$

guarantees that $p_{\mathscr{U}_t} = \sum_{i \in \mathscr{U}_t} p_{i,t} \geq \delta$ *and* $p_{\mathscr{L}_t} = \sum_{i \in \mathscr{L}_t} p_{i,t} < 1 - \delta$.

The proof is provided in Appendix C.2.

At each round t, the arm set $\mathscr{A}$ is divided into two subsets $\mathscr{U}_t$ and $\mathscr{L}_t$ with $\mathscr{U}_t \cup \mathscr{L}_t = \mathscr{A}$ and $\mathscr{U}_t \cap \mathscr{L}_t = \emptyset$. Namely, $\mathscr{L}_t$ is the set of suboptimal arms (i.e., $i \in \mathscr{L}_t$ if $S_{i,t} < 0$) and $\mathscr{U}_t$ is the set of non-suboptimal arms (i.e., $i \in \mathscr{U}_t$ if $S_{i,t} \geq 0$). $\tilde{S}_{\max,t} = \max_{i \in \mathscr{U}_t} S_{i,t}$, $|\mathscr{L}_t|$ is the cardinality of $\mathscr{L}_t$ and δ is a probability hyper-parameter explained in the following Lemma. Lemma 8 guarantees that suboptimal arms ($i \in \mathscr{L}_t$) are selected with an arbitrary small probability (i.e., $p_{\mathscr{L}_t} < 1 - \delta \approx 0$ when $\delta \approx 1$). Furthermore, a positive γ_t guarantees $p_{i,t} \geq p_{j,t}$ if $S_{i,t} \geq S_{j,t} \geq 0$, $\forall i, j \in \mathscr{U}_t$ which obeys the *Optimism in Face of Uncertainty* principle.

Overall, `SoftUCB` (soft-) eliminates suboptimal arms and selects non-suboptimal arms according to the index in Eq. (5.7) which favours the selection of arms with either high estimated reward or high uncertainty.

5.4.2 Gradient Estimator of β

We now show that the expected cumulative reward of `SoftUCB` is a differentiable function over β and we introduce a gradient estimator. Formally, given the expected cumulative reward defined in Eq. (5.1) and `SoftUCB` described above, we have the optimization objective defined as

$$\max_{\beta} \sum_{t=1}^{T} \mathbb{E}[y_t] = \max_{\beta} \sum_{t=1}^{T} \sum_{i=1}^{K} p_{i,t} \mu_i, \;\; s.t. \; |\mu_i - \hat{\mu}_{i,t}| \leq \beta ||x_i||_{V_t^{-1}}, \;\; \forall i \in \mathscr{A}, t \in [T] \tag{5.10}$$

The imposed constraint ensures that $\beta ||x_i||_{V_t^{-1}}$ is indeed an actual upper confidence bound (UCB) at any round $t \in [T]$ for any arm $i \in \mathscr{A}$. Applying the Lagrange multipliers gives the new objective:

$$\max_{\beta} \sum_{t=1}^{T} \sum_{i=1}^{K} p_{i,t} \mu_i - \eta (|\mu_i - \hat{\mu}_{i,t}| - \beta ||x_i||_{V_t^{-1}}), \;\; s.t. \;\; \eta > 0 \tag{5.11}$$

The gradient of β, denoted as $g(\beta)$, can be derived as (proof in Appendix C.3):

$$g(\beta) = \sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\mu_i \left(\gamma_t \phi_{i,t} - \frac{\sum_{j=1}^{K} \gamma_t \phi_{j,t} \exp(\gamma_t S_{j,t})}{\sum_{j=1}^{K} \exp(\gamma_t S_{j,t})} \right) + \eta ||x_i||_{V_t^{-1}} \tag{5.12}$$

Note that μ_i is unknown in practice and it is therefore replaced by its empirical estimate $\hat{\mu}_{i,t}$, leading to the following gradient estimator

$$\hat{g}(\beta) = \sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\hat{\mu}_{i,t} \left(\gamma_t \phi_{i,t} - \frac{\sum_{j=1}^{K} \gamma_t \phi_{j,t} \exp(\gamma_t S_{j,t})}{\sum_{j=1}^{K} \exp(\gamma_t S_{j,t})} \right) + \eta ||x_i||_{V_t^{-1}} \tag{5.13}$$

The gradient estimator $\hat{g}(\beta)$ in Eq. (5.13) enables β to be learned via gradient ascent. As a stochastic gradient method, under standard condition of learning rate, e.g., RM [114], it is expected that $\hat{\beta}$ converges to local optimum.

$$\max_{\beta} \sum_{t=1}^{T} \mathbb{E}[y_t] = \max_{\beta} \sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\mu_i, \tag{5.14}$$

$$g(\beta) = \sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\mu_i \left(\gamma_t \phi_{i,t} - \frac{\sum_{j=1}^{K} \gamma_t \phi_{j,t} \exp(\gamma_t S_{j,t})}{\sum_{j=1}^{K} \exp(\gamma_t S_{j,t})} \right) \tag{5.15}$$

Remark 6. *Due to limits of knowledge in the area of convexity analysis, the author leave the analysis and discussion on the convexity of the problem as a future work.*

5.4.3 Training Settings

Equipped with the gradient estimator $\hat{g}(\beta)$(Eq. (5.13)), we now show how to learn β in offline and online settings. The corresponding algorithms named `SoftUCB offline` and `SoftUCB online` are presented in Appendix C.5.

Offline setting. In this setting, multiple T-rounds trajectories of the bandit problem with the same arm set $\mathscr{A}$ are used to train β, which is refined after each T-rounds trajectory. The key steps are to initialize $\hat{\beta}_0$ and run `SoftUCB` on $\mathscr{A}$ for N training trajectories – each trajectory containing T-rounds. After each trajectory $n \in [N]$, update $\hat{\beta}_n \leftarrow \hat{\beta}_{n-1} + \lambda \hat{g}(\beta)$ via Eq. (5.13) where λ is the learning step. At the end of the training, run `SoftUCB` on $\mathscr{A}$ with $\hat{\beta} = \hat{\beta}_N$.

As a result of the training, the value of $\hat{\beta}$ is optimized in such a way that it

maximizes the expected cumulative reward of arm set $\mathscr{A}$. Empirically, the $\hat{\beta}$ to which the algorithm converges is substantial less than its theoretical upper bound Eq. (5.5). This translates into a significant regret reduction. In the following subsection we provide a theoretical regret upper bound of `SoftUCB offline`

While the above method is fully adaptive to the structure of $\mathscr{A}$, it provides a burden on the computational complexity. Specifically, the computational complexity `SoftUCB offline` is $\mathscr{O}(NKT)$, since we run `SoftUCB` N trajectories with K arms and T rounds in each trajectory. This is much higher than other linear algorithms such as `LinUCB` [37] and `LinTS` [26]. To mitigate this issue, we propose `SoftUCB online` which learns β within one trajectory in an online fashion.

Online setting. In this setting, $\hat{\beta}$ is updated online during one T-rounds trajectory. Specifically, $\hat{\beta}_0$ is initialized and `SoftUCB` on $\mathscr{A}$ is run for T rounds. At the end of each round $t \in [T]$, update $\hat{\beta}_t \leftarrow \hat{\beta}_{t-1} + \lambda \hat{g}_t(\beta)$ where λ is the learning step and $\hat{g}_t(\beta)$ is the gradient estimator (Eq. (5.17) defined blow). This reduces the computationally complexity to $\mathscr{O}(KT)$ since it does not require the N-training trajectories, which is at the same level of `OFUL` [37], `LinUCB` [36] and `LinTS` [26].

In this setting, $Y_T = \sum_{t=1}^{T} \mathbb{E}[y_t]$, the objective function we aim at maximizing, is not available before the end of the trajectory. To obviate to this problem, similarly to policy gradient methods for non-episodic reinforcement learning problems [1], we update $\hat{\beta}$ to maximizes the average reward per round $\hat{Y}_t$. Formally, at each round t, $\hat{Y}_t$ consists of two parts: the observed cumulative reward up to round t and bootstrapped future reward under the current policy $\pi_t = [p_{1,t}, p_{2,t}, ..., p_{K,t}]$. This translates in the following problem formulation

$$\begin{aligned} \max_{\beta} \hat{Y}_t = \max_{\beta} \left(\sum_{s=1}^{t} \sum_{i=1}^{K} p_{i,s} \hat{\mu}_{i,s} + (T-t) \sum_{i=1}^{K} p_{i,t} \hat{\mu}_{i,t} \right) / T \\ s.t. \;\; |\hat{\mu}_{i,t} - \mu_{i,t}| \leq \beta ||x_i||_{V_t^{-1}}, \forall i \in \mathscr{A} \end{aligned} \tag{5.16}$$

The gradient estimator $\hat{g}_t(\beta)$ at round t can be derived as

$$\hat{g}_t(\beta) = \frac{1}{T} \left(\sum_{s=1}^{t} \sum_{i=1}^{K} \hat{\mu}_{i,s} \bigtriangledown_{\beta} p_{i,s} + (T-t) \sum_{i=1}^{K} \hat{\mu}_{i,t} \bigtriangledown_{\beta} p_{i,t} + \eta ||x_i||_{V_t^{-1}} \right) \tag{5.17}$$

It is worth noting that, at the end of trajectory $t = T$, the $\hat{Y}_t$ converges to. Y_T in the offline setting.

5.4.4 Theoretical Analysis

Theorem 12. *Define* $\mathbb{E}[r_t] = \mathbb{E}[\mu_* - \sum_{i=1}^{K} p_{i,t}\mu_i]$ *be the expected regret at round* $t \in [T]$. *Let* $\hat{\beta} = \beta_N$ *be the confidence bound learned from the offline training setting after N T-rounds trajectories. Let assume that* γ_t *follows Lemma 8 and* $\delta \approx 1$. *The cumulative regret of* `SoftUCB` *is bounded as*

$$R_T = \sum_{t=1}^{T} \mathbb{E}[r_t] \leq 4\sqrt{2}\hat{\beta}\delta\sqrt{Td\log\left(\alpha + \frac{T}{d}\right)} = \tilde{\mathcal{O}}\left(\hat{\beta}\sqrt{dT\log\left(1 + \frac{T}{d}\right)}\right) \quad (5.18)$$

where $\tilde{\mathcal{O}}(\cdot)$ *hides absolute constant. The proof is contained in Appendix C.4.*

Theorem 12 provides a regret upper bound of `SoftUCB` in the offline setting. To compare the regret bound with that of other algorithms, we show $\hat{\beta}$ explicitly in the upper bound. Our regret bound scales with d and T as the regret bound $\mathcal{O}(\beta\sqrt{dT})$ of existing UCB-typed algorithms, e.g., `OFUL` [37], `LinUCB` [36], `Giro` [113]. Since we make no assumption on the reward distribution, we cannot derive a theoretical upper bound on $\hat{\beta}$. However, it is worth to noting that empirical results (in next section) show that $\hat{\beta}$ is significantly smaller than its theoretical upper bound Eq. (5.5). The theoretical analysis for the online setting is left for future works.

5.5 Experiments

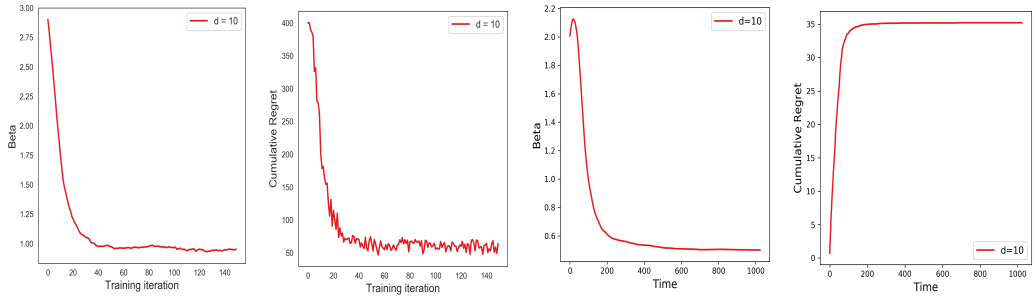
Our experimental evaluation aims to answer the following questions: (1) Does the learning curve of $\hat{\beta}$ converge in offline and online settings? (2) Is $\hat{\beta}$ lower than its theoretical counterpart? (3) How do our proposed algorithms perform compare to baseline ones?

To address the above questions, we carried out simulations with both synthetic and real-world datasets. In the former ones, we consider $K = 50$ arms, each one with a representative feature vector $x_i \in \mathbb{R}^d$, with $d = 10$ and 20. Each entry of the feature vector is drawn uniformly at random in the $[-1, 1]$ range. Arm feature vectors are then normalized to be unit vectors. The parameter vector θ is generated as a

Table 5.1: The comparison between $\hat{\beta}$ (offline) and theoretical bound $\tilde{\beta}$

.

$d=5, T=2^8$	$d=5, T=2^9$	$d=5, T=2^{10}$	$d=10, T=2^{10}$	$d=15, T=2^{10}$
$\hat{\beta}=\mathbf{0.5}$	$\hat{\beta}=\mathbf{0.6}$	$\hat{\beta}=\mathbf{0.9}$	$\hat{\beta}=\mathbf{1.1}$	$\hat{\beta}=\mathbf{1.2}$
$\tilde{\beta}=2.56$	$\tilde{\beta}=2.66$	$\tilde{\beta}=2.76$	$\tilde{\beta}=3.25$	$\tilde{\beta}=3.61$



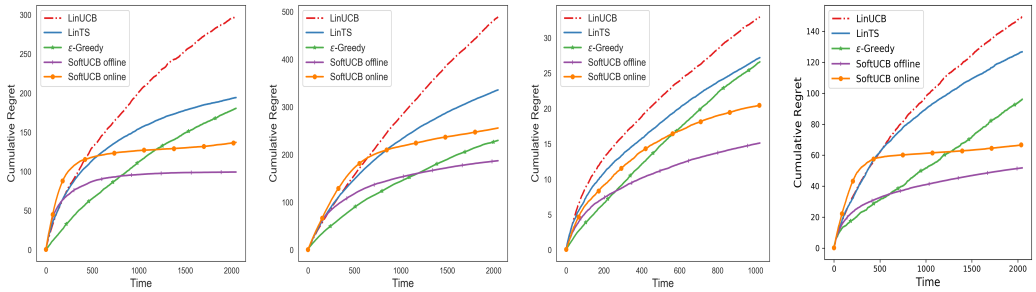
(a) $\hat{\beta}$ (offline) **(b)** R_T (offline) **(c)** $\hat{\beta}$ (online) **(d)** R_T (online)

Figure 5.1: Learning curves of SoftUCB offline and SoftUCB online

random unit vector. The noise variance is set as 0.5 and the regularizer parameter is $\alpha = 1$. For the real-world datasets we used **Jester** [115] and **Movielens** [74] datasets (see Appendix C.7 for more details). The proposed algorithms is compared against baseline ones, namely `LinUCB` [37], `LinTS` [26] and ε`-greedy` [1]. In `LinUCB`, the β is given by Eq. (5.5), `LinTS` follows [26], and in ε`-greedy` $\varepsilon = 0.05$.

Fig. 5.1 depicts the learning curves of $\hat{\beta}$ and the corresponding R_T in both offline and online settings for the synthetic datasets with $d = 10$. Note that in offline setting, $\hat{\beta}$ is optimized to maxmize the expected cumulative reward Eq. (5.14), while in online setting, $\hat{\beta}$ is optimized to maximize the average reward per round Eq. (5.16). It is worth noting that in both settings the gradient ascendant algorithm converges, i.e., $\hat{\beta}$ and R_T achieve convergence. In more details, in Table 5.1, we compare $\hat{\beta}$ to which the offline training convergences too and its theoretical value $\tilde{\beta}$ given by Eq. (5.5). Clearly, $\hat{\beta}$ is significantly less than $\tilde{\beta}$ consistently in all cases. This is because $\hat{\beta}$ is adaptive to the structure of $\mathscr{A}$, while $\tilde{\beta}$ is derived based on worst-case (minimax analysis). This leads to $\hat{\beta}$ being much less conservative (i.e., reaching lower values) than $\tilde{\beta}$, which aims at respective a set of problems instead of specific ones. The corresponding learning curves are depicted in Appendix C.6.

In Fig. 5.2 the expected cumulative regret is depicted over time for different



(a) $d = 10$ **(b)** $d = 20$ **(c) MovieLens** **(d) Jester**

Figure 5.2: Performance of algorithms on synthetic and real-world datasets

bandit algorithms and across different datasets. We note that the proposed `SoftUCB` algorithms converge to lower cumulative regret with respect to baselines. This is mainly motivated by the fact that *i*) the confidence bound $\hat{\beta}$ is tailored to each dataset (limiting the exploration when not necessarily needed), *ii*) the proposed algorithm eliminates (softly) suboptimal arms, limiting the suboptimality of the agent's actions. Worth a discussion is the beahvior of `SoftUCB online`, which encours into a large regret at the initial phase. This is mainly because $\gamma_0 = 0$, and $|\mathscr{L}_t| = 0$. Those initialized values lead to the `SoftUCB online` selecting arms uniformly at random, resulting in large regret. Later, when suboptimal arms are identified, i.e., $|\mathscr{L}_t| > 0$, $\gamma_t > 0$, non-suboptimal arms are selected following index Eq. (5.7) which controls the regret.

Finally, during our experiments, we noticed that the convergence of $\hat{\beta}$ in both offline and online setting is sensitive to the Lagrange multiplier η. With large η, the gradient ascent algorithm fails in converging, this is because the gradient estimator Eq. (5.17) is dominated by $\eta||x_i||_{V_t^{-1}}$. On the other hand, too small η does not ensure the key constraint $|\hat{\mu}_{i,t} - \mu_i| \leq \beta||x_i||_{V_t^{-1}}$. This can lead to erroneously eliminating the optimal arm. Therefore, the hyper-parameter η needs to be tuned carefully during experiments.

5.6 Conclusion

We propose `SoftUCB`, a novel UCB-typed linear bandit algorithm based on an *adaptive* confidence bound, resulting in a less conservative algorithm respect to

UCB-typed algorithms with *constructed* confidence bounds. The key novelty is to propose an expected cumulative reward which is a differentiable function of the confidence bound, and derive a gradient estimator, which enables confidence bound to be learned via gradient ascent. The estimated confidence bound $\hat{\beta}$ can be updated under offline/online training settings with the proposed `SoftUCB offline` and `SoftUCB online`, respectively. Theoretically, we provide a $\tilde{\mathcal{O}}(\hat{\beta}\sqrt{dT})$ regret upper bound of `SoftUCB` in the offline setting. Empirically, we show that $\hat{\beta}$ is significantly less that its theoretical counterpart leading to a reduction of the cumulative regret compared to state-of-the-art baselines.

There are several directions for future work. First, this chapter work can be combined with meta-learning algorithms, e.g., `MAML` [116], to learn a confidence bound which is adaptive to the common structure of a set of bandit tasks. Second, we believe this work can be generalized to reinforcement learning (RL) tasks where exploration and exploitation trade-off is a long-standing challenge.

Chapter 6

Learn Diffusion-Distance Induced State Representations

While the above works have been focused on the exploration-exploitation dilemma. We now focus on the problem of representation learning in Reinforcement Learning (RL). As already mentioned in Chapter 1, representation learning is critical for the sample efficiency of solving RL problems. In this chapter, we address this problem by learning state representations which follows the intrinsic structure of value function. In this chapter, we first identify that value functions are Lipschitz continuous over a *diffusion-distance* metric induced by rewarding states. Then we propose a sample-based approach to learn such metric and an auxiliary loss, which utilizes the learned metric to shape state representations. Empirical results demonstrate that the proposed auxiliary loss leads to the learning of informative and structural representations, and improves the control performance in comparison with state-of-the-art baselines methods that still exploit behavioural-based representation learning strategies. In addition, we theoretically compare with existing metrics to reveal advantages in terms of value function approximation bound and performance bounds.

6.1 Introduction

Many Reinforcement Learning (RL) tasks present raw, high-dimensional observations to agents, which usually contains redundant and distracting information. The

sample efficiency of RL algorithms heavily relies on the ability to learn from the redundant observation a compact state representation, which is then used to approximate the value function and enact new policies. To achieve this, representation learning techniques have arose as a key component in modern RL algorithms.

The majority of the works focused on representation learning in RL take a form of *Auxiliary tasks*. The agent, besides solving the primary RL task (learning an optimal policy), is required to learn other aspects of the task. Examples includes predicting the value function of other reward functions [52] or other policies [117], reconstructing the observation [55], and predicting the immediate reward [54] or next observation [53]. Most of such works demonstrate empirically the benefits of *Auxiliary tasks* on the performance of agents. While there is a limited theoretical understanding on the effectiveness of auxiliary tasks [56], a common view is that auxiliary tasks could improve the learning efficiency by shaping the representation in more semantic ways [52].

In this work, we aim at improving the sample efficiency of RL agents via learning a state representation able to reflect behavioral properties of the agent. Specifically, we focus on learning a representation able to encapsulate the distance between states, with the state distance reflecting the value function. Specifically, we first observe that the value difference between states pair is determined by their *diffusion-distance* with respect to *rewarding* states. The *diffusion-distance* refers to the pair-wise Successor Representation [118]. Based on this observation, we introduce a state distance metric, which is a weighted aggregation of the *diffusion-distance* from each rewarding state. We prove that value functions are Lipschitz continuous over the metric. We then derive theoretically the value function approximation error bound, showing the theoretical gain with respect to existing metrics. We then propose a novel representation learning strategies for the agent to learn a representation able to reflect such diffusion-distance metric. First, we learn the metric from transition data. Then, we propose a novel auxiliary loss, named **DDSR**, which utilizes the learned metric to shape state representations such that value functions are continuous with respect to the learned representations. The underpinning hypothesis is that such

representation learning that mimics the diffusion-distance (hence respect continuity of the value function) learns a value function approximation able to generalize to unseen states. Empirically, we demonstrate the gain of the proposed auxiliary loss by showing that i) it leads to structural and informative representation, and ii) it improves the control performance in comparison with baseline algorithms.

