Sequential crowdsourcing and recommendation strategies

Kang, Qiyu

2020

Kang, Q. (2020). Sequential crowdsourcing and recommendation strategies. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/137199>

<https://doi.org/10.32657/10356/137199>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 18:32:21 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

Sequential Crowdsourcing and Recommendation Strategies

KANG QIYU

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2020

Sequential Crowdsourcing and Recommendation Strategies

Kang, Qiyu

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

2020-Feb-09

.....

Date

.....

Kang, Qiyu

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

2020-Feb-09

.....

Date

.....

Tay, Wee Peng

Authorship Attribution Statement

This thesis contains material from 3 paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Q. Kang and W. P. Tay. Sequential multi-class labeling in crowdsourcing. IEEE Trans. Knowl. Data Eng., vol. 31, no. 11, pp. 2190 – 2199, Nov. 2019.

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof. Tay Wee Peng.
- Prof. Tay Wee Peng discussed the problem formulation, propositions/theorems proofs and simulations with me.

Chapter 4 is submitted as Q. Kang and W. P. Tay, "Task recommendation in crowdsourcing based on learning preferences and reliabilities." IEEE Trans. Services Comput., submitted.

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof. Tay Wee Peng.
- Prof. Tay Wee Peng discussed the problem formulation, propositions/theorems proofs and simulations with me.

Chapter 5 is submitted as Q. Kang and W. P. Tay, "Learning orthogonal projections in linear bandits." IEEE Trans. Signal Process., submitted.

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof. Tay Wee Peng.
- Prof. Tay Wee Peng discussed the problem formulation, propositions/theorems proofs and simulations with me.

.....2020-Feb-09.....

Date

.....

Kang, Qiyu

Acknowledgements

I wish to express my greatest gratitude to my supervisor, Prof. Tay Wee Peng, my family, and my friends.

Abstract

Crowdsourcing has emerged as an online, distributed problem-solving and production model where many participating workers are involved in performing tasks collaboratively in domains such as gathering language-related data, labeling the content of an image, or voting. In modern online platforms, including crowdsourcing, recommendation systems play an essential role and have become indispensable. To provide better service in crowdsourcing and recommendation, in this thesis, we study sequential crowdsourcing and recommendation strategies where historical data will be explored when making decisions at the current time step.

Firstly, we consider a multi-class labeling problem in crowdsourcing. As workers may be unreliable, we propose to perform sequential questioning in which the questions posed to the workers are designed based on previous questions and answers. We propose a partially-observable Markov decision process (POMDP) framework to determine the best questioning strategy, subject to the crowdsourcer’s budget constraint. As this POMDP formulation is in general intractable, we develop a sub-optimal approach based on a q -ary Ulam-Rényi game. We also propose a sampling heuristic, which can be used in tandem with standard POMDP solvers, using our Ulam-Rényi strategy. We demonstrate through simulations that our approaches outperform a non-sequential strategy based on error correction coding and which does not utilize workers’ previous responses.

Secondly, we consider a sequential task recommendation problem in a crowdsourcing platform where assigning tasks more likely to be accepted by a worker who is more likely to complete it reliably results in better performance for the crowdsourcer. Without prior information about a worker, his preferences and reliabilities need to be learned over time. We propose a multi-armed bandit (MAB) framework

to sequentially learn a worker’s preferences and reliabilities for different categories of tasks. However, unlike the classical MAB problem, the reward from the worker’s completion of a task is unobservable. We therefore include the use of gold tasks (i.e., tasks whose solutions are known *a priori* and which do not produce any rewards) in our task recommendation procedure. Our model could be viewed as a new variant of MAB, in which the random rewards can only be observed at those time steps where gold tasks are used, and the accuracy of estimating the expected reward of recommending a task to a worker depends on the number of gold tasks used. We show that the optimal regret is $O(\sqrt{n})$, where n is the number of time steps. We develop three task recommendation strategies to determine the number of gold tasks for different task categories, and show that they are order optimal. Simulations verify the efficiency of our approaches.

Finally, we propose a linear stochastic bandit formulation to model sequential non-discriminatory recommendations. Recommendation systems targeting at finding the best matching between items and users without considering “fairness” have been reported as generating discriminatory recommendations from users’ perspectives. In principle, to avoid discrimination, protected attributes such as race or gender should not be used in recommendation algorithms. However, directly discarding the protected attributes will potentially leave lingering discrimination in the large bias error of underfitted modelling without enough parameters. We propose to exclude the individual’s biases and maximize the (expected) cumulative reward like users’ clicks or ratings over a subspace of item attributes, based on the reward observed for the full space. We call this orthogonal projection in linear bandits. We propose sequential non-discriminatory recommendation strategies and prove that they achieve sublinear cumulative projection regret.

Contents

Acknowledgements	ix
Abstract	xi
List of Figures	xvii
List of Tables	xix
Symbols and Acronyms	xxi
1 Introduction	1
1.1 Background and Related Work	2
1.2 Major Contributions	6
1.3 Thesis Outline	10
2 Literature Review	11
2.1 Introduction	11
2.2 Partially-Observable Markov Decision Process	11
2.2.1 Partially-Observable Markov Decision Process Formulation	12
2.2.2 Solving a POMDP	14
2.3 Sequential Bandit Problem	18
2.3.1 Multi-Armed Bandit Problem	18
2.3.2 Linear Stochastic Bandit Problem	21
2.4 Ulam-Rényi Game	23
3 Sequential Multi-Class Labeling in Crowdsourcing	27
3.1 Introduction	27
3.2 Problem Formulation	30
3.2.1 Binary Microtasks with Code Matrix	32
3.2.2 POMDP Formulation	34
3.3 Ulam-Rényi Game Approach	36
3.3.1 A Heuristic Strategy for the q -ary Ulam-Rényi Game	36
3.3.2 POMDP Approximation	41
3.4 Sampling Using Ulam-Rényi Tree	43

3.5	Simulations and Performance Evaluation	43
3.5.1	Comparison of Sampling Strategies	44
3.5.2	Varying M	45
3.5.3	Varying $\underline{b}$	48
3.5.4	Varying ψ	49
3.5.5	Varying ρ	50
3.5.6	Optimality of URSQS	51
3.A	Proof of Proposition 3.1	54
4	Sequential Task Recommendation in Crowdsourcing	55
4.1	Introduction	55
4.2	System Model	59
4.3	Optimal Regret Order	62
4.4	Order Optimal Strategies	62
4.4.1	Greedy Recommendation Strategy	63
4.4.2	Uniform-Pulling Recommendation Strategy	64
4.4.3	ϵ -First Recommendation Strategy	65
4.4.4	Discussions	65
4.5	Simulations and Performance Evaluation	67
4.5.1	Comparison of Different Strategies	68
4.5.2	Varying Total Number of Task Categories K	69
4.5.3	Varying α	71
4.5.4	Varying $\min_{x \neq *} \Delta_x$	71
4.5.5	Learning Only Partial Information	72
4.A	Proof of Theorem 4.1	73
4.B	Proof of Theorem 4.2	75
4.C	Proof of Theorem 4.3	81
4.D	Proof of Theorem 4.4	82
5	Sequential Non-Discriminatory Recommendations	83
5.1	Introduction	83
5.2	System Model and Related Work	87
5.2.1	Orthogonal Projection in Linear Bandits	88
5.2.2	Related Work	91
5.3	Strategies and Regret Analysis	94
5.3.1	Projection Greedy Strategy	94
5.3.2	Regret Analysis	95
5.3.2.1	Finitely Many Arms	95
5.3.2.2	Infinitely Many Arms	96
5.3.3	Additional Smoothness Assumption	96
5.4	Simulations and Performance Evaluation	97
5.4.1	Experiments on Synthetic Data	98
5.4.1.1	Varying α	99

5.4.1.2 Results and Analysis	101
5.4.2 Experiments on Real Dataset	103
5.A Proof of Theorem 5.1	105
5.B Proof of Theorem 5.2	110
5.C Proof of Theorem 5.3	114
6 Conclusion and Future Work	117
List of Author's Publications	119
Bibliography	121

List of Figures

3.1	Code matrix representation of the DCFECC approach	33
3.2	Performance comparison between uniform sampling and URT sampling with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M	45
3.3	Performance comparison between strategies with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M	46
3.4	Performance comparison between strategies with $M = 32$, $\rho = 0.75$, $\psi = 0.05$ and different $\underline{b}$	48
3.5	Performance comparison between strategies with $M = 32$, $\rho = 0.75$, $\underline{b} = 9$ and different ψ	49
3.6	Performance comparison between strategies with $M = 32$, $\underline{b} = 9$, $\psi = 0.05$ and different ρ	50
3.7	Performance comparison between different strategies with $M = 128$, $\rho = 0.75$, $\underline{b} = 9$ and $\psi = 0.05$	51
3.8	Performance comparison between different strategies with $M = 32$, $\rho = 0.75$, $\underline{b} = 13$ and $\psi = 0.05$	52
4.1	Performance comparison between different strategies using setting no. 1.	68
4.2	Average regret order of UR(γ) using setting no. 1.	69
4.3	Performance comparison between UR and GR with $K = 10, 15$ and 25 using setting no. 3 – 5 in Table 4.1, respectively.	70
4.4	Performance of GR with different values of parameter α and different settings.	72
4.5	Performance of UR with different values of parameter α and different settings.	73
4.6	Performance of different strategies with varying $\min_{k \neq *} \Delta_k$ using setting no. 2 for different values of $(\underline{w}, \underline{y})$	74
4.7	Performance of UR, GR and the ϵ -first strategy considering either preferences or reliabilities under setting no. 1.	75
5.1	Regret comparison between different strategies with $\vartheta = 0.5$ and varying α	100
5.2	Performance comparison between different strategies using setting (a) with $d = 10$, $K = 45$, $\vartheta = 0.5$ and $u = 5$	101
5.3	Performance comparison between different strategies using setting (b) with $d = 10$, $K = 45$, $\vartheta = 0.5$ and $u = 5$	102

5.4	Average cumulative projection regret using setting (c) with $d = 4$, $u = 2$ and $\vartheta = 0.5$	103
5.5	Average cumulative projection regret using the Wine Quality dataset.	104

List of Tables

3.1	Estimated $B(2, e)$ of the $(M, 2, e)$ Ulam-Rényi game where $M = 2^m$.	41
3.2	Average reward for different $ \hat{\mathbb{Q}} $ with $M = 32$, $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and URT sampling.	44
3.3	$ \hat{\mathbb{Q}} $ and $ \mathbb{B} $ for different M .	44
3.4	Average offline computation times for different strategies with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M .	47
3.5	Average online computation times for different strategies with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M .	47
4.1	(p_x, q_x) for each task category. In setting no. 2, $(\underline{w}, \underline{y})$ is varied over $[0, 1]^2$.	67

Symbols and Acronyms

In this thesis, we use capital letters like X_t to denote random variables or vectors, lowercase letters like x to denote deterministic scalars, boldface lowercase letters like $\mathbf{x}$ to denote deterministic vectors, boldface capital letters like $\mathbf{T}$ to denote tuples or matrices. We assume that all random variables are defined on the same underlying probability measure space with probability measure $\mathbb{P}$, and $\mathbb{E}$ is the associated expectation operator. Beside these two special blackboard notations $\mathbb{P}$ and $\mathbb{E}$, other capital blackboard letters like $\mathbb{U}$, $\mathbb{R}$ and $\mathbb{D}$ all represent sets or spaces.

Symbols

$\mathbb{R}$	the set of real numbers
$\mathbb{N}$	the set of natural numbers
$\mathbb{R}^n$	the n -dimensional Euclidean space
$\langle \cdot, \cdot \rangle$	the inner product of two vectors
$\ \cdot \ $	the L_2 -norm of a vector in Euclidean space
$\ \cdot \ _{\mathbf{A}}$	the weighted L_2 -norm $\sqrt{\langle \mathbf{x}, \mathbf{A} \mathbf{x} \rangle}$ where $\mathbf{A}$ is a positive definite matrix
$\text{Bern}(p)$	the Bernoulli distribution with parameter p
x^+	$\max\{x, 0\}$
$(\cdot)^\top$	the transpose of vector or matrix
$\mathbb{E}_X$	the expectation w.r.t X (we will omit X when possible)
$\mathbb{P}(E)$	the probability of event E
$\mathbb{P}(E_1 E_2)$	the conditional probability of event E_1 given E_2
$\log$	the natural logarithm

$\mathbf{A}(i, k)$	the (i, k) -th element of matrix $\mathbf{A}$
$\mathbf{b}(i)$	the i -th element of vector $\mathbf{b}$
$\mathbb{U}^\perp$	the orthogonal complement of space $\mathbb{U}$
$P_{\mathbb{U}}$	the linear orthogonal projection operator to space $\mathbb{U}$
$O(\cdot)$	the Big-O order
$o(\cdot)$	the Little-O order
$p_{X Y}(x y)$	the conditional probability (we will omit X and Y when possible)
$\mathbf{1}_A$	the indicator function which takes value 1 if and only if $\omega \in A$, and 0 otherwise
$\mathbb{A} \times \mathbb{B}$	the Cartesian product of set $\mathbb{A}$ and $\mathbb{B}$
$\mathbb{B}^n$	the n -ary Cartesian power of set $\mathbb{B}$
$\sim$	the equality in distribution
$\mathcal{N}(\mu, \vartheta^2)$	Gaussian distribution with variance ϑ^2 and mean μ
$\mathcal{U}(a, b)$	the uniform distribution with density function support $[a, b]$

Acronyms

MAB	Multi-Armed Bandit
POMDP	Partially-Observable Markov Decision Process
PBVI	Point-Based Value Iteration Algorithm
POMCP	Partially Observable Monte-Carlo Planning (POMCP) Algorithm
MIQP	Mixed-Integer Quadratic Programming
DCFEC	Distributed Classification Fusion Using Error-Correcting Codes
URSQS	Ulam-Rényi Sequential Questioning Strategy
URT	Ulam-Rényi Tree
UR	Uniform-Pulling Recommendation Strategy
GR	Greedy Recommendation Strategy
GPS	Greedy Projection Strategy
OGS	Original Greedy Strategy
CGS	Careless Greedy Strategy
UE	Uncertainty Ellipsoid (UE) Policy

i.i.d. independent and identically distributed

Chapter 1

Introduction

This thesis shows how to design crowdsourcing and recommendation strategies where historical data will be explored. In order to provide better service in crowdsourcing, we sequentially design questions and recommend matching tasks to workers. We deal with the challenge of learning from historical answers with unobservable accuracy in crowdsourcing recommendations. For more general recommendation systems, we address the concerns of discriminatory biases when recommending items. We show the optimal regret (a measurement of the recommendation quality defined in subsequent chapters) is sublinear and also propose strategies with sublinear regret. Our sequential strategies introduced in this thesis allow platform designers to build crowdsourcing and recommendation systems more efficiently or fairly to some extent.

In the following, we provide some background to the current development of crowdsourcing and recommendation systems, introduce some related work, and present our contributions.

1.1 Background and Related Work

Crowdsourcing has emerged as an efficient tool to effectively provide solutions for tedious tasks which humans can complete notably better than the machine or few workers cannot complete easily. In crowdsourcing, tasks like image segmentation, image labeling or language annotation are distributed to a large number of online workers who have access to the internet. Crowdsourcing offers a way for crowdsourcers (e.g., institutions or companies) to find better and more scalable solutions at a lower cost than even the best in-house teams, and creates new opportunities for online individuals to change the way they work and earn rewards wherever they are as long as they have access to the internet. The leading crowdsourcing companies, such as Amazon Mechanical Turk [1], Figure Eight [2], eYeka [3] and InnoCentive [4], have helped businesses tackle large-scale problem solving, like big data annotation, with little effort successfully.

In crowdsourcing, workers are paid only a few cents for some simple tasks, such as indicating text categories after reading a snippet of text [5]. Since workers are not experts in general, the tasks in crowdsourcing are tedious and the reward is small typically, workers may be unreliable and errors are common in their answers [6]. Even though the error probabilities of final answers inferred using trivial majority voting [7] will decrease with more workers, it is tempting for crowdsourcers to adopt more efficient strategies to ensure the same accuracy of final answers with fewer workers used. To achieve this goal, over the past few years, many researchers have proposed various strategies. For example, in [6,8], the authors proposed a new task assignment scheme using a bipartite graph. In [9], the authors proposed an iterative inference algorithm based on a spectral method. In [10] an allocation algorithm is developed to assign tasks to different workers, and an inference method achieving an order-optimal redundancy-accuracy trade-off was developed. Redundancy here refers to assigning the same task to multiple workers and using a majority voting rule to infer the final answer.

A notable type of tasks in crowdsourcing is multi-class labeling, including the text categorization tasks introduced above, image labeling tasks (e.g. classifying an image of a dog into one breed), etc. These labeled data collected from crowdsourcing can be served as the basis for many supervised learning algorithms, especially for the deep supervised learning algorithms [11] in the era of big data. In order to collect the labels from crowdsourcing platforms efficiently with high accuracy, in [7], they developed an algorithm called Distributed Classification Fusion using Error-Correcting Codes (DCFEC) using error-correction coding to divide a single multi-class labeling task into binary questions, which are then assigned to the workers. The classification is then done by performing a Hamming distance decoding of all the workers' binary responses, which can be treated as the noisy versions of a coded message. We note that DCFEC is based on a non-feedback coding strategy which performs better than the trivial majority voting strategy. One can use a feedback coding strategy to further reduce the misclassification probability. In this thesis, we therefore propose to perform sequential questioning in which the questions posed to the workers are designed based on previous questions and answers as that feedback coding is designed based on the historical signals and feedbacks sent over the channel. We are the first one that proposes sequential questioning in crowdsourcing. Our results suggest that it is more efficient (in the sense of achieving the same answer accuracy with fewer workers) for crowdsourcers to design questions sequentially based on historical data.

The internet is the engine of modern crowdsourcing platforms, which provides both a mechanism for organisations to announce tasks throughout the world, and a network for connecting online individuals with diverse knowledge and skills. It is estimated that there will be approximately five billion internet users by 2020 [12], which will further contribute to the development of crowdsourcing. With such an increasingly large number of internet users, recommendation systems have become indispensable in modern online platforms. Amazon Mechanical Turk, which is one of the oldest crowdsourcing marketplaces, has been reported to maintain 500000 registered workers and 2000 to 5000 active online workers at any given moment,

which is roughly equivalent to a 10000-to-25000-person full-time workforce [5]. As of 2015, it was estimated there was an average of 1,278 active crowdsourcers per day in Amazon Mechanical Turk [13]. A user’s interest in other online platforms is unrestricted in the sense that the platform can recommend anything he or she likes. In contrast, a worker’s choice in a crowdsourcing platform is reasonably limited by his or her abilities. Workers participating in a crowdsourcing platform can have a wide range of abilities and interests. It is reported in [5] that some workers, who are trying to make Amazon Mechanical Turk a core source of income, spend hours one day scrolling through pages upon pages of job posting to search for tasks that they are interested in and good at, which obviously limit the platform productivity. As the number of tasks uploaded daily by crowdsourcers can be large, the crowdsourcing platform should aim to recommend tasks to each worker that he or she will likely accept. Including the workers’ preferences in task assignment can help workers to find their preferred tasks faster as they will have fewer recommendations to consider. At the same time, estimating the reliabilities of a worker in different tasks helps the crowdsourcer to collect high-quality results quicker.

Over the past few years, some researchers have proposed various crowdsourcing recommendation strategies. For example, the paper [14] proposed to perform task assignment using workers’ preferences based on many-to-one matching. However, the paper assumes that workers’ preferences and reliabilities are known *a priori*, which is unrealistic in a practical system. In [15], the authors proposed two recommendation strategies that compute the top- k most suitable tasks for a given worker and the top- k best workers for a given task. However, they assume that the workers can be evaluated immediately. Instead, in this thesis, we deal with the challenge that the accuracy of workers’ completed tasks is unobservable, as is often the case in real crowdsourcing platforms. Our sequential task recommendation strategy for crowdsourcing therefore will guide platform designers to build more efficient recommendation systems in crowdsourcing that will do a better matching in terms of workers’ preferences and reliabilities, and subsequently achieve more efficient task completion.

As we have discussed above, in general recommendation systems, a plausible recommendation scheme tends to get as many clicks/views as possible. Hitherto, collaborative filtering, content-based filtering, and sequential bandits have been commonly adopted in recommendation systems to analyze users' preferences and provide personalized recommendations that match users' tastes most appropriately. Content-based filtering provides the user with new items which are similar, in terms of the item description profiles, to those consumed already by the user [16]. Collaborative filtering tries to analyze the similarity relationships between users and their previously consumed items by mining the users' historical data without the description profiles [17]. Collaborative and content-based filtering are both offline approaches, while sequential bandits [18] are online approaches which take into account additional information about the users' behaviors that can be gleaned during the users' views sequentially over time.

Recently, the recommendation systems in many platforms like Google, Facebook, Netflix and Youtube have been reported generating discriminatory recommendations from users' perspectives. For example, in [19], the author analyzed the ads provided by Google AdSense and concluded that the platform tends to recommend more ads suggestive of arrests if some typical black names are typed in. In [20], it is reported that such platformed racism also exists in Facebook and YouTube whose recommendation algorithms tend to generate contents implying an Australian race-based controversy. Another example is that Netflix last year angered black subscribers with targeted posters containing black actors no matter how minor their roles in the film are [21]. All such rising concerns have suggested the importance of designing recommendation algorithms with "fairness" to eliminate discriminatory biases when recommending items to users. In principle, to avoid discriminations, protected attributes such as race or gender should not be used in recommendation algorithms. But directly discarding the protected attributes will potentially leave lingering discrimination in the large bias error of underfitted modelling without enough parameters [22, 23].

The sacrifice of the best matching, in the sense of getting the most clicks/views,

is inevitable in a discrimination-aware recommendation system. Many works, e.g. [24, 25], model the discrimination-aware recommendation as a constraint optimization problem. In [24], the authors exploited the concept arisen from fairness-aware data mining [26], and proposed to recommend items statistically independent of specified protected attributes. They modeled the recommendation objective by adding a discrimination regularization to the objective function from the previously studied best matching recommendation system [17], and employed the probabilistic matrix factorization technique to get solutions. Authors in [25] proposed a novel discrimination-aware tensor recommendation framework designed to improve fairness by isolating and extracting protected features in tensor decompositions. The aforementioned discrimination-aware recommendations are all offline. In this thesis, we focus on sequential non-discriminatory recommendations, which, to the best of our knowledge, is the first non-discriminatory work on sequential recommendations. The benefit of adopting our sequential non-discriminatory recommendation strategy is that the platform will tend to do a fair matching during the recommendations.

1.2 Major Contributions

In this thesis, we consider sequential crowdsourcing and recommendation strategies. Here, “sequential” means the decision made at the current time step is based on the users’ historical data. More specifically, we first consider performing sequential questioning to determine the hidden state of a multi-class labeling problem where the questions posed to the workers are designed based on previous questions and answers. Since workers participating in a crowdsourcing platform can have a wide range of abilities and interests, to optimize the selection of workers for particular questions, we then consider performing sequential task recommendations. Finally, regarding the discrimination problem in general recommendation systems, we propose a formulation and design strategies to avoid discriminatory recommendations

in sequential/online learning. Our sequential strategies introduced in this thesis allow platform designers to build crowdsourcing and recommendation systems more efficiently or fairly to some extent. In this section, the major contributions of this thesis are summarized as follows.

- In Chapter 3, we consider a multi-class labeling problem in a crowdsourcing platform. As workers may be unreliable, in this thesis, we propose to perform sequential questioning in which the questions posed to the workers are designed based on previous questions and answers. We model the problem as an optimal stopping under partial observation problem which can further be reformulated to a partially-observable Markov decision process (POMDP) to determine the best questioning strategy, subject to the crowdsourcer's budget constraint.

Since the decision space of this POMDP is extremely large, standard POMDP solvers like PBVI [27, 28] and POMCP [29], which enumerate over all possible decisions to find the optimal decision, cannot be used. Our first strategy is to approximate the POMDP using a q -ary Ulam-Rényi game [30–32] in which one player chooses a state out of M possible states, and another player asks the first player q -ary questions in order to determine the state chosen. This is an iterative game in which the questions are asked sequentially, and a question at one iteration can be based on the responses to the questions from previous iterations. The first player's answers to the questions may be wrong and his or her choice of the state corresponds to the true state of our multi-class labeling problem, while giving a wrong answer to a question corresponds to the unreliability of a worker in our crowdsourcing platform. We show how to find the best parameter q in order to achieve an optimal trade-off between the classification accuracy and the number of iterations of questions required. This trade-off is an approximation of the objective in the original optimal stopping problem.

As finding the optimal strategy for a Ulam-Rényi game is also in general intractable, heuristics for solving a *binary* Ulam-Rényi game have been proposed in [30, 31]. However, to the best of our knowledge, strategies for a general q -ary Ulam-Rényi game have been proposed only when the number of questions allowed is sufficiently large. In this thesis, we propose an efficient heuristic, which is however suboptimal in general.

Furthermore, based on our proposed Ulam-Rényi strategy, we further propose a decision sampling strategy for the POMDP to sample a small subset of decisions so that standard POMDP solvers like PBVI [27, 28] and POMCP [29] can be used. Simulations suggest that this sampling strategy is better than uniform sampling.

- In Chapter 4, we consider a sequential task recommendation problem in a crowdsourcing platform where workers can have a wide range of abilities and interests for different tasks. As workers' preferences and reliabilities for different tasks are not exactly known, online learning through sequential task recommendations improves the task recommendation quality over time. In this thesis, we therefore propose a multi-armed bandit (MAB) framework to learn these sequentially through exploration, while the cumulative reward (a measurement of the recommendation quality defined in subsequent chapters) is optimized through the exploitation of the empirical best match.

However, unlike the classical MAB problem in which the reward at each time step can be observed, the reward is *unobservable* in our formulation as the crowdsourcing platform does not know the true solution of each task. We therefore include the use of gold tasks whose solutions are known *a priori* in our task recommendation procedure.

Gold tasks do not generate any rewards. Including more gold tasks leads to better estimates of workers' preferences and reliabilities for different task categories, but also results in less reward. In a crowdsourcing platform, a plausible compensation scheme is to reward a worker for completed tasks according to his or her reliability [33–35]. Furthermore, to accurately fuse

the results from multiple workers, the knowledge of the workers' reliabilities are required [36, 37]. Therefore, the platform aims to estimate the workers' reliability as accurately as possible. To reflect these aspects, we include both the reward and estimation accuracy in computing the regret. We show that the optimal regret under our formulation has order $O(\sqrt{n})$, where n is the number of time steps. In addition, we propose three strategies that achieve the order optimal regret. Our model could be viewed as a new variant of MAB, in which the random rewards can only be observed at those time steps where gold tasks are used, and the accuracy of estimating the expected reward of recommending a task to a worker depends on the number of gold tasks used.

- In Chapter 5, we consider a discrimination problem in sequential recommendations. A traditional recommendation system in a platform typically aims to recommend products (items) that match users' interests most since users may migrate to an alternative platform if they are not interested in the items provided by the current platform's recommendation system. However, targeting at finding such best matching only has been reported generating discriminatory recommendations from users' perspectives. We therefore propose a linear stochastic bandit formulation to model the sequential non-discriminatory recommendations. We propose to exclude the individual's biases and maximize the (expected) cumulative reward like users' clicks or ratings over a subspace of item attributes, based on the reward observed for the full space. We call this orthogonal projection in linear bandits.

Sequential non-discriminatory recommendation strategies are introduced and proved to achieve sublinear cumulative projection regret. In the case where there are finitely many (a small fixed number) items, we develop a strategy to achieve $O(|\mathbb{D}| \log n)$ regret, where n is the number of time steps and $|\mathbb{D}|$ is the number of items. In the case where the platform has an extremely large number of items or the item repository has dynamic unlimited update over time, our strategy achieves $O(n^{2/3}(\log n)^{1/2})$ regret.

1.3 Thesis Outline

Chapter 1 introduces the scope of this thesis. The main contribution and the outline of the thesis are also presented briefly. Chapter 2 reviews the sequential models in the literature which will be used in this thesis, including partially-observable Markov decision process, multi-armed bandit problem, linear stochastic bandit problem, and Ulam-Rényi game. It also introduces necessary mathematical concepts which will be used throughout this thesis. Chapter 3 proposes to solve a sequential multi-class labeling problem in crowdsourcing. Chapter 4 proposes sequential task recommendation strategies based on learning workers' preferences and reliabilities over time in crowdsourcing. Chapter 5 proposes to avoid discriminations in general recommendation systems. Chapter 6 concludes the thesis.

Notations: We use a_i^j to denote $(a_i, a_{i+1}, \dots, a_j)$. The notation $p_{X|Y}(x | y)$ (we will omit X and Y when possible) represents the conditional probability of the random variable X given Y , where Y can be a random variable or an event. We assume that all random variables and events are defined on a common probability space with probability measure $\mathbb{P}$ and expectation operator $\mathbb{E}$. We use $\text{Bern}(p)$ to denote the distribution of a Bernoulli random variable Y with $\mathbb{P}(Y = 1) = p$. The notation $\sim$ denotes equality in distribution. We use $\mathbb{N}$ to denote the set of positive integers and E^c to denote the complement of the event E . We let $x^+ = \max\{x, 0\}$. The indicator function $\mathbf{1}_A(\omega) = 1$ if and only if $\omega \in A$. $\mathbf{I}$ is the identity matrix. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, let $\mathbf{x}^\top$ be the transpose of $\mathbf{x}$, and $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}$. We use $\|\mathbf{x}\|_2$ to denote the L_2 -norm $\sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ and $\|\mathbf{x}\|_{\mathbf{A}}$ to denote the weighted L_2 -norm $\sqrt{\langle \mathbf{x}, \mathbf{A} \mathbf{x} \rangle}$ for any $\mathbf{A} \in \mathbb{R}^{d \times d}$ that is a positive definite matrix. For non-negative functions $f(n)$ and $g(n)$, we write $f(n) = O(g(n))$ if $\limsup_{n \rightarrow \infty} f(n)/g(n) < \infty$, $f(n) = o(g(n))$ if $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$, and $f(n) = \Theta(g(n))$ if $0 < \liminf_{n \rightarrow \infty} f(n)/g(n) \leq \limsup_{n \rightarrow \infty} f(n)/g(n) < \infty$. We use $\mathcal{U}(a, b)$ to denote the continuous uniform distribution whose density function has support $[a, b]$, and $\mathcal{N}(0, \vartheta^2)$ to denote the Gaussian distribution with variance ϑ^2 and mean 0. The set of real numbers is denoted as $\mathbb{R}$. The set of natural numbers is denoted as $\mathbb{N}$.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews the background of the sequential models used in this thesis, including partially-observable Markov decision process, multi-armed bandit problem, linear stochastic bandit problem, and Ulam-Rényi game. Partially-observable Markov decision process and Ulam-Rényi game will be used in Chapter 3, while multi-armed bandit problem and the more general linear stochastic bandit problem will be used in Chapters 4 and 5.

2.2 Partially-Observable Markov Decision Process

In this section, we introduce the partially-observable Markov decision process (POMDP) model, and two standard POMDP solvers which will be used in Chapter 3 to perform sequential questioning.

2.2.1 Partially-Observable Markov Decision Process Formulation

A partially-observable Markov decision process (POMDP) provides a framework to model optimal decision-making in environments which are only partially observable to the decision maker [38]. In the following, we give the definition and the objective cumulative reward function of a POMDP.

