Bandits and Preference Learning

By
ANIRUDDHA BHARGAVA

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY
(ELECTRICAL AND COMPUTER ENGINEERING)

at the
UNIVERSITY *of* WISCONSIN-MADISON
2017

Date of final oral examination: November 27, 2017.
The dissertation is approved by the following members of the Final Oral Committee:
Robert Nowak, Professor, Electrical and Computer Engineering,
William Sethares, Professor, Electrical and Computer Engineering,
Rebecca Willett, Professor, Electrical and Computer Engineering,
Stephen Wright, Professor, Computer Science,
Xiaojin (Jerry) Zhu, Professor, Computer Science.

The great Lord is boundless; the universe which comes from Her is also boundless.
Still the essence of creation continues to be boundless.

- Upanishads

Abstract

The internet revolution has brought a large population access to a vast array of information since the mid 1990s. More recently, with the advent of smartphones, it has become an essential part of our everyday life. This has lead to, among many other developments, the personalization of the online experience with great benefits to all involved. Companies have particular interest in showing products and advertisements that match what particular users are looking for, and users desire getting personalized recommendations from internet for entertainment and consumer goods that suit them as individuals. In machine learning, this is popularly achieved using the theory of the multi-armed bandits, methods which allow us to zero in on the consumer's personal preferences.

The last few decades have seen great advances in the theory and practice of multi-armed bandits exploiting either the context of the user, the context of the objects, or both. Great theoretical improvements have brought algorithms' performance close to their theoretical optimal. However, various challenges exist in the practical use of multi-armed bandits. In this thesis, we explore some of these challenges and endeavor to overcome them. First, we examine how multiple populations can be catered to simultaneously. We then address the issue of scaling multi-armed bandits to situations where there are many arms. We also look at how to incorporate generalized linear reward models while maintaining computational efficiency. Finally, we address how we can use feature feedback to focus the bandits exploration to a limited subset of features. This leads to algorithms that are still tractable for high-dimensional datasets where the preferences of the user are explained by a sparse subset of them.

Acknowledgements

My parents Vasumathi and Jamadagni are one of the biggest reasons I got into research from an early age. They instilled a spirit of curiosity and did all they could to put me in the best position for my education. My aunt Saraswati was pivotal in the upbringing too: shuttling me back and forth from schools. My wife Emily has been key to keeping me on track and helping me work hard. She has been a great companion through the last few years, having been there herself. To them, I owe a big debt of gratitude.

I would like to thank my advisor Rob. He gave me a chance to come to Wisconsin and allowed me to find my way in the world of research. He has been there to help me grow as a researcher and as a person. In my research journey I would like to thank other professors who have played a key role: Rebecca Willett, Jerry Zhu, Steve Wright, and many other.

Thanks to my co-authors Ravi Ganti and Kwang-Sung Jun for being great people and friends to work with on projects. I would like to thank Scott Beddia, Lalit Jain, Matt Malloy, Xin Hunt, Kevin Jamieson, John Ehrmanntraut, Urvashi Oswal, Blake Mason, Nikhil Rao, Gautam Dasarathy and many other. My friends have kept me grounded and provided me with constant support.

Thanks to the Kohler fellows for being great people to talk to and for being an outlet for creative ideas. I would like to thank the Madison community in general for allowing me to enjoy so many valuable moments. Finally, thanks to all the great forces that bring us together and to allow us to dig deep on research projects.

Contents

Chapter 1

Introduction

Multi-armed bandits first arose in the 1930s with the study of clinical trials (Thompson, 1933).), which required a small amount of data to be used as a guide to act on the effective collection of further data. Then, as it is now, this is a problem of sequential decision making, a problem that is often encountered in the modern internet.

In general, a multi-armed bandit problem consists of a set of possible actions, that we call the arms. At each time, one must propose an action. Then one gets to observe the reward for that action only. Based upon this information, one decides upon a next action to take, and the series continues.

The name multi-armed bandit itself comes from a term used for slot machines. Slot machines are, in fact, a great example for the concept of a bandit problem. Consider having $100 to spend on slot machines at a casino. In a room full of slot machines, there may be some that give a higher reward than others. Suppose you must pay 1 ¢to try a particular slot machine. The question addressed by bandit algorithms is this: how should we decide to spend the money? On one hand, we have to try different machines to estimate their rewards (exploration) in order to identify the best machines, and among those machines we have found empirically to be best, we want to spend the

remaining money to get the best chance of getting a maximal payoff (exploitation). All bandit algorithms trade off these two factors, exploration and exploitation, in order to seek an optimum payout.

This thesis will focus on the particular problem of contextual bandits in a stochastic setting. Contextual refers to the assumption that for each action or arm, we also get to observe the context or features associated with it. A stochastic setting refers to the assumption that we observe a noisy reward, where the outcome contains noise which is random and independent of the choice of the arm.

In stochastic multi-armed bandits, two objectives are often important: finding the best action, and minimizing the regret. In the first case the goal is clear: to find the best possible action with as few queries as possible. In the second case, minimizing regret, regret refers to the difference in rewards between the action we actually took and best possible action we could have taken. This thesis focuses on the problem of minimizing regret.

Work in this particular area of multi-armed bandits can be traced back to (Auer and Long, 2002) who proposed the linear-bandit model. Further seminal work in this area was done by (Dani et al., 2008), (Rusmevichientong and Tsitsiklis, 2010) and (Abbasi-Yadkori et al., 2011) leading to algorithms whose regrets scale with the square-root of time.

While this area of research started with the study of clinical trials, it has a multitude of applications today in many areas. For example, the spread of the internet has led to personalization of the information that we are exposed to. Netflix recommends movies based on our previous viewing profiles, Google personalizes search results based on our browsing history and physical location, Amazon recommends products based on our shopping history. The list goes on.

One key consideration presented in this thesis is how to make the most of human feedback in the process of bandit design. We propose the use of a mathematical struc-

ture to aid in aggregating information from multiple populations. We also propose modifications for existing algorithms in order to obtain better results. An example of such solicitation of human feedback can be seen in Figure 1.1.

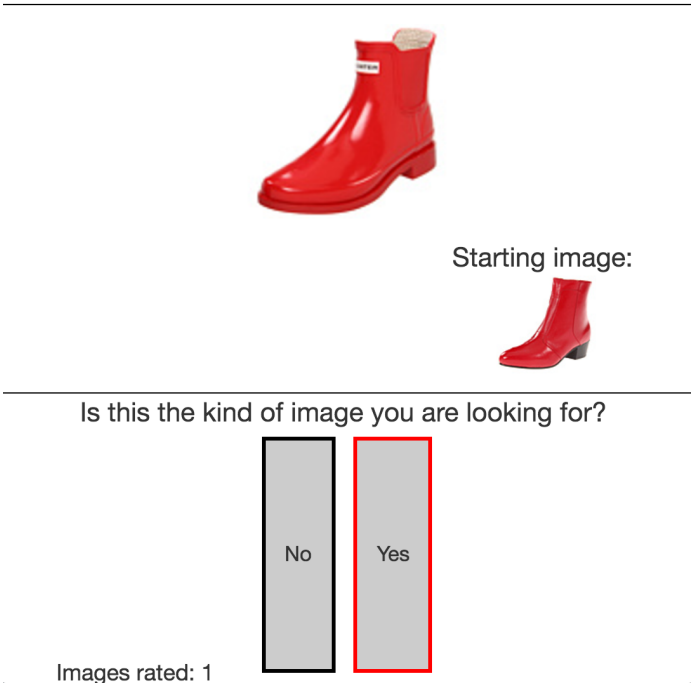


Figure 1.1: Example of reward in bandit setting for image search. We would like to propose as many images as possible that leads to the users liking them.

While bandits have comprised an important component in the enabling of personalized interfaces, many challenges exist in their use on a large scale. Critically, it is computationally expensive to run these algorithms. This thesis will address some of the issues that hinder a wider use of multi-armed bandits by exploiting the structure of multiple populations, enabling the use of hashing, and by making use of user feedback.

Below are some of the big themes addressed in this thesis.

1. **Estimating the preference of multiple-populations blindly:** (Chapter 2). This chapter addresses a more general problem: completing low-rank positive semi-definite matrices by asking queries actively. Then giving a max-norm bound on

the estimating a matrix when we sample elements of the matrix through a random process. Then it shows that estimating the preferences of multiple populations can be seen as a sub-problem of the matrix completion.

2. **Incorporating hashing and more general linear models into contextual bandits:** (Chapters 3 and, Chapter 4) When we can come up with good query words, we can use many search algorithms. When we have a hard time formulating what we are looking for, the search problem becomes much harder. We propose a bandit framework, where we get answers to the question: (how much) do you like the following image? Our response could be a score (which translates to standard linear bandits) or Yes/No answers.

   Besides, all current bandit algorithms scale linearly (Thompson sampling (Agrawal and Goyal, 2013)) or super-linearly (OFUL (Abbasi-Yadkori et al., 2011)) in the number of arms. In the area of uncertainty sampling, people have previously used hashing to scale sub-linearly in the number of query items. We propose an algorithm whose regret is slightly worse that state-of-the-art but whose computational complexity scales sub-linearly. Besides, we also show that practically, Thompson sampling can be used with hashing with little penalty on the regret on the Zappos50K dataset.

   There has been theory of bandits that can deal with generalized linear models but to the best of our knowledge, it applied to either logistic models only or, needed the storage of all previous queries. We therefore propose a new algorithm that can deal with much more general models while not requiring all previous data.

3. **Using what the user says:** When we are running bandits on arms whose features are high dimensional, it may be impractical to run them in real-world situations. This is because, at every iteration, for every person, we need to update a $d \times d$ matrix. We look at how we may restrict ourselves to a much small dimension k and how we could use feedback about the features themselves, in addition to the

linear rewards we get, in order to identify the k relevant features. We propose a model for getting feature feedback for documents and show on both a synthetic dataset and a dataset generated from real documents that we can improve the regret by using this feature feedback.

1.1 Chapters at a Glance

Chapter 2. Bandits and Low Rank Matrix Completion

Completion of low-rank matrices using active queries. Bound on the max-norm between the difference of the completed matrix and the original matrix. Connection to the problem of estimating the preferences of more than one population without the knowledge of which population the user belongs to.

Chapter 3. Image Search

Using linear bandits to perform personalized search. Biasing linear bandit algorithms towards the starting point to potentially get better regret scaling. Changing the objective function of OFUL ((Abbasi-Yadkori et al., 2011)) in order to allow the use of hashing. Showing that hashing, under certain assumptions, does not degrade the regret guarantees. The application of our algorithm and, other state-of-the-art algorithms to the Zappos50k dataset. Real life data from the NEXT system using these algorithms.

Chapter 4. Generalized Linear Bandits

Our premise is that it is hard for the users to search large datasets with keywords. Therefore, it our belief that it is more reasonable to expect Yes/No answers to questions rather than getting a real number in $[-1, 1]$. We propose a new algorithm that does not require the storage of all previously pulled arms and their rewards, and which applies

to a broad class of generalized linear models. We apply our algorithm to synthetic data to show the advantages of the new method.

Chapter 5. Bandits with Feature Feedback

How do we use feature feedback to recognize relevant dimensions to run bandit algorithms? How do we apply this to searching for documents from a large corpus? We apply the proposed algorithms to a synthetic dataset and to the Newsgroup 20 dataset.

Appendix: Chapter 6. Socioscope

Can we estimate real-world signals from Twitter? More specifically, if we model event intensities as happening in space and time and people as sensors of those events, can we estimate the intensity of event at different locations across different times? We apply our algorithm to estimating animal extant estimates using tweets about roadkill. We then show that these are close to the maps produced by domain scientists. We also show that we can see the diurnal-nocturnal behavior of animals.

Appendix: Chapter 7

This chapter summarizes the contribution to NEXT. It documents the modifications that have been introduced to enable the running of contextual linear bandits.

Appendix: Chapter 8

This chapter talks about the contributions made to the software called IDTaxa. It is a classification algorithm for getting taxonomy of biological samples. It talks about some of the problems of existing software and what we addressed in order to improve classification leading to state-of-the-art results.

Chapter 2

Bandits and Low-Rank Matrix Completion [1]

[1] Paper presented at AISTATS 2017

2.1 Introduction

The problem of matrix completion is a fundamental problem in machine learning and data mining where one needs to estimate an unknown matrix using only a few entries from the matrix. This problem has seen an explosion in interest in recent years perhaps fueled by the famous Netflix prize challenge Bell and Koren (2007) which required predicting the missing entries of a large movie-user rating matrix. Candès and Recht (2009) showed that by solving an appropriate semidefinite programming problem it is possible to recover a low-rank matrix given a few entries at random. Many improvements have since been made both on the theoretical side (Keshavan et al., 2009; Foygel and Srebro, 2011) as well as on the algorithmic side (Tan et al., 2014; Vandereycken, 2013; Wen et al., 2012).

Very often in applications the matrix of interest has more structure than just low rank. One such structure is positive semi-definiteness which appears when dealing with covariance matrices in applications like PCA, and kernel matrices when dealing with kernel learning. *In this chapter we study the problem of matrix completion of low-rank, symmetric positive semidefinite (PSD) matrices and provide simple, and computationally efficient algorithms that actively query a few elements of the matrix and output an estimate of the matrix that is provably close to the true PSD matrix*. More precisely, we are interested in algorithms that ouptput a matrix that is provably (ϵ, δ) close to the true underlying matrix in the max norm [2]. This means that if $\boldsymbol{L}$ is the true, underlying PSD matrix then we want our algorithms to output a matrix $\hat{\boldsymbol{L}}$ such that $\|\hat{\boldsymbol{L}} - \boldsymbol{L}\|_{\max} \leq \epsilon$, with probability at least $1 - \delta$. Our goal is strongly motivated by applications to certain multi-armed bandit problem where there are a large number of arms. In certain cases the losses of these arms can be arranged as a PSD matrix and finding the (ϵ, δ) best arm can be reduced to the above defined (ϵ, δ) PSD matrix completion (PSD-MC) problem. Our contributions are as follows

[2] The max norm of a matrix is the maximum of the absolute value of all the elements in a matrix

1. Let $\boldsymbol{L}$ be a $K \times K$ rank r PSD matrix, which is apriori unknown. We propose two models for the PSD-MC problem. In both the models the algorithm has access to an oracle $\mathcal{O}$ which when queried with a pair-of-indices (i, j) obtains a response $y_{i,j}$. The main difference between these two oracle models is the power of the oracle. In the first model, which we call as a deterministic oracle model, the oracle is a powerful, deterministic, but expensive oracle where $y_{i,j} = L_{i,j}$. In the second model, called as the stochastic oracle model, we shall assume that all the elements of the matrix $\boldsymbol{L}$ are in $[0, 1]$, and we have access to a less powerful, but cheaper oracle, whose output $y_{i,j}$ is sampled from a Bernoulli distribution with parameter $\boldsymbol{L}_{i,j}$. These models are sketched in Figure (1).

2. We propose algorithms for PSD-MC problem, under the above two models. Our algorithms, called MCANS [3], in the deterministic oracle model, and S-MCANS [4] in the stochastic oracle model are both based on the following key insight: In the case of PSD matrices it is possible to find linearly independent columns by using few, adaptively chosen queries. In the case of S-MCANS we use the above insight along with techniques from multi-armed bandits literature in order to tackle the randomness of the stochastic oracle.

3. We prove that the MCANS algorithm outputs a $(\epsilon = 0, \delta = 0)$ estimate of the matrix $\boldsymbol{L}$ (exact recovery) after making at most $K(r + 1)$ queries, and the S-MCANS algorithm outputs $\hat{\boldsymbol{L}}$ that is (ϵ, δ) close to $\boldsymbol{L}$ using queries that is linear in K and a low-order polynomial in the rank r of matrix $\boldsymbol{L}$. Establishing such sample complexity bounds leads to interesting problems in matrix approximation in the max-norm. The contributions we make here could be of independent interest in the low-rank matrix approximation literature.

4. We introduce a multi-armed bandit (MAB) problem motivated by applications

[3] MCANS stands for Matrix Completion via Adaptive Nystrom Sampling
[4] S in S-MCANS stands for stochastic

to advertising in Figure (5) and show how this MAB problem can be reduced to a PSD-MC problem. This reduction allows us to use MCANS and S-MCANS algorithms to solve the problem of finding an (ϵ, δ) optimal arm using far fewer queries than what standard multi-armed bandit based algorithms such as successive elimination or median elimination would need. Our experiments show that exploiting the spectral structure in the MAB problem allows us to design algorithms that are much more query efficient than state-of-art bandit algorithms that do not exploit the spectral structure in the bandit problem.

5. We also show that the MCANS algorithm can be effectively used to complete kernel matrices, under budget constraints. The completed kernel matrix when used in a kernel dimensionality reduction task leads to better results than a kernel matrix which has been completed using standard low-rank matrix completion.

Notation. Δ_r represents the r dimensional probability simplex. Matrices and vectors are represented in bold font. For a matrix $\boldsymbol{L}$, unless otherwise stated, the notation $\boldsymbol{L}_{i,j}$ represents (i, j) element of $\boldsymbol{L}$, and $\boldsymbol{L}_{i:j,k:l}$ is the submatrix consisting of rows $i, i+1, \ldots, j$ and columns $k, k+1, \ldots, l$. The matrix $\|\cdot\|_1$ and $\|\cdot\|_2$ norms are always operator norms. The matrix $\|\cdot\|_{\max}$ is the element wise infinity norm. Finally, let $\mathbb{1}$ be the all 1 column vector.

2.2 Related Work

The problem of PSD-MC has been considered by many other authors (Bishop and Byron, 2014; Laurent and Varvitsiotis, 2014a,b). However, all of these papers consider the passive case, i.e. the entries of the matrix that have been revealed are not under their control. In contrast, we have an active setup, where we can decide which entries in the matrix to reveal. The Nyström algorithm for approximation of low rank PSD matrices has been well studied both empirically and theoretically. Nyström methods typically choose random columns to approximate the original low-rank matrix (Gittens and Mahoney, 2013; Drineas and Mahoney, 2005). Adaptive schemes where the columns used for Nystrom approximation are chosen adaptively have also been considered in the literature. To the best of our knowledge these algorithms either need the knowledge of the full matrix (Deshpande et al., 2006) or have no provable theoretical guarantees (Kumar et al., 2012). Moreover, to the best of our knowledge all analysis of Nystrom approximation that has appeared in the literature assume that one can get error free values for entries in the matrix. Adaptive matrix completion algorithms have also been proposed and such algorithms have been shown to be less sensitive to the incoherence in the matrix (Krishnamurthy and Singh, 2013). The bandit problem that we study in the latter half of the chapter is related to the problem of pure exploration in multi-armed bandits. In such pure exploration problems one is interested in designing algorithms with low, simple regret or designing algorithms with low (ϵ, δ) query complexity. Algorithms with small simple regret have been designed in the past (Audibert and Bubeck, 2010; Gabillon et al., 2011; Bubeck et al., 2013). Even-Dar et al. (2006) suggested the Successive Elimination (SE) and Median Elimination (ME) to find near optimal arms with provable sample complexity guarantees. These sample complexity guarantees typically scale linearly with the number of arms. In principal, one could naively reduce our problem to a pure exploration problem where we need to find an

(ϵ, δ) good arm. However, such naive reductions ignore any dependency information among the arms. The S-MCANS algorithm that we design builds on the SE algorithm but crucially exploits the matrix structure in the problem to give much better algorithms than a naive reduction.

2.3 Algorithms in the deterministic oracle model

Our deterministic oracle model is shown in Figure (1) and assumes the existence of a powerful, deterministic oracle that returns queried entries of the unknown matrix accurately. Our algorithm in this model, called MCANS, is shown in Figure (2). It

Model 1 Description of deterministic and stochastic oracle models

Figure (1)

1: **while** TRUE **do**
2: Algorithm chooses a pair-of-indices (i_t, j_t).
3: Algorithm receives the response y_t defined as follows

$$y_{t,\text{det}} = \boldsymbol{L}_{i_t,j_t} \text{ // if model is deterministic} \tag{2.1}$$
$$y_{t,\text{stoc}} = \text{Bern}(\boldsymbol{L}_{i_t,j_t}) \text{ // if model is stochastic} \tag{2.2}$$

4: Algorithm stops if it has found a good approximation to the unknown matrix $\boldsymbol{L}$.
5: **end while**

is an iterative algorithm that determines which columns of the matrix are independent. MCANS maintains a set of indices (denoted as $\mathcal{C}$ in the pseudo-code) corresponding to independent columns of matrix $\boldsymbol{L}$. Initially $\mathcal{C} = \{1\}$. MCANS then makes a single pass over the columns in $\boldsymbol{L}$ and checks if the current column is independent of the columns in $\boldsymbol{C}$. This check is done in line 5 of Figure (2) and most importantly requires *only the principal sub-matrix*, of $\boldsymbol{L}$, indexed by the set $\mathcal{C} \cup \{c\}$. If the column passes this test then all the elements in this column i whose values have not been queried in the past are queried and the matrix $\hat{\boldsymbol{L}}$ is updated with these values. The test in line 5 is the column selection step of the MCANS algorithm and is justified by Proposition (2.3.1). Finally, once r independent columns have been chosen, we impute the matrix by using Nystrom extension. Nystrom based methods have been proposed in the past to handle large scale kernel matrices in the kernel based learning literature Drineas and Mahoney (2005); Kumar et al. (2012). The major difference between this work and ours is that the column selection procedure in our algorithms is deterministic, whereas in Nystrom

methods columns are chosen at random. The following proposition simply follows from the fact that any principal submatrix of an PSD matrix is also PSD and hence admits an eigen-decomposition.

Algorithm 2 Matrix Completion via Adaptive Nystrom Sampling (MCANS)

Input: A deterministic oracle that takes a pair of indices (i, j) and outputs $\boldsymbol{L}_{i,j}$.
Output: $\hat{\boldsymbol{L}}$

1: Choose the pairs $(j, 1)$ for $j = 1, 2, \ldots, K$ and set $\hat{\boldsymbol{L}}_{j,1} = \boldsymbol{L}_{j,1}$. Also set $\hat{\boldsymbol{L}}_{1,j} = \boldsymbol{L}_{j,1}$
2: $\mathcal{C} = \{1\}$ {Set of independent columns discovered till now}
3: **for** $(c = 2;\ c \leftarrow c + 1;\ c \leq K)$ **do**
4: Query the oracle for (c, c) and set $\hat{\boldsymbol{L}}_{c,c} \leftarrow \boldsymbol{L}_{c,c}$
5: **if** $\sigma_{\min}\left(\hat{\boldsymbol{L}}_{\mathcal{C}\cup\{c\},\mathcal{C}\cup\{c\}}\right) > 0$ **then**
6: $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$
7: Query $\mathcal{O}$ for the pairs $(\cdot, c)$ and set $\hat{\boldsymbol{L}}(\cdot, c) \leftarrow \boldsymbol{L}(\cdot, c)$ and by symmetry $\hat{\boldsymbol{L}}(c, \cdot) \leftarrow \boldsymbol{L}(\cdot, c)$.
8: **end if**
9: **if** $(|\mathcal{C}| = r)$ **then**
10: break
11: **end if**
12: **end for**
13: Let $\boldsymbol{C}$ denote the tall matrix comprised of the columns of $\boldsymbol{L}$ indexed by $\mathcal{C}$ and let $\boldsymbol{W}$ be the principle submatrix of $\boldsymbol{L}$ corresponding to the indices in $\mathcal{C}$. Then, construct the Nystrom extension $\mathbf{L} = \boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^{\top}$.

Proposition 2.3.1. *Let $\boldsymbol{L}$ be any PSD matrix of size K. Given a subset $\mathcal{C} \subset \{1, 2, \ldots, K\}$, the columns of the matrix $\boldsymbol{L}$ indexed by the set $\mathcal{C}$ are independent iff the principal submatrix $\boldsymbol{L}_{\mathcal{C},\mathcal{C}}$ is non-degenerate, equivalently iff, $\lambda_{\min}(\boldsymbol{L}_{\mathcal{C},\mathcal{C}}) > 0$.*

It is not hard to verify the following theorem. The proof has been relegated to the appendix.

Theorem 2.3.2. *If $\boldsymbol{L} \in \mathbf{R}^{K\times K}$ is an PSD matrix of rank r, then the matrix $\hat{\mathbf{L}}$ output by the MCANS algorithm* (2) *satisfies $\hat{\mathbf{L}} = \boldsymbol{L}$. Moreover, the number of oracle calls made by MCANS is at most $K(r+1)$. The sampling algorithm* (2) *requires: $K + (K - 1) + (K - 2) + \ldots + (K - (r - 1)) + (K - r) \leq (r + 1)K$ samples from the matrix $\boldsymbol{L}$.*

Note that the sample complexity of the MCANS algorithm is better than typical sample complexity results for LRMC and Nystrom methods. We managed to avoid factors logarithmic in dimension and rank that appear in LRMC and Nystrom methods (Gittens and Mahoney, 2013), as well as incoherence factors that are typically found in LRMC results (Candès and Recht, 2009). Also, our algorithm is purely deterministic, whereas LRMC uses randomly drawn samples from a matrix. In fact, this careful, deterministic choice of entries of the matrix is what helps us do better than LRMC.

Moreover, MCANS algorithm is optimal in a min-max sense. This is because any PSD matrix of size K and rank r is characterized via its singular value decomposition by Kr degrees of freedom. Hence, any algorithm for completion of an PSD matrix would need to see at least Kr entries. As shown in theorem (2.3.2) the MCANS algorithm makes at most $K(r+1)$ queries and hence is order optimal.

The MCANS algorithm needs the rank r as an input. However, the MCANS algorithm can be made to work even if r is unknown by simply removing the condition on line 9 in the MCANS algorithm. In this case, once r independent columns have been found, all future checks on the if statement in line 5 of MCANS will fail, and the algorithm eventually exits the for loop. Even in this case the sample complexity guarantees in Theorem (2.3.2) hold. Finally, if the matrix is not exactly rank r but can be approximated by a matrix of rank r, then we might be able to modify MCANS to output the best rank r approximation, by modifying line 5 to use an appropriate $\sigma_{\text{thresh}} > 0$. We leave this modification to future work.

2.4 Algorithms in the stochastic oracle model

For the stochastic model considered in this chapter we shall propose an algorithm, called S-MCANS, which is a stochastic version of MCANS. Like MCANS, the stochastic version discovers a set of independent columns iteratively and then uses the Nyström extension to impute the matrix. Figure (3) provides a pseudo-code of the S-MCANS algorithm.

S-MCANS like the MCANS algorithm repeatedly performs column selection steps to select a column of the matrix $\boldsymbol{L}$ that is linearly independent of the previously selected columns, and then uses these selected columns to impute the matrix via a Nystrom extension. In the case of deterministic models, due to the presence of a deterministic oracle, the column selection step is pretty straight-forward and requires calculating the smallest singular-value of certain principal sub-matrices. In contrast, for stochastic models the stochastic oracle outputs a Bernoulli random variable $\mathrm{Bern}(\boldsymbol{L}_{i,j})$ when queried with the indices (i, j). This makes the column selection step much harder. We resort to the successive elimination algorithm (shown in Fig (4)) where principal sub-matrices are repeatedly sampled to estimate the smallest singular-values for those matrices. The principal sub-matrix that has the largest smallest singular-value determines which column is selected in the column selection step.

Given a set $\mathcal{C}$, define $\boldsymbol{C}$ to be a $K \times r$ matrix corresponding to the columns of $\boldsymbol{L}$ indexed by $\mathcal{C}$ and define $\boldsymbol{W}$ to be the $r \times r$ principal submatrix of $\boldsymbol{L}$ corresponding to indices in $\mathcal{C}$. S-MCANS constructs estimators $\widehat{\boldsymbol{C}}, \mathbf{W}$ of $\boldsymbol{C}, \boldsymbol{W}$ respectively by repeatedly sampling independent entries of $\boldsymbol{C}, \boldsymbol{W}$ (which are Bernoulli) for each index and averaging these entries. The sampling is such that each entry of the matrix $\boldsymbol{C}$ is sampled at least m_1 times and each entry of the matrix $\boldsymbol{W}$ is sampled at least m_2 times,

where

$$m_1 = 100C_1(\boldsymbol{W}, \boldsymbol{C}) \log(2Kr/\delta) \max\left(\frac{r^{5/2}}{\epsilon}, \frac{r^2}{\epsilon^2}\right) \tag{2.3}$$

$$m_2 = 200C_2(\boldsymbol{W}, \boldsymbol{C}) \log(2r/\delta) \max\left(\frac{r^3}{\epsilon}, \frac{r^5}{\epsilon^2}\right) \tag{2.4}$$

and C_1, C_2 are problem dependent constants defined as

$$\begin{aligned} C_1(\boldsymbol{W}, \boldsymbol{C}) = \max\big(&\left\|\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}, \left\|\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}^2 \\ &\left\|\boldsymbol{W}^{-1}\right\|_{\max}, \left\|\boldsymbol{C}\boldsymbol{W}^{-1}\right\|_1^2, \\ &\left\|\boldsymbol{W}^{-1}\right\|_2 \left\|\boldsymbol{W}^{-1}\right\|_{\max}\big) \end{aligned} \tag{2.5}$$

$$\begin{aligned} C_2(\boldsymbol{W}, \boldsymbol{C}) = \max\big(&\left\|\boldsymbol{W}^{-1}\right\|_2^2 \left\|\boldsymbol{W}^{-1}\right\|_{\max}^2, \\ &\left\|\boldsymbol{W}^{-1}\right\|_2 \left\|\boldsymbol{W}^{-1}\right\|_{\max}, \\ &\left\|\boldsymbol{W}^{-1}\right\|_2, \left\|\boldsymbol{W}^{-1}\right\|_2^2\big) \end{aligned} \tag{2.6}$$

S-MCANS then returns the Nyström extension constructed using matrices $\widehat{\boldsymbol{C}}, \mathbf{W}$.

Algorithm 3 Stochastic Matrix Completion via Adaptive Nystrom Sampling (S-MCANS)

Input: $\epsilon > 0, \delta > 0$ and a stochastic oracle $\mathcal{O}$ that when queried with indices (i, j) outputs a Bernoulli random variable $\mathrm{Bern}(L_{i,j})$

Output: A PSD matrix $\hat{\boldsymbol{L}}$, which is an approximation to the unknown matrix $\boldsymbol{L}$, such that with probability at least $1-\delta$, all the elements of $\hat{\boldsymbol{L}}$ are within ϵ of the elements of $\boldsymbol{L}$.

1: $\mathcal{C} \leftarrow \{1\}$.
2: $\mathcal{I} \leftarrow \{2, 3, \ldots, K\}$.
3: **for** $(t = 2; t \leftarrow t + 1; t \leq r)$ **do**
4: Define, $\tilde{\mathcal{C}_\rangle} = \mathcal{C} \bigcup \{i\}, \forall i \in \mathcal{I}$.
5: Run the successive elimination algorithm 4 on matrices $\boldsymbol{L}_{\tilde{\mathcal{C}_\rangle}, \tilde{\mathcal{C}_\rangle}}$, $i \in \mathcal{I}$, with given $\delta \leftarrow \frac{\delta}{2r}$ to get $i_t^\star$.
6: $\mathcal{C} \leftarrow \mathcal{C} \bigcup \{i_t^\star\}; \mathcal{I} \leftarrow \mathcal{I} \setminus \{i_t^\star\}$.
7: **end for**
8: Obtain estimators $\widehat{\boldsymbol{C}}, \mathbf{W}$ of $\boldsymbol{C}, \boldsymbol{W}$ by repeatedly sampling and averaging entries. Calculate the Nystrom extension $\mathbf{L} = \widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top$.

Algorithm 4 Successive elimination on principal submatrices

Input: Square matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ of size $p \times p$, which share the same $p-1 \times p-1$ left principal submatrix; a failure probability $\delta > 0$; and a stochastic oracle $\mathcal{O}$

Output: An index

1: Set $t = 1$, and $\mathcal{S} = \{1, 2, \ldots, m\}$ (Here $m = K - \tau + 1$ where τ is the iteration number in MCANS, when successive elimination is invoked).
2: Sample each entry of the input matrices once.
3: **while** $|\mathcal{S}| > 1$ **do**
4: Set $\delta_t = \frac{6\delta}{\pi^2 m t^2}$
5: Let $\widehat{\sigma}^{\max} = \max_{k \in \mathcal{S}} \sigma_{\min}(\widehat{\boldsymbol{A}}_k)$ and let $k_\star$ be the index that attains argmax.
6: For each $k \in \mathcal{S}$, define $\alpha_{t,k} = \frac{2\log(2p/\delta_t)}{3 \min_{i,j} n_{i,j}(\widehat{\boldsymbol{A}}_k)} + \sqrt{\frac{\log(2p/\delta_t)}{2} \sum_{i,j} \frac{1}{n_{i,j}(\widehat{\boldsymbol{A}}_k)}}$
7: For each index $k \in \mathcal{S}$, if $\widehat{\sigma}^{\max} - \widehat{\sigma}_{\min}(\widehat{\boldsymbol{A}}_k) \geq \alpha_{t,k^\star} + \alpha_{t,k}$ then do $\mathcal{S} \leftarrow \mathcal{S} \setminus \{k\}$.
8: $t \leftarrow t + 1$
9: Sample each entry of the matrices indexed by the indices in $\mathcal{S}$ once.
10: **end while**
11: Output k, where $k \in \mathcal{S}$.

2.4.1 Sample complexity of the S-MCANS algorithm

As can be seen from the S-MCANS algorithm, samples are consumed both in the successive elimination steps (step 5 of S-MCANS) as well as during the construction of the Nyström extension. We analyze both these steps next.

Sample complexity analysis of successive elimination. Before we provide a sample complexity analysis of the S-MCANS algorithm, we need a bound on the spectral norm of random matrices with 0 mean where each element is sampled possibly different number of times. This bound plays a key role in correctness of the successive elimination algorithm. The proof of this bound follows from matrix Bernstein inequality. We relegate the proof to the appendix due to lack of space.

Lemma 2.4.1. *Let $\hat{\boldsymbol{P}}$ be a $p \times p$ random matrix that is constructed as follows. For each index (i, j), set $\hat{\boldsymbol{P}}_{i,j} = \frac{H_{i,j}}{n_{i,j}}$, where $H_{i,j}$ is an independent random variable drawn from the distribution* $\mathrm{Binomial}(n_{i,j}, p_{i,j})$. *Then, $||\hat{\boldsymbol{P}} - \boldsymbol{P}||_2 \leq \frac{2\log(2p/\delta)}{3 \min_{i,j} n_{i,j}} + \sqrt{\frac{\log(2p/\delta)}{2} \sum_{i,j} \frac{1}{n_{i,j}}}$. Furthermore, if we denote by Δ the R.H.S. in the above bound,*

then $|\sigma_{\min}(\hat{\boldsymbol{P}}) - \sigma_{\min}(\boldsymbol{P})| \leq \Delta$.

Lemma 2.4.2. *The successive elimination algorithm shown in Figure* (4) *on* m *square matrices of size* $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ *each of size* $p \times p$ *outputs an index* $i_\star$ *such that, with probability at least* $1 - \delta$*, the matrix* $\boldsymbol{A}_{i_\star}$ *has the largest minimum singular value among all the input matrices. Let,* $\Delta_{k,p} := \max_{j=1,\ldots,m} \sigma_{\min}(\boldsymbol{A}_j) - \sigma_{\min}(\boldsymbol{A}_k)$*. Then number of queries to the stochastic oracle are*

$$\sum_{k=2}^{m} O\left(p^3 \log(2p\pi^2 m^2/3\Delta_{k,p}^2\delta)/\Delta_{k,p}^2\right) + O\left(p^4 \max_k \log(2p\pi^2 m^2/3\Delta_{k,p}^2\delta)/\Delta_{k,p}^2\right) \quad (2.7)$$

Sample complexity analysis of Nystrom extension. The following theorem tells us how many calls to a stochastic oracle are needed in order to guarantee that the Nystrom extension obtained by using matrices $\widehat{\boldsymbol{C}}, \mathbf{W}$ is accurate with high probability. The proof has been relegated to the appendix.

Theorem 2.4.3. *Consider the matrix* $\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top$ *which is the Nystrom extension constructed in step 10 of the S-MCANS algorithm. Given any* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$*,* $\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top - \widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max} \leq \epsilon$ *after making a total of* $Krm_1 + r^2 m_2$ *number of oracle calls to a stochastic oracle, where* m_1, m_2 *are given in equations* (2.3)*,* (2.4)*.*

The following corollary follows directly from theorem (2.4.3), and lemma (2.4.2).

Corollary 2.4.4. *The S-MCANS algorithm outputs an* (ϵ, δ) *good arm after making at most*

$$Krm_1 + r^2 m_2 + \sum_{p=1}^{r} \sum_{k=2}^{K-r} \tilde{O}\left(\frac{p^3}{\Delta_{k,p}^2} + p^4 \max_k \frac{1}{\Delta_{k,p}^2}\right)$$

number of calls to a stochastic oracle, where $\tilde{O}$ *hides factors that are logarithmic in*

$K, r, \frac{1}{\delta}, 1/\Delta_{k,p}$, *and* m_1, m_2 *are given in equations* (2.3)*,* (2.4)*.*

In principal, precise values of m_1, m_2 given in equations (2.3), (2.4) are application dependent, and often unknown apriori. If, for a given PSD matrix $\boldsymbol{L}$, and for all possible choices of submatrices $\boldsymbol{C}$ of $\boldsymbol{L}$, which admit an invertible principal $r \times r$ sub-matrix $\boldsymbol{W}$, the terms involved in Equation (2.3), (2.4) can be upper bounded by a universal constant $\theta(\boldsymbol{L})$, then one can use $\theta(\boldsymbol{L})$ instead of the terms $C_1(\boldsymbol{W}, \boldsymbol{C}), C_2(\boldsymbol{W}, \boldsymbol{C})$ in the expressions for m_1, m_2 in equations (2.3), (2.4). For our experiments, we assume that we are given some sampling budget B that we can use to query elements of the matrix $\boldsymbol{L}$, and once we run out of this budget we stop and report the necessary error metrics. As we see MCANS and S-MCANS allow us to properly allocate our budget to obtain good estimates of the matrix $\boldsymbol{L}$.

2.5 Applications to multi-armed bandits

We shall now look at a multi-armed bandit (MAB) problem where there are a large number of arms and show how this MAB problem can be reduced to a PSD-MC problem. To motivate the MAB problem consider the following example: Suppose an advertising engine wants to show different advertisements to users. Each incoming user belongs to one of r different unknown sub-populations. Each sub-population may have different taste in advertisements. For example, if there are $r = 3$ sub-populations, then sub-population P_1 may like advertisements about vacation rentals, while P_2 may like advertisements about car rentals and population P_3 may like advertisements about motorcycles. Suppose, the advertising company has a constraint that it can show only two advertisements each time to a random, unknown incoming user. The question of interest is what would be a good pair of advertisements to show to a random incoming user in order to maximize click probability?

Such problems and more can be cast in a MAB framework, where the MAB algorithm actively elicits response from users on different pairs of advertisements. In Figure (5) we sketch the two models for the above mentioned advertising problem. In both the models, there are K ads in total, and in each round t, we choose a pair of ads and receive a reward which is a function of the pair. Let, Z_t be a multinomial random variable defined by a probability vector $\mathbf{p} \in \Delta_r$, whose output space is the set $\{1, 2, \ldots, r\}$. Let $\mathbf{u}_{Z_t}$ be a reward vector in $[0, 1]^K$ indexed by Z_t. On displaying the pair of ads (i_t, j_t) in round t the algorithm receives a scalar reward y_t. This reward is large if either of the ads in the chosen pair is "good". For both the models we are interested in designing algorithms that discover an (ϵ, δ) best pair of ads using as few trials as possible, i.e. algorithms which can output, with probability at least $1-\delta$, a pair of ads that is ϵ close to the best pair of ads in terms of the expected reward of the pair. The difference between the models is whether the reward is stochastic or deterministic.

In the deterministic model y_t is deterministic and is given by Equation (2.8), whereas

Model 5 Description of our proposed models

1: **while** TRUE **do**
2: In the case of stochastic model, nature chooses $Z_t \sim \text{Mult}(\mathbf{p})$, but does not reveal it to the algorithm.
3: Algorithm chooses a pair of items (i_t, j_t).
4: Algorithm receives the reward y_t defined as follows: If the model is deterministic

$$y_{t,\text{det}} = 1 - \mathbb{E}_{Z_t \sim \mathbf{p}}(1 - \mathbf{u}_{Z_t}())(1 - \mathbf{u}_{Z_t}(j_t)) \tag{2.8}$$

If the model is stochastic

$$y_{t,\text{stoc}} = \max\{y, y_{j_t}\} \tag{2.9}$$
$$y \sim \text{Bern}(\mathbf{u}_{Z_t}()) \tag{2.10}$$
$$y_{j_t} \sim \text{Bern}(\mathbf{u}_{Z_t}(j_t)) \tag{2.11}$$

5: Algorithm stops if it has found a certifiable (ϵ, δ) optimal pair of items.
6: **end while**

in the stochastic model y_t is a random variable that depends on the random variable Z_t as well as additional external randomness. However, a common aspect of both these models is that the expected reward associated with the pair of choices (i_t, j_t) in round t is the same and is equal to the expression given in Equation (2.8). It is clear from Figure (5) that the optimal pair of ads satisfies the equation

$$(i_\star, j_\star) = \underset{i,j}{\text{argmin}}\, \mathbb{E}_{Z_t \sim \mathbf{p}}(1 - \mathbf{u}_{Z_t}(i))(1 - \mathbf{u}_{Z_t}(j)). \tag{2.12}$$

A naive way to solve this problem is to treat this problem as a best-arm identification problem in stochastic multi-armed bandits where there are $\Theta(K^2)$ arms each corresponding to a pair of items. One could now run a Successive Elimination (SE) algorithm or a Median Elimination algorithm on these $\Theta(K^2)$ pairs Even-Dar et al. (2006) to find an (ϵ, δ) optimal pair. The sample complexity of the SE or ME algorithms

on these $\Theta(K^2)$ pairs would be roughly $\tilde{\mathcal{O}}(\frac{K^2}{\epsilon^2})$ [5]. In the advertising application that we mentioned before and other applications K can be very large, and therefore the sample complexity of such naive algorithms can be very large. However, these simple reductions throw away information between different pairs of items and hence are sub-optimal. We next show that via a simple reduction it is possible to convert this MAB problem to a PSD-MC problem.

