UNIVERSITY OF INSUBRIA

DiSTA

Department of Theoretical and Applied Sciences

PhD Dissertation

to obtain the degree of

Doctor of Philosophy in Computer Science and Computational Mathematics

Defended by

Shuai LI

The Art of Clustering Bandits

Advisor: Claudio GENTILE

Cycle XXIX, 2016

To my wonderful parents and sister:

words cannot describe how lucky I am to have you in my life. I would especially like to thank you for your love, support, and constant encouragement I have gotten over the years. Your love, laughter and music have kept me smiling and inspired.

Acknowledgment

First I would like to express my deepest gratitude to my PhD adviser Claudio Gentile, for his patience, motivation, and immense knowledge; for the autonomy he let me and for his trust.

I want to thank Alexandros Karatzoglou for his continuous support as well as for his friendship.

A big thank also goes to Róbert Busa-Fekete who accepted to review this thesis.

Last but not least, I would love to thank the University of Cambridge.

Abstract

Multi-armed bandit problems are receiving a great deal of attention because they adequately formalize the exploration-exploitation trade-offs arising in several industrially relevant applications, such as online advertisement and, more generally, recommendation systems. In many cases, however, these applications have a strong social component, whose integration in the bandit algorithms could lead to a dramatic performance increase. For instance, we may want to serve content to a group of users by taking advantage of an underlying network of social relationships among them. The purpose of this thesis is to introduce novel and principled algorithmic approaches to the solution of such networked bandit problems. Starting from a global (Laplacian-based) strategy which allocates a bandit algorithm to each network node (user), and allows it to "share" signals (contexts and payoffs) with the neghboring nodes, our goal is to derive and experimentally test more scalable approaches based on different ways of clustering the graph nodes. More importantly, we shall investigate the case when the graph structure is not given ahead of time, and has to be inferred based on past user behavior. A general difficulty arising in such practical scenarios is that data sequences are typically nonstationary, implying that traditional statistical inference methods should be used cautiously, possibly replacing them with by more robust nonstochastic (e.g., game-theoretic) inference methods.

In this thesis, we will firstly introduce the centralized clustering bandits. Then, we propose the corresponding solution in decentralized scenario. After that, we explain the generic collaborative clustering bandits. Finally, we extend and showcase the state-of-the-art clustering bandits that we developed in the quantification problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective

The ability of a website to present personalized content recommendations is playing an increasingly key role in achieving user satisfaction. Due to the occurrence of new content, as well as to the ever-changing nature of content popularity, modern approaches to content recommendation are strongly adaptive, and attempt to match as closely as possible users' interests by repeatedly learning good mappings between users and contents. These mappings are based on context information (i.e., sets of features) which are typically extracted from both users and contents. The need to focus on content that raises users' interest, combined with the need of exploring new content so as to globally improve users' experience, generates a well-known exploration-exploitation dilemma, which is commonly formalized as a multi-armed bandit problem. In such scenarios, contextual bandit algorithms have rapidly become a reference technique for implementing adaptive recommender systems. Yet, in many cases, the users targeted by such systems form a social network, whose structure may provide valuable information regarding user interest affinities. Being able to exploit such affinities can lead to a dramatic increase in the quality of recommendations.

The starting point of our investigation is to leverage user similarities represented as a graph, and running an instance of a contextual bandit algorithm at each graph node. These instances are allowed to interact during the learning process, sharing contexts and user feedbacks. Under the modeling assumption that user similarities are properly reflected by the graph structure, interactions allow to effectively speed up the learning process that takes place at each node. This mechanism is implemented by running instances of a linear contextual bandit algorithm in a specific Reproducing Kernel Hilbert Space (RKHS). The underlying kernel, previously used for solving online multitask classification problems, is defined in terms of the Laplacian matrix of the graph. The Laplacian matrix provides the information we rely upon to share user feedbacks from one node to the others, according to the network structure. Since the Laplacian kernel is linear, the implementation in kernel space is conceptually straightforward. Moreover, the existing performance guarantees for the specific bandit algorithm we use can be directly lifted to the RKHS, and expressed in terms of spectral properties of the user network.