Definition 2.1. A POMDP is a tuple $\langle \mathbb{S}, \mathbb{D}, \Omega, \mathbb{O}, \Psi, r, \underline{\rho} \rangle$, where

- $\mathbb{S}$ is a finite set of discrete states
- $\mathbb{D}$ is a finite set of decisions
- $\mathbb{O}$ is a finite set of observations
- $\Psi : \mathbb{S} \times \mathbb{D} \times \mathbb{S} \mapsto [0, 1]$ is the transition function
- $\Omega : \mathbb{S} \times \mathbb{D} \times \mathbb{O} \mapsto [0, 1]$ is the observation function.
- $r : \mathbb{S} \times \mathbb{D} \mapsto \mathbb{R}$ is the reward function, where $\mathbb{R}$ is the set of real number.
- $\underline{\rho}$: a constant discount factor satisfying $0 < \underline{\rho} \leq 1$.

The environment is represented by a finite set of states $\mathbb{S}$. At each time step t , if the environment is at state $H_t = s \in \mathbb{S}$, and the decision maker chooses a decision $X_t = x \in \mathbb{D}$, the environment will be changed into a new state $H_{t+1} = s'$ stochastically according to the probability transition function Ψ , where $\mathbb{P}(H_{t+1} = s' \mid H_t = s, X_t = x) = \Psi(s, x, s')$, at time step $t + 1$. The transition only depends on the current state s and is independent of all previous states, which forms the Markov process property. The decision maker receives an immediate reward $r(s, x)$ after each decision-making. The main difference between a POMDP and a fully observable MDP is that in a POMDP the decision maker is unable to directly observe the current state and can only perceive an observation $O_{t+1} = o \in \mathbb{O}$ according to the observation function Ω , where $\mathbb{P}(O_{t+1} = o \mid H_{t+1} = s', X_t = x) =$

$\Omega(s', x, o)$ is the probability of observing o when the environment is in state s' after the decision maker's decision choice x . As a result, the decision maker can only maintain a probability distribution over states, called a belief $\mathbf{b}$ instead of the exact current state information. All beliefs are contained in a simplex $\Delta = \{\mathbf{b} \in [0, 1]^{|\mathbb{S}|} : \sum_{s \in \mathbb{S}} \mathbf{b}(s) = 1\}$ whose dimension is $(|\mathbb{S}| - 1)$, where $\mathbf{b}(i)$ is the i -th element of $\mathbf{b}$. It is shown that the belief is a sufficient statistic for a given history that starts from the initial belief $\mathbf{b}_0$ in POMDP :

$$\mathbf{b}_t(s) := \mathbb{P}(H_t = s \mid \mathbf{b}_0, x_1, o_1, \dots, o_{t-1}, x_t, o_t), \forall s \in \mathbb{S}, \quad (2.1)$$

since a POMDP is equivalent to a fully observable MDP problem with the belief state [39].

The belief $\mathbf{b}_{t-1}$ after choosing x and observing o is updated to $\mathbf{b}_t$ by Bayes' rule:

$$\mathbf{b}_t^{xo}(s') = \frac{\Omega(s', x, o)}{p(o \mid \mathbf{b}_{t-1}, x)} \sum_{s \in \mathbb{S}} \Psi(s, x, s') \mathbf{b}_{t-1}(s), \quad (2.2)$$

where

$$p(o \mid \mathbf{b}_{t-1}, x) = \sum_{s' \in \mathbb{S}} \Omega(s', x, o) \sum_{s \in \mathbb{S}} \Psi(s, x, s') \mathbf{b}_{t-1}(s)$$

is a normalizing factor. Each POMDP problem assumes an initial belief $\mathbf{b}_0$, which reflects the decision maker's prior knowledge about the environment. The value function $V^\pi : \Delta \mapsto \mathbb{R}$ for a finite n -time-step POMDP with a specific policy $\pi(\mathbf{b})$, which chooses decisions (deterministically or randomly) given the current belief $\mathbf{b}$, is defined by

$$V^\pi(\mathbf{b}) = \mathbb{E}_\pi \left[\sum_{t=1}^n \underline{\rho}^t r_t \mid \mathbf{b}_0 = \mathbf{b} \right], \quad (2.3)$$

where n is the number of time steps, $0 < \underline{\rho} \leq 1$ is the discount factor, r_t is the reward received at each time step t , and the expectation is taken over all randomness including random transition, observation and the policy stochasticity.

The goal is to find a policy π^* maximizing the value function. In Chapter 3, we will see the crowdsourcers in crowdsourcing systems play the role of the decision maker in POMDP, and to perform sequential questioning is to choose decisions from the decision space $\mathbb{D}$ sequentially.

2.2.2 Solving a POMDP

There are two branches of POMDP solvers, offline and online solvers, in the literature [27–29, 39]. As indicated by the names, offline POMDP solvers specify the best decisions to choose for all possible situations prior to the start of the process, while online solvers will compute a policy by planning online only for the current belief $\mathbf{b}$ encountered during planning. In this thesis, we introduce one offline approach, Point-Based Value Iteration (PBVI) algorithm, [27, 28], together with another online POMDP solver, POMCP [29], to compute the policy for a POMDP. PBVI is more feasible when the number of time steps and the state space dimension are small while POMCP is more feasible in large state dimension problems.

The optimal value function, i.e. the value function corresponding to the optimal policy π^* which maximizes (2.3), is always piecewise linear and convex in belief [40]. The optimal policy π^* in a finite n time steps POMDP can be computed by iterating $n - 1$ times, starting from time step n (i.e., horizon 1), and at each horizon considering one time step into the past. Let V_i^* be the optimal value function at time step $n - i + 1$ (horizon i), then (2.3) can be formulated as dynamic programming (DP):

$$V_{i+1}^*(\mathbf{b}) = \max_{x \in \mathbb{D}} \left[\sum_{s \in \mathbb{S}} r(s, x) \mathbf{b}(s) + \rho \sum_{o \in \mathbb{O}} p(o \mid \mathbf{b}, x) V_i^*(\mathbf{b}^{xo}) \right], \quad (2.4)$$

where i varies from 1 to $n - 1$.

At horizon 1, i.e., at time $t = n$, the decision maker only has one decision left to choose, hence the value function is

$$V_1^*(\mathbf{b}) = \max_{x \in \mathbb{D}} \left[\sum_{s \in \mathbb{S}} r(s, x) \mathbf{b}(s) \right], \quad (2.5)$$

which can be rewritten as:

$$V_1^*(\mathbf{b}) = \max_{\boldsymbol{\alpha} \in \mathbb{V}_1^*} \langle \mathbf{b}, \boldsymbol{\alpha} \rangle, \quad (2.6)$$

where the solution set $\mathbb{V}_1^* = \{\boldsymbol{\alpha}_1^{x_1}, \dots, \boldsymbol{\alpha}_1^{x_{|\mathbb{D}|}}\}$ contains $|\mathbb{D}|$ vectors, and $\boldsymbol{\alpha}_1^x(s) = r(s, x)$, $\forall x \in \mathbb{D}, s \in \mathbb{S}$. In general, at horizon i the solution can be represented by a set of vectors $\mathbb{V}_i = \{\boldsymbol{\alpha}_i^0, \boldsymbol{\alpha}_i^1, \dots, \boldsymbol{\alpha}_i^m\} \subseteq \mathbb{R}^{|\mathbb{S}|}$ and the value function for the current horizon is $V_i(\mathbf{b}) = \max_{\boldsymbol{\alpha} \in \mathbb{V}_i} \langle \mathbf{b}, \boldsymbol{\alpha} \rangle$. In addition, a decision $x'(\boldsymbol{\alpha}_i^j) \in \mathbb{D}$ is associated with each vector $\boldsymbol{\alpha}_i^j$ in $\mathbb{V}_i$. The policy for belief $\mathbf{b}$ is to take the decision $x'(\boldsymbol{\alpha}_i^j)$ maximizing $\langle \mathbf{b}, \boldsymbol{\alpha}_i^j \rangle$.

At horizon $i + 1$, in the worst case, the optimal update [40] will lead $|\mathbb{V}_{i+1}|$ into size $|\mathbb{D}||\mathbb{V}_i|^{|\mathbb{O}|}$, which is an exponential growth and will cause its impracticality in large real world problems. Instead, PBVI, the approximate solver, solves a POMDP for a finite set of belief points $\mathbb{B} \subseteq \Delta$ rather than the entire Δ . The details [27] are listed below:

First, for $\forall x \in \mathbb{D}, \forall o \in \mathbb{O}$, create $|\mathbb{D}||\mathbb{O}|$ sets of vectors:

$$\mathbb{F}^{x,o} = \left\{ \boldsymbol{\alpha}^{x,o} \in \mathbb{R}^{|\mathbb{S}|} \mid \boldsymbol{\alpha}^{x,o}(s) = \rho \sum_{s' \in \mathbb{S}} \Psi(s, x, s') \Omega(s', x, o) \boldsymbol{\alpha}_i^j(s'), \forall s \in \mathbb{S}, j \in |\mathbb{V}_i| \right\}, \quad (2.7)$$

which is similar to the second term in (2.4) but with the change of considering only solutions in $\mathbb{V}_i$ (whose size is at most $|\mathbb{B}|$).

Next, for $\forall \mathbf{b} \in \mathbb{B}$, construct:

$$\mathbb{F}_{\mathbf{b}} = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^{|\mathbb{S}|} \mid \boldsymbol{\alpha} = \boldsymbol{\alpha}_1^x + \sum_{o \in \mathbb{O}} \arg \max_{\boldsymbol{\alpha}' \in \mathbb{F}^{x,o}} \langle \boldsymbol{\alpha}', \mathbf{b} \rangle, \forall x \in \mathbb{D} \right\}, \quad (2.8)$$

which contains all possible solutions, each for one decision in $\mathbb{D}$, given a particular belief $\mathbf{b}$ using (2.4), (2.6) and (2.7).

Finally, find the best decision for each belief point in $\mathbb{B}$:

$$\mathbb{V}_{n+1} = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^{|\mathbb{S}|} \mid \boldsymbol{\alpha} = \arg \max_{\boldsymbol{\alpha}' \in \mathbb{F}_{\mathbf{b}}} \langle \boldsymbol{\alpha}', \mathbf{b} \rangle, \forall \mathbf{b} \in \mathbb{B} \right\}. \quad (2.9)$$

Obviously, $|\mathbb{V}_{n+1}|$ contains at most $|\mathbb{B}|$ vectors which relieves the crisis of exponential growth during iteration. How to construct the belief sets is actually crucial in PBVI. In [27], it proposed Stochastic Simulation with Explorative decision (SSEA) approach to expand $\mathbb{B}$. Initialized with $\mathbf{b}_0$ in $\mathbb{B}$, SSEA on each expansion randomly simulates decisions and observations for every belief in $\mathbb{B}$ to sample new beliefs, and maintains the new ones that are farthest away from points already in $\mathbb{B}$. After several iterations, SSEA can cover a wide area of the ‘reachable’ beliefs which forms a low-dimensional manifold in the belief simplex Δ .

The online solver POMCP [29] is a *history*-based tree algorithm using the PO-UCT [29, 41] forward searching policy that chooses decisions, and a particle filter that updates the belief state. Here the history is defined as a sequence of decisions and observations, $h_t = \{x_1, o_1, \dots, x_t, o_t\}$ or $h_t x_{t+1} = \{x_1, o_1, \dots, x_t, o_t, x_{t+1}\}$.

Since (2.1) and (2.2) indicate that each history h corresponds to a particular belief $\mathbf{b}$, we abuse $V^*(h)$ to indicate the optimal value of a history h . We also use $V^*(hx) = \mathbb{E}_O [V^*(\{h, x, O\})]$ to indicate the optimal value taking expectation over all observations if the decision maker chooses decision x given history h .

Each node, representing a particular history, in the rooted tree estimates the optimal value of the history by PO-UCT which samples a sequence of decisions and observations as a path starting away from the current node in each Monte-Carlo

simulation. In each Monte-Carlo simulation, PO-UCT chooses the decision that maximizes

$$\widehat{V}(hx) + c\sqrt{\frac{\log \widehat{N}(h)}{\widehat{N}(hx)}}, \quad (2.10)$$

where $\widehat{V}(hx)$, the empirical average of rewards received from all Monte-Carlo simulations till now which have chosen decision x given history h , is an estimation of $V^*(hx)$, $\widehat{N}(hx)$ counts the number of times during monte-carlo simulations that hx has been visited, and $\widehat{N}(h) = \sum_{x \in \mathbb{D}} \widehat{N}(hx)$. PO-UCT takes the search as a multi-armed bandit [42], and (2.10) is similar to the index term in UCB1 [43] which will be introduced in Section 2.3.1 in this thesis. In each simulation, PO-UCT samples one state from the starting node according to its belief, and samples observations and states according to the observation function Ω and transition function Ψ respectively during the forward planning.

The search tree is constructed from nothing but an empty root. When the simulation leaves the tree, an alternative rollout policy [29] will be used to choose a decision and evaluate the reward. After each simulation, one new node, which is the first new history encountered by rollout policy, will be added to the search tree.

When the search of PO-UCT is complete upon meeting some constraints, e.g. number of simulations or running time, the decision maker will choose the decision x maximizing $\widehat{V}^*(hx)$ and receive a real observation o from the environment. Afterwards, POMCP will search only for the updated history $\{h, x, o\}$ encountered. POMCP can overcome the curse of dimensionality by Monte-Carlo sampling in some sense. When the state space $\mathbb{S}$ is large, POMCP use an unweighted particle filter to estimate the belief of a history when the exact Bayes update (2.2) is computationally infeasible. However, in Chapter 3, we will see that POMCP is not feasible when the decision set $\mathbb{D}$ is too large to enumerate in a POMDP.

2.3 Sequential Bandit Problem

The sequential bandit problem is a special reinforcement learning problem which does not assume knowledge of an explicit mathematical model of the MDP. To maximize the expected cumulative discounted reward in infinite time steps case, i.e. $0 < \underline{\rho} < 1$, the Gittins index policy [44] gives the optimal performance. In this thesis, for the sake of simplicity, we only consider the $\underline{\rho} = 1$ case. Either, we don't consider the transition of states, i.e. in the following introduction, we exclude the state and the observation concepts introduced in Definition 2.1. We refer the reader to [44, 45] for the details of the exclusive cases. Unlike the exactly known reward of choosing each decision in the decision set $\mathbb{D}$, in sequential bandits, the decision maker can only receive a *stochastic* reward after each decision-making. In this thesis, with a slight abuse of notation, we still use r to denote the received reward at each time step, which however is a sample from a specific distribution dependent on the chosen decision at each time step. In this section, we introduce the classical independent multi-armed bandit problem and its generalization, the linear stochastic bandit problem. In Chapters 4 and 5, we will use these problems to model the sequential learning of users' properties in recommendation systems over time. The items in the systems serve as the decisions in bandits. Two different kinds of reward will be defined respectively in Chapters 4 and 5.

In the following, we use the terms “decision” and “arm” interchangeably in the formulation since in the bandits literature, the term “arm”, which refers to the arms of slot machines, is often used.

2.3.1 Multi-Armed Bandit Problem

Multi-armed bandit (MAB) problems, introduced by Robbins [46] model the exploration and exploitation trade-off of sequential decision making under uncertainty. In its most basic paradigm [42, 43, 47–49], at each time step, the decision maker is given a decision set $\mathbb{D} = \{1, 2, \dots, K\}$ containing $|\mathbb{D}| = K$ decisions from which he

or she is supposed to select one, and as a response, he or she observes a *stochastic* reward after each decision-making. Specifically, each decision $x \in \mathbb{D}$ is associated with an unknown and fixed probability distribution ν_x , whose expectation is μ_x . Stochastic independence of rewards is assumed for different decisions in the most basic paradigm. At each time step t , the decision maker chooses $X_t \in \mathbb{D}$ and then receives a stochastic reward r_t drawn at random according to ν_{X_t} . We denote the best decision as $* = \arg \max_{x \in \mathbb{D}} \mu_x$. The goal of the decision maker is to minimize the (expected) *cumulative regret* over n time steps defined by

$$\begin{aligned} R(n) &= \sum_{t=1}^n (n\mu_* - \mathbb{E}[r_t]) \\ &= n\mu_* - \sum_{x=1}^K \mu_x \mathbb{E}[N_n^x], \end{aligned} \quad (2.11)$$

where $N_n^x = \sum_{t=1}^n \mathbf{1}_{\{X_t=x\}}$ is the number of times decision x is chosen within n time steps.

The decision made at each time step is based on the information gathered at all the previous time steps. Therefore, in order to minimize the cumulative regret, exploitation of the current empirically best decision and exploration of less frequently chosen decisions should be balanced carefully.

Lai and Robbins [42] established an *asymptotically* cumulative regret lower bound for MAB with ν_x indexed by a single real parameter satisfying some mild assumptions for $x \neq *$:

$$\liminf_{n \rightarrow \infty} \frac{\mathbb{E}[N_n^x]}{\ln n} \geq \frac{1}{D_{\text{KL}}(\nu_x \parallel \nu_*)}, \quad (2.12)$$

where

$$D_{\text{KL}}(\nu_x \parallel \nu_*) := \int \ln \frac{d\nu_x}{d\nu_*} d\nu_x \quad (2.13)$$

is the Kullback-Leibler divergence between the distributions ν_x and ν_* .

In [42], they also proposed policies that attain the above asymptotic lower bound (2.12) such that

$$\mathbb{E}[N_n^x] \leq \ln n \left(\frac{1}{D_{\text{KL}}(\nu_x \parallel \nu_*)} + o(1) \right), \quad (2.14)$$

where $o(1) \rightarrow 0$ as $n \rightarrow \infty$ for all suboptimal $x \neq *$.

Later in 2002, Auer proposed policies [43] that achieve logarithmic cumulative regret uniformly over time, rather than only asymptotically. Specifically, their policies have the order optimal cumulative regret $O(\ln n)$ for finite time steps under bounded support reward distributions situation, where $O(\cdot)$ is the Big-O order.

We extract two policies from [43] as follows. The indicator function $\mathbf{1}_A$ takes value 1 if and only if clause A is true, and 0 otherwise.

Algorithm 2.1 UCB1

Require: Pull each arm once.

- 1: **loop**
 - 2: Pull the arm x that maximizes $\bar{y}_x + \sqrt{\frac{2 \ln t}{N_t^x}}$, where $\bar{y}_x = \frac{\sum_{i=1}^t r_i \mathbf{1}_{\{X_i=x\}}}{N_t^x}$ is the empirical average reward obtained by pulling arm x which is associated with the reward distribution ν_x .
 - 3: Set $t = t + 1$.
 - 4: **end loop**
-

Algorithm 2.2 ϵ_t -greedy

Require: Let $\epsilon_t = \min\{1, \frac{\alpha}{t}\}$ and set parameter $\alpha > 0$.

- 1: **loop**
 - 2: Update ϵ_t .
 - 3: With probability $1 - \epsilon_t$, choose $\arg \max_x \bar{y}_x$ and with probability ϵ_t choose a random arm from $\mathbb{D}$.
 - 4: Set $t = t + 1$.
 - 5: **end loop**
-

The first policy, UCB1, maintains an index, which is the sum of two terms, for every arm in $\mathbb{D}$. The first term, indicating the exploitation of the policy is the empirical average reward of pulling each arm. The second term, indicating the exploration

of the policy, is related to the size of an upper confidence bound for the average reward within which the true expected reward falls with high probability. UCB1 does not take into consideration the variance of the received rewards. In studies like [47], policies that further use the empirical variance of different arms perform better than UCB1 provided that the reward variances of the suboptimal arms are low.

The second policy, ϵ_t -greedy, which will be revised and used in Chapters 4 and 5 in this thesis, chooses a random arm from $\mathbb{D}$ with a decreasing probability ϵ_t to do the exploration while choosing the empirical best arm with the increasing probability $1 - \epsilon_t$ to do the exploitation.

2.3.2 Linear Stochastic Bandit Problem

The stochastic independent rewards of choosing different decisions assumed in Section 2.3.1 enables the decision maker to learn the statistical information of each decision separately. In other works like [50–54], reward dependence between decisions is assumed, which enables the decision maker to gather information for more than one decision at each time step. One specific assumption of dependence is that decisions are vectors containing numerical elements, and the expected reward of choosing each decision is an unknown linear function of the decision vector [53–55]. Formally, the linear stochastic bandit model is defined as follows. Note that the independent d -armed bandit problem can be viewed as a special case of the general linear stochastic bandit problem, where the set of d available decisions serves as a standard orthonormal basis of $\mathbb{R}^d$ [53].

Let $\mathbb{D} \subseteq \mathbb{R}^d$ be a compact set of decisions from which the decision maker has to choose a decision X_t at each time step t . The observed reward r_t after choosing X_t is given by

$$r_t = \langle X_t, \boldsymbol{\theta} \rangle + \eta_t, \quad (2.15)$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a fixed but unknown parameter and η_t is the zero mean independent and identically distributed (i.i.d.) noise. In this thesis, to emphasize that the decisions are vectors in linear stochastic bandits, we use $\mathbf{x}$ instead of x to denote elements in $\mathbb{D}$ when $\mathbb{D} \subseteq \mathbb{R}^d$.

In the standard linear bandit problem, the performance of a policy is measured by the difference between the decision maker's cumulative reward and the cumulative reward achieved by the oracle policy with knowledge of $\boldsymbol{\theta}$. Formally, the goal of the decision maker is to minimize the *cumulative regret* over n time steps defined by

$$R(n) = \sum_{t=1}^n (\langle X^*, \boldsymbol{\theta} \rangle - \mathbb{E}[\langle X_t, \boldsymbol{\theta} \rangle]), \quad (2.16)$$

where $X^* = \arg \max_{\mathbf{x} \in \mathbb{D}} \langle \mathbf{x}, \boldsymbol{\theta} \rangle$ is the optimal decision in the standard linear bandit problem.

In contrast to the finite multi-armed bandit problem where the optimal asymptotic cumulative regret is $O(\ln n)$, in [54, 56], the authors have proved the existence of the case that when $\mathbb{D}$ contains infinitely many arms, the lower bound of the cumulative regret is $\Omega(\sqrt{n})$.

In [53, 54], the authors proposed effective strategies that balance exploration and exploitation based on the *optimism-in-the-face-of-uncertainty* principle. This principle maintains a high probability confidence set for the estimation of parameter $\boldsymbol{\theta}$, and at each time step, the decision maker chooses a pair of decision vector and co-efficient vector from the decision set and the confidence set respectively so that the product of them is maximized. Specifically, their strategies maintain a confidence set $\mathbb{C}_{t-1} \subseteq \mathbb{R}^d$, which is computed from $X_1, X_2, \dots, X_{t-1}$ and $r_1, r_2, \dots, r_{t-1}$, such that the parameter $\boldsymbol{\theta}$ lies in $\mathbb{C}_{t-1} \subseteq \mathbb{R}^d$ with a high probability. The strategies then choose the pair

$$(X_t, \tilde{\boldsymbol{\theta}}) = \arg \max_{(\mathbf{x}, \boldsymbol{\theta}') \in \mathbb{D} \times \mathbb{C}_{t-1}} \langle \mathbf{x}, \boldsymbol{\theta}' \rangle \quad (2.17)$$

They have proved that their strategies has a $O(\sqrt{n}\text{polylog}(n))$ cumulative regret upper bound [53, 54] for the general compact decision set with infinite arms, where $\text{polylog}(n)$ is the polylogarithmic order of n .

2.4 Ulam-Rényi Game

Ulam-Rényi game [30] is an iterative game in which one player chooses a state out of M possible states, and another player asks the first player q -ary questions in order to determine the state chosen. The questions are asked sequentially, and a question at one iteration can be based on the responses to the questions from previous iterations. Ulam-Rényi game with $q = 2$ is equivalent to the binary block coding with noiseless feedback [57] in the communication literature. We will see in Chapter 3 that in crowdsourcing systems, the workers, who are supposed to keep answering questions, serve as the first player in a Ulam-Rényi game, while the crowdsourcers play a similar role to the second player in asking questions.

Formally, in a (M, q, e) Ulam-Rényi game [30–32], a responder first chooses a state $H \in \mathbb{S} = \{1, 2, \dots, M\}$, and a questioner asks the responder a sequence of q -ary questions, which is a q -ary tuple $\mathbf{T} = (\mathbb{T}_1, \mathbb{T}_2, \dots, \mathbb{T}_q)$, to determine H . The q -tuple is interpreted as a question of the form “Which one of the sets $\mathbb{T}_1, \dots, \mathbb{T}_q$ does H belong to?”.

The responder is allowed to lie at most e times. The questioner chooses his or her question at each step based on the previous responses, and his or her goal is to determine H with the minimum number of questions. In a Ulam-Rényi game, a *game status* is defined as $\mathbf{\Lambda} = (\mathbb{A}_0, \mathbb{A}_1, \dots, \mathbb{A}_e)$, where $\mathbb{A}_i$, $i \in \{0, 1, \dots, e\}$, are pairwise disjoint subsets of $\mathbb{S}$. The set $\mathbb{A}_i$ contains all states $s \in \mathbb{S}$ that are potentially H if the responder has lied exactly i times. The initial game status $\mathbf{\Lambda}$ is given by $(\mathbb{S}, \emptyset, \dots, \emptyset)$. At each question, if the current status is $\mathbf{\Lambda} = (\mathbb{A}_0, \mathbb{A}_1, \dots, \mathbb{A}_e)$ and the answer to the question $\mathbf{T} = (\mathbb{T}_1, \mathbb{T}_2, \dots, \mathbb{T}_q)$ is j , where $\mathbb{T}_1, \dots, \mathbb{T}_q$ are

pairwise disjoint sets and $\cup_{j=1}^q \mathbb{T}_j = \cup_{i=0}^e \mathbb{A}_i$, then the game status is updated as:

$$\begin{aligned} \Lambda^j &= (\mathbb{A}_0 \cap \mathbb{T}_j, (\mathbb{A}_0 \setminus \mathbb{T}_j) \cup (\mathbb{A}_1 \cap \mathbb{T}_j), \\ &\quad \dots, (\mathbb{A}_{e-1} \setminus \mathbb{T}_j) \cup (\mathbb{A}_e \cap \mathbb{T}_j)). \end{aligned} \quad (2.18)$$

For example, in the $(4, 2, 1)$ Ulam-Rényi game, the initial status is $\Lambda = (\{1, 2, 3, 4\}, \emptyset)$. Suppose the responder chooses $H = 1$ and the questioner's first question is $\mathbf{T} = (\mathbb{T}_1, \mathbb{T}_2)$ where $\mathbb{T}_1 = \{1, 2\}$ and $\mathbb{T}_2 = \{3, 4\}$. The responder can lie to this question and answer “ H is in $\mathbb{T}_2$ ”, and the questioner updates the game status Λ to $(\{3, 4\}, \{1, 2\})$. This game status tells the questioner that if $H \in \{1, 2\}$, then the responder has lied once, whereas if $H \in \{3, 4\}$, then the responder has not lied. Suppose the second question posed to the responder is $\mathbf{T} = (\{1, 3\}, \{2, 4\})$. Since the responder has already lied once, he or she can only give the answer “ H is in $\mathbb{T}_1$ ”. The game status then becomes $\Lambda = (\{3\}, \{1, 4\})$. Finally, if the questioner poses the question $\mathbf{T} = (\{1\}, \{3, 4\})$ followed by another question $\mathbf{T} = (\{1\}, \{3\})$, the game status $\Lambda = (\emptyset, \{1\})$ is reached, at which point the questioner determines that $H = 1$.

For each $i \in \{0, 1, \dots, e\}$, let $|\mathbb{A}_i|$ be the number of states in $\mathbb{A}_i$. Let $|\Lambda| = (|\mathbb{A}_0|, |\mathbb{A}_1|, \dots, |\mathbb{A}_e|)$, which is called the type of Λ [58]. Same as [59], we define the weight of an element $s \in \mathbb{A}_i$ in status Λ with w iterations left as

$$W_w(i) = \sum_{k=0}^{e-i} \binom{w}{k} (q-1)^k,$$

and the weight of the status Λ as the cumulative weight of all the elements in it, i.e.,

$$\mathcal{V}_w(|\Lambda|) = \sum_{i=0}^e |\mathbb{A}_i| W_w(i).$$

The weight $W_w(i)$ of an element $s \in \mathbb{A}_i$ has an intuitive interpretation: if $H = s$, then there are $\binom{w}{k} (q-1)^k$ ways for the responder to lie k times in the remaining w

iterations, so $W_w(i)$ is the total number of ways of lying for the responder. A game status $\mathbf{\Lambda}$ with high $\mathcal{V}_w(|\mathbf{\Lambda}|)$ implies that the responder has more ways to obfuscate the true state H , making it more difficult for the questioner to determine H .

From Proposition 3.1 of [60], we have $\mathcal{V}_w(|\mathbf{\Lambda}|) = \sum_{j=1}^q \mathcal{V}_{w-1}(|\mathbf{\Lambda}^j|)$. Furthermore, since the responder lies at most e times, Proposition 3.1 of [60] and Theorem 3.1 of [59] show that the questioner is guaranteed to find H in w questions only if

$$\mathcal{V}_w(|\mathbf{\Lambda}|) \leq q^w. \quad (2.19)$$

In [60, 61], the authors proposed sufficient conditions for the questioner to find H successfully in w questions when the number of questions is sufficiently large. Specifically, in [61], the author proved that:

For a fixed e , if $q = 2$, $|\mathbf{\Lambda}| = (|\mathbb{A}_0|, |\mathbb{A}_1|, \dots, |\mathbb{A}_e|)$, and there are sufficiently large constants c , w_0 (dependent on e) so that the following holds for all $w > w_0$: If $\mathcal{V}_w(|\mathbf{\Lambda}|) \leq 2^w$ and

$$|\mathbb{A}_e| > cw^e, \quad (2.20)$$

then there exists a policy that guarantees the questioner to find H in w questions.

In [60], the author proved its generalization to the general q -ary question case:

For a fixed e , when q and w are sufficiently large, the questioner is guaranteed to find H in w questions if

$$w \geq N_{\min}^{[q]}(e) + 1, \quad (2.21)$$

where $N_{\min}^{[q]}(e) = \min\{n \mid M \sum_{j=0}^e \binom{n}{j} (q-1)^j \leq q^n\}$.

Chapter 3

Sequential Multi-Class Labeling in Crowdsourcing

3.1 Introduction

In a crowdsourcing platform, workers are given a task, like classifying or labeling an object in a picture, to perform. The crowdsourcer then makes a final decision based on the collective answers from all participating workers. For example, in [62], crowdsourcing was used to produce taxonomies whose quality approaches that of human experts. Crowdsourcing platforms like Amazon Mechanical Turk [63] typically have many participating workers. The goal is to make use of the abundance of workers to perform simple but tedious microtasks that do not require much domain expertise. However, workers may be highly unreliable [64, 65]. Therefore, the microtasks are usually designed to be simple binary questions [7]. In this chapter, we use the terms microtask and question interchangeably by assuming that workers are always required to answer a question posed by the crowdsourcer.