2.5.1 Reduction from MAB to PSD matrix completion

Since, we are interested in returning an (ϵ, δ) optimal pair of ads it is enough if the pair returned by our algorithm attains an objective function value that is at most ϵ more than the optimal value of the objective function shown in equation (2.12), with probability at least $1 - \delta$. Let $\mathbf{p} \in \Delta_r$, and let the reward matrix $\boldsymbol{R} \in \mathbb{R}^{K \times K}$ be such that its $(i, j)^{\text{th}}$ entry is the expected reward obtained using the pair of ads (i, j). Then from equation (2.12) we know that the $(i, j)^{\text{th}}$ element of matrix $\boldsymbol{R}$ has the form

$$\begin{aligned}
\boldsymbol{R}_{i,j} &= 1 - \mathbb{E}_{Z_t \sim \mathbf{p}}(1 - \mathbf{u}_{Z_j}(i))(1 - \mathbf{u}_{Z_j}(j)) \\
&= 1 - \sum_{k=1}^{r} \mathbf{p}_k (1 - \mathbf{u}_k(i))(1 - \mathbf{u}_k(j)) \qquad (2.13) \\
\boldsymbol{R} &= \mathbb{1}\mathbb{1}^\top - \underbrace{\sum_{k=1}^{r} \mathbf{p}_k (\mathbb{1} - \mathbf{u}_k)(\mathbb{1} - \mathbf{u}_k)^\top}_{\boldsymbol{L}}. \qquad (2.14)
\end{aligned}$$

It is enough to find an entry in the matrix $\boldsymbol{L}$ that is ϵ close to the smallest entry in the matrix $\boldsymbol{L}$ with probability at least $1 - \delta$. In order to do this it is enough to estimate the matrix $\boldsymbol{L}$ using repeated trials and then use the pair-of-indices corresponding to the smallest entry as an (ϵ, δ) optimal pair. In order to do this we exploit the structural properties of matrix $\boldsymbol{L}$. From equation (2.14) it is clear that the matrix $\boldsymbol{L}$ can be written as a sum of r rank-1 matrices. Hence $\mathrm{rank}(\boldsymbol{L}) \leq r$. Furthermore, since these rank-1

[5] The $\tilde{\mathcal{O}}$ notation hides logarithmic dependence on $\frac{1}{\delta}, K, \frac{1}{\delta}$

matrices are all positive semi-definite and $\boldsymbol{L}$ is a convex combination of such, we can conclude that $\boldsymbol{L} \succeq 0$. We have proved the following proposition:

Proposition 2.5.1. *The matrix $\boldsymbol{L}$ shown in equation* (2.14) *satisfies the following two properties: (i)* $\mathrm{rank}(\boldsymbol{L}) \leq r$ *(ii)* $\boldsymbol{L} \succeq 0$.

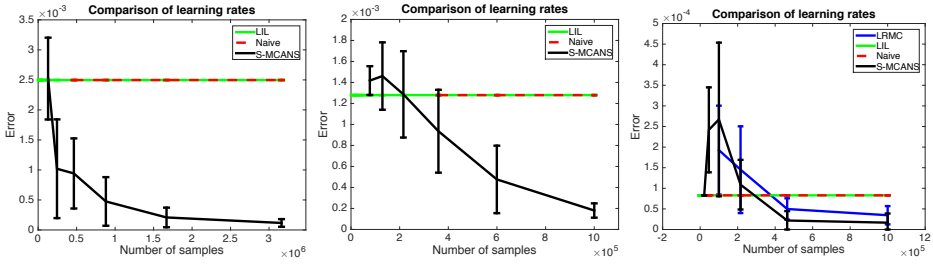
The above property immediately implies that we can treat the MAB problem as a MC-PSD problem.

Proposition 2.5.2. *The (ϵ, δ) optimal pair for the MAB problem shown in model* (5) *with deterministic rewards can be reduced to a PSD-MC problem with a deterministic oracle. Using the MCANS algorithm we can obtain a $(0, 0)$ optimal arm using less than $(r + 1)K$ queries. Similarly, the (ϵ, δ) optimal pair for the MAB problem shown in Figure* (5)*, under the stochastic model can be reduced to a PSD-MC problem with a stochastic oracle. Using the S-MCANS algorithm we can obtain an (ϵ, δ) optimal pair-of-arms using number of trials equal to the quantity shown in Corollary* (2.4.4).

2.5.2 Related work to multi-armed bandits

Bandit problems where multiple actions are selected have also been considered in the past usually in the context of computational advertising (Kale et al., 2010), information retrieval Radlinski et al. (2008), Yue and Guestrin (2011), resource allocation Streeter and Golovin (2009). A major difference between the above mentioned works and our work is that our feedback and reward model is different and that we are not interested in cumulative regret guarantees but rather in finding a good pair of arms as quickly as possible. Furthermore our linear-algebraic approach to the problem is very different from the approaches taken in the previous papers. Finally we would like to mention that our model shown in Figure (5) on the surface bears resemblance to dueling bandit problems (Yue et al., 2012a). However, in duleing bandits two arms are compared which is not the case in the bandit problem that we study. A more thorough literature survey has been relegated to the appendix due to lack of space.

2.6 Experiments



(a) ML-100K; $K = 800, r = 2$ (b) ML-100K; $K = 800, r = 4$ (c) ML-1M; $K = 200, r = 2$

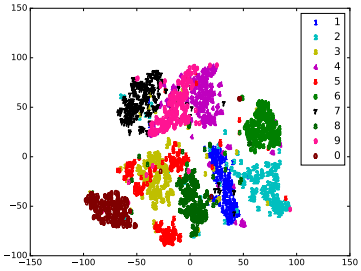
Figure 2.1: Error of various algorithms with increasing budget. The error is defined as $\boldsymbol{L}_{\hat{i},\hat{j}} - \boldsymbol{L}_{i_\star,j_\star}$ where $(\hat{i}, \hat{j})$ is a pair of optimal choices as estimated by each algorithm. Note that the Naive and LIL' UCB have similar performances and do not improve with budget and both are outperformed by R-PLANS. This is because both Naive and LIL' UCB have few samples that they can use on each pair of movies. All experiments were repeated 10 times.

In this section we demonstrate experiments to show the efficacy of our proposed algorithms: MCANS and S-MCANS.

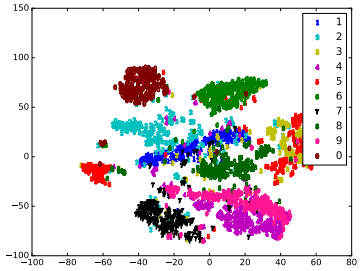
2.6.1 Movie reommendation as a MAB problem

We describe a multi-armed bandit task where the target is to recommend a good pair of movies to users.

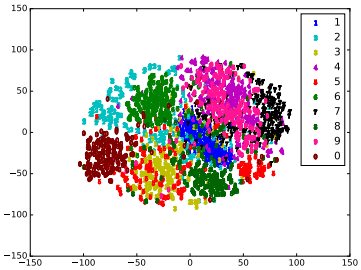
Experimental setup. We used the Movie Lens datasets (Harper and Konstan, 2015), namely ML-100K, ML-1M. This dataset contains incomplete movie ratings provided by users for different movies. We pre-process this dataset to make it suitable for a bandit experiment as follows: We use this incomplete user-movie ratings dataset as an input to an LRMC solver called OptSpace. The complete ratings obtained from an LRMC solver are then thresholded to obtain binary values. More precisely, all ratings of at least 3 are set to 1 and ratings less than 3 are set to 0. All the users are assigned to different sub-populations, based on some attribute of the user. For example in Figures (2.1a), (2.1b) the gender attribute is used, to create 2 sub-populations and in



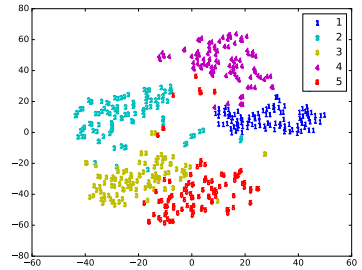
(a) t-SNE with the full kernel matrix .



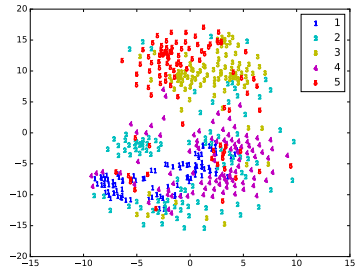
(b) t-SNE + MCANS, $B = 250K$



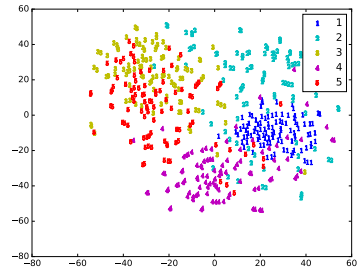
(c) t-SNE + LRMC, $B = 250K$



(d) t-SNE with the full kernel matrix.



(e) t-SNE + MCANS, $B = 8K$



(f) t-SNE + LRMC, $B = 8K$

Figure 2.2: $2-$ dimensional visualization obtained by using a partially observed RBF kernel matrix with the t-SNE algorithm for kernel dimensionality reduction. The budget B specifies how many entries of the RBF kernel matrix the algorithms are allowed to query. The kernel matrix obtained using MCANS and LRMC are then fed into t-SNE. The subfigures (a) to (c) show the results on the MNIST2500 dataset and the subfigures (d) to (f) show the results on USPS500 dataset, obtained by subsampling digits $1-5$ of the USPS dataset. Figures (2.2a),(2.2d) shows the result of KDR when the entire kernel matrix is observed. The MNIST2500 dataset is available at `https://lvdmaaten.github.io/tsne/`

Figure (2.1b) occupation of the user is used to define the resulting 4 sub-populations. In the final step we averaged the binary ratings of all users in a certain population to get the probability that a random user from a given sub-population likes a certain movie. This gets us matrices $\boldsymbol{R}$ and $\boldsymbol{L} = 1 - \boldsymbol{R}$. In the experiments we provide the different algorithms with increasing budget and measure the error of each algorithm in finding the best pair of movies. The algorithms that we use for comparison are Naive, LiL'UCB (Jamieson et al., 2014) and LRMC using OptSpace (Keshavan et al., 2009). The naive algorithm uniformly distributes the given budget equally among all the $K(K+1)/2$ pairs of movies. LIL applies the LiL' UCB algorithm treating each pair of movies as an arm in a stochastic multi-armed bandit game. All algorithms can access entries of the matrix $\boldsymbol{L}$ via noisy queries of the form (i, j) and obtain a Bernoulli outcome with probability $\boldsymbol{L}_{i,j}$. No other information such as sub-populations are available to any of the algorithms. The setup faithfully imitates the stochastic oracle model shown in Figure (5).

As can be seen from the figures (2.1) the Naive and LIL'UCB algorithms have similar performance on all the datasets. On the ML-100K datasets LIL'UCB quickly finds a good pair of movies but fails to improve with an increase in the budget. To see why, observe that there are about 32×10^4 pairs of movies. The maximum budget here is on the order of 10^6. Therefore, Naive samples each of those pairs on an average at most four times. Since many entries in the matrix are of the order of 10^{-4}, Naive algorithm a lot of sees 0's when sampling. The same thing happens with the LIL'UCB algorithm too; very few samples are available for each pair to improve its confidence bounds. This explains why the Naive and LIL' UCB algorithm have such poor and similar performances. In contrast, S-MCANS focuses most of the budget on a few select pairs and infers the value of other pairs via Nystrom extensio. This is why, in our experiments we see that S-MCANS finds good pair of movies quickly and finds even better pairs with increasing budget, outperforming all the other algorithms. S-MCANS

is also better than LRMC, because we specifically exploit the SPSD structure in our matrix $\boldsymbol{L}$, which enables us to do better. We would like to mention that on ML-100K dataset the performance of LRMC was much inferior and this result and more results are in the appendix.

2.6.2 Kernel dimensionality reduction under budget

Kernel based dimensionality reduction (KDR) is a suite of powerful non-linear dimensionality reduction techniques which all use a kernel matrix in order to perform dimensionality reduction. Given a collection of points residing in a d dimensional space where d is very large most KDR based techniques require constructing a kernel matrix between all pairs of points. A popular kernel matrix used in KDR is an RBF kernel matrix, obtained using all pairwise distances. Calculating all pairwise distances takes $O(K^2d)$ time which can be large when d is very high. Hence, we need algorithms that can use only a few pair-wise distance measurements and use the incomplete kernel matrix to perform dimensionality reduction. Given a budget of $B = O(Kr)$ we know that MCANS can recover the underlying kernel matrix, and hence using MCANS in KDR, one can expect good results in time $O(Krd + K^2r) << O(K^2d)$. In most cases $r << d$, and hence savings can be substantial. In the experiments shown in this section, we want to investigate how the estimate of the kernel matrix provided by MCANS and LRMC effect KDR. In order to do this we use as our true kernel matrix $\boldsymbol{L}$ a matrix obtained by applying the RBF kernel to all pairs of points. All algorithms are assumed to have an access to a deterministic, oracle that can query at the most B entries of $\boldsymbol{L}$. We compare MCANS with LRMC using SoftImpute (Mazumder et al., 2010) as implemented in the python package fancyimpute. For the LRMC implementation we sample B indices randomly from the upper triangle of the kernel matrix $\boldsymbol{L}$, and use these sampled values in the corresponding lower triangle too. The completed matrices are then used in t-SNE (Maaten and Hinton, 2008) to visualize the USPS digits dataset

and the MNIST2500 dataset. The results are shown in Figure (2.2). As we can see even with a limited budget t-SNE when used with MCANS as the matrix completion algorithm provides clusters with comparable quality as the ones obtained by having the full kernel matrix. On the MNIST2500 dataset the cluster quality using MCANS is as good as one obtained by the using t-SNE with full kernel matrix. However, the LRMC algorithm when used with t-SNE output poor quality clusters.

2.7 Appendix

2.7.1 Preliminaries

We shall repeat a proposition that was stated earlier in the chapter.

Proposition 2.7.1. *Let $\boldsymbol{L}$ be any SPSD matrix of size K. Given a subset $\mathcal{C} \subset \{1, 2, \ldots, K\}$, the columns of the matrix $\boldsymbol{L}$ indexed by the set $\mathcal{C}$ are independent iff the principal submatrix $\boldsymbol{L}_{\mathcal{C},\mathcal{C}}$ is non-degenerate, equivalently iff, $\lambda_{\min}(\boldsymbol{L}_{\mathcal{C},\mathcal{C}}) > 0$.*

We would also need the classical matrix Bernstein inequality, which we borrow from the work of Joel Tropp Tropp (2015).

Theorem 2.7.2. *Let $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ be independent, centered random matrices with dimension $d_1 \times d_2$ and assume that each one is uniformly bounded*

$$\mathbb{E}\boldsymbol{S}_k = 0, \|\boldsymbol{S}_k\| \leq L \text{for each } k = 1, \ldots, n.$$

Introduce the sum $\boldsymbol{Z} = \sum_{k=1}^{n} \boldsymbol{S}_k$, and let $\nu(\boldsymbol{Z})$ denote the matrix variance statistic of the sum:

$$\nu(\boldsymbol{Z}) = \max\left\{\left\|\mathbb{E}\boldsymbol{Z}\boldsymbol{Z}^\top\right\|, \left\|\mathbb{E}\boldsymbol{Z}^\top\boldsymbol{Z}\right\|\right\} \tag{2.15}$$

$$= \max\left\{\left\|\sum_{k=1}^{n}\mathbb{E}\boldsymbol{S}_k\boldsymbol{S}_k^\top\right\|, \left\|\sum_{k=1}^{n}\mathbb{E}\boldsymbol{S}_k^\top\boldsymbol{S}_k\right\|\right\} \tag{2.16}$$

Then,

$$\mathbb{P}(\|\boldsymbol{Z}\| \geq t) \leq (d_1 + d_2)\exp\left(-\frac{t^2/2}{\nu(\boldsymbol{Z}) + \frac{Lt}{3}}\right)$$

2.7.2 Sample complexity of MCANS algorithm: Proof of Theorem 2.3.2

Theorem. *If $\boldsymbol{L} \in \mathbf{R}^{K\times K}$ is an SPSD matrix of rank r, then the matrix $\hat{\boldsymbol{L}}$ output by the MCANS algorithm satisfies $\hat{\boldsymbol{L}} = \boldsymbol{L}$. Moreover, the number of oracle calls made by MCANS is at most $K(r+1)$. The sampling algorithm requires: $K + (K-1) + (K-2) + \ldots + (K-(r-1)) + (K-r) \leq (r+1)K$ samples from the matrix $\boldsymbol{L}$.*

Proof. MCANS checks one column at a time starting from the second column, and uses the test in line 5 to determine if the current column is independent of the previous columns. The validity of this test is guaranteed by proposition (2.3.1). Each such test needs just one additional sample corresponding to the index (c, c). If a column c is found to be independent of the columns $1, 2, \ldots, c-1$ then rest of the entries in column c are queried. Notice, that by now we have already queried all the columns and rows of matrix $\boldsymbol{L}$ indexed by the set $\mathcal{C}$, and also queried the element (c, c) in line 4. Hence we need to query only $K - |\mathcal{C}| - 1$ more entries in column c in order to have all the entries of column c. Combined with the fact that we query only r columns completely and in the worst case all the diagonal entries might be queried, we get the total query complexity to be $(K-1) + (K-2) + \ldots (K-r) + K \leq K(r+1)$. □

2.7.3 Proof of Lemma 2.4.1

We begin by stating the lemma.

Lemma. *Let $\hat{\boldsymbol{P}}$ be a $p \times p$ random matrix that is constructed as follows. For each index (i, j) independent of other indices, set $\hat{\boldsymbol{P}}_{i,j} = \frac{H_{i,j}}{n_{i,j}}$, where $H_{i,j}$ is a random variable drawn from the distribution* $\mathrm{Binomial}(n_{i,j}, p_{i,j})$. *Let $\boldsymbol{Z} = \hat{\boldsymbol{P}} - \boldsymbol{P}$. Then,*

$$||\boldsymbol{Z}||_2 \leq \frac{2\log(2p/\delta)}{3\min_{i,j} n_{i,j}} + \sqrt{\frac{\log(2p/\delta)}{2}\sum_{i,j}\frac{1}{n_{i,j}}}. \tag{2.17}$$

Furthermore, if we denote by Δ *the R.H.S. in Equation* (2.17)*, then* $|\sigma_{\min}(\hat{\boldsymbol{P}}) - \sigma_{\min}(\boldsymbol{P})| \leq \Delta$.

Proof. Define, $\boldsymbol{S}_{i,j}^t = \frac{1}{n_{i,j}}(X_{i,j}^t - p_{i,j})\boldsymbol{E}_{i,j}$, where $\boldsymbol{E}_{i,j}$ is a $p \times p$ matrix with a 1 in the $(i,j)^{\text{th}}$ entry and 0 everywhere else, and $X_{i,j}^t$ is a random variable sampled from the distribution $\mathrm{Bern}(p_{i,j})$. If $X_{i,j}^t$ are independent for all t, i, j, then it is easy to see that $Z = \sum_{i,j} \frac{1}{n_{i,j}} \sum_{t=1}^{n_{i,j}} \boldsymbol{S}_{i,j}^t$. Hence $\boldsymbol{S}$ is a sum of independent random matrices and this allows to apply matrix Bernstein type inequalities. In order to apply the matrix Bernstein inequality, we would need upper bound on maximum spectral norm of the summands, and an upper bound on the variance of $\boldsymbol{Z}$. We next bound these two quantities as follows,

$$||\boldsymbol{S}_{i,j}^t||_2 = ||\frac{1}{n_{i,j}}(X_{i,j}^t - p_{i,j})\boldsymbol{E}_{i,j}||_2 = \frac{1}{n_{i,j}}|X_{i,j}^t - p_{i,j}| \leq \frac{1}{n_{i,j}}. \tag{2.18}$$

To bound the variance of $\boldsymbol{Z}$ we proceed as follows

$$\nu(Z) = ||\sum_{i,j}\sum_{t=1}^{n_{i,j}} \mathbb{E}(\boldsymbol{S}_{i,j}^t)^\top \boldsymbol{S}_{i,j}^t|| \bigwedge ||\sum_{i,j}\sum_{t=1}^{n_{i,j}} \mathbb{E}\boldsymbol{S}_{i,j}^t(\boldsymbol{S}_{i,j}^t)^\top|| \tag{2.19}$$

Via elementary algebra and using the fact that $\mathrm{Var}(X_{i,j}^t) = p_{i,j}(1 - p_{i,j})$It is easy to see that,

$$\mathbb{E}(\boldsymbol{S}_{i,j}^t)^\top \boldsymbol{S}_{i,j}^t = \frac{1}{n_{i,j}^2}\mathbb{E}(X_{i,j}^t - p_{i,j})^2(\boldsymbol{E}_{i,j}^t)^\top \boldsymbol{E}_{i,j}^t \tag{2.20}$$

$$= \frac{1}{4n_{i,j}^2}\boldsymbol{E}_{i,i}. \tag{2.21}$$

Using similar calculations we get $\mathbb{E}\boldsymbol{S}_{i,j}^t(\boldsymbol{S}_{i,j}^t)^\top = \frac{1}{4n_{i,j}^2}\boldsymbol{E}_{j,j}$. Hence, $\nu(Z) = \sum_{i,j}\sum_{t=1}^{n_{i,j}} \frac{1}{4n_{i,j}^2} = \sum_{i,j} \frac{1}{4n_{i,j}}$. Applying matrix Bernstein, we get with probability at least $1 - \delta$

$$||Z||_2 \leq \frac{2\log(2p/\delta)}{3\min_{i,j} n_{i,j}} + \sqrt{\frac{\log(2p/\delta)}{2}\sum_{i,j}\frac{1}{n_{i,j}}}. \tag{2.22}$$

The second part of the result follows immediately from Weyl's inequality which says that $|\sigma_{\min}(\hat{\boldsymbol{P}}) - \sigma_{\min}(\boldsymbol{P})| \leq ||\hat{\boldsymbol{P}} - \boldsymbol{P}|| = ||Z||$. □

2.7.4 Sample complexity of successive elimination algorithm: Proof of Lemma 2.4.2

Lemma. *The successive elimination algorithm 4 on m square matrices of size $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ each of size $p \times p$ outputs an index $i_\star$ such that, with probability at least $1-\delta$, the matrix $\boldsymbol{A}_{i_\star}$ has the largest smallest singular value among all the input matrices. The total number of queries to the stochastic oracle are*

$$\sum_{k=2}^{m} O\left(\frac{p^3 \log(2p\pi^2 m^2/3\Delta_k^2\delta)}{\Delta_k^2}\right) + O\left(p^4 \max_k \left(\frac{\log(2p\pi^2 m^2/3\Delta_k^2\delta)}{\Delta_k^2}\right)\right) \quad (2.23)$$

where $\Delta_{k,p} := \max_{j=1,\ldots,m} \sigma_{\min}(\boldsymbol{A}_j) - \sigma_{\min}(\boldsymbol{A}_k)$

Proof. Suppose matrix $\boldsymbol{A}_1$ has the largest smallest singular value. From lemma (2.4.1), we know that with probability at least $1-\delta_t$, $|\sigma_{\min}(\widehat{\boldsymbol{A}}_k) - \sigma_{\min}(\boldsymbol{A}_k)| \leq \frac{2\log(2p/\delta_t)}{3\min\limits_{i,j} n_{i,j}(\boldsymbol{A})} + \sqrt{\frac{\log(2p/\delta_t)}{2}\sum_{i,j}\frac{1}{n_{i,j}(\boldsymbol{A})}}$. Hence, by union bound the probability that the matrix $\boldsymbol{A}_1$ is eliminated in one of the rounds is at most $\sum_t \sum_{k=1}^m \delta_t \leq \sum_{t=1}^{\max} \sum_{k=1}^m \frac{6\delta}{\pi^2 m t^2} = \delta$. This proves that the successive elimination step identifies the matrix with the largest smallest singular value.

An arm k is eliminated in round t if $\alpha_{t,1} + \alpha_{t,k} \leq \hat{\sigma}_t^{\max} - \sigma_{\min}(\widehat{\boldsymbol{A}}_k)$. By definition,

$$\begin{aligned}\Delta_{k,p} - (\alpha_{t,1} + \alpha_{t,k}) &= (\sigma_{\min}(\boldsymbol{A}_1) - \alpha_{t,1}) - (\sigma_{\min}(\boldsymbol{A}_k) + \alpha_{t,k}) \\ &\geq \sigma_{\min}(\hat{\boldsymbol{A}}_1) - \sigma_{\min}(\boldsymbol{A}_k) \geq \alpha_{t,1} + \alpha_{t,k} \end{aligned} \quad (2.24)$$

That is if $\alpha_{t,1} + \alpha_{t,k} \leq \frac{\Delta_{k,p}}{2}$, then arm k is eliminated in round t. By construction, since in round t each element in each of the surviving set of matrices has been queried at least t times, we can say that $\alpha_{t,j} \leq \frac{2\log(2p/\delta_t)}{3t} + \sqrt{\frac{p^2\log(2p/\delta_t)}{2t}}$ for any index j

corresponding to the set of surviving arms. Hence arm k gets eliminated after

$$t_k = O\left(\frac{p^2 \log(2p\pi^2 m^2/3\Delta_{k,p}^2\delta)}{\Delta_{k,p}^2}\right) \tag{2.25}$$

In each round t the number of queries made are $O(p)$ for each of the m matrices corresponding to the row and column which is different among them, and $O(p^2)$ corresponding to the left $p-1 \times p-1$ submatrix that is common to all of the matrices $A_1, \ldots, A_m$. Hence, the total number of queries to the stochastic oracle is

$$p\sum_{k=2}^{m} t_k + p^2 \max_k t_k = \sum_{k=2}^{m} O\left(\frac{p^3 \log(2p\pi^2 m^2/3\Delta_{k,p}^2\delta)}{\Delta_{k,p}^2}\right) + O\left(p^4 \max_k \left(\frac{\log(2p\pi^2 m^2/3\Delta_{k,p}^2\delta)}{\Delta_{k,p}^2}\right)\right) \qquad \square$$

2.7.5 Proof of Nystrom method

In this supplementary material we provide a proof of Nystrom extension in $\max$ norm when we use a stochastic oracle to obtain estimators $\widehat{\boldsymbol{C}}, \mathbf{W}$ of the matrices $\boldsymbol{C}, \boldsymbol{W}$. The question that we are interested in is how good is the estimate of the Nystrom extension obtained using matrices $\widehat{\boldsymbol{C}}, \mathbf{W}$ w.r.t. the Nystrom extension obtained using matrices $\boldsymbol{C}, \boldsymbol{W}$. This is answered in the theorem below.

Theorem 2.7.3. *Suppose the matrix $\boldsymbol{W}$ is an invertible $r \times r$ matrix. Suppose, by multiple calls to a stochastic oracle we construct estimators $\widehat{\boldsymbol{C}}, \mathbf{W}$ of $\boldsymbol{C}, \boldsymbol{W}$. Now, consider the matrix $\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top$ as an estimate $\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top$. Given any $\delta \in (0,1)$, with probability atleast $1-\delta$,*

$$\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top - \widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max} \leq \epsilon$$

after making M number of oracle calls to a stochastic oracle, where

$$M \geq 100C_1(W,C)\log(2Kr/\delta)\max\left(\frac{Kr^{7/2}}{\epsilon},\frac{Kr^3}{\epsilon^2}\right)+200C_2(W,C)\log(2r/\delta)\max\left(\frac{r^5}{\epsilon},\frac{r^7}{\epsilon^2}\right)$$

where $C_1(\boldsymbol{W},\boldsymbol{C})$ and $C_2(\boldsymbol{W},\boldsymbol{C})$ are given by the following equations

$$C_1(\boldsymbol{W},\boldsymbol{C})=\max\left(\left\|\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max},\left\|\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}^2,\left\|\boldsymbol{W}^{-1}\right\|_{\max},\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\right\|_1^2,\left\|\boldsymbol{W}^{-1}\right\|_2\left\|\boldsymbol{W}^{-1}\right\|_{\max}\right)$$
$$C_2(\boldsymbol{W},\boldsymbol{C})=\max\left(\left\|\boldsymbol{W}^{-1}\right\|_2^2\left\|\boldsymbol{W}^{-1}\right\|_{\max}^2,\left\|\boldsymbol{W}^{-1}\right\|_2\left\|\boldsymbol{W}^{-1}\right\|_{\max},\left\|\boldsymbol{W}^{-1}\right\|_2,\left\|\boldsymbol{W}^{-1}\right\|_2^2\right)$$

Our proof proceeds by a series of lemmas, which we state next.

Lemma 2.7.4.

$$\begin{aligned}\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max}&\leq\left\|(\boldsymbol{C}-\widehat{\boldsymbol{C}})\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}+\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1}(\boldsymbol{C}-\widehat{\boldsymbol{C}})^\top\right\|_{\max}\\&\quad+\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1}-\mathbf{W}^{-1})\boldsymbol{C}^\top\right\|_{\max}\end{aligned}$$

Proof.

$$\begin{aligned}\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max}&=\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\boldsymbol{W}^{-1}\boldsymbol{C}^\top+\widehat{\boldsymbol{C}}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max}\\&\leq\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}\\&\quad+\left\|\widehat{\boldsymbol{C}}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max}\\&=\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}+\\&\quad\left\|\widehat{\boldsymbol{C}}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\boldsymbol{C}^\top+\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max}\\&\leq\left\|\boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}+\left\|\widehat{\boldsymbol{C}}W^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}\\&\quad+\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\boldsymbol{C}^\top-\widehat{\boldsymbol{C}}\mathbf{W}^{-1}\widehat{\boldsymbol{C}}^\top\right\|_{\max}\\&=\left\|(\boldsymbol{C}-\widehat{\boldsymbol{C}})\boldsymbol{W}^{-1}\boldsymbol{C}^\top\right\|_{\max}+\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1}(\boldsymbol{C}-\widehat{\boldsymbol{C}})^\top\right\|_{\max}\\&\quad+\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1}-\mathbf{W}^{-1})\boldsymbol{C}^\top\right\|_{\max}\end{aligned}$$

□

In the following lemmas we shall bound the three terms that appear in the R.H.S of the bound of Lemma (2.7.4).

Lemma 2.7.5.

$$\left\|(C-\hat{C})W^{-1}C^{\top}\right\|_{\max} \leq \frac{2||\boldsymbol{W}^{-1}\boldsymbol{C}^{\top}||_{\max}}{3m}\log(2Kr/\delta)+\sqrt{\frac{r\,||\boldsymbol{W}^{-1}\boldsymbol{C}^{\top}||^2_{\max}\log(2Kr/\delta)}{2m}} \tag{2.26}$$

Proof. Let $\boldsymbol{M}=\boldsymbol{W}^{-1}\boldsymbol{C}^{\top}$, then $\left\|(\boldsymbol{C}-\widehat{\boldsymbol{C}})\boldsymbol{W}^{-1}\boldsymbol{C}^{\top}\right\|_{\max} = \left\|(\boldsymbol{C}-\widehat{\boldsymbol{C}})\boldsymbol{M}\right\|_{\max}$. By the definition of max norm we have

$$\left\|(\boldsymbol{C}-\widehat{\boldsymbol{C}})M\right\|_{\max} = \max_{i,j}\left|\sum_{p=1}^{l}(\boldsymbol{C}-\widehat{\boldsymbol{C}})_{i,p}\boldsymbol{M}_{p,j}\right|$$

Fix a pair of indices (i,j), and consider the expression $\left|\sum_{p=1}^{l}(\boldsymbol{C}-\widehat{\boldsymbol{C}})_{i,p}\boldsymbol{M}_{p,j}\right|$

Define $r_{i,p}=(\boldsymbol{C}-\widehat{\boldsymbol{C}})_{i,p}$. By definition of $r_{i,p}$ we can write $r_{i,p}=\frac{1}{m}\sum_{t=1}^{m}r^t_{i,p}$, where $r^t_{i,p}$ are a set of independent random variables with mean 0 and variance at most $1/4$. This decomposition combined with scalar Bernstein inequality gives that with probability at least $1-\delta$

$$\begin{aligned}\left|\sum_{p=1}^{l}(\widehat{\boldsymbol{C}}-\widehat{\boldsymbol{C}})_{i,p}\boldsymbol{M}_{p,j}\right| &= \left|\sum_{p=1}^{l}r_{i,p}\boldsymbol{M}_{p,j}\right| \\ &= \left|\sum_{p=1}^{l}\sum_{t=1}^{m}\frac{1}{m}r^t_{i,p}\boldsymbol{M}_{p,j}\right| \\ &\leq \frac{2||M||_{\max}}{3m}\log(2/\delta)+\sqrt{\frac{r\,||M||^2_{\max}\log(2/\delta)}{2m}}\end{aligned}$$

Applying a union bound over all possible Kr choices of index pairs (i,j), we get the desired result. □

Before we establish bounds on the remaining two terms in the RHS of Lemma (2.7.4) we state and prove a simple proposition that will be used at many places in the rest of the proof.

Proposition 2.7.6. *For any two real matrices $M_1 \in \mathbf{R}^{n_1 \times n_2}, M_2 \in \mathbf{R}^{n_2 \times n_3}$ the following set of inequalities are true:*

1. $\|\boldsymbol{M}_1\boldsymbol{M}_2\|_{\max} \leq \|\boldsymbol{M}_1\|_{\max} \|\boldsymbol{M}_2\|_1$
2. $\|\boldsymbol{M}_1\boldsymbol{M}_2\|_{\max} \leq \left\|\boldsymbol{M}_1^\top\right\|_1 \|\boldsymbol{M}_2\|_{\max}$
3. $\|\boldsymbol{M}_1\boldsymbol{M}_2\|_{\max} \leq \|\boldsymbol{M}_1\|_2 \|\boldsymbol{M}_2\|_{\max}$
4. $\|\boldsymbol{M}_1\boldsymbol{M}_2\|_{\max} \leq \|\boldsymbol{M}_2\|_2 \|\boldsymbol{M}_1\|_{\max}$

where, the $\|\cdot\|_p$ is the induced p norm.

Proof. Let $\mathbf{e}_i$ denote the i^{th} canonical basis vectors in $\mathbb{R}^K$. We have,

$$\begin{aligned}
\|\boldsymbol{M}_1\boldsymbol{M}_2\|_{\max} &= \max_{i,j} \left|e_i^\top \boldsymbol{M}_1\boldsymbol{M}_2 e_j\right| \\
&\leq \max_{i,j} \left\|e_i^\top \boldsymbol{M}_1\right\|_{\max} \|\boldsymbol{M}_2 e_j\|_1 \\
&= \max_{i} \left\|e_i^\top \boldsymbol{M}_1\right\|_{\max} \max_{i} \|\boldsymbol{M}_2 e_j\|_1 \\
&= \|\boldsymbol{M}_1\|_{\max} \|\boldsymbol{M}_2\|_1 .
\end{aligned}$$

To obtain the first inequality above we used Holder's inequality and the last equality follows from the definition of $||\cdot||_1$ norm. To get the second inequality, we use the observations that $\|\boldsymbol{M}_1\boldsymbol{M}_2\|_{\max} = \left\|\boldsymbol{M}_2^\top \boldsymbol{M}_1^\top\right\|_{\max}$. Now applying the first inequality to this expression we get the desired result. Similar techniques yield the other two inequalities. □

Lemma 2.7.7. *With probability at least $1-\delta$, we have*

$$\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1}(\boldsymbol{C}-\widehat{\boldsymbol{C}})^\top\right\|_{\max} \leq \frac{r^2}{2m}\left(\left\|\mathbf{W}^{-1}-\boldsymbol{W}^{-1}\right\|_{\max} + \left\|\boldsymbol{W}^{-1}\right\|_{\max}\right)\log(2Kr/\delta) +$$

$$r^2 \left\|\mathbf{W}^{-1} - \boldsymbol{W}^{-1}\right\|_{\max} \sqrt{\frac{\log(2Kr/\delta)}{2m}} + r \left\|\boldsymbol{C}\boldsymbol{W}^{-1}\right\|_1 \sqrt{\frac{\log(2Kr/\delta)}{2m}}$$

Proof.

$$\begin{aligned}
\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1}(\boldsymbol{C} - \widehat{\boldsymbol{C}})^\top\right\|_{\max} &\leq \left\|(\widehat{\boldsymbol{C}}\mathbf{W}^{-1} - \boldsymbol{C}\boldsymbol{W}^{-1} + \boldsymbol{C}\boldsymbol{W}^{-1})(\boldsymbol{C} - \widehat{\boldsymbol{C}})^\top\right\|_{\max} \\
&\overset{(a)}{\leq} \left\|(\widehat{\boldsymbol{C}}\mathbf{W}^{-1} - \boldsymbol{C}\boldsymbol{W}^{-1})(\boldsymbol{C} - \widehat{\boldsymbol{C}})^\top\right\|_{\max} + \left\|\boldsymbol{C}\boldsymbol{W}^{-1}(\boldsymbol{C} - \widehat{\boldsymbol{C}})^\top\right\|_{\max} \\
&\overset{(b)}{\leq} \left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1} - \boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max} \left\|(\boldsymbol{C} - \widehat{\boldsymbol{C}})^\top\right\|_1 + \left\|\boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max} \left\|(\boldsymbol{C} - \widehat{\boldsymbol{C}})^\top\right\|_1
\end{aligned} \tag{2.27}$$

To obtain inequality (a) we used triangle inequality for matrix norms, and to obtain inequality (b) we used Proposition (2.7.6). We next upper bound the first term in the R.H.S. of Equation (2.27).

We bound the term $\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1} - \boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max}$ next.

$$\begin{aligned}
\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1} - \boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max} &\leq \left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1} - \boldsymbol{C}\mathbf{W}^{-1} + \boldsymbol{C}\mathbf{W}^{-1} - \boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max} \\
&\leq \left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1} - \boldsymbol{C}\mathbf{W}^{-1}\right\|_{\max} + \left\|\boldsymbol{C}\mathbf{W}^{-1} - \boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max} \\
&= \left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})\mathbf{W}^{-1}\right\|_{\max} + \left\|\boldsymbol{C}(\mathbf{W}^{-1} - \boldsymbol{W}^{-1})\right\|_{\max} \\
&\overset{(a)}{\leq} \left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})^\top\right\|_1 \left\|\mathbf{W}^{-1}\right\|_{\max} + \left\|\boldsymbol{C}^\top\right\|_1 \left\|\mathbf{W}^{-1} - \boldsymbol{W}^{-1}\right\|_{\max}
\end{aligned} \tag{2.28}$$

We used Proposition (2.7.6) to obtain inequality (a). Combining equations (2.27) and (2.28) we get,

$$\begin{aligned}
\left\|\widehat{\boldsymbol{C}}\mathbf{W}^{-1}(\boldsymbol{C} - \widehat{\boldsymbol{C}})^\top\right\|_{\max} &\leq \left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})^\top\right\|_1 \Big(\left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})^\top\right\|_1 \left\|\mathbf{W}^{-1}\right\|_{\max} \\
&\quad + \left\|\boldsymbol{C}^\top\right\|_1 \left\|\mathbf{W}^{-1} - \boldsymbol{W}^{-1}\right\|_{\max} + \left\|\boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max}\Big) \\
&= \left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})^\top\right\|_1^2 \left\|\mathbf{W}^{-1}\right\|_{\max}
\end{aligned}$$

$$+ \left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})^\top\right\|_1 \left\|\boldsymbol{C}^\top\right\|_1 \left\|\mathbf{W}^{-1} - \boldsymbol{W}^{-1}\right\|_{\max}$$
$$+ \left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})^\top\right\|_1 \left\|\boldsymbol{C}\boldsymbol{W}^{-1}\right\|_{\max} \tag{2.29}$$

Since all the entries of the matrix $\boldsymbol{C}$ are probabilities we have $\|\boldsymbol{C}\|_{\max} \leq 1$ and $\left\|\boldsymbol{C}^\top\right\|_1 \leq r$. Moreover, since each entry of the matrix $\hat{\boldsymbol{C}} - \boldsymbol{C}$ is the average of m independent random variables with mean 0, and each bounded between $[-1, 1]$, by Hoeffding's inequality and union bound, we get that with probability at least $1 - \delta$

$$\left\|(\widehat{\boldsymbol{C}} - \boldsymbol{C})^\top\right\|_1 \leq r\sqrt{\frac{\log(2Kr/\delta)}{2m}} \tag{2.30}$$

□

The next proposition takes the first steps towards obtaining an upper bound on $\left\|\hat{C}(W^{-1} - \hat{W}^{-1})C^\top\right\|_{\max}$

Proposition 2.7.8.

$$\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1} - \mathbf{W}^{-1})\boldsymbol{C}^\top\right\|_{\max} \leq \min\left\{r^2\left\|\boldsymbol{W}^{-1} - \mathbf{W}^{-1}\right\|_{\max}, r\left\|\boldsymbol{W}^{-1} - \mathbf{W}^{-1}\right\|_1\right\}$$

Proof.

$$\begin{aligned}\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1} - \mathbf{W}^{-1})\boldsymbol{C}^\top\right\|_{\max} &\overset{\text{(a)}}{\leq} \left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1} - \mathbf{W}^{-1})\right\|_{\max}\left\|\boldsymbol{C}^\top\right\|_1 \\ &\overset{\text{(b)}}{\leq} r\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1} - \mathbf{W}^{-1})\right\|_{\max} \\ &\overset{\text{(c)}}{\leq} \min\left\{r^2\left\|\boldsymbol{W}^{-1} - \mathbf{W}^{-1}\right\|_{\max}, r\left\|\boldsymbol{W}^{-1} - \mathbf{W}^{-1}\right\|_1\right\}\end{aligned} \tag{2.31}$$

In the above bunch of inequalities (a) and (c) we used Proposition (2.7.6) and to obtain inequality (b) we used the fact that $||\boldsymbol{C}^\top||_{\max} \leq r$. □

Hence, we need to bound $\left\|\boldsymbol{W}^{-1} - \mathbf{W}^{-1}\right\|_{\max}$ and $\left\|\boldsymbol{W}^{-1} - \mathbf{W}^{-1}\right\|_1$.