Despite its crispness, the principled approach described above has two drawbacks hindering its practical usage. First, running a network of linear contextual bandit algorithms with a Laplacian-based feedback sharing mechanism may cause significant scaling problems, even on small to medium-sized social networks. Second, it is common wisdom in recommender system research that the social information provided by the network structure at hand need not be fully reliable in accounting for user behavior similarities. Given the above state of affairs, we shall consider methods that reduce "graph noise" by either removing edges in the network of users and/or cluster the users (so as to reduce the graph size). We expect both these two methods to achieve dramatic scalability improvements, but also to have increased prediction performance under different market share conditions, the edge removal strategy being more effective in the presence of many niche products, the clustering strategy being more effective in the presence of few hit products. More importantly, we shall consider methods where the graph information is inferred adaptively from past user behavior. In this case, unlike many traditional methods of user similarity modeling and prediction, we are not relying on low rank factorization assumptions of the user-product matrix (which would again be computationally prohibitive even on mid-sized networks of users), but rather on clusterabilty assumptions of the users, the number of clusters setting the domain-dependent trade-off between hits and niches. In this scenario, we shall develop robust online learning methods which can suitably deal with the nonstationarity of real data, e.g., due to a drift in user interests and/or social behavior.

Last but not least, a great deal of effort within this thesis will be devoted to carrying out careful experimental investigations on real-world datasets of various sizes, so as to compare our algorithms to state-of-the-art methods that do not leverage the graph information. Comparison will be in terms of both scalability properties (running time and space requirements) and prediction performance. In our comparison, we shall also consider different methods for sharing contextual and feedback information in a set of users, such as feature hashing techniques.

In short, we are aimed at:

- Developing algorithmic approaches to reducing the graph size in a social network (by either removing edges or clustering nodes) so as to retain as much information as possible on the underlying users and, at the same time, obtain a dramatic reduction in the running time and storage requirements of the involved graph-based contextual bandit algorithms;
- Developing scalable and principled algorithmic approaches to inferring the graph structure from past user behavior based on clusterability assumptions over the set of users;
- Carrying out a careful experimental comparison of the above methods on small, medium and large datasets with state-of-the-art contextual bandit

methods that do not exploit the network information, as well as to different methods for sharing contextual and feedback information, such as feature hashing techniques.

In all cases, our algorithms will be online learning algorithms designed to operate on nonstationary data sequences.

1.2 Main Contributions

This thesis summarizes the major findings refer to chapter 2 to 5 correspondingly:

- We introduce a novel algorithmic approach to content recommendation based on adaptive clustering of exploration-exploitation ("bandit") strategies. We provide a sharp regret analysis of this algorithm in a standard stochastic noise setting, demonstrate its scalability properties, and prove its effectiveness on a number of artificial and real-world datasets. Our experiments show a significant increase in prediction performance over state-of-the-art methods for bandit problems.
- We provide two distributed confidence ball algorithms for solving linear bandit problems in peer to peer networks with limited communication capabilities. For the first, we assume that all the peers are solving the same linear bandit problem, and prove that our algorithm achieves the optimal asymptotic regret rate of any centralised algorithm that can instantly communicate information between the peers. For the second, we assume that there are clusters of peers solving the same bandit problem within each cluster, and we prove that our algorithm discovers these clusters, while achieving the optimal asymptotic regret rate within each one. Through experiments on several real-world datasets, we demonstrate the performance of proposed algorithms compared to the state-of-the-art.
- Classical collaborative filtering, and content-based filtering methods try to learn a static recommendation model given training data. These approaches are far from ideal in highly dynamic recommendation domains such as news recommendation and computational advertisement, where the set of items and users is very fluid. In this work, we investigate an adaptive clustering technique for content recommendation based on exploration-exploitation strategies in contextual multi-armed bandit settings. Our algorithm takes into account the collaborative effects that arise due to the interaction of the users with the items, by dynamically grouping users based on the items under consideration and, at the same time, grouping items based on the similarity of the clusterings induced over the users. The resulting algorithm thus takes advantage of preference patterns in the data in a way akin to collaborative filtering methods. We provide an empirical analysis on medium-size real-world datasets, showing scalability and increased prediction performance (as measured by click-through rate) over state-of-the-art methods for clustering

bandits. We also provide a regret analysis within a standard linear stochastic noise setting.

- The estimation of class prevalence, i.e., the fraction of a population that belongs to a certain class, is a very useful tool in data analytics and learning, and finds applications in many domains such as sentiment analysis, epidemiology, etc. For example, in sentiment analysis, the objective is often not to estimate whether a specific text conveys a positive or a negative sentiment, but rather estimate the overall distribution of positive and negative sentiments during an event window. A popular way of performing the above task, often dubbed *quantification*, is to use supervised learning to train a prevalence estimator from labeled data.

  Contemporary literature cites several performance measures used to measure the success of such prevalence estimators. In this work we propose the first online stochastic algorithms for *directly* optimizing these quantification-specific performance measures. We also provide algorithms that optimize *hybrid* performance measures that seek to balance quantification and classification performance. Our algorithms present a significant advancement in the theory of multivariate optimization and we show, by a rigorous theoretical analysis, that they exhibit optimal convergence. We also report extensive experiments on benchmark and real data sets which demonstrate that our methods significantly outperform existing optimization techniques used for these performance measures.