To improve the reliability of the final decision, various inference algorithms and question allocation methods have been proposed. For example, in [6, 8], the authors proposed a task assignment scheme using a bipartite graph to model the affinity of

workers for different binary tasks and an iterative algorithm based on belief propagation to infer the final decision from the workers' responses. In [9], the authors proposed an iterative inference algorithm based on a spectral method. They assumed that tasks have different difficulties and workers have different reliabilities using a generalized Dawid-Skene model [66]. A multi-class labeling problem was considered in [10], in which an allocation algorithm is developed to assign tasks to different workers, and an inference method achieving an order-optimal redundancy-accuracy trade-off was developed. Redundancy here refers to assigning the same task to multiple workers and using a majority voting rule to infer the final answer. Some researchers have considered asking additional gold questions inserted amongst the actual work tasks to determine the reliability of workers [67]. In [68], the authors presented a strategy to improve workers' reliability by teaching them classification rules. Several incentive strategies to motivate workers to participate in crowdsourcing have been studied in [69, 70]. However, how to design the micro-tasks or questions for the workers has not been addressed in the aforementioned works.

In this chapter, we assume that the crowdsourcer wishes to solve an M -ary multi-class labeling problem. From coding theory [71, 72], we know that it is possible to reduce the inference error through the use of error correction codes. The reference [7] developed an algorithm called Distributed Classification Fusion using Error-Correcting Codes (DCFEC) using error-correction coding to divide a single multi-class labeling task into binary questions, which are then assigned to the workers. The classification is then done by performing a Hamming distance decoding of all the workers' binary responses, which can be treated as the noisy versions of a coded message. We note that DCFEC is based on a non-feedback coding strategy. One can use a feedback coding strategy to further reduce the misclassification probability. In this chapter, we consider the same multi-class labeling problem as [7] but now allow questions posed to the workers to depend on the workers' previous responses. Our goal is to develop a sequential approach to design questions for the crowdsourcing platform. We assume that at each questioning round, the

crowdsourcer asks a group of workers a q -ary question (with $q \leq M$ and to be optimized), which we design based on the workers' answers in previous rounds. The q -ary question is further decomposed into binary questions for each worker using a similar error correction coding approach as DCFECC.

We formulate the sequential q -ary question design problem as a partially-observable Markov decision process (POMDP) [27–29, 39, 40], which is however intractable due to its extremely large decision space. This difficulty together with the well-known curse of dimensionality [28, 29] in POMDP makes optimally solving POMDP computationally intractable. We therefore develop approximate methods to solve the POMDP.

Our first strategy is to approximate the POMDP using a q -ary Ulam-Rényi game [30–32] introduced in Section 2.4. The first player may answer some of the questions posed wrongly. The first player's choice of the state corresponds to the true state of our multi-class labeling problem, while giving a wrong answer to a question corresponds to the unreliability of a worker in our crowdsourcing platform. We show how to find the best parameter q in order to achieve an optimal trade-off between the classification accuracy and the number of iterations of questions required. As finding the optimal strategy for a Ulam-Rényi game is in general intractable, heuristics for solving a *binary* Ulam-Rényi game have been proposed in [30, 31]. However, to the best of our knowledge, strategies for a general q -ary Ulam-Rényi game have been proposed only when the number of questions allowed is sufficiently large. In this chapter, we propose an efficient heuristic, which is however suboptimal in general.

Furthermore, based on our proposed Ulam-Rényi strategy, we further propose a decision sampling strategy for the POMDP to sample a small subset of decisions so that standard POMDP solvers like PBVI [27, 28] and POMCP [29] can be used. Simulations suggest that this sampling strategy is better than uniform sampling.

The rest of this chapter is organized as follows. In Section 3.2, we present our system model and assumptions, and formulate the crowdsourcing problem as a

POMDP. In Section 3.3, we introduce the Ulam-Rényi game, and develop a novel heuristic to solve it. We then apply this strategy to an approximation of our POMDP formulation. In Section 3.4, we describe our new decision sampling strategy for traditional POMDP solvers. In Section 3.5, we present simulations to compare the performance of our approaches with other strategies.

3.2 Problem Formulation

Consider a crowdsourcing platform that allows a crowdsourcer to pose questions or tasks to workers on the platform. In this chapter, a crowdsourcer wishes to solve a M -ary multi-class labeling problem: determine a hidden state $H \in \mathbb{S} = [1, \dots, M]$ (in this chapter, we abuse $[a, b]$ to denote the set of integers $\{a, a + 1, \dots, b\}$ since real line intervals are not used in this chapter) with the help of groups consisting of N workers each. For example, the crowdsourcer may wish to classify an image of a dog into one of $M = 9$ breeds: Dachshund, Irish Water Spaniel, Afghan Hound, Bull Terrier, Westie, Alaskan Malamute, American Pit Bull Terrier, Fox Terrier and Rough Collie. In this case, we have $\mathbb{S} = \{1, 2, \dots, 9\}$, where each state corresponds to a particular dog breed in the order listed. However, since workers may not be experts in such image classification tasks, and can only answer simple multiple choice questions, the crowdsourcer asks each group of workers a sequence of q -ary questions, where $q \in [2, M]$. A q -ary question can be represented as a q -tuple $\mathbf{T} = (\mathbb{T}_1, \mathbb{T}_2, \dots, \mathbb{T}_q)$, where the sets $\mathbb{T}_j$, $j \in [1, q]$, are pairwise disjoint subsets of $\mathbb{S}$. The q -tuple is interpreted as a question of the form “Which one of the sets $\mathbb{T}_1, \dots, \mathbb{T}_q$ does H belong to?”. At each iteration, the q -ary question posed is designed based on the responses from previous iterations. Each q -ary question is assigned to the whole group of N workers, and is further broken down into simpler microtasks that are assigned to each individual worker within the group.

For a concrete illustration, consider again the example of classifying an image of a dog into one of nine breeds as described above. Suppose that the crowdsourcer

uses ternary ($q = 3$) questions. Then the question, “Does the dog in the image have drop, erect or half-erect ears?”, can be represented as the ternary tuple $\mathbf{T} = (\{1, 2, 3\}, \{4, 5, 6\}, \{7, 8, 9\})$ since $\{1, 2, 3\}$ corresponds to breeds with drop ears, $\{4, 5, 6\}$ corresponds to breeds with erect ears, and $\{7, 8, 9\}$ corresponds to breeds with half-erect ears. The crowdsourcer then assigns this question to the first group of workers. Suppose that he or she receives the answer “The dog has drop ears”, then his or her next question could be “Does the dog in the image have smooth, curly, or long and straight fur?”, which can be represented as $\mathbf{T} = (\{1, 4, 7\}, \{2, 5, 8\}, \{3, 6, 9\})$. By iteratively asking different questions to groups of workers, the crowdsourcer is able to narrow down the exact dog breed in the image.

To shed light in the direction of translating questions in terms of pairwise disjoint sets to human-understandable summaries automatically, we use a simpler 4-breed dog labeling problem as an illustration. Together with dog images, the crowdsourcer is supposed to upload specific human-understandable features for each breed. For example, we may have a feature set F_i , containing human-understandable features for each hidden state i . Assume we have $\mathbb{S} = \{1, 2, 3, 4\}$, corresponding to Breed 1 (Dachshund), Breed 2 (Afghan Hound), Breed 3 (Westie) and Breed 4 (Alaskan Malamute). Furthermore, we have $F_1 = \{\text{short legs, drop ears}\}$, $F_2 = \{\text{long legs, drop ears}\}$, $F_3 = \{\text{short legs, erect ears}\}$ and $F_4 = \{\text{long legs, erect ears}\}$. With a binary question $\mathbf{T} = (\{1, 3\}, \{2, 4\})$, we may translate it using $(\cup_{i \in \mathbb{T}_1} F_i) \setminus (\cup_{i \in \mathbb{T}_2} F_i) = \{\text{short legs}\}$ and $(\cup_{i \in \mathbb{T}_2} F_i) \setminus (\cup_{i \in \mathbb{T}_1} F_i) = \{\text{long legs}\}$. The question: “Which feature does the class in the image have? A. It has $\{\text{short legs}\}$. B. It has $\{\text{long legs}\}$.” is therefore automatically generated. For more complex feature combinations, options with multiple features like “A. short legs or long fur.” and “B. long legs and short fur.”, are possible.

3.2.1 Binary Microtasks with Code Matrix

Suppose a q -ary question $\mathbf{T}$ is posed to a worker group g , consisting of N workers randomly chosen from the worker pool. Let $y_{g,k} \in [1, q]$ be the opinion or response of a worker $k \in [1, N]$ in the group if he or she is posed the question $\mathbf{T}$. We assume that each worker k has an associated reliability $\Lambda_{g,k}^q$, drawn independently from a common distribution, with mean $\underline{\mu}^{[q]}$. Let $\Upsilon_g^q = (\Lambda_{g,k}^q)_{k=1}^N$. Conditioned on H and Υ_g^q , we assume that workers' responses are independent from each other, and the response of worker k in group g has the following probability mass function:

$$p(y_{g,k} \mid H, \Upsilon_g^q) = \begin{cases} \Lambda_{g,k}^q & \text{if } H \in \mathbb{T}_{y_{g,k}}, \\ \frac{1 - \Lambda_{g,k}^q}{q-1} & \text{otherwise.} \end{cases} \quad (3.1)$$

Note that we have assumed that the distribution generating $\Lambda_{g,k}$ depends only on the number q of possible responses, and not on the specific question $\mathbf{T}$. This is because workers are typically non-experts without specific domain knowledge of the labeling problem assigned. Furthermore, we will typically assume in our simulations that $\underline{\mu}^{[q]}$ is a decreasing function of q , i.e., workers become more unreliable as the question posed becomes more complex. Indeed, it has been observed that workers on crowdsourcing platforms typically do not have enough expertise to answer non-binary questions reliably [73]. Therefore, we first transform the q -ary question posed to each worker group into binary questions, which are then assigned to individual workers within the group.

In the DCFECC approach proposed by [7], a code matrix is designed to determine what questions are assigned to each worker to solve a M -ary labeling problem. For N workers, a code matrix is a $M \times N$ binary matrix, where each column corresponds to a binary question for each worker, and each row corresponds to a code for each state of the M -ary labeling problem. The responses from all the N workers can then be viewed as a noisy version of one of the rows in the code matrix, and a Hamming distance decoder is used to decode for the hidden state H .

In our proposed approach, a q -ary question $\mathbf{T} = (\mathbb{T}_1, \dots, \mathbb{T}_q)$ is posed at each time to a worker group. We can interpret this as a q -ary labeling problem, and let $\mathbf{G}_q$ be the corresponding $q \times N$ code matrix generated by DCFECC when applied to solving this q -ary labeling problem.

FIG. 3.1: Code matrix representation of the DCFECC approach

The j -th column of the code matrix $\mathbf{G}_q$ represents the binary question $\mathbf{T}' = (\mathbb{T}'_0, \mathbb{T}'_1)$ posed to the j -th worker, where $\mathbb{T}'_k = \{l \in [1, q] : \mathbf{G}_q(l, j) = k\}$ for $k = 0, 1$ and $\mathbf{G}_q(l, j)$ being the (l, j) -th element of $\mathbf{G}_q$. Let $\mathbf{u} \in \{0, 1\}^N$ be the vector of responses from the N workers. The j -th element in vector $\mathbf{u}$ is set as bit k if the j -th worker decides that $\mathbb{T}'_k$ contains the correct class. As workers may be unreliable, the responses from the N workers can be viewed as a noisy version of one of the rows in $\mathbf{G}_q$. Using the same decision rule in [71], we then decode $\mathbf{u}$ to $\mathbb{T}_l$ if the l -th row of $\mathbf{G}_q$ has the smallest Hamming distance to $\mathbf{u}$, with ties broken randomly. The set of responses whose Hamming distance to the l -th row of $\mathbf{G}_q$ is minimal is called the decision region of $\mathbb{T}_l$.

Under the worker reliability model (3.1), it is shown in [7] that for any $\mathbf{i} = [i_1, i_2, \dots, i_N] \in \{0, 1\}^N$, we have

$$\begin{aligned} \mathbb{P}(\mathbf{u} = \mathbf{i} \mid H \in \mathbb{T}_l) \\ = \prod_{j=1}^N \left[1 - i_j + 2(i_j - 1) \left(\underline{\mu}^{[q]} \mathbf{G}(l, j) + \frac{1 - \underline{\mu}^{[q]}}{q - 1} \sum_{k \neq l} \mathbf{G}(k, j) \right) \right], \end{aligned} \quad (3.2)$$

where $\mathbf{G}(l, j)$ is the (l, j) -th element of $\mathbf{G}_q$. Let o be the row number of $\mathbf{G}_q$ found by the Hamming distance decoder. We have

$$p(o \mid H \in \mathbb{T}_l) = \sum_{\mathbf{i}} \mathbb{P}(\mathbf{u} = \mathbf{i} \mid H \in \mathbb{T}_l) C^o(\mathbf{i}), \quad (3.3)$$

where

$$C^o(\mathbf{i}) = \begin{cases} \frac{1}{n_{\mathbf{i}}} & \text{if } \mathbf{i} \text{ is in the decision region of } \mathbb{T}_o, \\ 0 & \text{otherwise,} \end{cases}$$

with $n_{\mathbf{i}}$ being the number of decision regions $\mathbf{i}$ belongs to. We call the $q \times q$ matrix $\mathbf{P}_q$ whose (l, o) -th element is $p(o \mid H \in \mathbb{T}_l)$, the *performance matrix* of the code matrix $\mathbf{G}_q$. Similar to DCFECC, we choose the code matrix $\mathbf{G}_q$ to minimize $\sum_{l=1}^q \sum_{o \neq l} p(o \mid H \in \mathbb{T}_l)/q$.

3.2.2 POMDP Formulation

We assume that no prior information about H is available. Therefore, we choose a uniform prior over $\mathbb{S}$ for H . At each time t , the crowdsourcer chooses a decision x_t : to either pose a question to worker group g , or to stop and make a final decision. Let τ be the stopping time. Since most crowdsourcing platforms like Amazon Mechanical Turk require the crowdsourcer to financially compensate a worker for each task he or she completes, the crowdsourcer's expenditure increases with increasing τ . Let $\underline{b}$ be the budget of the crowdsourcer so that we have the constraint $\tau \leq \underline{b}$. Let $\mathbb{Q}(q)$ be the set of all q -tuples $\mathbf{T}$. For each time $t < \tau$, the

crowdsourcer chooses a tuple $x_t \in \mathbb{Q}(q)$, poses it to the worker group g , and fuses all the answers from the workers in group g to obtain an answer $o_t \in [1, q]$. The choice of the tuple x_t is determined by a policy $\phi_t : \mathbb{Q}(q)^{t-1} \times [1, q]^{t-1} \mapsto \mathbb{Q}(q)$. At time τ , the decision x_τ is given by a decision rule $\hat{H} : \mathbb{Q}(q)^{\tau-1} \times [1, q]^{\tau-1} \mapsto \mathbb{S}$. Let $\mathcal{R} = \mathbb{P}(\hat{H} = H)$ be the probability that the final decision $\hat{H}$ is correct. We call $\mathcal{R}$ the *reliability* of the strategy. We define a *policy* to be $(q, \tau, (\phi_t)_{t=1}^{\tau-1}, \hat{H})$. Let the cost of each question posed be ψ . The aim of the crowdsourcer is to

$$\begin{aligned} & \max \mathcal{R} - \psi(\mathbb{E}[\tau] - 1) \\ & \text{subject to } \tau \in [1, \underline{b}], \end{aligned} \tag{3.4}$$

over all policies.

Let $\mathbf{b}_0(s) = 1/M$ for all $s \in \mathbb{S}$. As we have shown in (2.1), if history $h_t = \{\mathbf{b}_0, x^1, o^1, \dots, x^t, o^t\}$, the belief at time t is

$$\mathbf{b}_t(s) = \mathbb{P}(H = s \mid h_t),$$

for each $s \in \mathbb{S}$. It can be shown that $\mathcal{R}$ is a function of $\mathbf{b}_\tau = (\mathbf{b}_\tau(s))_{s \in \mathbb{S}}$ since the optimal decision rule at time τ , for a given $(q, \tau, (\phi_t)_{t=1}^{\tau-1})$ is to choose $\hat{H} = \arg \max_{s \in \mathbb{S}} \mathbf{b}_\tau(s)$. Thus, to solve (3.4), it suffices to keep track of the sufficient statistics $\mathbf{b}_t = (\mathbf{b}_t(s))_{s \in \mathbb{S}}$, which can be computed sequentially via Bayes' rule as

$$\mathbf{b}_t(s) = \begin{cases} \frac{p(o_t \mid H=s, x_t=\mathbf{T}) \mathbf{b}_{t-1}(s)}{p(o_t \mid \mathbf{b}_{t-1}, x_t=\mathbf{T})} & \text{if } s \in \cup_{j=1}^q \mathbb{T}_j, \\ \mathbf{b}_{t-1}(s) & \text{otherwise.} \end{cases} \tag{3.5}$$

where $p(o_t \mid H = s, x_t = \mathbf{T})$ is the (l, o_t) -th element of the performance matrix $\mathbf{P}_q$ if $s \in \mathbb{T}_l$, and

$$\begin{aligned} & p(o_t \mid \mathbf{b}_{t-1}, x_t = \mathbf{T}) \\ &= \frac{\sum_{s \in \cup_{j=1}^q \mathbb{T}_j} p(o_t \mid H = s, x_t = \mathbf{T}) \mathbf{b}_{t-1}(s)}{\sum_{s \in \cup_{j=1}^q \mathbb{T}_j} \mathbf{b}_{t-1}(s)}. \end{aligned}$$

The formulation in (3.4) is an optimal stopping problem under partial observation [74] problem which can be viewed as a POMDP, as we have described in Section 2.2, in which we observe only the belief $\mathbf{b}_t$ but not the hidden state H at each time t .

However, due to the large size of the decision space $\mathbb{Q}(q)$, solving this POMDP exactly is intractable [28]. Furthermore, we need to optimize over $q \in [2, M]$. Most POMDP solvers, including PBVI [27, 28] and POMCP [29], enumerate over all possible decisions to find the optimal decision. When the decision space is large, such enumeration is no longer feasible. Several solvers [75–77] have even been proposed for continuous decision spaces under some technical conditions. However, to the best of our knowledge, there is no general POMDP solver for large discrete decision spaces. Therefore, in Section 3.3, we propose a heuristic approach based on the Ulam-Rényi game to solve (3.4). In addition, as part of this heuristic approach, we obtain a sampling strategy to sample a small subset of decisions from $\mathbb{Q}(q)$, which can be used in tandem with traditional POMDP solvers like PBVI and POMCP, in Section 3.4.

3.3 Ulam-Rényi Game Approach

In this section, we first briefly introduce the concept of a Ulam-Rényi game, and propose a novel mixed-integer quadratic programming (MIQP) formulation to finding a heuristic strategy for the Ulam-Rényi game. We then approximate our POMDP in (3.4) with a Ulam-Rényi game and use our proposed Ulam-Rényi strategy to solve it. Finally, we propose a decision sampling approach based on the Ulam-Rényi game, which can be used in tandem with standard POMDP solvers.

3.3.1 A Heuristic Strategy for the q -ary Ulam-Rényi Game

As we have introduced in Section 2.4, in a (M, q, e) Ulam-Rényi game [30–32], a responder first chooses a number $H \in \mathbb{S}$, and a questioner asks the responder a

sequence of q -ary questions to determine H . The responder is allowed to lie at most e times. The questioner chooses his or her question at each step based on the previous responses, and his or her goal is to determine H with the minimum number of questions. The Ulam-Rényi game is similar to our crowdsourcing problem: the questioner is our crowdsourcer, while the responder is a group of workers. Also the *game status* $\mathbf{\Lambda} = (\mathbb{A}_0, \mathbb{A}_1, \dots, \mathbb{A}_e)$ can be viewed as a discretization of belief $\mathbf{b}$. The difference is that in our crowdsourcing formulation, the workers do not know H a priori, therefore a wrong answer is modeled to be stochastic and there is no upper limit e to the number of wrong answers the workers can give. In the following, we first give a brief overview of the Ulam-Rényi game and present a heuristic strategy to solve it. We then apply this strategy to our problem. In the following, we use the same notation as we have defined in Section 2.4.

In general, it is intractable to find an optimal strategy for a (M, q, e) Ulam-Rényi game [30, 59]. As we have shown in Section 2.4, search strategies that are near optimal in the literature exist only when the number of questions is sufficiently large. In the following, we propose a heuristic adaptive strategy which is applicable for our work in this thesis.

Consider the case where $\mathcal{V}_w(|\mathbf{\Lambda}|) = q^w$ for some w . If the next question yields $\mathcal{V}_{w-1}(|\mathbf{\Lambda}^j|) > q^{w-1}$ for some $j \in [1, q]$, (2.19) shows that the responder can answer j to prevent the questioner from finding H . Therefore, the questioner should use a question $\mathbf{T}$ such that $\mathcal{V}_{w-1}(|\mathbf{\Lambda}^j|)$ is approximately the same for all $j \in [1, q]$. For each $j \in [1, q]$, we let

$$\mathbb{T}_j = \bigcup_{i=0}^e \mathbb{T}_{j,i},$$

where $\mathbb{T}_{j,i} = \mathbb{T}_j \cap \mathbb{A}_i$, $\bigcup_{j=1}^q \mathbb{T}_{j,i} = \mathbb{A}_i$ and $\mathbb{T}_{j,i}$, $j \in [1, q]$, are pairwise disjoint sets. We use $|\mathbb{T}_j|$ to denote $(|\mathbb{T}_{j,0}|, |\mathbb{T}_{j,1}|, \dots, |\mathbb{T}_{j,e}|)$.

We observe from our experiments that $W_w(i)$ is usually much larger for smaller values of i . For example, when $q = 4$ and $e = 7$, we have $W_{10}(0) = 497,452$ while $W_{10}(7) = 1$. This suggests the following heuristic: We first determine the value of $|\mathbb{T}_{j,i}|$ for small values of i and then use $|\mathbb{T}_{j,i}|$ with larger values of i to even out the weights $\mathcal{V}_{w-1}(|\mathbf{\Lambda}^j|)$ for $j \in [1, q]$.

Suppose $|\mathbb{T}_{j,0}|, \dots, |\mathbb{T}_{j,i-1}|$, for all $j \in [1, q]$ have been determined. Subsequently for $i \in [0, e]$, let $|\mathbf{\Lambda}_i^j| = (|\mathbb{A}_0^j|, \dots, |\mathbb{A}_i^j|, |\mathbb{A}_{i+1}^j|, 0, \dots, 0)$, where

$$|\mathbb{A}_i^j| = |\mathbb{A}_{i-1}| - |\mathbb{T}_{j,i-1}| + |\mathbb{T}_{j,i}|,$$

with $|\mathbb{A}_{-1}| = |\mathbb{T}_{j,-1}| = |\mathbb{T}_{j,i+1}| = 0$. We propose to solve the following optimization problem:

$$\begin{aligned} \min_{(\mathbb{T}_{j,i})_{j=1}^q} \quad & \sum_{j,l} (\mathcal{V}_{w-1}(|\mathbf{\Lambda}_i^j|) - \mathcal{V}_{w-1}(|\mathbf{\Lambda}_i^l|))^2 \\ \text{subject to} \quad & |\mathbb{T}_{j,i}| \in [0, |\mathbb{A}_i|], \quad \sum_{i=1}^q |\mathbb{T}_{j,i}| = |\mathbb{A}_i|. \end{aligned} \tag{3.6}$$

Let $\mathcal{V}_j = \sum_{k=0}^{i-1} W_{w-1}(k) |\mathbb{A}_i^j| - W_{w-1}(i) |\mathbb{T}_{j,i-1}|$. The objective function in (3.6) can be written as

$$\sum_{j,l} (\mathcal{V}_j - \mathcal{V}_l + (W_{w-1}(i) - W_{w-1}(i+1))(|\mathbb{T}_{j,i}| - |\mathbb{T}_{l,i}|))^2,$$

where $W_{w-1}(e+1) = 0$. Then, the optimization problem (3.6) can be reformulated as a MIQP as follows:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \frac{1}{2} \mathbf{y}^\top \mathbf{F} \mathbf{y} + \mathbf{c}^\top \mathbf{y} \\ \text{subject to} \quad & \mathbf{1}^\top \mathbf{y} = |\mathbb{A}_i| \\ & \mathbf{y}(j) \in [0, |\mathbb{A}_i|], j \in [1, q] \end{aligned} \tag{3.7}$$

where $\mathbf{y}^\top$ is the transpose of $\mathbf{y}$, $\mathbf{F} \in \mathbb{R}^{q \times q}$ has diagonal entries $(q-1) \times (W_{w-1}(i) - W_{w-1}(i+1))$, and for $j \neq k$, its (j, k) -th entry is $W_{w-1}(i+1) - W_{w-1}(i)$. The vector

Algorithm 3.1 Heuristic strategy for a (M, q, e) Ulam-Rényi game.

Require: Current status $\Lambda = (\mathbb{A}_0, \mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_e)$, with w questions left.

Ensure: $\mathbf{T}$

Let k denote the smallest index s.t. $|\mathbb{A}_k| \neq 0$. Let $\delta = \text{mod}(|\mathbb{A}_k|, q)$, and $\omega = \lfloor \frac{|\mathbb{A}_k|}{q} \rfloor$.

for $j = 1$ to δ **do**

Set $|\mathbb{T}_{j,k}| = \omega + 1$.

end for
for $j = \delta + 1$ to q **do**

Set $|\mathbb{T}_{j,k}| = \omega$.

end for

Set $|\mathbb{T}_{j,i}| = 0$ for all $j \in [1, q]$ and $i \in [0, k - 1]$.

for $i = k + 1$ to e **do**
if $|\mathbb{A}_i| \neq 0$ **then**

Solve (3.7) to obtain an optimal $\mathbf{y}$.

for $j = 1$ to q **do**

Set $|\mathbb{T}_{j,i}| = \mathbf{y}(j)$.

end for
else

Set $|\mathbb{T}_{j,i}| = 0$, for all $j \in [1, q]$.

end if
end for

For all $i \in [0, e]$, $\mathbb{A}_i$ is arbitrarily partitioned by randomly choosing $|\mathbb{T}_{j,i}|$ elements from it to form $\mathbb{T}_{j,i}$, for all $j \in [1, q]$.

$\mathbf{c} \in R^q$ has j -th entry $\mathbf{c}(j) = \sum_{k=0}^{q-1} (\mathcal{V}_j - \mathcal{V}_k)$, and $\mathbf{y}(j)$ denotes the j -th entry of vector $\mathbf{y}$.

After obtaining the optimal $(|\mathbb{T}_{j,i}|)_{j \in [1, q]}$ from solving (3.7), $\mathbb{A}_i$ can be arbitrarily partitioned by randomly choosing $|\mathbb{T}_{j,i}|$ elements from it to form $\mathbb{T}_{j,i}$. However, more advanced partition strategies for $\mathbb{A}_i$ are possible if we consider automatically translating questions in terms of pairwise disjoint sets to human-understandable questions, based on more complicated feature combination techniques than which has been shown in Section 3.2. The above heuristic is presented in detail in Algorithm 3.1.

Let $B(q, e)$ be the minimum number of questions required to determine H if at most e of the questions have wrong responses. We propose a heuristic to compute $B(q, e)$ in Algorithm 3.2. The Ulam-Rényi game ends when the game status contains only a single state. Given the initial game status $\Lambda = \{\mathbb{S}, \emptyset, \dots, \emptyset\}$ and the number of

Algorithm 3.2 Heuristic to compute $B(q, e)$.

Require: M, q, e **Ensure:** $B(q, e)$ Set $w = N_{\min}^{[q]}(e) = \min\{n \mid M \sum_{j=0}^e \binom{n}{j} (q-1)^j \leq q^n\}$.**while 1 do**Use Algorithm 3.1 to generate a q -ary Ulam-Rényi tree whose depth is w and root is the initial status.**if** every leaf node's game status contains only one state **then**

break;

else $w = w + 1$;**end if****end while****return** $B(q, e) = w$;

questions w , we use Algorithm 3.1 to generate a q -ary tree whose depth is w and root corresponds to the initial game status. Each node in the tree corresponds to a game status. The j -th child of a node Λ is the status Λ^j updated from answer j as defined in (2.18). To find $B(q, e)$ then corresponds to finding the minimum w that leads to a q -ary tree in which every leaf node has a game status that contains only one state. We initialize w as $N_{\min}^{[q]}(e) = \min\{n \mid M \sum_{j=0}^e \binom{n}{j} (q-1)^j \leq q^n\}$ since $B(q, e) \geq N_{\min}^{[q]}(e)$ [58, 59]. We then increment w by one each time until the game status at every leaf node contains only one state. We call this a (M, q, e) *Ulam-Rényi tree*. The above procedure is summarized in Algorithm 3.2.

To verify the performance of our heuristics, we compare our estimated $B(2, e)$ with that of the optimal $(M, 2, e)$ Ulam-Rényi game strategy in [30], where $M = 2^m$ is a power of 2. (Note that finding optimal strategies for $q > 2$ is an open problem.) The results are shown in Table 3.1, in which the numbers in brackets indicate the values found by [30] if our method differs from the optimal values. We see that our algorithm computes $B(2, e)$ correctly in most cases.

TABLE 3.1: Estimated $B(2, e)$ of the $(M, 2, e)$ Ulam-Rényi game where $M = 2^m$.

m	e							
	1	2	3	4	5	6	7	8
1	3	5	7	9	11	13	15	17
2	5	8	11	14	17	20	23	26
3	6	9	12	15	18	21	24	27
4	7	10	13	16	19	22	25	28
5	9	12	15	18	21	24	27	30
6	10	13	16	19	22	25	28	31
7	11	14	17	20	23	26	29	32
8	12	15	18	21	24	27	30	33
9	13	17	20	23	26	29	32	35
10	14	18	21	24	27	30	33	36
11	15	19	22	25	28	31	34	37
12	17	20	23	27	30	33	36	39
13	18	21	25	28	31	34	37	40
14	19	22	26	29	32	35	39 (38)	42 (41)
15	20	24	27	30	34	37	40	43
16	21	25	28	32	35	38	41	44

3.3.2 POMDP Approximation

We now approximate the POMDP in (3.4) using a (M, q, e) Ulam-Rényi game, and optimize over (q, e) . For a given (q, e) , we treat the crowdsourcer as the questioner in the (M, q, e) Ulam-Rényi game, and each group of workers as the responder. We can think of the group of workers giving an incorrect response after decoding via the code matrix as the responder in the Ulam-Rényi game lying. However, in our crowdsourcing problem, the worker groups are not constrained to making at most e errors. Therefore, after $B(q, e)$ questions, the crowdsourcer is not guaranteed to infer the correct hidden state H . Let $\mathcal{R}(q, e)$ be the probability that the crowdsourcer finds the correct H by applying our proposed heuristic in Algorithm 3.1 iteratively till the game status contains only a single state, which is then declared to be the inferred hidden state $\hat{H}$. Let $\hat{B}(q, e)$ be the maximum number of iterations required as computed in Algorithm 3.2. Then we have $\tau \leq \hat{B}(q, e) + 1$.