Let us define $\hat{\boldsymbol{W}} = \boldsymbol{W} + \boldsymbol{E}_W$ where $\boldsymbol{E}_W$ is the error-matrix and $\hat{\boldsymbol{W}}$ is the sample average of m independent samples of a random matrix where $\mathbb{E}\hat{\boldsymbol{W}}_k(i,j) = \boldsymbol{W}(i,j)$.

Lemma 2.7.9. *Let us define $\hat{\boldsymbol{W}} - \boldsymbol{W} = \boldsymbol{E}_W$. Suppose, $\left\|\boldsymbol{W}^{-1}\boldsymbol{E}_W\right\|_2 \leq \frac{1}{2}$, then*

$$\left\|\hat{\boldsymbol{W}}^{-1} - \boldsymbol{W}^{-1}\right\|_{\max} \leq 2\left\|\boldsymbol{W}^{-1}\right\|_2 \|\boldsymbol{E}_W\|_2 \left\|\boldsymbol{W}^{-1}\right\|_{\max}$$

.

Proof. Since $\left\|\boldsymbol{W}^{-1}\boldsymbol{E}_W\right\|_2 < 1$, we can apply the Taylor series expansion:

$$(\boldsymbol{W} + \boldsymbol{E}_W)^{-1} = \boldsymbol{W}^{-1} - \boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1} + \boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1} + \cdots$$

Therefore:

$$\begin{aligned}
\left\|\hat{\boldsymbol{W}}^{-1} - \boldsymbol{W}^{-1}\right\|_{\max} &= \left\|\boldsymbol{W}^{-1} - \boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1} + \boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1} + \cdots - \boldsymbol{W}^{-1}\right\|_{\max} \\
&\overset{(a)}{\leq} \left\|\boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1}\right\|_{\max} + \left\|\boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1}\boldsymbol{E}_W\boldsymbol{W}^{-1}\right\|_{\max} + \cdots \\
&\overset{(b)}{\leq} \left\|\boldsymbol{W}^{-1}\boldsymbol{E}_W\right\|_2 \left\|W^{-1}\right\|_{\max} + \left\|\boldsymbol{W}^{-1}\boldsymbol{E}_W\right\|_2^2 \left\|W^{-1}\right\|_{\max} + \ldots \\
&\overset{(c)}{\leq} 2\left\|\boldsymbol{W}^{-1}\right\|_2 \|\boldsymbol{E}_W\|_2 \left\|\boldsymbol{W}^{-1}\right\|_{\max}
\end{aligned}$$

To obtain the last inequality we used the hypothesis of the lemma, and to obtain inequality (a) we used the triangle inequality for norms, and to obtain inequality (b) we used proposition (2.7.6). Inequlaity (c) follows from the triangle inequality. □

Thanks to Lemma (2.7.9) and proposition (2.7.8) we know that $\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1} - \mathbf{W}^{-1})\boldsymbol{C}^\top\right\|_{\max} \leq r^2\epsilon$. We now need to guarantee that the hypothesis of lemma (2.7.9) applies. The next lemma helps in doing that.

Lemma 2.7.10. *With probability at least* $1-\delta$ *we have*

$$\|\boldsymbol{E}_W\| = \left\|\hat{\boldsymbol{W}} - \boldsymbol{W}\right\| \leq \frac{2r}{3m}\log(2r/\delta) + \sqrt{\frac{r\log(2r/\delta)}{2m}} \tag{2.32}$$

Proof. The proof is via matrix Bernstein inequality. By the definition of $\hat{\boldsymbol{W}}$, we know that $\hat{\boldsymbol{W}} - \boldsymbol{W} = \frac{1}{m}\sum(\boldsymbol{W}_i - \boldsymbol{W})$, where $\hat{\boldsymbol{W}}$ is $0-1$ random matrix where the $(i,j)^{\text{th}}$ entry of the matrix $\hat{\boldsymbol{W}}$ is a single Bernoulli sample sampled from $\mathrm{Bern}(\boldsymbol{W}_{i,j})$. For notational convenience denote $Z_i := \frac{1}{m}\hat{\boldsymbol{W}}_i - \boldsymbol{W}$ This makes $\hat{\boldsymbol{W}} - \boldsymbol{W} = \frac{1}{m}\sum \boldsymbol{W}_i - \boldsymbol{W}$ an average of m independent random matrices each of whose entry is a 0 mean random variable with variance at most $1/4$, with each entry being in $[-1,1]$. In order to apply the matrix Bernstein inequality we need to upper bound ν, L (see Theorem (2.7.2)), which we do next.

$$\left\|\frac{1}{m}(\hat{\boldsymbol{W}}_i - \boldsymbol{W})\right\|_2 \leq \frac{1}{m}\sqrt{r^2} = \frac{r}{m}. \tag{2.33}$$

In the above inequality we used the fact that each entry of $(\hat{\boldsymbol{W}}_i - \boldsymbol{W})$ is between $[-1,1]$ and hence the spectral norm of this matrix is at most $\sqrt{r^2}$. We next bound the parameter ν.

$$\nu = \frac{1}{m^2}\max\left\{\left\|\sum_i \mathbb{E}\boldsymbol{Z}_i\boldsymbol{Z}_i^\top\right\|, \left\|\sum_i \mathbb{E}\boldsymbol{Z}_i^\top\boldsymbol{Z}_i\right\|\right\} \tag{2.34}$$

It is not hard to see that the matrix $\mathbb{E}\boldsymbol{Z}_i\boldsymbol{Z}_i^\top$ is a diagonal matrix, where each diagonal entry is at most $\frac{l}{4}$. The same holds true for $\mathbb{E}\boldsymbol{Z}_i\boldsymbol{Z}_i^\top$. Putting this back in Equation (2.34) we get $\nu \leq \frac{r}{4m}$. Putting $L = \frac{r}{m}$ and $\nu = \frac{r}{4m}$, we get

$$\left\|\hat{\boldsymbol{W}} - \boldsymbol{W}\right\| \leq \frac{2r}{3m}\log(2r/\delta) + \sqrt{\frac{r\log(2r/\delta)}{2m}} \tag{2.35}$$

□

We are now ready to establish the following bound

Lemma 2.7.11. *Assuming that* $m \geq m_0 := \frac{4r\left\|\boldsymbol{W}^{-1}\right\|}{3} + 2r\log(2r/\delta)\left\|\boldsymbol{W}^{-1}\right\|_2^2$, *with probability at least* $1-\delta$ *we will have*

$$\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1} - \mathbf{W}^{-1})\boldsymbol{C}^\top\right\|_{\max} \leq 2r^2\left\|\boldsymbol{W}^{-1}\right\|_2\left\|\boldsymbol{W}^{-1}\right\|_{\max}\left(\frac{2r}{3m}\log(2r/\delta) + \sqrt{\frac{r\log(2r/\delta)}{2m}}\right). \tag{2.36}$$

Proof.

$$\begin{aligned}
\left\|\widehat{\boldsymbol{C}}(\boldsymbol{W}^{-1} - \mathbf{W}^{-1})\boldsymbol{C}^\top\right\|_{\max} &\overset{\text{(a)}}{\leq} r^2\left\|\boldsymbol{W}^{-1} - \hat{\boldsymbol{W}}^{-1}\right\|_{\max} \\
&\overset{\text{(b)}}{\leq} 2r^2\left\|\boldsymbol{W}^{-1}\boldsymbol{E}_{\boldsymbol{W}}\right\|_2\left\|\boldsymbol{W}^{-1}\right\|_{\max} \\
&\overset{\text{(c)}}{\leq} 2r^2\left\|\boldsymbol{W}^{-1}\right\|_2\left\|\boldsymbol{E}_{\boldsymbol{W}}\right\|_2\left\|\boldsymbol{W}^{-1}\right\|_{\max} \\
&\overset{\text{(d)}}{\leq} 2r^2\left\|\boldsymbol{W}^{-1}\right\|_2\left\|\boldsymbol{W}^{-1}\right\|_{\max}\left(\frac{2r}{3m}\log(2r/\delta) + \sqrt{\frac{r\log(2r/\delta)}{2m}}\right)
\end{aligned}$$

To obtain inequality (a) above we used proposition (2.7.8), to obtain inequality (b) we used lemma (2.7.9), and finally to obtain inequality (c) we used the fact that matrix 2-norms are submultiplicative.

With this we now have bounds on all the necessary quantities. The proof of our theorem essentially requires us to put all these terms together.

2.7.6 Proof of Theorem 4.3

Since we need the total error in $\max$ norm to be at most ϵ, we will enforce that each term of our expression be atmost $\frac{\epsilon}{10}$. From lemma (2.7.4) we know that the maxnorm is the sum of three terms. Let us call the three terms in the R.H.S. of Lemma (2.7.4) T_1, T_2, T_3 repsectively. We then have that if we have m_1 number of copies of the matrix

$\boldsymbol{C}$, where

$$m_1 \geq \frac{20 \left\| \boldsymbol{W}^{-1}\boldsymbol{C}^\top \right\|_{\max} \log(2Kr/\delta)}{3\epsilon} \bigwedge \frac{100r \left\| \boldsymbol{W}^{-1}\boldsymbol{C}^\top \right\|_{\max}^2 \log(2Kr/\delta)}{2\epsilon^2} \tag{2.37}$$

then $T_1 \leq \epsilon/5$. Next we look at T_3. From lemma (2.7.11) it is easy to see that we need m_3 independent copies of the matrix $\boldsymbol{W}$ so that $T_3 \leq \epsilon/5$, where m_3 is equal to

$$m_3 \geq \frac{40r^3 \left\| \boldsymbol{W}^{-1} \right\|_2 \left\| \boldsymbol{W}^{-1} \right\|_{\max} \log(2r/\delta)}{3\epsilon} \bigwedge \frac{400r^5 \left\| \boldsymbol{W}^{-1} \right\|_2^2 \left\| \boldsymbol{W}^{-1} \right\|_{\max}^2 \log(2r/\delta)}{2\epsilon^2} \tag{2.38}$$

Finally we now look at T_2. Combining lemma (2.7.7), and lemma (2.7.9) and (2.7.10) and after some elementary algebraic calculations we get that we need m_2 independent copies of the matrix $\boldsymbol{C}$ and $\boldsymbol{W}$ to get $T_2 \leq \frac{3\epsilon}{5}$, where m_2 is

$$m_2 \geq 100 \max(\left\| \boldsymbol{W}^{-1} \right\|_{\max}, \left\| \boldsymbol{C}\boldsymbol{W}^{-1} \right\|_1^2, \left\| \boldsymbol{W}^{-1} \right\|_2 \left\| \boldsymbol{W}^{-1} \right\|_{\max}) \log(2Kr/\delta) \left(\frac{r^{5/2}}{\epsilon}, \frac{r^2}{\epsilon^2} \right) \tag{2.39}$$

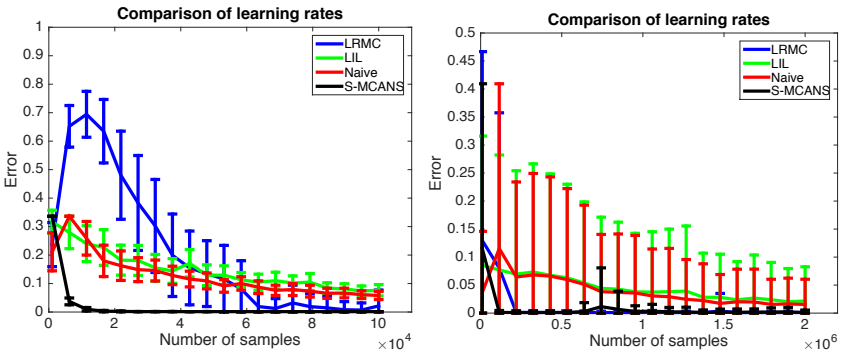
The number of calls to stochastic oracle is $r^2(m_0 + m_3) + Kr(m_1 + m_2)$, where m_0 is the number as stated in Lemma (2.7.11). Using the above derived bounds for $m_0 + m_1, m_2, m_3$ we get

$$Kr(m_1 + m_2) + r^2(m_0 + m_3) \geq 100 \log(2Kr/\delta) C_1(W, C) \max \left(\frac{Kr^{7/2}}{\epsilon}, \frac{Kr^3}{\epsilon^2} \right) + 200 C_2(W, C) \log(2r/\delta) \max \left(\frac{r^5}{\epsilon}, \frac{r^7}{\epsilon^2} \right)$$

where $C_1(\boldsymbol{W}, \boldsymbol{C})$ and $C_2(\boldsymbol{W}, \boldsymbol{C})$ are given by the following equations

$$C_1(\boldsymbol{W}, \boldsymbol{C}) = \max \left(\left\| \boldsymbol{W}^{-1}\boldsymbol{C}^\top \right\|_{\max}, \left\| \boldsymbol{W}^{-1}\boldsymbol{C}^\top \right\|_{\max}^2, \left\| \boldsymbol{W}^{-1} \right\|_{\max}, \left\| \boldsymbol{C}\boldsymbol{W}^{-1} \right\|_1^2, \left\| \boldsymbol{W}^{-1} \right\|_2 \left\| \boldsymbol{W}^{-1} \right\|_{\max} \right)$$

$$C_2(\boldsymbol{W}, \boldsymbol{C}) = \max \left(\left\| \boldsymbol{W}^{-1} \right\|_2^2 \left\| \boldsymbol{W}^{-1} \right\|_{\max}^2, \left\| \boldsymbol{W}^{-1} \right\|_2 \left\| \boldsymbol{W}^{-1} \right\|_{\max}, \left\| \boldsymbol{W}^{-1} \right\|_2, \left\| \boldsymbol{W}^{-1} \right\|_2^2 \right)$$



(a) Synthetic; $(80, 2)$ (b) Synthetic; $(400, 2)$

Figure 2.3: Error of various algorithms with increasing budget. Numbers in the brackets represent values for (K, r). The error is defined as $\boldsymbol{L}_{\hat{i},\hat{j}} - \boldsymbol{L}_{i_\star,j_\star}$ where $(\hat{i}, \hat{j})$ is a pair of optimal choices as estimated by each algorithm.

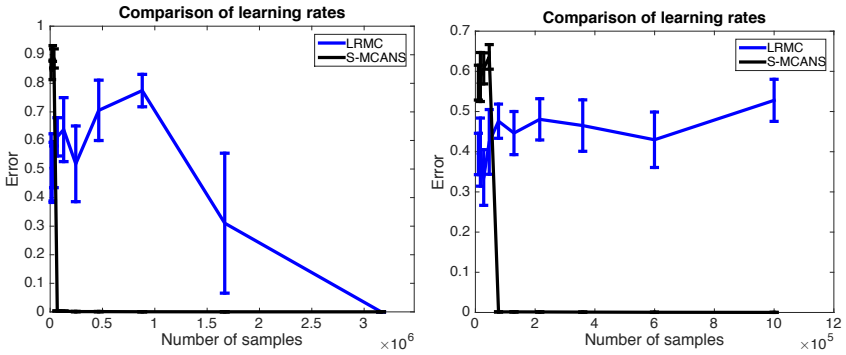
2.7.7 Additional experimental results: Comparison with LRMC on Movie Lens datasets

First we present the results on the synthetic dataset. To generate a low-rank matrix, we take a random matrix in $\boldsymbol{L}_1 = [0, 1]^{K \times r}$ and then define $\boldsymbol{L}_2 = \boldsymbol{L}_1 \boldsymbol{L}_1^\top$. Then get $\boldsymbol{L} = \boldsymbol{L}_2 / \max_{i,j}(\boldsymbol{L}_2)_{i,j}$. This matrix $\boldsymbol{L}$ will be $K \times K$ and have rank r.

In Figure 2.4, you can find the comparison of LRMC and S-MCANS on the ML-100K dataset.

2.7.8 Further discussion and related work

Bandit problems where multiple actions are selected have also been considered in the past. Kale et al. (2010) consider a setup where on choosing multiple arms the reward obtained is the sum of the rewards of the chosen arms, and the reward of each chosen arm is revealed to the algorithm. Both these works focus on obtaining guarantees on the cumulative regret compared to the best set of arms in hindsight. Radlinski et al. (2008) consider a problem, in the context of information retrieval, where multiple bandit arms are chosen and the reward obtained is the maximum of the rewards corresponding to



(a) ML-100K; $(200, 2)$ (b) ML-100K; $(200, 4)$

Figure 2.4: Error of LRMC and S-MCANS algorithms with increasing budget. Numbers in the brackets represent values for (K, r). The error is defined as $\boldsymbol{L}_{\hat{i},\hat{j}} - \boldsymbol{L}_{i_\star,j_\star}$ where $(\hat{i}, \hat{j})$ is a pair of optimal choices as estimated by each algorithm.. This is for the ML-100K dataset

the chosen arms. Apart from this reward information the algorithm also gets a feedback that tells which one of the chosen arms has the highest reward. Similar models have also been studied in Streeter and Golovin (2009) and Yue and Guestrin (2011). A major difference between the above mentioned works and our work is the feedback and reward model and the fact that we are not interested in regret guarantees but rather in finding a good pair of arms as quickly as possible. Furthermore our linear-algebraic approach to the problem is very different from previous approaches which were either based on multiplicative weights (Kale et al., 2010) or online greedy submodular maximization (Streeter and Golovin, 2009; Yue and Guestrin, 2011; Radlinski et al., 2008). Simchowitz et al. (2016) also consider similar subset selection problems and provide algorithms to identify the top set of arms. In the Web search literature click models have been proposed to model user behaviour (Guo et al., 2009; Craswell et al., 2008) and a bandit analysis of such models have also been proposed (Kveton et al., 2015). However, these models assume that all the users come from a single population and tend to use richer information in their formulations (for example information about which exact link was clicked). Finally we would like the Model 5 earlier in this chapter bears resemblance, on the surface, to dueling bandit problems (Yue et al., 2012a).

However, in dueling bandits two arms are compared which is not the case in the bandit problem that we study. Interactive collaborative filtering (CF) and bandit approaches to such problems have also been investigated (Kawale et al., 2015). Though, the end goal in CF is different from our goal in this work.

Chapter 3

Image Search

3.1 Introduction

Modern sensors are collecting imaging data at unprecedented volume and speed. It is not uncommon for NASA satellite systems to collect hundreds of terabytes of images every hour, and modern microscopy systems generate large field-of-view time-lapse images of dynamic biological systems. In these and other imaging settings, the data is neither thoroughly annotated nor meaningfully catalogued. If the user of such data finds an interesting "target" image in her database, she often has no simple mechanism for efficiently sifting through her database to find similar examples. Ultimately we desire a system where a human expert may input an example image (e.g., an "interesting" galaxy from a large scale sky survey) and ask for a list of other similar images. In general, different human experts engaged in different tasks may have varying notions of similarity between images.

This chapter proposes a novel system for interactive image search, in which our system interacts with the human expert to home in on the right direction in the image feature space to enable personalized search results. Specifically, the user provides her initial target image to the system as an example, and the system presents the user with a series of images from the database, using feedback from the human expert on whether each successive image is relevant to her search; see Figure 3.1 for a precise protocol. Such a system is in stark contrast to traditional search systems passively retrieve images which are close to a query image in feature space Jégou et al. (2008, 2010); Jain et al. (2008); Kulis et al. (2009); Kulis and Grauman (2009). Depending on the problem context or the user's preference, some features are more relevant to the search than others, and the importance of each feature must be learned from user feedback on the fly.

The goal of our system is to present the user with as many relevant images as possible within a short timeframe due to users' limited capacity for examining images

Initialization: the user inputs an initial target image $\mathbf{x}_0$ to the system.
At each time $t = 1, 2, \ldots, T$,

- The system chooses an image $\mathbf{x}_t$ from the database and shows it to the user.
- The user provides feedback $y_t \in [-1, 1]$ indicating the relevance of $\mathbf{x}_t$.
- The system incorporates the feedback $(\mathbf{x}_t, y_t)$.

Figure 3.1: Interactive Image Search Protocol

and providing feedback. To achieve this goal we must balance between presenting the user with a diverse collection of images to better assess the user's preference and presenting the user with images the algorithm predicts as most likely to be relevant to her search. We model the relevance of an image to a user as a noisy measurement of some unknown linear combination of the image's d features that are extracted by a deep learning framework Caffe Jia et al. (2014a). Note that this is an instance of the linear stochastic multi-armed bandit (MAB) problem that has gained popularity in both theory Auer and Long (2002); Dani et al. (2008); Abbasi-Yadkori et al. (2011); Agrawal and Goyal (2013) and applications Li et al. (2010); Yue et al. (2012b). Each "arm" corresponds to a different image (and associated image feature vector) from a database. The performance of a MAB algorithm is analyzed by a quantity called "regret" that measures excess cumulative reward an algorithm could have received if it knew the best choices; we precisely define regret in (3.2). The smaller the regret an algorithm has, the more cumulative rewards (positive feedback) an algorithm receives. We elaborate on our mathematical formulation of the system in Section 3.3.

Existing linear MAB algorithms, however, cannot be applied directly to our large-scale system for two reasons:

- First, the real-time constraint disallows any algorithm with a linear dependency on the number N of items in the database. Specifically, when an extremely large

collection of images is stored in distributed clusters, reading every image from the disks itself is already impractical.

- Second, existing algorithms cannot incorporate the initial target image $\mathbf{x}_0$ which is valuable information for guiding search.

For the first issue, most practical linear MAB algorithms needs modification since they have linear dependency on N; see Table 3.1. Hashing (specifically, locality-sensitive or maximum inner product search hashing) is commonly used to facilitate retrieving elements from a large database in response to a query in sublinear time in N. These methods assume that the query will be of the form "Find the image that is most similar to my target image" or "Find the image that maximize the inner product with a given vector". Unfortunately, many linear MAB algorithms choose the next arm (i.e., image from the database) by solving an optimization problem over the collection of arms, and this optimization problem cannot generally be expressed as the kind of nearest neighbor query described above. In this case, we say that the linear MAB algorithm is not *hash-amenable*.

To this end, we propose a new algorithm Quadratic Optimism in the Face of Uncertainty for Linear rewards (QOFUL) that can be combined with Maximum Inner Product Hashing (MIPS) hashing algorithms Shrivastava and Li (2014); Neyshabur and Srebro (2015) to enjoy sublinear time in N. We show that QOFUL has an improved regret bound upon linear Thompson sampling (LTS), the state-of-the-art hash-amenable algorithm. We summarize the regret bounds of various algorithms in Table 3.1. We also prove a theorem that states the same order of guarantee as QOFUL holds true even with the presence of an approximation error from the hashing when the target time horizon T is fixed a priori.

We solve the second issue by proposing a novel method called *shifted estimation* that exploits $\mathbf{x}_0$ to improve search. Being a theoretically well-justified alternative to the standard ridge regression estimator, shifted estimation can be applied to not only

Algorithms	Regret	Hash-amenable	Time
OFUL Abbasi-Yadkori et al. (2011)	$\tilde{O}(d\sqrt{T})$	✗	Nd^2
Rarely-switching OFUL Abbasi-Yadkori et al. (2011)	$\tilde{O}(d\sqrt{T})$	✗	$d^2 + Nd + Nd^2(\log T)/T$
LTS Agrawal and Goyal (2013)	$\tilde{O}(d^{3/2}\sqrt{T})$	✓	$d^2 + Nd$
QOFUL (ours)	$\tilde{O}(d^{5/4}\sqrt{T})$	✓	Nd^2

Table 3.1: Linear MAB algorithms. T is the time horizon and d is the image feature vector dimension. We also present the per-iteration time complexity in order notation. Our proposed algorithm QOFUL achieves the smallest regret among hash-amenable algorithms.

QOFUL but also many other popular algorithms such as OFUL Abbasi-Yadkori et al. (2011) and LTS. Our empirical study shows that our biasing technique significatly increases the search quality.

Finally, we evaluate the candidate algorithms by running experiments on the UT Zappos50k dataset Yu and Grauman (2014). First, we ran simulations using labels given in the dataset. Here, the goal was to compare their cumulative rewards and running times for different starting points. This allows us to observe how well the running times and rewards compared to their theoretical guarantees. This also lets us tune the different parameters of each algorithm in a controlled environment.

We then implemented some of the algorithms in the NEXT Jamieson et al. (2015) framework. Using this system we collect data from people using the Amazon Mechanical Turk as well as by emailing the survey to users at our university. This allowed us to compare the algorithms in a real world environment. This brings to fore some of the challenges associated with deploying MAB algorithms.

3.2 Related Work

In recent literature, a lot of effort has been made to improve the scalability and efficacy of image search in large-scale databases Jain et al. (2008); Jégou et al. (2008, 2010); Kulis and Grauman (2009); Kulis et al. (2009); Hadi Kiapour et al. (2015). These search systems focus on learning a representation or similarity measure (e.g., Mahalanobis metric Jain et al. (2008)) so that a simple nearest neighbor search can return relevant images for users. The nearest neighbor search can then be conducted using locality sensitive hashing to speed up the searching process. However, these methods rely on a large training set of labeled images, and learn a metric or similarity measure based on these labels. Thus the methods cannot adapt to different users' specific tasks or preferences. For example, if two users initiate the search with a picture of red boots where one user focuses on the color and the other user focuses on the specific type of boots, non-interactive search systems would retrieve the same set of images whereas our interactive search system would quickly adapt to what aspect of the image the user likes to focus. We would like to point out that our interactive search system is complementary to these systems in that our system can be built on top of the trained representation or metric in many cases; e.g., incorporating Mahalanobis distance Jain et al. (2008) is only a matter of projecting an image vector.

There also exist studies that interact with users to guide the search. Kiapour *et al.* Hadi Kiapour et al. (2015) ask users to draw a rectangle in an image to provide the region of interest. However, the purpose is to identify the target item in an image rather than picking out certain features or aspects of the image such as colors and shapes. Kovashka *et al.* Kovashka et al. (2015) incorporate user feedback to tailor the image search process for individual users. The users are asked to compare their desired image with other images on a few pre-defined "attributes" (*e.g.,* in shoe searches, answer "is it much pointier than this shoe?" or "is it less feminine than this shoe?"). The authors

argue that answering the well-designed questions quickly reduces ambiguity and hence improves search results. However, the set of questions needs to be well-curated by human experts which can be costly. In contrast, in our interactive system users only need to answer whether each new image is relevant to their task or goal – a question that does not need to be curated by experts.

Deep neural networks are playing an increasingly large role in modern computer vision. Large, annotated databases such as ImageNet Deng et al. (2009) have allowed researchers to train deep networks that yield state-of-the-art classification results Krizhevsky et al. (2012); Sermanet et al. (2013); Szegedy et al. (2015); Simonyan and Zisserman (2014). However, a prevailing question in vision is how these results can be leveraged in other contexts with far fewer labeled training images. This problem is sometimes referred to as *transfer learning* or *domain adaptation* Pan and Yang (2010). Our proposed image search algorithm can be considered an efficient and interactive mechanism for domain adaptation. Specifically, we use the outputs of the second-to-last layer of a deep network trained on ImageNet as the feature vectors of our images. Our method then learns a model for the user's task base on these features, adapting the learned features to the user's current domain.

The first formal linear stochastic MAB analysis appears in Auer and Long (2002) but is under a more general family of problems. Dani *et al.* Dani et al. (2008) focus on the linear stochastic MAB case and proposed an algorithm based on confidence bounds with a regret bound that is later shown to be optimal up to logarithmic factors by Rusmevichientong and Tsitsiklis (2010). Abbasi-Yadkori *et al.* propose a significantly tighter confidence bound, which improves the regret bound slightly and the practical performance by orders of magnitude. The Linear Thompson Sampling (LTS) algorithm is a linear extension of the Thompson sampling that is known to perform very well in practice. The regret analysis of LTS was proposed as an open problem Chapelle and Li (2012) which is then solved by Agrawal and Goyal Agrawal and Goyal (2013). We

emphasize that, unlike our work, all these existing bandit studies have not attempted to improve the time complexity dependence on the number N of arms, to our knowledge.

3.3 Preliminaries: Interactive Image Search and Linear Bandits

Consider a database $\mathcal{X}$ of N images that are represented by d-dimensional feature vectors denoted by $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} \in \mathbb{R}^d$. Without loss of generality, assume that we have scaled every image $\mathbf{x} \in \mathcal{X}$ to have ℓ_2 norm at most 1. The interactive search session starts by a user who presents the system with an initial target image $\mathbf{x}_0 \in \mathbb{R}^d$. The system then proceeds in multiple rounds of interplay with the user. Define $\mathcal{X}_t := \mathcal{X} \backslash \{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ that is all the images except those already presented to the user. In each time step t, the system selects an image $\mathbf{x}_t$ in $\mathcal{X}_t$ and presents it to the user. The user labels the image with a value $y_t \in [-1, 1]$ where the sign indicates whether the image is relevant to her task (+) or not (-) and the magnitude indicates the strength of the feedback. Depending on the system interface design, one can ask a continuous value in $[-1, 1]$, a five-star rating where one star maps to -1 and five stars maps to 1, or rather simply ask a yes or no question that is mapped to 1 and -1 respectively. We summarize our protocol in Figure 3.1.

We model the feedback as an unknown, noisy linear function of x_t; specifically,

$$y_t = \langle \boldsymbol{\theta}_*, \mathbf{x}_t \rangle + \eta_t \ , \tag{3.1}$$

where η_t is a noise term that is R-sub-Gaussian conditioning on $\mathbf{x}_{1:t}$ and $\eta_{1:(t-1)}$, and $\boldsymbol{\theta}_* \in \mathbb{R}^d$ is an unknown weight vector reflecting how much different features of the images contribute to whether they are relevant to the users' task. It is easy to see that such a linear formulation fits our reward encoding of y_t. In the binary encoding, for example, one can postulate $y' := \langle \boldsymbol{\theta}_*, \mathbf{x}_t \rangle \in (-1, 1)$ and define the noise η_t to be $1 - y'$ with probability $(1 + y')/2$ and $-1 - y'$ with probability $(1 - y')/2$. The noise η_t is now conditionally R-sub-Gaussian for some $R < \infty$ since it is zero-mean and bounded.

If we knew $\boldsymbol{\theta}_*$, then we could easily choose images $\mathbf{x}_t \in \mathcal{X}_t$ which maximize the

likelihood of a user finding the image relevant — i.e. for which $\mathbb{E}[y_t \mid \mathbf{x}_t]$ is maximal. However, since $\boldsymbol{\theta}_*$ is unknown we must learn it. If we were to present our user with d images with linearly independent feature vectors, then we could estimate $\boldsymbol{\theta}_*$ accurately. However, d can easily be in the thousands or tens of thousands, exhausting the users' capacity for labeling images. Thus our aim is to present the user with a series of images with *both* allow us to estimate $\boldsymbol{\theta}^*$ and quickly yield a large number of relevant images. To accomplish this task, we propose using linear stochastic MABs.

Existing linear MAB algorithms provide analysis on their cumulative rewards up to time step T. Define $[T] := \{1, \ldots, T\}$. Instead of guaranteeing an absolute quantity on the cumulative rewards, these algorithms guarantee a relative quantity called *cumulative regret*, which measures the extra cumulative reward one could have received with the best choices $\mathbf{x}_{t,*} := \max_{\mathbf{x} \in \mathcal{X}_t} \langle \boldsymbol{\theta}_*, \mathbf{x} \rangle, \forall t \in [T]$:

$$\sum_{t=1}^{T} \langle \boldsymbol{\theta}_*, \mathbf{x}_{t,*} \rangle - \langle \boldsymbol{\theta}_*, \mathbf{x}_t \rangle \ , \tag{3.2}$$

which we refer to as simply *regret*.

We now review a state-of-the-art linear MAB algorithm and discuss why it cannot be blindly applied to our interactive image search system.

Optimism in the Face of Uncertainty for Linear Rewards (OFUL)

We introduce the algorithm OFUL Abbasi-Yadkori et al. (2011), the state-of-the-art linear bandit algorithm that is closely related to our proposed algorithm. Define $\mathbf{X}_t := [\mathbf{x}_1^\top; \cdots; \mathbf{x}_t^\top]$, $\mathbf{y}_t := [y_1; \cdots; y_t]$, and $\overline{\mathbf{V}}_t := \lambda \mathbf{I} + \sum_{s=1}^{t} \mathbf{x}_s \mathbf{x}_s^\top$. Let $\widehat{\boldsymbol{\theta}}_t := \overline{\mathbf{V}}_t^{-1} \mathbf{X}_t \mathbf{y}_t$ be the ridge regression estimator. Assume $||\boldsymbol{\theta}_*||_2 \leq S$ for some known $S > 0$. Let $\delta \in (0, 1)$ be the target failure rate. Define

$$\sqrt{\beta_t} := R \sqrt{2 \log \left(\frac{\det(\overline{\mathbf{V}}_t)^{1/2} \det(\lambda \mathbf{I})^{-1/2}}{\delta} \right)} + \sqrt{\lambda} S$$

and the confidence set at time t

$$C_t := \{\boldsymbol{\theta} \in \mathbb{R}^d : ||\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t||_{\overline{\mathbf{V}}_t} \leq \sqrt{\beta_t}\}.$$

We define the event that the sequence of confidence sets $C_t(\delta)$ traps $\boldsymbol{\theta}_*$:

$$E_1(\delta) := \{\forall t \in \mathbb{N}, \boldsymbol{\theta}_* \in C_t(\delta)\} .$$

By Theorem 2 of Abbasi-Yadkori et al. (2011), C_t traps $\boldsymbol{\theta}_*$ for all $t = 1, 2, \ldots$ with high probability: $\mathbb{P}(E_1(\delta)) \geq 1 - \delta$. Hereafter, we use notation C_t in place of $C_t(\delta)$ for brevity.

OFUL chooses which arm to pull by solving the following optimization problem:

$$(\tilde{\boldsymbol{\theta}}_t, \mathbf{x}_t^{\text{OFUL}}) := \arg \max_{\boldsymbol{\theta} \in C_{t-1}, \mathbf{x} \in \mathcal{X}_t} \langle \boldsymbol{\theta}, \mathbf{x} \rangle . \tag{3.3}$$

The idea is that the inner product $\langle \tilde{\boldsymbol{\theta}}_t, \mathbf{x}_t^{\text{OFUL}} \rangle$ becomes a high probability upper bound on $\langle \boldsymbol{\theta}_*, \mathbf{x}_{t,*} \rangle$, which is the largest expected reward one could possibly receive at time t. This is the key step of the regret analysis.

Define $||\mathbf{x}||_A := \sqrt{\mathbf{x}^\top A \mathbf{x}}$. One can fix $\mathbf{x}$ and find the maximizer $\tilde{\boldsymbol{\theta}}(\mathbf{x}) := \max_{\boldsymbol{\theta} \in \mathbf{C}_{t-1}} \langle \boldsymbol{\theta}, \mathbf{x} \rangle$ in a closed form using the Lagrangian method:

$$\tilde{\boldsymbol{\theta}}(\mathbf{x}) = \widehat{\boldsymbol{\theta}}_{t-1} + \sqrt{\beta_{t-1}} \cdot \frac{\overline{\mathbf{V}}_{t-1}^{-1} \mathbf{x}}{||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}}} . \tag{3.4}$$

To obtain $\mathbf{x}_t^{\text{OFUL}}$, we plug in $\boldsymbol{\theta} \leftarrow \tilde{\boldsymbol{\theta}}(\mathbf{x})$ in (3.3) to arrive at a simpler problem:

$$\mathbf{x}_t^{\text{OFUL}} = \arg \max_{\mathbf{x} \in \mathcal{X}_t} \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x} \rangle + \sqrt{\beta_{t-1}} \cdot ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}} , \tag{3.5}$$

which has a natural interpretation of balancing between the first "exploitation" term that encourages arms aligned with the current ridge regression estimator $\widehat{\boldsymbol{\theta}}_t$ and the second "exploration" term that encourages arms aligned with the directions that are probed less so far.

OFUL achieves the near-optimal regret $\tilde{O}(d\sqrt{T})$ Rusmevichientong and Tsitsiklis

(2010), which implies that OFUL will retrieve as large number of relevant images as one could hope for. However, OFUL must compute (3.5) for every arm $\mathbf{x} \in \mathcal{X}_t$. This results in $O(Nd^2)$ time complexity. In many domains, such as astronomy image databases in which terabytes of image data are collected every day in typical NASA missions Clavin (2013), the number of images is extremely large. As a result, reading all the images from the disk alone could be infeasible, especially when the data is stored in distributed storage systems. Even in a mildly large database where the computation can be performed in a reasonable time, if the system has to handle thousands of users, the system must reduce the computational overhead to save the operational cost.

Linear Bandits Towards Scalable Interactive Image Search

For the reasons above, one must turn to a method with computation sublinear in N. One approach is to simply search over a random subset of images to choose x_t at each round t. We refer to this as a naïve subsampling method. Resorting to a naïve subsampling approach could reduce the search performance drastically, as we show in our experiments in Section 3.6. Enjoying sublinear time in N without such a quality loss is the key to a successful interactive image search system. To this end, locality sensitive hashing (LSH) or Maximum inner product search (MIPS) hashing that work with Euclidean distance and inner product similarity, respectively, seem to be relevant to solving (3.5) while suffering some hashing error. However, it is unclear as to:

1. How to turn the objective function in (3.5) into a distance computation or an inner product.

2. How the hashing error affects the performance of the bandit algorithm.

Note that (i) is not straightforward because the term $||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}}$ in (3.5) involves a square root that makes it neither linear nor quadratic. We address (i) in Section 3.4 by proposing a new algorithm that is amenable to MIPS hash and analyze its regret guarantee. We introduce the MIPS hashing and address (ii) in Section 3.5.

3.3.1 Review of Existing MAB Algorithms

We review various bandit algorithms in the literature whose regret bounds and time complexity is summarized in Table 4.1.

Rarely-Switching OFUL (OFUL-Lazy) is a fast version of OFUL proposed by Abbasi-Yadkori et al. (2011). The idea is to perform the heavy OFUL optimization (3.5), which we call the *full update*, only for a few time steps t depending on the growth of $\det(\overline{\mathbf{V}}_t)$, and for the other time steps solve a simpler problem $\max_{\mathbf{x}\in\mathcal{X}_t} \langle \tilde{\boldsymbol{\theta}}_\tau, \mathbf{x}\rangle$, which we call the *lazy update*, where τ is the last time step (3.5) was solved and $\tilde{\boldsymbol{\theta}}_\tau$ is the solution of (3.3). The analysis of OFUL-Lazy shows that the regret bound is the same as OFUL upto constant factors and that the full update is performed $O(\log(T))$ times. Furthermore, computing $\overline{\mathbf{V}}_t^{-1}$ with rank-one update takes $O(d^2)$ time. Thus, the per-iteration time complexity is $O(d^2 + Nd(1 + d\log(T)/T))$.

Linear Thompson Sampling (LTS) is a linear extension of the Thompson sampling algorithm that is known to perform very well in practice. Departing from OFUL-based algorithms, LTS samples a vector $\bar{\boldsymbol{\theta}}_t$ from a multivariate normal distribution $\mathcal{N}(\widehat{\boldsymbol{\theta}}_t, v^2\overline{\mathbf{V}}_t^{-1})$ for some $v \in \mathbb{R}$ and then chooses the arm $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}_t} \langle \bar{\boldsymbol{\theta}}_t, \mathbf{x}\rangle$. The regret bound of LTS is shown to be $\tilde{O}(d^{3/2}\sqrt{T})$ by Agrawal and Goyal (2013). Note that the sampling can be performed in $O(d^2)$ per iteration by maintainig Cholesky decomposition of $\overline{\mathbf{V}}_t^{-1}$ with rank-one updates. Thus, the per-iteration time complexity of LTS is $O(d^2 + Nd)$.

3.4 Quadratic Optimism in the Face of Uncertainty for Linear Rewards (QOFUL)

We now describe our proposed algorithm called Quadratic Optimism in the Face of Uncertainty for Linear rewards (QOFUL) that can be used for the interactive search system with hashing. Define $r = \min_{\mathbf{x}\in\mathcal{X}} ||\mathbf{x}||_2$ and

$$m_{t-1} := \min_{\mathbf{x}:||\mathbf{x}||_2\in[r,1]} ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}} , \tag{3.6}$$

which is r times the square root of the smallest eigen value of $\overline{\mathbf{V}}_{t-1}^{-1}$. It is easy to see that $m_{t-1} \leq ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}}$ for all $\mathbf{x} \in \mathcal{X}$ using $\mathcal{X} \subseteq \{\mathbf{x} \in \mathbb{R}^d \mid ||\mathbf{x}||_2 \in [r, 1]\}$ and that $m_{t-1} \geq \frac{r}{\sqrt{t+\lambda}}$ using $||\mathbf{x}||_2 \leq 1$ and the definition of $\overline{\mathbf{V}}_{t-1}$. Let $c_1 > 0$ be a constant. At time t, QOFUL chooses the arm $\mathbf{x}_t^{\text{QOFUL}} :=$

$$\arg\max_{\mathbf{x}\in\mathcal{X}} \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}\rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} \cdot ||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}, \tag{3.7}$$

Notice that the objective function is now quadratic in $\mathbf{x}$, thus the name *Quadratic* OFUL.

The key property of QOFUL is that one can write down the objective function as an inner product as follows:

$$\begin{aligned} &\max_{\mathbf{x}\in\mathcal{X}} \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}\rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} \|\mathbf{x}\|^2_{\overline{\mathbf{V}}_{t-1}^{-1}} \\ &= \max_{\mathbf{x}\in\mathcal{X}} \left\langle \begin{pmatrix} \widehat{\boldsymbol{\theta}}_{t-1} \\ \text{vec}\left(\frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} \overline{\mathbf{V}}_{t-1}^{-1}\right) \end{pmatrix}, \begin{pmatrix} \mathbf{x} \\ \text{vec}(\mathbf{x}\mathbf{x}^\top) \end{pmatrix} \right\rangle \end{aligned} \tag{3.8}$$

where $\text{vec}(\mathbf{A}) := [\mathbf{A}_{\cdot 1}; \mathbf{A}_{\cdot 2}; \cdots; \mathbf{A}_{\cdot d}] \in \mathbb{R}^{d^2}$. Thus, one can use an existing maximum inner product search (MIPS) hashing to find an approximate maximizer in time sublinear in N. We elaborate on how we use MIPS in Section 3.5.