1.3 List of Publications

- "Collaborative Filtering Bandits", Shuai Li, Alexandros Karatzoglou, and Claudio Gentile, The 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Acceptance Rate: 18%, (SIGIR 2016)
- "Distributed Clustering of Linear Bandits in Peer to Peer Networks", Nathan Korda, Balázs Szörényi, and Shuai Li, The 33rd International Conference on Machine Learning, Journal of Machine Learning Research, New York, USA, Acceptance Rate: 24%, (ICML 2016)
- "Online Optimization Methods for the Quantification Problem", Purushottam Kar, Shuai Li, Harikrishna Narasimhan, Sanjay Chawla and Fabrizio Sebastiani, The 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, Acceptance Rate: 18%, (SIGKDD 2016)
- "Mining λ-Maximal Cliques from a Fuzzy Graph", Fei Hao, Doo-Soon Park, Shuai Li, and HwaMin Lee, Journal of Advanced IT based Future Sustainable Computing, 2016
- "An Efficient Approach to Generating Location-Sensitive Recommendations

in Ad-hoc Social Network Environments", Fei Hao, Shuai Li, Geyong Min, Hee-Cheol Kim, Stephen S. Yau, and Laurence T. Yang, IEEE Transactions on Services Computing 2015

- "Online Clustering of Bandits", Claudio Gentile, Shuai Li, and Giovanni Zappella, The 31st International Conference on Machine Learning, Journal of Machine Learning Research, Acceptance Rate: 25%, (ICML 2014)
- "Dynamic Fuzzy Logic Control of Genetic Algorithm Probabilities", Huijuan Guo, Yi Feng, Fei Hao, Shentong Zhong, and Shuai Li, Journal of Computers, DOI: 10.4304/JCP. 9.1.22-27, Vol. 9, No. 1, pp. 22-27, Jan. 2014

Chapter 2

Centralized Clustering Bandits

2.1 Introduction

Presenting personalized content to users is nowdays a crucial functionality for many online recommendation services. Due to the ever-changing set of available options, these services have to exhibit strong adaptation capabilities when trying to match users' preferences. Coarsely speaking, the underlying systems repeatedly learn a mapping between available content and users, the mapping being based on *context* information (that is, sets of features) which is typically extracted from both users and contents. The need to focus on content that raises the users' interest, combined with the need of exploring new content so as to globally improve users' experience, generates a well-known exploration-exploitation dilemma, which is commonly formalized as a multi-armed bandit problem (e.g., [66, 6, 4, 20]). In particular, the contextual bandit methods (e.g., [5, 67, 69, 25, 11, 1, 27, 64, 91, 106, 34], and references therein) have rapidly become a reference algorithmic technique for implementing adaptive recommender systems.

Within the above scenarios, the widespread adoption of online social networks, where users are engaged in technology-mediated social interactions (making product endorsement and word-of-mouth advertising a common practice), raises further challenges and opportunities to content recommendation systems: On one hand, because of the mutual influence among friends, acquaintances, business partners, etc., users having strong ties are more likely to exhibit similar interests, and therefore similar behavior. On the other hand, the nature and scale of such interactions calls for adaptive algorithmic solutions which are also computationally affordable.

Incorporating social components into bandit algorithms can lead to a dramatic increase in the quality of recommendations. For instance, we may want to serve content to a group of users by taking advantage of an underlying network of social relationships among them. These social relationships can either be explicitly encoded in a graph, where adjacent nodes/users are deemed similar to one another, or implicitly contained in the data, and given as the outcome of an inference process that recognizes similarities across users based on their past behavior. Examples of the first approach are the recent works [15, 31, 41], where a social network structure over the users is assumed to be given that reflects actual interest similarities among users – see also [19, 102] for recent usage of social information to tackle the so-called "cold-start" problem. Examples of the second approach are the more traditional collaborative-filtering (e.g., [90]), content-based filtering, and hybrid approaches (e.g. [16]).

Both approaches have important drawbacks hindering their practical deployment. One obvious drawback of the "explicit network" approach is that the social network information may be misleading (see, e.g., the experimental evidence reported by [31]), or simply unavailable. Moreover, even in the case when this information is indeed available and useful, the algorithmic strategies to implement the needed feedback sharing mechanisms might lead to severe scaling issues [41], especially when the number of targeted users is large. A standard drawback of the "implicit network" approach of traditional recommender systems is that in many practically relevant scenarios (e.g., web-based), content universe and popularity often undergo dramatic changes, making these approaches difficult to apply.