Recall that $\mathbf{P}_q$ is the performance matrix of the DCFECC code matrix $\mathbf{G}_q$ used to transform the q -ary question $\mathbf{T}$ into binary microtasks for each worker. Let

$\underline{p} = \min_{i \in [1, q]} \mathbf{P}_q(i, i)$, where $\mathbf{P}_q(i, i)$ is the (i, i) -th element of $\mathbf{P}_q$ and denotes the least probability that the question is answered correctly. Since $\mathcal{R}(q, e)$ is the probability that all the worker groups make at most e errors in at most $\hat{B}(q, e)$ questions, we have

Proposition 3.1.

$$\mathcal{R}(q, e) \geq \hat{\mathcal{R}}(q, e) =_{\text{def}} \sum_{k=0}^e \binom{\hat{B}(q, e)}{k} (1 - \underline{p})^k \underline{p}^{\hat{B}(q, e) - k}, \quad (3.8)$$

Proof. See Section 3.A. □

In place of the problem (3.4), we consider instead the following optimization problem:

$$\begin{aligned} \max_{q, e} L(q, e) &= \hat{\mathcal{R}}(q, e) - \psi \hat{B}(q, e) \\ \text{subject to } \hat{B}(q, e) &\leq \underline{b} - 1, \\ q &\in [2, M], \quad e \in [0, \infty). \end{aligned} \quad (3.9)$$

Since the problem (3.9) is a discrete optimization problem, it is in general computationally intractable to find the exact optimum. For small M , the solution space is relatively small and the solution can be found through enumeration. For large M , an integer programming method like branch and bound [78, 79] or local search strategy [80] can be used to find an approximate solution.

Let (q^*, e^*) be the optimal solution of (3.9). We call the (M, q^*, e^*) Ulam-Rényi game strategy found using Algorithm 3.1 together with the use of the DCFECC code matrix to transform the q^* -ary questions into binary questions, the Ulam-Rényi Sequential Questioning Strategy (URSQS).

3.4 Sampling Using Ulam-Rényi Tree

To solve the POMDP (3.4) directly using standard POMDP solvers is computationally expensive without using some sampling strategy to reduce the size of the decision space. A typical sampling strategy is uniform random sampling. In this section, we propose a new sampling strategy based on the Ulam-Rényi tree.

Let $\hat{\mathbb{Q}}$ denote the sampled subset of the decision space, where $|\hat{\mathbb{Q}}|$ is predetermined. Let (q^*, e^*) be the solution of (3.9). To sample each decision in $\hat{\mathbb{Q}}$, use Algorithm 3.2 to find a (M, q^*, e^*) Ulam-Rényi tree. We then randomly sample $|\hat{\mathbb{Q}}|$ nodes in the tree, and include the corresponding questions that generated these nodes into $\hat{\mathbb{Q}}$. We call this the Ulam-Rényi tree (URT) sampling strategy.

3.5 Simulations and Performance Evaluation

In this section, we perform simulations¹ and compare the performance of our proposed approaches with the non-sequential DCFECC approach, and strategies found using the standard POMDP solvers PBVI [27, 28] and POMCP [29] with uniform sampling.

We perform simulations using $N = 10$, and $\underline{\mu}^{[q]} = \rho q^{-0.2}$. We vary the number of classes M , the budget $\underline{b}$, cost of each question ψ , and the parameter ρ , which controls the workers' reliability via $\underline{\mu}^{[q]}$. We choose $\underline{\mu}^{[q]} = \rho q^{-0.2}$ to be a decreasing function of q , as a worker's reliability typically decreases with increasing q [73]. The parameter ρ controls the workers' general reliability. A larger ρ indicates that the workers are more reliable. In each simulation, we perform 50,000 trials using URSQS, DCFECC, and the PBVI solver. Since DCFECC is a non-sequential strategy, we assume that it uses $(\underline{b} - 1)N$ workers, which is the maximum total number of workers used by URSQS and the POMDP solvers. We perform 2000 trials when using the POMCP solver since it is significantly more computationally complex.

¹Simulation codes can be found in <https://github.com/KANGQIYU/Sequential-Multi-Class-Labeling-in-Crowdsourcing>

To compare the performances of different approaches, we compute the empirical average, over all the trials, of the objective function in (3.4). For convenience, this average is referred to as the *average reward*.

3.5.1 Comparison of Sampling Strategies

When using the standard POMDP solvers, we choose the decision space sample size $|\hat{\mathbb{Q}}|$ according to the number of classes M . For the PBVI solver, we also randomly sample a finite set of beliefs, $\mathbb{B}$.

The performance of PBVI depends on the size of $\mathbb{B}$ relative to M . PBVI first computes the solutions for all samples in $\mathbb{B}$ to approximate the solutions for the full belief space. With denser belief samples, PBVI can converge to the optimal value [27]. However, since the full belief space grows exponentially with M , it becomes computationally intractable for large M . Due to computation limits, we are forced to reduce $|\mathbb{B}|/M$ for large M , leading to worst performance compared to POMCP, which only considers the current belief encountered in the process [29, 39].

The full decision space $\mathbb{Q}$ also grows exponentially with M . From Table 3.2, we observe that the performance of both PBVI and POMCP deteriorate when $|\hat{\mathbb{Q}}|$ is too small. In our simulations, keeping within our computational limits, we set the values of $|\hat{\mathbb{Q}}|$ and $|\mathbb{B}|$ for different M as shown in Table 3.3.

TABLE 3.2: Average reward for different $|\hat{\mathbb{Q}}|$ with $M = 32$, $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and URT sampling.

$ \hat{\mathbb{Q}} $	2	50	100	300
PBVI	0.122	0.433	0.485	0.494
POMCP	0.102	0.453	0.488	0.502

TABLE 3.3: $|\hat{\mathbb{Q}}|$ and $|\mathbb{B}|$ for different M .

M	8	16	32	64	128
$ \hat{\mathbb{Q}} $	100	200	300	300	300
$ \mathbb{B} $	2000	4000	6000	7000	8000

We now compare the performance of different sampling strategies when used in tandem with PBVI and POMCP. We vary M while setting $\rho = 0.75$, $\underline{b} = 9$ and $\psi = 0.05$.

From Fig. 3.2, we observe that the URT sampling produces a higher reward compared to uniform sampling for both PBVI and POMCP. In particular, URT sampling significantly outperforms uniform sampling when M is large. Therefore, in the remaining simulations, we adopt the URT sampling strategy when using PBVI and POMCP.

FIG. 3.2: Performance comparison between uniform sampling and URT sampling with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M .

3.5.2 Varying M

We next fix $\rho = 0.75$, $\psi = 0.05$, $\underline{b} = 9$, and vary the number of classes M . The results are shown in Fig. 3.3. We observe the following.

- When M is small, PBVI and POMCP with URT sampling achieve better rewards than URSQS. When M is large, the rewards for POMCP and URSQS are comparable, while PBVI performs worse.
- DCFECC is a non-sequential strategy and has the worst reward in most cases, since it does not make use of the answers from earlier workers to help design questions for the later workers.
- We note that PBVI performs the worst when $M = 128$ is large. One reason for this phenomenon is that PBVI is an offline approximate solver for POMDP, suffering from the curse of dimensionality as we have mentioned above. Also, as the full decision space $\mathbb{Q}$ grows exponentially with M , sampling a small subset $\hat{\mathbb{Q}}$ cannot guarantee a good optimal solution. We observe that POMCP is better than PBVI when M is large.

FIG. 3.3: Performance comparison between strategies with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M .

To compare the computation complexity of the different methods under different M values, we perform experiments on a setup with an Intel Xeon CPU E3-1226 v3 and 16GB RAM. The computation time is divided into two parts:

1. Offline computation: In all the methods, we first solve (3.9) to find the optimal (q^*, e^*) and then use Algorithm 3.2 to construct a Ulam-Rényi tree. POMCP then performs URT sampling and stores the sample points. PBVI performs URT sampling and computes the solutions for all the samples, and is thus the most computationally expensive. URSQS does not require any additional offline processing, and thus has the lowest offline computation overhead. The computation times are shown in Table 3.4.
2. Online computation: The average running time for each method over multiple trials and for different M values are shown in Table 3.5. We note that the computation complexity for POMCP is significantly higher than the other methods, although it produces the best average reward. PBVI has the lowest running time but its performance deteriorates for large M . URSQS has comparable performance as POMCP but much lower computation time. Our experiments suggest that PBVI with URT sampling is the recommended approach for small M while URSQS is the recommended approach for large M .

TABLE 3.4: Average offline computation times for different strategies with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M .

	8	16	32	64	128
URSQS (s)	386	651	1295	1492	2405
PBVI (s)	518	1244	2645	3602	7065
POMCP (s)	387	653	1300	1497	2411

TABLE 3.5: Average online computation times for different strategies with $\rho = 0.75$, $\underline{b} = 9$, $\psi = 0.05$ and different M .

	8	16	32	64	128
URSQS (ms)	7.04	8.00	16.34	18.15	19.31
PBVI (ms)	1.81	2.34	2.78	3.95	5.30
POMCP (s)	30	124	259	295	352

3.5.3 Varying $\underline{b}$

FIG. 3.4: Performance comparison between strategies with $M = 32$, $\rho = 0.75$, $\psi = 0.05$ and different $\underline{b}$.

We next fix $M = 32$, $\rho = 0.75$, $\psi = 0.5$ and vary $\underline{b}$, the budget of the crowdsourcer. The results are shown in Fig. 3.4. We observe the following:

- When the budget $\underline{b}$ is relatively small, the rewards for POMCP, PBVI and URSQS increase with increasing budgets, as Fig. 3.4 depicts. This is because with larger budgets, more questions can be assigned to the workers, increasing the probability that the final decision is correct.
- However, the reward is not a strictly increasing function of $\underline{b}$. This is because the increasing cost resulting from more questions undermines the gain in decision accuracy R . When the budget $\underline{b}$ is sufficiently large, as observed from the two cases, $\underline{b} = 13$ and $\underline{b} = 17$, in Fig. 3.4, the reward stays almost

constant, since both POMDP and URSQS tend to stop and make a decision at a time before $\underline{b}$ steps.

- We observe the reward for DCFECC could even decrease with increasing $\underline{b}$. This is because DCFECC is a strategy that will use all the budgets without any optional stopping before the time step $\underline{b}$. Even though decision accuracy R increases with increasing number of questions, it's not optimal to fully use the budgets $\underline{b}$ of the crowdsourcer, i.e. to always stop and make the final decision $\hat{H}$ at time $\underline{b}$.

3.5.4 Varying ψ

FIG. 3.5: Performance comparison between strategies with $M = 32$, $\rho = 0.75$, $\underline{b} = 9$ and different ψ .

We next fix $M = 32$, $\rho = 0.75$, $\underline{b} = 9$ and vary the cost ψ of posing a question. The results are shown in Fig. 3.5. We can see that the reward decreases with increasing ψ . This is expected, since the larger cost will render the average reward lower.

3.5.5 Varying ρ

FIG. 3.6: Performance comparison between strategies with $M = 32$, $\underline{b} = 9$, $\psi = 0.05$ and different ρ .

We vary the parameter ρ in $\underline{\mu}^{[q]}$ with fixed $M = 32$, $\underline{b} = 9$, and $\psi = 0.05$. A larger ρ implies a larger $\underline{\mu}^{[q]}$, i.e., the workers are more reliable. From Fig. 3.6, we observe that the reward increases with increasing ρ , as the workers become more reliable.

FIG. 3.7: Performance comparison between different strategies with $M = 128$, $\rho = 0.75$, $\underline{b} = 9$ and $\psi = 0.05$.

3.5.6 Optimality of URSQS

Finally, to verify the optimality of URSQS, we evaluate URSQS against the following benchmarks:

- $L_{\text{URSQS}}(q) = \max_e L(q, e)$, for $q \in [2, M]$, and where $L(q, e)$ is the objective function of URSQS in (3.9) and the maximization is over all e such that $\hat{B}(q, e) \leq \underline{b} - 1$.
- $U^*(q)$, for each $q \in [2, M]$, is the maximum reward achievable by our proposed Ulam-Rényi strategy in Algorithm 3.1 over all e such that $\hat{B}(q, e) \leq \underline{b} - 1$. This is found through an exhaustive search in each simulation trial, and serves as a benchmark to measure how well our URSQS formulation in (3.9) performs.

FIG. 3.8: Performance comparison between different strategies with $M = 32$, $\rho = 0.75$, $\underline{b} = 13$ and $\psi = 0.05$.

We observe that in most of our experiments, $L_{\text{URSQS}}(q)$ achieves its maximum at the same (q, e) as the maximum of $U^*(q)$, similar to that shown in Fig. 3.7. However, in some cases like Fig. 3.8, the maximum reward for URSQS is different from the optimal reward given by $U^*(q)$, but not significantly so. The average difference between the rewards obtained by URSQS and $U^*(q)$ is 0.017.

3.A Proof of Proposition 3.1

Define

$$f(x, e) = \sum_{l=0}^e \binom{x}{l} (1 - \underline{p})^l \underline{p}^{(x-l)}.$$

We claim that $f(\cdot, e)$ is a decreasing function with a fixed e , which can be proved as follows.

It is sufficient to prove $f(x, e) > f(x + 1, e)$. Since $f(x + 1, e) = \underline{p}f(x, e) + (1 - \underline{p})f(x, e - 1)$, it is sufficient to prove $(1 - \underline{p})f(x, e) > (1 - \underline{p})f(x, e - 1)$, which is correct obviously.

The proof is now complete.

Chapter 4

Sequential Task Recommendation in Crowdsourcing

4.1 Introduction

In a typical crowdsourcing platform, a worker may be recommended a wide variety of tasks to choose from. For example, on Amazon Mechanical Turk (MTurk) [1] and Figure Eight [2], tasks can include labeling the content of an image, determining whether or not sentences extracted from movie reviews are positive, or determining whether or not a website has adult contents. In these examples, the different tasks require different sets of skills from a worker. In the image labeling task, a worker who is good at visual recognition will perform reliably, whereas identifying whether a movie review is positive or not requires a worker with language skills and knowledge of the nuances of that particular language the review is written in. Workers may also have different interests, and may choose not to accept tasks that they are qualified for. Therefore, the crowdsourcing platform needs to find a way to best match the available tasks to the most suitable workers to improve the likelihood of obtaining high-quality solutions to the tasks.

In this chapter, we only consider how to distribute the tasks to workers. Crowdsourcers can of course use inference algorithms introduced in Chapter 3 to come up with a final answer with higher quality from answers collected from many distributed workers. We assume that tasks are organized into several categories. Tasks belonging to the same category need similar domain knowledge to be completed. Within the task assignment context, a number of studies have focused on offline strategies, which assign the tasks to workers without any adaptation or feedback during the workers' tasks completion process. For example, in [8, 10], the authors proposed a task assignment scheme using a bipartite graph to model the affinity of workers for different binary tasks and an iterative algorithm based on belief propagation to infer the final decision from the workers' responses. Reference [81] studied a task allocation model that provides distributional fairness among workers, while maximizing the task allocation ratio. A privacy-preserving task recommendation scheme was proposed in [82], which only considers workers' preferences without taking workers' reliabilities into account. The paper [9] considered heterogeneous tasks with different difficulties using the generalized Dawid-Skene model [66], while [83] utilizes the expectation maximization approach to estimate the tasks' solutions and workers' reliabilities. The aforementioned approaches are all offline, i.e., the task assignments are made without taking into account additional information about the workers' performances and behaviors that can be gleaned while the workers complete tasks sequentially over time.

Online learning for adaptive task assignments has also been considered in recent years. For instance, in [84, 85], as workers may be unreliable, the authors proposed to perform sequential questioning in which the questions posed to the workers are designed based on previous questions and answers. Reference [86] studied sequential user selection using a Bayes update aided online solution in the problem of news disseminating. The paper [36] investigated the problem of label inference for heterogeneous classification tasks by applying online primal-dual techniques. In [87], the authors studied sequential task assignment with budgets that specify

how many times the crowdsourcer would like each task to be completed. Expert-crowdsourcing in which workers are experts who have unknown reliabilities was studied in [88], which also assumed that the crowdsourcer has a budget constraint and each worker has a maximum task limit. The authors proposed an algorithm, which uses an initial exploration phase to uniformly sample the performance of a wide range of workers, and in the latter exploitation phase solves a bounded knapsack problem [89]. In the mobile micro-task allocation problem in spatial crowdsourcing scenarios, where the orders of arriving tasks and workers are dynamic, [90] proposed an online two-phase-based bipartite graph matching algorithm with good average performance. In [91], the authors proposed an adaptive crowdsourcing framework, which continuously estimates the reliability of a worker by evaluating her performance on the completed tasks, and predicts which tasks the worker is well acquainted with. In the above work, the authors assumed that the crowdsourcing platform is able to quickly and accurately evaluate the quality of all the tasks that have been completed. However, as is often the case in practice, the crowdsourcing platform cannot directly and immediately evaluate the quality of a completed task [92]. For example, in tasks where workers are asked to determine whether an image contains a particular object, the crowdsourcing platform has no way to determine if the task completed by a worker is successful or not. In [93], the authors proposed an online algorithm, in which workers' reliabilities are estimated by repetitively assigning each worker the same tasks occasionally.

The above-mentioned works do not take workers' interests or preferences for certain tasks into consideration when performing a task assignment. In crowdsourcing platforms, workers may choose to accept or reject a recommended task. As the number of tasks uploaded by crowdsourcers daily can be large [15], the crowdsourcing platform should aim to recommend tasks to each worker that he or she will likely accept in order to optimize the platform productivity. Including the workers' preferences in task assignment can also help workers to find their preferred tasks faster as they will have fewer recommendations to consider. At the same time, by estimating the

reliability of a worker in each task category helps the crowdsourcer to collect high-quality results quicker [94]. The paper [14] proposed to perform task assignment using workers' preferences based on many-to-one matching. However, the paper assumes that workers' preferences and reliabilities are known *a priori*, which is unrealistic in a practical system. In [15], the authors proposed two recommendation strategies that compute the top- k most suitable tasks for a given worker and the top- k best workers for a given task. However, they assume that the workers can be evaluated immediately. The references [94, 95] employed matrix factorization techniques to capture workers' preferences and reliability where, however, the evaluated scores in the matrix are also assumed to be gotten immediately.

In this chapter, we propose an online learning approach to estimate workers' preferences and reliabilities. We assume that tasks can be categorized, and that each worker can choose to accept or reject a recommended task at each time step. If he or she chooses to accept the task, he or she completes it within the current time step. For each time step, our goal is to recommend a task from a given number of task categories so that it best matches the worker's preferences and reliabilities according to a reward function. As the worker's preferences and reliabilities for different task categories are unknown *a priori*, we adopt a multi-armed bandit (MAB) formulation [42, 43] in which these are learnt through exploration, while the cumulative sum reward is optimized through the exploitation of the empirical best match. However, unlike the classical MAB problem in which the reward at each time step can be observed, the reward is *unobservable* in our formulation as the crowdsourcing platform does not know the true solution of each task. To overcome this, we include *gold* tasks [96] whose solutions are known *a priori*. In [67, 97], gold tasks are used for training and estimating workers' reliabilities. However, gold tasks do not generate any rewards. Including more gold tasks leads to better estimates of the worker's preferences and reliabilities for different task categories, but also results in less reward. In a crowdsourcing platform, a plausible compensation scheme is to reward a worker for completed tasks according to his or her reliability [33–35]. Furthermore, to accurately fuse the results from multiple workers, the

knowledge of the workers' reliabilities are required [36,37]. Therefore, the platform aims to estimate the worker's reliability as accurately as possible. To reflect this, we include both the reward and estimation accuracy in computing the regret. We show that the optimal regret under our formulation has order $O(\sqrt{n})$, where n is the number of time steps. In addition, we propose three strategies that achieve the order optimal regret.

Our MAB formulation is related to, but different from, the risk-averse MAB problem considered in [98–100], which uses mean-variance [101,102] as its risk objective. The differences between our MAB formulation and the risk-averse MAB are: (i) The worker in our problem may choose not to accept a task at each time step, leading to no reward at that time step, whereas in the risk-averse MAB, such an option is not available. (ii) The variance of the reward is used in the formulation of the risk-averse MAB, while we use the variance of the reward *estimator* in formulating our regret. This is because as explained above, in a crowdsourcing platform, the reward at each time step is unobservable, and our regret is dependent on the estimation accuracy from using gold tasks to estimate the expected reward instead of the reward variance.

The rest of this chapter is organized as follows. In Section 4.2, we present our system model and assumptions. In Section 4.3, we derive the optimal order of the regret and introduce three strategies that are order optimal in Section 4.4. In Section 4.5, we present simulations to compare the performance of our approaches.

4.2 System Model

We consider a crowdsourcing platform where each task belongs to one of K categories. The platform recommends a task from some category to a worker at each time $n \in \mathbb{N}$. The worker has different reliabilities and preferences for different categories. The worker's reliability for category x is the probability p_x , $0 < p_x < 1$, of completing a task from category x correctly, and his or her preference refers

to the probability q_x of accepting a task from category x . We let $A_{x,t} = 1$ if the worker accepts the t -th task recommended to him from category x , and $A_{x,t} = 0$ otherwise. We let $Y_{x,t} = 1$ if the worker completes the task correctly, and $Y_{x,t} = 0$ otherwise. For all x and t , we assume that $A_{x,t} \sim \text{Bern}(q_x)$ are independent and identically distributed. Similarly, $Y_{x,t} \sim \text{Bern}(p_x)$ are independent and identically distributed. We assume that $A_{x,t}$ and $Y_{x,t}$ are independent.

We model the task recommendation problem as a MAB problem. The decision set $\mathbb{D} = \{1, \dots, K\}$ contains $|\mathbb{D}| = K$ decisions, each corresponding to one task category in crowdsourcing. However, since $Y_{x,t}$ is typically unobservable since the system does not know the task solution *a priori*, we include the use of gold tasks in our recommendation. A gold task is a task whose solution is known *a priori* to the system. A reward is obtained only if the worker accepts and completes a non-gold task correctly.

Let $C_{x,t} = A_{x,t}Y_{x,t}$, whose distribution is $\text{Bern}(q_x p_x)$. We denote the task category that maximizes $q_x p_x$ as $* = \arg \max_{1 \leq x \leq K} q_x p_x$. For each time step t , let $G_{x,t}$ denote the event that the t -th task from category x recommended to the worker is a gold task. Let

$$g_x(t) = \sum_{i=1}^t A_{x,i} \mathbf{1}_{G_{x,i}} \quad (4.1)$$

be the number of *completed* gold tasks. If the worker completes a non-gold task, he or she is rewarded in proportion to p_x , the probability that he or she has completed the task correctly. However, since we do not know p_x *a priori*, the crowdsourcing platform estimates it using the empirical mean from the $g_x(t)$ completed gold tasks:

$$\bar{p}_x(t) = \frac{1}{g_x(t)} \sum_{i=1}^t A_{x,i} Y_{x,i} \mathbf{1}_{G_{x,i}}. \quad (4.2)$$

Due to the uncertainty in the estimate (4.2), we assume that the reward for the t -th non-gold task from category x if the worker completes it is given by

$$\left(p_x - \beta \frac{\sigma_x^2}{g_x(t)} \right)^+, \quad (4.3)$$

where $\beta > 0$ is a predefined weight that quantifies the importance of the estimator $\bar{p}_x(t)$'s accuracy to the crowdsourcing platform, and $\sigma_x^2 = p_x(1 - p_x)$ is the variance of a task reward from category x . The quantity $\sigma_x^2/g_x(t)$ is the variance of $\bar{p}_x(t)$ conditioned on $g_x(t)$.

For each $x = 1, \dots, K$ and time n , let N_n^x denote the number of tasks recommended so far till time n from category x . We define the cumulative reward function at time n as:

$$\begin{aligned} \mathfrak{R}(n) &= \mathbb{E} \sum_{x=1}^K \sum_{t=1}^{N_n^x} A_{x,t} \left(Y_{x,t} - \beta \frac{\sigma_x^2}{g_x(t)} \right)^+ \mathbf{1}_{G_{x,t}^c} \\ &= \mathbb{E} \sum_{x=1}^K \sum_{t=1}^{N_n^x} q_x \left(p_x - \beta \frac{\sigma_x^2}{g_x(t)} \right)^+ \mathbf{1}_{G_{x,t}^c}, \end{aligned} \quad (4.4)$$

since $A_{x,t}$, $Y_{x,t}$ and $(G_{x,t}, N_n^x)$ are independent. $\mathbf{1}_{G_{x,t}^c}$ guarantees the cumulative reward function $\mathfrak{R}(n)$ does not include the reward if the t -th task from category x recommended to the worker is a gold task. Note that the coefficient β can also be interpreted as the Lagrange multiplier for a constrained optimization problem in which the reward

$$\mathbb{E} \sum_{x=1}^K \sum_{t=1}^{N_n^x} q_x p_x \mathbf{1}_{G_{x,t}^c}$$

is maximized subject to a constraint on the uncertainty in the estimator (4.2).

To avoid dividing by zero, i.e., to ensure $g_x(t) > 0$, we assume that the worker completes one gold task from each category before the recommendation starts, which can be done through a calibration process in a crowdsourcing platform.

To maximize the above reward function (4.4) is equivalent to minimizing the regret function at time n , defined as:

$$R(n) = nq_*p_* - \mathfrak{R}(n). \quad (4.5)$$

Note, with a slight abuse of notation, the regret $R(n)$ defined here is different from that defined in (2.11), however in this chapter, we keep using this notation. For simplicity, we assume that every task category is non-empty at each time step. This is a reasonable assumption since the task pool in a crowdsourcing platform is updated constantly, and a single task may be assigned to more than one worker. In the following, we use the terms “task category” and “arm” interchangeably in our MAB formulation.

4.3 Optimal Regret Order

In this section, we first show that the optimal order of the regret function (4.5) is $O(\sqrt{n})$, where n is the number of time steps. We then propose three recommendation strategies, all of which achieve the order optimal regret.

Theorem 4.1. *The optimal order of the regret function (4.5) is $O(\sqrt{n})$, where n is the number of time steps.*

Proof. See Section 4.A. □

4.4 Order Optimal Strategies

In this section, we propose three strategies that achieve the optimal order regret of $O(\sqrt{n})$ in Theorem 4.1, and discuss the advantages of each.

4.4.1 Greedy Recommendation Strategy

In our Greedy Recommendation Strategy (GR), we divide time into epochs. In each epoch m , a single task category or arm c_m is chosen and all tasks in that epoch are drawn from that category. In the first K epochs, each of which consists of a single time step, the worker completes a gold task from each of the K categories. Subsequently, the m -th epoch, where $m \geq K + 1$, consists of $\kappa(m) - \kappa(m - 1) + 1$ time steps, where $\kappa(m) = \lceil \alpha m^2 \rceil$, and $\alpha > 0$. In each of these epochs, we set the first task to be a gold task, while for all other time steps in the epoch, the tasks recommended are non-gold tasks chosen from a particular task category. For each $m \geq K + 1$, the task category c_m is chosen based on a ϵ_m -greedy policy [43] as follows. In the epoch based policies, let $n_x(m)$ be the number of category x gold tasks *completed* in the first m epochs.

Let $\Delta_x = q_* p_* - q_x p_x$, and choose $\underline{d}$ so that $0 < \underline{d} \leq \min_{x \neq *} \Delta_x$. For each $m = 1, 2, \dots$, let $\epsilon_m = \min\{1, cK/(\underline{d}^2 m)\}$, where $c > 0$ is a fixed constant. At the beginning of each epoch m , where $m \geq K + 1$, we choose

$$z_m = \arg \max_x \bar{C}_x(T_x(m - 1)), \quad (4.6)$$

where $T_x(m)$ is the number of epochs within the first m epochs that category x is chosen,

$$\bar{C}_x(g) = \frac{1}{g} \sum_{i=1}^g A_{x,j_i} Y_{x,j_i}, \quad (4.7)$$

and $j_1, \dots, j_g$ are the indices of the first g gold tasks recommended from category x . Then, with probability $1 - \epsilon_m$, we let $c_m = z_m$ to be the task category chosen for epoch m , and with probability ϵ_m , we let c_m to be a randomly chosen category. The GR strategy is summarized in Algorithm 4.1.

Theorem 4.2. *Suppose that $c > \max\{5\underline{d}^2, 2\}$. Then, GR has regret of order $O(\sqrt{n})$, where n is the number of time steps.*

Algorithm 4.1 GR**Require:** $0 < \underline{d} \leq \min_{x \neq *} \Delta_x$, $c > 0$, $\alpha > 0$.**Ensure:**

- 1: Recommend one gold task from each category.
- 2: Set $m = K + 1$, $T_x(m) = 1$ and $\bar{C}_x(1) = C_{x,1}$ for all $x = 1, \dots, K$.
- 3: **loop**
- 4: Find z_m in (4.6).
- 5: Set $\epsilon_m = \min\{1, \frac{cK}{d^2 m}\}$. With probability $1 - \epsilon_m$, choose $c_m = z_m$, otherwise set c_m to be a randomly chosen arm.
- 6: Recommend one gold task from arm c_m at the first time step of epoch m . Update $T_{c_m}(m) = T_{c_m}(m - 1) + 1$, $n_{c_m}(m)$ and $\bar{C}_{c_m}(T_{c_m}(m))$.
- 7: Recommend a non-gold task drawn from arm c_m at each of the following $\kappa(m) - \kappa(m - 1)$ time steps.
- 8: Set $m = m + 1$.
- 9: **end loop**

Proof. See Section 4.B. □**4.4.2 Uniform-Pulling Recommendation Strategy****Algorithm 4.2** UR**Require:** $\alpha > 0$.**Ensure:** Recommend one gold task from each arm. Set $m = 2$.

- 1: **loop**
- 2: At the first K time steps of the m -th epoch, recommend K gold tasks, one from each arm.
- 3: Choose arm $c_m = \arg \max_x \bar{C}_x(m)$.
- 4: Recommend a non-gold task drawn from arm c_m at each of the following $\kappa(m) - \kappa(m - 1)$ time steps.
- 5: Set $m = m + 1$.
- 6: **end loop**

In this subsection, we propose another strategy, which we call the Uniform-Pulling Recommendation Strategy (UR). We again divide time into epochs, with the m -th epoch containing $K + \kappa(m) - \kappa(m - 1)$ time steps. In the first K time steps of the m -th epoch, where $m > 1$, we first recommend K gold tasks, one from each of the K task categories. Next, we choose arm $c_m = \arg \max_{1 \leq x \leq K} \bar{C}_x(T_x(m))$, where now $T_x(m) = m$ for all $x \in [1, K]$. In the subsequent $\kappa(m) - \kappa(m - 1)$ time steps,

we recommend a non-gold task from arm c_m at each time step. The UR strategy is summarized in Algorithm 4.2.

Theorem 4.3. *UR has regret of order $O(\sqrt{n})$, where n is the number of time steps.*

Proof. See Section 4.C. □

4.4.3 ϵ -First Recommendation Strategy

Algorithm 4.3 ϵ -First

Require: $\underline{H} = \lfloor \sqrt{n} \rfloor$.

- 1: Recommend $\underline{H}$ gold tasks from each of the K arms.
 - 2: Choose arm $c_{\underline{H}} = \arg \max_x \bar{C}_x(\underline{H})$.
 - 3: Recommend a non-gold task drawn from arm $c_{\underline{H}}$ at each of the following $n - K\underline{H}$ time steps.
-

The ϵ -first strategy was designed to deal with the budget-limited MAB problem [103], where the budget refers to the total number of time steps. In the ϵ -first strategy, the first ϵ fraction of the budget is used to explore the reward distributions of all arms, while the remaining $1 - \epsilon$ fraction is spent on the empirically best arm found in the initial exploration phase. To apply the ϵ -first strategy to our task recommendation problem, we let $\epsilon = \Theta(1/\sqrt{n})$. Uniform pulling of arms is used in the exploration phase. The ϵ -first strategy is summarized in Algorithm 4.3, where $\epsilon = K\underline{H}/n$.