We present the regret bound of QOFUL (3.7) in Theorem 3.4.1. The key step of the proof is that the maximum of the QOFUL objective function (3.7) plus $c_1\beta^{3/4}m_{t-1}$ is a tight upper bound of the maximum of the OFUL objective function (3.5). We derive such a tight upper bound using the Lagrangian method. We present all the proofs in our supplementary material.

Theorem 3.4.1. *Assume* $\langle \boldsymbol{\theta}_*, \mathbf{x} \rangle \in [-1, 1], \forall \mathbf{x} \in \mathcal{X}$. *Then, with probability at least* $1 - \delta$ *the cumulative regret of QOFUL* (3.7) *after* T *time steps is*

$$\sum_{t=1}^{T} \langle \boldsymbol{\theta}_*, \mathbf{x}_{t,*} \rangle - \langle \boldsymbol{\theta}_*, \mathbf{x}_t \rangle = O\left(d^{5/4}\sqrt{T}\log^{5/4}(T) \right) .$$

Thus, the regret bound of QOFUL is the best among hash-amenable algorithms, specifically an improvement over that of LTS; see Table 4.1.

Besides the regret guarantee, QOFUL has an attractive characteristic. Note that, in practice, existing bandit algorithms like OFUL or LTS usually perform exploration much more than necessary, so one often enforces more exploitation by multiplying a small constant less than 1 to $\sqrt{\beta_t}$; e.g., see Yue et al. (2012b). Applying such a trick is theoretically not justified and foregoes the regret guarantee, so a practitioner must take a leap of faith. The same can be done in QOFUL by using a large c_1. The difference is, however, that such an encouragement on exploitation does not go beyond the theoretical guarantee. As one can see from the proof of Theorem 3.4.1, c_1 balances between some two large terms where the best value is not known in general[1]. As long as c_1 is independent of T and d, the regret bound remains the same up to constant factor.

[1] In our experience, $c_1 = 4$ works well.

3.5 Fast QOFUL using Maximum Inner Product Search (MIPS) Hashing

The key to a successful interactive image search lies in fast computation, since the whole point is to incorporate user feedback in real time. Specifically, we would like to enjoy sublinear time in the number of images N in the database. QOFUL's objective function (3.7) can be written as an inner product of a query vector and a transformation of $\mathbf{x}$ as noted in (3.8). We write (3.8) compactly as $\max_{\mathbf{x}\in\mathcal{X}} \langle \mathbf{q}_t, \phi(\mathbf{x})\rangle$, where $\mathbf{q}_t := [\widehat{\boldsymbol{\theta}}_{t-1}; \text{vec}(\frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} \overline{\mathbf{V}}_{t-1}^{-1})]$ and $\phi(\mathbf{x}) := [\mathbf{x}; \text{vec}(\mathbf{x}\mathbf{x}^\top)]$. Such a reduction to an inner product maximization allows us to utilize existing approximate maximum inner product search (MIPS) algorithms and achieve query time sublinear in N.

Ultimately we would like a MIPS algorithm that has a guarantee on its approximation error for an input parameter c as follows:

Definition 1. *Let $\mathcal{X} \subseteq \mathbb{R}^d$ such that $|\mathcal{X}| < \infty$. A data point $\tilde{\mathbf{x}} \in \mathcal{X}$ is called c-MIPS w.r.t. a given query $\mathbf{q}$ if it satisfies $\langle \mathbf{q}, \tilde{\mathbf{x}}\rangle \geq c \cdot \max_{\mathbf{x}\in\mathcal{X}} \langle \mathbf{q}, \mathbf{x}\rangle$. An algorithm is called c-MIPS if, given a query $\mathbf{q} \in \mathbb{R}^d$, it retrieves $\mathbf{x} \in \mathcal{X}$ that is c-MIPS w.r.t. $\mathbf{q}$.*

Unfortunately, existing MIPS algorithms do not directly offer such a guarantee but instead provide a different and less convenient guarantee for an input parameter c and M as follows:

Definition 2. *Let $\mathcal{X} \subseteq \mathbb{R}^d$ such that $|\mathcal{X}| < \infty$. A data point $\tilde{\mathbf{x}} \in \mathcal{X}$ is called (c, M)-MIPS w.r.t. a given query $\mathbf{q}$ if it satisfies $\langle \mathbf{q}, \tilde{\mathbf{x}}\rangle \geq cM$. An algorithm is called (c, M)-MIPS if, given a query $\mathbf{q} \in \mathbb{R}^d$, it retrieves $\mathbf{x} \in \mathcal{X}$ that is (c, M)-MIPS w.r.t. $\mathbf{q}$ whenever there exists $\mathbf{x}' \in \mathcal{X}$ such that $\langle \mathbf{q}_t, \mathbf{x}'\rangle \geq M$.*

Note that when there is no such $\mathbf{x}$ that $\langle \mathbf{q}, \mathbf{x}\rangle \geq M$, retrieving any arbitrary vector in $\mathcal{X}$ is qualified as being (c, M)-MIPS. This also means that if the hashing returns a vector that is not (c, M)-MIPS w.r.t. $\mathbf{q}$, then there is no $\mathbf{x}$ such that $\langle \mathbf{q}, \mathbf{x}\rangle \geq M$ with high probability. Since a good value of M varies depending on $\mathbf{q}_t$, it seems nat-

ural to construct a c-MIPS hashing using multiple (c, M)-MIPS hashings with various M values that covers a wide range. Indeed, such a construction is available in Har-Peled et al. (2012) for locality-sensitive hashing based on the Euclidean distance but is arguably complicated.

We propose a significantly simpler construction thanks to the existence of a high-probability upper and lower bound on the maximum of the QOFUL objective (3.7) as we show in Lemma 3.5.1. Note that in the lemma we assume that $\max_{\mathbf{x}\in\mathbf{X}} \langle \boldsymbol{\theta}_*, \mathbf{x} \rangle \geq 1/2$. This is not a restrictive assumption since it means that there exists at least one item for which user gives positive feedback in expectation; if not, any algorithm would work almost equally bad since there are no relevant images in the first place. Depending on the system interface design, one can change $1/2$ any reasonable number for which the user's feedback is considered to be positive. Note that there exists a simple upper bound on $\sqrt{\beta_t}$:

$$\sqrt{\beta_t} \leq R\sqrt{d \log\left(\frac{1 + t/(d\lambda)}{\delta}\right)} + \sqrt{\lambda} S =: \sqrt{\overline{\beta}_t} \,, \tag{3.9}$$

which is due to (Abbasi-Yadkori et al., 2011, Lemma 10).

Lemma 3.5.1. *Assume $E_1(\delta)$ and $\max_{\mathbf{x}\in\mathbf{X}} \langle \boldsymbol{\theta}_*, \mathbf{x} \rangle \geq 1/2$. Suppose the target time horizon T is given. Then,*

$$M_{\min} := 1/2 \leq \max_{t\in[T]} \max_{\mathbf{x}\in\mathcal{X}} \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x} \rangle + \frac{\beta_{t-1}^{1/4}}{4 c_1 m_{t-1}} ||\mathbf{x}||^2_{\mathbf{V}_{t-1}^{-1}} \tag{3.10}$$

$$\leq \frac{2\sqrt{\overline{\beta}_{T-1}}}{\sqrt{\lambda}} + \overline{\beta}_{T-1}^{1/4} \cdot \frac{\sqrt{T+\lambda}}{4 c_1 r \lambda} := M_{\max} \,. \tag{3.11}$$

Given a target approximation rate $c_{\mathrm{H}} < 1$, we construct a c_{H}-MIPS as follows. Define $J := \left\lceil \log_{1/\sqrt{c_{\mathrm{H}}}} \frac{M_{\max}}{M_{\min}} \right\rceil$. We build a series of MIPS hashings

$$(c_{\mathrm{H}}^{1/2}, c_{\mathrm{H}}^{j/2} M_{\max})\text{-MIPS for } j \in [J] \,. \tag{3.12}$$

We say that the MIPS hashing *succeeds* (*fails*) for a query $\mathbf{q}$ if the retrieved vector is (not) (c, M)-MIPS w.r.t. $\mathbf{q}$. Theorem 3.5.2 shows that one can perform a binary search to find a vector $\mathbf{x} \in \mathcal{X}$ that is c_{H}-MIPS.

Theorem 3.5.2. *Upon given a query* $\mathbf{q}$*, perform a binary search over the* J *MIPS hashings* (3.12) *to find the smallest* $j^* \in [J]$ *for which the retrieved vector* $\mathbf{x}^{(j^*)}$ *from the* j^**-th hashing succeeds. Then,* $\mathbf{x}^{(j^*)}$ *is* c_{H}*-MIPS w.r.t.* $\mathbf{q}$ *with high probability.*

Among various (c, M)-MIPS algorithms Shrivastava and Li (2014, 2015); Neyshabur and Srebro (2015); Guo et al. (2016), we adopt Shrivastava *et al.* Shrivastava and Li (2014). Shrivastava *et al.* propose a reduction of MIPS to locality sensitive hashing (LSH) and present a result that their algorithm is (c, M)-MIPS with $O(N^{\rho^*} \log N)$ inner product computations and space $O(N^{1+\rho^*})$ for an optimized value ρ^* that is guaranteed to be less than 1; see (Shrivastava and Li, 2014, Theorem 5) for detail. We adopt their reduction with the LSH algorithm of Har-Peled et al. (2012).

Approximation error from MIPS hashing do not increase regret

For most image search applications, it seems reasonable to assume that the total number of steps T is bounded by a known constant (fixed budget in bandit terminology). We present the regret of QOFUL combined with MIPS hashing in Theorem 3.5.3. The theorem states that in the fixed budget setting T with the target approximation level c_{H} a function of T, we suffer the same order of regret as exactly solving the maximization problem (3.7).

Theorem 3.5.3. *Assume* $\langle \boldsymbol{\theta}_*, \mathbf{x} \rangle \in [-1, 1], \forall \mathbf{x} \in \mathcal{X}$*. Let* $T \geq 2$ *and* $c_{\mathrm{H}} = \left(1 + \frac{\log(T)}{\sqrt{T}}\right)^{-1}$*. Suppose we run QOFUL for* T *iterations where we invoke a* c_{H}*-MIPS algorithm to approximately find* $\mathbf{x}_t^{QOFUL}$ *for every* $t \in [T]$*. Assume that all the* T *queries to the* c_{H}*-MIPS algorithm successfully return vectors that are* c_{H}*-MIPS with probability at least*

$1-\delta$. *Then, with probability* $1-2\delta$*, the cumulative regret after* T *time steps is:*

$$\sum_{t=1}^{T} \langle \boldsymbol{\theta}_*, \mathbf{x}_{t,*} \rangle - \langle \boldsymbol{\theta}_*, \mathbf{x}_t \rangle = O(d^{5/4}\sqrt{T}\log^{5/4} T) \ .$$

Theorem 3.5.3 is the first theoretical result that combines a bandit with a hashing, to our knowledge. Note that we have made an assumption that all the queries are simultaneously c_{H}-MIPS with high probability. However, showing that the assumption holds true is hard since the number of possible trajectories of $\{\mathbf{q}_t\}$ is uncountably infinite in general. This is in contrast to the case where the set of queries is finite and fixed ahead, and so the union bound technique can be applied as shown in (Har-Peled et al., 2012, Lemma 3.6). Resolving the issue above is of theoretical interest and left as future work.

One can show that the choice of c_{H} in Theorem 3.5.3 implies that $J = O(\sqrt{T}(1 + \log(d)/\log(T))$. For $d = 1000$, $T = 50$, $\lambda = 1$, J is roughly 16.

Comparison to Other Hash-Amenable Algorithms

QOFUL requires $d + d^2$-dimensional MIPS hashing. Note that this does not mean we need to store vec$(\mathbf{x}\mathbf{x}^\top)$ since MIPS uses the vector $\mathbf{x}$ to compute its bucket index only. The time complexity of QOFUL together with the c_{H}-MIPS hashing described above is $O(N^{\rho^*}\log(N)\log(J)d^2)$ per iteration, where $\rho^* < 1$ depends on c_{H}. The extra space required by the c_{H}-MIPS algorithm consists of $O(JN^{\rho^*}\log(N))$ projection vectors of $(d + d^2)$-dimension and the hash tables $O(JN^{1+\rho^*})$ containing the buckets, which becomes total $O(JN^{\rho^*}(N + d^2\log(N)))$.

LTS is the other hash-amenable linear MAB algorithm that has a worse regret bound than QOFUL. However, since LTS requires d-dimensional hashing, its time complexity is $O(d + N^{\rho^*}\log(N)\log(J)d)$, which is a factor of d smaller than that of QOFUL. The extra space complexity of LTS is $O(JN^{\rho^*}(N + d\log(N)))$, which is smaller than that of QOFUL, too.

3.5.1 Biasing towards the Initial Target Image

Note that in bandit algorithms the choice of the first arm (image) is not specified and one can make an arbitrary choice. An obvious way to bias the search towards the initial target image $\mathbf{x}_0$ is to find the first retrieved image $\mathbf{x}_1$ by solving $\max_{\mathbf{x}\in\mathcal{X}} \langle \mathbf{x}, \mathbf{x}_0 \rangle$.

In this section, we propose a more effective mechanism for leveraging x_0 than this naïve approach We first describe how one can retrieve more relevant images when a good guess $\boldsymbol{\theta}_0$ on $\boldsymbol{\theta}_*$ is given, and then describe how to construct such a $\boldsymbol{\theta}_0$ based on the initial target image $\mathbf{x}_0$. Let $\boldsymbol{\theta}_0$ be a known vector that is close to $\boldsymbol{\theta}_*$. That is, $||\boldsymbol{\theta}_* - \boldsymbol{\theta}_0|| \leq S_0$ for some known $S_0 < S$. Define $\sqrt{\tilde{\beta}_t} := R\sqrt{2\log\left(\frac{\det(\overline{\mathbf{V}}_t)^{1/2}\det(\mathbf{V})^{-1/2}}{\delta}\right)} + \sqrt{\lambda}S_0$ and

$$\overline{\boldsymbol{\theta}}_t := \boldsymbol{\theta}_0 + \widehat{\boldsymbol{\theta}}_t \ , \tag{3.13}$$

which we call the biased ridge regression estimator. Departing from the confidence set C_t that is centered around $\widehat{\boldsymbol{\theta}}_t$, we propose to use $\widetilde{C}_t$ that is centered around $\overline{\boldsymbol{\theta}}_t$:

$$\widetilde{C}_t := \left\{ \boldsymbol{\theta} \in \mathbb{R}^d : ||\boldsymbol{\theta} - \overline{\boldsymbol{\theta}}_t||_{\overline{\mathbf{V}}_t} \leq \sqrt{\tilde{\beta}_t} \right\} \ .$$

It is easy to see that the ellipsoid $\widetilde{C}_t$ is strictly smaller than the ellipsoid C_t since $S_0 < S$.

Lemma 3.5.4 below shows that $\widetilde{C}_t$ traps $\boldsymbol{\theta}_*$ with high probability, and so one can replace C_t with $\widetilde{C}_t$ in OFUL and improve the regret bound by a constant factor. Such a change results in replacing $\widehat{\boldsymbol{\theta}}_t$ and β_t in (3.5) with $\overline{\boldsymbol{\theta}}_t$ and $\tilde{\beta}_t$, respectively. One can show that the same modification can be applied to (3.7) to improve the regret bound of QOFUL.

Lemma 3.5.4. *Suppose* $||\boldsymbol{\theta}_* - \boldsymbol{\theta}_0|| \leq S_0$. *Then,* $\boldsymbol{\theta}_* \in \widetilde{C}_t$ *with probability at least* $1 - \delta$.

Note that this idea can also be applied to LTS by changing the mean of the sampling

distribution from $\widehat{\boldsymbol{\theta}}_t$ (denoted by $\tilde{\mu}(t)$ in Agrawal and Goyal (2013)) to $\overline{\boldsymbol{\theta}}_t$.

In general, one does not have a good initial guess $\boldsymbol{\theta}_0$. For our interactive image search, however, the initial target image $\mathbf{x}_0$ can be used to construct $\boldsymbol{\theta}_0$. If we encode positive feedback as 1, it is reasonable to expect $\langle \boldsymbol{\theta}_*, \mathbf{x}_0 \rangle$ to be close to 1. We propose to set

$$\boldsymbol{\theta}_0 = \frac{\mathbf{x}_0}{||\mathbf{x}_0||_2^2} \tag{3.14}$$

since then $\langle \boldsymbol{\theta}_0, \mathbf{x}_0 \rangle = 1 \approx \langle \boldsymbol{\theta}_*, \mathbf{x}_0 \rangle$. Although this does not guarantee that $||\boldsymbol{\theta}_0 - \boldsymbol{\theta}_*||_2$ is small in general, we found that bandit algorithms based on estimator $\overline{\boldsymbol{\theta}}_t$ with $\boldsymbol{\theta}_0$ defined above successfully exploit the information in $\mathbf{x}_0$ and outperform other naive ways to incorporate $\mathbf{x}_0$ as we show in Section 3.6.

3.6 Empirical Study

We now perform empirical evaluations of various algorithms for the interactive search system.

3.6.1 Evaluation on Real-World Datasets

Dataset and Feature Extraction

We test the algorithms on the UT Zappos50k dataset Yu and Grauman (2014). UT Zappos50k is a large shoe image dataset with a total of 50,025 shoe images from Zappos.com. The dataset includes a variety of styles and types of footware. All shoes are pictured in the same orientation, and centered on the image with a white background.

In the implementation, we extract a 1000-dimensional feature vector for each image based on the convolutional neural network (CNN) proposed by Krizhevsky et al. Krizhevsky et al. (2012), which is a popular deep learning based feature that has shown good performance in areas like image classification and object identification tasks Russakovsky et al. (2015); Sharif Razavian et al. (2014). To generate the features, we first pass the UT Zappos50k images through the first seven layers of the Caffe implementation Jia et al. (2014b) of Krizhevsky et al. (2012), which yields a 4096-dimensional feature vector for each image. We then reduce the feature dimension down to 1000 by projecting the 4096-dimensional features onto the rank-1000 subspace spanned by the first thousand left-singular vectors (correspond to the largest thousand singular values) of the 4096×50025 feature matrix. Finally, we normalize the features so that each feature vector has unit ℓ_2 norm.

Tested Algorithms

We implement various algorithms for the interactive search system in Python. All the algorithms except for nearest neighbors use the biased search technique presented in

Section 3.5.1 unless indicated otherwise. Also, the first retrieved image of all the algorithms is the one that maximizes the inner product with $\mathbf{x}_0$ unless explained otherwise. We summarize our tested algorithms as follows:

- Nearest neighbor (NN): we simply find T images that are closest to the given initial target image $\mathbf{x}_0$ in Euclidean distance.
- OFUL and QOFUL: we play the arm defined in (3.5) and (3.7) respectively.
- LTS: see Section 3.3.1.
- {OFUL,QOFUL,LTS,OFUL,OFUL-Lazy}-Light: We implement a naïve-but-fast version where at every iteration t we restrict our attention to a fraction p subset $\mathcal{X}_t'$ of images drawn uniformly at random from $\mathcal{X}_t$ where $p \in \{0.01, 0.02, 0.05\}$. In OFUL, for example, we replace $\mathcal{X}_t$ with $\mathcal{X}_t'$ in (3.5).
- {QOFUL,LTS}-Hash: For hash-amenable algorithms QOFUL and LTS, we implement a version that combines the algorithm with a MIPS hashing (see below for detail). We expect to observe gains over -Light versions.
- OFUL-Lazy-Hash: a partially hash-amenable version OFUL-Lazy (described in Section 3.3.1 where the full updates are performed with subsampling as in -Light versions above, but the lazy updates are performed with a MIPS hashing as in -LTS versions above.
- {OFUL,QOFUL,LTS}-NBS1: As a comparison to our biased search technique, we try a naïve biased search (NBS) method where we retrieve the first image $\mathbf{x}_1$ by finding the closest image from $\mathbf{x}_0$ without further biasing afterwards.
- {OFUL,QOFUL,LTS}-NBS2: We try another naïve biased search where we treat $\mathbf{x}_0$ as the first retrieved image $\mathbf{x}_1$ with reward 1. This means that we shift the time index by one; the first retrieved image the user sees is $\mathbf{x}_2$ from the algorithm.

Our implementation of MIPS hashing is a modification of existing Python package OptimalLSH [2] that is written based on Slaney et al. (2012). We use binary binning of the projections with $k = 16$ keys and $\ell = 32$ tables. We use the multi-probe technique that explores nearby buckets Slaney et al. (2012), which has the effect of building a series of (c, M)-MIPS hashings to perform c-MIPS hashing. Note that QOFUL-Hash uses $(d + d^2)$-dimensional MIPS hashing, and LTS-Hash and OFUL-Lazy-Hash uses d-dimensional MIPS hashing. This means that the former requires $O(\ell k d^2)$ extra storage for projection vectors and per-query computation whereas the latter requires $O(\ell k d)$. We use the hashing with fraction parameter p so that the hashing returns pN images. We perform the multi-probe up to 5000 buckets and stop, where we break ties uniformly at random among buckets that are equally nearby. If this does not return total pN images, we subsample images uniformly at random to make up the difference.

We summarize the time complexity of each algorithm in Table 3.2. Although the per-iteration time complexity of evaluating the OFUL and QOFUL objective function is $O(Nd^2)$ with a naïve implementation, one can make it $O(Nd)$ using rank-one updates of the $(\overline{\mathbf{V}}_t^{-1})$-norm of each image. Note that all the algorithms except for NN have some parameters to tune. Parameter tuning in bandit algorithms is hard in general. Thus, we simply present the best performing parameter set. Details on the parameters can be found in our supplementary material.

Results

We assume that each user has a target type of shoes in mind and simulated the user feedback as a binary value 1 and -1 encoding positive and negative feedback respectively. We manually select 6 shoe types whose cardinality is not too large or small: red boots, Asics, pre-walker, over-the-knee, boat, and toddler shoes. From each shoe type, we randomly select 6 starting points. For each starting points, we perform $T = 50$ interactive search iterations. For algorithms with internal randomization such as LTS

[2] `https://github.com/yahoo/Optimal-LSH`

Algorithm	Time order	Wall-clock time	Red boots	Asics	Pre-walker	Over-the-knee	Boat	Toddler
NN	Nd/T	2	17.1±4.3	11.5±2.7	3.7±1.3	3.5±0.9	17.5±4.5	4.1±1.3
OFUL-NBS1	$d^2 + Nd$	522	25.3±5.9	15.3±5.7	4.9±2.8	1.6±1.1	34.9±3.7	6.0±1.7
OFUL-NBS2		522	37.7±1.3	29.4±2.4	14.0±2.3	4.1±1.2	34.5±3.8	8.4±2.2
OFUL		522	**39.5±1.2**	**31.9±3.2**	14.9±2.9	6.8±1.5	38.0±4.3	11.0±2.8
OFUL-Light (p=0.01)	pNd^2	56	25.5±1.8	14.2±2.1	5.6±1.3	3.5±0.9	24.7±3.8	6.8±1.8
OFUL-Light (p=0.02)		98	30.0±2.6	17.3±2.6	7.3±1.6	3.6±1.1	32.0±2.9	7.7±1.7
OFUL-Light (p=0.05)		222	34.4±2.6	23.1±3.5	10.2±2.2	5.3±1.4	34.8±3.6	8.5±1.9
QOFUL-NBS1	$d^2 + Nd$	522	25.3±5.9	15.3±5.7	4.9±2.8	1.6±1.1	34.9±3.7	6.0±1.7
QOFUL-NBS2		522	37.6±1.4	29.2±2.2	14.3±2.3	4.5±1.1	33.6±3.9	8.7±2.0
QOFUL		522	39.1±1.2	31.3±3.7	**15.1±2.8**	**7.7±1.4**	**38.7±3.7**	**11.3±2.8**
QOFUL-Light (p=0.01)	pNd^2	54	25.3±2.7	12.9±2.4	6.1±1.6	4.0±1.1	27.5±4.2	5.7±1.7
QOFUL-Light (p=0.02)		96	31.6±2.4	18.1±2.7	7.8±1.8	3.7±1.0	31.3±4.3	8.1±2.1
QOFUL-Light (p=0.05)		220	36.1±1.9	23.4±3.1	9.8±2.5	5.6±1.0	34.0±4.7	9.9±2.3
QOFUL-Hash (p=0.01)	$(pN + \ell k)d^2$	184	35.7±2.5	18.9±3.7	5.6±1.8	4.4±0.9	30.5±4.0	8.6±2.2
QOFUL-Hash (p=0.02)		232	35.1±3.3	22.9±3.4	9.2±2.3	4.9±1.2	34.4±4.1	9.3±2.2
QOFUL-Hash (p=0.05)		348	37.6±2.6	29.5±2.9	11.9±2.8	5.3±1.3	35.6±3.9	9.1±2.4
LTS-NBS1	$d^2 + Nd$	281	23.6±6.8	15.1±6.3	3.2±2.3	1.7±1.3	31.2±6.5	5.7±2.4
LTS-NBS2		281	36.9±3.1	24.2±6.0	11.1±3.0	5.1±1.7	35.7±4.9	8.9±2.5
LTS		281	**39.5±1.2**	**31.9±3.2**	14.9±2.9	6.8±1.5	38.0±4.3	11.0±2.8
LTS-Light (p=0.01)	$d^2 + pNd$	40	23.7±2.3	13.8±2.2	5.9±1.5	3.1±0.9	28.5±3.0	6.4±1.6
LTS-Light (p=0.02)		52	30.9±2.0	18.7±2.3	7.9±2.0	4.1±0.9	33.1±3.1	9.5±2.1
LTS-Light (p=0.05)		86	36.3±2.3	24.7±2.7	9.9±2.5	5.4±1.3	33.1±4.5	8.1±2.0
LTS-Hash (p=0.01)	$d^2 + (pN + \ell k)d$	54	37.4±2.1	25.9±3.4	8.3±2.5	5.1±1.3	36.1±3.9	9.6±2.2
LTS-Hash (p=0.02)		74	38.4±1.8	28.5±3.2	9.4±2.4	5.1±1.3	36.2±4.4	9.2±2.1
LTS-Hash (p=0.05)		136	38.8±1.8	29.0±3.6	13.2±2.8	6.4±1.5	37.5±3.8	9.9±2.5
OFUL-Lazy-Light (p=0.01)	$pN(d + d^2(\log T)/T)$	36	23.2±2.2	13.9±2.3	5.3±1.3	3.4±0.8	24.6±3.8	5.7±1.5
OFUL-Lazy-Light (p=0.02)		54	29.9±2.2	17.8±2.6	6.7±1.9	3.8±0.9	29.7±3.3	7.1±1.6
OFUL-Lazy-Light (p=0.05)		114	34.3±2.2	21.8±4.0	9.9±2.2	4.8±1.3	30.9±4.7	7.7±1.8
OFUL-Lazy-Hash (p=0.01)		44	32.9±2.5	20.6±2.4	9.6±2.0	3.8±0.9	30.5±4.3	7.1±2.0
OFUL-Lazy-Hash (p=0.02)		62	36.0±1.5	22.1±3.0	9.5±2.5	5.1±1.1	34.2±3.3	9.5±2.1
OFUL-Lazy-Hash (p=0.05)		130	36.3±2.2	27.3±3.1	11.9±2.7	5.1±1.3	32.3±4.8	8.5±2.1

Table 3.2: The number of relevant images retrieved by each algorithm with 95% confidence bounds. We show the per-iteration time in both order notation and wall-clock (milli seconds).

and those with suffix -Light or -Hash, we repeat 5 times with the same starting point but with different random seeds. Algorithms are fairly optimized with rank-one vectors updates on $\overline{\mathbf{V}}_t^{-1}$ and with numpy array computations.

We summarize the results as the number of relevant images retrieved for each shoe type in Table 3.2 where we also show the average wall-clock running time. We make five observations.

- All the bandit-based algorithms outperform NN. Even a naïve version like OFUL-Light with $p = 0.01$ is near or better than the performance of NN. The implication is significant in that the individual needs from the users are not well modeled by the nearest neighbor search.

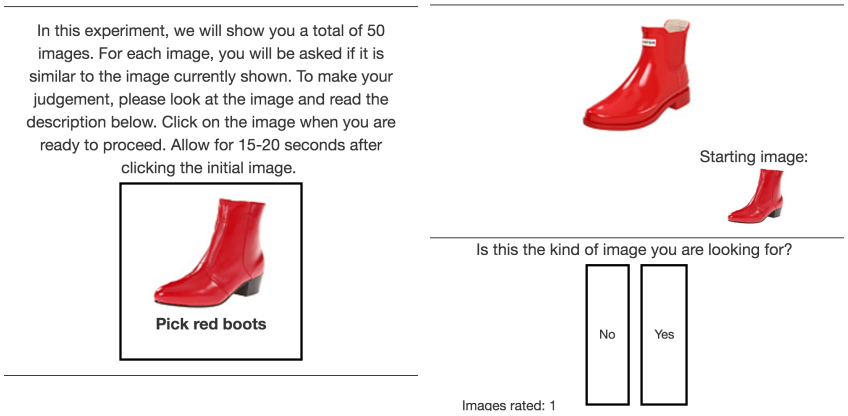
- For every algorithm, hashing versions outperform light versions. Overall, the hashing version of an algorithm with $p = 0.01$ is better than the light version with $p = 0.02$ and slightly worse than with $p = 0.05$. For example, for LTS, LTS-Hash with $p = 0.01$ is close to LTS-Light with $p = 0.05$ in all shoe types

but Asics. This aligns well with the MIPS hashing theory that guarantees the quality of the output of the hashing w.r.t. the inner product.

- Our proposed biased search technique consistently outperforms the two naïve biased search methods NBS1 and NBS2 in all shoe types. Specifically, the far inferior performance of NBS1 shows that biasing just the first image is not sufficient.

- We were not able to find the differences between OFUL, QOFUL, and TS in their search quality. This is different from what the theoretical regret bounds state where OFUL has the smallest regret order and TS has the largest regret order. We are not sure if the analysis of QOFUL and TS is loose or if the short time horizon T blocks us from observing the order difference in regret. A further investigation is left as future work.

- We observe that when fixing the time budget, LTS-Hash performs the best. Although QOFUL-Hash uses the hashing, too, its hashing dimension d^2+d is much larger than d of LTS. We found that this requires a large storage $O(\ell k d^2)$ (2GB in our setting) for storing the projection vectors, which becomes a bottleneck in speed.



(a) Initial Image (b) New Query

Figure 3.2: These are screenshots of the NEXT system for image search. Part (a) is the landing page where the users are assigned a starting point and given instructions on how to go about the task. (b) shows the system where the user is proposed a query image and asked for feedback.

3.7 Implementation with NEXT

NEXT Jamieson et al. (2015) is an open-source software platform designed for the testing and deployment of active learning algorithms. The NEXT system allows researchers to implement multiple active learning algorithms, design an interface, and collect data from people on the internet (*e.g.,* Mechanical Turk) by asking participants simple questions. We implemented our proposed algorithm (QOFUL), Linear Thompson Sampling, OFUL Light, OFUL Lazy Hashing, and Nearest Neighbor. We then compared the five algorithms by collecting user feedback.

There are many challenges to implementing these systems. Two of them we encountered are, the volume of traffic to the system, and users not willing to wait for the system to update in order to get their next query. For example, let us assume that for one of the algorithms, the model update takes 100 ms. So if 20 people are using the system at the same time, it could take up to 2 seconds for the next query to show up. This is too slow.

In order to address this challenge, we used the asynchronous update system avail-

able in NEXT. There are two main functions that are exposed to the API that the user interacts with in NEXT: getQuery and processAnswer. Please refer to Figure 1 in Jamieson et al. (2015) for a visual of the NEXT system.

getQuery is called when the browser needs to display a new query to the user. We need to make sure that this is optimized for time, as users may not want to wait for long between queries. processAnswer takes the answer given by the user and updates the models for a particular algorithm. It is in processAnswer we do asynchronous updates. Every algorithm has to compute an objective value over all the arms. For example, for Thompson sampling, this is:

$$\text{argmax}_{x \in \mathcal{X}_t} \left\langle \overline{\theta}_t, x \right\rangle$$

In order to make getQuery fast, we compute $f(x_i) = \left\langle \overline{\theta}_t, x_i \right\rangle, \forall x_i \in \mathcal{X}$. Then all we need to do is take $\text{argmax}_i x_i$ in getQuery. This means that as soon as the user is ready for the next query, we can quickly decide which image should be shown. Once the user gives her feedback on the image, we push the model update into a queue that getQuery does not wait on.

Theoretical performance bounds and even simulation results are based on selecting x_t as some function of $\overline{\theta_t}$. However, when a large number of users are simultaneously accessing the system, the optimal x_t may not have been computed by the time the user is ready to view a new image. To address this challenge, at each round t we compute a set of good candidate arms, rank them according to the objective function associated with the bandit algorithm being used, and sequentially present these images to the user until the next set of candidate arms is computed and available. This practical but suboptimal approach addresses computational lags associated with MAB algorithms. Algorithms with good regret bounds but high computational complexity may experience more such lags, and hence rely more upon this approach and yield lower rewards overall.

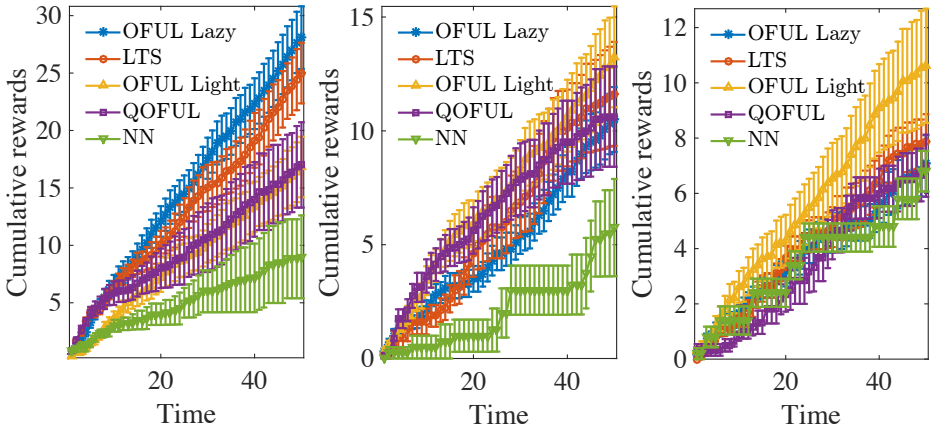
In order to get as many user responses as possible, we used Amazon Mechanical Turk as well as sending out the survey to a university mailing list. Each user was presented with one of three different starting points. They were assigned an algorithm in a round-robin manner. The users were blind to which algorithm they were assigned. The users are then sequentially presented with 50 images and asked whether each is similar to the initial image (e.g. "Is this shoe also a red boot?"). Figure 3.2 shows one instance of the system in practice.

Figure 3.3 shows the results of our experiments on the NEXT system. First we note that all the algorithms are doing better than nearest neighbors (NN). We observe that the there is a deterioration in the performance of QOFUL compared to our simulations. This is explained by Figure 3.4 (in the supplemental material), which shows the amount of time each algorithm took to update models.

The reason for the high running time of QOFUL is that the current version of the NEXT system computes independent hash tables for each run of the algorithm for each user, requiring a large amount of time and memory (2 GB). In ongoing work we are computing a single hash table on the database, which will significantly reduce the computational burden of QOFUL.

Next, we note that OFUL Lazy did well on red boots but not on others and note that OFUL Lazy needed to update the model for fewer times for this starting point but not others. This is because the local geometry in the vicinity of the starting point influences how fast the matrix $\overline{\mathbf{V}}_t$ grows. This affects the number of times that OFUL Lazy has to update the models.

The main point is that all these algorithms are doing better than nearest neighbors. We think that with larger datasets, we would get a bigger improvement of QOFUL.



(a) Red Boots (b) Toddlers (c) ASICS

Figure 3.3: Cumulative rewards for three different starting points. The lines represent the cumulative sum of the rewards. The users were shown the starting image and then asked if the image looked similar to starting one for 50 different. If users answered Yes, then the reward was 1, if they said No, the reward was 0. Note that even though NN is a deterministic algorithm, there is variability in how different users evaluate different images, leading to the error bars. The error bars represent the standard error.

3.8 Supplemental Materials

3.8.1 Proof of Theorem 3.4.1

The following lemma shows that the objective function (3.7) plus $c_1\beta^{3/4}m_{t-1}$ is an upper bound of the OFUL's objective function (3.5).

Lemma 3.8.1.

$$\begin{aligned}&\langle\widehat{\boldsymbol{\theta}}_{t-1},\mathbf{x}\rangle+\sqrt{\beta_{t-1}}||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}}\\&\leq\langle\widehat{\boldsymbol{\theta}}_{t-1},\mathbf{x}\rangle+\frac{\beta_{t-1}^{1/4}}{4c_1m_{t-1}}\cdot||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}+c_1\beta^{3/4}m_{t-1}\ .\end{aligned}$$

Furthermore,

$$\langle\widehat{\boldsymbol{\theta}}_{t-1},\mathbf{x}_t^{OFUL}\rangle+\sqrt{\beta_{t-1}}||\mathbf{x}_t^{OFUL}||_{\overline{\mathbf{V}}_{t-1}^{-1}}$$

$$\leq \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}_t^{QOFUL} \rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}_t^{QOFUL}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} + c_1 \beta_{t-1}^{3/4} m_{t-1} \ . \tag{3.15}$$

Proof. Recall the standard OFUL optimization problem defined in (3.3). For a fixed $\mathbf{x}$, we need to solve

$$\begin{aligned} \tilde{\boldsymbol{\theta}}(\mathbf{x}) = \arg\min_{\boldsymbol{\theta}} \quad & -\langle \boldsymbol{\theta}, \mathbf{x} \rangle \\ \text{s.t.} \quad & ||\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{t-1}||^2_{\overline{\mathbf{V}}_{t-1}} - \beta_{t-1} \leq 0 \end{aligned}$$

The Lagrangian is $\mathcal{L}(\boldsymbol{\theta}, \mu) := -\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \mu(||\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{t-1}||^2_{\overline{\mathbf{V}}_{t-1}} - \beta_{t-1})$, and thus we need to solve

$$\max_{\mu \geq 0} \min_{\boldsymbol{\theta}} \quad \mathcal{L}(\boldsymbol{\theta}, \mu) \ .$$

By the first-order optimality condition,

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{\theta}, \mu)}{\partial \boldsymbol{\theta}} &= \mathbf{x} - \mu(2\overline{\mathbf{V}}_{t-1}\boldsymbol{\theta} - 2\overline{\mathbf{V}}_{t-1}\widehat{\boldsymbol{\theta}}_{t-1}) = 0 \\ \boldsymbol{\theta} &= \widehat{\boldsymbol{\theta}}_{t-1} + (2\mu)^{-1}\overline{\mathbf{V}}_{t-1}^{-1}\mathbf{x} \ , \end{aligned} \tag{3.16}$$

which results in $(\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mu)) = -\langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x} \rangle - (4\mu)^{-1}||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} - \mu\beta_{t-1}$. It remains to solve

$$\max_{\mu \geq 0} \quad -\langle \hat{\boldsymbol{\theta}}, \mathbf{x} \rangle - (4\mu)^{-1}||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} - \mu\beta_{t-1},$$

whose first-order optimality says that the solution is $\mu_* := (2\sqrt{\beta_{t-1}})^{-1}||\mathbf{x}||_{\overline{\mathbf{V}}^{-1}}$. Plugging $\mu \leftarrow \mu_*$ in (3.16) leads to the solution $\tilde{\boldsymbol{\theta}}(\mathbf{x})$ defined in (3.4). Note that any choice of $\mu \geq 0$ leads to a lower bound on the Lagrangian $\mathcal{L}(\tilde{\boldsymbol{\theta}}(\mathbf{x}), \mu)$. Define

$$\tilde{\mu} := c_1 \beta_{t-1}^{-1/4} m_{t-1} \ .$$

Then,

$$\begin{aligned}\mathcal{L}(\tilde{\boldsymbol{\theta}}(\mathbf{x}), \tilde{\mu}) &= -\langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x} \rangle - \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} - c_1 \beta_{t-1}^{3/4} m_{t-1} \\ &\leq \mathcal{L}(\tilde{\boldsymbol{\theta}}(\mathbf{x}), \mu_*) = -\langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x} \rangle - \sqrt{\beta_{t-1}} ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}} \ .\end{aligned}$$

This concludes the first part of the lemma. The second part of the lemma trivially follows from the first part by the definition (3.7). □

Define $\boldsymbol{\eta}_t = [\eta_1; \cdots; \eta_t]$. Note that, assuming $d \geq 2$,

$$E_1(\delta) \implies ||\mathbf{X}_t^\top \boldsymbol{\eta}_t||_{\overline{\mathbf{V}}_t^{-1}} \leq R\sqrt{d \log\left(\frac{1 + t/(d\lambda)}{\delta}\right)}\ , \tag{3.17}$$

$$\sum_{t=1}^{T} \log\left(1 + ||\mathbf{x}_t||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}\right) \leq 2d \log\left(1 + t/(d\lambda)\right)\ , \tag{3.18}$$

which are due to Theorem 1 and Lemma 11 of Abbasi-Yadkori et al. (2011) respectively.

The following lemma becomes useful.

Lemma 3.8.2. *For any $q, x \geq 0$,*

$$\min\{q, x\} \leq \max\{2, q\} \log(1 + x)$$

Proof. It is not hard to see that

$$x \in [0, q] \implies x \leq \frac{q}{\log(1+q)} \log(1 + x) \tag{3.19}$$

We consider the following two cases.

Case 1. $q \leq 2$

If $x \leq 2$, by (3.19), $\min\{2, x\} = x \leq \frac{2}{\log(3)} \log(1 + x) \leq 2 \log(1 + x)$. If $x > 2$, $\min\{2, x\} = 2 \leq 2 \log(1 + 2) \leq 2 \log(1 + x)$. Thus, for any $x \in [0, q]$, $\min\{q, x\} \leq$

$\min\{2, x\} \leq 2\log(1+x)$.