In such settings, most notably in the relevant case when the involved users are many, it is often possible to identify a few subgroups or communities within which users share similar interests [87, 17], thereby greatly facilitating the targeting of users by means of *group* recommendations. Hence the system need not learn a different model for each user of the service, but just a single model for each group.

In this paper, we carry out[1] a theoretical and experimental investigation of adaptive clustering algorithms for linear (contextual) bandits under the assumption that we have to serve content to a set of n users organized into $m << n$ groups (or *clusters*) such that users within each group tend to provide similar feedback to content recommendations. We give a $O(\sqrt{T})$ regret analysis holding in a standard stochastically linear setting for payoffs where, importantly, the hidden constants in the big-oh depend on m, rather than n, as well as on the geometry of the user models within the different clusters. The main idea of our algorithm is to use confidence balls of the users' models to both estimate user similarity, and to share feedback across (deemed similar) users. The algorithm adaptively interpolates between the case when we have a single instance of a contextual bandit algorithm making the same predictions for all users and the case when we have n-many instances providing fully personalized recommendations. We show that our algorithm can be implemented efficiently (the large n scenario being of special concern here) by means of off-the-shelf data-structures relying on random graphs. Finally, we test our algorithm on medium-size synthetic and real-world datasets, often reporting a significant increase in prediction performance over known state-of-the-art methods for bandit problems.

[1] We postpone the discussion of related work to the chapter 2's supplementary material.

2.2 Learning Model

We assume the user behavior similarity is encoded as an *unknown* clustering of the users. Specifically, let $V = \{1, \ldots, n\}$ represent the set of n users. Then V can be partitioned into a small number m of clusters $V_1, V_2, \ldots, V_m$, with $m << n$, such that users lying in the same cluster share similar behavior and users lying in different clusters have different behavior. The actual partition of V (including the number of clusters m) and the common user behavior within each cluster are unknown to the learner, and have to be inferred on the fly.

Learning proceeds in a sequential fashion: At each round $t = 1, 2, \ldots,$ the learner receives a user index $i_t \in V$ together with a set of context vectors $C_{i_t} = \{\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \boldsymbol{x}_{t,c_t}\} \subseteq \mathbb{R}^d$. The learner then selects some $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_{t,k_t} \in C_{i_t}$ to recommend to user i_t, and observes some payoff $a_t \in \mathbb{R}$, which is a function of both i_t and the recommended $\bar{\boldsymbol{x}}_t$. The following assumptions are made on how index i_t, set C_{i_t}, and payoff a_t are generated in round t. Index i_t represents the user to be served by the system, and we assume i_t is selected uniformly at random[2] from V. Once i_t is selected, the number of context vectors c_t in C_{i_t} is generated arbitrarily as a function of past indices $i_1, \ldots, i_{t-1}$, payoffs $a_1, \ldots, a_{t-1}$, and sets $C_{i_1}, \ldots, C_{i_{t-1}}$, as well as the current index i_t. Then the sequence $\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \boldsymbol{x}_{t,c_t}$ of context vectors within C_{i_t} is generated i.i.d. (conditioned on i_t, c_t and all past indices $i_1, \ldots, i_{t-1}$, payoffs $a_1, \ldots, a_{t-1}$, and sets $C_{i_1}, \ldots, C_{i_{t-1}}$) from a random process on the surface of the unit sphere, whose process matrix $\mathbb{E}[XX^\top]$ is full rank, with minimal eigenvalue $\lambda > 0$. Further assumptions on the process matrix $\mathbb{E}[XX^\top]$ are made later on. Finally, payoffs are generated by noisy versions of unknown linear functions of the context vectors. That is, we assume each cluster V_j, $j = 1, \ldots, m$, hosts an unknown parameter vector $\boldsymbol{u}_j \in \mathbb{R}^d$ which is common to each user $i \in V_j$. Then the payoff value $a_i(\boldsymbol{x})$ associated with user i and context vector $\boldsymbol{x} \in \mathbb{R}^d$ is given by the random variable

$$a_i(\boldsymbol{x}) = \boldsymbol{u}_{j(i)}^\top \boldsymbol{x} + \epsilon_{j(i)}(\boldsymbol{x}) \ ,$$