Theorem 4.4. *If $\epsilon = \Theta(1/\sqrt{n})$, then the ϵ -first strategy has regret of order $O(\sqrt{n})$, where n is the number of time steps.*

Proof. See Section 4.D. □

4.4.4 Discussions

The ϵ -first strategy assumes that the total number of tasks n to be recommended to each worker is known beforehand, while the other two strategies do not need

such an assumption. If both UR and the ϵ -first strategy recommend the same total number of gold tasks, then the ϵ -first strategy achieves a smaller regret than UR. This is because the ϵ -first strategy recommends all the gold questions in the initial exploration phase, leading to a better estimation of the reward distributions before any non-gold tasks are recommended. However, in a practical crowdsourcing platform, knowing the total number of tasks for a worker may not be feasible as a worker is not guaranteed to be active or to remain in the system. An alternative is to use a hybrid UR and ϵ -first strategy, where the initial exploration phase in each epoch of the UR strategy is set to be an ϵ fraction of the total number of time steps in that epoch.

On the other hand, UR can become more costly than GR when the number of arms K is large. If UR and GR both recommend the same total number of gold tasks to the worker, GR tends to recommend more gold tasks from the best task category in the long run, leading to a smaller estimation variance of (4.2) for the best category and a smaller asymptotic regret than UR.

In both the GR and UR strategies, we choose $\kappa(m) = \Theta(m^2)$, which determines how frequently gold tasks are recommended to the worker. A more general choice is $\kappa(m) = \Theta(m^\gamma)$ for $\gamma \geq 1$. We now show that $\gamma = 2$ is essentially the only choice that gives optimal order regret in GR and UR. Let $f(n)$ be the number of gold tasks recommended till time n . Then, since GR and UR recommends a fixed number of gold tasks in each epoch, we have $f(n) = \Theta(m)$ if n is within epoch m . The number of time steps up to and including epoch m is $\Theta(\kappa(m))$, which implies that $f(n) = \Theta(n^{1/\gamma})$. Then from (4.10), we have

$$\begin{aligned} R(n) &\geq \min \left\{ \frac{an}{b_1 n^{1/\gamma}} + q_* p_* b_2 n^{1/\gamma} - a, n q_* p_* \right\} \\ &= \min \left\{ \frac{a}{b_1} n^{1-1/\gamma} + q_* p_* b_2 n^{1/\gamma} - a, n q_* p_* \right\}, \end{aligned}$$

for some positive constants a , b_1 , and b_2 . Therefore, if $\gamma \neq 2$, $R(n)/\sqrt{n} \rightarrow \infty$ as $n \rightarrow \infty$.

4.5 Simulations and Performance Evaluation

In this section, we perform simulations¹ and compare the performance of GR, UR, and the ϵ -first strategy. We also compare with the UR strategy that uses $\kappa(m) = \lceil \alpha m^\gamma \rceil$, for $\gamma > 0$, which we call the UR(γ) strategy. The UR(2) strategy is equivalent to the UR strategy. We perform simulations using different settings shown in Table 4.1.

TABLE 4.1: (p_x, q_x) for each task category. In setting no. 2, $(\underline{w}, \underline{y})$ is varied over $[0, 1]^2$.

Setting no.	Task category x			
	1	2	3	4 – 10
1	(0.7, 0.7)	(0.9, 0.3)	(0.3, 0.9)	(0.4, 0.4)
2	(0.7, 0.7)	$(\underline{w}, \underline{y})$	(0.4, 0.4)	(0.4, 0.4)

Setting no.	Task category x			
	1	2 – 10	11 – 15	16 – 25
3	(0.8, 0.8)	(0.4, 0.4)		
4	(0.8, 0.8)	(0.4, 0.4)	(0.4, 0.4)	
5	(0.8, 0.8)	(0.4, 0.4)	(0.4, 0.4)	(0.4, 0.4)

We track the regret as our performance measure. In each simulation, we perform 2000 trials, each with 10^3 time steps. To compare the performance of different approaches, we compute the empirical average regret, over all the trials. For convenience, this average is referred to as the *average regret*.

In Theorem 4.2, we show that for GR, a sufficient condition, under which the order optimal regret is achieved, is $c > \max\{5\underline{d}^2, 2\}$. However, in simulations we find that a smaller c can lead to better performance for all settings listed in Table 4.1. Therefore, in the simulations, we set $c = 0.05$ in GR. In practice, the parameter $\underline{d}$ in GR can be set to be a fixed small value like 0.1. We set the parameter $\underline{d}$ in GR to be 0.1. In all simulations except those in Section 4.5.3, we set $\alpha = 0.1$ for GR, UR and UR(γ) for all γ . We use $\beta = 10$ in all simulations.

¹Simulation codes can be found in <https://github.com/KANGQIYU/Sequential-Task-Recommendation-in-Crowdsourcing>

FIG. 4.1: Performance comparison between different strategies using setting no. 1.

4.5.1 Comparison of Different Strategies

We first compare the performance of different strategies. In Figs. 4.1 and 4.2, we use setting no. 1 in Table 4.1 for our simulations. We observe the following.

- GR, UR, and the ϵ -first strategy outperform UR(1.5) and UR(10) when n is sufficiently large. In UR(1.5), gold tasks are recommended more frequently than GR and UR. Therefore, less non-gold tasks are recommended to the worker. On the other hand, UR(10) corresponds to the case where gold tasks are recommended infrequently compared to GR and UR. In Fig. 4.2, we see that $\gamma = 2$ achieves the optimal regret order, in line with our observations in Section 4.4.4.
- The ϵ -first strategy performs best among all the strategies, but it requires to know the total number of time steps n beforehand.

FIG. 4.2: Average regret order of $UR(\gamma)$ using setting no. 1.

- GR outperforms UR. One reason is the exploration of all arms at every epoch in UR reduces the cumulative reward with fewer non-gold tasks assigned. Another reason is even if GR and UR have recommended the same total number of gold tasks to the worker, the number of gold tasks recommended from the best category using UR is less than that using GR. Therefore, the variance of the empirical mean (4.2) for the best category using UR is larger than that using GR, leading to a larger average regret.

4.5.2 Varying Total Number of Task Categories K

We next show the performance of UR, GR and the ϵ -first strategy for different number of arms K , using the settings no. 3 – 5 listed in Table 4.1. From Fig. 4.3, we observe the following.

- GR outperforms UR in all cases, the explanations are given in Section 4.5.1.
- The average regret with more task categories is larger than that with fewer task categories, which is expected.
- GR outperforms the ϵ -first strategy in this section, while it performs worse than the ϵ -first strategy in Section 4.5.1. One reason for this phenomenon is that the ϵ -first strategy has $\underline{H} = \lfloor \sqrt{n} \rfloor$, which restricts the number of gold questions assigned and may not always be the optimal option for different settings. It may very well happen that $\underline{H} = \lfloor \sqrt{n} \rfloor$ is a superior option for setting no. 1, but an inferior option for setting no. 3-5.

FIG. 4.3: Performance comparison between UR and GR with $K = 10, 15$ and 25 using setting no. 3 – 5 in Table 4.1, respectively.

4.5.3 Varying α

We next vary the parameter α in GR and UR separately using setting no. 1 and 3. The results are shown in Figs. 4.4 and 4.5, respectively. We observe the following:

- We note that a moderate α is optimal for both UR and GR. When α is too large, the strategies recommend gold tasks too infrequently, leading to a larger estimation variance. On the other hand, a small α results in too many gold tasks.
- The optimal α for GR is smaller than that for UR. This is because if α is set to be the same for UR and GR, UR tends to recommend more gold tasks than GR, which makes UR more costly as we have discussed in Section 4.4.4. Also, we know the system will assign fewer gold tasks with larger α .
- The value $\min_{x \neq *} \Delta_x$ in setting no. 3 is larger than that in setting no. 1. The optimal α in both GR and UR tends to decrease with decreasing $\min_{x \neq *} \Delta_x$. This is because when $\min_{x \neq *} \Delta_x$ decreases, increasing the number of gold questions allows for greater exploration of the task categories.

4.5.4 Varying $\min_{x \neq *} \Delta_x$

When $\min_{x \neq *} \Delta_x$ is small, distinguishing the best task category $*$ from the other categories is difficult. We vary the values $(\underline{w}, \underline{y})$ in setting no. 2 in Table 4.1 to obtain different values of $\min_{x \neq *} \Delta_x = q_* p_* - \underline{w} \underline{y} = 0.49 - \underline{w} \underline{y}$. The results are shown in Fig. 4.6, where we plot the average regret at the time step $n = 10^3$. We observe that a relatively smaller $\min_{x \neq *} \Delta_x$ results in a larger average regret for GR. This is because a smaller $\min_{x \neq *} \Delta_x$ makes it more difficult to distinguish between the best arm and the others. We also observe that the average regrets for UR and the ϵ -first strategy remain almost constant with varying $\min_{x \neq *} \Delta_x$ because these strategies use a sufficiently large number of gold tasks to estimate the parameters of each arm.

FIG. 4.4: Performance of GR with different values of parameter α and different settings.

4.5.5 Learning Only Partial Information

We next consider a crowdsourcing platform that learns only either a worker's preferences or reliabilities but not both, when performing task recommendations. The simulation results are shown in Fig. 4.7 using setting no. 1 in Table 4.1. We see that learning either only the preferences or reliabilities results in a lower cumulative reward, which is expected. The loss in cumulative reward for GR, UR or the ϵ -first strategy learning either only the preferences or reliabilities widens compared to GR, UR or the ϵ -first strategy respectively when the number of time steps n becomes large.

FIG. 4.5: Performance of UR with different values of parameter α and different settings.

4.A Proof of Theorem 4.1

Let

$$f(n) = \sum_{x=1}^K \sum_{t=1}^{N_n^x} \mathbf{1}_{G_{x,t}} \quad (4.8)$$

be the total number of gold tasks recommended till time n . Then, from (4.4), we have

$$\begin{aligned} \mathfrak{R}(n) &= \mathbb{E} \sum_{x=1}^K \sum_{t=1}^{N_n^x} q_x \left(p_x - \beta \frac{\sigma_x^2}{g_x(t)} \right)^+ \mathbf{1}_{G_{x,t}^c} \\ &\leq \mathbb{E} \sum_{x=1}^K \sum_{t=1}^{N_n^x} \left(q_* p_* - \beta \frac{q_x \sigma_x^2}{f(n)} \right)^+ \mathbf{1}_{G_{x,t}^c} \\ &\leq \mathbb{E} \left[(n - f(n)) \left(q_* p_* - \beta \frac{\min_x q_x \sigma_x^2}{f(n)} \right) \right]^+. \end{aligned} \quad (4.9)$$

FIG. 4.6: Performance of different strategies with varying $\min_{k \neq *} \Delta_k$ using setting no. 2 for different values of $(\underline{w}, \underline{y})$.

From (4.5) and (4.9), with $k = \beta \min_x q_x \sigma_x^2$, when n is sufficiently large, we then have

$$\begin{aligned}
 R(n) &\geq nq_*p_* - \mathbb{E} \left[(n - f(n)) \left(q_*p_* - \beta \frac{\min_x q_x \sigma_x^2}{f(n)} \right) \right]^+ \\
 &\geq \mathbb{E} \left[\min \left\{ k \frac{n}{f(n)} + q_*p_*f(n) - a, nq_*p_* \right\} \right] \tag{4.10}
 \end{aligned}$$

$$\geq \min \left\{ 2\sqrt{kq_*p_*n} - k, nq_*p_* \right\} \tag{4.11}$$

$$= 2\sqrt{kq_*p_*n} - k, \tag{4.12}$$

where the inequality in (4.11) follows because $k \frac{n}{f(n)} + q_*p_*f(n) - k \geq 2\sqrt{kq_*p_*n} - k$ with probability 1. The theorem is now proved.

FIG. 4.7: Performance of UR, GR and the ϵ -first strategy considering either preferences or reliabilities under setting no. 1.

4.B Proof of Theorem 4.2

For each arm x in epoch m , we denote by $P_{x,m} = \mathbb{P}(c_m = x)$, the probability that arm x is chosen. From Theorem 3 in [43], if $c > \max\{5\underline{d}^2, 2\}$ and $x \neq *$, we have

$$P_{x,m} \leq \frac{c}{\underline{d}^2 m} + o(m^{-1}), \quad (4.13)$$

so that

$$P_{*,m} \geq 1 - \frac{c(K-1)}{\underline{d}^2 m} - o(m^{-1}). \quad (4.14)$$

Let v be the number of epochs completed till the time step n , i.e., v is the largest integer satisfying $K + \sum_{m=K+1}^v (\kappa(m) - \kappa(m-1) + 1) = \lceil \alpha v^2 \rceil - \lceil \alpha K^2 \rceil + v \leq n$. We have $v = O(\sqrt{n})$. Recall that $n_x(m)$ be the number of category x gold tasks

completed in the first m epochs. Since a single gold question is recommended in each epoch, we have $g_{c_m}(t) = n_{c_m}(m)$ for all time steps t in epoch m . From (4.4), we obtain

$$\begin{aligned}
\mathfrak{R}(n) &\geq \mathbb{E} \sum_{m=K+1}^v q_{c_m} (\kappa(m) - \kappa(m-1)) \left(p_{c_m} - \frac{\beta \sigma_{c_m}^2}{n_{c_m}(m)} \right)^+ \\
&\geq \mathbb{E} \sum_{m=K+1}^v q_*(2\alpha m - \alpha - 1)^+ \left(p_* - \frac{\beta \sigma_*^2}{n_*(m)} \right) \mathbf{1}_{\{c_m=*\}} \\
&= \sum_{m=K+1}^v q_*(2\alpha m - \alpha - 1)^+ \left(p_* P_{*,m} - \mathbb{E} \left[\frac{\beta \sigma_*^2}{n_*(m)} \right] \right). \tag{4.15}
\end{aligned}$$

Recall also that $T_x(m) = \sum_{i=1}^m \mathbf{1}_{\{c_i=x\}}$ is the number of epochs within the first m epochs in which arm x was chosen. Since GR assumes that the worker completes a gold task from arm $*$ before the recommendation starts, we have

$$n_*(m) - 1 \mid T_*(m) \sim \text{Bern}(T_*(m) - 1, q_*),$$

and

$$\begin{aligned}
&\mathbb{E} \left[\frac{1}{n_*(m)} \right] \\
&= \mathbb{E} \left[\mathbb{E} \left[\frac{1}{n_*(m)} \mid T_*(m) \right] \right] \\
&= \mathbb{E} \sum_{j=0}^{T_*(m)-1} \frac{1}{1+j} \binom{T_*(m)-1}{j} q_*^j (1-q_*)^{T_*(m)-1-j} \\
&= \mathbb{E} \sum_{j=0}^{T_*(m)-1} \frac{1}{T_*(m)} \binom{T_*(m)}{j+1} q_*^j (1-q_*)^{T_*(m)-1-j} \\
&= \mathbb{E} \frac{1}{T_*(m)q_*} \sum_{j=0}^{T_*(m)-1} \binom{T_*(m)}{j+1} q_*^{j+1} (1-q_*)^{T_*(m)-1-j} \\
&= \mathbb{E} \frac{1}{T_*(m)q_*} (1 - (1-q_*)^{T_*(m)}) \\
&\leq \mathbb{E} \left[\frac{1}{T_*(m)q_*} \right]. \tag{4.16}
\end{aligned}$$

From (4.14), (4.15) and (4.16), we obtain

$$\begin{aligned}
\mathfrak{R}(n) &\geq \sum_{m=K+1}^v q_* p_* (2\alpha m - \alpha - 1)^+ (1 - O(m^{-1})) \\
&\quad - \sum_{m=K+1}^v (2\alpha m - \alpha - 1)^+ \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(m)} \right] \\
&\geq q_* p_* \alpha v^2 - O(v) - \sum_{m=K+1}^v (2\alpha m - \alpha - 1)^+ \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(m)} \right] \\
&\geq n q_* p_* - O(\sqrt{n}) - \sum_{m=K+1}^v (2\alpha m - \alpha - 1)^+ \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(m)} \right]. \tag{4.17}
\end{aligned}$$

We next prove the following lemma.

Lemma 4.1. $\mathbb{E} \left[\frac{1}{T_*(m)} \right] \leq O(m^{-1})$.

We have

$$\begin{aligned}
&\mathbb{E} \left[\frac{1}{T_*(m)} \right] \\
&= \mathbb{E} \left[\frac{1}{T_*(m)} \middle| T_*(m) \leq \frac{m}{K} \right] \mathbb{P} \left(T_*(m) \leq \frac{m}{K} \right) \\
&\quad + \mathbb{E} \left[\frac{1}{T_*(m)} \middle| T_*(m) > \frac{m}{K} \right] \mathbb{P} \left(T_*(m) > \frac{m}{K} \right) \\
&\leq \mathbb{P} \left(T_*(m) \leq \frac{m}{K} \right) + O(m^{-1}). \tag{4.18}
\end{aligned}$$

Therefore, to show the lemma, it suffices to show that the tail probability $\mathbb{P} \left(T_*(m) \leq \frac{m}{K} \right)$ is of order $O(m^{-1})$.

Consider $x \neq *$. Let $T_x^B(m)$ be the number of times $z_j = x$ for $j \leq m$, and $T_x^R(m)$ be the number of times arm x was randomly chosen for $j \leq m$. From the union bound, we have

$$\mathbb{P} \left(T_x(m) \geq \frac{m}{K} \right) \leq \mathbb{P} \left(T_x^R(m) \geq \frac{m}{2K} \right) + \mathbb{P} \left(T_x^B(m) \geq \frac{m}{2K} \right). \tag{4.19}$$

From Hoeffding's inequality, we obtain

$$\begin{aligned} \mathbb{P}\left(T_x^R(m) \geq \frac{m}{2K}\right) &\leq e^{-2(\frac{1}{2K} - \mathbb{E}T_x^R(m)/m)^2 m} \\ &\leq e^{-2(\frac{1}{2K} - O(\frac{c \ln m}{d^2 m}))^2 m} \\ &= e^{-O(m)}, \end{aligned} \tag{4.20}$$

where the second inequality follows because the probability of randomly choosing arm x at each epoch j is not more than $cK/(\underline{d}^2 j) \cdot 1/K = c/(\underline{d}^2 j)$ and

$$\mathbb{E}T_x^R(m) \leq \sum_{j=1}^m \frac{c}{\underline{d}^2 j} \leq \frac{c(\ln m + 1)}{\underline{d}^2}.$$

In order to bound the second term in (4.19), we use a similar argument as that in [47]. Let $\underline{u}(m) = \frac{c}{\underline{d}^2} \ln \frac{rd^2 e^{1/2}}{2cK^2}$ and $\underline{s}(m) = \lceil m/(2K) \rceil$. Define the following events:

$$\begin{aligned} E_1 &= \left\{ \bar{C}_x(j) \leq q_x p_x + \frac{\Delta_x}{2}, \text{ for all } j \in [\underline{s}(m), m] \right\}, \\ E_2 &= \left\{ \bar{C}_*(j) \geq q_* p_* - \frac{\Delta_x}{2}, \text{ for all } j \in (\underline{u}(m), m] \right\}, \\ E_3 &= \{T_*(\underline{s}(m)) > \underline{u}(m)\}. \end{aligned}$$

Under the event $E_1 \cap E_2 \cap E_3$, we have for all $i \in [\underline{s}(m), m]$,

$$m \geq T_*(i) \geq T_*(\underline{s}(m)) > \underline{u}(m),$$

which implies that

$$\begin{aligned} \bar{C}_*(T_*(i)) &\geq q_* p_* - \frac{\Delta_x}{2} \\ &> q_x p_x + \frac{\Delta_x}{2} \\ &\geq \bar{C}_x(j), \end{aligned} \tag{4.21}$$

for all $j \in [\underline{s}(m), m]$. Since $T_x^B(\underline{s}(m)) \leq \underline{s}(m)$, we have $T_x^B(m) \leq \underline{s}(m)$ because otherwise there exists m' such $T_x^B(m') \in [\underline{s}(m), m]$ and $\bar{C}_x(T_x(m')) > \bar{C}_*(T_*(m'))$,

which contradicts (4.21). Therefore, $E_1 \cap E_2 \cap E_3 \subset \{T_x^B(m) \leq \underline{s}(m)\}$, and

$$\begin{aligned} & \mathbb{P}(T_x^B(m) > \underline{s}(m)) \\ & \leq \mathbb{P}(E_1^c \cup E_2^c \cup E_3^c) \\ & \leq \mathbb{P}(E_1^c) + \mathbb{P}(E_2^c) + \mathbb{P}(E_3^c), \end{aligned} \quad (4.22)$$

where the last inequality is the union bound. We next bound each of the terms on the right hand side of (4.22) separately. For the first term, we have

$$\begin{aligned} \mathbb{P}(E_1^c) & \leq \sum_{j=\underline{s}(m)}^m \mathbb{P}\left(\bar{C}_x(j) > q_x p_x + \frac{\Delta_x}{2}\right) \\ & \leq \sum_{j=\underline{s}(m)}^m e^{-\frac{\Delta_x^2}{2}j} \\ & \leq \frac{e^{-\frac{\Delta_x^2 m}{4K}}}{1 - e^{-\frac{\Delta_x^2}{2}}} \\ & = o(m^{-1}), \end{aligned} \quad (4.23)$$

where the second inequality follows from Hoeffding's inequality. Similarly, for the second term on the right hand side of (4.22), we obtain

$$\begin{aligned} \mathbb{P}(E_2^c) & \leq \sum_{j=\underline{u}(m)}^m \mathbb{P}\left(\bar{C}_*(j) < q_* p_* - \frac{\Delta_x}{2}\right) \\ & \leq \sum_{j=\underline{u}(m)}^m e^{-\frac{\Delta_x^2}{2}j} \\ & = O(e^{-\frac{\Delta_x^2 c}{2d^2} \ln m}) \\ & = o(m^{-1}), \end{aligned} \quad (4.24)$$

since $c > 2$ and $0 < \underline{d} \leq \min_{x \neq *} \Delta_x$.

From the proof of Theorem 3 in [43], we have $\underline{u}(m) \leq \frac{1}{2K} \sum_{i=1}^{\underline{s}(m)} \epsilon_i$ and

$$\mathbb{P}(E_3^c) \leq \mathbb{P}(T_*^R(\underline{s}(m)) \leq \underline{u}(m))$$

$$\begin{aligned}
&\leq \mathbb{P} \left(T_*^R(\underline{s}(m)) \leq \frac{1}{2K} \sum_{i=1}^{\underline{s}(m)} \epsilon_i \right) \\
&\leq e^{-\frac{1}{5} \frac{1}{2K} \sum_{i=1}^{\underline{s}(m)} \epsilon_i} \\
&\leq e^{-\underline{u}(m)/5} \\
&= \left(\frac{2cK^2}{m\underline{d}^2 e^{1/2}} \right)^{\frac{c}{5\underline{d}^2}} \\
&= o(m^{-1}), \tag{4.25}
\end{aligned}$$

where the third inequality follows from Bernstein's inequality (see (13) of [43] for a detailed proof).

Applying (4.23) to (4.25) on the right hand side of (4.22), we obtain $\mathbb{P}(T_x^B(m) > \underline{s}(m)) = o(m^{-1})$. From (4.19) and (4.20), we then have

$$\mathbb{P} \left(T_x(m) \geq \frac{m}{K} \right) = o(m^{-1}),$$

which yields

$$\begin{aligned}
\mathbb{P} \left(T_*(m) \leq \frac{m}{K} \right) &= \mathbb{P} \left(\sum_{x \neq *} T_x(m) \geq \frac{m(K-1)}{K} \right) \\
&\leq \sum_{x \neq *} \mathbb{P} \left(T_x(m) \geq \frac{m}{K} \right) \\
&= o(m^{-1}), \tag{4.26}
\end{aligned}$$

and Lemma 4.1 is proved.

From (4.17) and Lemma 4.1, we obtain

$$\begin{aligned}
\mathfrak{R}(n) &\geq nq_*p_* - O(\sqrt{n}) - \sum_{m=K+1}^v (2\alpha m - \alpha - 1)^+ O(m^{-1}) \\
&\geq np_*q_* - O(\sqrt{n}) - O(v) \\
&\geq np_*q_* - O(\sqrt{n}),
\end{aligned}$$

since $v = O(\sqrt{n})$. The proof of Theorem 4.2 is now complete.

4.C Proof of Theorem 4.3

The proof uses a similar argument and the same notations as that in the proof of Theorem 4.2. For $m > 1$, we have

$$\begin{aligned}
 P_{*,m} &\geq 1 - \sum_{x \neq *} \mathbb{P}(\bar{C}_x(m) \geq \bar{C}_*(m)) \\
 &= 1 - \sum_{x \neq *} \mathbb{P}(\bar{C}_x(m) - \bar{C}_*(m) + \Delta_x \geq \Delta_x) \\
 &\geq 1 - \sum_{x \neq *} e^{-2\Delta_x^2 m} \\
 &\geq 1 - o(m^{-1}),
 \end{aligned} \tag{4.27}$$

where the penultimate inequality follows from Hoeffding's inequality.

We choose v be the largest integer satisfying $K + \sum_{m=2}^v (\kappa(m) - \kappa(m-1) + K) = \lceil \alpha v^2 \rceil - \lceil \alpha^2 \rceil + Kv \leq n$ so that $v = O(\sqrt{n})$, we have from (4.15) and (4.16),

$$\begin{aligned}
 \mathfrak{R}(n) &\geq \sum_{m=2}^v q_*(2\alpha m - \alpha - 1)^+ \left(p_* P_{*,m} - \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(m)} \right] \right) \\
 &\geq \sum_{m=2}^v q_* p_* (2\alpha m - \alpha - 1)^+ (1 - o(m^{-1})) \\
 &\quad - \sum_{m=2}^v (2\alpha m - \alpha - 1)^+ \frac{\beta \sigma_*^2}{m} \\
 &\geq q_* p_* \alpha v^2 - O(v) - \sum_{m=2}^v (2\alpha m - \alpha - 1)^+ \frac{\beta \sigma_*^2}{m} \\
 &\geq n q_* p_* - O(\sqrt{n}).
 \end{aligned}$$

where the second inequality follows from (4.27). Therefore, $R(n) \leq O(\sqrt{n})$, and the proof of Theorem 4.3 is complete.

4.D Proof of Theorem 4.4

Let $P_{x,\underline{H}} = \mathbb{P}(c_{\underline{H}} = x)$. Similar to (4.27) in the proof of Theorem 4.3, it can be shown that $P_{*,\underline{H}} \geq 1 - o(\underline{H}^{-1})$. From (4.4) and (4.16), we obtain

$$\begin{aligned}
& \mathfrak{R}(n) \\
& \geq \mathbb{E} \left[q_{c_{\underline{H}}} (n - K\underline{H}) \left(p_{c_{\underline{H}}} - \frac{\beta \sigma_{c_{\underline{H}}}^2}{\underline{H} q_{c_{\underline{H}}}} \right)^+ \right] \\
& \geq \mathbb{E} \left[q_* (n - K\underline{H}) \left(p_{c_*} - \frac{\beta \sigma_*^2}{\underline{H} q_*} \right) \mathbf{1}_{\{c_{\underline{H}} = *\}} \right] \\
& \geq q_* (n - K\underline{H}) \left(p_* P_{*,\underline{H}} - \frac{\beta \sigma_*^2}{\underline{H} q_*} \right) \\
& \geq n q_* p_* - O(\sqrt{n}),
\end{aligned} \tag{4.28}$$

since $\underline{H} = O(\sqrt{n})$. Therefore, $R(n) \leq O(\sqrt{n})$, and the proof is complete.

Chapter 5

Sequential Non-Discriminatory Recommendations

5.1 Introduction

Recommendation systems play an essential role and have become indispensable in modern online platforms with a large number of products and users. The traditional recommendation system in a platform typically aims to recommend products (items) that match users' interests most since users may migrate to an alternative platform if they are not interested in the items provided by the current platform's recommendation system. Collaborative filtering, content-based filtering, sequential bandits, and the hybrid approaches are commonly adapted in recommendation systems to analyze users' preferences and provide personalized recommendations that match users' tastes most appropriately. Classical content-based filtering provides the user with new items which are similar, in terms of the item description profiles, to those consumed already by the user [16]. Collaborative filtering tries to analyze the similarity relationships between users and their previously consumed items by mining the users' historical data without the description profiles [17]. The hybrid of these two filtering approaches are often utilized jointly to improve the

recommendation performance [104] further since each of them has its own shortcomings, like the cold-start issue in collaborative filtering [18], and jointly utilization can provide important complements to each other. Classical collaborative and content-based filtering are both offline approaches, while sequential bandits [18] are online approaches and provide solutions to the optimal exploration and exploitation trade-off in sequential recommendation under uncertainty. For example, in news recommendation applications, the news articles can be treated as arms in sequential bandits, while the user’s clicks are the rewards. The sequential bandits recommendation approach tries to balance the maximization of user satisfaction in the long run (exploitation) and the information gathering about the matching quality (exploration). It has been shown in the literature that a linear bandit model is sufficient for real applications like the news recommendation [18, 104].

A plausible recommendation scheme is to get as many clicks/views as possible. However, recently it has been reported that in many cases, a recommendation procedure is not “fair” from users’ perspectives. For example, in [19], the author analyzed the ads provided by Google AdSense and concluded that the platform tends to recommend more ads suggestive of arrests if some typical black names are typed in. Reference [20] reported that such platformed racism also exists in Facebook and YouTube whose recommendation algorithms tend to generate contents implying an Australian race-based controversy. Another example is that Netflix last year angered black subscribers with targeted posters containing black actors no matter how minor their roles in the film are [21]. All such rising concerns have suggested the importance of designing recommendation algorithms with “fairness” to eliminate discriminatory biases when recommending items to users. In the context of crowdsourcing systems considered in Chapters 3 and 4, the recommendations may also potentially have discriminatory biases like recommending tasks with greater monetary awards to males than females. In principle, users that are similar in terms of non-protected attributes should receive similar recommendations, and differences in recommendations across protected groups of users can only be justified by non-protected attributes.

The discriminations mentioned above have suggested it is therefore not in the right way if we only target at best matching in a discrimination-aware recommendation system. In the other extreme case, if a recommendation system employs a uniformly random recommendation strategy that provides the users with randomly selected items from the repository contains no discrimination. Such a strategy is neutral from all perspectives but is no longer an efficient recommendation [105]. The sacrifice of the best matching quality is therefore inevitable in a discrimination-aware recommendation system. Many works [24, 25] model the discrimination-aware recommendation as a constraint optimization problem. In principle, to avoid discriminations, protected attributes such as race or gender should not be used in recommendation algorithms. But directly discarding the protected attributes will lead to the redlining effect [106]. In [24], the authors exploited the concept arisen from fairness-aware data mining [26], and proposed to recommend items statistically independent of specified protected attributes. They modeled the recommendation objective by adding a discrimination regularization to the objective function from the previously studied best matching recommendation system [17], and employed the probabilistic matrix factorization technique to get solutions. Reference [25] proposed a novel discrimination-aware tensor recommendation framework designed to improve fairness by isolating and extracting protected features in tensor decompositions.