A similar line of work in RL representation learning [119][120][121] are based on bisimulation-metrics [122], which quantify the "behaviour" similarity between states, and declares two states to be bisimilar if both their immediate rewards and the transition dynamics are similar. These works first estimate the bisimulation metrics from transition data and then shape the representation accordingly. Specifically, states have similar behaviours (low value of bisimulation-metric) are associated with similar representations. Meanwhile, states have dissimilar behaviours (high value of bisimulation-metric) are associated with dissimilar representations. However, we argue that states could have similar values even they are behaviourally dissimilar (large value of bisimulation metric and should associated with similar representations. Empirically, in Fig 6.1 we compare the ground-truth state value distance (a) with $\pi-$bisimulation (b) and bisimulation metrics (c), showing that these metrics are an upper bound of the actual value function. Specifically, it shows that some states have similar values (in blue (a)) are associated with large value of bisimulation metrics (in red (b-c)).

Alternative methods leverage on the dynamic/temporal structure of the Markov Decision Process (MDP) as state representations. For example, SR [118] and Deep successor feature (DSF) [123] use successor representation and successor feature, respectively, as state representation, which measure the future state/state-feature discounted occupancy. PVF [124] use the eigenvectors of Laplacian of the state transition matrix as representations, which encodes the geometry of state space. However, these works are focused on reflecting the agent behavior or MDP structure. In this work, we claim that a state representation should aim at guaranteeing the continuity property of value function over their representation. This is mainly motivated by theoretical works [125], showing that continuity leads to better regret bound – i.e.,

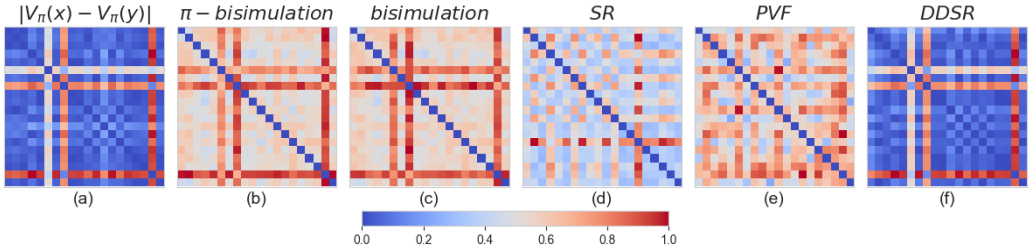


Figure 6.1: Compare the intrinsic structure of state values and structures induced by methods including π-bisumaltion, bisimulation, successor representation (SR), proto-value functions (PVF) and DDSR (this work).

more data-efficiency during exploration. Specifically, continuity guarantee means that states with similar representations have similar values. When this is not the case: a) states with similar representations have different values. It would be challenging for the RL agent to differentiate them; b) states with dissimilar representations have similar values. This will burden the RL agent to learn redundant mapping. Both cases hinder the sample efficiency. In addition, SR, DF, and PVF representations are reward-agnostic, which might result in the good behaviour of generalizing under different reward settings. However, this comes at the price of a less informative representation as value functions are jointly determined by MDP dynamic and rewards. As an empirical illustration, Fig 6.1 shows the induced topology of SR (d) and PVFs (e). A noticeable inconsistency with the value function topology (a) and (d-e) can be observed. In addition, Fig 6.1 (f) also shows the proposed metric (this work), named **DDSR**, which demonstrates a high consistency with the value function topology (a).

In summary, this work has several contributions: (1) to define a *diffusion-distance* metric between states; (2) to prove the Lipschitz continuity of value function over a diffusion-distance metric; (3) to propose a sample-based approach to learn the metric from transition data; (4) to propose an auxiliary loss which leverages the learned metric to generate metric-respect representations; (5) to empirically demonstrate that the proposed auxiliary loss results in an informative and structural representation, leading to improve the control performance in comparison with baseline algorithms.

6.2 Related Work

Auxiliary Task Representation Learning

Auxiliary task was first introduced in [52]. Later, various auxiliary tasks were proposed to improve the representation learning in RL. The main motivation behind auxiliary task is encouraging the learned representation to encode diverse aspects of the underlying MDP. Some recent examples include estimate the return under other rewards [52], policies [117] or discount factor [126], reconstruct the current observation [55], predict the reward function [54] or future observation [53] [8]. Contrastive learning [127] is also used as a form of auxiliary task. The temporal structure [128, 129] , spatial structure [130] and image augmentation [131] have been leveraged to shape representations as contrastive losses. Most works demonstrate the empirical benefits of *Auxiliary tasks* on the performance of RL agents, while the theoretical understanding on the effectiveness of auxiliary tasks is still overlooked [56, 132, 133].

In contrast, this work is built on a clear theoretical interpretation. We introduce a state distance metric upon which value functions are provable Lipschitz continuous and then shape representation according to the metric.

Dynamic-Based Representation Learning

Many works aim to adapt the representation to the dynamic structure of the underlying MDP. Successor representation (SR) [118] and successor feature [123] are used as state representation, which measure the future state/state-feature discounted occupancy. PVF [124] used the eigenvectors of the normalized Laplacian of a undirected graph form by state transitions as representations, which encodes the geometry of state space. Similarly, [134] uses the eigenvector of symmetrized transition matrix and [135] utilizes the singular vectors of transition matrix as representations. These works are reward-agnostic and lack theoretical guarantees on the continuity property of value function over their representations. As already clarified in the introduction, this represents major limitations for the representation learning algorithms. Interestingly, our diffusion-metric distance is derived from SR and yet it captures a very different behavior – Fig 6.1 (d) and (f).

Bisimulation-Based Representation Learning

Bisimulation-metric [136][122], which measures the "behaviour" similarity of states in MDP, was defined as the difference in terms of immediate reward and next state transition distribution. It was used as state abstraction [137] (state aggregation [138]) to reduce the state space by grouping together states that are behaviourally similar. Bisimualtion-metric is expensive to compute, requiring a fully enumeration over state space [139] [140]. To tackle the computation issue, $\pi-$bisimilation metric [139] was proposed along with a sample-based approximate approach. Recent works [119] [120] [121] [141][142] employed the π-bisimulation metric (and its variants) for RL representation learning. The common intuition behind is to learn a representation able to map states close in the latent space if they are behaviourally similar under bisimulation metrics. This could reduce the state space and improve the learning efficiency. However, from the perspective of value function approximation, we argue that states that "behave" dissimilarly (different immediate reward and next state transition) could still have similar values. In this case, states will have large value of bisimulation-metrics and be wrongly assigned to dissimilar representations. This may hinder the learning efficiency in value function approximation and then policy learning. Rather than relying on the "behaviour" similarity metrics, this work is built on a novel state distance metric which characterises the intrinsic structure of value functions. We provide theoretical guarantees that under the diffusion-distance metric, two states close/far-apart to each other have similar/dissimilar values, respectively.

6.3 Preliminaries

We consider the standard Reinforcement Learning (RL) setting where an agent interacts with the environment which is modelled as a Markov Decision Process (MDP). An MDP is described as a five-tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, r, \gamma)$ where $\mathscr{S}$ is a finite set of states, $\mathscr{A}$ is a finite set of actions, $\mathscr{P}$ is the transition function with $\mathscr{P}(s'|s,a)$ denoting the distribution of next state s' given action a is taken in state s. r is the reward function where the expected reward obtained if action a is taken in state s is denoted as $r(s,a)$; and $\gamma \in [0,1)$ is a discount factor.

A policy π is a mapping from states to actions: $\mathscr{S} \rightarrow \mathscr{A}$. The state value function $V_\pi(s)$ under a policy π is defined as the expected discounted cumulative reward received by executing π from state s. Similarly, the state-action value function $Q_\pi(s,a)$ is the expected cumulative reward received by taking action a in state s and following π afterward.

$$V_\pi(s) = \mathbb{E}[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_t \sim \pi(\cdot|s_t)] \tag{6.1}$$

$$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} \mathscr{P}(s'|s,a) V_\pi(s') \tag{6.2}$$

An agent attempts to learn an optimal policy π^* whose value functions are denoted by $V_{\pi^*}(s)$ and $Q_{\pi^*}(s)$, respectively, where $V_{\pi^*} = \max_\pi V_\pi$ and $Q_{\pi^*} = \max_\pi Q_\pi$.

The Q-learning algorithm [43] [44] updates the Q-value in a lookup table iteratively, using the following rule:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q_t(a_t, a) - Q_t(s_t, a_t)] \tag{6.3}$$

In case of large or continuous state/action space, it is infeasible to maintain a lookup table and typically resort to approximate the value functions through function approximation. A common approach is to parameterize the value functions as a linear function of representation:

$$V(s) \approx \phi(s)^T \theta, \;\; or \;\; Q(s,a) \approx \phi(s,a)^T \theta \tag{6.4}$$

where θ are learnable weights, $\phi(\cdot) \in \mathbb{R}^d$ is representation. In deep reinforcement learning (e.g. DQN [3]), ϕ and θ are jointly learned via neural network where ϕ can be viewed as the output of the penultimate layer and θ is the weights of the final layer.

In this work, we aim at learning the representation $\phi(\cdot)$. In later sections, we first investigate the continuity property of value functions and identify a state distance metric upon which value functions are provable Lipschitz continuous. We believe that the metric captures the value function intrinsic structure and then utilizes it to

guide the representation learning and the agent behaviour.

6.4 Value Functions Continuity Property

In this section, we carry out the theoretical characterization of the Lipschitz continuity property of value functions. Then, in the next section, we build on this result to propose a practical method for representation learning.

The Successor Representation (SR) [118] provides us a neat tool to analyze the properties of state value functions. For simplicity, we first focus on the tabular case and then move to continuous state setting. Suppose a MDP $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, r, \gamma)$ with $|\mathscr{S}| = n$ discrete states. Given the state transtion matrix $P_\pi \in \mathbb{R}^{n \times n}$ and state rewards $\mathbf{r} \in \mathbb{R}^n$, the state values $V_\pi \in \mathbb{R}^n$ can be written as:

$$V_\pi = \mathbf{r} + \gamma P_\pi V_\pi = (I - \gamma P_\pi)^{-1}\mathbf{r} = M_\pi \mathbf{r} \tag{6.5}$$

Where $M_\pi = (I - \gamma P_\pi)^{-1}$ is the successor representation (SR).

Let $M_\pi(x)$ denote the x-th row of M_π. The value of state $x \in \mathscr{S}$ can be written as:

$$V_\pi(x) = M_\pi(x)^T \mathbf{r} = \sum_{k=1}^{n} r(k) M_\pi(x,k) \tag{6.6}$$

where $M_\pi(x,k)$ is the pair-wise SR, called *diffusion-distance*, between state pair $(x,k) \in \mathscr{S}$:

$$M_\pi(x,k) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}\{s_t = k\} | s_0 = x] = \sum_{t=0}^{\infty} \gamma^t P_\pi^t(x,k) \tag{6.7}$$

Note that *diffusion-distance* is not invariant with respects to its arguments: $M_\pi(x,k) \neq M_\pi(k,x)$.

$$V_\pi(x) = r(x) + \gamma \sum_{x'} P_\pi(x,x') V_\pi(x') \tag{6.8}$$

An alternative expression of state value $V_\pi(x)$ is defined in Eq. (6.8). Although mathematically equivalently (proof in the Appendix D.1), we emphasise that Eq. (6.6) illustrates clearly the contribution of each reward $r(k)$ to $V_\pi(x)$. Formally, the term

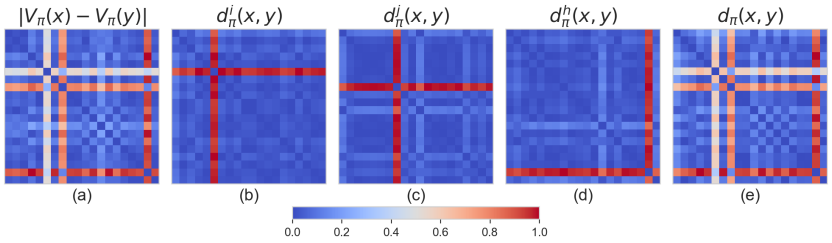


Figure 6.2: Decomposition of value function structure: A toy MDP with $|\mathscr{S}| = 20$ states. There are 3 rewarding states and 17 no-rewarding states. Denote rewarding states as i, j, h. (a): the heatmap of pair-wise value distance $|V_\pi(x) - V_\pi(y)|$. (b): The heatmap of $d^i_\pi(x,y) = |M_\pi(x,i) - M_\pi(y,i)|$. (c): The heatmap of $d^j_\pi(x,y) = |M_\pi(x,j) - M_\pi(y,j)|$. (d): The heatmap of $d^h_\pi(x,y) = |M_\pi(x,h) - M_\pi(y,h)|$. (d): The heatmap of $d_\pi(x,y) = |\sum_{k\in\{i,,j,h\}} \alpha(k)(M_\pi(x,k) - M_\pi(y,k))|$.

$M_\pi(x,k)$ quantifies the discounted state occupancy of k starting from state x. The term $M_\pi(x,k)$ essentially measures the *diffusion-distance* from k to x. Hence, the term $r(k)M_\pi(x,k)$ is the strength of reward $r(k)$ after diffusing from state k to state x. Note the close resemblance to the procedure of computing state value in temporal-difference methods (e.g., TD(0)), where rewards are back-propagated (discounted at each step) from rewarding states k to other states $x \in \mathscr{S}$.

From Eq. (6.6), the value difference between states pair $(x,y) \in \mathscr{S}$ can be written as

$$V_\pi(x) - V_\pi(y) = \sum_{k\in\mathscr{S}_r} r(k)\Big(M_\pi(x,k) - M_\pi(y,k)\Big) \tag{6.9}$$

where $\mathscr{S}_r \subset \mathscr{S}$ is the set of rewarding states $\mathscr{S}_r := \{k : r(k) \neq 0, \forall k \in \mathscr{S}\}$. Note that non-rewarding states $r(k) = 0$ are ignored in Eq. (6.9) as $r(k)M_\pi(x,k) = 0$.

In Definition 1, we define a state distance to characterize the value distance. Lemma 9 establishes the continuity property of state value function with respect to the defined distance.

Definition 1. *(State distance) Under a MDP $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, R, \gamma)$ and a policy π. Suppose $r \in [0, r_{max}]$. Denote $M_\pi(x,k)$ be the diffusion-distance between state pair $(x,k) \in \mathscr{S}$. Denote $\mathscr{S}_r$ as the set of rewarding states. Define $\alpha_k = r(k)/r_{max}$. Given a state-pair $(x,y) \in \mathscr{S}$, we define a pseudo-metric as:*

$$d_\pi(x,y) = |\sum_{k\in\mathscr{S}_r} \alpha_k M_\pi(x,k) - \sum_{k\in\mathscr{S}_r} \alpha_k M_\pi(y,k)| \tag{6.10}$$

Definition 2. *A pseudo-metric on $\mathscr{S}$ is a map $d : \mathscr{S} \times \mathscr{S} \in [0,\infty)$ such that for all $s,s',s'' \in \mathscr{S}$:*

1. $s = s' \implies d(s,s') = 0$
2. $d(s,s') = d(s',s)$
3. $d(s,s'') \leq d(s,s') + d(s',s'')$

The first two conditions are trivial to prove. We focus on the third condition.

Proof.

$$\begin{aligned} d_\pi(x,z) &= |\sum_{k\in\mathscr{S}_r} \alpha_k M_\pi(x,k) - \sum_{k\in\mathscr{S}_r} \alpha_k M_\pi(z,k)| \\ &= |\sum_{k\in\mathscr{S}_r} \alpha_k \Big(M_\pi(x,k) - M_\pi(z,k) \Big)| \\ &= |\sum_{k\in\mathscr{S}_r} \alpha_k \Big(M_\pi(x,k) - M_\pi(y,k) + M_\pi(y,k) - M_\pi(z,k) \Big)| \\ &\leq |\sum_{k\in\mathscr{S}_r} \alpha_k (M_\pi(x,k) - M_\pi(y,k))| + |\sum_{k\in\mathscr{S}_r} \alpha_k (M_\pi(y,k) - M_\pi(z,k))| \\ &= d_\pi(x,y) + d_\pi(y,z) \end{aligned} \tag{6.11}$$

Therefore, $d_\pi(x,y)$ is a pseudo-metric. □

Lemma 9. *(Lipschitz continuity of V-value function). Under the same setting as in Definition 1 and denote $c = r_{max}$, the V-value functions satisfies the following Lipschitz continuity property:*

$$|V_\pi(x) - V_\pi(y)| \leq c d_\pi(x,y) \tag{6.12}$$

See proof in Appendix D.2.

This result establishes a distance metric $d_\pi(x,y)$ under which two states close to each other have similar values. Knowing the continuity property of the value function under the metric allows us to inform how well we can interpolate the value of other states given $V_\pi(x)$. This is essential the generalization ability of the value

function [143]. Lemma 9 should be distinguished from other works in the literature that **assume** the Lipschitz continuity of the value function [144] [145] [146] [54], without giving a specific form of $d_\pi(x,y)$.

In other words, Lemma 1 implies that the topology of the value function is the aggregation of topology induced by each reward. As an empirical illustration, in Fig 6.2, we compare the topology of $|V_\pi(x) - V_\pi(y)|$ with the topology induced by each rewarding state $d_\pi^k(x,y) = |M_\pi(x,k) - M_\pi(y,k)|, k \in \mathscr{S}_r$ and the weighted aggregated topology $d_\pi(x,y)$. It shows clearly the topology of value function can be decomposed into the topology induced by each rewarding state, Fig 6.2 (b-d).

Similarly, pair-wise SR for state-action pair (x,a) is defined as the expected discounted future state occupancy: $M_\pi(x,a;k) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}\{s_t = k\} | s_0 = x, a_0 = a]$. Upon this, we define a state-action distance as:

$$d_\pi(x,a;y,b) = |\sum_{k \in \mathscr{S}^r} \alpha_k M_\pi(x,a;k) - \sum_{k \in \mathscr{S}^r} \alpha_k M_\pi(y,b;k)| \tag{6.13}$$

Lemma 10. *(Lipschitz continuity of Q-value function). Denote $c = r_{max}$, the Q-value function satisfies the following Lipschitz continuity property:*

$$|Q_\pi(x,a) - Q_\pi(y,b)| \leq c d_\pi(x,a;y,b) \tag{6.14}$$

See proof in Appendix D.2.

The Lipschitz continuity of value function $V_\pi(x)$ with respect to $d_\pi(x,y)$ results in the following value function approximation guarantee.

Lemma 11. ***(Value function approximate bound)*** *Under state aggregation, given the metric $d_\pi(x,y)$, the value function approximation is upper bound as follows:*

$$|V_\pi(x) - \hat{V}_\pi(x)| \leq 2c d_\pi(x,y) \tag{6.15}$$

Assume value function is linearly realization $V(x) = \phi(x)^T \theta$ and $||\phi(x) - \phi(y)||_2 =$

$d_\pi(x,y)$. Denote $|\theta| = \ell$. We have

$$|V_\pi(x) - \hat{V}_\pi(x)| = (c+\ell)d_\pi(x,y) \tag{6.16}$$

See proof in Appendix D.3

This Lemma indicates the value function approximation error is determined by the Lipschitz constant c and the metric $d_\pi(x,y)$.

6.5 DDSR: Diffusion-Distance induced State Representation

In the previous section, we establish the continuity property of value functions over the metric $d_\pi(x,y)$, which captures the intrinsic structure of value functions. In this section, we aim to shape the representation according to $d_\pi(x,y)$. In particular, we first introduce a sampled-based approach to learn the metric from the transition data. Next, we propose a novel loss to shape state representations utilizing the learned metric.

6.5.1 Metric Learning

In tabular setting, the *diffusion-distance*, $M_\pi(x,k)$, can be learned following the recursive rule:

$$M_\pi(x,k) = \mathbb{I}\{x=k\} + \gamma M_\pi(x',k) \tag{6.17}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, x' is the next state after x following policy π.

In continuous state setting, we leverage neural network to approximate the *diffusion-distance*. Let $f_w(\cdot) : \mathscr{S} \times \mathscr{S} \to \mathbb{R}$ be a neural network parameterized by w, which takes as input a state observations pair, written $(o(x), o(k)) \in \mathscr{S} \times \mathscr{S}$, and outputs the estimated *diffusion-distance*. Namely,

$$M_\pi(x,k) \approx f_w\big(o(x), o(k)\big) \tag{6.18}$$

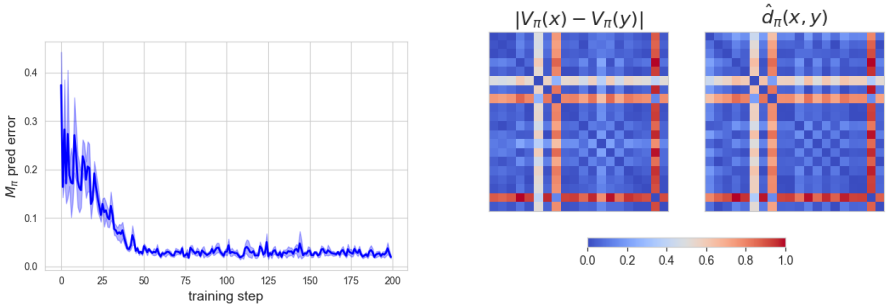


Figure 6.3: The learning of diffusion-distance $M_\pi(x,y)$ and $d_\pi(x,y)$

The loss function used to train $f_w(\cdot,\cdot)$, derived from the recursive rule Eq (6.17), is

$$\mathscr{L}_{SR} = \sum_{(x,k)\in\mathscr{S}\times\mathscr{S}} \Big(f_w\big(o(x),o(k)\big) - \mathbb{I}\{o(x)=o(k)\} - \gamma f_{\bar{w}}\big(o(x'),o(k)\big) \Big)^2 \tag{6.19}$$

The policy dependency of the *diffusion-distance* $M_\pi(x,y)$ suggests that a rapidly change of policy results in rapid change of $M_\pi(x,y)$. To alleviate the instability of learning $M_\pi(x,y)$, we use a delayed target network, a commonly adopted technique in model-free algorithms [3]. In particular, we use the target network in calculating the bootstrapped target with the parameters $\bar{w}$ which is updated periodically.