Case 2. $q > 2$

If $x \leq q$, by (3.19), $\min\{q, x\} = x \leq \frac{q}{\log(1+q)}\log(1+x) < q\log(1+x)$. If $x > q$, $\min\{q, x\} = q \leq q\log(1+2) \leq q\log(1+x)$.

Combine both cases to complete the proof. □

Using the lemmas above, we proof Theorem 3.4.1 below.

Proof. Assume $E_1(\delta)$. Suppose we pull $\mathbf{x}_t^{\text{QOFUL}}$ at every iteration. The instantaneous regret is

$$
\begin{aligned}
r_t &:= \langle \boldsymbol{\theta}_*, \mathbf{x}_{t,*} \rangle - \langle \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL}} \rangle \\
&\leq \langle \tilde{\boldsymbol{\theta}}_{t1}, \mathbf{x}_t^{\text{OFUL}} \rangle - \langle \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL}} \rangle \\
&= \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}_t^{\text{OFUL}} \rangle + \sqrt{\beta_{t-1}} ||\mathbf{x}_t^{\text{OFUL}}||_{\overline{\mathbf{V}}_{t-1}^{-1}} - \langle \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL}} \rangle \\
&\overset{\text{Lem. 3.8.1}}{\leq} \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}_t^{\text{QOFUL}} \rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} + c_1 \beta_{t-1}^{3/4} m_{t-1} - \langle \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL}} \rangle \\
&\leq \langle \widehat{\boldsymbol{\theta}}_{t-1} - \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL}} \rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} + c_1 \beta_{t-1}^{3/4} m_{t-1} \\
&\leq \underbrace{||\widehat{\boldsymbol{\theta}}_{t-1} - \boldsymbol{\theta}_*||_{\overline{\mathbf{V}}_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||_{\overline{\mathbf{V}}_{t-1}^{-1}}}_{=:A_1(t)} + \underbrace{\frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}}_{:=A_2(t)} + \underbrace{c_1 \beta_{t-1}^{3/4} m_{t-1}}_{=:A_3(t)} .
\end{aligned}
$$

Note that $\langle \boldsymbol{\theta}_*, \mathbf{x} \rangle \in [-S, S]$ implies that $r_t \leq 2S$. Then,

$$
\begin{aligned}
r_t &\leq \min\{2S, A_1(t) + A_2(t) + A_3(t)\} \\
&\leq \min\{2S, A_1(t)\} + \min\{2S, A_2(t)\} + \min\{2S, A_3(t)\}
\end{aligned}
$$

where the last inequality can be shown by a case-by-case analysis on each $\min$ operator.

Now, consider the cumulative regret $\sum_{t=1}^{T} r_t$. The term $\sum_{t=1}^{T} \min\{2S, A_1(t)\}$ is bounded by $O(d\sqrt{T}\log(T))$ in the proof of Theorem 3 of Abbasi-Yadkori et al. (2011).

Then, since β_t is non-decreasing,

$$\begin{aligned}
\sum_{t=1}^{T} \min\left\{2S, \frac{\beta_{T-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}\right\} &\leq \sum_{t=1}^{T} \min\left\{2S, c_2 ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}\right\} \\
&\leq c_2 \sum_{t=1}^{T} \min\left\{\frac{2S}{c_2}, ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}\right\} \\
&\overset{\text{(Lem. 3.8.2)}}{\leq} c_2 \max\left\{2, \frac{2S}{c_2}\right\} \sum_{t=1}^{T} \log(1 + ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}) \\
&\overset{(3.9),(3.18)}{=} O((d\log T)^{1/4}) \cdot O(\sqrt{T}) \cdot O(d \log T) \\
&= O\left(d^{5/4}\sqrt{T \log^{2.5} T}\right)
\end{aligned}$$

and

$$\begin{aligned}
\sum_{t=1}^{T} \min\{2S, c_1 \beta_{t-1}^{3/4} m_{t-1}\} &\overset{(3.6)}{\leq} \sum_{t=1}^{T} \min\{2S, c_1 \beta_{T-1}^{3/4} ||\mathbf{x}_t^{\text{QOFUL}}||_{\overline{\mathbf{V}}_{t-1}^{-1}}\} \\
&\leq c_3 \sum_{t=1}^{T} \min\left\{\frac{2S}{c_3}, ||\mathbf{x}_t^{\text{QOFUL}}||_{\overline{\mathbf{V}}_{t-1}^{-1}}\right\} \\
&\overset{(a)}{\leq} c_3 \sqrt{T \sum_{t=1}^{T} \min\left\{\left(\frac{2S}{c_3}\right)^2, ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}}\right\}} \\
&\overset{\text{(Lem. 3.8.2)}}{\leq} c_3 \sqrt{T \max\left\{2, \left(\frac{2S}{c_3}\right)^2\right\} \sum_{t=1}^{T} \log(1 + ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}})} \\
&\overset{(3.9),(3.18)}{=} O((d\log T)^{3/4}) \cdot O(\sqrt{Td \log T}) \\
&= O\left(d^{5/4}\sqrt{T \log^{2.5} T}\right) ,
\end{aligned}$$

where (a) is due to the Cauchy-Schwartz inequality, $c_2 = \frac{\beta_{T-1}^{1/4}}{4c_1 r}\sqrt{T+\lambda}$, and $c_3 = c_1 \beta_{T-1}^{3/4}$. □

3.8.2 Proof of Lemma 3.5.1

Proof. To show the lowerbound, recall that the objective function (3.7) is derived as an upperbound of the original OFUL objective function (3.3). Since the original OFUL

objective function has $\boldsymbol{\theta}_*$ as a feasible point by $E_1(\delta)$, $\max_{x\in\mathcal{X}} \langle \boldsymbol{\theta}_*, \mathbf{x}\rangle$ becomes a trivial lowerbound of the maximum of the OFUL objective and also of the maximum of H-OFUL objective (3.7). This proves the lowerbound of the Lemma.

For the remaining part of the proof, we use notation $\mathbf{X}$, $\boldsymbol{\eta}$, and $\overline{\mathbf{V}}$ in place of $\mathbf{X}$ and $\boldsymbol{\eta}$, $\overline{\mathbf{V}}$ respectively. We claim that

$$||\overline{\mathbf{V}}^{-1}\mathbf{X}^\top\boldsymbol{\eta}||_2 = ||\mathbf{X}^\top\boldsymbol{\eta}||_{\overline{\mathbf{V}}^{-2}} \leq \frac{1}{\sqrt{\lambda}}||\mathbf{X}^\top\boldsymbol{\eta}||_{\overline{\mathbf{V}}^{-1}}. \tag{3.20}$$

To see this, note that the eigen values of $\overline{\mathbf{V}}^{-1}$ are of form $1/(\lambda+q_i)$ where $q_i \geq 0$, and the eigen values of $\overline{\mathbf{V}}^{-2}$ are of form $1/(\lambda+q_i)^2$ while the eigen vectors remain the same. The fact $1/(\lambda+q_i)^2 = 1/(\lambda^2+2\lambda q_i+q_i^2) \leq 1/(\lambda^2+\lambda q_i) = (1/\lambda)\cdot(1/(\lambda+q_i))$ implies that $||\mathbf{X}^\top\boldsymbol{\eta}||_{\overline{\mathbf{V}}^{-2}} \leq \frac{1}{\sqrt{\lambda}}||\mathbf{X}^\top\boldsymbol{\eta}||_{\overline{\mathbf{V}}^{-1}}$.

Furthermore, $||\overline{\mathbf{V}}^{-1}\boldsymbol{\theta}_*||_2 = \sqrt{\boldsymbol{\theta}_*^\top\overline{\mathbf{V}}^{-2}\boldsymbol{\theta}_*} \leq (1/\lambda)||\boldsymbol{\theta}_*||_2$ since the eigen values of $\overline{\mathbf{V}}^{-2}$ are no larger than $1/\lambda^2$.

Then,

$$\begin{aligned}
||\widehat{\boldsymbol{\theta}}_{t-1}||_2 &\leq ||\overline{\mathbf{V}}^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}_*||_2 + ||\overline{\mathbf{V}}^{-1}\mathbf{X}^\top\boldsymbol{\eta}||_2 \\
&= ||\overline{\mathbf{V}}^{-1}(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I})\boldsymbol{\theta}_* - \overline{\mathbf{V}}^{-1}(\lambda\mathbf{I})\boldsymbol{\theta}_*||_2 + ||\overline{\mathbf{V}}^{-1}\mathbf{X}^\top\boldsymbol{\eta}||_2 \\
&\leq ||\boldsymbol{\theta}_*||_2 + \lambda||\overline{\mathbf{V}}^{-1}\boldsymbol{\theta}_*||_2 + \frac{1}{\sqrt{\lambda}}||\mathbf{X}^\top\boldsymbol{\eta}||_{\overline{\mathbf{V}}^{-1}} \\
&\leq 2||\boldsymbol{\theta}_*||_2 + \frac{1}{\sqrt{\lambda}}||\mathbf{X}^\top\boldsymbol{\eta}||_{\overline{\mathbf{V}}^{-1}}
\end{aligned}$$

Finally, upperbound the maximum of the objective function (3.7) plus $c_1\beta^{3/4}m_{t-1}$:

$$\begin{aligned}
\max_{t\in[T]}\max_{\mathbf{x}\in\mathcal{X}} \langle\widehat{\boldsymbol{\theta}}_{t-1},\mathbf{x}\rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}}||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} &\leq \max_{t\in[T]}\max_{\mathbf{x}\in\mathcal{X}} ||\widehat{\boldsymbol{\theta}}_{t-1}||_2 + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}}||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} \\
&\leq \max_{t\in[T]} 2S + (1/\sqrt{\lambda})||\mathbf{X}^\top\boldsymbol{\eta}||_{\overline{\mathbf{V}}_{t-1}^{-1}} + \beta_{t-1}^{1/4}\cdot\frac{\sqrt{t+\lambda}}{4c_1 r}\cdot\frac{1}{\lambda} \\
&\overset{(3.17),(3.9)}{<} \frac{2}{\sqrt{\lambda}}\sqrt{\overline{\beta}_{T-1}} + \overline{\beta}_{T-1}^{1/4}\cdot\frac{\sqrt{T+\lambda}}{4c_1 r}\cdot\frac{1}{\lambda}\,.
\end{aligned}$$

□

3.8.3 Proof of Theorem 3.5.2

Proof. Throughout the proof, assume the event that a retrieved vector $\mathbf{x}^{(j)}$ from j-th MIPS satisfies $(c_{\mathrm{H}}^{1/2}, c_{\mathrm{H}}^{j/2} M_{\max})$-MIPS for $j \in [J]$, which happens with high probability.

Define the maximum $M^* := \max_{\mathbf{x} \in \mathcal{X}} \langle \mathbf{q}, \mathbf{x} \rangle$. The result of the binary search is that, for query $\mathbf{q}$, j^*-th MIPS succeeds but $(j^* - 1)$-th MIPS fails. By the definition of being $(c_{\mathrm{H}}^{1/2}, c_{\mathrm{H}}^{(j^*-1)/2} M_{\max})$-MIPS, the fact that $(j^* - 1)$-th hashing fails implies $M^* < c_{\mathrm{H}}^{(j^*-1)/2} M_{\max}$. Then,

$$\langle \mathbf{q}, \mathbf{x}^{(j^*)} \rangle \geq c_{\mathrm{H}}^{1/2} \cdot c_{\mathrm{H}}^{j^*} M_{\max} = c_{\mathrm{H}} \cdot c_{\mathrm{H}}^{(j^*-1)/2} M_{\max} > c_{\mathrm{H}} M^* \ .$$

□

3.8.4 Proof of Theorem 3.5.3

Proof. Assume $E_1(\delta)$. Denote by $\mathbf{x}_t^{\text{QOFUL,H}}$ the solution returned by the MIPS algorithm. Then, being c_{H}-MIPS guarantees that, $\forall t \in [T]$,

$$\langle \widehat{\boldsymbol{\theta}}_{t-1}, x_t^{\text{QOFUL,H}} \rangle + \frac{\beta_{t-1}^{1/4}}{4 c_1 m_{t-1}} ||x_t^{\text{QOFUL,H}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} \geq c_{\mathrm{H}} \left(\langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}_t^{\text{QOFUL}} \rangle + \frac{\beta_{t-1}^{1/4}}{4 c_1 m_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} \right) \ . \tag{3.21}$$

To avoid clutter, we use $\mathbf{X}$, $\mathbf{y}$, and $\boldsymbol{\eta}$ in place of $\mathbf{X}_{t-1}$, $\mathbf{y}_{t-1}$, and $\boldsymbol{\eta}_{t-1}$, respectively, when it is clear from the context. Note that

$$\begin{aligned}
\langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}_t^{\text{QOFUL,H}} \rangle &= (x_t^{\text{QOFUL,H}})^\top \overline{\mathbf{V}}_{t-1}^{-1} \mathbf{X}^\top \mathbf{y} = (x_t^{\text{QOFUL,H}})^\top \overline{\mathbf{V}}_{t-1}^{-1} \mathbf{X}^\top \mathbf{X} \boldsymbol{\theta}_* + (x_t^{\text{QOFUL,H}})^\top \overline{\mathbf{V}}_{t-1}^{-1} \mathbf{X}^\top \boldsymbol{\eta} \\
&= (x_t^{\text{QOFUL,H}})^\top \boldsymbol{\theta}_* + (x_t^{\text{QOFUL,H}})^\top \overline{\mathbf{V}}_{t-1}^{-1} (-\lambda \mathbf{I}) \boldsymbol{\theta}_* + (x_t^{\text{QOFUL,H}})^\top \overline{\mathbf{V}}_{t-1}^{-1} \mathbf{X}^T \boldsymbol{\eta}
\end{aligned}$$

$$\overset{(a)}{\leq} S + \lambda ||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}} ||\boldsymbol{\theta}_*||_{\overline{\mathbf{V}}_{t-1}^{-1}} + ||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}} ||\mathbf{X}^\top \boldsymbol{\eta}||_{\overline{\mathbf{V}}_{t-1}^{-1}}$$
$$\overset{(b)}{\leq} S + ||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}} (\sqrt{\lambda} S + ||\mathbf{X}^\top \boldsymbol{\eta}||_{\overline{\mathbf{V}}_{t-1}^{-1}})$$
$$\overset{(3.17),(3.9)}{\leq} S + ||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}} \sqrt{\overline{\beta}_{t-1}}, \tag{3.22}$$

where (a) is due to the Cauchy-Schwartz inequality and (b) is due to $||\boldsymbol{\theta}_*||_{\overline{\mathbf{V}}_{t-1}^{-1}} \leq \sqrt{1/\lambda}||\boldsymbol{\theta}_*||_2 \leq \sqrt{1/\lambda} S$.

The instantaneous regret at time t is

$$\begin{aligned}
r_t &:= \langle \boldsymbol{\theta}_*, \mathbf{x}_{t,*} \rangle - \langle \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL,H}} \rangle \\
&\leq \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}_t^{\text{QOFUL}} \rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} + c_1 \beta_{t-1}^{3/4} m_{t-1} - \langle \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL}} \rangle \\
&\overset{(3.21)}{\leq} \frac{1}{c_{\text{H}}} \left(\langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}_t^{\text{QOFUL}} \rangle + \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||\mathbf{x}_t^{\text{QOFUL}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} \right) + c_1 \beta_{t-1}^{3/4} m_{t-1} - \langle \boldsymbol{\theta}_*, \mathbf{x}_t^{\text{QOFUL}} \rangle \\
&= \left(\frac{1}{c_{\text{H}}} - 1 \right) \langle \widehat{\boldsymbol{\theta}}_{t-1}, x_t^{\text{QOFUL,H}} \rangle + \langle \widehat{\boldsymbol{\theta}}_{t-1} - \boldsymbol{\theta}_*, x_t^{\text{QOFUL,H}} \rangle + \frac{1}{c_{\text{H}}} \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||x_t^{\text{QOFUL,H}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} + c_1 \beta_{t-1}^{3/4} m_{t-1} \\
&\overset{(3.22)}{\leq} \underbrace{\frac{\log T}{\sqrt{T}} S}_{A_1(t)} + \underbrace{\frac{\log T}{\sqrt{T}} ||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}} \sqrt{\overline{\beta}_{t-1}}}_{A_2(t)} \\
&\quad + \underbrace{\langle \widehat{\boldsymbol{\theta}}_{t-1} - \boldsymbol{\theta}_*, x_t^{\text{QOFUL,H}} \rangle + \frac{1}{c_{\text{H}}} \frac{\beta_{t-1}^{1/4}}{4c_1 m_{t-1}} ||x_t^{\text{QOFUL,H}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} + c_1 \beta_{t-1}^{3/4} m_{t-1}}_{A_3(t)} .
\end{aligned}$$

Note that $r_t \leq 2S$. Since $1/c_{\text{H}} \leq 1$, it is not hard to see that $\sum_{t=1}^T \min\{2S, A_3(t)\}$ is $O(d^{5/4}\sqrt{T}\log^{5/4} T)$ using the same technique as the proof of Theorem 3.4.1. Let C be a generic absolute constant. It remains to bound $\sum_{t=1}^T \min\{2S, A_1(t)\}$ and $\sum_{t=1}^T \min\{2S, A_2(t)\}$:

$$\begin{aligned}
\sum_{t=1}^T \min\{2S, A_1(t)\} &\leq \sum_{t=1}^T \min\{2S, \frac{\log T}{\sqrt{T}} 2S\} \\
&\overset{(\log T < \sqrt{T})}{\leq} 2S \sum_{t=1}^T \frac{\log T}{\sqrt{T}}
\end{aligned}$$

$$
\begin{aligned}
&= O(\sqrt{T}\log T) \\
\sum_{t=1}^{T}\min\{2S, A_2(t)\} &= \sum_{t=1}^{T}\min\left\{2S, \frac{\log T}{\sqrt{T}}||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}}\sqrt{\overline{\beta}_{t-1}}\right\} \\
&\leq \sum_{t=1}^{T}\min\left\{2S, ||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}}\sqrt{\overline{\beta}_{T-1}}\right\} \\
&= \sqrt{\overline{\beta}_{T-1}}\sum_{t=1}^{T}\min\left\{\frac{2S}{\sqrt{\overline{\beta}_{T-1}}}, ||x_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}}\right\}
\end{aligned}
$$

Using the same technique used in the proof of Theorem 3.4.1, one can show that

$$
\begin{aligned}
\sqrt{\overline{\beta}_{T-1}}\sum_{t=1}^{T}\min\left\{\frac{2S}{\sqrt{\overline{\beta}_{T-1}}}, ||\mathbf{x}_t^{\text{QOFUL,H}}||_{\overline{\mathbf{V}}_{t-1}^{-1}}\right\} &\leq \sqrt{\overline{\beta}_{T-1}}\sqrt{T\max\left\{2, \frac{4S^2}{\overline{\beta}_{T-1}}\right\}\sum_{t=1}^{T}\log(1+||\mathbf{x}_t^{\text{QOFUL,H}}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}})} \\
&\overset{(3.17),(3.18)}{=} O(d\sqrt{T}\log(T))\ .
\end{aligned}
$$

Altogether, $\sum_{t=1}^{T}\min\{2S, A_1(t)\} + \min\{2S, A_2(t)\} = O(d\sqrt{T}\log(T))$, which is dominated by $\sum_{t=1}^{T}\min\{2S, A_3(t)\}$. This concludes the proof.

□

3.8.5 Proof of Lemma 3.5.4

Proof. The shifted-ridge regression expression we use is:

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_0 + (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$$

then, we can see what effect this has on OFUL. In Appendix B of Abbasi-Yadkori et al. (2011), they derive a bound for $|\mathbf{x}^\top\hat{\boldsymbol{\theta}} - \mathbf{x}^\top\boldsymbol{\theta}_*|$. Let us plug in our modified expression for shifted regression and derive the same bounds as they do.

$$\begin{aligned}\left|\mathbf{x}^\top\hat{\boldsymbol{\theta}} - \mathbf{x}^\top\boldsymbol{\theta}_*\right| &= \left|\mathbf{x}^\top(\boldsymbol{\theta}_0 + (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}) - \mathbf{x}^\top\boldsymbol{\theta}_*\right| \\ &= \left|\mathbf{x}^\top\boldsymbol{\theta}_0 + \mathbf{x}^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}) + \mathbf{x}^\top\boldsymbol{\theta}_0 - \mathbf{x}^\top\boldsymbol{\theta}_0 - \mathbf{x}^\top\boldsymbol{\theta}_*\right| \\ &= \left|\mathbf{x}^\top\boldsymbol{\theta}_{OFUL} - \mathbf{x}^\top(\boldsymbol{\theta}_* - \boldsymbol{\theta}_0)\right| \text{ , where } \boldsymbol{\theta}_{OFUL} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y})\end{aligned}$$

This is exactly in the same form as the expression in the Appendix B of Abbasi-Yadkori et al. (2011), proof of Theorem 2, except that we've replaced $\boldsymbol{\theta}_*$ by $\boldsymbol{\theta}_* - \boldsymbol{\theta}_0$. Proceeding exactly in the same way as the paper, we get the bound:

$$\begin{aligned}\left\|\hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_*\right\|_{\overline{\mathbf{V}}_t} &\leq R\sqrt{2\log\left(\frac{\det(\overline{\mathbf{V}}_t)^{1/2}\det(\lambda\mathbf{I})^{-1/2}}{\delta}\right)} + \sqrt{\lambda}\left\|\boldsymbol{\theta}_* - \boldsymbol{\theta}_0\right\|_2 \\ &\leq R\sqrt{2\log\left(\frac{\det(\overline{\mathbf{V}}_t)^{1/2}\det(\lambda\mathbf{I})^{-1/2}}{\delta}\right)} + \sqrt{\lambda}S_0 \\ \Rightarrow \boldsymbol{\theta}_* &\in \widetilde{C}_t \text{with probability at leaset } 1-\delta\end{aligned}$$

□

3.8.6 Plot of time taken to update models

Figure 3.4 shows the timing results of the different algorithms. We used the timing function within NEXT to compute how much time different algorithms took to update their models after receiving feedback. This is an important consideration in practical bandit algorithms.

We need to serve users with queries as soon as they are ready to answer questions. Therefore the compromise is to use a stale model for slower algorithms. Therefore algorithms that may have higher theoretical regret might perform better in practice

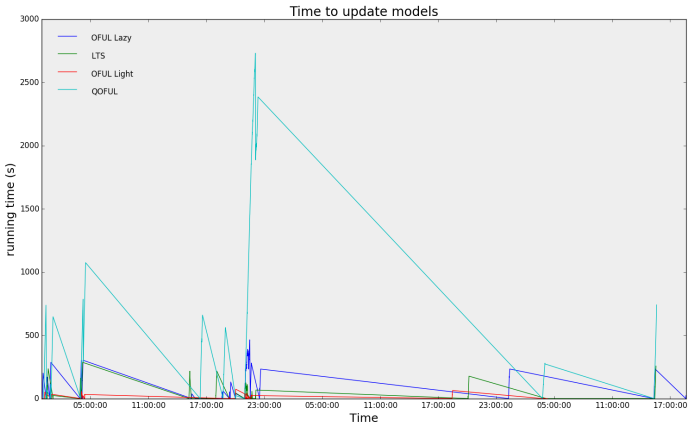


Figure 3.4: Time taken to run model updates for different algorithms. processAnswer in the NEXT system updates the model asynchronously. This plot shows how much time those asynchronous updates took. Nearest neighbors is not shown since it doesn not have a model to be updated. As it can be seen, QOFUL took the most time and OFUL Light took the least amount of time.

since they may update their models quicker.

Chapter 4

Generalized Linear Bandits [1]

[1] Paper to be presented at NIPS 2017

4.1 Introduction

This chapter considers the problem of making generalized linear bandits (GLBs) scalable. In the stochastic GLB problem, a learner makes successive decisions to maximize her cumulative rewards. Specifically, at time t the learner observes a set of arms $\mathcal{X}_t \subseteq \mathbb{R}^d$. The learner then chooses an arm $\mathbf{x}_t \in \mathcal{X}_t$ and receives a stochastic reward y_t that is a noisy function of $\mathbf{x}_t$: $y_t = \mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*) + \eta_t$, where $\boldsymbol{\theta}^* \in \mathbb{R}^d$ is unknown, $\mu : \mathbb{R} \to \mathbb{R}$ is a known nonlinear mapping, and $\eta_t \in \mathbb{R}$ is some zero-mean noise. This reward structure encompasses generalized linear models McCullagh and Nelder (1989); e.g., Bernoulli, Poisson, etc.

The key aspect of the bandit problem is that the learner does not know how much reward she would have received, had she chosen another arm. The estimation on $\boldsymbol{\theta}^*$ is thus biased by the history of the selected arms, and one needs to mix in exploratory arm selections to avoid ruling out the optimal arm. This is well-known as the exploration-exploitation dilemma. The performance of a learner is evaluated by its *regret* that measures how much cumulative reward she would have gained additionally if she had known the true $\boldsymbol{\theta}^*$. We provide backgrounds and formal definitions in Section 4.2.

A linear case of the problem above ($\mu(z) = z$) is called the (stochastic) linear bandit problem. Since the first formulation of the linear bandits Auer and Long (2002), there has been a flurry of studies on the problem Dani et al. (2008); Rusmevichientong and Tsitsiklis (2010); Abbasi-Yadkori et al. (2012); Chu et al. (2011); Agrawal and Goyal (2013). In an effort to generalize the restrictive linear rewards, Filippi et al. (2010) propose the GLB problem and provide a low-regret algorithm, whose Thompson sampling version appears later in Abeille and Lazaric (2017). Li et al. (2012) evaluates GLBs via extensive experiments where GLBs exhibit lower regrets than linear bandits for 0/1 rewards. Li et al. (2017) achieves a smaller regret bound when the arm set $\mathcal{X}_t$ is finite, though with an impractical algorithm.

However, we claim that all existing GLB algorithms Filippi et al. (2010); Li et al. (2017) suffer from two scalability issues that limit their practical use: (i) under a large time horizon and (ii) under a large number N of arms.

First, existing GLBs require storing all the arms and rewards appeared so far, $\{(\mathbf{x}_s, y_s)\}_{s=1}^{t}$, so the space complexity grows linearly with t. Furthermore, they have to solve a batch optimization problem for the maximum likelihood estimation (MLE) at each time step t whose per-time-step time complexity grows at least linearly with t. While Zhang et al. (2016) provide a solution whose space and time complexity do not grow over time, they consider a specific 0/1 reward with the logistic link function, and a generic solution for GLBs is not provided.

Second, existing GLBs have linear time complexities in N. This is impractical when N is very large, which is not uncommon in applications of GLBs such as online advertisements, recommendation systems, and interactive retrieval of images or documents Li et al. (2010, 2012); Yue et al. (2012b); Hofmann et al. (2011); Konyushkova and Glowacka (2013) where arms are items in a very large database. Furthermore, the interactive nature of these systems requires prompt responses as users do not want to wait. This implies that the typical linear time in N is not tenable. Towards a *sublinear* time in N, locality sensitive hashings Har-Peled et al. (2012) or its extensions Shrivastava and Li (2014, 2015); Neyshabur and Srebro (2015) are good candidates as they have been successful in fast similarity search and other machine learning problems like active learning Jain et al. (2010), where the search time scales with N^ρ for some $\rho < 1$ (ρ is usually optimized and often ranges from 0.4 to 0.8 depending on the target search accuracy). Leveraging hashing in GLBs, however, relies critically on the objective function used for arm selections. The function must take a form that is readily optimized using *existing* hashing algorithms.[2] For example, algorithms whose objective function (a function of each arm $\mathbf{x} \in \mathcal{X}_t$) can be written as a distance or inner product

[2] Without this designation, no *currently known* bandit algorithm achieves a sublinear time complexity in N.

between $\mathbf{x}$ and a query $\mathbf{q}$ are hash-amenable as there *exist* hashing methods for such functions.

To be scalable to a large time horizon, we propose a new algorithmic framework called Generalized Linear Online-to-confidence-set Conversion (GLOC) that takes in an online learning (OL) algorithm with a low 'OL' regret bound and turns it into a GLB algorithm with a low 'GLB' regret bound. The key tool is a novel generalization of the online-to-confidence-set conversion technique used in Abbasi-Yadkori et al. (2012) (also similar to Dekel et al. (2012); Crammer and Gentile (2013); Gentile and Orabona (2014); Zhang et al. (2016)). This allows us to construct a confidence set for $\boldsymbol{\theta}^*$, which is then used to choose an arm $\mathbf{x}_t$ according to the well-known optimism in the face of uncertainty principle. By relying on an online learner, GLOC inherently performs online computations and is thus free from the scalability issues in large time steps. While any online learner equipped with a low OL regret bound can be used, we choose the online Newton step (ONS) algorithm and prove a tight OL regret bound, which results in a practical GLB algorithm with almost the same regret bound as existing inefficient GLB algorithms. We present our proposed algorithms and their regret bounds in Section 4.3.

For large number N of arms, our proposed algorithm GLOC is not hash-amenable, to our knowledge, due to its non-linear criterion for arm selection. As the first attempt, we derive a Thompson sampling Agrawal and Goyal (2013); Abeille and Lazaric (2017) extension of GLOC (GLOC-TS), which is hash-amenable due to its linear criterion. However, its regret bound scales with $d^{3/2}$ for d-dimensional arm sets, which is far from d of GLOC. Towards closing this gap, we propose a new algorithm Quadratic GLOC (QGLOC) with a regret bound that scales with $d^{5/4}$. We summarize the comparison of our proposed

Algorithm	Regret	Hash-amenable
GLOC	$\tilde{O}(d\sqrt{T})$	✗
GLOC-TS	$\tilde{O}(d^{3/2}\sqrt{T})$	✓
QGLOC	$\tilde{O}(d^{5/4}\sqrt{T})$	✓

Table 4.1: Comparison of GLBs algorithms for d-dimensional arm sets T is the time horizon. QGLOC achieves the smallest regret among hash-amenable algorithms.

GLB algorithms in Table 4.1. In Section 4.4, we present GLOC-TS, QGLOC, and their regret bound.

Note that, while hashing achieves a time complexity sublinear in N, there is a nontrivial overhead of computing the projections to determine the hash keys. As an extra contribution, we reduce this overhead by proposing a new sampling-based approximate inner product method. Our proposed sampling method has smaller variance than the state-of-the-art sampling method proposed by Jain et al. (2010); Kannan et al. (2009) when the vectors are normally distributed, which fits our setting where projection vectors are indeed normally distributed. Moreover, our method results in thinner tails in the distribution of estimation error than the existing method, which implies a better concentration. We elaborate more on reducing the computational complexity of QOFUL in Section 4.5.

4.2 Preliminaries

We review relevant backgrounds here. $\mathcal{A}$ refers to a GLB algorithm, and $\mathcal{B}$ refers to an online learning algorithm. Let $\mathcal{B}_d(S)$ be the d-dimensional Euclidean ball of radius S, which overloads the notation $\mathcal{B}$. Let $\mathbf{A}_{\cdot i}$ be the i-th column vector of a matrix $\mathbf{A}$. Define $||\mathbf{x}||_{\mathbf{A}} := \sqrt{\mathbf{x}^\top \mathbf{A}\mathbf{x}}$ and $\text{vec}(\mathbf{A}) := [\mathbf{A}_{\cdot 1}; \mathbf{A}_{\cdot 2}; \cdots; \mathbf{A}_{\cdot d}] \in \mathbb{R}^{d^2}$ Given a function $f : \mathbb{R} \to \mathbb{R}$, we denote by f' and f'' its first and second derivative, respectively. We define $[N] := \{1, 2, \ldots, N\}$.

Generalized Linear Model (GLM)

Consider modeling the reward y as one-dimensional exponential family such as Bernoulli or Poisson. When the feature vector $\mathbf{x}$ is believed to correlate with y, one popular modeling assumption is the generalized linear model (GLM) that turns the *natural parameter* of an exponential family model into $\mathbf{x}^\top \boldsymbol{\theta}^*$ where $\boldsymbol{\theta}^*$ is a parameter McCullagh and Nelder (1989):

$$\mathbb{P}(y \mid z = \mathbf{x}^\top \boldsymbol{\theta}^*) = \exp\left(\frac{yz - m(z)}{g(\tau)} + h(y, \tau)\right) , \tag{4.1}$$

where $\tau \in \mathbb{R}^+$ is a known scale parameter and m, g, and h are normalizers. It is known that $m'(z) = \mathbb{E}[y \mid z] =: \mu(z)$ and $m''(z) = \text{Var}(y \mid z)$. We call $\mu(z)$ the *inverse link* function. Throughout, we assume that the exponential family being used in a GLM has a *minimal representation*, which ensures that $m(z)$ is strictly convex (Wainwright and Jordan, 2008, Prop. 3.1). Then, the negative log likelihood (NLL) $\ell(z, y) := -yz + m(z)$ of a GLM is strictly convex. We refer to such GLMs as the *canonical* GLM. In the case of Bernoulli rewards $y \in \{0, 1\}$, $m(z) = \log(1 + \exp(z))$, $\mu(z) = (1 + \exp(-z))^{-1}$, and the NLL can be written as the logistic loss: $\log(1 + \exp(-y'(\mathbf{x}_t^\top \boldsymbol{\theta}^*)))$, where $y' = 2y - 1$.

Generalized Linear Bandits (GLB)

Recall that $\mathbf{x}_t$ is the arm chosen at time t by an algorithm. We assume that the arm set $\mathcal{X}_t$ can be of an infinite cardinality, although we focus on finite arm sets in hashing part of this chapter (Section 4.4). One can write down the reward model (4.1) in a different form:

$$y_t = \mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*) + \eta_t, \tag{4.2}$$

where η_t is conditionally R-sub-Gaussian given $\mathbf{x}_t$ and $\{(\mathbf{x}_s, \eta_s)\}_{s=1}^{t-1}$. For example, Bernoulli reward model has η_t as $1 - \mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*)$ w.p. $\mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*)$ and $-\mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*)$ otherwise. One can show that the sub-Gaussian scale R is determined by μ: $R = \sup_{z \in (-S,S)} \sqrt{\mu'(z)} \leq \sqrt{L}$, where L is the Lipschitz constant of μ. Throughout, we assume that each arm has ℓ_2-norm at most 1: $||\mathbf{x}||_2 \leq 1, \forall \mathbf{x} \in \mathcal{X}_t, \forall t$. Furthermore, $||\boldsymbol{\theta}^*||_2 \leq S$, where S is known. Let $\mathbf{x}_{t,*} := \max_{\mathbf{x} \in \mathcal{X}_t} \mathbf{x}^\top \boldsymbol{\theta}^*$. The performance of a GLB algorithm $\mathcal{A}$ is analyzed by the expected cumulative regret (or simply *regret*): $\text{Regret}_T^{\mathcal{A}} := \sum_{t=1}^T \mu(\mathbf{x}_{t,*}^\top \boldsymbol{\theta}^*) - \mu((\mathbf{x}_t^{\mathcal{A}})^\top \boldsymbol{\theta}^*)$, where $\mathbf{x}_t^{\mathcal{A}}$ makes the dependence on $\mathcal{A}$ explicit.

We remark that our results in this chapter hold true for a strictly larger family of distributions than the canonical GLM, which we call the *non-canonical* GLM and explain below. The condition is that the reward model follows (4.2) where the R is now independent from μ that satisfies the following:

Assumption 1. *μ is L-Lipschitz on $[-S, S]$ and continuously differentiable on $(-S, S)$. Furthermore, $\inf_{z\in(-S,S)} \mu'(z) = \kappa$ for some finite $\kappa > 0$ (thus μ is strictly increasing).*

Define $\mu'(z)$ at $\pm S$ as their limits. Under Assumption 1, m is defined to be an integral of μ. Then, one can show that m is κ-strongly convex on $\mathcal{B}_1(S)$. An example of the non-canonical GLM is the probit model for 0/1 reward where μ is the Gaussian CDF, which is popular and competitive to the Bernoulli GLM as evaluated by Li et al. (2012). Note that canonical GLMs satisfy Assumption 1.

4.3 Generalized Linear Bandits with Online Computation

We describe and analyze a new GLB algorithm called Generalized Linear Online-to-confidence-set Conversion (GLOC) that performs online computations, unlike existing GLB algorithms.

GLOC employs the optimism in the face of uncertainty principle, which dates back to Auer and Long (2002). That is, we maintain a confidence set C_t (defined below) that traps the true parameter $\boldsymbol{\theta}^*$ with high probability (w.h.p.) and choose the arm with the largest feasible reward given C_{t-1} as a constraint:

$$(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t) := \arg \max_{\mathbf{x}\in\mathcal{X}_t, \boldsymbol{\theta}\in C_{t-1}} \langle \mathbf{x}, \boldsymbol{\theta} \rangle \tag{4.3}$$

The main difference between GLOC and existing GLBs is in the computation of the C_t's. Prior methods involve "batch" computations that involve all past observations,

and so scale poorly with t. In contrast, GLOC takes in an *online* learner $\mathcal{B}$, and uses $\mathcal{B}$ as a co-routine instead of relying on a batch procedure to construct a confidence set. Specifically, at each time t GLOC feeds the loss function $\ell_t(\boldsymbol{\theta}) := \ell(\mathbf{x}_t^\top \boldsymbol{\theta}, y_t)$ into the learner $\mathcal{B}$ which then outputs its parameter prediction $\boldsymbol{\theta}_t$. Let $\mathbf{X}_t \in \mathbb{R}^{t \times d}$ be the design matrix consisting of $\mathbf{x}_1, \ldots, \mathbf{x}_t$. Define $\overline{\mathbf{V}}_t := \lambda \mathbf{I} + \mathbf{X}_t^\top \mathbf{X}_t$, where λ is the ridge parameter. Let $z_t := \mathbf{x}_t^\top \boldsymbol{\theta}_t$ and $\mathbf{z}_t := [z_1; \cdots; z_t]$. Let $\widehat{\boldsymbol{\theta}}_t := \overline{\mathbf{V}}_t^{-1} \mathbf{X}_t^\top \mathbf{z}_t$ be the ridge regression estimator taking $\mathbf{z}_t$ as responses. Theorem 1 below is the key result for constructing our confidence set C_t, which is a function of the parameter predictions $\{\boldsymbol{\theta}_s\}_{s=1}^t$ and the online (OL) regret bound B_t of the learner $\mathcal{B}$. All the proofs are in the supplementary material (SM).

Theorem 1. *(Generalized Linear Online-to-Confidence-Set Conversion) Suppose we feed loss functions $\{\ell_s(\boldsymbol{\theta})\}_{s=1}^t$ into online learner $\mathcal{B}$. Let $\boldsymbol{\theta}_s$ be the parameter predicted at time step s by $\mathcal{B}$. Assume that $\mathcal{B}$ has an OL regret bound B_t: $\forall \boldsymbol{\theta} \in \mathcal{B}_d(S), \forall t \geq 1$,*

$$\textstyle\sum_{s=1}^t \ell_s(\boldsymbol{\theta}_s) - \ell_s(\boldsymbol{\theta}) \leq B_t \ . \tag{4.4}$$

Let $\alpha(B_t) := 1 + \frac{4}{\kappa} B_t + \frac{8R^2}{\kappa^2} \log(\frac{2}{\delta}\sqrt{1 + \frac{2}{\kappa} B_t + \frac{4R^4}{\kappa^4 \delta^2}})$. Then, with probability (w.p.) at least $1 - \delta$,

$$\forall t \geq 1, ||\boldsymbol{\theta}^* - \widehat{\boldsymbol{\theta}}_t||_{\overline{\mathbf{V}}_t}^2 \leq \alpha(B_t) + \lambda S^2 - \left(||\mathbf{z}_t||_2^2 - \widehat{\boldsymbol{\theta}}_t^\top \mathbf{X}_t^\top \mathbf{z}_t\right) =: \beta_t \ . \tag{4.5}$$

Note that the center of the ellipsoid is the ridge regression estimator on the predicted natural parameters $z_s = \mathbf{x}_s^\top \boldsymbol{\theta}_s$ rather than the rewards. Theorem 1 motivates the following confidence set:

$$C_t := \{\boldsymbol{\theta} \in \mathbb{R}^d : ||\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t||_{\overline{\mathbf{V}}_t}^2 \leq \beta_t\} \tag{4.6}$$

which traps $\boldsymbol{\theta}^*$ for all $t \geq 1$, w.p. at least $1 - \delta$. See Algorithm 6 for pseudocode.

One way to solve the optimization problem (4.3) is to define the function $\boldsymbol{\theta}(\mathbf{x}) := \max_{\boldsymbol{\theta} \in C_{t-1}} \mathbf{x}^\top \boldsymbol{\theta}$, and then use the Lagrangian method to write:

$$\mathbf{x}_t^{\text{GLOC}} := \arg\max_{\mathbf{x} \in \mathcal{X}_t} \mathbf{x}^\top \widehat{\boldsymbol{\theta}}_{t-1} + \sqrt{\beta_{t-1}} ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}} \ . \tag{4.7}$$

We prove the regret bound of GLOC in the following theorem.