The aforementioned discrimination-aware recommendations are all offline, i.e., the recommendations are made without taking into account additional information about the users' behaviors that can be gleaned during the users' views sequentially over time. In this thesis, we focus on sequential non-discriminatory recommendations. Specifically, we adopted the linear stochastic bandits model as we have introduced in Section 2.3.2. As an illustration, consider the news recommendation application again. Typical features, including the news' topic categories, the users' race, gender, location, etc., can be treated as components in each arm or decision vector, the platform is the decision maker, while the users' clicks are the stochastic rewards. In the following, we use the terms "item", "decision" and

“arm” interchangeably in our formulation. In traditional linear bandits recommendations [18], given decision vectors containing typical features of each item, we assume the expected reward of choosing each decision is an unknown linear function of the decision vector, the recommendation algorithm tries to maximize the users’ satisfaction by finding the best matching. However, the observed reward (e.g. number of clicks) may be biased by racial factors. In a movie recommend system, a black user may have a history of following a particular black actor, but it does not mean that he or she should always be recommended movies with black actors. To avoid discriminations, in principle, the protected attributes such as race or gender should not be used in recommendation algorithms. Nonetheless, discarding such attributes directly and modeling the reward as a linear function of the other unprotected attributes will potentially leave lingering discrimination in the large bias error of underfitted modelling without enough parameters [22, 23].

In this chapter, we propose a linear stochastic bandit formulation to model the sequential non-discriminatory recommendations. We propose to maximize the (expected) cumulative reward like users’ clicks or ratings over a subspace of decision attributes, based on the reward observed for the full space. We call this *orthogonal projection in linear bandits*. Specifically, the decision vector is projected orthogonally onto a target subspace $\mathbb{U}$. The reward is then explicitly decomposed into two components, one of which is due to the decision vector projected onto $\mathbb{U}$, and the other is due to the decision vector projected onto $\mathbb{U}^\perp$. We call the first component the *projection reward*, corresponding to the unbiased user’s feedback after removing the discriminatory bias, and the second component the *corruption*, corresponding to the discriminatory bias in recommendations. In this thesis, we assume the abstract target subspace $\mathbb{U}$ is known beforehand, and orthogonal projection onto $\mathbb{U}$ will entirely remove the discriminatory bias. The target subspace $\mathbb{U}$ can be found by using mechanisms introduced in some works like [25, 107]. In the literature, there are many different ways to quantify discrimination [108]. For different measurements, the target subspace $\mathbb{U}$ can be different. We do not assume how to get

a specific target subspace $\mathbb{U}$. It is of interest in future research to develop methods to sequentially learn this subspace over time in conjunction with the best arm using possibly additional side-information. We develop a strategy that achieves $O(|\mathbb{D}| \log n)$ cumulative projection regret when there are finitely many (a small number of) decision vectors, where n is the number of time steps. In the case where the decision vectors are drawn from an infinite compact set (or a set with an extremely large number of items), we achieve a cumulative projection regret of $O(n^{2/3}(\log n)^{1/2})$. Here, the projection regret at each time step is defined as the difference between the current expected projection reward of making one decision and the oracle best expected projection reward. This algorithm is based on the ϵ_t -greedy policy [43], which is a simple and well-known algorithm for the standard finite multi-armed bandit problem. In the linear stochastic bandit literature, the best cumulative regret upper bound for the case of a compact decision set is $O(\sqrt{n} \text{polylog}(n))$ [53, 54], where $\text{polylog}(n)$ means polylogarithmic order of n . If a further smoothness assumption is made [54], we show that it is possible to achieve the $O(\sqrt{n})$ projection regret upper bound in our problem formulation. However, the existence of a policy with $O(\sqrt{n} \text{polylog}(n))$ upper bounded cumulative projection regret for the general compact decision set remains an open question. We verify our proposed policies on both synthetic simulations and experiments on a wine quality dataset [109].

The rest of this chapter is organized as follows. In Section 5.2, we present our system model and assumptions. We also review related work in sequential bandits literature to our formulation. In Section 5.3, we introduce our strategies and prove that they achieve sublinear cumulative projection regret. In Section 5.4, we present simulations and experiments to verify the performance of our strategies.

5.2 System Model and Related Work

In this section, we present how to model the sequential non-discriminatory recommendations as an orthogonal projection problem in linear bandits. We then discuss

the related work in sequential bandits literature to our formulation.

5.2.1 Orthogonal Projection in Linear Bandits

The sequential recommendation problem can be naturally modeled as a linear bandit problem as we introduced in Section 2.3.2, where typical features, including the item's categories, the user's race, gender, location, etc., can be treated as components in each arm or decision vector, while the user's feedback like click or rating is the reward. The decision chosen at each time step by the decision maker corresponds to the recommendation made by the platform at each time. Formally, let $\mathbb{D} \subseteq \mathbb{R}^d$ be a compact set of decisions from which the decision maker has to choose a decision X_t at each time step t . The observed reward r_t after choosing X_t is given by

$$r_t = \langle X_t, \boldsymbol{\theta} \rangle + \eta_t, \quad (5.1)$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a fixed but unknown parameter and η_t is a zero mean independent and identically distributed (i.i.d.) random noise.

In the standard linear bandit problem, the performance of a policy is measured by the difference between the decision maker's cumulative reward and the cumulative reward achieved by the oracle policy with knowledge of $\boldsymbol{\theta}$. This corresponds to learning the best matching in recommendations without taking discrimination issues into account. Formally, the goal of the decision maker is to minimize the *cumulative regret* over n time steps defined by

$$R(n) = \sum_{t=1}^n (\langle X^*, \boldsymbol{\theta} \rangle - \mathbb{E}[\langle X_t, \boldsymbol{\theta} \rangle]), \quad (5.2)$$

where $X^* = \arg \max_{\mathbf{x} \in \mathbb{D}} \langle \mathbf{x}, \boldsymbol{\theta} \rangle$ is the optimal decision in the standard linear bandit problem, corresponding to the best matching in recommendations without avoiding discrimination.

To avoid discrimination, we model the non-discriminatory sequential recommendations as the orthogonal projection in linear bandit model instead. Formally, we consider a decomposition of the observed reward into two components. Let $\mathbb{U}$, whose dimension is not more than d , be a subspace of decision attributes. Let $\mathbb{U}^\perp = \{\mathbf{q} \in \mathbb{R}^d : \langle \mathbf{q}, \mathbf{p} \rangle = 0, \forall \mathbf{p} \in \mathbb{U}\}$ be the orthogonal complement of $\mathbb{U}$. It is a standard result [110] that $\mathbb{R}^d = \mathbb{U} \oplus \mathbb{U}^\perp$, and each $\mathbf{w} \in \mathbb{R}^d$ can be written uniquely as a sum $\mathbf{p} + \mathbf{q}$, where $\mathbf{p} \in \mathbb{U}$ and $\mathbf{q} \in \mathbb{U}^\perp$. We define the linear orthogonal projection operator as $P_{\mathbb{U}} : \mathbb{R}^d \mapsto \mathbb{U}$ such that $P_{\mathbb{U}}(\mathbf{w}) = \mathbf{p}$. In general, $P_{\mathbb{U}}$ can be represented by a $d \times d$ matrix whose rank is u . The expected reward can therefore be decomposed as

$$\begin{aligned} \langle X_t, \boldsymbol{\theta} \rangle &= \langle P_{\mathbb{U}}(X_t) + P_{\mathbb{U}^\perp}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) + P_{\mathbb{U}^\perp}(\boldsymbol{\theta}) \rangle \\ &= \langle P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle + \langle P_{\mathbb{U}^\perp}(X_t), P_{\mathbb{U}^\perp}(\boldsymbol{\theta}) \rangle. \end{aligned} \quad (5.3)$$

We call $\langle P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle$ the *projection reward* on subspace $\mathbb{U}$, which is denoted by $r_t^{\mathbb{U}}$, corresponding to the unbiased user's feedback like click or rating. The other part of the observed reward, $\langle P_{\mathbb{U}^\perp}(X_t), P_{\mathbb{U}^\perp}(\boldsymbol{\theta}) \rangle$, is called the *corruption*, corresponding to the discriminatory bias in recommendations. Different from the standard linear bandit model, where the reward can be observed (with some random perturbation), in our orthogonal projection linear bandit model, we do not directly observe the projection reward $r_t^{\mathbb{U}}$ since in recommendations we can only observe the direct feedback like click or rating from users. Furthermore, the *cumulative projection regret* is defined as

$$R_{\mathbb{U}}(n) = \sum_{t=1}^n (\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle - \mathbb{E}[\langle P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle]), \quad (5.4)$$

where

$$X_{\mathbb{U}}^* = \arg \max_{\mathbf{x} \in \mathbb{D}} \langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \quad (5.5)$$

is the *best projection decision*. The objective in our model is to minimize the cumulative projection regret $R_{\mathbb{U}}(n)$ rather than the cumulative regret $R(n)$, based

on the observed returns $r_1, \dots, r_{n-1}$. This indicates that we aim to maximize the cumulative non-discriminatory user's feedbacks rather than the traditional best matching reward in recommendations.

For a concrete illustration, consider again the discrimination prevention problem in a movie recommendation system described in Section 5.1. An arm is a movie to be recommended to a user and is represented by the features of the movie and the user. The features include the movie genre, the movie length, music types used, the age, location and race of the user, etc. Suppose that each arm is a d dimensional vector, where some of its elements like the ages and races may lead to discriminatory recommendations and are "protected". To eliminate the effect of those features when performing the recommendations, the system should consider only the non-protected dimensions. In [107], the authors proposed to control the discrimination effect in a linear regression model, where P_U extracted from their formulation can also be viewed as a projection matrix.

We use $\text{span}(\mathbb{D})$ to denote the set of finite linear combinations of arms in $\mathbb{D}$. In the ideal case, if at each time step t the return r_t is available without the random noise η_t , and if $\mathbb{D}$ is specified by linear inequality constraints, the problem then degenerates to a linear programming problem that finds $\mathbf{x} \in \mathbb{D}$ to maximize

$$\begin{aligned}
& \langle P_U(\mathbf{x}), P_U(\boldsymbol{\theta}) \rangle \\
&= \langle P_U(\mathbf{x}), \boldsymbol{\theta} \rangle \\
&= \langle P_U(\mathbf{x}), P_{\text{span}(\mathbb{D})}(\boldsymbol{\theta}) + P_{\text{span}(\mathbb{D})^\perp}(\boldsymbol{\theta}) \rangle \\
&= \langle P_U(\mathbf{x}), P_{\text{span}(\mathbb{D})}(\boldsymbol{\theta}) \rangle + \langle P_U(\mathbf{x}), P_{\text{span}(\mathbb{D})^\perp}(\boldsymbol{\theta}) \rangle.
\end{aligned} \tag{5.6}$$

However, since $P_{\text{span}(\mathbb{D})^\perp}(\boldsymbol{\theta})$ is unobservable even in the ideal case, and $\arg \max_{\mathbf{x} \in \mathbb{D}} \langle P_U(\mathbf{x}), P_{\text{span}(\mathbb{D})}(\boldsymbol{\theta}) \rangle$ is not equal to X_U^* in (5.5) when $\langle P_U(\mathbf{x}), P_{\text{span}(\mathbb{D})^\perp}(\boldsymbol{\theta}) \rangle \neq 0$, a linear cumulative projection regret is therefore inevitable in the worst case. In order to get a sublinear cumulative regret, further assumptions are required.

A possible assumption is $\langle P_{\mathbb{U}}(\mathbf{x}), P_{\text{span}(\mathbb{D})^\perp}(\boldsymbol{\theta}) \rangle = 0$ for all $\boldsymbol{\theta}$. This is equivalent to saying $\mathbb{U} \subseteq \text{span}(\mathbb{D})$. To see this, note that this assumption means that $P_{\text{span}(\mathbb{D})^\perp}(\boldsymbol{\theta}) \in \text{kernel}(P_{\mathbb{U}}) = P_{\mathbb{U}}^\perp = P_{\mathbb{U}^\perp}$. It follows that $\text{span}(\mathbb{D})^\perp \subseteq \mathbb{U}^\perp$, which is equivalent to $\mathbb{U} \subseteq \text{span}(\mathbb{D})$. We make the following assumption throughout this chapter.

Assumption 5.1. We have $\mathbb{U} \subseteq \text{span}(\mathbb{D})$, the dimension of $\text{span}(\mathbb{D})$ is k , where $\dim(\mathbb{U}) \leq k \leq d$, and $\|P_{\mathbb{U}}(\boldsymbol{\theta})\|_2 > 0$. Furthermore, the decision maker has access to k linearly independent arms in $\mathbb{D}$.

We use $\mathbb{D}_k \subseteq \mathbb{D}$ to denote the set containing the k linearly independent arms in Assumption 5.1. The condition $\|P_{\mathbb{U}}(\boldsymbol{\theta})\|_2 > 0$ guarantees $\boldsymbol{\theta} \notin \mathbb{U}^\perp$. We also make another assumption which is standard in the literature for the linear bandit problem.

Assumption 5.2. We have $\|\boldsymbol{\theta}\|_2 \leq S$, and $\max_{\mathbf{x} \in \mathbb{D}} \|\mathbf{x}\|_2 \leq Z$, where S and Z are positive finite constants. The random variables η_t is drawn i.i.d. from a zero-mean sub-Gaussian distribution with parameter ϑ , i.e., $\mathbb{E}[e^{\xi \eta_t}] \leq e^{\xi^2 \vartheta^2 / 2}$ for all $\xi \in \mathbb{R}$.

In this thesis, we only consider one fixed but unknown parameter $\boldsymbol{\theta}$. In a real recommendation platform with a large number of users and items, to improve the recommendation performance, the above model can be extended to a set of different $\boldsymbol{\theta}$. We can first use cluster techniques like k -means to get several user or item clusters and assign each cluster a specific $\boldsymbol{\theta}$. This is a common technique that has been used in works like [104] to improve the performance.

5.2.2 Related Work

To the best of our knowledge, few studies have been done in the decomposition of the linear stochastic bandit problem. But there are several lines of related work: 1) the linear stochastic bandit model; 2) the multi-objective Pareto bandit problem.

In the linear stochastic bandit model, the decision maker predicts the reward of choosing one decision based on the given context vector of this decision. In [111], the author provided the first analysis of this problem under the name of “Associative Reinforcement Learning with Linear Value Functions”. He proposed an algorithm based on least squares estimation and high probability confidence bounds, and showed that it has $O(\sqrt{dn} \log^{3/2}(n|\mathbb{D}|))$ regret upper bound for the case of finitely many decisions, where $|\mathbb{D}|$ is the number of decisions. Reference [56] extended the algorithm of [111] to the problem with an arbitrary compact set of arms and proved that their policy has $O(d\sqrt{n} \log^{3/2} n)$ regret upper bound. References [53,54] further improved the regret upper bound of the policies based on the new smaller confidence sets established using martingale techniques in their work. In [104], the authors proposed a linear bandit formulation with hidden features where the reward of choosing one decision is the summation of two components, one of which is a linear function of the observable features, and the other is a linear function of the unobservable features. They applied a UCB type linear bandit policy with the coordinate descent [112,113] algorithm in which estimating the hidden features and the unknown coefficients together over time is achieved. However, our work is different from the above work in two ways. First, our objective is to maximize only the cumulative projection reward rather than the reward observed for the full space. Second, we consider a more general formulation where the decision vectors are projected onto two arbitrary orthogonal subspaces.

In the stochastic multi-objective multi-armed bandit (MOMAB) problem [114–117], the reward of making one decision is a vector rather than the single objective in the standard multi-armed bandit problem. Because of the possible conflicting objectives, a set of Pareto optimal arms [115] with the partial order on the multi-objective space [118] is considered in the MOMAB problem instead of a single best arm. Scalarization techniques can be used to transform MOMAB problem to a single objective MAB, and one arm belongs to the Pareto optimal set is regarded as the best arm in a particular scalarization function. In our work, we consider the decomposition of the reward where the decomposition formulation is similar to a

two-objective MAB applied with a linear scalarization function [115]. However, the difference between our model and MOMAB with the linear scalarization is that we can only observe the reward summation at each time step rather than the reward vector in MOMAB, and our objective is to minimize the cumulative projection regret rather than minimizing the three types of regret defined in [115].

Another common scenario chosen to motivate the bandit scheme in the literature is sequential clinical trials [56, 119, 120] which have the aim of balancing the correct identifying of the best treatment (exploration) and the as effective as possible treatment of patients during the trials (exploitation). Here, we show our orthogonal projection in linear bandits have further applications beyond recommendations. The classical sequential clinical trials containing d different treatments can be modeled as an independent d -armed bandit problem, while the more feasible and general way is to model the treatment decision set as a set of mixed drugs instead. One decision here corresponds to one mixed drugs treatment with specific dosages of each kind. The reward at each time step is the curative efficacy after applying the mixed drug treatment to a patient at each trial. However, some clinical trials seek to maximize the curative effect on one disease of a mixed drugs treatment, while at the same time the patients who have this disease may concurrently have another disease. These patients need to take other drugs that have a positive or negative impact on the targeted disease. One typical example is the treatment of hypertension in chronic kidney disease. Hypertension is present in more than 80% of the patients with chronic kidney disease, and drugs like ACEi and ARB targeted for reducing proteinuria may also have an effect on blood pressure [121]. To study the effect of a mixed drug treatment targeted for controlling hypertension, the decision maker is supposed to discount the impact of drugs like ACEi and ARB in the decision process.

5.3 Strategies and Regret Analysis

In this section, we introduce our strategies and prove that they achieve sublinear cumulative projection regret.

5.3.1 Projection Greedy Strategy

The decision set $\mathbb{D}$ may contain finitely (a small fixed number) or infinitely (indicating an extremely large number like several millions of items or a dynamically updating over time repository) many arms. In this section, we present a strategy with two settings, the first is applicable for the finite-arm case, and the other is applicable for the infinite-arm case. In the following, we use $\hat{\boldsymbol{\theta}}_t$ to denote the L_2 regularized least square estimate of $\boldsymbol{\theta}$ with parameter $\lambda > 0$, after the decision making at time step t :

$$\hat{\boldsymbol{\theta}}_t = \mathbf{V}_t^{-1} \sum_{i=1}^t r_i X_i, \quad (5.7)$$

where $\mathbf{V}_t = \sum_{i=1}^t X_i X_i^\top + \lambda \mathbf{I}$. Our Greedy Projection Strategy (GPS) is based on ϵ_t -greedy policy [43], which is a simple and well-known algorithm for the standard finite multi-armed bandit problem. At each time step t , the ϵ_t -greedy policy chooses with probability $1 - \epsilon_t$ the arm with the highest empirical average reward, and with probability ϵ_t a random arm. Since in our problem, we focus our attention on the projection reward, GPS chooses with probability $1 - \epsilon_t$ the arm with the highest empirical average projection reward defined as:

$$\bar{r}_{t-1}^{\mathbf{x}} = \left\langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\rangle, \quad (5.8)$$

for all $\mathbf{x} \in \mathbb{D}$. Another difference is, at each time step t , GPS chooses with probability ϵ_t a random arm from $\mathbb{D}_k$ rather than $\mathbb{D}$. We define two different settings of

ϵ_t as follows, to handle the finite- and infinite-arm cases, respectively:

$$\epsilon_t^f = \min \left\{ 1, \frac{\alpha |\mathbb{D}_k|}{t} \right\}, \epsilon_t^i = \min \left\{ 1, \frac{\alpha |\mathbb{D}_k|}{t^{1/3}} \right\}, \quad (5.9)$$

where “ f ” stands for finite, “ i ” stands for infinite, and $\alpha > 0$ is a fixed positive constant.

The GPS strategy is summarized in Algorithm 5.1.

Algorithm 5.1 GPS

Require: Set ϵ_t to ϵ_t^f or ϵ_t^i according to the number of arms; set hyper parameter $\alpha > 0$ and $\lambda > 0$.

1: **loop**

2: Update ϵ_t .

3: With probability $1 - \epsilon_t$, choose $X_t = \arg \max_{\mathbf{x} \in \mathbb{D}} \bar{r}_{t-1}^{\mathbf{x}}$ and with probability ϵ_t choose a random arm from $\mathbb{D}_k$.

4: Update $\bar{r}_t^{\mathbf{x}}$. Set $t = t + 1$.

5: **end loop**

Note that we have a hyper parameter $\alpha > 0$ in Algorithm 5.1. In the following theorems, we give a sufficient condition for the strategy to achieve $O(\log n)$ for the finite-arm case. We also show that our infinite-arm strategy achieves $O(n^{2/3}(\log n)^{1/2})$ regret for the infinite-arm case. The empirical impact of α is further studied in Section 5.4. Parameter λ can be set as a moderate value like Z^2 .

5.3.2 Regret Analysis

5.3.2.1 Finitely Many Arms

Theorem 5.1. *Suppose $\mathbb{D}$ contains finitely many arms,*

$$\alpha > \max \left\{ \frac{24dZ^2\vartheta^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2 \delta_{\mathbb{D}_k}}, 10 \right\} \quad (5.10)$$

where $\Delta_{\mathbf{x}} = \langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle$ and $\delta_{\mathbb{D}_k}$ is a constant that depends on $\mathbb{D}_k$, then GPS has cumulative projection regret of order $O(|\mathbb{D}| \log n)$, where n is the number of time steps and $|\mathbb{D}|$ is the number of arms.

Theorem 5.1 shows that if we choose α to be sufficiently large, GPS achieves order optimal regret. We prove Theorem 5.1 in the following.

Proof. See Section 5.A. □

5.3.2.2 Infinitely Many Arms

Theorem 5.2. *When $\mathbb{D}$ is a compact set containing infinitely many arms, GPS has cumulative projection regret of order $O(n^{2/3}(\log n)^{1/2})$, where n is the number of time steps.*

Proof. See Section 5.B. □

5.3.3 Additional Smoothness Assumption

In this subsection, we discuss that if additional smoothness assumptions of the decision set $\mathbb{D}$ are available, we can achieve better cumulative projection regret order than $O(n^{2/3}(\log n)^{1/2})$. Define $X_{\mathbb{U}}^*(\boldsymbol{\theta}) = \arg \max_{\mathbf{x} \in \mathbb{D}} P_{\mathbb{U}}^T(\mathbf{x})P_{\mathbb{U}}(\boldsymbol{\theta})$. One typical assumption is

Assumption 5.3. If $\forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d, \exists J \in \mathbb{R}^+, \text{ s.t.}$

$$\|P_{\mathbb{U}}(X_{\mathbb{U}}^*(\boldsymbol{\theta}_1) - X_{\mathbb{U}}^*(\boldsymbol{\theta}_2))\|_2 \leq J \left\| \frac{P_{\mathbb{U}}(\boldsymbol{\theta}_1)}{\|P_{\mathbb{U}}(\boldsymbol{\theta}_1)\|} - \frac{P_{\mathbb{U}}(\boldsymbol{\theta}_2)}{\|P_{\mathbb{U}}(\boldsymbol{\theta}_2)\|} \right\|_2. \quad (5.11)$$

This is to say the projection of the decision set $\mathbb{D}$ and all $\boldsymbol{\theta} \in \mathbb{R}^d$ onto the subspace $\mathbb{U}$ satisfies the SBAR(J) condition [54, 122]. For example, if the decision set $\mathbb{D}$ is a ball or an ellipsoid, it satisfies the condition (5.11). In the proof of Theorem 5.2, we bound (5.29) with $O\left(\beta_t(\mathbb{E}\left[1/\lambda_{t,\min}^{\mathbb{D}_k}\right])^{1/2}\right) = O(t^{-1/3} \log t)$. With the additional Assumption 5.3, we can improve this bound, and further get a tighter upper bound for the cumulative projection regret. We state the following:

Theorem 5.3. *If in GPS, $\epsilon_t = \epsilon_t^s = \min \left\{ 1, \frac{\alpha |\mathbb{D}_k|}{\sqrt{t}} \right\}$, GPS has cumulative projection regret of order $O(\sqrt{n})$, where n is the number of time steps.*

Proof. See Section 5.C. □

5.4 Simulations and Performance Evaluation

In this section, we verify the efficiency of our strategy by performing simulations on synthetic data and a wine quality [109] dataset¹.

We include the following strategies for comparison:

- 1) the Uncertainty Ellipsoid (UE) policy [54], which chooses

$$\arg \max_{\mathbf{x} \in \mathbb{D}} \mathbf{x}^\top \hat{\boldsymbol{\theta}}_{t-1} + \alpha \sqrt{\log t \min\{d \log t, |\mathbb{D}|\}} \|\mathbf{x}\|_{\mathbf{V}_{t-1}^{-1}}$$

at each time step t ,

- 2) the Original Greedy Strategy (OGS) by replacing $\arg \max_{\mathbf{x} \in \mathbb{D}} \bar{r}_{t-1}^{\mathbf{x}}$ with $\arg \max_{\mathbf{x} \in \mathbb{D}} \mathbf{x}^\top \hat{\boldsymbol{\theta}}_{t-1}$ in GPS,
- 3) the Careless Greedy Strategy (CGS) where the X_i in GPS are all replaced by $P_{\mathbb{U}}(X_i)$.

The full CGS is shown in Algorithm 5.2, where

$$\tilde{\boldsymbol{\theta}}_t = \left(\sum_{i=1}^t P_{\mathbb{U}}(X_i) (P_{\mathbb{U}}(X_i))^\top + \lambda \mathbf{I} \right)^{-1} \sum_{i=1}^t r_i P_{\mathbb{U}}(X_i).$$

¹Codes can be found in <https://github.com/KANGQIYU/Sequential-Non-Discriminatory-Recommendations>

Algorithm 5.2 CGS

Require: Set ϵ_t to ϵ_t^f or ϵ_t^i according to the number of arms; set hyper parameter $\alpha > 0$ and $\lambda > 0$.

- 1: **loop**
- 2: Update ϵ_t .
- 3: With probability $1 - \epsilon_t$, choose

$$\arg \max_{\mathbf{x} \in \mathbb{D}} \left\langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\tilde{\boldsymbol{\theta}}_{t-1}) \right\rangle,$$

and with probability ϵ_t choose a random arm from $\mathbb{D}_k$.

- 4: Update $\left\langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\tilde{\boldsymbol{\theta}}_{t-1}) \right\rangle$. Set $t = t + 1$.
 - 5: **end loop**
-

In each simulation, we perform 2000 trials, each with 10^4 time steps. To compare the projection regret performance of different strategies, we compute the empirical average cumulative projection regret over all the trials. For convenience, this average is referred to as the *average cumulative projection regret*.

5.4.1 Experiments on Synthetic Data

In this section, we compare the performance of different strategies using synthetic data. We use three settings in the simulations:

- (a) For the finite-arm case, in each trial, we generate the decision set $\mathbb{D} \subseteq \mathbb{R}^d$ containing K arms, in which each arm is associated with a d -dimension feature vector. Each dimension of the feature vector is drawn i.i.d. from the uniform distribution $\mathcal{U}(-1, 1)$. Each dimension of the ground-truth parameter $\boldsymbol{\theta}$ is also drawn from $\mathcal{U}(-1, 1)$. We next generate the orthogonal projection matrix $P_{\mathbb{U}} = \mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top$, where $\mathbf{A}$ is a $d \times u$ ($u < d$) matrix whose elements are generated randomly from $\mathcal{U}(-1, 1)$. η_t at each time step is drawn i.i.d. from the Gaussian distribution $\mathcal{N}(0, \vartheta^2)$ with variance ϑ^2 . The decision set $\mathbb{D}$, parameter $\boldsymbol{\theta}$ and projection matrix $P_{\mathbb{U}}$ are fixed in each trial.
- (b) We use the same setting as in setting (a) except that the projection matrix $P_{\mathbb{U}}$ in this setting is a diagonal matrix whose (i, i) entry is 1 for $i = 1, \dots, u$

and 0 otherwise. This means that the last $d - u$ dimensions of decision vectors are the protected features.

- (c) For the infinite-arm case, in each trial, the decision set $\mathbb{D}$ is limited to a convex set for ease of computation. Specifically, we use the convex set $\mathbb{D}$:

$$\mathbb{D} = \left\{ \mathbf{x} : \sum_{i=1}^d \mathbf{x}(i) \log \mathbf{x}(i) \leq 5 \text{ and } \forall i, \mathbf{x}(i) \geq 0 \right\}, \quad (5.12)$$

where $\mathbf{x}(i)$ is the i -th entry of $\mathbf{x}$ and $0 \log 0 := 0$. $\boldsymbol{\theta}$, $P_{\mathbb{U}}$ and η_t are generated the same way we have done in setting (a).

In the following, GPS, OGS, and CGS use the setting $\epsilon = \epsilon_t^f$ described in Section 5.3 when using setting (a) and (b) for the finite-arm case. Accordingly, $\epsilon = \epsilon_t^i$ is used when using setting (c) for the infinite-arm case. In all simulations, the parameter λ is set to 1.

5.4.1.1 Varying α

All the strategies require a parameter α . We do simulations using setting (a) with $d = 10$, $K = 45$, $\vartheta = 0.5$, $u = 5$, and varied α , and setting (c) with $d = 4$, $\vartheta = 0.5$, $u = 2$, and varied α . Figs. 5.1a and 5.1b show how the average cumulative projective regret at time step 10^4 changes with varied α in each strategy. We observe the following:

- We note that a moderate α is optimal for each strategy. When α is too large, the strategy explores too frequently, leading to a large projection regret as expected. On the other hand, a small α results in little exploration, therefore, good decisions cannot be detected efficiently.
- In Fig. 5.1a, GPS with $\alpha = 1$ outperforms all the other strategies. This is because the other strategies do not have an asymptotically unbiased estimation of the projection reward. Fig. 5.1b shows similar results.

(A) Using setting (a) with $d = 10$, $K = 45$, and $u = 5$ (B) Using setting (c) with $d = 4$, and $u = 2$ FIG. 5.1: Regret comparison between different strategies with $\vartheta = 0.5$ and varying α .

In the following simulations, for a fair comparison, we tune the parameter α for each strategy. Specifically, we set $\alpha = 1$ for all the strategies except UE, which is given a $\alpha = 0.1$ when using setting (a) and (b). When using setting (c), we set $\alpha = 0.1$ for all the strategies except GPS, which is given a $\alpha = 0.01$.

(A) Average cumulative projection regret

(B) Percentage of trials where the best decision is found

FIG. 5.2: Performance comparison between different strategies using setting (a) with $d = 10$, $K = 45$, $\vartheta = 0.5$ and $u = 5$.

5.4.1.2 Results and Analysis

In the simulations for the finite-arm case using setting (a) and (b), we set $d = 10$, $K = 45$, $\vartheta = 0.5$ and $u = 5$. The simulation results are shown in Figs. 5.2a, 5.2b, 5.3a and 5.3b. We observe the following:

- The average cumulative projection regrets of different strategies are shown in Figs. 5.2a and 5.3a. We see that GPS has obvious sublinear cumulative

(A) Average cumulative projection regret

(B) Percentage of trials where the best decision is found

FIG. 5.3: Performance comparison between different strategies using setting (b) with $d = 10$, $K = 45$, $\vartheta = 0.5$ and $u = 5$.

projection regret performance. UE, OGS and CGS all suffer from a linearly increasing cumulative projection regret. This verifies that if our objective is to maximize the cumulative projection reward, GPS is more applicable.

- Figs. 5.2b and 5.3b show the percentage of trials where the best decision is found successfully by different strategies. We see that GPS finds the best decision in most trials.

In the simulations for the infinite-arm case using setting (c), we set $d = 4$, $\vartheta =$

0.5 and $u = 2$. From Fig. 5.4, we also observe that GPS has obvious sublinear cumulative projection regret performance. UE, OGS and CGS all suffer from a linearly increasing cumulative projection regret. This verifies the efficiency of GPS for the infinite-arm case.