Given a trained $f_w(\cdot)$, the state distance metric can be approximately measured as

$$\hat{d}_\pi(x,y) = |\sum_{k\in\mathscr{S}^r} \alpha(k) f_{\bar{w}}(x,k) - \sum_{k\in\mathscr{S}^r} \alpha(k) f_{\bar{w}}(y,k)| \tag{6.20}$$

where $\alpha(k) = r(k)/r_{max}$ and $r(k)$ is the observed reward of state k. If r_{max} is unknown, we use the observed maximum reward. It is worth noting that $d_\pi(x,y)$ is only dependent on rewarding states $k \in \mathscr{S}_r$, this is because we motivated the value function being a composition of only the diffused rewarding states. In practice, we maintain a replay buffer, denoted as $\mathscr{D}_r$, which contains only rewarding transitions. At each training iteration, we sample a mini-batch $\mathscr{B}_r \in \mathscr{D}_r$ to estimate $d_\pi(x,y)$ following Eq. (6.20). The sampled mini-batch B_r might contain duplicated rewarding states, which would lead to incorrect estimation of $d_\pi(x,y)$. To circumvent this issue, duplicated states in B_r, are filtered out and keep only one instance for each state.

To demonstrate the validation of the learning procedure. Fig. 6.3 (left) shows the

learning curve of *diffusion-distance* which presents the estimation error $|f_w(x,y) - M_\pi(x,y)|$. Fig 6.3 (right) compare the ground-truth state value distance $|V_\pi(x) - V_\pi(y)|$ (heatmap) and $\hat{d}_\pi(x,y)$ learned via Eq (6.20). It shows that $\hat{d}_\pi(x,y)$ indeed captures the intrinsic structure of value function.

In this section, we have introduced a sample-based approach to learn the *diffusion-distance* which is then used to estimated state distance metric $d_\pi(x,y)$. Next, we show how to utilizes the learned metric for representation learning.

6.5.2 The DDSR Loss

We now present a novel loss to embed the distance metric $\hat{d}_\pi(x,y)$ into the representation space, which can be incorporated with RL agents so as to improve the learning. We first convert the learned distance $\hat{d}_\pi(x,y)$ to a similarity measure $\Gamma_\pi(x,y) \in [0,1]$ by using the Gaussian kernel $\Gamma_\pi(x,y) = \exp(-\hat{d}_\pi(x,y)/\sigma)$ with $\sigma > 0$ be a positive scale parameter controlling the kernel bandwidth. Let $\phi(x), \phi(y)$ denote the representation states pair $(x,y) \in \mathscr{S}$. We define a loss function to shape representations, named $\mathscr{L}_{DDSR}$, as follows:

$$\mathscr{L}_{DDSR} = \arg\min_{\phi} \sum_{(x,y)\in\mathscr{S}} \Big(CosSim(\phi(x),\phi(y)) - \Gamma_\pi(x,y) \Big)^2 \tag{6.21}$$

where $CosSim(\phi(x),\phi(y))$ is the cosine similarity between representations.

Intuitively, $\mathscr{L}_{DDSR}$ aims to match the cosine similarity between state representations with the similarity $\Gamma_\pi(x,y)$ provided by the distance metric $d_\pi(x,y)$. We can adapt $\mathscr{L}_{DDSR}$ as an auxiliary loss in combination with RL agent to learn representation with meaningful semantics. The composite loss function is defined as:

$$\mathscr{L} = \mathscr{L}_{TD} + \alpha \mathscr{L}_{DDSR} \tag{6.22}$$

where α is a hyperparameter and

$$\mathscr{L}_{TD} = \arg\min_{\theta,\phi} \sum_{(s_t,,a_t,r,s_{t+1})\in\mathscr{D}} \Big(r(s_t,a_t) + \gamma \max_a \hat{Q}_\theta(\phi(s_{t+1}),a) - \hat{Q}_\theta(\phi(s_t),a_t) \Big)^2 \tag{6.23}$$

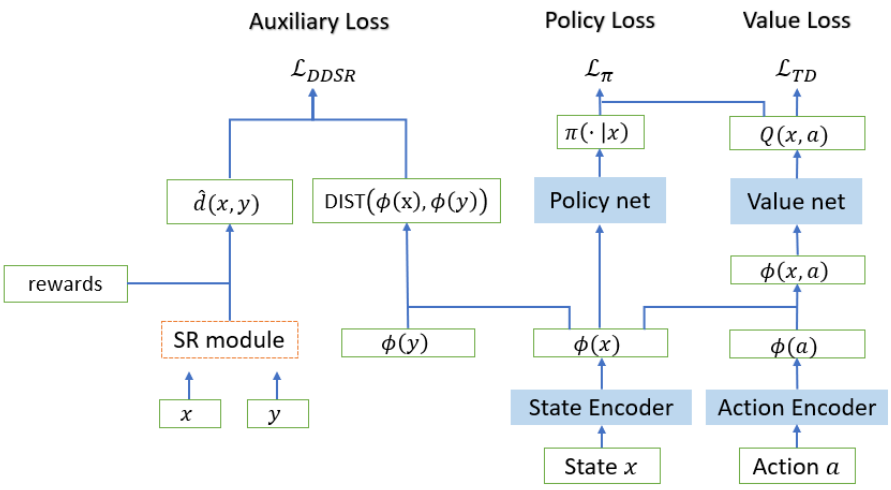


Figure 6.4: Overall Architect

In principle, $\mathscr{L}_{DDSR}$ can be used in combination with any RL agents including value-based and policy-gradient algorithms. Fig 6.4 shows the overall architect and Algorithm 14 contains the pseudocode.

The encoder is trained jointly by $\mathscr{L}_{DDSR}$ and $\mathscr{L}_{TD}$. At each training step, the agent interacts with the MDP and collects transitions in a replay buffer $\mathscr{D}$. A batch is randomly sampled from the replay buffer. To compute $\mathscr{L}_{DDSR}$, we permute the batch to form a batch of state pairs, and minimize the loss defined in Eq. (6.21).

6.6 Experiments

We now study the proposed representation learning strategy from an empirical perspective. We are interested in showing if the proposed auxiliary loss $\mathscr{L}_{DDSR}$ leads to informative representations and improves the learning efficiency of RL agents. In the following experiments, we first test the value function approximation performance. Next, we visualize the learnt representations. Finally, we test its control performance.

6.6.1 Baseline Algorithms

Here, we provide the details of baseline algorithms we used alongside the proposed algorithm in experiments. Baseline algorithms including DQN, DSF, PVF, DBC and MICO. Specifically, DSF[123] refers the deep successor representation.

Algorithm 14: DDSR Algorithm (Value-Based)

Input : discount factor: γ; hyper-parameter: α

Initialization: the encoder $\phi(\cdot)$, SR module $f_w(\cdot,\cdot)$, value module $f_\theta(\cdot)$, replay buffer D.

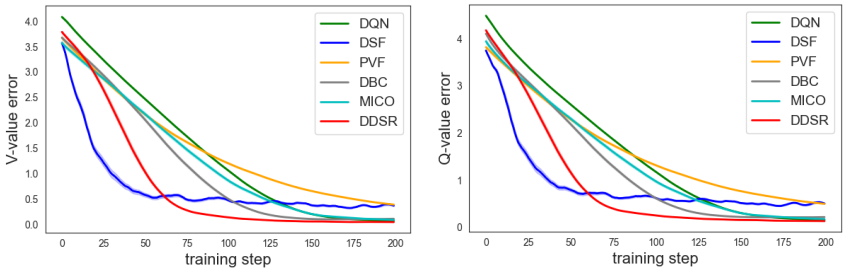
for *each step* **do**

1. Running the current policy and collect transition samples.
2. Store transition samples to replay buffer D.
3. Draw a mini-batch of transition samples B from replay buffer D.
4. Calculate the SR loss $\mathscr{L}_{SR}$ in Eq. (6.19).
5. Update the parameters of SR module $f_w(\cdot,\cdot)$.
6. Estimate the state distance $\hat{d}_\pi(x,y)$ in Eq. (6.20).
7. Calculate the DDSR loss $\mathscr{L}_{DDSR}$ in Eq. (6.21).
8. Calculate the Value loss $\mathscr{L}_{TD}$.
9. Update the parameters of encoder $\phi(\cdot)$ and value module $f_\theta(\cdot)$ via minimizing the compound loss:
$$\mathscr{L} = \mathscr{L}_{TD} + \alpha\mathscr{L}_{DDSR}$$

end

PVF [124] refers to the proto-value functions which is the eigenvectors of the normalized Laplacian of the state transition matrix P. DBC [119] and MICO [120] are algorithms designed on the notion of bisimulation metric [122]. DBC learns a dynamic model $\hat{\mathscr{P}}$ to predict the next state distribution. The state distance is defined as the sum of the difference in immediate rewards and next state distribution: $d_\pi^{dbc}(x,y) = |r_\pi(x) - r_\pi(y)| + \gamma W_2(\hat{\mathscr{P}}(\cdot|x), \hat{\mathscr{P}}(\cdot|y))$. Similarly, MICO parameterises the bisimulation metric as a function of state representation: $d_\pi^{mico}(x,y) = (||\phi(x)||_2 + ||\phi(y)||_2)/2 + \beta\eta(\phi(x),\phi(y)))$ where $\eta(\cdot,\cdot)$ is the angel between vectors.

In the following, we compare these benchmarking solutions with the one proposed in this work. First the evaluation is conducted on the Granet MDPs [147], which allows us to know the ground truth value function, for comparison purposing. We then expand out results to a well-known and deeply used environment, MiniGrid, to validate the proposed method in control settings.



(a) $|V_\pi(x) - \hat{V}_\pi(x)|$ **(b)** $|Q_\pi(x,a) - \hat{Q}_\pi(x,a)|$

Figure 6.5: Prediction Errors

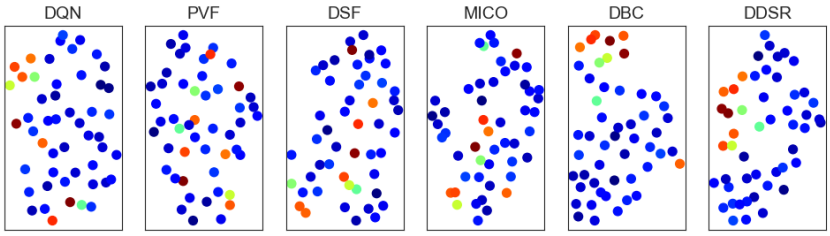


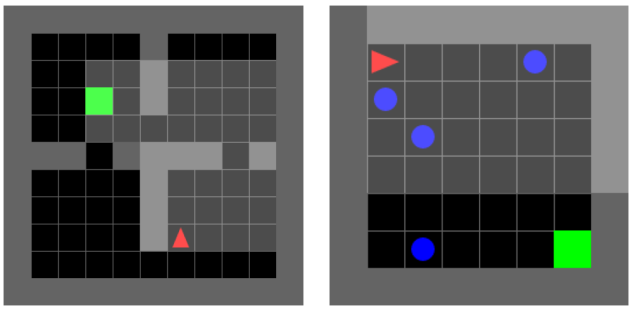
Figure 6.6: Visualization of learned representations

6.6.2 Experiments on Value Function Approximation

In this section, we test algorithms on their performance on value function approximation (policy-evaluation). Experiments are conducted on Granet MDPs [147] where we can compute the exact state values. It allows us to measure the value approximation error. For a fair comparison, all algorithms are trained by the same mini-batches $\mathscr{B}$ sampled from a shared relay buffer $\mathscr{D}$. The replay buffer is filled with transition samples generated by running a fixed random policy π. All algorithms differ only in their auxiliary loss while keeping all other parts the same. They share the same backbone algorithm (DQN, here) and hyper-parameter α. For PVF, the proto-value functions are obtained from the ground-truth state transition matrix, which is used as state representations directly without the need to learn representations.

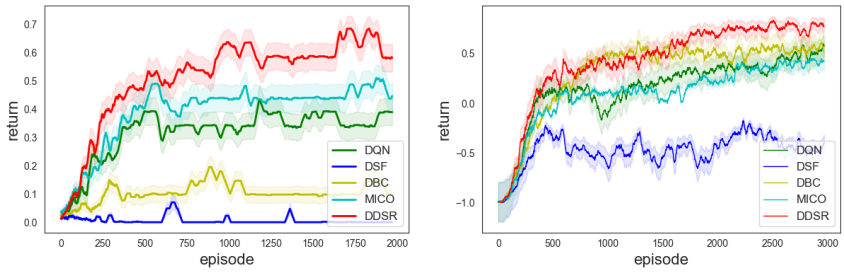
We depict results as the average error of 10 independent generated Random MDPs and 10 random policies for each. Fig. 6.5 (a-b) shows the approximation error of V-value and Q-value, respectively, from which we can see DDSR provides lower error and faster convergence rate.

In Fig. 6.6, we visualize the learned state representations by projecting them



(a) Four-Rooms Maze **(b)** Dynamic Obstacle

Figure 6.7: Mini-Grid Environments



(a) Four-Room Maze **(b)** Dynamic-Obstacles

Figure 6.8: MiniGrid Environment

into 2D dimension space through UMAP [148]. Each dot stands for a state and color represents the corresponding V-value. We can see that with the help of $\mathscr{L}_{DDSR}$, states have similar/dissimilar values are associated with similar/dissimilar representations. In contrast, this is not observed in the learnt representation from other algorithms.

6.6.3 Experiments on Control

In this section, we examine the proposed algorithm on its control performance. We test algorithms on two environments from *Gym-MiniGrid* library [149] where agents are required to first learn a compact state representation based on pixel observations and based which approximate the value function and learn new polices.

Fig. 6.7(a) shows the four-room navigation task in which the agent aims to reach the goal (green spot) and receives a single reward ($r = 1$) upon success. The four-room maze environment represents the spare-reward scenario where a single reward can be obtained only if the agent arrives the goal spot within the maximum step number limitation. Fig. 6.7(b) shows the dynamic-obstacle navigation task

where the agent receives a reward ($r = 1$) if reach the goal (green spot). If the agent collides with a moving obstacle, it receives a penalty $r = -1$ and the episode finishes.

In both environments, DDSR is the most performant algorithm. DDSR employs DQN as its base algorithm, which means the only difference is auxiliary loss. Results show a clear boost of performance demonstrating the benefit of the proposed auxiliary loss. In sparse-reward setting (Four-room maze), MICO and DBC show worse performance compared with DQN and DDSR. Both MICO and DBC assign states with similar/dissimilar representations if these states behave similarly/dis-similarly measures by bisimulation metrics. As mentioned previously, similar valued states behave differently and yet they are associated with similar representations. We conjecture this could be particular true in spare-reward setting. DSF fails in spare-reward environment. DSF first learns a state feature linearly support the immediate reward and then learn successor feature (SF) on the basis of such state features, which is used to fit state value function linearly. In spare-reward setting, most immediate rewards are zero which challenges DSF to learn meaningful state feature and therefore hinder the learning of SF.

6.7 Conclusion

In this work, we have introduced a novel algorithm DDSR for representation learning in RL, which is built on a metric-based auxiliary loss to facilitate the representation learning. The auxiliary loss is theoretically justified where state representations are encouraged to reflect the intrinsic structure of value function. Such structure is encoded by the distance metric $d_\pi(x,y)$. Experimental results demonstrate the superior performance in term of representation learning, policy evaluation and control.

Chapter 7

Graph-Based Recommendation System

After presenting theories and methodologies to address sequential decision making problems, we now provide a use-case application of recommender systems. Similar to previous chapters, we achieve data-efficiency by exploiting structural prior knowledge while learning. In this chapter, we study a recommender system which is modelled as contextual multi-armed bandit (MAB) problem. We propose a graph-based algorithm which learns and exploits clusters of users which are obtained through graph-clustering techniques. To study the impact of graph sparsity and cluster size on the recommendation performance, we conduct exhaustive simulations on both synthetic and in real-world dataset.

7.1 Introduction

Recommending products to users have been an essential function of commercial stores like Amazon and Netflix, etc. [150]. The goal of a recommender system is to propose products based on users' preference. The challenge lies in the fact that the knowledge about users' preference is usually unknown and need to be accumulated by trial and error based on feedbacks of users. This learning process can be formalised as multi-armed bandit (MAB) problem [151, 152, 153, 154]. As already mentioned in the Chapter 1 the performance of MAB learning strategies scales with the ambient dimension, either linearly or as a square root [19], which

makes the problem intractable in scenarios with infinitely large strategy sets, as in recommender system.

To overcome the dimensionality limitation, clustering techniques have been proposed to properly quantize the user space [155]. Users' preference relationships can be encoded as a graph, where adjacent nodes represents users with similar preferences [156, 157]. This graph may be known a-priori or be inferred based on user behaviour data. We are interested in the latter case. In recent works, the geometrical and irregular structure of user space have been considered in designing recommender algorithms [157, 158, 159, 57, 160, 64, 66, 161]. For example, [158] proposed CLUB, an online clustering strategy where m clusters are optimized for n ($>> m$) users. In [64], a similar idea has been implemented on both user and product space results in COFIBA. In both CLUB and COFIBA, an iterative graph learning process is implemented starting from a fully connected graph. At each time step, edges are deleted if adjacent users show a sufficient difference in terms of their preference. User clusters are then constructed by connected components of the updated graph. This leads to an effective MAB algorithm, but with several limitations: *i*) irreversible edge deletion (edges can only be deleted and cannot be added); *ii*) the number of clusters rapidly increasing with time, which we show not to be the best trend for MABs. In contrast, DYnUCB [67] utilizes K-means to group users dynamically into clusters. However, it requires a pre-defined cluster number K. While in theory K can be optimized with iterative solutions (e.g., elbow method), an appropriate cluster number is typically unknown in practice, hard to guess, and may change dynamically over time.

To overcome those limitations, we propose SCLUB-CD, a novel graph-based MAB algorithm that learns and exploits the geometry of user space. Specifically, at each time step, a user graph is constructed based on estimated users' preference. Then, clusters are constructed by applying community detection algorithm [162] on the graph. In summary, this work makes following contributions:

- We adopt graph clustering techniques to form dynamic user clustering for recommender system.

- We show the proposed algorithm maintains a better control over the cluster number, an opposite behaviour with respect to CLUB.
- We test the proposed algorithm in both synthetic and real-world datasets, showing an improved performance over baselines.

7.2 Problem Setting

We now describe the foundation of recommender system and how it can be formalized as a MAB problem . We denote the user set and item (product) set as $\mathscr{U}$. Suppose there are $|\mathscr{U}| = n$ users. Each user $u \in \mathscr{U}$ is associated with an unknown preference vector $\theta_u \in \mathbb{R}^d$. The recommender system is operated by an agent which selects item for users in order to fulfill their preferences. Concretely, at each time step $t \in [T]$, the agent receives an user index i_t, which is assumed to be uniformly sampled form $\mathscr{U}$, and an item set $\mathscr{A}_t = \{x_{1,t}, x_{2,t}, ..., x_{|\mathscr{A}|_t,t}\} \in \mathbb{R}^d$. No assumptions are made on how $\mathscr{A}_t$ and its size $|\mathscr{A}_t|$ are generated. The agent recommends an item a_t from $\mathscr{A}_t$ to user u, and receives a payoff y_t which is defined as a linear function of θ_u and x_{a_t}. Namely,

$$y_t = \theta_u^T x_{a_t} + \varepsilon_t \tag{7.1}$$

Where ε_t is a noise term following a Gaussian distribution. $\varepsilon_t \sim \mathscr{N}(0, \sigma_\varepsilon)$.

Next, we formalize the recommendation task as a Multi-Armed Bandit (MAB) problem. Specifically, the goal of the agent is to maximize the cumulative payoff over the time horizon T. Or equivalently to minimize the cumulative regret with respect to the optimal item selection strategy which always selects the item with the highest payoff for each user. Formally, the agent aims to minimise the cumulative regret R_T over the time horizon T.

$$R_T = \sum_{t=1}^{T} r_t \tag{7.2}$$

where $r_t = \max_{a \in \mathscr{A}_t}(\theta_u^T x_a - \theta_u^T x_{a_t})$.

To solve this problem efficiently, we assume users can be formed into K non-overlapping clusters according to their preference vectors. We suppose the number

of cluster K is unknown. Denote V_k, $k \in [K]$ as the kth user cluster. Intuitively, users belong to the same cluster share similar preference. For example, $||\theta_u - \theta_{u'}|| \leq ||\theta_u - \theta_{u''}||$ if $(u,u') \in V_k$ and $u'' \notin V_k$.

Due to the linearity assumption of payoff, users belonging to the same cluster share similar payoffs to a given item. It follows that rather than learning a preference vector for each user, the agent can learn a preference vector for each cluster. Consequently, the agent only needs to learn K rather than n preference vectors. We denote $\theta_k, k \in [K]$ as the unknown preference vector of cluster V_k.

To solve the bandit problem, we utilize the well-known upper-confidence-bound (UCB) method to balance the exploration and exploitation trade-off. In the next section, we describe the proposed algorithm in detail.

7.3 Graph-Based MAB Algorithm

We now describe the proposed SCLUB-CD algorithm, depicted in Algorithm 15. At each time t, the agent recommends an item a_t to user u and receives a payoff y_t. The agent estimates θ_u via least-squared estimator. Formally,

$$\hat{\theta}_u = (A_u + \alpha I)^{-1} B_u \tag{7.3}$$

where $A_u = \sum_{\{t:u_t=u\}} x_{a_t} x_{a_t}^T$, $B_u = \sum_{\{t:u_t=u\}} y_t x_{a_t}$, and I is the identity matrix.

In addition, at each time step t, the algorithm constructs a user graph $\mathscr{G}_t = (\mathscr{V}_t, \mathscr{E}_t)$ on the basis of estimated preference vectors of all users $\hat{\theta}_u, u \in \mathscr{U}$. The node set $\mathscr{V}_t$ corresponds to the user set $\mathscr{U}$. The edge set $\mathscr{E}_t$ encodes the similarity between users' preference. The graph $\mathscr{G}_t$ is obtained by following a 3-step procedure:

STEP 1. Given $\{\hat{\theta}_u, u \in \mathscr{U}\}$, the edge $w_t(u,u')$ is calculated via Gaussian *RBF* kernel. Precisely,

$$w_t(u,u') = \frac{-||\hat{\theta}_u - \hat{\theta}_{u'}||_2}{w} \tag{7.4}$$

where w is the bandwidth of the Gaussian kernel.

STEP 2. The edge weight $w_t(u,u')$ is then converted into a binary value 1 or 0 in the following way. For each user u, only the edges that connect to top s most

similar neighbors are kept, and the corresponding edge weights are set as 1. The rest edges are set as 0. As a result, each node (user) is only directly connected to s other nodes (users) on graph $\mathcal{G}_t$.