Theorem 2. *Let $\{\overline{\beta}_t\}$ be a nondecreasing sequence such that $\overline{\beta}_t \geq \beta_t$. Then, w.p. at least $1 - \delta$,*

$$\text{Regret}_T^{\text{GLOC}} = O\left(L\sqrt{\overline{\beta}_T dT \log T}\right)$$

Algorithm 6 GLOC

Input: $R > 0$, $\delta \in (0,1)$, $S > 0$, $\lambda > 0$, $\kappa > 0$, an online learner $\mathcal{B}$ with known regret bounds $\{B_t\}_{t \geq 1}$.
Set $\overline{\mathbf{V}}_0 = \lambda \mathbf{I}$.
for $t = 1, 2, \ldots$ **do**
  Compute $\mathbf{x}_t$ by solving (4.3).
  Pull $\mathbf{x}_t$ and then observe y_t.
  Receive $\boldsymbol{\theta}_t$ from $\mathcal{B}$.
  Feed into $\mathcal{B}$ the loss $\ell_t(\boldsymbol{\theta}) = \ell(\mathbf{x}_t^\top \boldsymbol{\theta}, y_t)$.
  Update $\overline{\mathbf{V}}_t = \overline{\mathbf{V}}_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$ and $z_t = \mathbf{x}_t^\top \boldsymbol{\theta}_t$
  Compute $\widehat{\boldsymbol{\theta}}_t = \overline{\mathbf{V}}_t^{-1} \mathbf{X}_t^\top \mathbf{z}_t$ and β_t as in (4.5).
  Define C_t as in (4.6).
end for

Algorithm 7 ONS-GLM

Input: $\kappa > 0$, $\epsilon > 0$, $S > 0$.
$\mathbf{A}_0 = \epsilon \mathbf{I}$.
Set $\boldsymbol{\theta}_1 \in \mathcal{B}_d(S)$ arbitrarily.
for $t = 1, 2, 3, \ldots$ **do**
  Output $\boldsymbol{\theta}_t$.
  Observe $\mathbf{x}_t$ and y_t.
  Incur loss $\ell(\mathbf{x}_t^\top \boldsymbol{\theta}_t, y_t)$.
  $\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$
  $\boldsymbol{\theta}'_{t+1} = \boldsymbol{\theta}_t - \frac{\ell'(\mathbf{x}_t^\top \boldsymbol{\theta}_t, y_t)}{\kappa} \mathbf{A}_t^{-1} \mathbf{x}_t$
  $\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta} \in \mathcal{B}_d(S)} ||\boldsymbol{\theta} - \boldsymbol{\theta}'_{t+1}||_{\mathbf{A}_t}^2$
end for

Although any low-regret online learner can be combined with GLOC, one would

like to ensure that $\overline{\beta}_T$ is $O(\text{polylog}(T))$ in which case the total regret can be bounded by $\tilde{O}(\sqrt{T})$. This means that we must use online learners whose OL regret grows logarithmically in T such as Hazan et al. (2007a); Orabona et al. (2012). In this work, we consider the online Newton step (ONS) algorithm Hazan et al. (2007a).

Online Newton Step (ONS) for Generalized Linear Models

Note that ONS requires the loss functions to be α-exp-concave. One can show that $\ell_t(\boldsymbol{\theta})$ is α-exp-concave (Hazan et al., 2007a, Sec. 2.2). Then, GLOC can use ONS and its OL regret bound to solve the GLB problem. However, motivated by the fact that the OL regret bound B_t appears in the radius $\sqrt{\beta_t}$ of the confidence set while a tighter confidence set tends to reduce the bandit regret in practice, we derive a tight data-dependent OL regret bound tailored to GLMs.

We present our version of ONS for GLMs (ONS-GLM) in Algorithm 7. $\ell'(z, y)$ is the first derivative w.r.t. z and the parameter ϵ is for inverting matrices conveniently (usually $\epsilon = 1$ or 0.1). The only difference from the original ONS Hazan et al. (2007a) is that we rely on the strong convexity of $m(z)$ instead of the α-exp-concavity of the loss thanks to the GLM structure.[3] Theorem 3 states that we achieve the desired poly-logarithmic regret in T.

Theorem 3. *Define* $g_s := \ell'(\mathbf{x}_s^\top \boldsymbol{\theta}_s, y_s)$. *The regret of ONS-GLM satisfies, for any* $\epsilon > 0$ *and* $t \geq 1$,

$$\sum_{s=1}^{t} \ell_s(\boldsymbol{\theta}_s) - \ell_s(\boldsymbol{\theta}^*) \leq \frac{1}{2\kappa} \sum_{s=1}^{t} g_s^2 ||\mathbf{x}_s||^2_{\mathbf{A}_s^{-1}} + 2\kappa S^2 \epsilon =: B_t^{\text{ONS}} ,$$

where $B_t^{\text{ONS}} = O(\frac{L^2 + R^2 \log(t)}{\kappa} d \log t), \forall t \geq 1$ *w.h.p. If* $\max_{s \geq 1} |\eta_s|$ *is bounded by* $\bar{R}$ *w.p. 1,* $B_t^{\text{ONS}} = O(\frac{L^2 + \bar{R}^2}{\kappa} d \log t)$.

We emphasize that the OL regret bound is data-dependent. A confidence set constructed by combining Theorem 1 and Theorem 3 directly implies the following regret

[3] A similar change to ONS has been applied in Gentile and Orabona (2014); Zhang et al. (2016).

bound of GLOC with ONS-GLM.

Corollary 1. *Define* β_t^{ONS} *by replacing* B_t *with* B_t^{ONS} *in* (4.5)*. With probability at least* $1-2\delta$*,*

$$\forall t \geq 1, \boldsymbol{\theta}^* \in C_t^{\text{ONS}} := \left\{ \boldsymbol{\theta} \in \mathbb{R}^d : ||\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t||^2_{\overline{\mathbf{V}}_t} \leq \beta_t^{\text{ONS}} \right\} . \tag{4.8}$$

Corollary 2. *Run GLOC with* C_t^{ONS}*. Then, w.p. at least* $1-2\delta$*,* $\forall T \geq 1$*,* $\text{Regret}_T^{\text{GLOC}} = \hat{O}\left(\frac{L(L+R)}{\kappa} d\sqrt{T}\log^{3/2}(T)\right)$ *where* $\hat{O}$ *ignores* $\log\log(t)$*. If* $|\eta_t|$ *is bounded by* $\bar{R}$*,* $\text{Regret}_T^{\text{GLOC}} = \hat{O}\left(\frac{L(L+\bar{R})}{\kappa} d\sqrt{T}\log(T)\right)$.

We make regret bound comparisons ignoring $\log\log T$ factors. For generic arm sets, our dependence on d is optimal for linear rewards Rusmevichientong and Tsitsiklis (2010). For the Bernoulli GLM, our regret has the same order as Zhang et al. (2016). One can show that the regret of Filippi et al. (2010) has the same order as ours if we use their assumption that the reward y_t is bounded by $R_{\max}$ (e.g., $\bar{R} = 1/2$ for Bernoulli). For unbounded noise, Li et al. (2017) have regret $O((LR/\kappa)d\sqrt{T}\log T)$, which is $\sqrt{\log T}$ factor smaller than ours and has LR in place of $L(L+R)$. While $L(L+R)$ could be an artifact of our analysis, the gap is not too large for canonical GLMs. Let L be the smallest Lipschitz constant of μ. Then, $R = \sqrt{L}$. If $L \leq 1$, R satisfies $R > L$, and so $L(L+R) = O(LR)$. If $L > 1$, then $L(L+R) = O(L^2)$, which is larger than $LR = O(L^{3/2})$. For the Gaussian GLM with known variance σ^2, $L = R = 1$.[4] For finite arm sets, SupCB-GLM of Li et al. (2017) achieves regret of $\tilde{O}(\sqrt{dT\log N})$ that has a better scaling with d but is not a practical algorithm as it wastes a large number of arm pulls. Finally, we remark that none of the existing GLB algorithms are scalable to large T. Zhang et al. (2016) is scalable to large T, but is restricted to the Bernoulli GLM; e.g., theirs does not allow the probit model (non-canonical GLM) that is popular and shown to be competitive to the Bernoulli GLM Li et al. (2012).

Discussion The trick of obtaining a confidence set from an online learner ap-

[4] The reason why R is not σ here is that the sufficient statistic of the GLM is y/σ, which is equivalent to dealing with the normalized reward. Then, σ appears as a factor in the regret bound.

peared first in Dekel et al. (2010, 2012) for the linear model, and then was used in Crammer and Gentile (2013); Gentile and Orabona (2014); Zhang et al. (2016). GLOC is slightly different from these studies and rather close to Abbasi-Yadkori et al. (2012) in that the confidence set is a function of a known regret bound. This generality frees us from re-deriving a confidence set for every online learner. Our result is essentially a nontrivial extension of Abbasi-Yadkori et al. (2012) to GLMs.

4.4 Hash-Amenable Generalized Linear Bandits

We now turn to a setting where the arm set is finite but very large. For example, imagine an interactive retrieval scenario Rui et al. (1998); Konyushkova and Glowacka (2013); Ahukorala et al. (2015) where a user is shown K images (e.g., shoes) at a time and provides relevance feedback (e.g., yes/no or 5-star rating) on each image, which is repeated until the user is satisfied. In this chapter, we focus on showing one image (i.e., arm) at a time.[5] Most existing algorithms require maximizing an objective function (e.g., (4.7)), the complexity of which scales linearly with the number N of arms. This can easily become prohibitive for large numbers of images. Furthermore, the system has to perform real-time computations to promptly choose which image to show the user in the next round. Thus, it is critical for a practical system to have a time complexity sublinear in N.

One naive approach is to select a subset of arms ahead of time, such as volumetric spanners Hazan et al. (2007b). However, this is specialized for an efficient exploration only and can rule out a large number of good arms. Another option is to use hashing methods. Locality-sensitive hashing and Maximum Inner Product Search (MIPS) are effective and well-understood tools but can only be used when the objective function is a distance or an inner product computation; (4.7) cannot be written in this form. In this section, we consider alternatives to GLOC which are compatible with hashing.

[5] One image at a time is a simplification of the practical setting. One can extend it to showing multiple images at a time, which is a special case of the combinatorial bandits of Qin et al. (2014).

Thompson Sampling We present a Thompson sampling (TS) version of GLOC called GLOC-TS that chooses an arm $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}_t} \mathbf{x}^\top \dot{\boldsymbol{\theta}}_t$ where $\dot{\boldsymbol{\theta}}_t \sim \mathcal{N}(\widehat{\boldsymbol{\theta}}_{t-1}, \beta_{t-1}\overline{\mathbf{V}}_{t-1}^{-1})$. TS is known to perform well in practice Chapelle and Li (2011) and can solve the polytope arm set case in polynomial time[6] whereas algorithms that solve an objective function like (4.3) (e.g., Abbasi-Yadkori et al. (2012)) cannot since they have to solve an NP-hard problem Agrawal and Goyal (2013). We present the regret bound of GLOC-TS below. Due to space constraints, we present the pseudocode and the full version of the result in SM.

Theorem 4. *(Informal) If we run GLOC-TS with* $\dot{\boldsymbol{\theta}}_t \sim \mathcal{N}(\widehat{\boldsymbol{\theta}}_{t-1}, \beta_{t-1}^{\text{ONS}}\overline{\mathbf{V}}_{t-1}^{-1})$, $\text{Regret}_T^{\text{GLOC-TS}} = \hat{O}\left(\frac{L(L+R)}{\kappa} d^{3/2}\sqrt{T}\log^{3/2}(T)\right)$ *w.h.p. If* η_t *is bounded by* $\bar{R}$*, then* $\hat{O}\left(\frac{L(L+\bar{R})}{\kappa} d^{3/2}\sqrt{T}\log(T)\right)$.

Notice that the regret now scales with $d^{3/2}$ as expected from the analysis of linear TS Agrawal and Goyal (2012), which is higher than scaling with d of GLOC. This is concerning in the interactive retrieval or product recommendation scenario since the relevance of the shown items is harmed, which makes us wonder if one can improve the regret without loosing the hash-amenability.

Quadratic GLOC We now propose a new hash-amenable algorithm called Quadratic GLOC (QGLOC). Recall that GLOC chooses the arm $\mathbf{x}^{\text{GLOC}}$ by (4.7). Define $r = \min_{\mathbf{x}\in\mathcal{X}} ||\mathbf{x}||_2$ and

$$\overline{m}_{t-1} := \min_{\mathbf{x}:||\mathbf{x}||_2\in[r,1]} ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}} , \tag{4.9}$$

which is r times the square root of the smallest eigenvalue of $\overline{\mathbf{V}}_{t-1}^{-1}$. It is easy to see that $\overline{m}_{t-1} \leq ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}}$ for all $\mathbf{x} \in \mathcal{X}$ and that $\overline{m}_{t-1} \geq r/\sqrt{t+\lambda}$ using the definition of $\overline{\mathbf{V}}_{t-1}$. There is an alternative way to define $\overline{m}_{t-1}$ without relying on r, which we present in SM.

Let $c_0 > 0$ be the exploration-exploitation tradeoff parameter (elaborated upon later). At time t, QGLOC chooses the arm

[6] ConfidenceBall$_1$ algorithm of Dani et al. (2008) can solve the problem in polynomial time as well.

$$\mathbf{x}_t^{\text{QGLOC}} := \arg\max_{\mathbf{x}\in\mathcal{X}_t} \langle \widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}\rangle + \frac{\beta_{t-1}^{1/4}}{4c_0\overline{m}_{t-1}}||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} = \arg\max_{\mathbf{x}\in\mathcal{X}_t} \langle \mathbf{q}_t, \phi(\mathbf{x})\rangle \ , \tag{4.10}$$

where $\mathbf{q}_t = [\widehat{\boldsymbol{\theta}}_{t-1}; \text{vec}(\frac{\beta_{t-1}^{1/4}}{4c_0\overline{m}_{t-1}}\overline{\mathbf{V}}_{t-1}^{-1})] \in \mathbb{R}^{d+d^2}$ and $\phi(\mathbf{x}) := [\mathbf{x}; \text{vec}(\mathbf{x}\mathbf{x}^\top)]$. The key property of QGLOC is that the objective function is now quadratic in $\mathbf{x}$, thus the name *Quadratic* GLOC, and can be written as an inner product. Thus, QGLOC is hash-amenable. We present the regret bound of QGLOC (4.10) in Theorem 5. The key step of the proof is that the QGLOC objective function (4.10) plus $c_0\beta^{3/4}\overline{m}_{t-1}$ is a tight upper bound of the GLOC objective function (4.7).

Theorem 5. *Run QGLOC with* C_t^{ONS}. *Then, w.p. at least* $1-2\delta$, $\text{Regret}_T^{QGLOC} = O\left(\left(\frac{1}{c_0}\left(\frac{L+R}{\kappa}\right)^{1/2} + c_0\left(\frac{L+R}{\kappa}\right)^{3/2}\right) Ld^{5/4}\sqrt{T}\log^2(T)\right)$. *By setting* $c_0 = \left(\frac{L+R}{\kappa}\right)^{-1/2}$, *the regret bound is* $O(\frac{L(L+R)}{\kappa}d^{5/4}\sqrt{T}\log^2(T))$.

Note that one can have a better dependence on $\log T$ when η_t is bounded (available in the proof). The regret bound of QGLOC is a $d^{1/4}$ factor improvement over that of GLOC-TS; see Table 4.1. Furthermore, in (4.10) c_0 is a free parameter that adjusts the balance between the exploitation (the first term) and exploration (the second term). Interestingly, the regret guarantee *does not break down* when adjusting c_0 in Theorem 5. Such a characteristic is not found in existing algorithms but is attractive to practitioners, which we elaborate in SM.

Maximum Inner Product Search (MIPS) Hashing

While MIPS hashing algorithms such as Shrivastava and Li (2014, 2015); Neyshabur and Srebro (2015) can solve (4.10) in time sublinear in N, these necessarily introduce an approximation error. Ideally, one would like the following guarantee on the error with probability at least $1-\delta_{\text{H}}$:

Definition 3. *Let* $\mathcal{X} \subseteq \mathbb{R}^{d'}$ *satisfy* $|\mathcal{X}| < \infty$. *A data point* $\tilde{\mathbf{x}} \in \mathcal{X}$ *is called* c_{H}*-MIPS w.r.t. a given query* $\mathbf{q}$ *if it satisfies* $\langle \mathbf{q}, \tilde{\mathbf{x}}\rangle \geq c_{\text{H}} \cdot \max_{\mathbf{x}\in\mathcal{X}} \langle \mathbf{q}, \mathbf{x}\rangle$ *for some* $c_{\text{H}} < 1$. *An algorithm is called* c_{H}*-MIPS if, given a query* $\mathbf{q} \in \mathbb{R}^{d'}$, *it retrieves* $\mathbf{x} \in \mathcal{X}$ *that is*

c_{H}-*MIPS w.r.t.* $\mathbf{q}$.

Unfortunately, existing MIPS algorithms do not directly offer such a guarantee, and one must build a series of hashing schemes with varying hashing parameters like Har-Peled et al. (2012). Under the fixed budget setting T, we elaborate our construction that is simpler than Har-Peled et al. (2012) in SM.

Time and Space Complexity

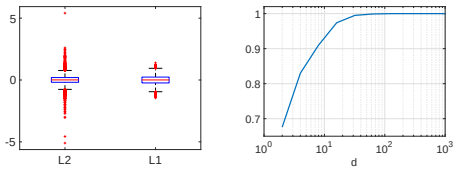
Our construction involves saving Gaussian projection vectors that are used for determining hash keys and saving the buckets containing pointers to the actual arm vectors. The time complexity for retrieving a c_{H}-MIPS solution involves determining hash keys and evaluating inner products with the arms in the retrieved buckets. Let $\rho^* < 1$ be an optimized value for the hashing (see Shrivastava and Li (2014) for detail). The time complexity for d'-dimensional vectors is $O\left(\log\left(\frac{\log(dT)}{\log(c_{\mathrm{H}}^{-1})}\right) N^{\rho^*} \log(N) d'\right)$, and the space complexity (except the original data) is $O\left(\frac{\log(dT)}{\log(c_{\mathrm{H}}^{-1})} N^{\rho^*}(N + \log(N)d')\right)$. While the time and space complexity grows with the time horizon T, the dependence is mild; $\log\log(T)$ and $\log(T)$, respectively. QGLOC uses $d' = d + d^2$,[7] and GLOC-TS uses $d' = d'$. While both achieve a time complexity sublinear in N, the time complexity of GLOC-TS scales with d that is better than scaling with d^2 of QGLOC. However, GLOC-TS has a $d^{1/4}$-factor worse regret bound than QGLOC.

Discussion

While it is reasonable to incur small errors in solving the arm selection criteria like (4.10) and sacrifice some regret in practice, the regret bounds of QGLOC and GLOC-TS do not hold anymore. Though not the focus of this chapter, we prove a regret bound under the presence of the hashing error in the fixed budget setting for QGLOC; see SM. Although the result therein has an inefficient space complexity that is linear in T, it provides the first low regret bound with time sublinear in N, to our knowledge.

[7] Note that this does not mean we need to store $\mathrm{vec}(\mathbf{x}\mathbf{x}^\top)$ since an inner product with it is structured.

4.5 Approximate Inner Product Computations with L1 Sampling



(a) (b)

Figure 4.1: (a) A box plot of estimators. L1 and L2 have the same variance, but L2 has thicker tails. (b) The frequency of L1 inducing smaller variance than L2 in 1000 trials. After 100 dimensions, L1 mostly has smaller variance than L2.

While hashing allows a time complexity sublinear in N, it performs an additional computation for determining the hash keys. Consider a hashing with U tables and length-k hash keys. Given a query $\mathbf{q}$ and projection vectors $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(Uk)}$, the hashing computes $\mathbf{q}^\top \mathbf{a}^{(i)}$, $\forall i \in [Uk]$ to determine the hash key of $\mathbf{q}$. To reduce such an overhead, approximate inner product methods like Jain et al. (2010); Kannan et al. (2009) are attractive since hash keys are determined by discretizing the inner products; small inner product errors often do not alter the hash keys.

In this section, we propose an improved approximate inner product method called *L1 sampling* which we claim is more accurate than the sampling proposed by Jain et al. (2010), which we call *L2 sampling*. Consider an inner product $\mathbf{q}^\top \mathbf{a}$. The main idea is to construct an unbiased estimate of $\mathbf{q}^\top \mathbf{a}$. That is, let $\mathbf{p} \in \mathbb{R}^d$ be a probability vector. Let

$$i_k \overset{\text{i.i.d.}}{\sim} \text{Multinomial}(\mathbf{p}) \quad \text{and} \quad G_k := q_{i_k} a_{i_k} / p_{i_k}, \; k \in [m] \; . \tag{4.11}$$

It is easy to see that $\mathbb{E} G_k = \mathbf{q}^\top \mathbf{a}$. By taking $\frac{1}{m} \sum_{k=1}^m G_k$ as an estimate of $\mathbf{q}^\top \mathbf{a}$, the time complexity is now $O(mUk)$ rather than $O(d'Uk)$. The key is to choose the right $\mathbf{p}$. L2 sampling uses $\mathbf{p}^{(\text{L2})} := [q_i^2 / ||\mathbf{q}||_2^2]_i$. Departing from L2, we propose $\mathbf{p}^{(\text{L1})}$ that we call L1 sampling and define as follows:

$$\mathbf{p}^{(\text{L1})} := [|q_1|; \cdots; |q_{d'}|] / ||\mathbf{q}||_1 \; . \tag{4.12}$$

We compare L1 with L2 in two different point of view. Due to space constraints, we summarize the key ideas and defer the details to SM.

The first is on their concentration of measure. Lemma 1 below shows an error bound of L1 whose failure probability decays exponentially in m. This is in contrast to decaying polynomially of L2 Jain et al. (2010), which is inferior.[8]

Lemma 1. *Define G_k as in* (4.11) *with $\mathbf{p} = \mathbf{p}^{(\mathrm{L1})}$. Then, given a target error $\epsilon > 0$,*

$$\mathbb{P}\left(\left|\frac{1}{m}\sum_{k=1}^{m} G_k - \mathbf{q}^\top \mathbf{a}\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{m\epsilon^2}{2||\mathbf{q}||_1^2||\mathbf{a}||_{\max}^2}\right) \tag{4.13}$$

To illustrate such a difference, we fix $\mathbf{q}$ and $\mathbf{a}$ in 1000 dimension and apply L2 and L1 sampling 20K times each with $m = 5$ where we scale down the L2 distribution so its variance matches that of L1. Figure 4.1(a) shows that L2 has thicker tails than L1. Note this is not a pathological case but a typical case for Gaussian $\mathbf{q}$ and $\mathbf{a}$. This confirms our claim that L1 is safer than L2.

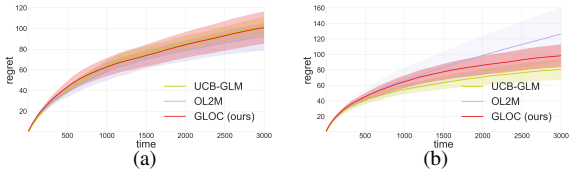
Another point of comparison is the variance of L2 and L1. We show that the variance of L1 may or may not be larger than L2 in SM; there is no absolute winner. However, if $\mathbf{q}$ and $\mathbf{a}$ follow a Gaussian distribution, then L1 induces smaller variances than L2 for large enough d.

Figure 4.1(b) confirms such a result. The actual gap between the variance of L2 and L1 is also nontrivial under the Gaussian assumption. For instance, with $d = 200$, the average variance of G_k induced by L2 is 0.99 whereas that induced by L1 is 0.63 on average. Although a stochastic assumption on the vectors being inner-producted is often unrealistic, in our work we deal with projection vectors $\mathbf{a}$ that are truly normally distributed.

4.6 Experiments

We now show our experiment results comparing GLB algorithms and hash-amenable algorithms.

[8] In fact, one can show a bound for L2 that fails with exponentially-decaying probability. However, the bound introduces a constant that can be arbitrarily large, which makes the tails thick. We provide details on this in SM.



(a) (b)

Algorithm	Cum. Regret
QGLOC	266.6 (±19.7)
QGLOC-Hash	285.0 (±30.3)
GLOC-TS	277.0 (±36.1)
GLOC-TS-Hash	289.1 (±28.1)

(c)

Figure 4.2: Cumulative regrets with confidence intervals under the (a) logit and (b) probit model. (c) Cumulative regrets with confidence intervals of hash-amenable algorithms.

GLB Algorithms We compare GLOC with two different algorithms: UCB-GLM Li et al. (2017) and Online Learning for Logit Model (OL2M) Zhang et al. (2016).[9] For each trial, we draw $\boldsymbol{\theta}^* \in \mathbb{R}^d$ and N arms ($\mathcal{X}$) uniformly at random from the unit sphere. We set $d = 10$ and $\mathcal{X}_t = \mathcal{X}, \forall t \geq 1$. Note it is a common practice to scale the confidence set radius for bandits Chapelle and Li (2011); Li et al. (2012). Following Zhang et al. (2016), for OL2M we set the squared radius $\gamma_t = c \log(\det(\mathbf{Z}_t)/\det(\mathbf{Z}_1))$, where c is a tuning parameter. For UCB-GLM, we set the radius as $\alpha = \sqrt{cd \log t}$. For GLOC, we replace β_t^{ONS} with $c \sum_{s=1}^{t} g_s^2 ||\mathbf{x}_s||^2_{\mathbf{A}_s^{-1}}$. While parameter tuning in practice is nontrivial, for the sake of comparison we tune $c \in \{10^1, 10^{0.5}, \ldots, 10^{-3}\}$ and report the best one. We perform 40 trials up to time $T = 3000$ for each method and compute confidence bounds on the regret.

We consider two GLM rewards: (i) the logit model (the Bernoulli GLM) and (ii) the probit model (non-canonical GLM) for 0/1 rewards that sets μ as the probit function. Since OL2M is for the logit model only, we expect to see the consequences of model mismatch in the probit setting. For GLOC and UCB-GLM, we specify the correct reward model. We plot the cumulative regret under the logit model in Figure 4.2(a). All three methods perform similarly, and we do not find any statistically significant difference based on paired t test. The result for the probit model in Figure 4.2(b) shows that OL2M indeed has higher regret than both GLOC and UCB-GLM due to the model mismatch in the probit setting. Specifically, we verify that at $t = 3000$ the difference between the regret of UCB-GLM and OL2M is statistically significant. Furthermore,

[9] We have chosen UCB-GLM over GLM-UCB of Filippi et al. (2010) as UCB-GLM has a lower regret bound.

OL2M exhibits a significantly higher variance in the regret, which is unattractive in practice. This shows the importance of being generalizable to *any* GLM reward. Note we observe a big increase in running time for UCB-GLM compared to OL2M and GLOC.

Hash-Amenable GLBs To compare hash-amenable GLBs, we use the logit model as above but now with N=100,000 and T=5000. We run QGLOC, QGLOC with hashing (QGLOC-Hash), GLOC-TS, and GLOC-TS with hashing (GLOC-TS-Hash), where we use the hashing to compute the objective function (e.g., (4.10)) on just 1% of the data points and save a significant amount of computation. Details on our hashing implementation is found in SM. Figure 4.2(c) summarizes the result. We observe that QGLOC-Hash and GLOC-TS-Hash increase regret from QGLOC and GLOC-TS, respectively, but only moderately, which shows the efficacy of hashing.

4.7 Future Work

In this chapter, we have proposed scalable algorithms for the GLB problem: (i) for large time horizon T and (ii) for large number N of arms. There exists a number of interesting future work. First, we would like to extend the GLM rewards to the single index models Kalai and Sastry (2009) so one does not need to know the function μ ahead of time under mild assumptions. Second, closing the regret bound gap between QGLOC and GLOC without loosing hash-amenability would be interesting: i.e., develop a hash-amenable GLB algorithm with $O(d\sqrt{T})$ regret. In this direction, a first attempt could be to design a hashing scheme that can directly solve (4.7) approximately.

Chapter 5

Linear Bandits with Feature Feedback [1]

[1] To be submitted to ICML 2018

5.1 Introduction

In this chapter the focus will be to study how we can run bandits when the number of features per arm is very large. Consider the scenario of recommending documents. Typical bag-of-words model such as TF-IDF has the number of features in the order of tens of thousands of words. Now, any given document may only contain a small subset of words. Nevertheless, taken as a whole, the total number of features is large.

This constraint makes it unfeasible to run state of the art algorithms such as OFUL Abbasi-Yadkori et al. (2011) since we would have to maintain and update a $d \times d$ matrix at every stage.

The approach taken in this chapter is to augment the available inner product information with feature feedback (to be explained in detail) in order to hone in on a subset of the features that are important to the user answering the questions. Here the underlying assumption is that the hidden vector $\boldsymbol{\theta}_*$ is a sparse vector.

There has been previous work in the area of sparse linear bandits: Carpentier and Munos (2012) consider a two-stage approach of first estimating the support with high probability before running standard linear bandits on the estimated support or of Abbasi-Yadkori et al. (2012) who use a black-box approach. The second approach still needs us to run the bandit algorithm in the intrinsic dimension of d which is not practical in very high dimensions.

The second approach does reduce the number of features that we consider but we would like to propose an alternative that is not explicitly two-stage and one that does not require the knowledge of the sparsity-level k.

There has been similar work done in the area of bandits on a graph (Valko et al., 2014). In that work they look at the notion of an effective dimension that is akin to sparsity. But the key difference is that they are looking at smoothness over a graph and they do not use an feature feedback information. Besides, they do all their computations

in the ambient dimension. In our case, we want to avoid this last issue since running linear bandits on more than an order of 1000 dimensions can be slow and not feasible practically when serving multiple users at the same time.

The closest work to this considers a similar problem but in the setting setting of learning a classifier (Poulis and Dasgupta, 2017). In that work, they quantify the complexity of learning classifiers with feature feedback. Our work is similar in models but we consider the problem of minimizing regret over time for contextual bandit algorithms.

We propose a way to exploit feedback from the users while running bandit algorithms. Our model is that the users are able to marka subset of the features to indicate that they are relevant to their search. These can be features that are positively or negatively correlated with their search. In practice, this can be done by highlighting important words in a document by a user. These words can be positive words to reinforce search in that direction or negative words to avoid in the document search. The exact model of feature feedback is discussed later in this chapter.

As an example, suppose the user is looking for articles about Wisconsin Football, then words such as Badgers, Football, Camp Randall may be relevant but also words such as politics, technology, and stock markets may be relevant since they both help in either finding or eliminating articles. But words such as tree, air, person, may be in common with a lot of the articles and therefore less relevant to performing the search. The goal is the give the user a tool to speed up their search.

5.2 Feature Feedback Model

Let us consider the following motivational situation: a user is looking for articles about Wisconsin Basketball or Football from a corpus of news documents. The words such as Wisconsin, Basketball, Football, Badgers or specific names of players may be relevant for positive words. Similarly, words as politics, congress, biology, Michigan, Lacrosse etc. may be deemed negative relevant words. Other words such as act, play, time, may be irrelevant.

We would like to show the user documents and get their feedback in terms of whether they like it or not (logistic model) or how much they like it (inner product model). We would also like them to highlight a few words, if they can find them to help orient the search.

The underlying assumptions about feature feedback are:

Assumption 2 (Sparsity)**.** *The hidden weight vector $\boldsymbol{\theta}_* \in \mathbf{R}^d$ is k-sparse for $k \ll d$.*

Assumption 3 (Discoverability)**.** *If we select an arm $\mathbf{x} \in \mathcal{X}$ uniformly at random, then the probability that a relevant feature is present in it is lower bounded by some $p > 0$.*

By Assumption 2, we are assuming that there are at most k relevant features in order to find the document the user is searching for. By Assumption 3, we are saying that while all documents may not have relevant words, we are able to find them with a non-zero probability when searching through documents at random.

Therefore, we will have the following set up: we have a set of arms, $\mathcal{X} \subseteq \mathbf{R}^d$ that we can propose to the users. The user has a hidden weight vector $\boldsymbol{\theta}_* \in \mathbf{R}^d$ that is k-sparse. We will further assume that $\|\boldsymbol{\theta}_*\| \leq S$ and the arms too are bounded in norm: $\forall \mathbf{x} \in \mathcal{X}, \|\mathbf{x}\| \leq L$. In order to make use of Theorem 3 from (Abbasi-Yadkori et al., 2011), we will need to further assume that the rewards $y_t \in [-1, 1]$. This reward is modeled as:

$$y_t = \langle \mathbf{x}_t, \boldsymbol{\theta}_* \rangle + \eta_t$$

where, $\mathbf{x}_t \in \mathcal{X}$ is the arm chosen at time t, and η_t are i.i.d sub-Gaussian random variables or parameter R. Besides this, at each time-step we also get $\mathcal{I}_t \subseteq \text{supp}(\boldsymbol{\theta}_*)$ which is the relevance feedback information. The model we have further specifies that $\forall j \in \text{supp}(\boldsymbol{\theta}_*), \mathbb{P}(j \in \mathcal{I}_t) = p$. It would be easy to extend our theory to the model $\forall j \in \text{supp}(\boldsymbol{\theta}_*), \mathbb{P}(j \in \mathcal{I}_t) \geq p$.

Therefore, we assume that the probability that we find a relevant feature at random is lower bounded by some p. We need this assumption to make sure that we can find all the relevant vectors.

5.3 Hybrid OFUL

We being by reminding ourselves of the following theorem that bounds the regret of the OFUL algorithm from Abbasi-Yadkori et al. (2011):

Theorem 6. *Assume that for all t and all $\mathbf{x} \in \mathcal{X}_t$, $\langle \mathbf{x}, \boldsymbol{\theta}_* \rangle \in [-1, 1]$. Then with probability at least $1 - \delta$, the regret of OFUL satisfies:*

$$\forall t, R_t \leq 4\sqrt{td\log(\lambda + tL/d)}\left(\lambda^{1/2}S + R\sqrt{2\log(1/\delta) + d\log(1 + tL/(\lambda d))}\right)$$

We will use the same Theorem 6 but try to reduce the dependence on the dimension d to k. In order to do so, we must discover the support of $\boldsymbol{\theta}_*$. This requires feedback information from the previous chapter.

As a reminder 8 is the algorithm described in (Abbasi-Yadkori et al., 2011). We will combine this with a form of ϵ-greedy algorithm (Sutton and Barto, 1998). The idea will be that we will set a portion of the arm pulls to random arms in order to guarantee that we can find all the relevant arms, and the remaining portion of time we will run OFUL on the identified relevant dimensions. We will also reduce the proportion of the random arm pulls over time in order guarantee that the regret remains sub-linear in time.

We propose two different algorithms that exploit feature feedback. The first, ϵ-Greedy Epoch OFUL (Algorithm 9) runs in epochs of doubling length. During each epoch, we will reduce the dimensions to those that have been deemed relevant in previous epochs. If nothing is marked as relevant and then by default, we will pull arms at random, potentially suffering the worst possible reward. At the same time, pulling random arms is increasing our chances of getting relevance feedback.

The second algorithm Algorithm 10), is the version we would recommend using in practice. Here, at each time, with probability $\propto 1/\sqrt{t}$, we pull a random arm and

we pull the arm recommended by OFUL otherwise. All the while, we are running all updates only in the dimensions that have been marked as relevant.

In both algorithms, as time goes on, we expect to get more relevance information. This means that we begin by pulling random arms when we have no relevance information and we grow the dimension of the arms as we get more information over time. We find that in practice, this has a way of self-regularization.

Algorithm 8 OFUL from Abbasi-Yadkori et al. (2011)

for $s = 1, 2, \ldots, M-1$ **do**
  $(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t) = \text{argmax}_{(\mathbf{x},\boldsymbol{\theta}) \in \mathcal{X}_t \times \mathcal{C}_t} \langle \mathbf{x}, \boldsymbol{\theta} \rangle$
  Play $\mathbf{x}_t$ and receive reward y_t
  Update $\mathcal{C}_t$
end for

Algorithm 9 OFUL Epoch Greedy

Let the set of relevant indices, $\mathcal{R}_0 = \{\}$.
for $s = 1, 2, \ldots, M-1$ **do**
  Let $\boldsymbol{X}_s$ be the original feature matrix with only the features in $\mathcal{R}_s$.
  Set $\epsilon_s = 1/\sqrt{2^s}$
  Let the set of relevant indices revealed in this epoch be $\mathcal{I}_s = \{\}$.
  for $t = 1, \ldots, 2^s$ **do**
    With probability ϵ_s, pick an arm uniformly at random
    Or with probability $1 - \epsilon_s$, pick arm according to OFUL restricted on $\boldsymbol{X}_s$
    Update $\mathcal{I}_s = \mathcal{I}_s \bigcup \{$ indices revealed for this arm $\}$
  end for
  update $\mathcal{R}_{s+1} = \mathcal{R}_s \bigcup \mathcal{I}_s$
end for

Algorithm 10 OFUL Continuous Greedy

Let the set of relevant indices, $\mathcal{R}_0 = \{\}$.
for $t = 1, 2, \ldots, T$ **do**
  Let $\boldsymbol{X}_t$ be the original feature matrix with only the features in $\mathcal{R}_{t-1}$.
  Set $\epsilon_t = 1/\sqrt{t}$
  With probability ϵ_t, pick an arm uniformly at random
  Or with probability $1 - \epsilon_t$, pick arm according to OFUL restricted on $\boldsymbol{X}_t$
  Update $\mathcal{I}_t = \mathcal{I}_t \bigcup \{$ indices revealed for this arm $\}$
  update $\mathcal{R}_{t+1} = \mathcal{R}_t \bigcup \mathcal{I}_t$
end for

5.4 Regret Analysis

We restate our previous assumptions that the norm of the arms are upper bounded by L: $\forall \mathbf{x} \in \mathcal{X}, \|\mathbf{x}\| \leq L$. The hidden weight vector $\boldsymbol{\theta}_* \in \mathbf{R}^d$ is also bounded in norm by S: $\|\boldsymbol{\theta}_*\| \leq S$. Therefore it is easy to see that for any arm, the instantaneous regret is bounded using Cauchy-Schwarz:

$$\begin{aligned} |\langle \mathbf{x}^*, \boldsymbol{\theta}_* \rangle - \langle \mathbf{x}, \boldsymbol{\theta}_* \rangle| &\leq |\langle \mathbf{x}^*, \boldsymbol{\theta}_* \rangle| + |\langle \mathbf{x}^*, \boldsymbol{\theta}_* \rangle| \\ &\leq \|\mathbf{x}^*\| \, \|\boldsymbol{\theta}_*\| + \|\mathbf{x}\| \, \|\boldsymbol{\theta}_*\| \\ &\leq 2SL \end{aligned}$$

Theorem 7. *The cumulative regret after T time steps for the Algorithm 9: OFUL Epoch Greedy is:*

$$R_T \leq \frac{8SL}{\log 6M/\delta} \left(\frac{\log 3k/\delta}{\log 1/(1-p)} \right)^2 + \log \frac{T}{2} \left(6SL \sqrt{\frac{T}{4} \log \frac{6M}{\delta}} \right.$$
$$\left. + \, 4\sqrt{\frac{T}{2} k \log(\lambda + nL/k)} \left(\lambda^{1/2} S + R\sqrt{2\log(3M/\delta) + k \log(1 + TL/(2\lambda k))} \right) \right)$$

with probability $\geq 1 - \delta$. i.e. with at least probability $\geq 1 - \delta$, the regret scales like $\tilde{O}(k\sqrt{T})$.

Proof sketch/summary:

- The cumulative regret is summed over all epochs. The epoch size is doubled every time. This ensures that the last epoch dominates the regret.
- For each epoch, we break down the regret into 2 events: all relevant features have been identified up to that epoch or not.
- First, we bound the regret conditioned on the event that all the relevant features have been marked. We then use Hoeffings to get both a lower and upper bound

on the number of times we pull a random arm during an epoch. We cannot say anything about the random part so we use the worst case regret bound but this is fine since ϵ_s is decreasing and it does not dominate the OFUL term.

- Second, we bound the probability that some of the relevant features are not marked so far, which is a constant depending on k and p since it quickly goes to zero after enough epochs are run. We need the random sampling to ensure the probability that some features are not marked so far goes down. We bound the regret in this case with the worst case regret, and it does not depend on T.

Proof. We will have to bound the probability of three different events:

1. The number of times we pull a random arm during an epoch is close to its expectation
2. We have seen all the relevant arms before the current epoch.
3. OFUL regret bound holds

Bounding the number of times we pull a random arm

Lemma 2. *During epoch s, there are $T_s = 2^s$ time steps. Let N_s be the number of random arm pulls during epoch s. Given that the probability of pulling a random arm during epoch s is $\epsilon_s = c/\sqrt{T_s}$, then for any $\delta_1 > 0$:*

$$\mathbb{P}\left(\left|N_s - c\sqrt{T_s}\right| \geq \sqrt{\frac{T_s}{2}\log\frac{2}{\delta_1}}\right) \leq \delta_1$$

Proof. We can see N_s as the sum of T_s i.i.d. Bernoulli random variables with probability of success of ϵ_s. It is easy to see that $\mathbf{E}N_s = T_s \cdot c/\sqrt{T_s} = c\sqrt{T_s}$. Now, let's

apply the Hoeffding's inequality to the sum of the Bernoulli random variables. We get:

$$\mathbb{P}\left(|N_s - \mathbf{E}N_s| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{T_s}(b_i - a_i)^2}\right)$$
$$= 2\exp\left(-\frac{2t^2}{T_s}\right) \text{ , for Bernoulli rvs } b_i = 1 \text{ and } a_i = 0$$
$$\Rightarrow \mathbb{P}\left(\left|N_s - c\sqrt{T_s}\right| \geq \sqrt{\frac{T_s}{2}\log\frac{2}{\delta_1}}\right) \leq \delta_1$$

where the last line was by substituting for the expectation and taking $t = \sqrt{\frac{T_s}{2}\log\frac{2}{\delta_1}}$.

□

Corollary 3. *With probability* $\geq 1 - \delta_1$*:*

$$\sqrt{\frac{T_s}{2}\log\frac{2}{\delta_1}} \leq N_s \leq 3\sqrt{\frac{T_s}{2}\log\frac{2}{\delta_1}}$$

Proof. This is a simple consequence of taking $c = \sqrt{2\log\frac{2}{\delta_1}}$ in Lemma 2. □

Probability of having identified all the relevant arms

Conjecture 1. *Let* $\alpha_0 = \sqrt{2}$ *and* $\alpha_i = \sqrt{\frac{T_s}{2}} = \sqrt{2^{i-1}}$ *for* $i > 0$*. Then:*

$$\sum_{i=0}^{s-1}\alpha_i \geq \sqrt{2^s}$$

Proposition 1. *The number of random arms pulled before an epoch* s *can be bounded*

as:

$$\sqrt{2^s \log \frac{2}{\delta_1}} \leq \sum_{i=0}^{s-1} N_i \leq 3\sqrt{2^s \log \frac{2}{\delta_1}}$$

with probability $\geq 1 - s\delta_1$.