FIG. 5.4: Average cumulative projection regret using setting (c) with $d = 4$, $u = 2$ and $\vartheta = 0.5$.

5.4.2 Experiments on Real Dataset

In this section, we compare the performance of different strategies on the wine quality dataset which contains 11-dimension description vectors of 4898 white wines, and their ratings (scores between 0 and 10). At each trial, we randomly select 200 wines with rating 4, 5, 6, 7 and 8, as the decision set $\mathbb{D}$, since among all the wines there are only 5 wines with ratings larger than 8 and 20 wines with ratings less than 4. Each dimension of the wine description vector is a physicochemical characteristic like volatile acidity, chlorides, or density. Due to privacy and logistic issues [109], only the physicochemical characteristics and ratings are available (e.g., there is no data about grape types, wine brand, wine selling price, etc.), we add one additional feature which is drawn i.i.d from $\mathcal{U}(0, 1)$ as the protected feature.

The corresponding rating of each wine is then corrupted by subtracting 4 times this protected feature value from the original rating. In this experiment, we take the original rating as the projection reward. Finally, we add a constant 1 as the constant feature to each description vector. If we put the protected feature as the final dimension of each wine, then $d = 13$, and the projection matrix P_U is defined as a diagonal matrix whose (i, i) entry is 1 for $i = 1, \dots, 12$ and 0 otherwise. We use $\epsilon = \epsilon_t^f$ in this experiment. The results are shown in Fig. 5.5, from which we observe that GPS outperforms all the other strategies.

FIG. 5.5: Average cumulative projection regret using the Wine Quality dataset.

5.A Proof of Theorem 5.1

For any arm $\mathbf{x} \in \mathbb{D}$, let the random variable $N_n^{\mathbf{x}}$ denote the total number of time steps within the first t time steps at which arm $\mathbf{x}$ was chosen. It follows that $N_n^{\mathbf{x}} = \sum_{t=1}^n \mathbf{1}_{\{X_t = \mathbf{x}\}}$.

Since

$$\begin{aligned} \mathbb{E}[R_U(n)] &= \mathbb{E} \sum_{t=1}^n \left(\langle P_U(X_t^*), P_U(\boldsymbol{\theta}) \rangle - \langle P_U(X_t), P_U(\boldsymbol{\theta}) \rangle \right) \\ &= \sum_{\mathbf{x} \in \mathbb{D}} \mathbb{E}[N_n^{\mathbf{x}}] \Delta_{\mathbf{x}} \\ &= \sum_{\mathbf{x} \in \mathbb{D}} \Delta_{\mathbf{x}} \sum_{t=1}^n \mathbb{P}(X_t = \mathbf{x}), \end{aligned} \quad (5.13)$$

where $\Delta_{\mathbf{x}} = \langle P_U(X_U^*) - P_U(\mathbf{x}), P_U(\boldsymbol{\theta}) \rangle$ is the projection regret of arm $\mathbf{x}$, it is enough to show the probability $\mathbb{P}(X_t = \mathbf{x}) = O(t^{-1})$ for all suboptimal $\mathbf{x} \in \mathbb{D} \setminus \{X_U^*\}$. In the following, for simplicity, we use a simplified notation $\bar{r}_t^* = \bar{r}_t^{X_U^*}$. It is obvious that

$$\mathbb{P}(X_t = \mathbf{x}) \leq \frac{\epsilon_t}{|\mathbb{D}_k|} + (1 - \epsilon_t) \mathbb{P}(\bar{r}_{t-1}^{\mathbf{x}} \geq \bar{r}_{t-1}^*), \quad (5.14)$$

and the term $\mathbb{P}(\bar{r}_{t-1}^{\mathbf{x}} \geq \bar{r}_{t-1}^*)$ on the right hand side can be decomposed to

$$\begin{aligned} \mathbb{P}(\bar{r}_{t-1}^{\mathbf{x}} \geq \bar{r}_{t-1}^*) &\leq \mathbb{P}\left(\bar{r}_{t-1}^{\mathbf{x}} \geq \langle P_U(\mathbf{x}), P_U(\boldsymbol{\theta}) \rangle + \frac{\Delta_{\mathbf{x}}}{2}\right) \\ &\quad + \mathbb{P}\left(\bar{r}_{t-1}^* \leq \langle P_U(X_U^*), P_U(\boldsymbol{\theta}) \rangle - \frac{\Delta_{\mathbf{x}}}{2}\right). \end{aligned} \quad (5.15)$$

We next bound the term $\mathbb{P}(\bar{r}_{t-1}^{\mathbf{x}} \geq \langle P_U(\mathbf{x}), P_U(\boldsymbol{\theta}) \rangle + \frac{\Delta_{\mathbf{x}}}{2})$ in (5.15). The analysis for $\mathbb{P}(\bar{r}_{t-1}^* \leq \langle P_U(X_U^*), P_U(\boldsymbol{\theta}) \rangle - \frac{\Delta_{\mathbf{x}}}{2})$ is the same.

Let $\beta_t = \vartheta \sqrt{3d \log(t+1)} + \lambda^{1/2} S$. From the proof of Theorem 2 in [53], we have that with probability at least $1 - \delta$, $\forall t \geq 0, \forall \mathbf{x} \in \mathbb{R}^d$

$$\left| \langle \mathbf{x}, \hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta} \rangle \right| \leq \|\mathbf{x}\|_{\mathbf{V}_t^{-1}} \left(\vartheta \sqrt{d \log \left(\frac{1+t}{\delta} \right)} + \lambda^{1/2} S \right). \quad (5.16)$$

Since $\langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle = \langle P_{\mathbb{U}}(\mathbf{x}), \boldsymbol{\theta} \rangle$ for orthogonal projection operator $P_{\mathbb{U}}$, by setting $\delta = \frac{1}{t^2}$ in (5.16), we get for all $\mathbf{x} \in \mathbb{D} \setminus \{X_{\mathbb{U}}^*\}$,

$$\begin{aligned} & \mathbb{P} \left(\bar{r}_t^{\mathbf{x}} - \langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle > \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right) \\ & \leq \mathbb{P} \left(\left\langle P_{\mathbb{U}}(\mathbf{x}), \hat{\boldsymbol{\theta}}_t \right\rangle - \langle P_{\mathbb{U}}(\mathbf{x}), \boldsymbol{\theta} \rangle > \right. \\ & \quad \left. \left(\vartheta \sqrt{d \log \left(\frac{1+t}{t^2} \right)} + \lambda^{1/2} S \right) \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right) \\ & \leq \frac{1}{t^2}. \end{aligned} \quad (5.17)$$

It follows that

$$\begin{aligned} & \mathbb{P} \left(\bar{r}_t^{\mathbf{x}} \geq \langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle + \frac{\Delta_{\mathbf{x}}}{2} \right) \\ & = \mathbb{P} \left(\bar{r}_t^{\mathbf{x}} \geq \langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle + \frac{\Delta_{\mathbf{x}}}{2} \mid \frac{\Delta_{\mathbf{x}}}{2} > \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right) \\ & \quad \cdot \mathbb{P} \left(\frac{\Delta_{\mathbf{x}}}{2} > \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right) \\ & + \mathbb{P} \left(\bar{r}_t^{\mathbf{x}} \geq \langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle + \frac{\Delta_{\mathbf{x}}}{2} \mid \frac{\Delta_{\mathbf{x}}}{2} \leq \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right) \\ & \quad \cdot \mathbb{P} \left(\frac{\Delta_{\mathbf{x}}}{2} \leq \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right) \\ & \leq \frac{1}{t^2} + \mathbb{P} \left(\frac{\Delta_{\mathbf{x}}}{2} \leq \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right). \end{aligned} \quad (5.18)$$

We next show that when the hyper parameter α is large enough,

$$\mathbb{P}\left(\frac{\Delta_{\mathbf{x}}}{2} \leq \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}}\right) = O(t^{-1}).$$

From Assumption 5.1, since $\mathbb{U} \subseteq \text{span}(\mathbb{D})$, we show the increasing of the eigenvalues of $\mathbf{V}_t$ in $\text{span}(\mathbb{D})$ which will help us to bound the value of $\|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}}$.

We have

$$\begin{aligned} \mathbf{V}_t &= \lambda \mathbf{I} + \sum_{\mathbf{x} \in \mathbb{D}} N_t^{\mathbf{x}} \mathbf{x} \mathbf{x}^{\top} \\ &= \lambda P_{\text{span}(\mathbb{D})^{\perp}}(\mathbf{I}) + \lambda P_{\text{span}(\mathbb{D})}(\mathbf{I}) + \sum_{\mathbf{x} \in \mathbb{D}} N_t^{\mathbf{x}} \mathbf{x} \mathbf{x}^{\top}. \end{aligned}$$

The eigenvectors and corresponding eigenvalues of $\mathbf{V}_t$ can therefore be divided into two groups, in one of which the eigenvectors are in $\text{span}(\mathbb{D})$, and in the other the eigenvectors are in $\text{span}(\mathbb{D})^{\perp}$. This conclusion is obvious and can be easily proved by induction. Let $\lambda_{t,\min}^{\mathbb{D}}$ be the smallest eigenvalue of $\mathbf{V}_t$ in $\text{span}(\mathbb{D})$ and $\lambda_{t,\min}^{\mathbb{D}_k}$ be the smallest eigenvalue of $(\lambda \mathbf{I} + \sum_{\mathbf{x} \in \mathbb{D}_k} N_t^{\mathbf{x}} \mathbf{x} \mathbf{x}^{\top})$ in $\text{span}(\mathbb{D})$. If we define

$$\mathbf{y}_1 = \arg \min_{\mathbf{y} \in \text{span}(\mathbb{D}), \|\mathbf{y}\|_2^2=1} \langle \mathbf{y}, \mathbf{V}_t \mathbf{y} \rangle,$$

we then have

$$\begin{aligned} \lambda_{t,\min}^{\mathbb{D}} &= \mathbf{y}_1^{\top} \mathbf{V}_t \mathbf{y}_1 \\ &\geq \mathbf{y}_1^{\top} (\lambda \mathbf{I} + \sum_{\mathbf{x} \in \mathbb{D}_k} N_t^{\mathbf{x}} \mathbf{x} \mathbf{x}^{\top}) \mathbf{y}_1 \\ &\geq \min_{\mathbf{y} \in \text{span}(\mathbb{D}), \|\mathbf{y}\|_2^2=1} \mathbf{y}^{\top} (\lambda \mathbf{I} + \sum_{\mathbf{x} \in \mathbb{D}_k} N_t^{\mathbf{x}} \mathbf{x} \mathbf{x}^{\top}) \mathbf{y} \\ &= \lambda_{t,\min}^{\mathbb{D}_k} \end{aligned} \tag{5.19}$$

Because $P_{\mathbb{U}}(\mathbf{x}) \in \text{span}(\mathbb{D})$, using (5.19), we obtain

$$\mathbb{P}\left(\frac{\Delta_{\mathbf{x}}}{2} \leq \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}}\right)$$

$$\begin{aligned}
&\leq \mathbb{P} \left(\left(\frac{\Delta_{\mathbf{x}}}{2\beta_t} \right)^2 \leq \frac{\|P_{\mathbb{U}}(\mathbf{x})\|_2^2}{\lambda_{t,\min}^{\mathbb{D}}} \right) \\
&= \mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}} \leq \frac{4\beta_t^2 \|P_{\mathbb{U}}(\mathbf{x})\|_2^2}{\Delta_{\mathbf{x}}^2} \right) \\
&\leq \mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2} \right)
\end{aligned} \tag{5.20}$$

Set

$$f : \{\mathbf{y} : \mathbf{y} \in \text{span}(\mathbb{D}), \|\mathbf{y}\|_2^2 = 1\} \mapsto \mathbf{y}^\top \left(\sum_{\mathbf{x} \in \mathbb{D}_k} \mathbf{x} \mathbf{x}^\top \right) \mathbf{y} \tag{5.21}$$

Then $f(\mathbf{y}) > 0$ since $\mathbb{D}_k$ contains k linearly independent arms. We can get $\delta_{\mathbb{D}_k} =_{\text{def}} \inf f(\mathbf{y}) > 0$ since $f(\mathbf{y})$ is a continuous function on a compact set. Define the event:

$$\left\{ \delta_{\mathbb{D}_k} N_t^{\mathbf{x}} > \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2}, \text{ for all } \mathbf{x} \in \mathbb{D}_k \right\}.$$

Under this event, from the last equality in (5.19), we get $\lambda_{t,\min}^{\mathbb{D}_k} > \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2}$.

Therefore, we have

$$\begin{aligned}
&\mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2} \right) \\
&\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(N_t^{\mathbf{x}} \leq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2 \delta_{\mathbb{D}_k}} \right) \\
&\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(\tilde{N}_t^{\mathbf{x}} \leq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2 \delta_{\mathbb{D}_k}} \right),
\end{aligned} \tag{5.22}$$

where $\tilde{N}_t^{\mathbf{x}}$ is the number of times that arm $\mathbf{x} \in \mathbb{D}_k$ was chosen *randomly* during the first t time steps. From the proof of Theorem 3 in [43] (where Bernstein's inequality [123] was used), we have

$$\mathbb{P} \left(\tilde{N}_{\mathbf{x}}^t \leq \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \right) \leq \exp \left(-\frac{1}{10|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \right) \tag{5.23}$$

For $t \geq t' = \alpha|\mathbb{D}_k|$, $\epsilon_t = \epsilon_t^f = \frac{\alpha|\mathbb{D}_k|}{t}$, and we obtain

$$\begin{aligned} \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i &= \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^{t'} \epsilon_i + \frac{1}{2|\mathbb{D}_k|} \sum_{i=t'+1}^t \epsilon_i \\ &\geq \frac{\alpha}{2} \log \frac{t}{\alpha|\mathbb{D}_k|} \end{aligned} \quad (5.24)$$

If $\frac{\alpha}{2} \log \frac{t}{\alpha|\mathbb{D}_k|} \geq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2 \delta_{\mathbb{D}_k}}$ and $\alpha > 10$, from (5.22) to (5.24), we have

$$\begin{aligned} &\mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2} \right) \\ &\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(\tilde{N}_t^{\mathbf{x}} \leq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2 \delta_{\mathbb{D}_k}} \right) \\ &\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(\tilde{N}_t^{\mathbf{x}} \leq \frac{\alpha}{2} \log \frac{t}{\alpha|\mathbb{D}_k|} \right) \\ &\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(\tilde{N}_t^{\mathbf{x}} \leq \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \right) \\ &\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \exp \left(-\frac{1}{10|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \right) \\ &\leq k \left(\frac{\alpha|\mathbb{D}_k|}{t} \right)^{\frac{\alpha}{10}} \\ &\leq o(t^{-1}), \end{aligned} \quad (5.25)$$

where the second and third inequality follows from $\frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \geq \frac{\alpha}{2} \log \frac{t}{\alpha|\mathbb{D}_k|} \geq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2 \delta_{\mathbb{D}_k}}$, the penultimate inequality follows from (5.24), and (5.25) follows from $\alpha > 10$.

Recall that $\beta_t = \vartheta \sqrt{3d \log(t+1)} + \lambda^{1/2} S$. Therefore, if $\alpha > \max\{\frac{24dZ^2\vartheta^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2 \delta_{\mathbb{D}_k}}, 10\}$, from (5.18), (5.20) and (5.25), we have

$$\begin{aligned} &\mathbb{P} \left(\bar{r}_t^{\mathbf{x}} \geq \langle P_{\mathbb{U}}(\mathbf{x}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle + \frac{\Delta_{\mathbf{x}}}{2} \right) \\ &\leq \frac{1}{t^2} + \mathbb{P} \left(\frac{\Delta_{\mathbf{x}}}{2} \leq \beta_t \|P_{\mathbb{U}}(\mathbf{x})\|_{\mathbf{V}_t^{-1}} \right) \\ &\leq \frac{1}{t^2} + \mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{4\beta_t^2 Z^2}{\min_{\mathbf{x} \neq X_{\mathbb{U}}^*} \Delta_{\mathbf{x}}^2} \right) \end{aligned}$$

$$\leq o(t^{-1}), \quad (5.26)$$

where the first inequality follows from (5.18), and the second inequality follows from (5.20).

Similarly, we have $\mathbb{P}(\bar{r}_t^{\mathbf{x}} \leq \langle P_{\mathbb{U}}(X_{\mathbb{U}}^*), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle - \frac{\Delta_{\mathbf{x}}}{2}) = o(t^{-1})$. Then, using (5.14) and (5.15), we obtain

$$\mathbb{P}(X_t = \mathbf{x}) \leq \frac{\alpha}{t} + o(t^{-1}) = O(t^{-1}), \quad (5.27)$$

and, using (5.13), we conclude $\mathbb{E}[R_{\mathbb{U}}(n)] = O(|\mathbb{D}| \log n)$. The proof of Theorem 5.1 is now complete.

5.B Proof of Theorem 5.2

The contribution to the projection regret at time step t comes from two cases: 1) when X_t is chosen randomly, or 2) when X_t is chosen as $\arg \max_{\mathbf{x} \in \mathbb{D}} \bar{r}_{t-1}^{\mathbf{x}}$. We denote these two event as E_t and E_t^c respectively.

$$\begin{aligned} & \mathbb{E}[R_{\mathbb{U}}(n)] \\ &= \mathbb{E} \sum_{t=1}^n \langle P_{\mathbb{U}}(X_{\mathbb{U}}^*), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle - \langle P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \\ &= \sum_{t=1}^n \epsilon_t \mathbb{E}[\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle | E_t] \\ &\quad + (1 - \epsilon_t) \mathbb{E}[\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle | E_t^c] \\ &\leq \sum_{t=1}^n 2ZS\epsilon_t + \mathbb{E}[\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle | E_t^c], \end{aligned} \quad (5.28)$$

where the inequality follows from Assumption 5.1, $\|P_{\mathbb{U}}(\mathbf{x})\|_2 \leq \|\mathbf{x}\|_2$ for all $\mathbf{x}$, and Cauchy-Schwarz inequality.

We next get an upper bound for the second term $\mathbb{E}[\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle | E_t^c]$ in (5.28).

Under the event E_t^c , $X_t = \arg \max_{\mathbf{x} \in \mathbb{D}} \bar{r}_{t-1}^{\mathbf{x}}$. We have

$$\begin{aligned}
& \langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \\
&= \left\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\rangle \\
&\quad + \left\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) - P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\rangle \\
&\leq \left\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) - P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\rangle,
\end{aligned} \tag{5.29}$$

where the inequality follows from $X_t = \arg \max_{\mathbf{x} \in \mathbb{D}} \bar{r}_{t-1}^{\mathbf{x}}$.

We use the notation β_t defined in the proof of Theorem 5.1 again. For any $t \geq 2$, let the event F_t be

$$\left\{ \langle P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \geq \|P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_t)\|_{\mathbf{V}_{t-1}^{-1}} \beta_{t-1} \right\}.$$

From (5.16) and (5.29), since $P_{\mathbb{U}}^{\top}(\mathbf{x})P_{\mathbb{U}}(\boldsymbol{\theta}) = P_{\mathbb{U}}^{\top}(\mathbf{x})\boldsymbol{\theta}$ for orthogonal projection operator $P_{\mathbb{U}}$ we have

$$\begin{aligned}
& \mathbb{P}(F_{t+1}) \\
&\leq \mathbb{P}\left(\left\langle P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_{t+1}), \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t \right\rangle \geq \|P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_{t+1})\|_{\mathbf{V}_t^{-1}} \beta_t\right) \\
&\leq \mathbb{P}\left(\left\langle P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_{t+1}), \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t \right\rangle \geq \|P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_{t+1})\|_{\mathbf{V}_t^{-1}} \left(\vartheta \sqrt{d \log \left(\frac{1+t}{t^2}\right)} + \lambda^{1/2} S\right)\right) \\
&\leq \frac{1}{t^2},
\end{aligned} \tag{5.30}$$

where the final inequality follows from the concentration inequalities (5.16) with $\delta = \frac{1}{t^2}$.

We then have

$$\begin{aligned}
& \mathbb{E} \left[\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_{t+1}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \mid E_{t+1}^c \right] \\
&= \mathbb{E} \left[\langle P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_{t+1}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \mid E_{t+1}^c, F_{t+1} \right] \mathbb{P}(F_{t+1})
\end{aligned}$$

$$\begin{aligned}
& + \mathbb{E} \left[\langle P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_{t+1}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \mid E_{t+1}^c, F_{t+1}^c \right] \mathbb{P}(F_{t+1}^c) \\
& \leq \frac{2ZS}{t^2} + \mathbb{E} \left[\|P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_{t+1})\|_{\mathbf{V}_t^{-1}\beta_t} \mid E_{t+1}^c \right] \\
& \leq \frac{2ZS}{t^2} + \mathbb{E} \left[2Z\beta_t \sqrt{\frac{1}{\lambda_{t,\min}^{\mathbb{D}}}} \mid E_{t+1}^c \right] \tag{5.31}
\end{aligned}$$

$$\leq \frac{2ZS}{t^2} + \mathbb{E} \left[2Z\beta_t \sqrt{\frac{1}{\lambda_{t,\min}^{\mathbb{D}}}} \right] \tag{5.32}$$

$$\leq \frac{2ZS}{t^2} + 2Z\beta_t \sqrt{\mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}}} \right]} \tag{5.33}$$

$$\leq \frac{2ZS}{t^2} + 2Z\beta_t \sqrt{\mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \right]}. \tag{5.34}$$

Here the first equality follows from the independence of F_{t+1} and E_{t+1}^c . (5.31) follows because $\|P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_t)\|_{\mathbf{V}_t^{-1}}^2 \leq \|P_{\mathbb{U}}(X_{\mathbb{U}}^* - X_t)\|_2^2 (\lambda_{t,\min}^{\mathbb{D}})^{-1}$. (5.32) follows because E_{t+1}^c and $\lambda_{t,\min}^{\mathbb{D}}$ are independent. (5.33) follows from Jensen's inequality. (5.34) follows from (5.19).

We next show that $\mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \right] = O(t^{-2/3})$. We need to first show $\mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} < t^{2/3} \right) \leq O(e^{-t^{2/3}})$. Similar to what we have done in (5.22), define the following event:

$$\left\{ N_t^{\mathbf{x}} > \frac{3\alpha}{4} t^{2/3}, \text{ for all } \mathbf{x} \in \mathbb{D}_k \right\}.$$

Under this event, from the last equality in (5.19), we get $\lambda_{t,\min}^{\mathbb{D}_k} > \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k}$.

Therefore, we have

$$\begin{aligned}
& \mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k} \right) \\
& \leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(N_t^{\mathbf{x}} \leq \frac{3\alpha}{4} t^{2/3} \right) \\
& \leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(\tilde{N}_t^{\mathbf{x}} \leq \frac{3\alpha}{4} t^{2/3} \right), \tag{5.35}
\end{aligned}$$

where $\tilde{N}_T^{\mathbf{x}}$ is the number of times that arm $\mathbf{x} \in \mathbb{D}_k$ was chosen *randomly* during the first t time steps. Recall from (5.23) that

$$\mathbb{P} \left(\tilde{N}_{\mathbf{x}}^t \leq \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \right) \leq \exp \left(-\frac{1}{10|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \right) \quad (5.36)$$

For $t \geq t' = (\alpha|\mathbb{D}_k|)^3$, $\epsilon_t = \epsilon_t^i \frac{\alpha|\mathbb{D}_k|}{t^{1/3}}$ and we obtain

$$\begin{aligned} \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i &= \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^{t'} \epsilon_i + \frac{1}{2|\mathbb{D}_k|} \sum_{i=t'+1}^t \epsilon_i \\ &\geq \frac{3\alpha}{4} t^{2/3} + \frac{\alpha^3 |\mathbb{D}_k|^2}{2} \end{aligned} \quad (5.37)$$

It follows from (5.35) to (5.37) that

$$\begin{aligned} &\mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k} \right) \\ &\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(\tilde{N}_t^{\mathbf{x}} \leq \frac{3\alpha}{4} t^{2/3} \right), \\ &\leq \sum_{\mathbf{x} \in \mathbb{D}_k} \mathbb{P} \left(\tilde{N}_{\mathbf{x}}^t \leq \frac{1}{2|\mathbb{D}_k|} \sum_{i=1}^t \epsilon_i \right), \\ &\leq k \exp \left(-\frac{3\alpha}{20} t^{2/3} \right) \end{aligned} \quad (5.38)$$

Now, we can conclude

$$\begin{aligned} &\mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \right] \\ &= \mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \middle| \lambda_{t,\min}^{\mathbb{D}_k} > \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k} \right] \mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} > \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k} \right) \\ &\quad + \mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \middle| \lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k} \right] \mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} \leq \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k} \right) \\ &\leq \frac{4t^{-2/3}}{3\alpha\delta_{\mathbb{D}_k}} \mathbb{P} \left(\lambda_{t,\min}^{\mathbb{D}_k} > \frac{3\alpha}{4} t^{2/3} \delta_{\mathbb{D}_k} \right) + \frac{1}{\lambda} \exp \left(-\frac{3\alpha}{20} t^{2/3} \right) \\ &\leq \frac{5t^{-2/3}}{3\alpha\delta_{\mathbb{D}_k}}, \end{aligned} \quad (5.39)$$

when t is large enough, where the penultimate inequality follows from (5.19).

From (5.34) and (5.39), when t is large enough, we have

$$\begin{aligned}
& \mathbb{E} [\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_{t+1}), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \mid E_{t+1}^c] \\
& \leq \frac{2ZS}{t^2} + 2Z\beta_t \sqrt{\mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \right]} \\
& \leq \frac{2ZS}{t^2} + 2Z\beta_t \sqrt{\frac{5}{3\alpha\delta_{\mathbb{D}_k}}} t^{-\frac{1}{3}}
\end{aligned} \tag{5.40}$$

Finally, from (5.28), we have

$$\begin{aligned}
& \mathbb{E} [R_{\mathbb{U}}(n)] \\
& \leq \sum_{t=1}^n 2ZS\epsilon_t + \mathbb{E} [\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \mid E_t^c] \\
& \leq \sum_{t=1}^n \frac{2ZS\alpha|\mathbb{D}_k|}{t^{1/3}} + 2Z\beta_t \sqrt{\frac{5}{3\alpha\delta_{\mathbb{D}_k}}} t^{-\frac{1}{3}} + C_1 \\
& \leq \int_1^n \frac{2ZS\alpha|\mathbb{D}_k|}{x^{1/3}} + 4Z\vartheta \sqrt{\frac{5d \log x}{\alpha\delta_{\mathbb{D}_k}}} x^{-\frac{1}{3}} dx + C_2 \\
& = O(n^{2/3}(\log n)^{1/2}),
\end{aligned} \tag{5.41}$$

where C_1 and C_2 are two constants. The proof of Theorem 5.2 is now complete.

5.C Proof of Theorem 5.3

We can still use (5.28) and (5.29) from the proof of Theorem 5.2. We have

$$\begin{aligned}
& \mathbb{E} [\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \mid E_t^c] \\
& \leq \mathbb{E} \left[\left\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) - P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\rangle \mid E_t^c \right] \\
& \leq \mathbb{E} \left[\|P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t)\|_2 \left\| P_{\mathbb{U}}(\boldsymbol{\theta}) - P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\|_2 \mid E_t^c \right] \\
& \leq \mathbb{E} \left[J \left\| \frac{P_{\mathbb{U}}(\boldsymbol{\theta})}{\|P_{\mathbb{U}}(\boldsymbol{\theta})\|} - \frac{P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1})}{\|P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1})\|} \right\|_2 \left\| P_{\mathbb{U}}(\boldsymbol{\theta}) - P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\|_2 \mid E_t^c \right]
\end{aligned}$$

$$\begin{aligned}
&\leq \mathbb{E} \left[\frac{2J \left\| P_{\mathbb{U}}(\boldsymbol{\theta}) - P_{\mathbb{U}}(\hat{\boldsymbol{\theta}}_{t-1}) \right\|_2^2}{\left\| P_{\mathbb{U}}(\boldsymbol{\theta}) \right\|} \middle| E_t^c \right] \\
&\leq \mathbb{E} \left[\frac{2J \left\| P_{\mathbb{U}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{t-1}) \right\|_2^2}{\left\| P_{\mathbb{U}}(\boldsymbol{\theta}) \right\|} \right]
\end{aligned} \tag{5.42}$$

where, the first inequality follows from (5.29), the second inequality follows from Cauchy-Schwarz inequality, the third inequality follows from Assumption 5.3, the penultimate inequality follows from Lemma 3.5 in [54], and the last inequality follows from independence.

We next bound $\mathbb{E} \left[\left\| P_{\mathbb{U}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t) \right\|_2^2 \right]$. From (5.7), we have,

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_t &= \left(\sum_{i=1}^t X_i X_i^\top + \lambda I \right)^{-1} \sum_{i=1}^t r_i X_i \\
&= \left(\sum_{i=1}^t X_i X_i^\top + \lambda I \right)^{-1} \sum_{i=1}^t X_i (X_i^\top \boldsymbol{\theta} + \eta_i).
\end{aligned}$$

which yields

$$\begin{aligned}
&\mathbb{E} \left[\left\| P_{\mathbb{U}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t) \right\|_2^2 \right] \\
&\leq \mathbb{E} \left[\left\| P_{\mathbb{U}}(\lambda V_t^{-1} \boldsymbol{\theta}) \right\|_2^2 + \left\| \sum_{i=1}^t P_{\mathbb{U}}(V_t^{-1} X_i) \eta_i \right\|_2^2 \right] \\
&\leq \lambda^2 S^2 \mathbb{E} \left[\frac{1}{(\lambda_{t,\min}^{\mathbb{D}})^2} \right] + \zeta \mathbb{E} \left[\sum_{i=1}^t P_{\mathbb{U}}(V_t^{-1} X_i)^\top P_{\mathbb{U}}(V_t^{-1} X_i) \right] \\
&\leq \lambda^2 S^2 \mathbb{E} \left[\frac{1}{(\lambda_{t,\min}^{\mathbb{D}})^2} \right] + \zeta \mathbb{E} \left[\sum_{i=1}^t \text{tr} (P_{\mathbb{U}}(V_t^{-1}) V_t^{-1} (X_i X_i^\top)) \right] \\
&\leq \lambda^2 S^2 \mathbb{E} \left[\frac{1}{(\lambda_{t,\min}^{\mathbb{D}})^2} \right] + \zeta \mathbb{E} \left[\text{tr} \left(P_{\mathbb{U}}(V_t^{-1}) V_t^{-1} \left(\sum_{i=1}^t X_i X_i^\top \right) \right) \right] \\
&\leq \lambda^2 S^2 \mathbb{E} \left[\frac{1}{(\lambda_{t,\min}^{\mathbb{D}})^2} \right] + \zeta \mathbb{E} [d \lambda_{\max}(P_{\mathbb{U}}(V_t^{-1}))] \\
&\leq \lambda^2 S^2 \mathbb{E} \left[\frac{1}{(\lambda_{t,\min}^{\mathbb{D}})^2} \right] + d \zeta \mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}}} \right]
\end{aligned}$$

$$\leq \lambda^2 S^2 \mathbb{E} \left[\frac{1}{(\lambda_{t,\min}^{\mathbb{D}_k})^2} \right] + d\zeta \mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \right] \quad (5.43)$$

where the second inequality has used the condition that η_t are i.i.d. and the standard result that $\mathbb{E}[\eta_t^2] \leq \zeta$ where ζ is a constant depending on ϑ since η_t has the zero-mean sub-Gaussian distribution with parameter ϑ .