Note that we control the sparsity via setting the number of edges rather than eliminating edges below a pre-defined edge weight threshold. We believe this way provides a better control over the number of clusters which needs to remain low for an efficient learning. In addition, we also believe the binary edge weight is more robust to the estimation error of $\hat{\theta}_u$. Both aspects will be discussed in the experiment section.

STEP 3. Given the constructed graph $\mathcal{G}_t$, user clusters are formed by applying community detection (Louvain Method [162]) to $\mathcal{G}_t$. Specifically, Louvain Method takes $\mathcal{G}_t$ as an input and outputs a set of non-overlapping clusters $\{V_1, .., V_k\}$.

Now, we are ready to introduce the item selection strategy of SCLUB-CD. At each time step t, a random user $u \sim \mathcal{U}$ appears to be served, which belongs to cluster V_k. The agent selects an item a_t for this user following the rule below:

$$a_t = \arg\max_{a \in \mathcal{A}_t} \hat{\theta}_{V_k}^T x_a + \beta ||x_a||_{M_k^{-1}} \tag{7.5}$$

where $M_k = \sum_{u:u \in V_k} A_u$ and β is a constant controls the degree of exploration.

The term $\hat{\theta}_{V_k}$ is the estimated preference of cluster k which is obtained by

$$\hat{\theta}_{V_k} = (M_k + \alpha I)^{-1} B_k \tag{7.6}$$

where $B_k = \sum_{t:u_t \in V_k} y_t x_t$.

Remark 7. *In this work, user clusters are obtained by Louvain Method which detects communities from the user graph. We note that other clustering techniques can also be used. For example, spectral clustering [163]. Viewing users within the same cluster as a single user (as this work) might lose the distinction between users. Instead of grouping users into clusters, the constructed graph structure $\mathcal{G}_t$ can also be exploited by Laplacian-regularized estimator as shown in Chapter 3. We left this extension as a future work.*

Algorithm 15: SCLUB-CD Algorithm

Initial:

1. $B_u = \mathbf{0} \in \mathbb{R}^d$, $A_u = \mathbf{0} \in \mathbb{R}^{d \times d}$ for $u \in \mathscr{U}$
2. $\mathscr{G}_0$ a empty graph.

Input: edge deletion parameter s.
for $t = 1, 2, ..., T$ **do**

1. A random user $u \in \mathscr{U}$ to serve.
2. Find the user clustering $u \in V_k$ via applying Louvian-method on $\mathscr{G}_t$.
3. Calculate $\hat{\theta}_{V_k}$ via Eq. 7.6.
4. Select the item a_t according to Eq. 7.5.
5. Receive the payoff y_t.
6. Update A_u, B_u and $\hat{\theta}_u$ via Eq. 7.3
7. Update $\mathscr{G}_t$ via Eq. 7.4.

end

7.4 Experiments

We carried out experiments on both synthetic and real-world datasets. The synthetic dataset allows us to simulate scenarios where we can control the similarity among users (user graph structure). The real-world datasets are tested to validate the proposed algorithm in realistic scenarios.

In the synthetic case, $n = 100$ users are clustered in $K = 5$ clusters and $m = 1000$ products are considered. We set the dimensionality as $d = 25$. To control the similarity among users within a cluster, an intra-cluster noise σ_k is introduced. For each cluster V_k, we first generate θ_k. For each user $u \in V_k$, the preference vector θ_u is generated by perturbing the θ_k with a white noise vector whose elements are drawn independently from $\mathscr{N}(0, \sigma_k)$. Smaller σ_k means more compact cluster. Regarding the payoff, the noise term is sampled from $\varepsilon \in \mathscr{N}(0, \sigma_\varepsilon)$.

In realistic scenarios, we consider LastFM dataset which contains tags of artists and listening records of users. Delicious dataset contains URLs bookmarked and tags provided by users [164]. LastFM dataset represents “few-hits" scenario

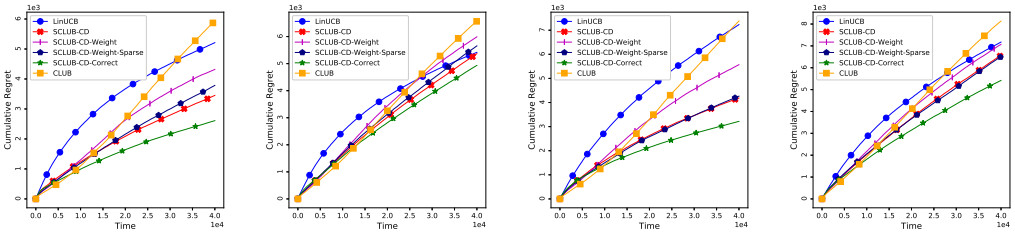


Figure 7.1: Cumulative regret for the synthetic dataset

where users' preference are coherent (therefore it is reasonable to assume that users can be clustered). Delicious dataset represents a "many-hits" scenario where users' preferences are diverse (therefore the clustering is a strong approximation). Simulation results are averaged over 10 runs.

LastFM and Delicious were processed following the same procedure as in [57]. First, tags contained in datasets were breakdown into single words by removing underscore, hyphens and apexes. Second, tags that appear less than 10 times were removed. Third, all tags related to each item was formed as a TF-IDF vector to represent the item. To reduce the dimension, PCA was applied to TF-IDF vectors retaining the top 25 principal components.

The proposed `SCLUB-CD` is compared to several baselines, namely `LinUCB` [35], `CLUB`[158]. In the synthetic dataset, we also provide results for the `SCLUB-CD` where the ground-truth user clusters is known. We label this algorithm as `SCLUB-CD-Correct` which represents a lower bound in terms of cumulative regret. To study the effect of the weighted edge and sparsity, we provide results for two other variants of `SCLUB-CD`: `SCLUB-CD-Weight` keeps all edges and weights. `SCLUB-CD-Weight-Sparse` keeps only the top s edges and their weights.

Fig. 7.1 shows the cumulative regret in the case of synthetic dataset under various levels of reward noise and intra-cluster noise. `SCLUB-CD` outperforms its competitors consistently over all scenarios, with a substantial gap under low intra-cluster noise ($\sigma_k = 0.25$), (Fig. 7.1(a) and Fig. 7.1(b)). It is interesting to observe that under a high intra-cluster noise (Fig. 7.1(c) and Fig. 7.1(d)), `SCLUB-CD` still enjoy a better performance comparing to `LinUCB`, which does not group users into

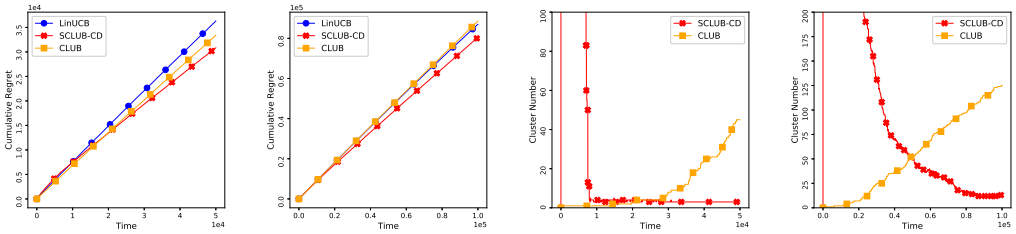


Figure 7.2: Cumulative regret and cluster number for the real-world dataset

clusters.

Comparing to CLUB, the gain is owing to *i*) user clusters are formed by community detection rather than connected components; *ii*) the binary and sparse graph structure. In Fig. 7.1(c), LinUCB might outperform SCLUB-CD for longer time horizon.

Finally, the difference between SCLUB-CD and SCLUB-CD-Correct shows a potential room for improvement. The comparison with SCLUB-CD-Weight and SCLUB-CD-Weight-Sparse shows the benefit of controlling the sparsity level (and therefore number of clusters) in the proposed algorithm.

Fig. 7.2 shows results on real-world datasets. With LastFM, SCLUB-CD maintains a leading margin. We believe this is due to a better clustering methodology. In particular, CLUB tends to group users into many clusters, while SCLUB-CD groups into fewer number of clusters, as shown in Fig. 7.2. This suggests that the proposed approach is able to find a better trade-off between dimensionality reduction and approximation owing to grouping users into clusters.

Under Delicious dataset, SCLUB-CD again outperforms baselines with a small margin. It shows that $K = 11$ user clusters are formed for $n = 700$ users. Delicious dataset represents the “many-hits" scenario, in which each user is interested in a few and similar websites. This means that each user will select few website only, therefore the agent can only gather a few feedbacks per user, which translates into a limited training dataset. Clustering users together allows us to estimate K vectors rather than n vectors. Hence, SCLUB-CD outperforms LinUCB and CLUB.

7.5 Conclusion

In this work, we present a graph-based bandit algorithm, which encodes users preference into a graph, and groups users into clusters. The key aspects of the proposed algorithm are that *i*) it utilizes graph-based clustering to extract meaningful clusters; *ii*) we form an unweighted graph to improve robustness to estimation errors; *iii*) the proposed method maintains a meaningful cluster number. All these components lead to an overall gain in terms of cumulative regret compared with baselines. These results also open new questions such as “What is the sensitivity of the proposed algorithm to the cluster size?", “Could we adopt graph signal processing to further improve the graph knowledge (exploiting also the smoothness of the reward function on the user graph)?". This work has been followed up during my internship at BBC, where the user historical behaviours have been modelled as a random walk process over item graph. We aim to predict the next interested item given past random walk paths. Being still under no-disclosure agreement, I will not be able to release information, but we will soon make the dataset and the novel algorithm publicly accessible.

Chapter 8

Conclusion and Future Work

In this thesis, we made several efforts toward data-efficient sequential decision strategies. We focus on two main open problems, namely the exploration-exploitation dilemma and the representation learning problem, which have critical impacts on learning efficiency.

8.1 Summary of Contributions

The first part of the thesis tackled the exploration-exploitation dilemma under the framework of MAB. In Chapter 3, we designed a novel algorithm for multi-task linear bandits, by leveraging on the existing similarity between bandit problems. Our algorithm provided much tighter confidence bound which than baselines, striking a better balance between exploration and exploitation and resulting in lower regret bounds. We proved this gain theoretically and empirically. In Chapter 4, we theoretically analyzed the error bound of the *Laplacian-regularized estimator*, the core technique used in Chapter 3, and illustrated clearly the benefit of incorporating underlying topological graph structure into the learning process. In Chapter 5, we proposed to learn *upper confidence bound* (UCB) in a data-driven fashion, which was previously derived based from assumptions on reward and noise process. This avoided potential mismatch between assumptions and reality.

In the second part of this thesis, we addressed the data efficiency in RL from the representation learning lens. We identified a topological structure induced by rewards diffusion process and showed that a distance metric reflects such structure . Upon

this insight, we proposed an auxiliary loss to shape state representations exploiting such topology and demonstrated an improved performance in both prediction and control problems.

8.2 Future Works

The research contained in this thesis is an effort toward more data-efficient online sequential learning algorithms, under the reinforcement learning paradigm. Collectively, contributions in the thesis bring us to a point with various promising research directions.

Settings without a prior knowledge

In this thesis, we assume that the structural knowledge is known a priori. For example, in Chapter 4, we assume the similarity relationship between bandit problems is known and leverage such structure in designing algorithms. However, in many practical problems, such structural knowledge is not always available. Thus, it is desirable to infer and learn structures from data. For instance, as shown in [165] graph-topology can be inferred from data under smooth assumption. In RL, arguably the most important structure is the state/state-action topology upon which value functions/polices satisfying certain continuity properties. Identifying and learning such structure provide a promising direction in designing data-efficient RL algorithms. Furthermore, another research direction is to investigate whether the structure and downstream tasks (value prediction or control) can be learned jointly.

Representation beyond value function approximation

In the effort of improving representation learning in RL, we exploit the intrinsic structure of value functions to shape representations. However, representations are not only used for value function approximation but also to generate new policies. A natural question is that whether representations learnt for value function approximation (prediction) are suitable for learning new policies (control)? Indeed, as pointed out in [166], more information is required for value function approximation than to learn the optimal policy. A theoretical understanding on the requirement of representation for prediction and control, respectively, worth a further investigation. Considering

the fact that there is a lack of consensus on the notion of optimal representation for RL, there is a compelling need to design data-efficient RL algorithms able to achieve optimality under different criteria such as value function approximation error, sampling complexity, convergence rate to optimal policy and exploration.

Modelling users' temporal/sequential behaviour in recommender systems

In this thesis, we have designed graph-based algorithms for recommender systems, where we treated users' relationships as static graph. However, in practice, such relationships are constantly changing due to the release of new items, social events, etc. In addition, also the item catalogues are constantly updated by the system provider, leading to a dynamic item graph as well. This open the door to a new line of research on learning the intrinsic temporal/sequential and spatial structure embedded in such dynamic processing to favour better recommendations.

Appendix A

Appendix of Chapter 3

A.1 Proof of Lemma 1

Proof.

$$vec(\hat{\Theta}_t) = (\mathbf{A}_t + \alpha\mathscr{L}_{\otimes})^{-1}\Phi_t\mathbf{Y}_t \tag{A.1}$$

Let $\mathbf{M}_t = \mathbf{A}_t + \alpha\mathscr{L}_{\otimes}$ and $\mathbf{B}_t = \Phi_t\mathbf{Y}_t$, then

$$vec(\hat{\Theta}_t) = \mathbf{M}_t^{-1}\mathbf{B}_t \tag{A.2}$$

Express $\mathbf{M}_t$ and $\mathbf{B}_t$ in partitioned form. For instance, if $n = 2$

$$\mathbf{M}_t^{-1} = \begin{bmatrix} \mathbf{M}_{11,t}^{-1} & \mathbf{M}_{12,t}^{-1} \\ \mathbf{M}_{21,t}^{-1} & \mathbf{M}_{22,t}^{-1} \end{bmatrix} \quad \mathbf{B}_t = \begin{bmatrix} \mathbf{B}_{1,t} \\ \mathbf{B}_{2,t} \end{bmatrix} \tag{A.3}$$

Then

$$\hat{\theta}_{i,t} = \sum_{j=1}^{2} \mathbf{M}_{ij,t}^{-1}\mathbf{B}_{j,t} \tag{A.4}$$

In general case, $n \geq 2$,

$$\hat{\theta}_{i,t} = \sum_{j=1}^{n} \mathbf{M}_{ij,t}^{-1}\mathbf{B}_{j,t} \tag{A.5}$$

To obtain the expression of $\hat{\theta}_{i,t}$, we need the close form of $\mathbf{M}_{ij,t}^{-1}$. Given $\mathbf{M}_t =$

$\mathbf{A}_t + \alpha\mathscr{L}_\otimes$ we have

$$\begin{aligned}\mathbf{M}_t^{-1} &= (\mathbf{A}_t + \alpha\mathscr{L}_\otimes)^{-1} \\ &= (\mathbf{A}_t\mathbf{A}_t^{-1}\mathbf{A}_t + \alpha\mathscr{L}_\otimes\mathbf{A}_t^{-1}\mathbf{A}_t)^{-1} \\ &= \left((\mathbf{I} + \alpha\mathscr{L}_\otimes\mathbf{A}_t^{-1})\mathbf{A}_t\right)^{-1} \\ &= \mathbf{A}_t^{-1}(\mathbf{I} + \alpha\mathscr{L}_\otimes\mathbf{A}_t^{-1})^{-1}\end{aligned} \tag{A.6}$$

This can be rewritten using Taylor expansion

$$\begin{aligned}&\mathbf{A}_t^{-1}(\mathbf{I} + \alpha\mathscr{L}_\otimes\mathbf{A}_t^{-1})^{-1} \\ &= \mathbf{A}_t^{-1}\left(\mathbf{I} - \alpha\mathscr{L}_\otimes\mathbf{A}_t^{-1} + (\alpha\mathscr{L}_\otimes\mathbf{A}_t^{-1})^2 - ...\right) \\ &\approx \mathbf{A}_t^{-1} - \alpha\mathbf{A}_t^{-1}\mathscr{L}_\otimes\mathbf{A}_t^{-1}\end{aligned} \tag{A.7}$$

The last step keeps the first two terms only. So,

$$\mathbf{M}_t^{-1} \approx \mathbf{A}_t^{-1} - \alpha\mathbf{A}_t^{-1}\mathscr{L}_\otimes\mathbf{A}_t^{-1} \tag{A.8}$$

To obtain the expression of $\mathbf{M}_{ij,t}^{-1}$, we need the partitioned form of $\mathbf{A}_t^{-1}$ and $\mathscr{L}_\otimes$.

Since $\mathbf{A}_t$ is a block diagonal matrix, the inversion of it is the inversion of its block matrix. For instance, if $n = 2$

$$\mathbf{A}_t^{-1} = \begin{bmatrix}\mathbf{A}_{1,t}^{-1} & 0 \\ 0 & \mathbf{A}_{2,t}^{-1}\end{bmatrix} \tag{A.9}$$

and

$$\mathscr{L}_\otimes = \begin{bmatrix}\mathscr{L}_{11}\mathbf{I} & \mathscr{L}_{12}\mathbf{I} \\ \mathscr{L}_{21}\mathbf{I} & \mathscr{L}_{22}\mathbf{I}\end{bmatrix} \tag{A.10}$$

Given Eq. (A.8), in general case, $n \geq 2$, it is trivial to show

$$\mathbf{M}_{ij,t}^{-1} \approx \begin{cases}\mathbf{A}_{i,t}^{-1} - \alpha\mathbf{A}_{i,t}^{-1}\mathscr{L}_{ii}\mathbf{A}_{i,t}^{-1} & \textit{when } i = j \\ -\alpha\mathbf{A}_{i,t}^{-1}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1} & \textit{when } i \neq j\end{cases} \tag{A.11}$$

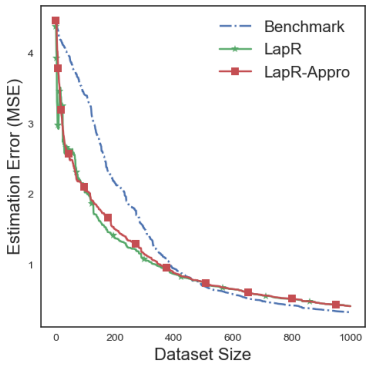


Figure A.1: Approximation accuracy

Finally, we have

$$\begin{aligned}\hat{\theta}_{i,t} &= \sum_{j=1}^{n} \mathbf{M}_{ij,t}^{-1}\mathbf{B}_{j,t} \\ &\approx (\mathbf{A}_{i,t}^{-1} - \alpha\mathbf{A}_{i,t}^{-1}\mathscr{L}_{ii}\mathbf{A}_{i,t}^{-1})\mathbf{B}_{i,t} \\ &- \alpha\mathbf{A}_{i,t}^{-1}\sum_{j\neq i}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\mathbf{B}_{j,t} \\ &= \mathbf{A}_{i,t}^{-1}\mathbf{B}_{i,t} - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\mathbf{B}_{j,t}\end{aligned} \tag{A.12}$$

We claim the approximation introduced in Eq. (A.12) is small. To see this, tracing back to Eq. (A.7) where the higher order terms are dropped. These higher orders terms are negligible when t is large. This is because the Gram matrix $\mathbf{A}_t$ grows with larger t, in turn $\mathbf{A}_t^{-1}$ decreases. In such case, higher order terms would be incomparable with lower order terms.

To support the tightness of this approximation, we provide an empirical evidence in Figure A.1. Red curve represents the single-user estimation error of The Laplacian-regularised estimator Eq. (3.4), while green curve is the estimation error of Lemma 1. Blue curve represents the estimation error of ridge regression, which is included as a benchmark. Clearly, Lemma 1 is a tight approximation of Eq. (3.4) and both converge faster than ridge regression due to the smoothness prior. □

A.2 Proof of Eq. (3.9)

Proof. Let $\mathbf{A}_t = \Phi_t \Phi_t^T$ and $\mathscr{L}_{\otimes} = \mathscr{L} \otimes \mathbf{I}$ and $\mathbf{M}_t = \mathbf{A}_t + \alpha \mathscr{L}_{\otimes}$. $\varepsilon_t = [\eta_1, \eta_2, ..., \eta_t]$

$$vec(\hat{\Theta}_t) = \mathbf{M}_t^{-1} \Phi_t \mathbf{Y}_t \tag{A.13}$$

The variance Σ_t is

$$\begin{aligned} \Sigma_t = Cov(vec(\hat{\Theta}_t)) &= Cov(\mathbf{M}_t^{-1} \Phi_t \mathbf{Y}_t) \\ &= \mathbf{M}_t^{-1} \Phi_t Cov(\mathbf{Y}_t) \Phi_t^T \mathbf{M}_t^{-1} \\ &= \sigma^2 \mathbf{M}_t^{-1} \mathbf{A}_t \mathbf{M}_t^{-1} \end{aligned} \tag{A.14}$$

where we use $Cov(\mathbf{Y}_t) = \sigma^2 \mathbf{I}$ since noise follows $\mathscr{N}(0, \sigma^2)$. Therefore, the precision matrix

$$\Lambda_t = \Sigma_t^{-1} = \frac{1}{\sigma^2} \mathbf{M}_t \mathbf{A}_t^{-1} \mathbf{M}_t \tag{A.15}$$

For simplicity, we assume $\sigma = 1$

$$\Lambda_t = \Sigma_t^{-1} = \mathbf{M}_t \mathbf{A}_t^{-1} \mathbf{M}_t \tag{A.16}$$

□

A.3 Proof of Eq. (3.10)

Proof. Recall

$$\Lambda_t = \mathbf{M}_t \mathbf{A}_t^{-1} \mathbf{M}_t \tag{A.17}$$

and $\Lambda_{i,t} \in \mathbb{R}^{d \times d}$ is the i-th block matrix along the diagonal of Λ_t.