Proof. This is a direct result of Corollary 3 and Conjecture 1. □

Definition 5.4.1. *Let* E_s^j *be a random variable:*

$$E_s^j = \begin{cases} 1 & \textit{if the } j\textit{th index marked as relevant at least once up till epoch } s \\ 0 & \textit{otherwise} \end{cases}$$

Let $E_s = \bigcap_{j=1}^{k} E_s^j$ *be the event that all the relevant features are marked.*

Let us bound $\mathbb{P}(E_s = 0)$, *i.e.,* the probability that all the k relevant features have not been marked upto epoch s:

$$\begin{aligned} \mathbb{P}(E_s = 0) &= \mathbb{P}\left(\bigcup_{j=1}^{k} E_s^j = 0\right) \\ &\leq \sum_{j=1}^{k} \mathbb{P}(E_s^j = 0) \\ &\leq k(1-p)^{\sum_{i=0}^{s-1} N_i} \\ &= k \exp\left(-\log \frac{1}{1-p} \sum_{i=0}^{s-1} N_i\right) \\ &\leq k \exp\left(-\log \frac{1}{1-p} \sqrt{2^s \log \frac{2}{\delta_1}}\right) \leq \delta_2 \text{ , we desire this} \end{aligned}$$

$$\Rightarrow \exp\left(-\log \frac{1}{1-p} \sqrt{2^s \log \frac{2}{\delta_1}}\right) \leq \frac{\delta_2}{k}$$

$$\Rightarrow \log \frac{1}{1-p} \left(\sqrt{2^s \log \frac{2}{\delta_1}}\right) \geq \log \frac{k}{\delta_2}$$

$$\Rightarrow \sqrt{2^s \log \frac{2}{\delta_1}} \geq \frac{\log k/\delta_2}{\log 1/(1-p)}$$
$$\Rightarrow 2^s \geq \frac{1}{\log 2/\delta_1} \left(\frac{\log k/\delta_2}{\log 1/(1-p)} \right)^2$$
$$\Rightarrow s \geq \log_2 \left(\frac{1}{\log 2/\delta_1} \left(\frac{\log k/\delta_2}{\log 1/(1-p)} \right)^2 \right)$$

From the above we get the following proposition:

Proposition 2. *After* $s = \left\lceil \log_2 \left(\frac{1}{\log 2/\delta_1} \left(\frac{\log k/\delta_2}{\log 1/(1-p)} \right)^2 \right) \right\rceil := s_{observed}$ *epochs, we have observed all the relevant features with probability* $\geq 1 - \delta_2$.

Regret after epoch $s_{observed}$

During each epoch after $s_{observed}$, we have at most $3\sqrt{\frac{T_s}{2} \log \frac{2}{\delta_1}}$ random arm pulls and at most $T_s - \sqrt{\frac{T_s}{2} \log \frac{2}{\delta_1}} \leq T_s$ OFUL arm pulls. Therefore we get the following lemma:

Lemma 3. *During each epoch* $s \geq \mathbf{s}_{observed}$*, the regret during the epoch* R_s *is bounded by:*

$$R_s \leq 6SL\sqrt{\frac{T_s}{2} \log \frac{2}{\delta_1}} + 4\sqrt{T_s d \log(\lambda + nL/k)} \left(\lambda^{1/2} S + R\sqrt{2\log(1/\delta_3) + d\log(1 + T_s L/(\lambda k))} \right)$$
$$= O(k\sqrt{T_s})$$

with probability $\geq \delta_3 - s\delta_1$.

Proof. The regret during the epoch is the sum of the regret when we pull the random arms added to the regret when we pull OFUL arms.

Now, we just have to use the upper bound on the number of times we pull a random arm in Corollary 3. During each random arm pull the worst case regret is $2SL$.

The number of times we pull an OFUL arm is trivially T_s. Then apply Theorem 3 from (Abbasi-Yadkori et al., 2011) with $\delta = \delta_3$ to get the result. □

Let us now analyze the regret of ϵ_s-greedy OFUL, which can be summed over the epochs as:

$$\begin{aligned}
R_T &= \sum_{s=0}^{M-1} R_s \\
&= \sum_{s=0}^{s_{observed}} R_s + \sum_{s=s_{observed}+1}^{M-1} R_s \\
&\leq \sum_{s=0}^{s_{observed}} 2SLT_s + \sum_{s=s_{observed}+1}^{M-1} R_s \\
&\leq 2SL2^{s_{observed}+1} + \sum_{s=s_{observed}+1}^{M-1} O(k\sqrt{T_s}) \\
&\leq 2SL2^{s_{observed}+1} + (M-1)O(k\sqrt{T_{M-1}}) \\
&= 2SL2^{s_{observed}+1} + \log \frac{T}{2} O(k\sqrt{\frac{T}{2}}) \\
&\leq \tilde{O}(k\sqrt{T})
\end{aligned}$$

Then, with probability $\geq 1 - (M\delta_1 + \delta 2 + M\delta_3)$. We can set $\delta_1 = \delta_3 = \delta/3M$ and $\delta_2 = \delta/3$ and we get the above result with probability $\geq 1 - \delta$.

Now, note that:

$$\begin{aligned}
2^{s_{observed}+1} &= 2 \cdot 2^{\left\lceil \log_2 \left(\frac{1}{\log 2/\delta_1} \left(\frac{\log k/\delta_2}{\log 1/(1-p)} \right)^2 \right) \right\rceil} \\
&\leq 2 \cdot 2^{\log_2 \left(\frac{1}{\log 2/\delta_1} \left(\frac{\log k/\delta_2}{\log 1/(1-p)} \right)^2 \right) + 1} \\
&= \frac{4}{\log 2/\delta_1} \left(\frac{\log k/\delta_2}{\log 1/(1-p)} \right)^2
\end{aligned}$$

Now, setting $\delta_1 = \delta_3 = \delta/3M$ and $\delta_2 = \delta/3$), we get the final regret expression:

$$R_T \leq \frac{8SL}{\log(6\log T)/\delta}\left(\frac{\log 3k/\delta}{\log 1/(1-p)}\right)^2 + \log\frac{T}{2}\left(6SL\sqrt{\frac{T}{4}\log\frac{6\log T}{\delta}}\right.$$
$$\left. + 4\sqrt{\frac{T}{2}k\log(\lambda + nL/k)}\left(\lambda^{1/2}S + R\sqrt{2\log(3\log T/\delta) + k\log(1 + TL/(2\lambda k))}\right)\right)$$

Thus finishing the proof for Theorem 7.

5.5 Results

5.5.1 Setup for the simuations

Synthetic Dataset

For the synthetic dataset, each document is an arm. To simulate a document we generated 1000 sparse arms in 40 dimensions to represent the fact that each document has a small subset of words. θ^* 5-sparse which is equivalent to saying that for each run, there are 5 relevant words. Since the features are word counts they're always positive.

Newsgroup20 Dateset

The Newsgroup20 dataset, (http://qwone.com/ jason/20Newsgroups/)has 20,000 documents for 20 news groups or topics (but we take only 5 classes in this code since we are replicating the Dasgupta experiments). The 5 topics are misc.forsale, rec.autos, sci.med, comp.graphics, talk.politics.mideast. There are about 4800 documents in these 5 topics. We then use the TF-IDF features for the documents which give us approximately 44000 features.

Note that it is not possible to run OFUL on 44000 dimensional data as that would require storing and updating a 44000×44000 matrix at each iteration. Therefore, we used logistic regression to train a classifier with a high classification accuracy but with sparse support. This led to the selection of 153 features. We then added an additional 847 features at random in order to simulate high dimensional features. This is similar to the way insert Dasgupta reference ran experiments in the classification setting.

In order to get an oracle of relevant arms, we used the support of the one vs many sparse logistic to get "oracle relevant features" for each class. For simulations, the support of θ^* is set of oracle relevant words.

For both settings we looked at intersection of support of arm and oracle relevant

words, and marked each word from the intersection as relevant with probability $p(= 0.1)$.

Dasgupta Dataset

This uses the same documents as the Newsgroup20 dataset. But Poulis et al (Poulis and Dasgupta, 2017) then took 50 of the documents and then had users annotate relevant words in 5 different categories. These are the same as we used in the Newsgroup20 results.

We took those relevant words as the relevance dimensions. This is closer to simulating human feedback since we are not using sparse logistic regression to estimate the sparse vectors. We take the user indicated relevant words instead.

Similar to before, we have one set of experiments where we augmented the 250 relevant words with 250 random words in order to generate a 500-dimensional dataset. We compared OFUL, and our two version of Hybrid OFUL on this data.

We also used the full data without reducing the dimensionality of the data which led to 4868 arms in 47781 dimensions. We only compared our two Hybrid algorithms on this dataset since it would take too many computational resources to run OFUL. This is because, OFUL would require storing and updating a 47781×47781 matrix at every time step.

5.5.2 Reward Model

Synthetic Dataset

Here we have access to $\boldsymbol{\theta}_*$ therefore for any arm $\mathbf{x}$, we use the standard linear model for the reward y_t:

$$y_t = \langle \mathbf{x}_t, \boldsymbol{\theta}_* \rangle + \eta_t$$

where, $\eta_t \sim \mathcal{N}(0, R^2)$ for some $R > 0$.

Newsgroup20 and Dasgupta Dataset

Here, the labels are already given. In order to come close to simulating a noisy setting, we used the logistic model:

$$y_t = \begin{cases} +1 & \text{w. p. } q_t \\ -1 & \text{w. p. } 1 - q_t \end{cases}$$

where, $q_t = 1/(1 - \exp(-\langle \mathbf{x}_t, \boldsymbol{\theta}_* \rangle)$

5.5.3 Parameter Tuning

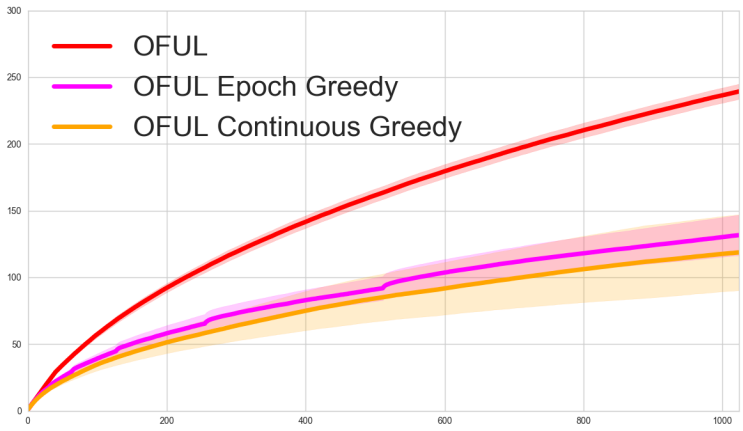
OFUL For OFUL we try different values of the ridge parameter in the range $\{2^{-7}, 2^{-6}, \ldots, 2^10\}$. All the tuned parameters that were selected for OFUL were strictly inside this range. For $d = 40, k = 5$ and $k = 40$, $\lambda_{OFUL} = 2^{-5}$. For $d = 1000$ (Newsgroup), $\lambda_{OFUL} = 2^9$.

OFUL Epoch Greedy and OFUL Continuous Greedy The remarkable feature is that these algorithm need no parameter tuning. We set $\lambda = 1$ for all the experiments.

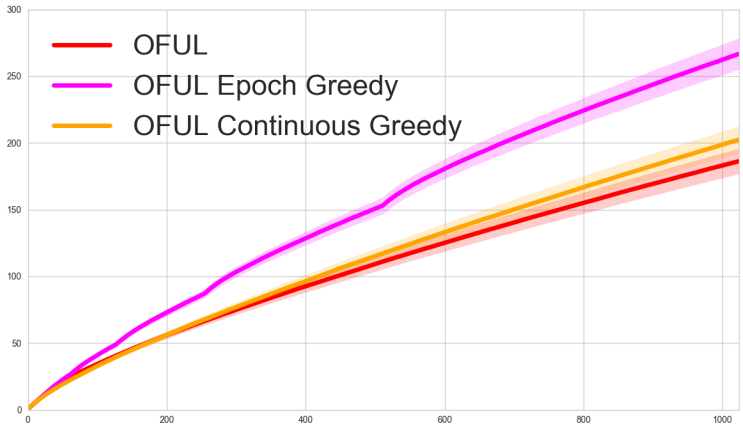
5.5.4 Plots

Simulated data Figure 5.1(a) shows the results of an average of 50 random trials in 40 dimensions where $\boldsymbol{\theta}_*$ is 5-sparse, with 1000 arms.

Figure 5.1(b) shows the results for non-sparse $\boldsymbol{\theta}_*$ in 40 dimensions. Epoch greedy performs worse as expected but the performance of the continuous version is still close to that of the standard OFUL algorithm.



(a) Simulated data with sparse θ_*.



(b) Simulated data with dense θ_*.

Figure 5.1: **(a)** Results on simulated data on non-sparse θ_*. Notice that the continouos greedy OFUL is very close to standard OFUL. **(b)** Results on simulated data on dense θ_*. Notice that the continouos greedy OFUL is very close to standard OFUL.

Newsgroup data Please see the section about generating the Newsgroup data previously. In these simulation averaged over 100 random θ_*, the Continuous Greedy version of OFUL outperforms OFUL significantly. OFUL was tuned by searching for

different values of $\lambda \in \{2^{-4}, 2^{-2}, \ldots, 2^{12}\}$. The best value was chosen which was 2^8.

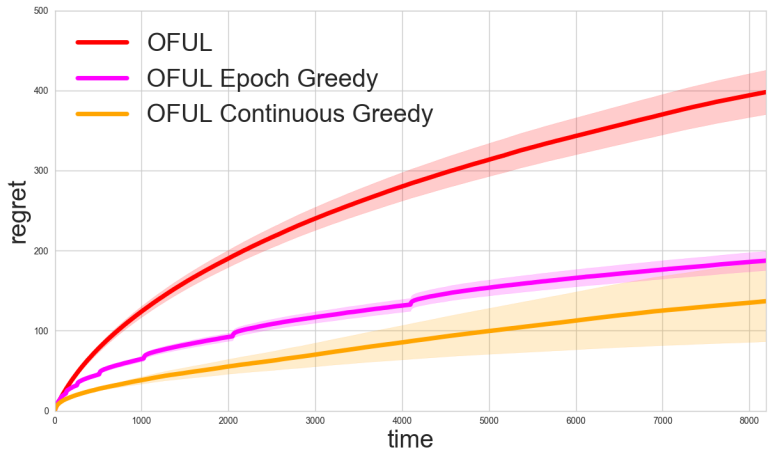


Figure 5.2: This plot compares OFUL with the two algorithms we propose on the Newsgroup dataset. It shows that in practice, we are outperforming OFUL with OFUL Continuous Greedy when running in 1000 dimensions.

Dasgupta data In the first comparison we compared OFUL to our two algorithms. In Figure 5.1, we can see that OFUL Continuous greedy is already outperforming OFUL. This is despite the fact that we are not in a very sparse regime with 250 out of 500 features being relevant.

Surprisingly, we found that tuning had little effect on the performance of OFUL Epoch Greedy and OFUL Continuous Greedy whereas it had a significant effect on OFUL. Figure 5.4 show the difference in performance when the tuning parameter: the ridge regularization weight λ is changed for a small value: 2^{-8} to a much larger vaue 2^6. We believe that this behavior is due to the fact that we gradually grow the number of dimensions that are relevant as we get more feedback therefore implicitly regularizing the number of free parameters in ridge regression and opposed to explicitly regularizing it by tuning λ.

Finally, we also found it remarkable that when we ran our experiments without reducing the dimensions of the features, we barely see a drop off in the regret. It is

also important to note that we did not tune the ridge regression parameters. For all these experiments we set $\lambda = 1$. We notice that both OFUL Epoch Greedy and OFUL Continuous Greedy are robust to changes in the ambient dimensions with respect to the parameter λ.

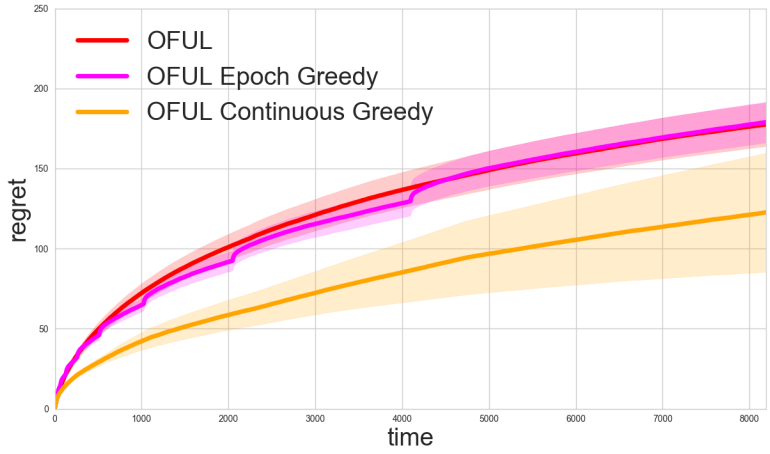
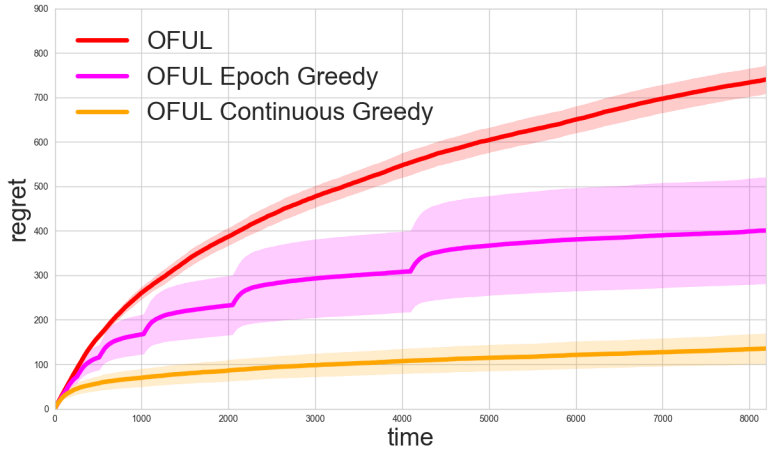
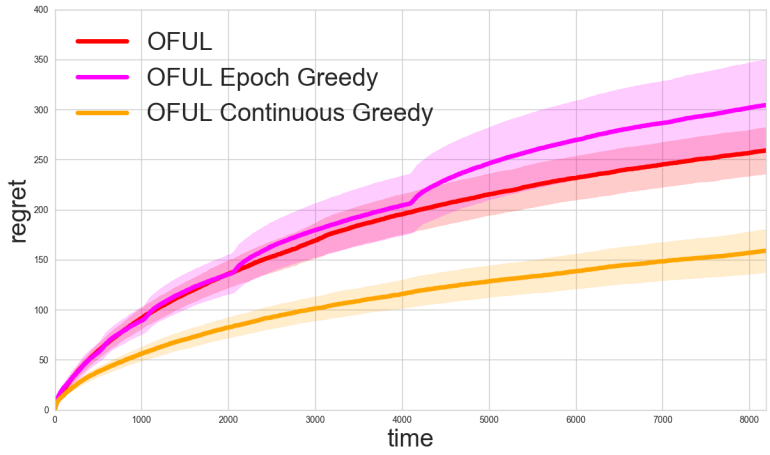


Figure 5.3: This plot compares OFUL with the two algorithms we propose on the Dasgupta data. It shows that in practice, we are already outperforming OFUL with OFUL Continuous Greedy when running in 500 dimensions.



(a) Results on the reduced 500 dimensions of the Dasgupta data, with $\lambda = 2^{-8}$.



(b) Results on the reduced 500 dimensions of the Dasgupta data, with $\lambda = 2^{6}$.

Figure 5.4: Looking at the difference in performace between **(a)** and **(b)** we notice that there is big performance gap of OFUL showing its sensitivity to the tuning paramter λ. In contrast to OFUL, our two proposed algorithms have a relatively modest difference in performace showing their robustness to the ridge regression parameter λ.

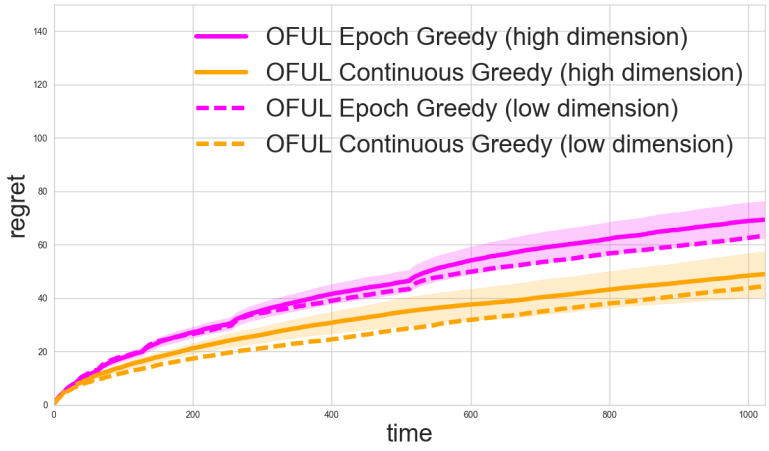


Figure 5.5: This plot compares two algorithms we propose on the Dasgupta data. It shows what happens when we run the algorithms on all the 47781 dimensions (high dimension) compared to the 500 reduced dimensions (low dimension). It shows the performances are very similar when running on orders of magnitude more dimensions. The remarkable fact is that we perform no tuning. We set the ridge parameter to be $\lambda = 1$.

Bibliography

Y. Abbasi-Yadkori, D. Pal, and C. Szepesvari. Improved Algorithms for Linear Stochastic Bandits. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–19, 2011.

Y. Abbasi-Yadkori, D. Pal, and C. Szepesvari. Online-to-Confidence-Set Conversions and Application to Sparse Stochastic Bandits. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

M. Abeille and A. Lazaric. Linear Thompson Sampling Revisited. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pages 176–184, 2017.

S. Agrawal and N. Goyal. Thompson Sampling for Contextual Bandits with Linear Payoffs. *CoRR*, abs/1209.3, 2012.

S. Agrawal and N. Goyal. Thompson Sampling for Contextual Bandits with Linear Payoffs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 127–135, 2013. URL `http://jmlr.org/proceedings/papers/v28/agrawal13.html`.

K. Ahukorala, A. Medlar, K. Ilves, and D. Glowacka. Balancing exploration and exploitation: Empirical parameterization of exploratory search systems. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1703–1706. ACM, 2015.

G. Allard, F. J. Ryan, I. B. Jeffery, and M. J. Claesson. Spingo: a rapid species-classifier for microbial amplicon sequences. *BMC bioinformatics*, 16(1):324, 2015.

J.-Y. Audibert and S. Bubeck. Best arm identification in multi-armed bandits. In *COLT*, 2010.

P. Auer and M. Long. Using Confidence Bounds for Exploitation-Exploration Tradeoffs. *Journal of Machine Learning Research*, 3:2002, 2002.

Y. Belkaid and J. A. Segre. Dialogue between skin microbiota and immunity. *Science*, 346(6212):954–959, 2014.

R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

W. E. Bishop and M. Y. Byron. Deterministic symmetric positive semidefinite matrix completion. In *Advances in Neural Information Processing Systems*, pages 2762–2770, 2014.

S. Bubeck, T. Wang, and N. Viswanathan. Multiple identifications in multi-armed bandits. In *ICML*, 2013.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *FOCM*, 2009.

A. Carpentier and R. Munos. Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In *International Conference on Artificial Intelligence and Statistics*, pages 190–198, 2012.

O. Chapelle and L. Li. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2249–2257, 2011.

O. Chapelle and L. Li. Open Problem: Regret Bounds for Thompson Sampling. In *Proceedings of the Conference on Learning Theory (COLT)*, 2012.

W. Chu, L. Li, L. Reyzin, and R. E. Schapire. Contextual Bandits with Linear Payoff Functions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pages 208–214, 2011.

W. Clavin. Managing the deluge of 'big data' from space, 2013. `http://www.jpl.nasa.gov/news/news.php?release=2013-299`.

K. Crammer and C. Gentile. Multiclass Classification with Bandit Feedback Using Adaptive Regularization. *Mach. Learn.*, 90(3):347–383, 2013.

N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*. ACM, 2008.

V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic Linear Optimization under Bandit Feedback. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 355–366, 2008.

O. Dekel, C. Gentile, and K. Sridharan. Robust selective sampling from single and multiple teachers. In *In Proceedings of the Conference on Learning Theory (COLT)*, 2010.

O. Dekel, C. Gentile, and K. Sridharan. Selective sampling and active learning from single and multiple teachers. *Journal of Machine Learning Research*, 13:2655–2697, 2012.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *SODA*, 2006.

D. Donoho, I. Johnstone, G. Kerkyacharian, and D. Picard. Density estimation by wavelet thresholding. *Ann. Stat.*, 24:508–539, 1996.

P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *JMLR*, 2005.

P. S. Earle, M. Guy, C. Ostrum, S. Horvath, and R. A. Buckmaster. OMG earthquake! Can Twitter improve earthquake response? *Eos Transactions, American Geophysical Union, Fall Meeting Supplement*, 90(52), 2009.

R. Edgar. Sintax: a simple non-bayesian taxonomy classifier for 16s and its sequences. *bioRxiv*, page 074161, 2016.

E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *JMLR*, 2006.

S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári. Parametric Bandits: The Generalized Linear Case. In *Advances in Neural Information Processing Systems (NIPS)*, pages 586–594. 2010.

R. Foygel and N. Srebro. Concentration-based guarantees for low-rank matrix reconstruction. In *COLT*, pages 315–340, 2011.

V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck. Multi-bandit best arm identification. In *NIPS*, 2011.

C. Gentile and F. Orabona. On Multilabel Classification and Ranking with Bandit Feedback. *Journal of Machine Learning Research*, 15:2451–2487, 2014.

A. Gittens and M. Mahoney. Revisiting the nystrom method for improved large-scale machine learning. In *ICML*, pages 567–575, 2013.

F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *WWW*. ACM, 2009.

R. Guo, S. Kumar, K. Choromanski, and D. Simcha. Quantization based Fast Inner Product Search. *Journal of Machine Learning Research*, 41:482–490, 2016. URL `http://arxiv.org/abs/1509.01469`.

M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. Where to buy it: Matching street clothing photos in online shops. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3343–3351, 2015.

S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, 8:321–350, 2012. ISSN 1557-2862. doi: 10.4086/toc.2012.v008a014. URL `http://www.theoryofcomputing.org/articles/v008a014/`.

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

E. Hazan, A. Agarwal, and S. Kale. Logarithmic Regret Algorithms for Online Convex Optimization. *Mach. Learn.*, 69(2-3):169–192, 2007a.

E. Hazan, C. Seshadhri, and S. Jose. Efficient learning algorithms for changing environments. pages 393–400, 2007b.

K. Hofmann, S. Whiteson, and M. de Rijke. Contextual Bandits for Information Retrieval. In *NIPS Workshop on Bayesian Optimization, Experimental Design and Bandits: Theory and Applications*, 2011.

P. Jain, B. Kulis, and K. Grauman. Fast Image Search for Learned Metrics. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. URL `http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4587841`.

P. Jain, S. Vijayanarasimhan, and K. Grauman. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 928–936, 2010.

K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. *COLT*, 2014.

K. Jamieson, L. Jain, C. Fernandez, N. Glattard, and R. Nowak. NEXT: A System for Real-World Development, Evaluation, and Application of Active Learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

H. Jégou, M. Douze, and C. Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 304–317, 2008.

H. Jégou, M. Douze, and C. Schmid. Improving Bag-of-Features for Large Scale Image Search. *International Journal of Computer Vision*, 87(3):316–336, 2010.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014a.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014b.

A. T. Kalai and R. Sastry. The Isotron Algorithm: High-Dimensional Isotonic Regression. In *Proceedings of the Conference on Learning Theory (COLT)*, 2009.

S. Kale, L. Reyzin, and R. E. Schapire. Non-stochastic bandit slate problems. In *NIPS*, 2010.

R. Kannan, S. Vempala, and Others. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3–4):157–288, 2009.

J. Kawale, H. H. Bui, B. Kveton, L. Tran-Thanh, and S. Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *NIPS*, 2015.

R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. In *Advances in Neural Information Processing Systems*, pages 952–960, 2009.

K. Konyushkova and D. Glowacka. Content-based image retrieval with hierarchical Gaussian Process bandits with self-organizing maps. In *21st European Symposium on Artificial Neural Networks*, 2013.

A. Kovashka, D. Parikh, and K. Grauman. Whittlesearch: Interactive image search with relative attribute feedback. *International Journal of Computer Vision*, 115(2): 185–210, 2015.

A. Krishnamurthy and A. Singh. Low-rank matrix and tensor completion via adaptive sampling. In *Advances in Neural Information Processing Systems*, pages 836–844, 2013.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2130–2137, 2009. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-77953184849{&}partnerID=40{&}md5=215f75697066ee3b2e5628faf3db131b`.

B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 31, pages 2143–2157, 2009. ISBN 9781424422432. doi: 10.1109/TPAMI.2009.151.

S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *JMLR*, 2012.

B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *ICML*, pages 767–776, 2015.

M. Laurent and A. Varvitsiotis. A new graph parameter related to bounded rank positive semidefinite matrix completions. *Mathematical Programming*, 145(1-2):291–325, 2014a.

M. Laurent and A. Varvitsiotis. Positive semidefinite matrix completion, universal rigidity and the strong arnold property. *Linear Algebra and its Applications*, 452: 292–317, 2014b.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. *Proceedings of the International Conference on World Wide Web (WWW)*, pages 661—-670, 2010.

L. Li, W. Chu, J. Langford, T. Moon, and X. Wang. An Unbiased Offline Evaluation of Contextual Bandit Algorithms with Generalized Linear Models. In *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*, volume 26, pages 19–36, 2012.

L. Li, Y. Lu, and D. Zhou. Provable Optimal Algorithms for Generalized Linear Contextual Bandits. *CoRR*, abs/1703.0, 2017.

L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

B. L. Maidak, J. R. Cole, T. G. Lilburn, C. T. Parker Jr, P. R. Saxman, J. M. Stredwick, G. M. Garrity, B. Li, G. J. Olsen, S. Pramanik, et al. The rdp (ribosomal database project) continues. *Nucleic acids research*, 28(1):173–174, 2000.

R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug): 2287–2322, 2010.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. London, 1989.

B. Neyshabur and N. Srebro. On Symmetric and Asymmetric LSHs for Inner Product Search. *Proceedings of the International Conference on Machine Learning (ICML)*, 37:1926–1934, 2015. URL `http://jmlr.org/proceedings/papers/v37/neyshabur15.html`.

R. Nussinov and J. A. Papin. How can computation advance microbiome research? *PLOS Computational Biology*, 13(9):e1005547, 2017.

F. Orabona, N. Cesa-Bianchi, and C. Gentile. Beyond Logarithmic Bounds in Online Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 22, pages 823–831, 2012.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

S. Poulis and S. Dasgupta. Learning with feature feedback: from theory to practice. In *Artificial Intelligence and Statistics*, pages 1104–1113, 2017.

L. Qin, S. Chen, and X. Zhu. Contextual Combinatorial Bandit and its Application on Diversified Online Recommendation. In *SDM*, pages 461–469, 2014.

F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML*. ACM, 2008.

Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

P. Rusmevichientong and J. N. Tsitsiklis. Linearly Parameterized Bandits. *Math. Oper. Res.*, 35(2):395–411, 2010.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

B. Settles. Closing the Loop: Fast, Interactive Semi-Supervised Annotation With Queries on Features and Instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1467–1478, Edinburgh, UK, 2011.

A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). *Advances in Neural Information Processing Systems 27*, pages 2321–2329, 2014. ISSN 10495258.

A. Shrivastava and P. Li. Improved Asymmetric Locality Sensitive Hashing (ALSH) for Maximum Inner Product Search (MIPS). In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 812–821, 2015. ISBN 9780000000002. URL `http://arxiv.org/abs/1410.5410`.

M. Simchowitz, K. Jamieson, and B. Recht. Best-of-k bandits. In *COLT*, 2016.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

M. Slaney, Y. Lifshits, and J. He. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012. ISSN 00189219. doi: 10.1109/JPROC.2012.2193849.

M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, 2009.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

M. Tan, I. W. Tsang, L. Wang, B. Vandereycken, and S. J. Pan. Riemannian pursuit for big matrix recovery. In *ICML*, pages 1539–1547, 2014.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

J. A. Tropp. An introduction to matrix concentration inequalities. *arXiv preprint arXiv:1501.01571*, 2015.

M. Valko, R. Munos, B. Kveton, and T. Kocák. Spectral bandits for smooth graph functions. In *International Conference on Machine Learning*, pages 46–54, 2014.

B. Vandereycken. Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

Y. Vardi, L. A. Shepp, and L. Kaufman. A statistical model for positron emission tomography. *Journal of the American Statistical Association*, 80(389):8–37, 1985.

M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, 2008.

Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 2012.

R. Willett and R. Nowak. Multiscale poisson intensity and density estimation. *IEEE Trans. Info. Th.*, 53:3171–3187, 2007.

A. Yu and K. Grauman. Fine-Grained Visual Comparisons with Local Learning. In *Computer Vision and Pattern Recognition (CVPR)*, June 2014.

Y. Yue and C. Guestrin. Linear submodular bandits and their application to diversified retrieval. In *NIPS*, pages 2483–2491, 2011.

Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012a.

Y. Yue, S. A. S. Hong, and C. Guestrin. Hierarchical exploration for accelerating contextual bandits. *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1895–1902, 2012b. URL `http://arxiv.org/abs/1206.6454`.

L. Zhang, T. Yang, R. Jin, Y. Xiao, and Z.-h. Zhou. Online Stochastic Linear Optimization under One-bit Feedback. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 48, pages 392–401, 2016.

Chapter 6

Appendix: Socioscope [1]

[1] Presented at ECML-PKDD 2012, won best student paper in knowledge discovery

6.1 Introduction

Consider a health organization interested in monitoring West Nile virus. An excellent early indicator is the crow – they are infected and turn up dead in people's backyards. The health organization therefore would want to know when, where and how many crows are dead. However, ordinary people tend not to report dead crows (or any other wildlife for that matter) through formal channels, making data collection difficult. Consider a different case of a limnologist interested in monitoring algae blooms in lakes. She wants to know the time, location, and intensity of algae anomalies, but densely instrumenting a large lake is expensive. Or the case of a meteorologist interested in assessing hail damage on the ground. Or a seismologist interested in assessing earthquake damage in near real-time. In all these cases, the quantity of interest can be formulated as a spatio-temporal signal $f(\text{location}, \text{time})$. In all these cases, direct sensing of f is difficult.

Social media such as Twitter and Facebook offers a unique sensing opportunity for such spatio-temporal signal. For instance, users readily discuss their encounter with dead animals on social media: *"I saw a dead crow on its back in the middle of the road. It was a bit SPLAT! I thought maybe it had fallen out of the sky."* A natural idea is to view social media users as distributed sensors for the target phenomenon. One may then estimate f by counting the number of relevant social media posts, as determined by natural language processing techniques. This idea underlines several emerging systems such as earthquake damage monitoring from Twitter Earle et al. (2009).

However, even assuming perfect relevance judgment as we do in this paper, such estimates are still unreliable due to the peculiar characteristics of the human "sensors." For example, given a constant signal f in space, there will be more relevant social media posts where there are more users. The users may indicate the wrong location, or not supply location information at all. The time of post may be delayed by an unknown amount from when the phenomenon happened.

We propose Socioscope, an inference algorithm specifically designed to recovery f from social media data more accurately by addressing these problems. The Socioscope is based on two types of observational data collected from a social media source. The data are counts of the occurrences of specific types of messages/reports, and we will refer to these occurrences as "events." One set of data are "target events" related to specific topics of interest (e.g., dead crows). The other data are "miscellaneous events" that are used to calibrate the number of observed target events relative to the background level of the source. We assume that each observed event is marked with a location and time stamp (although in general these may be unreliable or missing, issues we will deal with in our framework).

Space and time are discretized into bins, and counts of the events occurring in each bin are the data we will work from. Let $S \times T$ denote a set of space-time bins. In the simplest case, the collection of target event counts is denoted by $x_{s,t}, (s,t) \in S \times T$ and the collection of miscellaneous event counts is denoted by $y_{s,t}, (s,t) \in S \times T$. We will use these data to estimate the underlying spatio-temporal distribution of underlying phenomenon that gives rise to the target events. A naive estimate is to compute the

ratios $x_{s,t}/y_{s,t}$, based on the simple reasoning that $x_{s,t}$ should be proportional to the frequency of the phenomenon of interest multiplied by the density of the social media source (e.g., humans). Normalizing $x_{s,t}$ by $y_{s,t}$ accounts for the spatio-temporal variation of the source. There are several problems with the naive estimator that make it quite unreliable. Firstly, the number of counts may be quite small, making the estimator highly variable or *noisy*. Secondly, the location and time stamps may be frequently missing, erroneous or distorted. The naive estimator does not account for such errors. Thirdly, the naive estimator ignores valuable prior information. We may safely assume that the underlying phenomenon varies smoothly in time and/or space (e.g., if it is stormy here and now, it is probably stormy nearby and will be in the near future). The main contribution of our paper is the development of a regularized maximum likelihood procedure that addresses these issues and produces more accurate estimators.

6.2 The Socioscope

The Socioscope is based on the following point-process model. Recall in the simplest case our raw data are counts of target and miscellaneous events over space and time, $\{x_{s,t}\}$ and $\{y_{s,t}\}$, respectively. These counts are modeled as realizations of inhomogeneous spatio-temporal Poisson processes. In order to simplify our notation, we will often organize the count arrays into column vectors denoted by $\mathbf{x}$ and $\mathbf{y}$, and we will write x_i or y_i to refer the the i-th element of the vectors (corresponding to the counts in the i-th space-time bin), but may switch between the two notations when appropriate.

However, social media posts often are temporally delayed and spatially displaced from the target events (e.g., a driver seeing a dead crow may not tweet about it until he arrives at home). Furthermore, many of them do not even contain location information: in our study, only 1% of tweets contain precise GPS coordinates of where the tweet was created; another 31% contain a usable user self-declared location (e.g., "Anchorage, AK") which may be incorrect for many reasons, including the fact that it does not automatically change while they travel or move; the remaining 68% do not contain location meta data at all. It is therefore necessary to extend our representation so that the count arrays $\mathbf{x}$ and $\mathbf{y}$ have $m \geq |S||T|$ bins. We will define $n \stackrel{\text{def}}{=} |S||T|$, the number of true time-location bins, to distinguish this number from m. For example, there could be three kinds of spatial bins: $x^{(1)}_{s,t}$ holds the target event count where s is identified via a reliable means (e.g. GPS), $x^{(2)}_{s,t}$ where s is less reliable (e.g. user self-declared location), and $x^{(3)}_t$ where we know the time but not the location. This produces $m = |2S+1||T|$ bins. Following the analogy to tomography, we call these m bins the "detector bins," while the n bins for $\mathbf{f}$ the "source bins."

We then define an $m \times n$ matrix $\mathbf{Q}$ with non-negative entries, which models transformation from source to detector bins. Specifically, Q_{ij} is the fraction of events that truly happened in the j-th source bin but reported by social media in the i-th detector bin. For example, a large fraction of events produced by the source bin $f_{s,t}$ will produce posts without location information. They will end up in the detector bin $x^{(3)}_t$ (with a slight abuse of notation). This is captured by $Q_{x^{(3)}_t,st}$.

6.2.1 Likelihood Model

We are now ready to define our model. The miscellaneous counts $\mathbf{y}$ are modeled as Poisson distributed as follows. Let $\mathbf{g}$ denote vector of miscellaneous social media source intensities. The intensify of $\mathbf{y}$ is modeled as a transformation of $\mathbf{g}$ with the form $\mathbf{Qg}$. The matrix $\mathbf{Q}$ is assumed to be known, and we will discuss its design and construction below. With this notation, we model $\mathbf{y} \sim \text{Poisson}(\mathbf{Qg})$ or more explicitly the probability of $\mathbf{y}$ given $\mathbf{g}$ is

$$p(\mathbf{y}|\mathbf{g}) \;=\; \sum_{i=1}^{n} \exp(-\sum_j Q_{i,j} g_i) \frac{\left(\sum_j Q_{i,j} g_i\right)^{y_i}}{y_i!} \,.$$

Based on our observations, the expression above can be maximized with respect to $\mathbf{g}$ to obtain the maximum likelihood estimate of $\mathbf{g}$, denoted by $\mathbf{g}_{\text{MLE}}$. The MLE can be efficiently computed using the standard EM algorithm for the Poisson likelihood Vardi et al. (1985). Note that $\mathbf{g}_{\text{MLE}}$ is generally not equal to $\mathbf{y}$ since the model, $\mathbf{Q}$ in particular, accounts for the fact that the space and time stamps may be missing or erroneous. If we were to ignore missing stamps and treat the others as correct, then $\mathbf{Q}$ would be the $n \times n$ identity matrix, and only in this case would the MLE simply be the data $\mathbf{y}$ themselves.

The target counts $\mathbf{x}$ are modeled in a similar fashion. Let $\mathbf{f}$ denote vector of intensity distribution of the underlying phenomenon of interest (e.g., dead crows). The vector $\mathbf{f}$ is the main function of interest. The intensify of $\mathbf{x}$ is modeled as a transformation matrix $\mathbf{P}$ applied to the (element-by-element) product $\mathbf{f} \cdot \mathbf{g}$, since target events occur in proportion to the intensity of the phenomenon and the density of the social media source. The matrix $\mathbf{P}$ is another $m \times n$ matrix with non-negative entries, and has a similar (but not necessarily identical) form and function to $\mathbf{Q}$, above. Also, like $\mathbf{Q}$, $\mathbf{P}$ is assumed to be known. The probability of $\mathbf{x}$ given $\mathbf{f}$ and $\mathbf{g}$ is

$$p(\mathbf{x}|\mathbf{f},\mathbf{g}) \;=\; \sum_{i=1}^{n} \exp(-\sum_{j} P_{i,j} f_i g_i) \frac{\left(\sum_j P_{i,j} f_i g_i\right)^{y_i}}{y_i!} \;.$$

If we let $\mathbf{h} := \mathbf{f} \cdot \mathbf{g}$, then using the standard EM algorithm we can easily obtain the MLE of $\mathbf{h}$. Denote the MLE by $\mathbf{h}_{\text{MLE}}$. Then a simple plug-in estimate for is $\widehat{\mathbf{f}} = \mathbf{h}_{\text{MLE}}/\mathbf{g}_{\text{MLE}}$, assuming that $\mathbf{g}_{\text{MLE}}$ is strictly positive. In this case, $\widehat{\mathbf{f}}$ is the MLE of $\mathbf{f}$. Therefore, we will denote this estimate by $\widehat{\mathbf{f}}_{\text{MLE}}$ in the remainder of the paper. Note that only in the special case where both $\mathbf{Q}$ and $\mathbf{P}$ are identity matrices does this estimate coincide with the naive estimate $x_{s,t}/y_{s,t}$ mentioned in the introduction. In general, $\mathbf{f}_{\text{MLE}}$ is expected to be more accurate since the matrices $\mathbf{Q}$ and $\mathbf{P}$ account for missing data and distortions.