Using the similar but tedious argument as that in the proof of Theorem 5.2, we can show

$$\mathbb{E} \left[\frac{1}{\lambda_{t,\min}^{\mathbb{D}_k}} \right] = O(t^{-1/2}), \text{ and } \mathbb{E} \left[\frac{1}{(\lambda_{t,\min}^{\mathbb{D}_k})^2} \right] = O(t^{-1}). \quad (5.44)$$

Finally, using (5.28) and (5.43), we have

$$\begin{aligned} & \mathbb{E} [R_{\mathbb{U}}(n)] \\ & \leq \sum_{t=1}^n 2ZS\epsilon_t^s + \mathbb{E} [\langle P_{\mathbb{U}}(X_{\mathbb{U}}^*) - P_{\mathbb{U}}(X_t), P_{\mathbb{U}}(\boldsymbol{\theta}) \rangle \mid E_t^c] \\ & \leq \sum_{t=1}^n O(t^{-1/2}) \\ & = O(n^{1/2}). \end{aligned} \quad (5.45)$$

The proof of Theorem 5.3 is now complete.

Chapter 6

Conclusion and Future Work

In Chapter 3, we have developed sequential question design strategies for crowdsourcing in which workers' responses are used to determine the hidden state of a multi-class labeling problem. Our strategies allow the crowdsourcer to sequentially determine the question to pose to the workers, based on their previous responses. We formulated the sequential design as a POMDP, which maximizes a weighted function of the probability of inferring the correct hidden state after fusing the workers' responses, and the cost of posing additional questions, subject to a budget constraint for the crowdsourcer. To overcome the intractability of the POMDP, we have proposed question design approaches based on the Ulam-Rényi game. We have demonstrated through simulations that when the number of hidden states is relatively large, our approaches achieve good performance, with less computational complexity when compared with traditional POMDP solvers.

In Chapter 4, we have formulated the sequential task recommendation problem in crowdsourcing as a MAB problem in which the reward is unobservable. To learn the worker's preferences and reliabilities for different task categories, we have included gold tasks whose solutions are known *a priori*, but which however do not produce any rewards. We showed that the optimal regret order is $O(\sqrt{n})$, where n is the number of time steps. We proposed three strategies and showed that all

achieve the optimal regret order. Simulations verify the efficiency of our proposed strategies.

In Chapter 5, we have formulated sequential non-discriminatory recommendations as the orthogonal projection problem in the linear stochastic bandit model, where the objective is to maximize the cumulative projection reward over a subspace of decision attributes. We have developed a strategy and showed that it achieves sub-linear projection regret for the finite- and infinite-arm cases. Experiments verify the efficiency of our strategy. In this chapter, we have assumed that the target subspace is known beforehand when decomposing the reward. However, in some real scenarios, it is not the case and we should learn how to construct a reasonable subspace to perform the orthogonal projection jointly through sequential recommendations.

Sequential multi-class labeling problem can also be modeled and analyzed in an information theoretic rate-distortion framework. In [124], the authors identified the ultimate fidelity that one can achieve by any form of query from the crowdsourcer and any decoding (inference) algorithm with a given budget under the no feedback assumption. It is interesting to derive fundamental limits under specific sequential (feedback) settings.

List of Author's Publications

Journal Articles

- **Q. Kang** and W. P. Tay. Sequential multi-class labeling in crowdsourcing. *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 11, pp. 2190 – 2199, Nov. 2019.

Conference Proceedings

- **Q. Kang** and W. P. Tay, "Sequential multi-class labeling in crowdsourcing: A Ulam-Renyi game approach," in *Proc. Int. Conf. Web Intell.*, Leipzig, Germany, Aug. 2017, pp. 245–251
- **Q. Kang** and W. P. Tay, "Orthogonal projection in linear bandits," in *Proc. IEEE Global Conf. Signal Inform. Process.*, Ottawa, Canada, Nov. 2019

Journals under Submission/Revision

- **Q. Kang** and W. P. Tay, "Task recommendation in crowdsourcing based on learning preferences and reliabilitie," submitted to *IEEE Trans. Serv. Comput.*
- **Q. Kang** and W. P. Tay, "Learning orthogonal projections in linear bandits," submitted to *IEEE Trans. Signal Process.*

Bibliography

- [1] Amazon Mechanical Turk (MTurk) Website. Accessed on: Jun. 25, 2019. [Online]. Available: <https://www.mturk.com/> 2, 55
- [2] Figure Eight Company. Accessed on: Jun. 25, 2019. [Online]. Available: <https://www.figure-eight.com/> 2, 55
- [3] eYeka Website. Accessed on: Jun. 25, 2019. [Online]. Available: <https://en.eyeka.com/> 2
- [4] INNOCENTIVE Website. Accessed on: Jun. 25, 2019. [Online]. Available: <https://www.innocentive.com/> 2
- [5] M. L. Gray and S. Suri, *How to Stop Silicon Valley from Building a New Global Underclass*. New York, USA: Houghton Mifflin Harcourt, 2019. 2, 4
- [6] D. R. Karger, S. Oh, and D. Shah, “Iterative learning for reliable crowdsourcing systems,” in *Proc. Advances in Neural Inform. Process. Syst.*, Granada, Spain, Dec. 2011, pp. 1953–1961. 2, 27
- [7] A. Vempaty, L. R. Varshney, and P. K. Varshney, “Reliable crowdsourcing for multi-class labeling using coding theory,” *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 4, pp. 667–679, Aug. 2014. 2, 3, 27, 28, 32, 34
- [8] D. R. Karger, S. Oh, and D. Shah, “Budget-optimal task allocation for reliable crowdsourcing systems,” *Oper. Res.*, vol. 62, no. 1, pp. 1–24, Feb. 2014. 2, 27, 56

- [9] A. Khetan and S. Oh, “Reliable crowdsourcing under the generalized Dwid-Skene model,” *arXiv preprint arXiv:1602.03481*, 2016. 2, 28, 56
- [10] D. R. Karger, S. Oh, and D. Shah, “Efficient crowdsourcing for multi-class labeling,” in *Proc. ACM Inter. Conf. Measurement Modeling Comput. Syst.*, Pittsburgh, USA, Jun. 2013, pp. 81–92. 2, 28, 56
- [11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, May 2015. 3
- [12] V. Grewal-Carr and C. Bates. The three billion enterprise crowdsourcing and the growing fragmentation of work. Accessed on: Jun. 25, 2018. [Online]. Available: <https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/Innovation/deloitte-uk-crowdsourcing-pov.pdf> 3
- [13] The size of the mechanical turk marketplace. Accessed: Jun. 25, 2019. [Online]. Available: <https://www.pewresearch.org/internet/2016/07/11/the-size-of-the-mechanical-turk-marketplace/> 4
- [14] Y. Chen and X. Yin, “Stable job assignment for crowdsourcing,” in *Proc. IEEE Global Commun. Conf.*, Singapore, Dec. 2017, pp. 51–56. 4, 58
- [15] M. Safran and D. Che, “Real-time recommendation algorithms for crowdsourcing systems,” *Appl. Comput. Inform.*, vol. 13, no. 1, pp. 47–56, Jan. 2017. 4, 57, 58
- [16] R. M. Bell and Y. Koren, “Lessons from the netflix prize challenge,” *ACM SIGKDD Explorations Newslett.*, vol. 9, no. 2, pp. 75–79, Dec. 2007. 5, 83
- [17] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009. 5, 6, 83, 85
- [18] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proc. Int. Conf. World Wide Web*, New York, USA, Apr. 2010, pp. 661–670. 5, 84, 86

- [19] L. Sweeney, “Discrimination in online ad delivery,” *arXiv preprint arXiv:1301.6822*, 2013. 5, 84
- [20] A. Matamoros-Fernández, “Platformed racism: The mediation and circulation of an australian race-based controversy on twitter, facebook and youtube,” *Inform. Commun. Soc.*, vol. 20, no. 6, pp. 930–946, Feb. 2017. 5, 84
- [21] Why netflix features black actors in promos to black users. Accessed: Jun. 25, 2019. [Online]. Available: <https://www.wired.com/story/why-netflix-features-black-actors-promos-to-black-users> 5, 84
- [22] K. D. Johnson, D. P. Foster, and R. A. Stine, “Impartial predictive modeling: Ensuring fairness in arbitrary models,” *arXiv preprint arXiv:1608.00528*, 2016. 5, 86
- [23] K. P. Murphy, *Machine learning: a probabilistic perspective*. London, UK: MIT Press, 2012. 5, 86
- [24] T. Kamishima and S. Akaho, “Considerations on recommendation independence for a find-good-items task,” in *Proc. Workshop Responsible Recommendation*, Como, Italy, Aug. 2017, pp. 1–6. 6, 85
- [25] Z. Zhu, X. Hu, and J. Caverlee, “Fairness-aware tensor-based recommendation,” in *Proc ACM Inter. Conf. Inform. Knowledge Manage.*, Torino, Italy, Oct. 2018, pp. 1153–1162. 6, 85, 86
- [26] S. Hajian, F. Bonchi, and C. Castillo, “Algorithmic bias: From discrimination discovery to fairness-aware data mining,” in *Proc. ACM Int. Conf. Knowledge Discovery and Data mining*, California, USA, Aug. 2016, pp. 2125–2126. 6, 85
- [27] J. Pineau, G. Gordon, S. Thrun *et al.*, “Point-based value iteration: An anytime algorithm for pomdps,” in *Proc. Int. Joint Conf. Artificial Intell.*, Acapulco, Mexico, Aug. 2003, pp. 1025–1032. 7, 8, 14, 15, 16, 29, 36, 43, 44

- [28] J. Pineau, G. Gordon, and S. Thrun, “Anytime point-based approximations for large pomdps,” *J. Artificial Intell. Res.*, vol. 27, pp. 335–380, Nov. 2006. [7](#), [8](#), [14](#), [29](#), [36](#), [43](#)
- [29] D. Silver and J. Veness, “Monte-carlo planning in large pomdps,” in *Proc. Advances Neural Inform. Process. Syst.*, Vancouver, Canada, Dec. 2010, pp. 2164–2172. [7](#), [8](#), [14](#), [16](#), [17](#), [29](#), [36](#), [43](#), [44](#)
- [30] F. Cicalese and U. Vaccaro, “An improved heuristic for ‘Ulam–Rényi game’,” *Inform. Process. Lett.*, vol. 73, no. 3, pp. 119–124, Feb. 2000. [7](#), [8](#), [23](#), [29](#), [36](#), [37](#), [40](#)
- [31] E. L. Lawler and S. Sarkissian, “An algorithm for ‘Ulam’s Game’ and its application to error correcting codes,” *Inform. Process. Lett.*, vol. 56, no. 2, pp. 89–93, Oct. 1995. [7](#), [8](#), [23](#), [29](#), [36](#)
- [32] A. Pelc, “Searching games with errors—fifty years of coping with liars,” *Theoretical Comput. Sci.*, vol. 270, no. 1, pp. 71–109, Jan. 2002. [7](#), [23](#), [29](#), [36](#)
- [33] D. Geiger, S. Seedorf, T. Schulze, R. C. Nickerson, and M. Schader, “Managing the crowd: Towards a taxonomy of crowdsourcing processes,” in *Proc. Amer. Conf. Inform. Syst.*, Michigan, USA, Aug. 2011, [Online]. Available: http://aisel.aisnet.org/amcis2011_submissions/430/. [8](#), [58](#)
- [34] N. Kaufmann, T. Schulze, and D. Veit, “More than fun and money. Worker motivation in crowdsourcing – A study on mechanical turk,” in *Proc. Amer. Conf. Inform. Syst.*, Michigan, USA, Aug. 2011, [Online]. Available: http://aisel.aisnet.org/amcis2011_submissions/340/. [8](#), [58](#)
- [35] J. Jarrett, M. B. Blake, and I. Saleh, “Crowdsourcing, mixed elastic systems and human-enhanced computing—a survey,” *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 202–214, May 2018. [8](#), [58](#)
- [36] C.-J. Ho, S. Jabbari, and J. W. Vaughan, “Adaptive task assignment for crowdsourced classification,” in *Proc. Int. Conf. Mach. Learning*, Georgia, USA, Jun. 2013, pp. 534–542. [9](#), [56](#), [59](#)

- [37] H. Li and B. Yu, “Error rate bounds and iterative weighted majority voting for crowdsourcing,” *arXiv preprint arXiv:1411.4086*, 2014. [9](#), [59](#)
- [38] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intell.*, vol. 101, no. 1, pp. 99–134, May. 1998. [12](#)
- [39] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, “Online planning algorithms for pomdps,” *J. Artificial Intell. Res.*, vol. 32, pp. 663–704, Jul. 2008. [13](#), [14](#), [29](#), [44](#)
- [40] R. D. Smallwood and E. J. Sondik, “The optimal control of partially observable markov processes over a finite horizon,” *Oper. Res.*, vol. 21, no. 5, pp. 1071–1088, Sept. 1973. [14](#), [15](#), [29](#)
- [41] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in *Proc. Eur. Conf. Mach. Learn.*, Berlin, Germany, Sep. 2006, pp. 282–293. [16](#)
- [42] T. L. Lai and H. Robbins, “Asymptotically efficient adaptive allocation rules,” *Advances Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985. [17](#), [18](#), [19](#), [20](#), [58](#)
- [43] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Mach. Learning*, vol. 47, no. 2, pp. 235–256, May 2002. [17](#), [18](#), [20](#), [58](#), [63](#), [75](#), [79](#), [80](#), [87](#), [94](#), [108](#)
- [44] J. C. Gittins, “Bandit processes and dynamic allocation indices,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 2, pp. 148–164, Feb. 1979. [18](#)
- [45] A. Mahajan and D. Teneketzis, “Multi-armed bandit problems,” in *Foundations Applications Sensor Management*, 2008, pp. 121–151. [18](#)
- [46] H. Robbins, “Some aspects of the sequential design of experiments,” *Bull. Amer. Math. Soc.*, vol. 58, no. 5, pp. 527–535, Jul. 1952. [18](#)

- [47] J.-Y. Audibert, R. Munos, and C. Szepesvári, “Exploration–exploitation tradeoff using variance estimates in multi-armed bandits,” *Theoretical Comput. Sci.*, vol. 410, no. 19, pp. 1876–1902, Oct. 2009. [18](#), [21](#), [78](#)
- [48] R. Agrawal, “Sample mean based index policies by $O(\log n)$ regret for the multi-armed bandit problem,” *Advances Appl. Probability*, vol. 27, no. 4, pp. 1054–1078, Dec. 1995. [18](#)
- [49] A. N. Burnetas and M. N. Katehakis, “Optimal adaptive policies for sequential allocation problems,” *Advances Appl. Math.*, vol. 17, no. 2, pp. 122–142, Jun. 1996. [18](#)
- [50] S. Pandey, D. Chakrabarti, and D. Agarwal, “Multi-armed bandit problems with dependent arms,” in *Proc. Int. Conf. Mach. Learn.*, Oregon, USA, Jun. 2007, pp. 721–728. [21](#)
- [51] E. L. Presman, Sonin, I. N, and Sonin, *Sequential Control with Incomplete Information*. London, UK: Academic Press, 1990. [21](#)
- [52] A. Goldenshluger, A. Zeevi *et al.*, “Woodroffe’s one-armed bandit problem revisited,” *Ann. Appl. Probability*, vol. 19, no. 4, pp. 1603–1633, Nov. 2009. [21](#)
- [53] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, “Improved algorithms for linear stochastic bandits,” in *Proc. Advances Neural Inf. Process. Syst.*, Granada, Spain, Dec. 2011, pp. 2312–2320. [21](#), [22](#), [23](#), [87](#), [92](#), [106](#)
- [54] P. Rusmevichientong and J. N. Tsitsiklis, “Linearly parameterized bandits,” *Math. Oper. Res.*, vol. 35, no. 2, pp. 395–411, May 2010. [21](#), [22](#), [23](#), [87](#), [92](#), [96](#), [97](#), [115](#)
- [55] S. Agrawal and N. Goyal, “Thompson sampling for contextual bandits with linear payoffs,” in *Proc. Int. Conf. Mach. Learn.*, Georgia, USA, Jun. 2013, pp. 127–135. [21](#)

- [56] V. Dani, T. P. Hayes, and S. M. Kakade, “Stochastic linear optimization under bandit feedback,” in *Proc. Annu. Conf. Learn. Theory*, Helsinki, Finland, Jul. 2008, pp. 355–366. [22](#), [92](#), [93](#)
- [57] E. R. Berlekamp, “Block coding with noiseless feedback,” Ph.D. dissertation, Massachusetts Institute of Technology, 1964. [23](#)
- [58] F. Cicalese and C. Deppe, “Perfect minimally adaptive q-ary search with unreliable tests,” *J. Stat. Plan. Inference*, vol. 137, no. 1, pp. 162–175, Jan. 2007. [24](#), [40](#)
- [59] S. Muthukrishnan, “On optimal strategies for searching in presence of errors,” in *Proc. ACM-SIAM Symp. Discrete Algorithms*, Arlington, USA, Jan. 1994, pp. 680–689. [24](#), [25](#), [37](#), [40](#)
- [60] F. Cicalese, *Fault-Tolerant Search Algorithms: Reliable Computation with Unreliable Information*. New York, USA: Springer, 2013. [25](#)
- [61] J. Spencer, “Ulam’s searching game with a fixed number of lies,” *Theoretical Comput. Sci.*, vol. 95, no. 2, pp. 307–321, Mar. 1992. [25](#)
- [62] J. Bragg, Mausam, and D. S. Weld, “Crowdsourcing multi-label classification for taxonomy creation,” in *Proc. AAAI Conf. Hum. Comput. and Crowdsourcing*, California, USA, Nov. 2013, pp. 25–33. [27](#)
- [63] R. Snow, B. O’Connor, D. Jurafsky, and A. Y. Ng, “Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks,” in *Proc. Conf. Empir. Meth. Natural Lang. Process.*, Hawaii, USA, Oct. 2008, pp. 254–263. [27](#)
- [64] J. Ross, L. Irani, M. Silberman, A. Zaldivar, and B. Tomlinson, “Who are the crowdworkers? Shifting demographics in Mechanical Turk,” in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Atlanta, Georgia, USA, Apr. 2010, pp. 2863–2872. [27](#)

- [65] L. R. Varshney, “Privacy and reliability in crowdsourcing service delivery,” in *Proc. Annu. SRII Global Conf.*, San Jose, USA, Jul. 2012, pp. 55–60. [27](#)
- [66] D. Zhou, Q. Liu, J. C. Platt, C. Meek, and N. B. Shah, “Regularized minimax conditional entropy for crowdsourcing,” *arXiv preprint arXiv:1503.07240*, 2015. [28](#), [56](#)
- [67] D. Oleson, A. Sorokin, G. P. Laughlin, V. Hester, J. Le, and L. Biewald, “Programmatic gold: Targeted and scalable quality assurance in crowdsourcing,” in *Proc. AAAI Conf. Artificial Intell.*, San Francisco, USA, Aug. 2011, pp. 43–48. [28](#), [58](#)
- [68] A. Singla, I. Bogunovic, G. Bartók, A. Karbasi, and A. Krause, “Near-optimally teaching the crowd to classify,” in *Inter. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 154–162. [28](#)
- [69] J. Sun and H. Ma, “A behavior-based incentive mechanism for crowd sensing with budget constraints,” in *Proc. IEEE Int. Conf. Commun.*, Sydney, Australia, Jun. 2014, pp. 1314–1319. [28](#)
- [70] D. Zhao, X.-Y. Li, and H. Ma, “How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint,” in *Proc. IEEE Int. Conf. Comput. Commun.*, Toronto, Canada, Apr. 2014, pp. 1213–1221. [28](#)
- [71] C. Yao, P.-N. Chen, T.-Y. Wang, Y. S. Han, and P. K. Varshney, “Performance analysis and code design for minimum Hamming distance fusion in wireless sensor networks,” *IEEE Trans. Inf. Theory*, vol. 53, no. 5, pp. 1716–1734, Apr. 2007. [28](#), [33](#)
- [72] T.-Y. Wang, Y. S. Han, P. K. Varshney, and P.-N. Chen, “Distributed fault-tolerant classification in wireless sensor networks,” *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 724–734, Apr. 2005. [28](#)

- [73] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie, “Visual recognition with humans in the loop,” in *Proc. Eur. Conf. Comput. Vision*, Crete, Greece, Sep. 2010, pp. 438–451. [32](#), [43](#)
- [74] E. Zhou, “Optimal stopping under partial observation: Near-value iteration,” *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 500–506, Feb. 2012. [36](#)
- [75] K. M. Seiler, H. Kurniawati, and S. P. Singh, “An online and approximate solver for pomdps with continuous action space,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, USA, May 2015, pp. 2290–2297. [36](#)
- [76] J. Van den Berg, S. Patil, and R. Alterovitz, “Efficient approximate value iteration for continuous gaussian pomdps,” in *Proc. AAAI Conf. Artificial Intell.*, Toronto, Canada, Jul. 2012, pp. 1832–1836. [36](#)
- [77] H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis, “Global motion planning under uncertain motion, sensing, and environment map,” *Auton. Robot.*, vol. 33, no. 3, pp. 255–272, Oct. 2012. [36](#)
- [78] A. H. Land and A. G. Doig, “An automatic method of solving discrete programming problems,” *Econometrica: J. Econometric Soc.*, vol. 28, no. 3, pp. 497–520, Jul. 1960. [42](#)
- [79] E. L. Lawler and D. E. Wood, “Branch-and-bound methods: A survey,” *Oper. Res.*, vol. 14, no. 4, pp. 699–719, Feb. 1966. [42](#)
- [80] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen, “A survey of very large-scale neighborhood search techniques,” *Discrete Appl. Math.*, vol. 123, no. 1-3, pp. 75–102, Nov. 2002. [42](#)
- [81] F. Basık, B. Gedik, H. Ferhatosmanoglu, and K.-L. Wu, “Fair task allocation in crowdsourced delivery,” *IEEE Trans. Services Comput.*, 2018, to be published. doi: 10.1109/TSC.2018.2854866. [56](#)

- [82] J. Shu, X. Jia, K. YANG, and H. Wang, “Privacy-preserving task recommendation services for crowdsourcing,” *IEEE Trans. Services Comput.*, 2018, to be published. doi: 10.1109/TSC.2018.2791601. [56](#)
- [83] O. Cappé and E. Moulines, “Online EM algorithm for latent data models,” *J. Royal Statist. Soc.: Series B (Statist. Methodology)*, vol. 71, no. 3, pp. 593–613, Feb. 2009. [56](#)
- [84] Q. Kang and W. P. Tay, “Sequential multi-class labeling in crowdsourcing: A Ulam-Renyi game approach,” in *Proc. Int. Conf. Web Intell.*, Leipzig, Germany, Aug. 2017, pp. 245–251. [56](#)
- [85] —, “Sequential multi-class labeling in crowdsourcing,” *IEEE Trans. Knowl. Data Eng.*, 2018, to be published. doi: 10.1109/TKDE.2018.2874003. [56](#)
- [86] L. Massoulié, M. I. Ohannessian, and A. Proutière, “Greedy-Bayes for targeted news dissemination,” in *Proc. Int. Conf. Meas. Modelling Comput. Syst.*, Portland, USA, May. 2015, pp. 285–296. [56](#)
- [87] C.-J. Ho and J. W. Vaughan, “Online task assignment in crowdsourcing markets,” in *Proc. AAAI Conf. Artificial Intell.*, Toronto, Canada, Jul. 2012, pp. 45–51. [56](#)
- [88] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings, “Efficient crowdsourcing of unknown experts using bounded multi-armed bandits,” *Artificial Intell.*, vol. 214, no. 1, pp. 89–111, May 2014. [57](#)
- [89] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack problems. 2004.* Berlin, DE: Springer, 2003. [57](#)
- [90] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, “Online mobile micro-task allocation in spatial crowdsourcing,” in *Proc. IEEE Int. Conf. Data. Eng.*, Helsinki, Finland, May 2016, pp. 49–60. [57](#)

- [91] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, “iCrowd: An adaptive crowd-sourcing framework,” in *Proc. SIGMOD Int. Conf. Manage. Data*, Victoria, Australia, May 2015, pp. 1015–1030. [57](#)
- [92] S. K. Müller, C. Tekin, M. van der Schaar, and A. Klein, “Context-aware hierarchical online learning for performance maximization in mobile crowd-sourcing,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1334–1347, May 2018. [57](#)
- [93] Y. Liu and M. Liu, “An online learning approach to improving the quality of crowd-sourcing,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2166–2179, Aug. 2017. [57](#)
- [94] M.-C. Yuen, I. King, and K.-S. Leung, “Task recommendation in crowdsourcing systems,” in *Proc. Inter. Workshop Crowdsourcing and Data Mining*, Beijing, China, Aug. 2012, pp. 22–26. [58](#)
- [95] ———, “TaskRec: A task recommendation framework in crowdsourcing systems,” *Neural Process. Lett.*, vol. 41, no. 2, pp. 223–238, Feb. 2015. [58](#)
- [96] A. Kittur, E. H. Chi, and B. Suh, “Crowdsourcing user studies with Mechanical Turk,” in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, Florence, Italy, Apr. 2008, pp. 453–456. [58](#)
- [97] J. Le, A. Edmonds, V. Hester, and L. Biewald, “Ensuring quality in crowd-sourced search relevance evaluation: The effects of training question distribution,” in *Proc. SIGIR Workshop Crowdsourcing Search Evaluation*, Geneva, Switzerland, Jul. 2010, pp. 17–20. [58](#)
- [98] S. Vakili and Q. Zhao, “Risk-averse multi-armed bandit problems under mean-variance measure,” *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 6, pp. 1093–1111, Sep. 2016. [59](#)
- [99] ———, “Mean-variance and value at risk in multi-armed bandit problems,” in *Proc. Annu. Allerton Conf. Commun, Control, and Comput.*, Monticello, USA, Sep. 2015, pp. 1330–1335. [59](#)

- [100] A. Sani, A. Lazaric, and R. Munos, “Risk-aversion in multi-armed bandits,” in *Proc. Advances Neural Inf. Process. Syst.*, California, USA, Dec. 2012, pp. 3275–3283. [59](#)
- [101] M. C. Steinbach, “Markowitz revisited: Mean-variance models in financial portfolio analysis,” *SIAM review*, vol. 43, no. 1, pp. 31–85, Feb. 2001. [59](#)
- [102] H. Markowitz, “Portfolio selection,” *J. Finance*, vol. 7, no. 1, pp. 77–91, Mar. 1952. [59](#)
- [103] L. Tran-Thanh, A. Chapman, J. E. Munoz De Cote Flores Luna, A. Rogers, and N. R. Jennings, “ ϵ -First policies for budget-limited multi-armed bandits,” in *Proc. AAAI Conf. Artificial Intell.*, Georgia, USA, Jul. 2010, pp. 1211–1216. [65](#)
- [104] H. Wang, Q. Wu, and H. Wang, “Learning hidden features for contextual bandits,” in *Proc. Int. Conf. Inform. Knowl. Manag.*, Indianapolis, USA, Oct. 2016, pp. 1633–1642. [84](#), [91](#), [92](#)
- [105] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, “Efficiency improvement of neutrality-enhanced recommendation.” in *Proc. Recommender Syst.*, Hong Kong, China, Oct. 2013, pp. 1–8. [85](#)
- [106] T. Calders and S. Verwer, “Three naive bayes approaches for discrimination-free classification,” *Data Mining Knowledge Discovery*, vol. 21, no. 2, pp. 277–292, Jul. 2010. [85](#)
- [107] T. Calders, A. Karim, F. Kamiran, W. Ali, and X. Zhang, “Controlling attribute effect in linear regression,” in *Proc. Int. Conf. Data Mining*, Dallas, USA, Dec. 2013, pp. 71–80. [86](#), [90](#)
- [108] I. Zliobaite, “A survey on measuring indirect discrimination in machine learning,” *arXiv preprint arXiv:1511.00148*, 2015. [86](#)

- [109] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, “Modeling wine preferences by data mining from physicochemical properties,” *Decision Support Syst.*, vol. 47, no. 4, pp. 547–553, May 2009. [87](#), [97](#), [103](#)
- [110] S. J. Leon, *Linear Algebra with Applications*. Upper Saddle River, NJ: Pearson, 2010. [89](#)
- [111] P. Auer, “Using confidence bounds for exploitation-exploration trade-offs,” *J. Mach. Learn. Res.*, vol. 3, no. 11, pp. 397–422, Nov. 2002. [92](#)
- [112] A. Uschmajew, “Local convergence of the alternating least squares algorithm for canonical tensor approximation,” *J. Matrix Anal. Appl.*, vol. 33, no. 2, pp. 639–652, Jun. 2012. [92](#)
- [113] J. Friedman, T. Hastie, and R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent,” *J. Statist. Softw.*, vol. 33, no. 1, pp. 1–22, Aug. 2010. [92](#)
- [114] M. M. Drugan, A. Nowé, and B. Manderick, “Pareto upper confidence bounds algorithms: an empirical study,” in *Proc. IEEE Symp. Adaptive Dynamic Programming Reinforcement Learn.*, Orlando, USA, Dec. 2014, pp. 1–8. [92](#)
- [115] M. M. Drugan and A. Nowe, “Designing multi-objective multi-armed bandits algorithms: A study,” in *Proc. Int. Joint Conf. Neural Netw.*, Dallas, USA, Aug 2013, pp. 1–8. [92](#), [93](#)
- [116] S. Q. Yahyaa, M. M. Drugan, and B. Manderick, “Annealing-pareto multi-objective multi-armed bandit algorithm,” in *Proc. IEEE Symp. Adaptive Dynamic Programming Reinforcement Learn.*, Orlando, USA, Dec. 2014, pp. 1–8. [92](#)
- [117] —, “The scalarized multi-objective multi-armed bandit problem: An empirical study of its exploration vs. exploitation tradeoff,” in *Proc. Int. Joint Conf. Neural Netw.*, Beijing, China, Jul. 2014, pp. 2290–2297. [92](#)

-
- [118] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Da Fonseca Grunert, “Performance assessment of multiobjective optimizers: An analysis and review,” *IEEE Trans. Evol. Comput.*, vol. 139, no. 2, pp. 117–132, May 2002. [92](#)
- [119] S. S. Villar, J. Bowden, and J. Wason, “Multi-armed bandit models for the optimal design of clinical trials: Benefits and challenges,” *Statist. Sci.*, vol. 30, no. 2, p. 199, May 2015. [93](#)
- [120] V. Kuleshov and D. Precup, “Algorithms for multi-armed bandit problems,” *arXiv preprint arXiv:1402.6028*, 2014. [93](#)
- [121] R. D. Toto, “Treatment of hypertension in chronic kidney disease,” *Seminars Nephrology*, vol. 25, no. 6, pp. 435–439, Nov. 2005. [93](#)
- [122] E. S. Polovinkin, “Strongly convex analysis,” *Sbornik: Math.*, vol. 187, no. 2, p. 259, Feb. 1996. [96](#)
- [123] D. Pollard, *Convergence of Stochastic Processes*. Berlin, DE: Springer, 1984. [108](#)
- [124] F. Lahouti and B. Hassibi, “Fundamental limits of budget-fidelity trade-off in label crowdsourcing,” in *Proc. Advances Neural Inform. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 5058–5066. [118](#)