To get the expression of $\Lambda_{i,t}$, we need to express $\mathbf{M}_t$, $\mathbf{A}_t^{-1}$ in partitioned form. For instance, $n = 2$

$$\mathbf{M}_t = \begin{bmatrix} \mathbf{M}_{11,t} & \mathbf{M}_{12,t} \\ \mathbf{M}_{21,t} & \mathbf{M}_{22,t} \end{bmatrix}, \ \mathbf{A}_t^{-1} = \begin{bmatrix} \mathbf{A}_{1,t}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{2,t}^{-1} \end{bmatrix} \tag{A.18}$$

In general case $n \geq 2$, it is straightforward to see

$$\begin{aligned}\Lambda_{i,t} &= \sum_{j=1}^{n} \mathbf{M}_{ij,t}\mathbf{A}_{i,t}^{-1}\mathbf{M}_{ij,t} \\ &= \left(\mathbf{M}_{ii,t}\mathbf{A}_{i,t}^{-1}\mathbf{M}_{ii,t} + \sum_{j\neq i}\mathbf{M}_{ij,t}\mathbf{A}_{j,t}^{-1}\mathbf{M}_{ji,t}\right)\end{aligned} \tag{A.19}$$

From $\mathbf{M}_t = \mathbf{A}_t + \alpha\mathscr{L}_{\otimes}$, we know

$$\mathbf{M}_{ii,t} = \mathbf{A}_{i,t} + \alpha\mathscr{L}_{ii}\mathbf{I} \tag{A.20}$$

$$\mathbf{M}_{ij,t} = \alpha\mathscr{L}_{ij}\mathbf{I} \tag{A.21}$$

Hence,

$$\Lambda_{i,t} = \left(\mathbf{A}_{i,t} + 2\alpha\mathscr{L}_{ii}\mathbf{I} + \alpha^2\sum_{j=1}^{n}\mathscr{L}_{ij}^2\mathbf{A}_{j,t}^{-1}\right) \tag{A.22}$$

□

A.4 Proof of Lemma 2

Proof.

$$\mathscr{C}_t = \{\theta_i : ||\hat{\theta}_{i,t} - \theta_i||_{\Lambda_{i,t}} \leq \beta_{i,t}\} \tag{A.23}$$

From Lemma 1, we have

$$\hat{\theta}_{i,t} \approx \mathbf{A}_{i,t}^{-1}\mathbf{X}_{i,t}\mathbf{Y}_{i,t} - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\mathbf{X}_{j,t}\mathbf{Y}_{j,t} \tag{A.24}$$

Note that $\mathbf{Y}_{i,t} = \mathbf{X}_{i,t}^T\theta_i + \varepsilon_{i,t}$ and $\mathbf{Y}_{j,t} = \mathbf{X}_{j,t}^T\theta_j + \varepsilon_{j,t}$ where $\varepsilon_{i,t} = [\eta_{i,1}, ..., \eta_{i,\mathscr{T}_{i,t}}]$ is the collection of noise associated with user i. $\mathscr{T}_{i,t}$ the set of time user i is selected during the time period from 1 to t.

Then we have

$$\begin{aligned}\hat{\theta}_{i,t} &= \mathbf{A}_{i,t}^{-1}\mathbf{X}_{i,t}(\mathbf{X}_{i,t}^{T}\theta_i + \varepsilon_{i,t}) \\ &\quad - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\mathbf{X}_{j,t}(\mathbf{X}_{j,t}^{T}\theta_j + \varepsilon_{j,t}) \\ &= \theta_i + \mathbf{A}_{i,t}^{-1}\mathbf{X}_{i,t}\varepsilon_{i,t} - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j \\ &\quad - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j\neq i}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\mathbf{X}_{j,t}\varepsilon_{j,t}\end{aligned} \tag{A.25}$$

Hence

$$\begin{aligned}\hat{\theta}_{i,t} - \theta_i &= \mathbf{A}_{i,t}^{-1}\mathbf{X}_{i,t}\varepsilon_{i,t} - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j \\ &\quad - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\mathbf{X}_{j,t}\varepsilon_{j,t}\end{aligned} \tag{A.26}$$

Denote $\xi_{i,t} = \mathbf{X}_{i,t}\varepsilon_{i,t}$ and $\mathbf{V}_{i,t} = \mathbf{A}_{i,t} + \alpha\mathscr{L}_{ii}\mathbf{I}$, then,

$$\begin{aligned}||\hat{\theta}_{i,t} - \theta_i||_{\Lambda_{i,t}} &\leq ||\alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j||_{\Lambda_{i,t}} \\ &\quad + ||\mathbf{A}_{i,t}^{-1}\xi_{i,t} - \alpha\mathbf{A}_{i,t}^{-1}\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\xi_{j,t}||_{\Lambda_{i,t}}\end{aligned} \tag{A.27}$$

$$\begin{aligned}&= \alpha||\sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j||_{\mathbf{A}_{i,t}^{-1}\Lambda_{i,t}\mathbf{A}_{i,t}^{-1}} \\ &\quad + ||\xi_{i,t} - \alpha\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\xi_{i,t}||_{\mathbf{A}_{i,t}^{-1}\Lambda_{i,t}\mathbf{A}_{i,t}^{-1}}\end{aligned} \tag{A.28}$$

Here we apply $||\mathbf{A}_{i,t}^{-1}(\cdot)||_{\Lambda_{i,t}} = ||\cdot||_{\mathbf{A}_{i,t}^{-1}\Lambda_{i,t}\mathbf{A}_{i,t}^{-1}}$.

$$\leq \alpha||\sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j||_{\mathbf{A}_{i,t}^{-1}} + ||\xi_{i,t} - \alpha\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\xi_{i,t}||_{\mathbf{A}_{i,t}^{-1}} \tag{A.29}$$

Here we use $||\cdot||_{\mathbf{A}_{i,t}^{-1}\Lambda_{i,t}\mathbf{A}_{i,t}^{-1}} \leq ||\cdot||_{\mathbf{A}_{i,t}^{-1}}$.

$$\leq \alpha||\sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j||_{\mathbf{V}_{i,t}^{-1}} + ||\xi_{i,t} - \alpha\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\xi_{i,t}||_{\mathbf{V}_{i,t}^{-1}} \tag{A.30}$$

Here we use $||\cdot||_{\mathbf{A}_{i,t}^{-1}} \leq ||\cdot||_{\mathbf{V}_{i,t}^{-1}}$.

$$\leq \alpha||\sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j||_{\mathbf{V}_{i,t}^{-1}} + ||\xi_{i,t}||_{\mathbf{V}_{i,t}^{-1}} \tag{A.31}$$

Here we use

$$||\xi_{i,t} - \alpha\sum_{j=1}^{n}\mathscr{L}_{ij}\mathbf{A}_{j,t}^{-1}\xi_{i,t}||_{\mathbf{V}_{i,t}^{-1}} \leq ||\xi_{i,t}||_{\mathbf{V}_{i,t}^{-1}} \tag{A.32}$$

To support Eq. (A.32), we provide an empirical evidence in Figure A.2.

Denote $\Delta_i = \sum_{j=1}^{n}\mathscr{L}_{ij}\theta_j$, Finally, we have

$$||\hat{\theta}_i - \theta_i||_{\Lambda_{i,t}} \leq \alpha||\Delta_i||_{\mathbf{V}_{i,t}^{-1}} + ||\xi_{i,t}||_{\mathbf{V}_{i,t}^{-1}} \tag{A.33}$$

According to Theorem 2 in [37], we have the following upper bound holds with probability $1-\delta$ for $\delta \in [0,1]$.

$$||\xi_{i,t}||_{\mathbf{V}_{j,t}^{-1}} \leq \sigma\sqrt{2\log\frac{|\mathbf{V}_{i,t}|^{1/2}}{\delta|\alpha\mathbf{I}|^{1/2}}} \tag{A.34}$$

In addition, we known

$$\alpha||\Delta_i||_{\mathbf{V}_{i,t}^{-1}} \leq \sqrt{\alpha}||\Delta_i||_2 \tag{A.35}$$

where we use $||\Delta_i||_{\mathbf{V}_{i,t}^{-1}}^2 \leq \frac{1}{\lambda_{min}}(\mathbf{V}_{i,t})||\Delta_i||_2^2 \leq \frac{1}{\alpha}||\Delta_i||_2$, which means $\alpha||\Delta_i||_{\mathbf{V}_{i,t}^{-1}} \leq \frac{\alpha}{\sqrt{\alpha}}||\Delta_i||_2 = \sqrt{\alpha}||\Delta_i||_2$.

Finally, combine the above two upper bounds, we have the following upper bound holds with probability $1-\delta$ for $\delta \in [0,1]$.

$$||\hat{\theta}_{i,t} - \theta_i||_{\Lambda_{i,t}} \leq \sigma\sqrt{2\log\frac{|\mathbf{V}_{i,t}|^{1/2}}{\delta|\alpha\mathbf{I}|^{1/2}}} + \sqrt{\alpha}||\Delta_i||_2 \tag{A.36}$$

which means

$$\beta_{i,t} = \sigma\sqrt{2\log\frac{|\mathbf{V}_{i,t}|^{1/2}}{\delta|\alpha\mathbf{I}|^{1/2}}} + \sqrt{\alpha}||\Delta_i||_2 \tag{A.37}$$

□

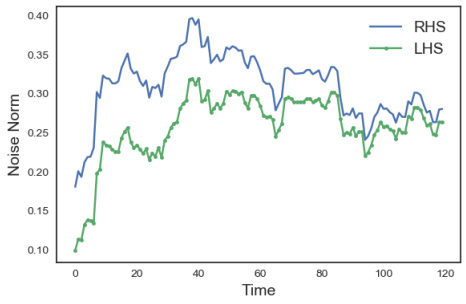


Figure A.2: Noise term approximation

The green curve represents the LHS term in Eq. (A.32), while the blue curve represents the RHS term. Clearly, the blue curve is above the green curve and they converges together with large t.

A.5 Pseudocode of GraphUCB-Local

A.6 Proof of Lemma 3

Recall $\Psi_{i,T} = \frac{\sum_{\tau\in\mathscr{T}_{i,T}}||\mathbf{x}_\tau||^2_{\Lambda^{-1}_{i,\tau}}}{\sum_{\tau\in\mathscr{T}_{i,T}}||\mathbf{x}_\tau||^2_{\mathbf{V}^{-1}_{i,\tau}}}$, where $\mathscr{T}_{i,T}$ is the set of time user i is served up to time T, $\mathbf{A}_{i,\tau} = \sum_{\ell\in\mathscr{T}_{i,\tau}} x_\ell x_\ell^T$, $\mathbf{V}_{i,\tau} = \mathbf{A}_{i,\tau} + \alpha\mathscr{L}_{ii}\mathbf{I}$ and $\Lambda_{i,T}$ defined in Eq. (3.10). Without loss of generality, assume $||\mathbf{x}_\tau||_2 \leq 1$ for any $\tau \leq T$.

Proof.

$$\sum_{\tau\in\mathscr{T}_{i,T}}||\mathbf{x}_\tau||^2_{\mathbf{V}^{-1}_{i,\tau}} \leq (1+\max_{\tau\in\mathscr{T}_{i,T}}||\mathbf{x}_\tau||_2)\log|\mathbf{V}_{i,\tau}| \tag{A.38}$$

In the same fashion

$$\sum_{\tau\in\mathscr{T}_{i,T}}||\mathbf{x}_\tau||^2_{\Lambda^{-1}_{i,\tau}} \leq (1+\max_{\tau\in\mathscr{T}_{i,T}}||\mathbf{x}_\tau||_2)\log|\Lambda_{i,\tau}| \tag{A.39}$$

Algorithm 16: GraphUCB-Local

Input : $\alpha, T, \mathscr{L}, \delta$

Initialization : For any $i \in \{1,2,...,n\}$ $\hat{\theta}_{0,i} = \mathbf{0} \in \mathbb{R}^d$, $\Lambda_{0,i} = \mathbf{0} \in \mathbb{R}^{d\times d}$, $\mathbf{A}_{0,i} = \mathbf{0} \in \mathbb{R}^{d\times d}$, $\beta_{i,t} = 0$.

for $t \in [1,T]$ **do**

User index i is selected

1. $\mathbf{A}_{i,t} \leftarrow \mathbf{A}_{i,t-1} + \mathbf{x}_{t-1}\mathbf{x}_{t-1}^T$.
2. $\mathbf{A}_{j,t} \leftarrow \mathbf{A}_{j,t-1}, \forall j \neq i$.
3. Update $\Lambda_{i,t}$ via Eq. (3.10).
4. Select $\mathbf{x}_t$ via Eq. (3.15) where $\beta_{i,t}$ is defined in Eq. (3.11).
5. Receive the payoff y_t.
6. Update $\hat{\theta}_{i,t}$ via Lemma. 1 if $i = i_t$.
7. $\hat{\theta}_{j,t} \leftarrow \hat{\theta}_{j,t-1} \ \ \forall j \neq i_t$.

end

Since we assume $||\mathbf{x}_\tau||_2 \leq 1$ for any $\tau \leq T$ and

$$\Psi_{i,T} = \frac{\sum_{\tau\in\mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}}}{\sum_{\tau\in\mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{V}_{i,\tau}^{-1}}} \tag{A.40}$$

Given

$$\begin{aligned} \mathbf{V}_{i,\tau} &= \mathbf{A}_{i,\tau} + \alpha\mathbf{I} \\ \Lambda_{i,\tau} &= \mathbf{A}_{i,\tau} + 2\alpha\mathscr{L}_{ii}\mathbf{I} + \alpha^2 \sum_{j=1}^{n} \mathscr{L}_{ij}^2 \mathbf{A}_{j,\tau}^{-1} \end{aligned} \tag{A.41}$$

[1]

Assume[2] $\mathbf{A}_{i,\tau}$ (and $\mathbf{A}_{j,\tau}$) are positive semi-definite for all τ. we know that $\Lambda_{i,\tau} > \mathbf{V}_{i,\tau}$, therefore

$$\Lambda_{i,\tau}^{-1} < \mathbf{V}_{i,\tau}^{-1} \tag{A.42}$$

[1] For isolated node, we set $\mathscr{L}_{ii} = 1$, $\mathscr{L}_{ij} = 0, j \neq i$.

[2] This can be ensured trivially by adding $\lambda\mathbf{I}$ to $\mathbf{A}_{i,\tau}$ with a small λ.

holds for any $\tau \leq T$, which means

$$||\mathbf{x}_\tau||_{\Lambda_{i,\tau}^{-1}} < ||\mathbf{x}_\tau||_{\mathbf{V}_{i,\tau}^{-1}} \tag{A.43}$$

holds for any $\tau \leq T$.

Thus, we have

$$\sum_{\tau \in \mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}} < \sum_{\tau \in \mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{V}_{i,\tau}^{-1}} \tag{A.44}$$

this means

$$\Psi_{i,T} = \frac{\sum_{\tau \in \mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}}}{\sum_{\tau \in \mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{V}_{i,\tau}^{-1}}} < 1 \tag{A.45}$$

In addition, since $||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}} > 0$ and $||\mathbf{x}_\tau||^2_{\mathbf{V}_{i,\tau}^{-1}} > 0$, This must hold

$$\Psi_{i,T} > 0 \tag{A.46}$$

Combine all together, we have

$$\Psi_{i,T} \in (0,1) \tag{A.47}$$

Furthermore, as $\mathbf{A}_{j,\tau}^{-1}$ decreases over time τ, $\Lambda_{i,\tau} \to \mathbf{A}_{i,\tau}$. It means $\Psi_{i,T} \to 1$.

□

A.7 Proof of Theorem 10

Proof. First, we show the instantaneous regret at time t can be upper bounded by $2\beta_{i,t}||\mathbf{x}_{i,t}||_{\Lambda_{i,t}^{-1}}$ where $\mathbf{x}_t$ is the arm selected by the learner at time t for user i. $\mathbf{x}_{i,*}$ is the optimal arm for user i.

$$\begin{aligned} r_{i,t} &= \mathbf{x}_{i,*}^T \theta_i - \mathbf{x}_t \theta_i \\ &\leq \mathbf{x}_t^T \hat{\theta}_{i,t} + \beta_{i,t}||\mathbf{x}_t||_{\Lambda_{i,t}^{-1}} - \mathbf{x}_t \theta_i \\ &\leq \mathbf{x}_t^T \hat{\theta}_{i,t} + \beta_{i,t}||\mathbf{x}_t||_{\Lambda_{i,t}^{-1}} - \mathbf{x}_t^T \hat{\theta}_{i,t} + \beta_{i,t}||\mathbf{x}_t||_{\Lambda_{i,t}^{-1}} \\ &= 2\beta_{i,t}||\mathbf{x}_t||_{\Lambda_{i,t}^{-1}} \end{aligned} \tag{A.48}$$

where we use the principle of optimistic

$$\mathbf{x}_{i,*}^T \theta_i + \beta_{i,t} ||\mathbf{x}_{i,*}||_{\Lambda_{i,t}^{-1}} \leq \mathbf{x}_t^T \hat{\theta}_{i,t} + \beta_{i,t} ||\mathbf{x}_t||_{\Lambda_{i,t}^{-1}} \tag{A.49}$$

and

$$\mathbf{x}_t^T \hat{\theta}_{i,t} \leq \mathbf{x}_t^T \theta_i + \beta_{i,t} ||\mathbf{x}_t||_{\Lambda_{i,t}^{-1}} \tag{A.50}$$

Next, we drive a upper bound of the cumulative regret of user i up to T

$$\begin{aligned} R_{i,T} = \sum_{\tau \in \mathscr{T}_{i,T}} r_{i,\tau} &\leq \sqrt{|\mathscr{T}_{i,T}| \sum_{\tau \in \mathscr{T}_{i,T}} r_{i,\tau}^2} \\ &\leq \sqrt{|\mathscr{T}_{i,T}| \sum_{\tau \in \mathscr{T}_{i,T}} 4\beta_{i,\tau}^2 ||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}}} \\ &\leq 2\beta_{i,T} \sqrt{|\mathscr{T}_{i,T}| \sum_{\tau \in \mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}}} \\ &\leq 2\beta_{i,T} \sqrt{|\mathscr{T}_{i,T}| \sum_{\tau \in \mathscr{T}_{i,T}} min(1, ||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}})} \end{aligned} \tag{A.51}$$

where we user $\beta_{i,T} \geq \beta_{i,\tau}$ since $\beta_{i,\tau}$ is an increasing function over τ and $r_{i,\tau} \leq 2$ since we assume payoff $\mathbf{x}^T \theta_i \in [-1, 1]$.

According to Lemma 11 in [37], we have

$$\begin{aligned} \sum_{\tau \in \mathscr{T}_{i,T}} min(1, ||\mathbf{x}_\tau||^2_{\mathbf{V}_{i,\tau}^{-1}}) &\leq 2 \log \frac{|\mathbf{V}_{i,T}|}{|\alpha \mathbf{I}|} \\ &\leq 2\sqrt{d \log(\alpha + |\mathscr{T}_{i,T}|/d)} \end{aligned} \tag{A.52}$$

where $\mathbf{A}_{i,T} = \sum_{\tau \in \mathscr{T}_{i,T}} \mathbf{x}_\tau \mathbf{x}_\tau^T$, $\mathbf{V}_{i,T} = \mathbf{A}_{i,T} + \alpha \mathscr{L}_{ii} \mathbf{I}$ and $\mathscr{L}_{ii} = 1$.

Recall in Lemma 3, we define

$$\Psi_{i,T} = \frac{\sum_{\tau \in \mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\Lambda_{i,\tau}^{-1}}}{\sum_{\tau \in \mathscr{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{V}_{i,\tau}^{-1}}} \tag{A.53}$$

Therefore

$$\begin{aligned}\sum_{\tau\in\mathscr{T}_{i,T}} min(1,||\mathbf{x}_\tau||^2_{\Lambda^{-1}_{i,\tau}}) &= \Psi_{i,T}\sum_{\tau\in\mathscr{T}_{i,T}}\min(1,||\mathbf{x}_\tau||^2_{\Lambda^{-1}_{i,\tau}}) \\ &\leq 2\Psi_{i,T}\sqrt{d\log(\alpha+|\mathscr{T}_{i,t}|/d)}\end{aligned} \tag{A.54}$$

From Eq. (A.37), we have

$$\beta_{i,T} = \sigma\sqrt{2\log\frac{|\mathbf{V}_{i,T}|^{1/2}}{\delta|\alpha\mathbf{I}|^{1/2}}} + \sqrt{\alpha}||\Delta_i||_2 \tag{A.55}$$

According to Theorem 2 in [37],

$$\begin{aligned}\sigma\sqrt{2\log\frac{|\mathbf{V}_{i,T}|^{1/2}}{\delta|\alpha\mathbf{I}|^{1/2}}} &\leq \sigma\sqrt{d\log\frac{1+|\mathscr{T}_{i,T}|/\alpha}{\delta}} \\ &\leq \mathscr{O}(\sqrt{d\log|\mathscr{T}_{i,T}|})\end{aligned} \tag{A.56}$$

Hence,

$$\beta_{i,t} \leq \mathscr{O}(\sqrt{d\log\mathscr{T}_{i,T}} + \sqrt{\alpha}||\Delta_i||_2) \tag{A.57}$$

Combine this with Eq. (A.54) and Eq. (A.51), we have

$$\begin{aligned}R_{i,T} \leq \mathscr{O}\Bigg(&\left(\sqrt{d\log|\mathscr{T}_{i,T}|} + \sqrt{\alpha}||\Delta_i||_2\right)\times \\ &\Psi_{i,T}\sqrt{d|\mathscr{T}_{i,T}|\log(|\mathscr{T}_{i,T}|)}\Bigg) \\ \leq \tilde{\mathscr{O}}&(\Psi_{i,T}d\sqrt{|\mathscr{T}_{i,T}|})\end{aligned} \tag{A.58}$$

where the constant term $\sqrt{\alpha}||\Delta_i||_2$ and logarithmic terms are hidden.