6.2.2 Penalized Likelihood

Next we will consider the possibility of penalizing (regularizing) the maximum likelihood procedure to reflect prior knowledge of the regularities in space and time. The counts at neighboring locations and times tend to be highly correlated. We will add regularization terms to the log-likelihood in order to promote solutions that reflect such correlations. We use a graph-based regularizer. First consider a regularizer for $\mathbf{g}$; the regularizer for $\mathbf{f}$ will be similar. Let $\mathbf{W}$ be an $n \times n$ symmetric and nonnegative weight matrix. The weight w_{ij} is large if i and j correspond to neighboring bins in space and time. The regularizer encourages the corresponding values of g_i and g_j to be similar. This is encoded by the following regularizer:

$$\Omega(\mathbf{g}) = \frac{1}{2} \sum_{i,j} w_{ij} (g_i - g_j)^2.$$

One may define the unnormalized graph Laplacian matrix

$$\mathbf{L}_W = \mathbf{D} - \mathbf{W}$$

where D is the diagonal degree matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$, and compactly define

$$\Omega(\mathbf{g}) = \frac{1}{2}\mathbf{g}^\top \mathbf{L}_W \mathbf{g}.$$

The regularizer for $\mathbf{f}$ is defined in a similar manner, using a (possibly different) weight matrix denoted by $\mathbf{V}$:

$$\Omega(\mathbf{f}) = \frac{1}{2}\mathbf{f}^\top \mathbf{L}_V \mathbf{f}.$$

The joint maximum a posteriori (MAP) estimate of $\mathbf{f}$ and $\mathbf{g}$ is the solution to the following optimization:

$$\max_{\mathbf{f},\mathbf{g}} \left\{ \log p(\mathbf{x}|\mathbf{f},\mathbf{g}) + \log p(\mathbf{y}|\mathbf{g}) - \tau_f \Omega(\mathbf{f}) - \tau_g \Omega(\mathbf{g}) \right\} ,$$

where $\tau_f > 0$ and $\tau_g > 0$ are regularization parameters that may be tuned via cross-validation. The optimization above is non-convex because of the bilinear coupling of $\mathbf{f}$ and $\mathbf{g}$ in $p(\mathbf{x}|\mathbf{f},\mathbf{g})$. Therefore, instead we will first solve the convex optimization

$$\max_{\mathbf{g}} \left\{ \log p(\mathbf{y}|\mathbf{g}) - \tau_g \Omega(\mathbf{g}) \right\} ,$$

and then plug the solution $\widehat{\mathbf{g}}$ into the optimization over $\mathbf{f}$:

$$\max_{\mathbf{f}} \left\{ \log p(\mathbf{x}|\mathbf{f},\widehat{\mathbf{g}}) - \tau_f \Omega(\mathbf{f}) \right\} .$$

The resulting solution is denoted by $\widehat{\mathbf{f}}$. The regularization encourages the spatiotemporal regularity we expect, and therefore with appropriate choices for τ_f and τ_g the estimate $\widehat{\mathbf{f}}$ should be even more accurate than $\mathbf{f}_{\text{MLE}}$.

6.2.3 Theoretical Considerations

The natural measure of signal-to-noise in this problem is the number of counts in each bin. The higher the counts, the more stable and "less noisy" our estimators will be. Indeed, if we directly observe $y_i \sim \text{Poisson}(g_i)$, then the normalized error $\mathbf{E}[(\frac{y_i - g_i}{g_i})^2] = g_i^{-1} \approx y_i^{-1}$. So larger counts, due to larger underlying intensities, lead to small errors on a relative scale. However, the accuracy of our recovery also depends on the regularity of the underlying functions $\mathbf{g}$ and $\mathbf{f}$. If these are very smooth, for example constant functions, then the error would be inversely proportional to the total number of counts, not the number in each individual bin. This is because in the extreme smooth case, $\mathbf{g}$ and $\mathbf{f}$ are determined by a single constant.

To give some insight into dependence of the quality of the estimate on the total number of counts, suppose that $\mathbf{g}$ (or $\mathbf{f}$) is a discretized version of a Hölder α-smooth function. The parameter α is related to the number of bounded and Lipschitz deriva-

tives of a function. Larger values of α correspond to smoother functions. Such a model is reasonable for the application at hand, as discussed in our motivation for regularization above. Furthermore, assume for the sake of this analysis that we directly observe $\mathbf{g}$ according to the model $\mathbf{y} \sim \text{Poisson}(\mathbf{g})$, and let N denote the total count (i.e., $N = \sum_i y_i$). We recall the following result, which follows from the results in Donoho et al. (1996); Willett and Nowak (2007).

Theorem 6.2.1. *Assume that* $\mathbf{g}$ *is a discretized version of a Hölder* α*-smooth* d*-dimensional function* g*. Then there exists a constant* $C_\alpha > 0$ *such that*

$$\inf_{\widehat{\mathbf{g}}} \sup_{g} \mathbf{E}[\|\widehat{\mathbf{g}} - \mathbf{g}\|_1^2] \;\leq\; C_\alpha N^{\frac{-2\alpha}{2\alpha+d}} \; .$$

The error is measured with the 1-norm, rather than two norm, since this is a more appropriate and natural norm in density and intensity estimation. The theorem tells us that no estimator can achieve a faster rate of error decay than the bound above. There exist estimators that nearly achieve this bound (e.g., to within a log factor), and with more work it is possible to show that our regularized estimators, with appropriate regularization parameter settings, could also nearly achieve this rate for $\alpha \leq 2$. For the purposes of this discussion, the lower bound, which certainly applies to our situation, will suffice.

In our case, $d = 2$ (space and time). Consider the situation when $\alpha = 1$, which corresponds to Lipschitz smooth functions, a very mild regularity assumption. Then the bound says that the error is proportional to $N^{-1/2}$. This gives useful insight into the minimal data requirements of our methods. It tells us, for example, that if we want to reduce the error of the estimator by a factor of say 2, then the total number of counts must be increased by a factor of 4. If the smoothness α is very large, then doubling the counts can halve the error. The message is simple. More events and higher counts will provide more accurate estimates.

6.3 Optimization and Tuning

6.3.1 EM Algorithm

The optimization problems posed above are convex, and so in principle any convex solver could be applied to our problem. However, the special structure of the Poisson likelihood function makes the EM algorithm an attractive choice. Here we review the standard EM algorithm for Poisson inverse problems and then show how to adapt it to incorporate the regularization. To keep the presentation simple, we will explain the algorithm for the estimation of $\mathbf{g}$; the procedure is analogous for the recovery of $\mathbf{h} = \mathbf{f} \cdot \mathbf{g}$.

The EM algorithm is based on the notion of an unobserved set of *complete data* defined as follows and follows the classic formulation of Vardi et al. (1985), originally developed for medical imaging problems. We observe the counts of events $\{y_{s,t}\}$ in detector bins, however the location and time stamps may be missing or erroneous. Define the complete data $z_{s,t,s',t'}$ as the number of events that truly occurred in source bin (s', t') but were observed in detector bin (s, t); recall that we have augmented the range of s and t to include special bins for cases where the location and/or time stamp are missing or uncertain. The complete data is not fully observed, but note that according to the definition of $z_{s,t,s',t'}$ the observed data are $y_{s,t} = \sum_{s',t'} z_{s,t,s',t'}$. Also observed that $\sum_{s,t} z_{s,t,s',t'}$ yields the true count at location/time (s', t'), which is ideally what we would like to have for data. Recall that we arranged the observed data into the column vector $\mathbf{y}$ and used the abstract index $i = 1, \ldots, m$ to refer to each of the m (s, t) bin. Following this we will use $\mathbf{Z}$ to denote an $m \times n$ matrix of the complete data counts. The element $z_{j,i}$ is the number of counts in detector bin j that truly originated in source bin i. Recall that $y_j \sim \text{Poisson}(\sum_i Q_{i,j} g_i)$. The distribution of $z_{j,i}$ is related to $\mathbf{g}$ and $\mathbf{Q}$ as follows:

$$z_{j,i} \;\sim\; \text{Poisson}(g_i Q_{j,i}) \; .$$

The EM algorithm uses the expectation of the complete data log-likelihood (the result of the *E-step*) as a surrogate for the observed data log-likelihood. The parameters of interest $\{g_1, \ldots, g_n\}$ are coupled in the observed likelihood, due to the action of the matrix $\mathbf{Q}$. The expected complete data log-likelihood is separable in these parameters and therefore trivial to maximize (the *M-step*). The E and M steps are iterated to convergence to the MLE, which is guaranteed since the observed data log-likelihood, while complicated, is nonetheless convex. The EM algorithm is often initialized with the backprojection $\mathbf{g}^{(0)} := \mathbf{Q}^T \mathbf{y}$ and subsequent iterations $k = 0, 1, \ldots$ follow two simple steps:

E-step Compute the expectation of the complete data log-likelihood under the current iterate $\mathbf{g}^{(k)}$ conditioned on the data

$$Q(\mathbf{g}, \mathbf{g}^{(k)}) \;:=\; \mathbf{E}_{\mathbf{g}^{(k)}} [\log p(\mathbf{Z}|\mathbf{g}) \,|\, \mathbf{y}] \; .$$

Since the complete data log-likelihood is linear in $\mathbf{Z}$, this step reduces to com-

puting $\mathbf{E}_{\mathbf{g}^{(k)}}[\mathbf{Z}|\mathbf{y}]$ and since $\mathbf{Z}|\mathbf{y}$ is multinomial

$$z_{j,i}^{(k)} \;=\; \frac{y_i g_j^{(k)} Q_{j,i}}{\sum_{\ell=0}^{n} g_\ell^{(k)} Q_{j,\ell}} \,.$$

M-step The complete data log-likelihood is separable in $g_i Q_{j,i}$ and so the updated iterate that maximizes it is

$$g_i^{(k+1)} \;=\; \sum_{j=1}^{m} z_{j,i}^{(k)} \,.$$

This algorithm monotonically increases the observed data log-likelihood function at each step and converges to the MLE $\mathbf{g}_{\text{MLE}}$. To obtain the regularized estimate $\widehat{\mathbf{g}}$, we must incorporate the regularization term $\Omega(\mathbf{g})$ and thereby modify the M-step. Instead of maximizing $Q(\mathbf{g}, \mathbf{g}^{(k)})$ with respect to $\mathbf{g}$ in the M-step, we should maximize $Q(\mathbf{g}, \mathbf{g}^{(k)}) \;-\; \tau_g \Omega(\mathbf{g})$. A simple closed-form solution does not exist in this case, but an iterative procedure such as gradient-descent can be used for this maximization. Alternatively, it is not necessary to maximize this criterion; improving it a bit will suffice. Incrementally improving the criterion at each M-step produces what is known as a generalized EM (GEM) algorithm, and it is guaranteed to converge to the global maximum of the regularized likelihood $\log p(\mathbf{y}|\mathbf{g}) - \tau_g \Omega(\mathbf{g})$, again because of the convexity of this function. In all our experiments, we use such a procedure by taking a small step in the direction of the gradient of the function $Q(\mathbf{g}, \mathbf{g}^{(k)}) \;-\; \tau_g \Omega(\mathbf{g})$ at each M-step.

6.3.2 Tuning τ_g and τ_f via Cross Validation

The choice of the regularization parameters has a profound effect on the smoothness of the estimates. It may be possible to select these parameters based on prior knowledge in certain problems, but for our experiments we select these parameters using a cross-validation (CV) procedure, which gives us a fully data-based and objective approach to regularization. We will explain the procedure for the selection of τ_g; a similar procedure can be used for τ_f.

CV is quite simple to implement in the Poisson setting. A hold-out set of data can be constructed by simply subsampling events from the total observation uniformly at random. This produces a partial data set of a subset of the counts that follows precisely the same distribution as the whole set, modulo a decrease in the total intensity per the level of subsampling. The complement of the hold-out set is what remains of the full dataset, and we will call this the training set.

CV is implemented by generating a number of random splits of this type (we can generate as many as we wish), and for each split we run the EM algorithm above on the training set for a range of values of τ_g and for each we obtain an estimate $\widehat{\mathbf{g}}_{\tau_g}$. Then compute the (unregularized) value of the log-likelihood of $\widehat{\mathbf{g}}_{\tau_g}$ on the hold-out set. This provides us with an estimate of the log-likelihood for each setting of τ_g. We simply select the setting that maximizes the estimated log-likelihood.

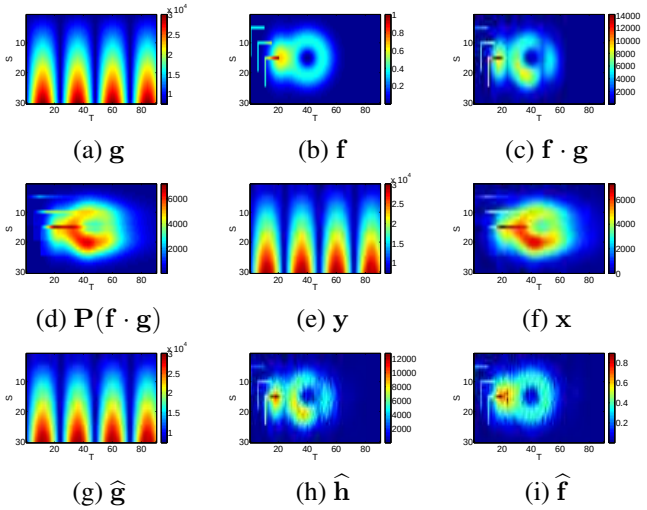
6.4 A Toy Example

To demonstrate the effectiveness of Socioscope, we created a toy example which simulates important aspects of real world social media posts. It has $|S| = 30$ locations and $|T| = 90$ time steps, with a total of $n = 2700$ source bins. Miscellaneous events are typically readily observed. Therefore, we made the magnitude of $\mathbf{g}$ large. For social media, $\mathbf{g}$ has a diurnal periodicity of 24 hours. Furthermore, human population is not uniform in different locations. With these considerations, our $\mathbf{g}$ takes the form in Figure 6.1(a).

We simulated four types of target phenomena with our design of $\mathbf{f}$ in Figure 6.1(b): (i) Bursty events which break out and die down suddenly in a very short time period. They are modeled by the three vertical line segments. (ii) Steady events which last for a long time, and are modeled by the three horizontal line segments. (iii) Spreading events with a gentle onset. They spread to a larger region and their intensities decrease with the distance to the event center. They are modeled by the yellow Gaussian blob. (iv) Recurring events, modeled by the ring. We allowed these events to overlap. Target events are usually sparse in the real world. Thus we assigned small magnitudes to $\mathbf{f}$. The product $\mathbf{f} \cdot \mathbf{g}$ is shown in Figure 6.1(c).

We use the same number $m = n$ of detector bins. We set $\mathbf{Q}$ to be the identity matrix and designed $\mathbf{P}$ to simulate the time delay as follows: A target phenomenon happens at time t. $P_{st',st}$ defines the fraction of social media posts for this phenomenon generated at a later time $t' > t$. $P_{st',st}$ decays exponentially with $t' - t$. There is no spatial spread in $\mathbf{P}$. The Poisson intensities for social media posts are thus $\mathbf{P}(\mathbf{f} \cdot \mathbf{gb})$, which appear as horizontal "smear" in Figure 6.1(d). We generated counts $\mathbf{y}$ and $\mathbf{x}$ with standard Poisson random number generators, see Figure 6.1(e,f).

Given $\mathbf{y}, \mathbf{x}, \mathbf{Q}, \mathbf{P}$, we ran Socioscope to recover $\mathbf{g}$ and $\mathbf{f}$. We regularized temporal and spatial neighbors similar to an Ising model. The resulting $\widehat{\mathbf{g}}$, $\widehat{\mathbf{h}}$, and $\widehat{\mathbf{f}} = \widehat{\mathbf{h}}/\widehat{\mathbf{g}}$ are shown in Figure 6.1(g,h,i), respectively. Note the reasonable recovery of $\mathbf{f}$ by $\widehat{\mathbf{f}}$, which removes the modulation introduced by $\mathbf{g}$ and the smearing introduced by $\mathbf{P}$ in counts $\mathbf{x}$.



(a) $\mathbf{g}$ (b) $\mathbf{f}$ (c) $\mathbf{f} \cdot \mathbf{g}$

(d) $\mathbf{P}(\mathbf{f} \cdot \mathbf{g})$ (e) $\mathbf{y}$ (f) $\mathbf{x}$

(g) $\widehat{\mathbf{g}}$ (h) $\widehat{\mathbf{h}}$ (i) $\widehat{\mathbf{f}}$

Figure 6.1: The toy example

6.5 Case Study: Roadkill

We now turn to a real world task of estimating the spatio-temporal distribution of roadkill for several common animal species, using Twitter posts as observations. Besides being a novel demonstration of the power of Socioscope, the study of roadkill has values in ecology, conservation, and transportation safety.

6.5.1 Data Preparation

We studied the roadkill events within the United States during September 22–28, 2011. Our spatial bins are set as state level and time bins as one hour. Therefore, s is indexed of 50 states plus District of Columbia and t is from 1 to 168. We chose Twitter as our data source because public tweets can be easily collected through its APIs. All tweets include a time stamp, some tweets contain (potentially noisy) spatial information on where they were created.

The Miscellaneous Counts $\mathbf{y}$

To obtain the miscellaneous counts $\mathbf{y}$, we collected tweets through the Twitter stream API using a set of bounding boxes covering the United States. The API supplied a random subsample of all tweets which were created with Geotagging API and from within our specified bounding boxes. Therefore, all these tweets include precise coordinates on where they were created. Through a reverse geocoding database (`http://www.datasciencetoolkit.org`), we mapped the coordinate to a US state. In doing so, we removed the tweets within our bounding boxes but outside the United States. Counting the number of tweets in each bin gave us $\mathbf{y}$.

The Target Counts $\mathbf{x}$

To produce the target event counts $\mathbf{x}$, we need to identify relevant tweets describing roadkill events. First, we collected tweets using a keyword list containing 370 wild animal names. Therefore, the tweets we collected by this way contained at least one animal name on our list. We further filtered the tweets by the phrase "*ran over*." 2,507 tweets passed these two filters during the study period.

However, many tweets surviving the filters did not actually describe roadkill events. We then built a binary text classifier to identify relevant tweets among the surviving ones. Following Settles (2011), the tweets were case-folded but without any stemming or stopword removal. Any user mentions preceded by a "@" were replaced by the anonymized user name "@USERNAME". Any URLs staring with "http" were replaced by the token "HTTPLINK". Hashtags (compound words following "#") were not split and were treated as a single token. Emoticons, such as ":)" or ":D", were also included as tokens. Our feature set included unigrams and bigrams on the resulting tokens. If any unigram or bigram included animal names on our list, we add an addition feature by replacing the animal name with the anonymized token "ANIMAL". For example, we created an extra feature "over ANIMAL" for the bigram "over raccoon". With this feature representation, we trained a linear Support Vector Machine (SVM) classification on 1,450 manually labeled tweets in August 2011 (i.e., the training set is *outside* our study period). The CV accuracy is nearly 90%. We then applied this SVM to classify tweets in our study period. 1,296 tweets among the 2,507 were classified as roadkill tweets. We treat these 1,296 as relevant target tweets.

Because these target tweets were collected by keywords rather than bounding boxes, the nature of the Twitter API means that most do not contain precise location information. Actually, only 1% of them contain coordinates. We processed this 1% by the same reverse geocoding database to map them to a US state s, and place them in the $x_{s,t}^{(1)}$ detection bins as discussed earlier. For the remaining tweets, some contained a self-declared "user location" in the user profiles. However, user location is a free text field and the users can fill in anything they want. We created several rules to extract US state names from this field. At the end, we were able to map 31% of target tweets in this way to a US state. These are placed in the $x_{s,t}^{(2)}$ detection bins. The remaining 68% contained no location meta data, and may come from the United States or other countries. We were able to estimate a time-dependent fraction of tweets originated in the US. These counts were placed in the $x_t^{(3)}$ detection bins.

The Transformation Matrices $\mathbf{Q}, \mathbf{P}$

For the roadkill study, we did not consider temporal delay or spatial diffusion. Since both temporal and spatial meta data were available for miscellaneous events, we simply take $\mathbf{Q}$ to be an identical matrix. As discussed above, there are three types of target event detector bins: these with precise coordinates $\mathbf{x}^{(1)}$, these with potentially distorted user-declared locations $\mathbf{x}^{(2)}$, and these without location information $\mathbf{x}^{(3)}$. We construct $\mathbf{P}$ to handle the distorted and missing location counts in detector bins $x_{s,t}^{(2)}$ and $x_t^{(3)}$. The matrix takes the following form:

- $\mathbf{P}_{x^{(1)}_{st},st} = 0.01$, and $\mathbf{P}_{x^{(1)}_{sit},st} = 0$ for $s' \neq s$ to reflect the fact that we know precisely 1% of the target tweets' location.
- $\mathbf{P}_{x^{(2)}_{s't},st} = 0.31 M_{s's}$ for all s', s, where M is a 51×51 "mis-identification" matrix. $M_{s's}$ is the probability that a user self-declare that they are in state s', but the coordinates of the tweet they sent show that they are in fact in state s. We were able to estimate M from a separate procedure on tweets. The M matrix itself contains interesting observations:
  - Many people who self-declare California or New York are actually producing tweets from all over the country, perhaps due to traveling.
  - Many people who self-declare District of Columbia are actually producing tweets from Maryland or Virgina.
  - M is asymmetric. For example, we found that more people who self-declare Wisconsin are actually in Illinois, than people who declare Illinois but are actually in Wisconsin.
- $\mathbf{P}_{x^{(3)}_t,st} = 0.68$. This aggregates tweets with missing information into the third kind of detector bins.

6.5.2 Results

One challenge we faced was the extreme sparse nature of roadkill events, especially when broken down into individual animal species (the 1,296 target tweets were the sum over all species). Often, a species had as little as 30 target tweets during the study period. This sparsity forced us to further coarsen our bins in order to obtain interpretable results. Specifically, we merged the states into five US regions: northwest, southwest, midwest, southeast, and northeast. We also folded the 7 days into 24 bins corresponding to the 24 hours of a day. This results in $|S| = 5, |T| = 24, n = |S||T| = 120, m = (2|S| + 1)|T| = 264$. Note the target counts are still sparse even in this coarser setting, thus signal recovery remains a challenge.

We ran Socioscope on the roadkill data. We used a linear graph regularizer on time for both $\mathbf{g}$ and $\mathbf{f}$. The graph connect adjacent hours. The regularizer weights $\tau_{\mathbf{g}}$ and $\tau_{\mathbf{f}}$ are tuned with CV. Figure 6.2 shows the estimate $\widehat{\mathbf{g}}$. There is a clear diurnal pattern. More interestingly, Figure 6.3 shows the estimate for four animal species. Each animal has three plots. From the top: (i) The recovered $\widehat{\mathbf{h}}$; (ii) The recovered $\widehat{\mathbf{f}}$; (iii) The naive estimate $\mathbf{x}/\mathbf{y}$. In each plot, the x-axis is the hour, and the y-axis is the five US regions, from top: northwest, southwest, midwest, southeast, and northeast in that order.

We observe that the naive estimate $\mathbf{x}/\mathbf{y}$ is clearly broken (too sparse). Our estimate $\widehat{\mathbf{f}}$ shows that chipmunks and turtles are diurnal animals, and raccoons and frogs are nocturnal, as expected. In addition, chipmunks mostly become victims in the northeast, raccoons mostly in the midwest, while turtles mostly in the southeast, also as expected.

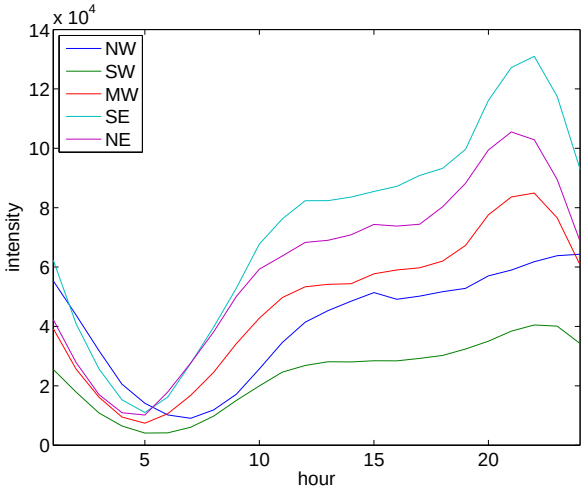


Figure 6.2: The background intensity in five US regions. The time is Central Daylight Saving Time.

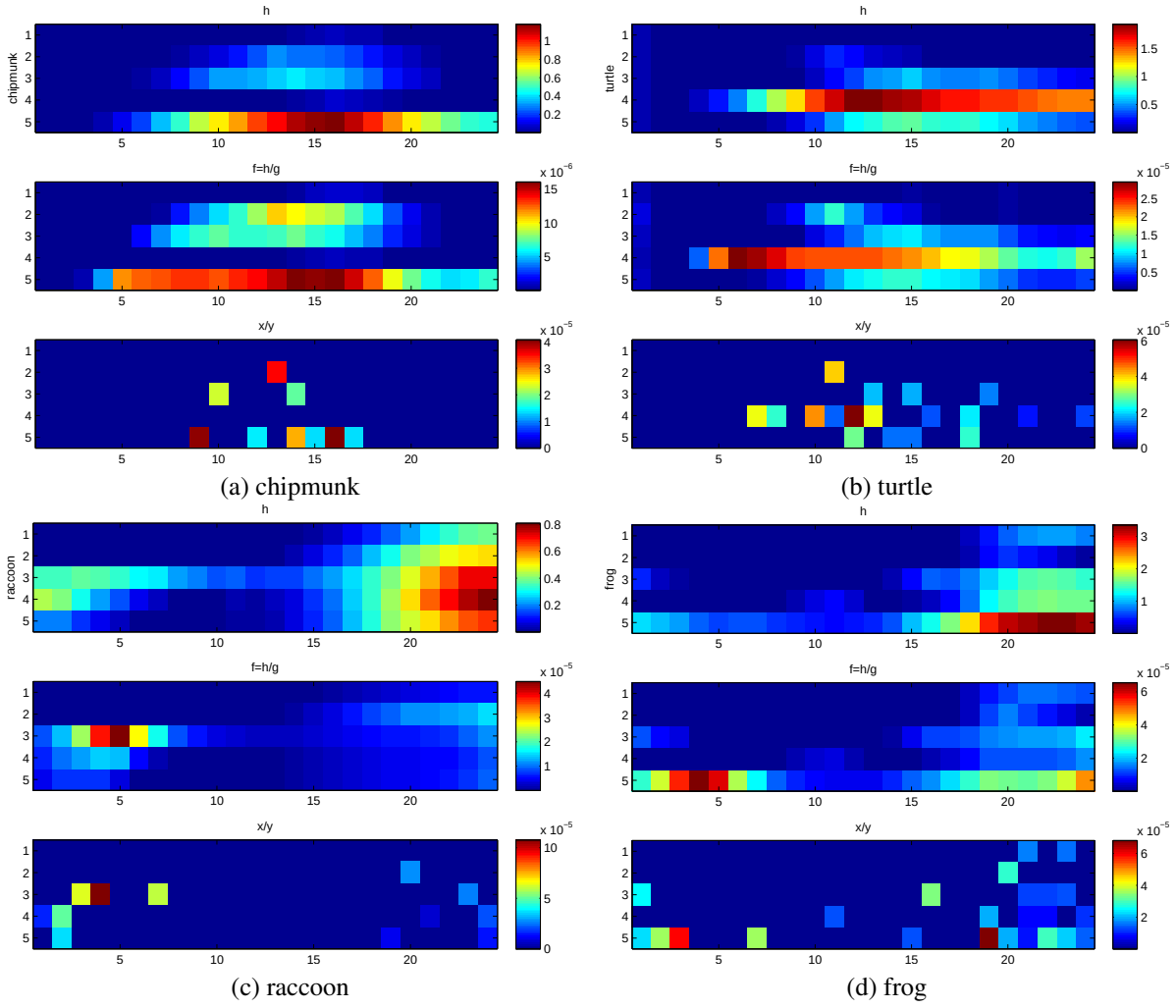


Figure 6.3: Roadkill intensity for four animal species.

Chapter 7

Appendix: NEXT Contributions

7.1 Introduction

NEXT (Jamieson et al., 2015) is a system where we can deploy and manage active learning algorithms. The system allows the running of multiple algorithms and an API that allows us to deploy these experiments on the web. This is a great platform to test algorithms mentioned in this thesis.

The summary of contributions to NEXT are:

- The creation and coding of a contextual bandit algorithm with option to show the users images corresponding to context.
- The following algorithms are available to the user:
    1. OFUL from (Abbasi-Yadkori et al., 2011)
    2. Linear Thompson Sampling from (Agrawal and Goyal, 2013)
    3. Q-OFUL from Chapter 3
    4. GLOC from Chapter 4
    5. ϵ-greedy from (Sutton and Barto, 1998)
    6. Random Sampling
- Creating on interfaces for both showing one image at a time or multiple images at a time.

For the details please see the BioImageSearch application in NEXT on the following page: `https://github.com/BhargavaA/NEXT`.

7.2 Usage

In this chapter, you will find documentation on how to use the BioImageSearch application. It will show where and how to specify the context, and other parameters to correctly run this code.

7.2.1 Required Packages

In order to correctly run this application of NEXT, you will need to install the following packages: numpy, cvxpy, and choldate (if needed to speed up Thompson Sampling).

In order to run the files locally, you will need the SimpleHTTPServer package to host images, and Docker (test only on a Mac) to run the docker containers for NEXT.

These are in addition to the requirements for NEXT which are described in `https://github.com/nextml/next/`. To run NEXT locally, please read the documentation carefully in the above link. For here on, it will be assumed that you are able to locally run a version of NEXT.

7.2.2 Creating the experiment dictionary

NEXT requires sending the system a dictionary at launch that contains all the parameters and context required in running the experiment. The elements of this dictionary are:

1. 'app_id': The ID of the application that is needed to be run. Here it is 'BioImageSearch'.
2. 'args': A dictionary with the following keys:
    - 'failure_probability': The δ parameter such that regret bounds hold with probability $\geq 1 - \delta$.
    - 'participant_to_algorithm_management': for us, this will be 'one_to_one' so that each person is tied to one algorithm.
    - 'algorithm_management_settings': see NEXT algorithm management settings
    - 'alg_list': to be described later
    - 'num_tries': number of questions to ask each user. e.g. 50
    - 'd': The ambient dimension d for the context
    - 'n': The number of items/arms n
    - 'targets': a dictionary of targets to be described later

The application requires the creation of a dictionary of algorithms to be used. These will also contain some algorithm specific parameters that are described here. 'alg_list' is a list of dictionaries. Where there is one dictionary for each algorithm in to be used. Each of these dictionaries requires the following keys:

1. 'alg_id': The name of the algorithm in the system. e.g. 'OFUL'

2. 'alg_label': The label for the algorithm e.g. 'OFUL from Abbasi-Yadkori et al'
3. 'params': A dictionary of all parameters needed for algorithm. Note, this is an important part of specification. Please specify a dictionary of paramters here. For example, OFUL needs the ridge parameter: λ, bound on the norm of $\boldsymbol{\theta}_*$: S, the sub-Gaussian parameter of the noise R, and bound on the norm of the arms: L. Therefore for OFUL, params will be the following dictionary:
    - 'ridge': λ,
    - 'R': R,
    - 'S': S.
    - 'L': L

This application requires the inclusion of a dictionary where every element itself is a dictionary where each element need the following keys:

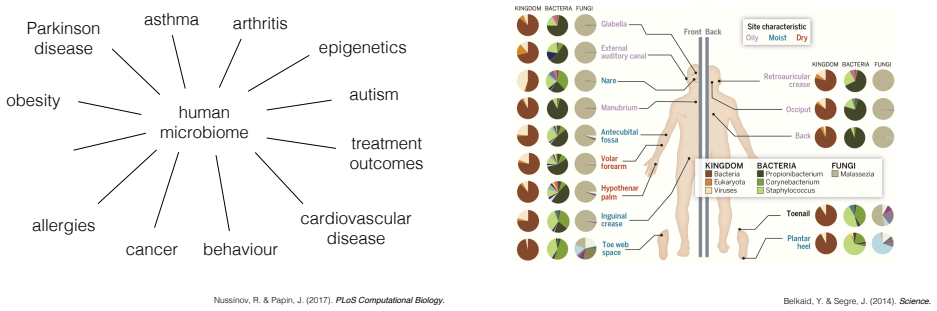
1. 'target_id': a number indication the number of the target.
2. 'primary_type': this should be set to 'image'
3. 'primary_description': url to where the image is hosted e.g. `http://localhost:8999/Image1.jpg`
4. 'context': a list of length d, representing the features of the target
5. 'alt_type': this should be typically set to 'text'.
6. 'alt_description': any title for the image can be added here e.g. 'Sunset image'

These dictionaries are specified in the file: `https://github.com/BhargavaA/NEXT/blob/master/examples/launch_bio_image_search.py`.

Chapter 8

Appendix: IDTaxa - Taxonomy Classification [1]

[1] In preparation for a journal submission



(a) Potential areas affected by human microbiome (b) Relation between skin microbiome and oily/dry skin

Figure 8.1: Examples of how the microbiome may affect human health.

8.1 Introduction

The area of studying the microbiome is has been been very popular recently. There has been evidence to suggest that it plays a large role in human health. It may have an impact from areas like cancer, to Parkinson's disease (Nussinov and Papin, 2017). It has also been shown that it can have an influence on whether our skin is oily or dry (Belkaid and Segre, 2014). These two examples are shown in Figure 8.1.

There has been a lot of work in classifying the microbiome. The popular methods that we will compare to are the popular RPD classifier (Maidak et al., 2000), and the more recent algorithms SINTAX (Edgar, 2016) and SPINGO (Allard et al., 2015).

The workflow of a typical classification task is shown in Figure 8.2. The key to note here is that while there are three broad ways of classifying that is used in practice today, we are interested in the taxonomic classification. This is due to the fact that we want the name of the organism that we are trying to classify. These names have been given by experts over the years and it remains an important way of studying biological organisms.

A more detailed explanation for our approach is that we have a dataset of previously classified sequences. We then generate a profile for each of these sequences by taking k-mers from them. These are overlapping length k sequences. This leads to a bag-of-words feature for every sample in the dataset. We then need to match a new sequence (which may only be partial length) to one of the sequences in the dataset. We would also like to reject a sequence if we are not confident in any match, and declare that such a sequence does not exist in our dataset.

There are many challenges in practice. For one, the length of the sample sequences vary in length. This means that ideal algorithms like aligning the sequences and then computing a Hellinger distance, or Hamming distance is not feasible in practice. Besides, in practice we are tasked with classifying millions of sequences at a time. The poses time and complexity constraints on the potential algorithms. The datasets are also biased towards sequences that are usually studied. Therefore, some branches of

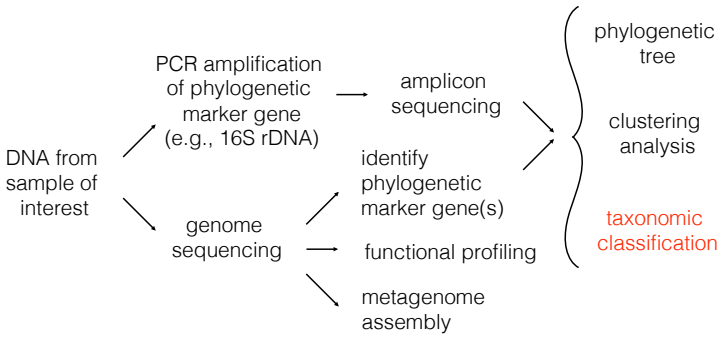


Figure 8.2: A typical workflow for classifying DNA sample. Our work focuses on the taxonomic classification where we are interested in getting the name of an organism.

the taxonomy contain orders of magnitude more sequences than other branches.

8.2 Method

8.2.1 Defining the problem

We are given a dataset of sequences $\mathcal{S}$. Each sequence $\mathbf{s} \in \mathcal{S}$ is comprised of the letters $\{A, C, T, G\}$. The length of each sequence is variable. Each sequence also has a series of taxonomic labels, $l = \{\text{phylum: protobacteria}, \text{class: } \gamma \ , \ldots , \text{species: E. Coli}\}$. Given a query sequence $\mathbf{q}$, our goal is to either find a match to the dataset $\mathcal{S}$, or to reject the sequence and declare it not to exist in the dataset.

In practice, we need to define a measure of errors. We cannot do this for totally new and unknown sequences. But suppose we do leave-one-out testing. Where we remove a sequence from the dataset during training. Then we know the ground truth. There are sequences which are the sole representatives of their labels. When we remove them, the classifier should reject that sequence and declare that it cannot be classified. This leads to the definition of the following two types of errors.

Definition 4. *Suppose the query sequence $\mathbf{q} \in \mathcal{S}$ but $\mathbf{q}$ is not a singleton sequence i.e. there are other examples in the dataset that have the same label. Then if the error of incorrectly predicting its label will be called a* ***misclassification error****.*

Suppose the query sequence $\mathbf{q} \in \mathcal{S}$ was a singleton sequence. If the classifier did not reject the sequence, then it will be called an ***overclassification error****.*

Our observation, which will be demonstrated in the results, is that most classifiers preform well when it comes to misclassifying sequences but then they are overconfident and classify singleton sequences leading to overclassification errors.

8.2.2 Reformulating matches

We found that the most common problem with current classifiers was that the classifiers would weight all the words equally. i.e. some sequences that may occur in all organisms are given equal weight to those sequences that may be crucial in identifying that sequences.

Besides, current algorithms bootstrap by random subsampling the words in the bag-of-words model. Then they generate a winner sequence that matches best to the subsampled version and repeat this process over and over again to generate a confidence. Let us assume that we ran 100 such random iterations. We do not distinguish between a winner who had 99/100 matches to one which had a 11/100 matches. We therefore propose to correct this by accounting probabilistically for how many matches we expect compared to a random match in order to get a better estimate for confidences.

In order to correct for meaningful words, we propose to use the standar TF-IDF technique to downweight sequences that may occur regularly and upweight sequences that are more meaningful for classification. This is a start method in bag-of-words models.

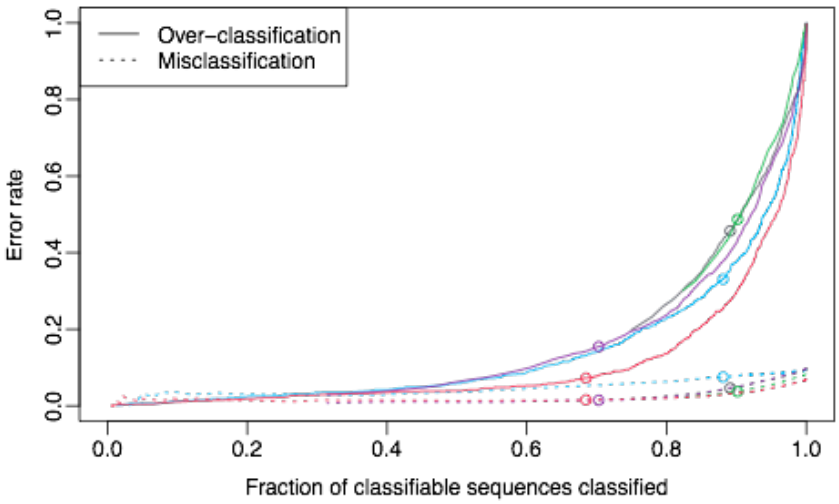


Figure 8.3: Comparison of various algorithms on the RDP dataset. The solid lines show the over-classification rates whereas the dashes lines show the misclassification rates.

8.3 Results

We compared various algorithms on both the RDP data (Maidak et al., 2000) and the ConTax data `https://rdrr.io/cran/microcontax/`. The algorithms are compared on the misclassification and over-classification rates.

Figures 8.3 and 8.4 show the comparison on the RDP anc ConTax datasets respectively. The biggest benefit of using our algorithm (IDTaxa) is to reduce the over-classification rate. There are slight improvements in misclassification rate too.her pt

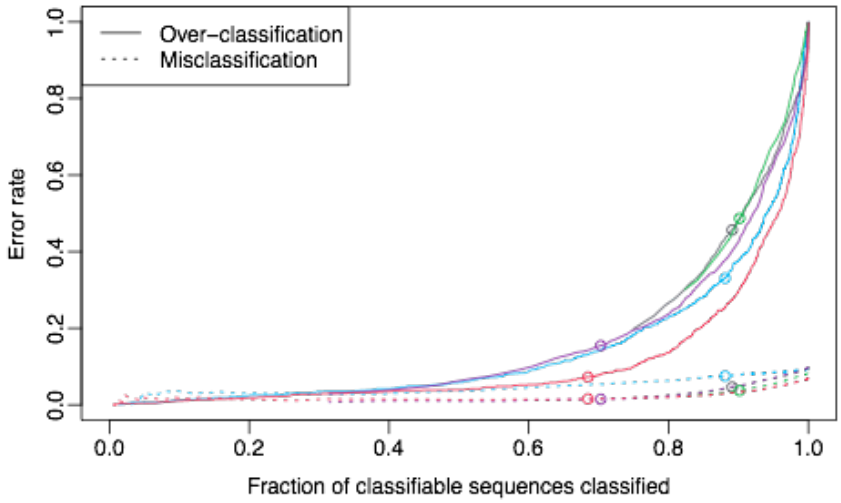


Figure 8.4: Comparison of various algorithms on the ConTax dataset. The solid lines show the over-classification rates whereas the dashes lines show the misclassification rates.