Assume users are served uniformly, i.e., $\mathscr{T}_{i,T} = T/n$. Then, over the time horizon T, the total cumulative regret experienced by all users satisfies the following

upper bound with probability $1-\delta$ with $\delta \in [0,1]$.

$$\begin{aligned} R_T &= \sum_{i=1}^{n} R_{i,T} = \sum_{i=1}^{n} \tilde{\mathcal{O}}\left(\Psi_{i,T} d\sqrt{|\mathcal{T}_{i,T}|}\right) \\ &= \sum_{i=1}^{n} \tilde{\mathcal{O}}\left(\Psi_{i,T} d\sqrt{T/n}\right) \\ &\leq \tilde{\mathcal{O}}\left(nd\sqrt{T/n}\max_{i\in\mathcal{U}} \Psi_{i,T}\right) \\ &= \tilde{\mathcal{O}}\left(d\sqrt{Tn}\max_{i\in\mathcal{U}} \Psi_{i,T}\right) \end{aligned} \tag{A.59}$$

□

A.8 The quadratic Laplacian form

Given $\mathscr{L} = \mathbf{D}^{-1}\mathbf{L}$ is the random-walk graph Laplacian, where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the combinatorial Laplacian. Recall $\Theta = [\theta_1, \theta_2, ..., \theta_n] \in \mathbb{R}^{n\times d}$ contains user features $\theta_i \in \mathbb{R}^d$ in rows. Then, the quadratic Laplacian form can be expressed in the following way:

$$tr(\Theta^T \mathscr{L}\Theta) = \sum_{k=1}^{d}\sum_{i\sim j} \frac{1}{4}\left(\frac{W_{ij}}{D_{ii}} + \frac{W_{ji}}{D_{jj}}\right)\left(\Theta_{ik} - \Theta_{jk}\right)^2 \tag{A.60}$$

Proof.

$$tr(\Theta^T \mathscr{L}\Theta) = \sum_{k=1}^{d} \Theta_{::k}^T \mathscr{L}\Theta_{::k} \tag{A.61}$$

where $\Theta_{::k} \in \mathbb{R}^n$ is the k-th column of Θ.

Note that $\mathscr{L} = \mathbf{D}^{-1}\mathbf{L}$ is an asymmetric matrix with off-diagonal element $\mathscr{L}_{ij} = -\frac{W_{ij}}{D_{ii}}$ and $\mathscr{L}_{ji} = -\frac{W_{ji}}{D_{jj}}$ and on-diagonal element $\mathscr{L}_{ii} = 1$.

From elementary linear algebra, we know that

$$\mathscr{L} = \frac{\mathscr{L} + \mathscr{L}^T}{2} + \frac{\mathscr{L} - \mathscr{L}^T}{2} \tag{A.62}$$

Then

$$\begin{aligned} tr(\Theta^T \mathscr{L} \Theta) &= tr(\Theta^T (\frac{\mathscr{L}+\mathscr{L}^T}{2} + \frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta) \\ &= tr(\Theta^T (\frac{\mathscr{L}+\mathscr{L}^T}{2})\Theta) + tr(\Theta^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta) \\ &= tr(\Theta^T (\frac{\mathscr{L}+\mathscr{L}^T}{2})\Theta) \end{aligned} \tag{A.63}$$

where

$$tr(\Theta^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta) = 0 \tag{A.64}$$

To see this

$$tr(\Theta^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta) = \sum_{k=1}^{d} \Theta_{::k}^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta_{::k} \tag{A.65}$$

for any $k \in \{1,2,..,d\}$

$$\Theta_{::k}^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta_{::k} = 0 \tag{A.66}$$

To see this, assume $p = \Theta_{::k}^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta_{::k}$, then

$$\begin{aligned} p &= \Theta_{::k}^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta_{::k} \\ &= \left(\Theta_{::k}^T (\frac{\mathscr{L}-\mathscr{L}^T}{2})\Theta_{::k}\right)^T \\ &= -p \end{aligned} \tag{A.67}$$

where $\left(\frac{\mathscr{L}-\mathscr{L}^T}{2}\right)^T = -\frac{\mathscr{L}^T-\mathscr{L}}{2}$. So $p = 0$.

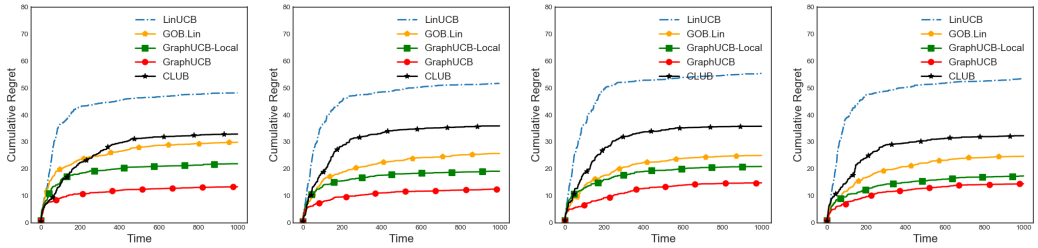
Then,

$$tr(\Theta^T \mathscr{L} \Theta) = tr(\Theta^T (\frac{\mathscr{L}+\mathscr{L}^T}{2})\Theta) \tag{A.68}$$

where the off-diagonal element, $i \neq j$, of $\frac{\mathscr{L}+\mathscr{L}^T}{2}$ is $\frac{1}{2}\left(\frac{W_{ij}}{D_{ii}} + \frac{W_{ji}}{D_{jj}}\right)$ and the on-diagonal element is 1.

Therefore

$$tr(\Theta^T \mathscr{L} \Theta) = \sum_{k=1}^{d} \sum_{i \sim j} \frac{1}{4} \left(\frac{W_{ij}}{D_{ii}} + \frac{W_{ji}}{D_{jj}}\right) \left(\Theta_{ik} - \Theta_{jk}\right)^2 \tag{A.69}$$



(a) RBF (s=0.5) **(b)** ER (p=0.4) **(c)** BA (m=5) **(d)** WS (m=7, p=0.2)

Figure A.3: Performance with respect to Graph Types

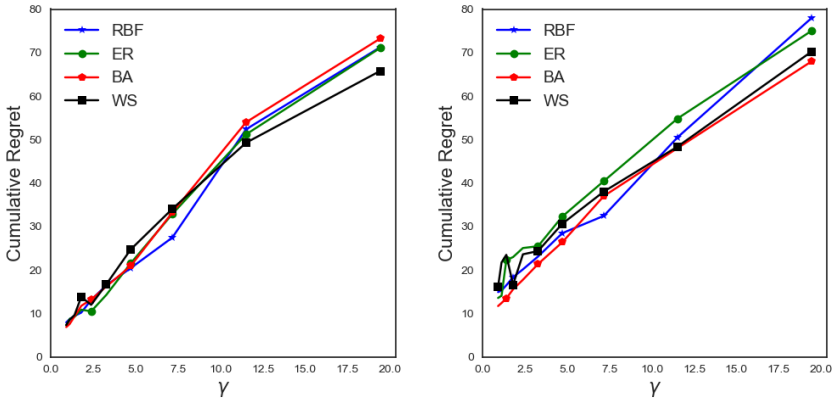
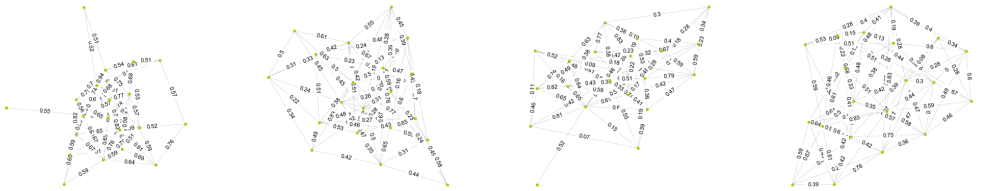


Figure A.4: Performance of proposed algorithms on Graph models

□

To simulate $\mathscr{G}$, we follow two random graph models commonly used in the network science community: 1) Radial basis function (*RBF*) model, a weighted fully connected graph, with edge weights $W_{ij} = exp(-\rho||\theta_i - \theta_j||^2)$; 2) Erdős Rényi (ER) model, an unweighted graph, in which each edge is generated independently and randomly with probability p. 3) *Barabási-Albert* (BA) model, an unweighted graph initialized with a connected graph with m nodes. Then, a new node is added to the graph sequentially with m edges connected to existing nodes following the rule of preferential attachment where existing nodes with more edges has more probability to be connected by the new node; 4) *Watts-Strogatz* (WS) model, an unweighted graph, which is a m-regular graph with edges randomly rewired with probability p. For each graph model, different topologies can be generated, leading to different level of sparsity and smoothness as show in the following

Comparing sub-figures in Fig A.3 shows that with the same level of sparsity and



(a) RBF (s=0.5) **(b)** ER (p=0.4) **(c)** BA (m=5) **(d)** [WS (m=7, p=0.2)

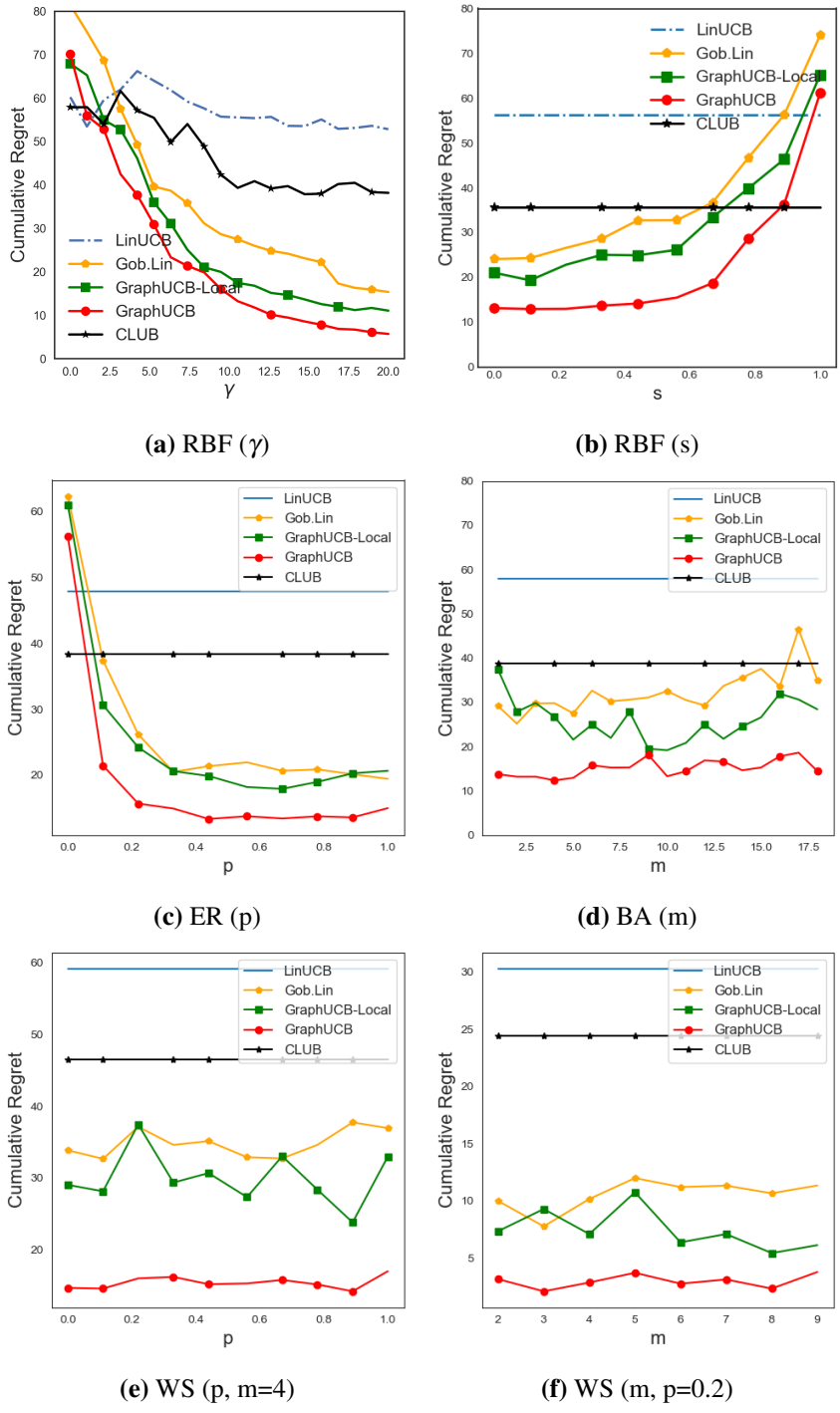
Figure A.5: Graphs

smoothness the effect of topology of graph performance seems to be unnoticeable. The generated graphs are shown in Fig A.5. This is also confirmed in Fig A.4, in which we generate graphs with different topology but with the same level connectivity. Algorithms are test on smoothness level.

A.9 The performance of the proposed algorithms

We test the performance of the proposed algorithms on the basis of graph properties such as smoothness, sparsity, p in ER graph, m in BA graph, m and p in WS graph.
Smoothness [γ]: We first generate a *RBF* graph. To control the smoothness, we vary $\gamma \in [0, 10]$. The term $sm = tr(\Theta^T \mathscr{L} \Theta)$ measures the corresponding smoothness level. To ensure the comparison is fair, Θ is normalized to be $||\Theta||_2 = n$.
Sparsity [s]: We first generate a *RBF* graph, then generate a smooth Θ via Eq. (3.29). To control the sparsity, we set a threshold $s \in [0, 1]$ on edge weights W_{ij} such that W_{ij} less than s are removed. The term $sp = \frac{number of edges}{n(n-1)}$ measures the corresponding level of sparsity, where $n(n-1)$ is the number of edges of a fully connected graph.

Results are shown in Fig A.6. In sub-figure (a) and (b), graph-based algorithms show similar pattern. Smoother signal leads to less regret because of the Laplacian-regularier estimator in Eq. (3.4). Sparse graph leads to more regret as less connectivity provides less graph information. This is also confirmed by sub-figure (c), p in ER controls the probability of edge. Small p leads to spare graph, in turn more regret.



(a) RBF (γ) **(b)** RBF (s)

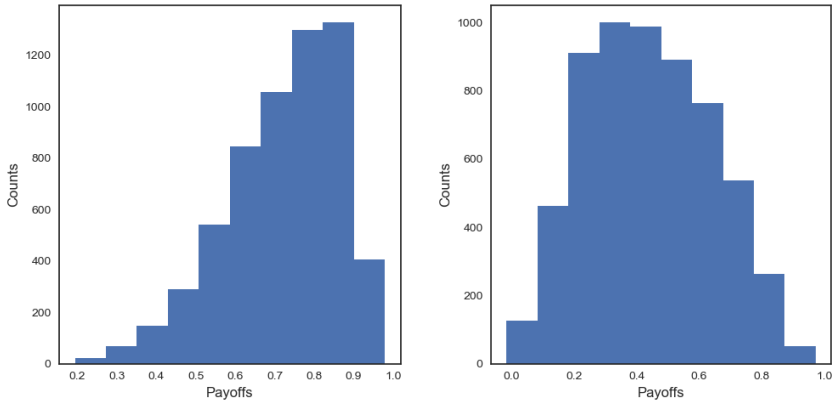
(c) ER (p) **(d)** BA (m)

(e) WS (p, m=4) **(f)** WS (m, p=0.2)

Figure A.6: Performance on Graph Properties

A.10 MovieLens and Netflix Data

Movilens contains 6k users and their ratings on 40k movies. Since every user does not give ratings on all movies, there are a large mount of missing ratings. We factorize the rating matrix via $\mathbf{M} = \mathbf{UX}$ to fill the missing values where $\mathbf{U}$ contain



(a) MovieLens **(b)** Netflix

Figure A.7: Histogram of signals in MovieLens (a) and Netflix (b).

users' latent vectors in rows and $\mathbf{X}$ contain movies' latent features in columns. The dimension is set as $d = 10$. Next, we create the user graph $\mathscr{G}$ from $\mathbf{U}$ via *RBF* kernel. **Netflix** contains rating of 480k users on 18k movies. We process the dataset in the same way as **Movielens**. In both datasets, original ratings range from 0 to 5, we normalize them into $[0, 1]$. After the data pre-processing, we sample 50 users and test algorithms over $T = 1000$.

Figure A.7 shows the distribution of ratings in **Movielens** and **Netflix**. Ratings in **Movielens** are highly concentrated which means a large number of users like a few set of movies. They show similar performance.

Appendix B

Appendix of Chapter 4

B.1 Proofs

Proof.

$$\mathscr{L}(\Theta^*) = \frac{1}{2n}||Y - \Theta^* X||_F^2 \tag{B.1}$$

Thus,

$$\begin{aligned} \bigtriangledown \mathscr{L}(\Theta^*) &= \frac{1}{n}(\Theta^* X - Y)X^T \\ &= \frac{1}{n}\Omega X^T \end{aligned} \tag{B.2}$$

where $\Omega = \Theta^* X - Y$. □

Proof.

$$\begin{aligned} \delta\mathscr{L}(\Theta^*) &= \mathscr{L}(\Theta^* + \triangle) - \mathscr{L}(\Theta^*) - \langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle \\ &= \frac{1}{2n}||Y - (\Theta^* + \triangle)X||_F^2 - \frac{1}{2n}||Y - \Theta^* X||_F^2 - \frac{1}{n}Tr(X\Omega^T \triangle) \end{aligned} \tag{B.3}$$

First, we expand the term $||Y - (\Theta^* + \triangle)X||_F^2 - ||Y - \Theta^* X||_F^2$ as

$$Tr[(Y - (\Theta^* + \triangle)X)^T (Y - (\Theta^* + \triangle)X)] - Tr[(Y - \Theta^* X)^T (Y - \Theta^* X)] \tag{B.4}$$

Which is

$$Tr[X^T \triangle^T \triangle X + 2X^T (\Theta^*)^T \triangle X - 2Y^T \triangle X] \tag{B.5}$$

Next, we expand the last term

$$\begin{aligned} Tr(X\Omega^T \triangle) &= Tr(X(\Theta^* X - Y)^T \triangle) \\ &= Tr(XX^T(\Theta^*)^T \triangle - XY^T \triangle) \end{aligned} \tag{B.6}$$

Then, we have $\delta\mathscr{L}(\Theta^*)$ be

$$\frac{1}{2n}(Tr(X^T \triangle^T \triangle X)) + \frac{1}{n}[Tr(X^T(\Theta^*)^T \triangle X) - Tr(Y^T \triangle X) + Tr(XY^T \triangle) - Tr(XX^T(\Theta^*)^T \triangle)] \tag{B.7}$$

Due to the cyclic property of trace operator, terms are cancelled out except the first term. Therefore,

$$\delta\mathscr{L}(\Theta^*) = \frac{1}{2n}Tr((\triangle X)^T(\triangle X)) = \frac{1}{2n}||\triangle X||_F^2 \tag{B.8}$$

□

B.2 Proof of Theorem 11

Proof. Due to the optimality of $\hat{\Theta}$,

$$\mathscr{L}(\hat{\Theta}) + \alpha\mathscr{R}(\hat{\Theta}) \leq \mathscr{L}(\Theta^*) + \alpha\mathscr{R}(\Theta^*) \tag{B.9}$$

Substituting $\hat{\Theta} = \Theta^* + \triangle$ and arrange the terms, we have

$$\mathscr{L}(\Theta^* + \triangle) - \mathscr{L}(\Theta^*) + \alpha(\mathscr{R}(\Theta^* + \triangle) - \mathscr{R}(\Theta^*)) \leq 0 \tag{B.10}$$

Given $\mathscr{R}(\Theta) = Tr(\Theta^T L\Theta)$, We expand the term $\mathscr{R}(\Theta^* + \triangle) - \mathscr{R}(\Theta^*)$ as

$$\begin{aligned}
\mathscr{R}(\Theta^* + \triangle) - \mathscr{R}(\Theta^*) &= Tr((\Theta^* + \triangle)^T L(\Theta^* + \triangle)) - Tr((\Theta^*)^T L\Theta^*) \\
&= Tr((\Theta^*)^T L\Theta^*) + (\Theta^*)^T L\triangle + \triangle^T L\Theta^* + \triangle^T L\triangle) - Tr((\Theta^*)^T L\Theta^*) \\
&= 2Tr((\Theta^*)^T L\triangle) + Tr(\triangle^T L\triangle) \\
&= 2Tr((\Theta^*)^T L\triangle) + \mathscr{R}(\triangle) \\
&= 2\langle L\Theta^*, \triangle\rangle + \mathscr{R}(\triangle) \\
&\geq -2|\langle L\Theta^*, \triangle\rangle| + \mathscr{R}(\triangle)
\end{aligned} \tag{B.11}$$

Substituting the last inequality, we have

$$\mathscr{L}(\Theta^* + \triangle) - \mathscr{L}(\Theta^*) + \alpha(-2|\langle L\Theta^*, \triangle\rangle| + \mathscr{R}(\triangle)) \leq 0 \tag{B.12}$$

From the definition of $\delta\mathscr{L}(\Theta^*)$ Eq. (4.10), we know that

$$\mathscr{L}(\Theta^* + \triangle) - \mathscr{L}(\Theta^*) = \langle \bigtriangledown\mathscr{L}(\Theta^*), \triangle\rangle + \delta\mathscr{L}(\Theta^*) \tag{B.13}$$

Substituting yields

$$\langle \bigtriangledown\mathscr{L}(\Theta^*), \triangle\rangle + \delta\mathscr{L}(\Theta^*) + \alpha(-2|\langle L\Theta^*, \triangle\rangle| + \mathscr{R}(\triangle)) \leq 0 \tag{B.14}$$

Note that

$$\langle \bigtriangledown\mathscr{L}(\Theta^*), \triangle\rangle \geq -|\langle \bigtriangledown\mathscr{L}(\Theta^*), \triangle\rangle| \tag{B.15}$$

With this, we have

$$-|\langle \bigtriangledown\mathscr{L}(\Theta^*), \triangle\rangle| + \delta\mathscr{L}(\Theta^*) + \alpha(-2|\langle L\Theta^*, \triangle\rangle| + \mathscr{R}(\triangle)) \leq 0 \tag{B.16}$$

Applying the Holder Inequality [101], we have $|\langle \bigtriangledown \mathscr{L}(\Theta^*), \triangle \rangle| \leq || \bigtriangledown \mathscr{L}(\Theta^*)||_\infty || \triangle ||_*$ and $|\langle L\Theta^*, \triangle \rangle| \leq ||L\Theta^*||_F || \triangle ||_F$, thus

$$\delta \mathscr{L}(\Theta^*, \triangle) + \alpha \mathscr{R}(\triangle) \leq || \bigtriangledown \mathscr{L}(\triangle, \Theta^*)||_\infty || \triangle ||_* + 2\alpha ||L\Theta^*||_F || \triangle ||_F \quad \text{(B.17)}$$

If we assume, $\alpha \geq || \bigtriangledown \mathscr{L}(\triangle, \Theta^*)||_\infty$, we have

$$\delta \mathscr{L}(\Theta^*, \triangle) + \alpha \mathscr{R}(\triangle) \leq \alpha || \triangle ||_* + 2\alpha ||L\Theta^*||_F || \triangle ||_F \quad \text{(B.18)}$$

Due to Lemma 4 $tr(\triangle^T L \triangle) \geq \lambda_2 || \triangle ||_F$ and the *strong convexity* condition Eq. (4.10), $\delta \mathscr{L}(\Theta^*) \geq \kappa || \triangle ||_F^2$, we have

$$\kappa || \triangle ||_F^2 + \alpha \lambda_2 || \triangle ||_F^2 \leq \alpha || \triangle ||_* + 2\alpha ||L\Theta^*|| || \triangle ||_F \quad \text{(B.19)}$$

Note the fact that if $rank(\triangle) \leq r$, then $|| \triangle ||_* \leq \sqrt{r} || \triangle ||_F$. Substituting and dividing both sides with $|| \triangle ||_F$ yields

$$|| \triangle ||_F \leq \frac{\alpha(\sqrt{r} + 2||L\Theta^*||_F)}{\kappa + \alpha\lambda_2} \quad \text{(B.20)}$$

□

B.3 Proof of Corollary 1

The estimator is defined in Eq. (4.4) is a ridge estimator applied to estimate the design matrix Θ. We denote it a ridge estimator below if not confusion introduced.

$$\hat{\Theta}_{ridge} = \arg \min_{\Theta \in \mathbb{R}^{m \times k}} \frac{1}{2n} ||Y - \Theta X||_F^2 + \alpha ||\Theta||_F^2 \quad \text{(B.21)}$$

Note that the ridge estimator is equivalent to

$$\hat{\Theta}_{ridge} = \arg \min_{\Theta \in \mathbb{R}^{m \times k}} \frac{1}{2n} ||Y - \Theta X||_F^2 + \alpha tr(\Theta^T I_m \Theta) \quad \text{(B.22)}$$

Where I_m is the identity matrix $I_m \in \mathbb{R}^{m\times m}$.

By following the same arguments in previous section. We have

$$||\hat{\Theta}_{ridge} - \Theta^*||_F \leq \frac{\alpha(\sqrt{r} + 2||\Theta^*||_F)}{\kappa + \alpha} \tag{B.23}$$

Note that $1 = \lambda_1(I_m) = \lambda_2(I_m) = ... = \lambda_m(I_m)$. So, there is no difference between employing $\lambda_1(I_m)$ or $\lambda_2(I_m)$.

B.4 Justification of Assumption 1

Given the definition $\triangle = \hat{\Theta} - \Theta^*$, it is reasonable to expect that entries of $\triangle_j$ varies around 0. *i.e.*, the set $\{\triangle_{j1}, \triangle_{j2}, ..., \triangle_{jm}\}$ consists of both positive, negative real number and 0. Under such condition, it is reasonable to assume $\sum_{i=1}^m \triangle_{ji}^2 \gg \frac{1}{m}(\sum_{i=1}^m \triangle_{ji})^2$ since positive and negative real numbers would cancel out to some extent, which leads to a small $\frac{1}{m}(\sum_{i=1}^m \triangle_{ji})^2$, while $\sum_{i=1}^m \triangle_{ji}^2$ is not affected.

This pattern is expected to be consistent across columns of $\triangle$ so that we assume

$$\sum_{j=1}^{k}\sum_{i=1}^{m} \triangle_{ji}^2 \gg \sum_{j=1}^{k} \frac{1}{m}(\sum_{i=1}^{m} \triangle_{ji})^2 \tag{B.24}$$

Appendix C

Appendix of Chapter 5

C.1 The proof of Lemma 7

Proof. Suppose $S_{i,t} < 0$, that is

$$\beta(||\mathbf{x}_{i_*}||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}) < \hat{\mu}_{i_*,t} - \hat{\mu}_{i,t} \tag{C.1}$$

Rearrange terms gives

$$\hat{\mu}_{i,t} + \beta||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} \leq \hat{\mu}_{i_*,t} - \beta||\mathbf{x}_{i_*}||_{\mathbf{V}_t^{-1}} \tag{C.2}$$

Note that $|\mu_i - \hat{\mu}_{i,t}| \leq \beta||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}$, $\forall i \in \mathscr{A}$. Then,

$$\hat{\mu}_{i_*,t} - \beta||\mathbf{x}_{i_*}||_{\mathbf{V}_t^{-1}} \leq \mu_{i_*} \tag{C.3}$$

and

$$\mu_i \leq \hat{\mu}_{i,t} + \beta||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} \tag{C.4}$$

Combine together we have

$$\mu_i \leq \mu_{i_*} \leq \mu_* \tag{C.5}$$

Recall by definition $i_* = \arg\max_{i \in \mathscr{A}} \hat{\mu}_{i,t} - \beta||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}$ is the arm with largest lower upper bound at round t. Therefore, $\Delta_i = \mu_* - \mu_i > 0$. In words, arm i is suboptimal.

Suppose $S_{j,t} \geq S_{i,t} \geq 0$

$$\beta(||\mathbf{x}_{j_*}||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_j||_{\mathbf{V}_t^{-1}}) - (\hat{\mu}_{j_*,t} - \hat{\mu}_{j,t}) \leq \beta(||\mathbf{x}_{i_*}||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}) - (\hat{\mu}_{i_*,t} - \hat{\mu}_{i,t}) \tag{C.6}$$

Recall the definition of i_*,

$$i_* = \arg\max_{j \in [K]} \hat{\mu}_{j,t} - \beta||\mathbf{x}_j||_{\mathbf{V}_t^{-1}} \tag{C.7}$$

Thus, at each time t, $i_* = j_*$. Then,

$$\beta||\mathbf{x}_j||_{\mathbf{V}_t^{-1}} + \hat{\mu}_{j,t} \leq \beta||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} + \hat{\mu}_{i,t} \tag{C.8}$$

□

C.2 The proof of Lemma 8

Proof.

$$p_{\mathscr{U}_t} = \frac{\sum_{i \in \mathscr{U}_t} \exp(\gamma_t S_{i,t})}{\sum_{i \in \mathscr{U}_t} \exp(\gamma_t S_{i,t}) + \sum_{j \in \mathscr{L}_t} \exp(\gamma_t S_{j,t})} \tag{C.9}$$

By definition, $S_{j,t} < 0, \forall j \in \mathscr{L}$. Thus,

$$\exp(\gamma S_{j,t}) < 1, \ \forall j \in \mathscr{L} \tag{C.10}$$

Then,

$$\sum_{j \in \mathscr{L}_t} \exp(\gamma S_{j,t}) < |\mathscr{L}_t| \tag{C.11}$$

Therefore,

$$p_{\mathscr{U}_t} > \frac{\sum_{i \in \mathscr{U}_t} \exp(\gamma_t S_{i,t})}{\sum_{i \in \mathscr{U}_t} \exp(\gamma_t S_{i,t}) + |\mathscr{L}_t|} \tag{C.12}$$

For any probability $\delta \in (0,1)$, we can find a γ_t such that $p_{\mathscr{U}_t} \geq \delta$, namely

$$\frac{\sum_{i \in \mathscr{U}_t} \exp(\gamma_t S_{i,t})}{\sum_{i \in \mathscr{U}_t} \exp(\gamma_t S_{i,t}) + |\mathscr{L}_t|} \geq \delta \tag{C.13}$$

Rearrange terms gives

$$\sum_{i\in\mathscr{U}_t}\exp(\gamma_t S_{i,t}) \geq \frac{\delta|\mathscr{L}_t|}{1-\delta} \tag{C.14}$$

Take logarithm on both sides,

$$\log\left(\sum_{i\in\mathscr{U}_t}\exp(\gamma_t S_{i,t})\right) \geq \log\left(\frac{\delta|\mathscr{L}_t|}{1-\delta}\right) \tag{C.15}$$

The left side term is LogSumExp which can be approximated by

$$\log\left(\sum_{i\in\mathscr{U}_t}\exp(\gamma_t S_{i,t})\right) \geq \max_{i\in\mathscr{U}_t}\gamma_t S_{i,t} = \gamma_t \max_{i\in\mathscr{U}_t} S_{i,t} \tag{C.16}$$

Denote $\tilde{S}_{\max,t} = \max_{i\in\mathscr{U}_t} S_{i,t}$ and let

$$\gamma_t \tilde{S}_{\max,t} \geq \log(\frac{\delta|\mathscr{L}_t|}{1-\delta}) \tag{C.17}$$

we have

$$\gamma_t \geq \frac{\log(\frac{\delta\mathscr{L}_t}{1-\delta})}{\tilde{S}_{\max,t}} \tag{C.18}$$

Therefore, if γ_t satisfies Eq. (C.18),

$$p_{\mathscr{U}_t} \geq \delta \tag{C.19}$$

Clearly, $p_{\mathscr{L}_t} < 1-\delta$ since $p_{\mathscr{L}_\sqcup} + p_{\mathscr{U}_t} = 1$. □

C.3 The derive of gradients

Proof.

$$\begin{aligned}\max_{\beta} Y(T) = \max_{\beta}\sum_{t=1}^{T}\mathbb{E}[y_t] = \max_{\beta,\gamma}\sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\mu_i \\ s.t.\ \ |\mu_i - \hat{\mu}_{i,t}| - \beta||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} \leq 0,\ \ \forall i \in \mathscr{A},\ \ \forall t \in [T]\end{aligned} \tag{C.20}$$

Apply the Lagrange multipliers, the optimization objective is

$$\max_{\beta} \sum_{t=1}^{T} \sum_{i=1}^{K} p_{i,t}\mu_i - \eta(|\mu_i - \hat{\mu}_{i,t}| - \beta||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}) \;\; s.t. \;\; \eta > 0 \tag{C.21}$$

Apply the score function $\bigtriangledown_\theta f(\theta) = f(\theta) \bigtriangledown_\theta \log f(\theta)$ to $p_{i,t}$

$$\log p_{i,t} = \gamma S_{i,t} - \log \sum_{j=1}^{K} \exp \gamma S_{j,t} \tag{C.22}$$

$$\bigtriangledown_\beta \log p_{i,t} = \gamma_t \phi_{i,t} - \frac{\sum_{j=1}^{K} \gamma_t \phi_{j,t} \exp \gamma_t S_{j,t}}{\sum_{j=1}^{K} \exp \gamma_t S_{j,t}} \tag{C.23}$$

Then, the gradient $g(\beta)$ is

$$g(\beta) = \sum_{t=1}^{T} \sum_{i=1}^{K} \mu_i p_{i,t} \left(\gamma_t \phi_{i,t} - \frac{\sum_{j=1}^{K} \gamma_t \phi_{j,t} \exp \gamma_t S_{j,t}}{\sum_{j=1}^{K} \exp \gamma_t S_{j,t}} \right) + \eta ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} \tag{C.24}$$

The gradient estimator $\hat{g}(\beta)$ is obtained by repalcing μ_i with $\hat{\mu}_{i,t} = \mathbf{x}_i^T \hat{\theta}_t$ where $\hat{\theta}_t = \mathbf{V_t^{-1}} \sum_{s=1}^{t} \mathbf{x}_s y_s$ is obtained via least-square estimator.

$$\hat{g}(\beta) = \sum_{t=1}^{T} \sum_{i=1}^{K} \hat{\mu}_{i,t} p_{i,t} \left(\gamma_t \phi_{i,t} - \frac{\sum_{j=1}^{K} \gamma_t \phi_{j,t} \exp \gamma_t S_{j,t}}{\sum_{j=1}^{K} \exp \gamma_t S_{j,t}} \right) + \eta ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} \tag{C.25}$$

□

C.4 The proof of Theorem 12

Proof. The probability of each arm is defined as

$$p_{i,t} = \frac{\exp(\gamma_t S_{i,t})}{\sum_{j=1}^{K} \exp(\gamma_t S_{j,t})} \tag{C.26}$$

$S_{i,t}$ is defined as

$$S_{i,t} = \hat{\beta}\phi_{i,t} - \hat{\Delta}_{i,t} = \hat{\beta}(||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_{i_*}||_{\mathbf{V}_t^{-1}}) - (\hat{\mu}_{i_*,t} - \hat{\mu}_{i,t}) \tag{C.27}$$

The cumulative regret to be minimized is defined as

$$\begin{aligned} R_T = \sum_{t=1}^{T} \mathbb{E}[r_t] = \sum_{t=1}^{T} \mu_* - \mathbb{E}[y_t] = \sum_{t=1}^{T} (\mu_* - \sum_{i=1}^{K} p_{i,t}\mu_i) \\ = \sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}(\mu_* - \mu_i) = \sum_{t=1}^{T}\sum_{i=1}^{K} p_{i,t}\Delta_i \end{aligned} \tag{C.28}$$

where we use $\sum_{i=1}^{K} p_{i,t} = 1$.

At each time t, trm set $\mathscr{A}$ is divided into two subsets $\mathscr{U}_t$ and $\mathscr{L}_t$ with $\mathscr{U}_t \cup \mathscr{L}_t = \mathscr{A}$. Arm $i \in \mathscr{U}_t$ if $S_{i,t} \geq 0$ and arm $i \in \mathscr{L}_t$ if $S_{i,t} < 0$.

$$\mathbb{E}[r_t] = \sum_{i=1}^{K} p_{i,t}\Delta_i = \sum_{i\in\mathscr{U}_t} p_{i,t}\Delta_i + \sum_{i\in\mathscr{L}_t} p_{i,t}\Delta_i \tag{C.29}$$

Suppose γ_t follows Lemma 8, then $\sum_{i\in\mathscr{L}_t} p_{i,t} < 1-\delta$. Assume $\Delta_i \leq 1, \forall i \in \mathscr{A}$. Then,

$$\mathbb{E}[r_t] = \sum_{i\in\mathscr{U}_t} p_{i,t}\Delta_i + \sum_{i\in\mathscr{L}_t} p_{i,t} \leq \sum_{i\in\mathscr{U}_t} p_{i,t}\Delta_i + (1-\delta) \tag{C.30}$$

By setting $\delta \approx 1$, we have $1-\delta \approx 0$. It means arms in $\mathscr{L}_t$ are unlikely to be selected. So, the second term can be dropped. Therefore,

$$\mathbb{E}[r_t] \leq \sum_{i\in\mathscr{U}_t} p_{i,t}\Delta_i \tag{C.31}$$

Thus,

$$\mathbb{E}[r_t] \leq \sum_{i\in\mathscr{U}_t} p_{i,t}\Delta_i = \sum_{i\in\mathscr{U}_t} p_{i,t}(\mu_* - \mu_i) \tag{C.32}$$

Note that at each time t, $|\hat{\mu}_{i,t} - \mu_i| \leq \hat{\beta}||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}$, $\forall i \in [K]$. Then

$$\mu_* \leq \hat{\mu}_{*,t} + \hat{\beta}||\mathbf{x}_*||_{\mathbf{V}_t^{-1}} \tag{C.33}$$

and

$$\mu_i \geq \hat{\mu}_{i,t} - \hat{\beta}||\mathbf{x}_i||_{\mathbf{V}_t^{-1}} \tag{C.34}$$

Thus,

$$\mu_* - \mu_i \leq \hat{\beta}(||\mathbf{x}_*||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}) + (\hat{\mu}_{*,t} - \hat{\mu}_{i,t}) \tag{C.35}$$

Note that $\hat{\mu}_{*,t} - \hat{\mu}_{i,t} \leq \hat{\mu}_{i_*,t} - \hat{\mu}_{i,t}$ where $i_* = \arg\max_{j\in[K]} \hat{\mu}_{j,t} - \hat{\mu}_{i,t}$. Therefore,

$$\mu_* - \mu_i \leq \hat{\beta}(||\mathbf{x}_*||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}) + (\hat{\mu}_{i_*,t} - \hat{\mu}_{i,t}) \tag{C.36}$$

Since $i \in \mathscr{U}_t$, $S_{i,t} \geq 0$. That is $\hat{\mu}_{i_*,t} - \hat{\mu}_{i,t} \leq \beta(||\mathbf{x}_*||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}})$. Then,

$$\begin{aligned} \mu_* - \mu_i &\leq \hat{\beta}(||\mathbf{x}_*||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}) + (\hat{\mu}_{i_*,t} - \hat{\mu}_{i,t}) \\ &\leq 2\hat{\beta}(||\mathbf{x}_*||_{\mathbf{V}_t^{-1}} + ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}) \end{aligned} \tag{C.37}$$

Define $\psi_t = \max_{i\in[K]} ||\mathbf{x}_i||_{\mathbf{V}_t^{-1}}$. We have

$$\mu_* - \mu_i \leq 4\hat{\beta}\psi_t \tag{C.38}$$

Plugging this into Eq. (C.32) gives

$$\mathbb{E}[r_t] \leq 4\hat{\beta} \sum_{i\in\mathscr{U}_t} p_{i,t}\psi_t \tag{C.39}$$

Since we assume γ_t follows Lemma 8, we have $p_{\mathscr{U}_t} = \sum_{i\in\mathscr{U}_t} p_{i,t} = \delta$. Therefore,

$$\mathbb{E}[r_t] \leq 4\hat{\beta} \sum_{i\in\mathscr{U}_t} p_{i,t}\psi_t = 4\hat{\beta}\phi_t \sum_{i\in\mathscr{U}_t} p_{i,t} = 4\hat{\beta}\psi_t p_{\mathscr{U}_t} \leq 4\hat{\beta}\delta\psi_t \tag{C.40}$$

Thus, the cumulative regret

$$R_T = \sum_{t=1}^{T} \mathbb{E}[r_t] \leq \sqrt{T\sum_{t=1}^{T} \mathbb{E}[r_t]^2} \leq 4\hat{\beta}\delta\sqrt{T\sum_{t=1}^{T}\psi_t^2} \tag{C.41}$$

From Lemma 12 (stated below), we have

$$\sum_{t=1}^{T} \psi_t^2 \leq 2d\log(\alpha + \frac{T}{d}) \tag{C.42}$$

Plugging in Eq. (C.41),

$$R_T \leq 4\hat{\beta}\delta\sqrt{2Td\log(\alpha+\frac{T}{d})} = \tilde{\mathcal{O}}(\hat{\beta}\sqrt{Td\log(1+\frac{T}{d})}) \tag{C.43}$$

where δ is the probability parameter chosen by user.

Lemma 12. *(Lemma 11 in [37])*

$$\sum_{t=1}^{T} ||\mathbf{x}||^2_{\mathbf{V}_t^{-1}} \leq \log det(\mathbf{V}_t) \leq 2d\log(\alpha+\frac{T}{d}) \tag{C.44}$$

□

C.5 The pseudocodes

Algorithm 17: SoftUCB

Input : β, $\mathscr{A}$, K, T, α.

Initialization : $\mathbf{V}_0 = \alpha\mathbf{I} \in \mathbb{R}^{d\times d}$, $\mathbf{b}_0 = \mathbf{0} \in \mathbb{R}^d$, $\hat{\theta}_0 = \mathbf{0} \in \mathbb{R}^d$, $\gamma_0 = 0$.

for $t \in [1,T]$ **do**

1. Find $S_{i,t}, \forall i \in \mathscr{A}$ via Eq. (5.7) with β.
2. Find π_t via Eq. (5.8) with γ_{t-1}.
3. Select arm $i_t \in \mathscr{A}$ randomly following π_t and receive payoff y_t.
4. Update $\mathbf{V}_t \leftarrow \mathbf{V}_t + \mathbf{x}_t\mathbf{x}_t^T$, $\mathbf{b}_t \leftarrow \mathbf{b}_{t-1} + \mathbf{x}_t y_t$ and $\hat{\theta}_t = \mathbf{V}_t^{-1}\mathbf{b}_t$.
5. Update γ_t.

end

Algorithm 18: SoftUCB offline

Input : $\mathscr{A}$, K, T, λ, η

Initialization : $\beta_0 = 0$, $\hat{\beta} = 0$.

for $n \in [1,N]$ **do**

1. Run SoftUCB on $\mathscr{A}$ rounds with $\beta = \beta_{n-1}$.
2. Update $\beta_n \leftarrow \beta_{n-1} + \lambda\hat{g}(\beta)$ via Eq. (5.13)

end

Output : $\hat{\beta} \leftarrow \beta_N$

Run SoftUCB on $\mathscr{A}$ with $\beta = \hat{\beta}$.

Algorithm 19: `SoftUCB online`

Input : $\mathscr{A}$, K, T, α, λ, η

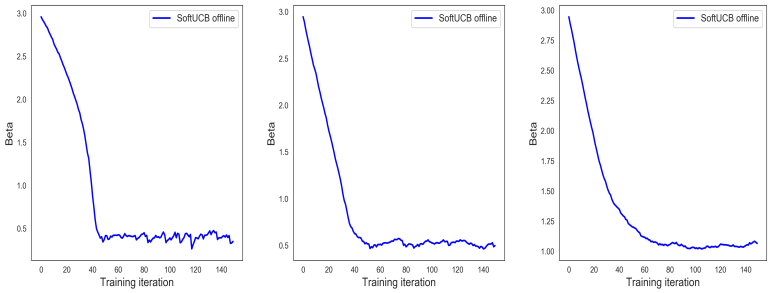
Initialization : $\beta_0 = 0$, $\mathbf{V}_0 = \alpha I \in \mathbb{R}^{d\times d}$, $\mathbf{b}_0 = \mathbf{0} \in \mathbb{R}^d$, $\hat{\theta}_0 = \mathbf{0} \in \mathbb{R}^d$, $\gamma_0 = 0$.

for $t \in [1,T]$ **do**

1. Select arm $i_t \in [K]$ randomly following π_t and receive payoff y_t.
2. Update $\mathbf{V}_t \leftarrow \mathbf{V}_t + \mathbf{x}_t\mathbf{x}_t^T$, $\mathbf{b}_t \leftarrow \mathbf{b}_{t-1} + \mathbf{x}_t\mathbf{y}_t$ and $\hat{\theta}_t = \mathbf{V}_t^{-1}\mathbf{b}_t$.
3. Update $\beta_t \leftarrow \beta_{t-1} + \lambda\hat{g}_t(\beta)$ via Eq. (5.17).

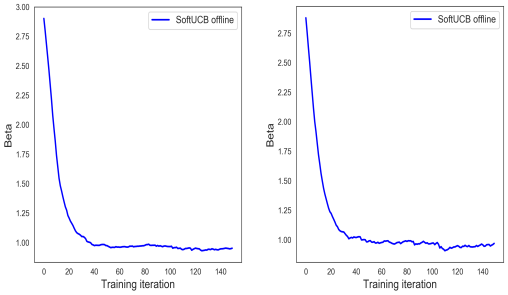
end

C.6 The learning curves of SoftUCB offline



(a) $d = 5, T = 2^8$ **(b)** $d = 5, T = 2^9$ **(c)** $d = 5, T = 2^{10}$

Figure C.1: Learning curves of SoftUCB offline



(a) $d = 10, T = 2^{10}$ **(b)** $d = 15, T = 2^{10}$

Figure C.2: Learning curves of SoftUCB offline

C.7 The dataset Jester

The dataset **Jester** contains ratings of 40 jokes from 19891 users. We sample $K = 50$ users randomly as arms. Their rating to top 39 jokes are used as feature vector. Then, to reduce the sparsity, we apply principle component analysis algorithm to reduce the dimension $d = 10$. Their rating on the 40th jokes are used as rewards. At each round, the algorithm selects on user to recommend the joke and the reward is the rating given by the user. **MovieLens** contains 6k users and their ratings on 40k movies. Since not every user gives ratings on all movies, there are a large mount of missing ratings. We factorize the rating matrix to fill the missing values. The rest works the same as in **Jester**.

Appendix D

Appendix of Chapter 6

D.1 The Proof of value functions

The equivalence between two value function expressions:

$$V_\pi(i) = r(i) + \gamma \sum_{i' \in \mathcal{S}} P_\pi(i,i') V_\pi(i') \tag{D.1}$$

$$V_\pi(i) = \sum_{k \in \mathcal{S}} r(k) M_\pi(i,k) \tag{D.2}$$

From the definition for Successor Representation (SR) [118], we have

$$m_\pi(i,k) = \mathbb{I}\{i=k\} + \gamma \sum_{i' \in \mathcal{S}} P_\pi(i,i') m_\pi(i',k) \tag{D.3}$$

where $\mathbb{I}\{\cdot\}$ denotes the indicator function.

Substituting Eq. (D.3) into Eq. (D.2) results in Eq. (D.1):

$$\begin{aligned} V_\pi(i) &= \sum_{k \in \mathcal{S}} r(k) \left(\mathbb{I}\{i=k\} + \gamma \sum_{i' \in \mathcal{S}} P_\pi(i,i') m_\pi(i',k) \right) \\ &= \sum_{k \in \mathcal{S}} r(k) \mathbb{I}\{i=k\} + \gamma \sum_{i' \in \mathcal{S}} P_\pi(i,i') \sum_{k \in \mathcal{S}} r(k) m_\pi(i',k) \\ &= \sum_{k \in \mathcal{S}} r(k) \mathbb{I}\{i=k\} + \gamma \sum_{i' \in \mathcal{S}} P_\pi(i,i') V_\pi(i') \\ &= r(i) + \gamma \sum_{i' \in \mathcal{S}} P_\pi(i,i') V_\pi(i') \end{aligned} \tag{D.4}$$

D.2 The Proof of Lemma 9

Proof.

$$\begin{aligned}
|V_\pi(x) - V_\pi(y)| &= |\sum_k r(k)(M_\pi(x,k) - M_\pi(y,k))| \\
&= |r_{max} \sum_k \frac{r(k)}{r_{max}}(M_\pi(x,k) - M_\pi(y,k))| \\
&= |r_{max} \sum_k \alpha_k (M_\pi(x,k) - M_\pi(y,k))| \\
&\leq |r_{max}| \cdot |\sum_k \alpha_k (M_\pi(x,k) - M_\pi(y,k))| \\
&= |r_{max}| \cdot |\sum_k \alpha_k M_\pi(x,k) - \sum_k \alpha_k M_\pi(y,k)|
\end{aligned} \tag{D.5}$$

Denote $d_\pi(x,y) = |\sum_k \alpha_k M_\pi(x,k) - \sum_k \alpha_k M_\pi(y,k)|$ and $c = |r_{max}|$, we have

$$|V_\pi(x) - V_\pi(y)| \leq c d_\pi(x,y) \tag{D.6}$$

□

D.3 The proof of Lemma 11

Given the metric $d_\pi(x,y)$, an navie method using it for value function approximation is nearest neighbor approximation. In the below, we show the value function approximation bound of this method. Given $V(x), x \in B$, for $y \neq x, y \in \mathscr{S}$, the state value $V(y)$ can be approximated by nearest-neighbor method: Suppose $x = \arg\min_{x \in B}(V(x) + c d_\pi(x,y))$

$$\hat{V}_\pi(y) = V_\pi(x) + c d_\pi(x,y) \tag{D.7}$$

$$\begin{aligned}
|V_\pi(y) - \hat{V}_\pi(y)| &= |V_\pi(y) - V_\pi(x) + V_\pi(x) - \hat{V}_\pi(y)| \\
&\leq |V_\pi(y) - V_\pi(x)| + |V_\pi(x) - \hat{V}_\pi(y)| \\
&\leq c d_\pi(x,y) + |V_\pi(x) - \hat{V}_\pi(y)| \\
&\leq c d_\pi(x,y) + c d_\pi(x,y) \\
&= 2 c d_\pi(x,y)
\end{aligned} \tag{D.8}$$

Assume value function is linearly realization $V(x) = \phi(x)^T \theta$ and shape the representation to match the metric $||\phi(x) - \phi(y)||_2 = d_\pi(x,y)$. In the following, we show the value function approximation bound. Suppose $||\phi(x) - \hat{\phi}(y)|| = d_\pi(x,y)$, we have

$$\begin{aligned}
\hat{V}_\pi(y) &= \hat{\phi}(y)^T \theta \\
&= (\hat{\phi}(y) - \phi(x) + \phi(x))^T \theta \\
&\leq ||\hat{\phi}(y) - \phi(x)||_2 ||\theta||_2 + V_\pi(x) \\
&= d_\pi(x,y)\ell + V_\pi(x)
\end{aligned} \tag{D.9}$$

$$\begin{aligned}
|V_\pi(y) - \hat{V}_\pi(y)| &= |V_\pi(y) - V_\pi(x) + V_\pi(x) - \hat{V}_\pi(y)| \\
&\leq |V_\pi(y) - V_\pi(x)| + |V_\pi(x) - \hat{V}_\pi(y)| \\
&= |V_\pi(y) - V_\pi(x)| + |\phi(x)^T \theta - \hat{\phi}(y)^T \theta| \\
&\leq c d_\pi(x,y) + ||\phi(x) - \hat{\phi}(y)||_2 ||\theta||_2 \\
&\leq c d_\pi(x,y) + d_\pi(x,y)\ell \\
&= (c + \ell) d_\pi(x,y)
\end{aligned} \tag{D.10}$$

Bibliography

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[4] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[5] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[6] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286*, 2021.

[7] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.

[8] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[9] William R 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.

[10] Aleksandrs Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

[11] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 19–26. JMLR Workshop and Conference Proceedings, 2011.

[12] Donald A Berry and Bert Fristedt. Bandit problems: sequential allocation of experiments (monographs on statistics and applied probability). *London: Chapman and Hall*, 5(71-87):7–7, 1985.

[13] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. Online learning in online auctions. *Theoretical Computer Science*, 324(2-3):137–146, 2004.

[14] Animashree Anandkumar, Nithin Michael, Ao Kevin Tang, and Ananthram Swami. Distributed algorithms for learning and cognitive medium access with logarithmic regret. *IEEE Journal on Selected Areas in Communications*, 29(4):731–745, 2011.

[15] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[16] George Konidaris, Sarah Osentoski, and Philip Thomas. Value function approximation in reinforcement learning using the fourier basis. In *Twenty-fifth AAAI conference on artificial intelligence*, 2011.

[17] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[18] Tze Leung Lai, Herbert Robbins, et al. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[19] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

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

[21] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[22] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[23] Sattar Vakili, Keqin Liu, and Qing Zhao. Deterministic sequencing of exploration and exploitation for multi-armed bandit problems. *IEEE Journal of Selected Topics in Signal Processing*, 7(5):759–767, 2013.

[24] Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on learning theory*, pages 359–376. JMLR Workshop and Conference Proceedings, 2011.

[25] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[26] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135, 2013.

[27] Jean-Yves Audibert and Sébastien Bubeck. Regret bounds and minimax policies under partial monitoring. *The Journal of Machine Learning Research*, 11:2785–2836, 2010.

[28] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

[29] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[30] Sébastien Bubeck and Che-Yu Liu. Prior-free and prior-dependent regret bounds for thompson sampling. *Advances in neural information processing systems*, 26, 2013.

[31] Rajeev Agrawal. The continuum-armed bandit problem. *SIAM journal on control and optimization*, 33(6):1926–1951, 1995.

[32] Stefan Magureanu, Richard Combes, and Alexandre Proutiere. Lipschitz bandits: Regret lower bound and optimal algorithms. In *Conference on Learning Theory*, pages 975–999. PMLR, 2014.

[33] Sébastien Bubeck, Gilles Stoltz, Csaba Szepesvári, and Rémi Munos. Online optimization in x-armed bandits. *Advances in Neural Information Processing Systems*, 21, 2008.

[34] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. 2008.

[35] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[36] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International*

Conference on Artificial Intelligence and Statistics, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[37] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

[38] Akshay Krishnamurthy, Alekh Agarwal, and Miro Dudik. Contextual semibandits via supervised learning oracles. *Advances In Neural Information Processing Systems*, 29, 2016.

[39] Marta Soare, Ouais Alsharif, Alessandro Lazaric, and Joelle Pineau. Multi-task linear bandits. In *NIPS2014 Workshop on Transfer and Multi-task Learning: Theory meets Practice*, 2014.

[40] Aniket Anand Deshmukh, Urun Dogan, and Clay Scott. Multi-task learning for contextual bandits. *Advances in neural information processing systems*, 30, 2017.

[41] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 2019.

[42] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[43] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[44] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[45] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[46] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[47] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[48] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[49] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[50] Marc Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[51] Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7160–7168, 2021.

[52] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768, 2011.

[53] Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

[54] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[55] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[56] Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 1–9. PMLR, 2021.

[57] Nicolo Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. A gang of bandits. In *Advances in Neural Information Processing Systems*, pages 737–745, 2013.

[58] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[59] Tor Lattimore and Csaba Szepesvari. The end of optimism? an asymptotic analysis of finite-armed linear bandits. *arXiv preprint arXiv:1610.04491*, 2016.

[60] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. In *International Conference on Machine Learning*, pages 757–765, 2014.

[61] Bo Liu, Ying Wei, Yu Zhang, Zhixian Yan, and Qiang Yang. Transferable contextual bandit for cross-domain recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[62] Sharan Vaswani, Mark Schmidt, and Laks VS Lakshmanan. Horde of bandits using gaussian markov random fields. *arXiv preprint arXiv:1703.02626*, 2017.

[63] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[64] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 539–548. ACM, 2016.

[65] Kaige Yang and Laura Toni. Graph-based recommendation system. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 798–802. IEEE, 2018.

[66] Nathan Korda, Balazs Szorenyi, and Shuai Li. Distributed clustering of linear bandits in peer to peer networks. In *ICML*, 2016.

[67] Trong T Nguyen and Hady W Lauw. Dynamic clustering of contextual multi-armed bandits. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1959–1962. ACM, 2014.

[68] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 529–538. ACM, 2016.

[69] Michal Valko, Rémi Munos, Branislav Kveton, and Tomáš Kocák. Spectral bandits for smooth graph functions. In *International Conference on Machine Learning*, pages 46–54, 2014.

[70] Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

[71] Tor Lattimore and Csaba Szepesvári. Bandit algorithms. *preprint*, 2018.

[72] Ran Raz. On the complexity of matrix product. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 144–151, 2002.

[73] Yael Yankelevsky and Michael Elad. Dual graph regularized dictionary learning. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):611–624, 2016.

[74] Shyong Lam and Jon Herlocker. Movielens data sets. *Department of Computer Science and Engineering at the University of Minnesota*, 2006.

[75] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.

[76] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*, pages 2107–2115, 2015.

[77] Shuyang Wang, Zhengming Ding, and Yun Fu. Marginalized denoising dictionary learning with locality constraint. *IEEE Transactions on Image Processing*, 27(1):500–510, 2018.

[78] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[79] Francis R Bach. Consistency of trace norm minimization. *Journal of Machine Learning Research*, 9(Jun):1019–1048, 2008.

[80] Hua Zhou and Lexin Li. Regularized matrix regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(2):463–483, 2014.

[81] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. *Journal of Machine Learning Research*, 12(Nov):3371–3412, 2011.

[82] Magdalini Eirinaki, Jerry Gao, Iraklis Varlamis, and Konstantinos Tserpes. Recommender systems for large-scale social networks: A review of challenges and solutions, 2018.

[83] Weiyu Huang, Antonio G Marques, and Alejandro Ribeiro. Matrix completion via graph signal processing. In *Proc. Int. Conf. Acoustics Speech Signal Process., p.(accepted)*, 2018.

[84] Hamid Dadkhahi and Sahand Negahban. Alternating linear bandits for online matrix-factorization recommendation. *arXiv preprint arXiv:1810.09401*, 2018.

[85] Richard H. Bartels and George W Stewart. Solution of the matrix equation ax+ xb= c [f4]. *Communications of the ACM*, 15(9):820–826, 1972.

[86] Kaiyi Ji, Jian Tan, Yuejie Chi, and Jinfeng Xu. Learning latent features with pairwise penalties in matrix completion. *arXiv preprint arXiv:1802.05821*, 2018.

[87] Dengyong Zhou and Bernhard Schölkopf. A regularization framework for learning from graph data. In *ICML workshop on statistical relational learning and Its connections to other fields*, volume 15, pages 67–8, 2004.

[88] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. Signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular data domains. 2012.

[89] Vassilis Kalofolias, Xavier Bresson, Michael Bronstein, and Pierre Vandergheynst. Matrix completion on graphs. *arXiv preprint arXiv:1408.1717*, 2014.

[90] Zhou Zhao, Lijun Zhang, Xiaofei He, and Wilfred Ng. Expert finding for question answering via graph regularized matrix completion. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):993–1004, 2015.

[91] James Sharpnack, Aarti Singh, and Alessandro Rinaldo. Sparsistency of the edge lasso over graphs. In *Artificial Intelligence and Statistics*, pages 1028–1036, 2012.

[92] David Hallac, Jure Leskovec, and Stephen Boyd. Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 387–396. ACM, 2015.

[93] Yu-Xiang Wang, James Sharpnack, Alexander J Smola, and Ryan J Tibshirani. Trend filtering on graphs. *The Journal of Machine Learning Research*, 17(1):3651–3691, 2016.

[94] Jan-Christian Hütter and Philippe Rigollet. Optimal rates for total variation denoising. In *Conference on Learning Theory*, pages 1115–1146, 2016.

[95] Yuan Li, Garvesh Raskutti, and Rebecca Willett. Graph-based regularization for regression problems with highly-correlated designs. *arXiv preprint arXiv:1803.07658*, 2018.

[96] Jiahao Pang and Gene Cheung. Graph laplacian regularization for image denoising: Analysis in the continuous domain. *IEEE Transactions on Image Processing*, 26(4):1770–1785, 2017.

[97] Sahand N Negahban, Pradeep Ravikumar, Martin J Wainwright, Bin Yu, et al. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Statistical Science*, 27(4):538–557, 2012.

[98] Sahand Negahban and Martin J Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *Journal of Machine Learning Research*, 13(May):1665–1697, 2012.

[99] Sahand Negahban and Martin J Wainwright. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics*, pages 1069–1097, 2011.

[100] Angelika Rohde and Alexandre B. Tsybakov. Estimation of high-dimensional low-rank matrices. *Annals of Statistics*, 39(2):887–930, 2011.

[101] LP Kuptsov. Hölder inequality. *SpringerLink Encyclopaedia of Mathematics*, 76, 2001.

[102] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[103] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 322–331. IEEE, 1995.

[104] Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical bernstein stopping. In *Proceedings of the 25th international conference on Machine learning*, pages 672–679, 2008.

[105] Ian Osband and Benjamin Van Roy. Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*, 2015.

[106] Branislav Kveton, Csaba Szepesvari, Zheng Wen, Mohammad Ghavamzadeh, and Tor Lattimore. Garbage in, reward out: Bootstrapping exploration in multi-armed bandits. *arXiv preprint arXiv:1811.05154*, 2018.

[107] Craig Boutilier, Chih-Wei Hsu, Branislav Kveton, Martin Mladenov, Csaba Szepesvari, and Manzil Zaheer. Differentiable bandit exploration. *arXiv preprint arXiv:2002.06772*, 2020.

[108] Botao Hao, Yasin Abbasi Yadkori, Zheng Wen, and Guang Cheng. Bootstrapping upper confidence bound. In *Advances in Neural Information Processing Systems*, pages 12123–12133, 2019.

[109] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

[110] Adam N Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. A practical method for solving contextual bandit problems using decision trees. *arXiv preprint arXiv:1706.04687*, 2017.

[111] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 323–332, 2015.

[112] Sharan Vaswani, Branislav Kveton, Zheng Wen, Anup Rao, Mark Schmidt, and Yasin Abbasi-Yadkori. New insights into bootstrapping for bandits. *arXiv preprint arXiv:1805.09793*, 2018.

[113] Branislav Kveton, Csaba Szepesvari, Mohammad Ghavamzadeh, and Craig Boutilier. Perturbed-history exploration in stochastic multi-armed bandits. *arXiv preprint arXiv:1902.10089*, 2019.

[114] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[115] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval*, 4(2):133–151, 2001.

[116] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[117] Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: To-

wards better representations for reinforcement learning. *arXiv preprint arXiv:2006.02243*, 2020.

[118] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[119] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[120] Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34, 2021.

[121] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021.

[122] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

[123] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

[124] Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 553–560, 2005.

[125] Charline Le Lan, Marc G Bellemare, and Pablo Samuel Castro. Metrics and continuity in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8261–8269, 2021.

[126] William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

[127] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018.

[128] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[129] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando De Freitas. Playing hard exploration games by watching youtube. *Advances in neural information processing systems*, 31, 2018.

[130] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. *Advances in Neural Information Processing Systems*, 32, 2019.

[131] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33:19884–19895, 2020.

[132] Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L Lewis, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. *Advances in Neural Information Processing Systems*, 32, 2019.

[133] Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G Bellemare. On the generalization of representations in reinforcement learning. *arXiv preprint arXiv:2203.00543*, 2022.

[134] Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

[135] Bahram Behzadian, Soheil Gharatappeh, and Marek Petrik. Fast feature selection for linear value function approximation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 601–609, 2019.

[136] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pages 162–169, 2004.

[137] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4(5):9, 2006.

[138] Satinder Singh, Tommi Jaakkola, and Michael Jordan. Reinforcement learning with soft state aggregation. *Advances in neural information processing systems*, 7, 1994.

[139] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

[140] Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. *arXiv preprint arXiv:2204.13060*, 2022.

[141] Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[142] Mete Kemertas and Allan Jepson. Trusted approximate policy iteration with bisimulation metrics. *arXiv preprint arXiv:2202.02881*, 2022.

[143] Charline Le Lan, Marc G Bellemare, and Pablo Samuel Castro. Metrics and continuity in reinforcement learning. *Science Bulletin of NCTU*, 2021.

[144] Junhong Shen and Lin F Yang. Theoretically principled deep rl acceleration via nearest neighbor function approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9558–9566, 2021.

[145] Chengzhuo Ni, Lin F Yang, and Mengdi Wang. Learning to control in metric space with optimal regret. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 726–733. IEEE, 2019.

[146] Emmanuel Rachelson and Michail G Lagoudakis. On the locality of action domination in sequential decision making. 2010.

[147] TW Archibald, KIM McKinnon, and LC Thomas. On the generation of markov decision processes. *Journal of the Operational Research Society*, 46(3):354–361, 1995.

[148] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[149] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

[150] Amazon. `Https://www.amazon.co.uk/`. Accessed: 2018-04-15.

[151] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–536, 1952.

[152] Xi Chen, Zibin Zheng, Xudong Liu, Zicheng Huang, and Hailong Sun. Personalized qos-aware web service recommendation and visualization. *IEEE Transactions on Services Computing*, 6(1):35–47, 2013.

[153] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[154] Hyea Kyeong Kim, Jae Kyeong Kim, and Young U Ryu. Personalized recommendation over a customer network for ubiquitous shopping. *IEEE Transactions on Services Computing*, 2(2):140–151, 2009.

[155] Aleksandrs Slivkins. Contextual bandits with similarity information. *Journal of Machine Learning Research*, 15(1):2533–2568, 2014.

[156] Trong T. Nguyen and Hady W. Lauw. Dynamic clustering of contextual multi-armed bandits. In *Proc. ACM Int. Conf. on Information and Knowledge Management*, CIKM '14, 2014.

[157] Fatemeh Rezaeimehr, Parham Moradi, Sajad Ahmadian, Nooruldeen Nasih Qader, and Mahdi Jalili. Tcars: time-and community-aware recommendation system. *Future Generation Computer Systems*, 78:419–429, 2018.

[158] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. *CoRR*, abs/1401.8257, 2014.

[159] Shuai Li, Claudio Gentile, Alexandros Karatzoglou, and Giovanni Zappella. Data-dependent clustering in exploration-exploitation algorithms. *arXiv preprint arXiv:1502.03473*, 2015.

[160] Shuai Li, Claudio Gentile, Alexandros Karatzoglou, and Giovanni Zappella. Online context-dependent clustering in recommendations based on exploration-exploitation algorithms. *ArXiv*, abs/1608.03544, 2016.

[161] Stéphane Caron, Branislav Kveton, Marc Lelarge, and Smriti Bhagat. Leveraging side observations in stochastic bandits. *ArXiv*, abs/1210.4839, 2012.

[162] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[163] Jialu Liu and Jiawei Han. Spectral clustering. In *Data clustering*, pages 177–200. Chapman and Hall/CRC, 2018.

[164] Ivan Cantador, Peter L Brusilovsky, and Tsvi Kuflik. *Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)*. ACM, 2011.

[165] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[166] Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 8787–8798. PMLR, 2021.