where $j(i) \in \{1, 2, \ldots, m\}$ is the index of the cluster that node i belongs to, and $\epsilon_{j(i)}(\boldsymbol{x})$ is a conditionally zero-mean and bounded variance noise term. Specifically, denoting by $\mathbb{E}_t[\,\cdot\,]$ the conditional expectation $\mathbb{E}\big[\,\cdot\,\big|\,(i_1, C_{i_1}, a_1), \ldots, (i_{t-1}, C_{i_{t-1}}, a_{t-1}), i_t\,\big]$, we assume that for any fixed $j \in \{1, \ldots, m\}$ and $\boldsymbol{x} \in \mathbb{R}^d$, the variable $\epsilon_j(\boldsymbol{x})$ is such that $\mathbb{E}_t[\epsilon_j(\boldsymbol{x})|\,\boldsymbol{x}\,] = 0$ and $\mathbb{V}_t\big[\epsilon_j(\boldsymbol{x})|\,\boldsymbol{x}\,\big] \le \sigma^2$, where $\mathbb{V}_t[\,\cdot\,]$ is a shorthand for the conditional variance $\mathbb{V}\big[\,\cdot\,\big|\,(i_1, C_{i_1}, a_1), \ldots, (i_{t-1}, C_{i_{t-1}}, a_{t-1}), i_t\,\big]$ of the variable at argument. So we clearly have $\mathbb{E}_t[a_i(\boldsymbol{x})|\,\boldsymbol{x}\,] = \boldsymbol{u}_{j(i)}^\top \boldsymbol{x}$ and $\mathbb{V}_t\big[a_i(\boldsymbol{x})|\,\boldsymbol{x}\,\big] \le \sigma^2$. Therefore, $\boldsymbol{u}_{j(i)}^\top \boldsymbol{x}$ is the expected payoff observed at user i for context vector $\boldsymbol{x}$. In the special case when the noise $\epsilon_{j(i)}(\boldsymbol{x})$ is a bounded random variable taking values in the range $[-1, 1]$, this implies $\sigma^2 \le 1$. We will make throughout the assumption that

[2] Any other distribution that insures a positive probability of visiting each node of V would suffice here.

$a_i(\boldsymbol{x}) \in [-1,1]$ for all $i \in V$ and $\boldsymbol{x}$. Notice that this implies $-1 \le \boldsymbol{u}_{j(i)}^\top \boldsymbol{x} \le 1$ for all $i \in V$ and $\boldsymbol{x}$. Finally, we assume well-separatedness among the clusters, in that $||\boldsymbol{u}_j - \boldsymbol{u}_{j'}|| \ge \gamma > 0$ for all $j \ne j'$. We define the regret r_t of the learner at time t as

$$r_t = \left(\max_{\boldsymbol{x} \in C_{i_t}} \boldsymbol{u}_{j(i_t)}^\top \boldsymbol{x}\right) - \boldsymbol{u}_{j(i_t)}^\top \bar{\boldsymbol{x}}_t \,.$$

We are aimed at bounding with high probability (over the variables i_t, $\boldsymbol{x}_{t,k}$, $k = 1, \ldots, c_t$, and the noise variables $\epsilon_{j(i_t)}$) the cumulative regret $\sum_{t=1}^T r_t$. The kind of regret bound we would like to obtain (we call it the *reference* bound) is one where the clustering structure of V (i.e., the partition of V into $V_1, \ldots, V_m$) is known to the algorithm ahead of time, and we simply view each one of the m clusters as an independent bandit problem. In this case, a standard contextual bandit analysis [5, 25, 1] shows that, as T grows large, the cumulative regret $\sum_{t=1}^T r_t$ can be bounded with high probability as[3]

$$\textstyle\sum_{t=1}^T r_t = \widetilde{O}\Big(\sum_{j=1}^m \Big(\sigma\, d + ||\boldsymbol{u}_j||\, \sqrt{d}\Big)\, \sqrt{T}\Big) \,.$$

For simplicity, we shall assume that $||\boldsymbol{u}_j|| = 1$ for all $j = 1, \ldots, m$. Now, a more careful analysis exploiting our assumption about the randomness of i_t (see the supplementary material) reveals that one can replace the $\sqrt{T}$ term contributed by each bandit j by a term of the form $\sqrt{T}\,\left(\frac{1}{m} + \sqrt{\frac{|V_j|}{n}}\right)$, so that under our assumptions the reference bound becomes

$$\sum_{t=1}^T r_t = \widetilde{O}\bigg(\Big(\sigma\, d + \sqrt{d}\Big)\, \sqrt{T}\Big(1 + \sum_{j=1}^m \sqrt{\frac{|V_j|}{n}}\Big)\bigg) \,. \tag{2.1}$$

Observe the dependence of this bound on the size of clusters V_j. The worst-case scenario is when we have m clusters of the same size $\frac{n}{m}$, resulting in the bound

$$\textstyle\sum_{t=1}^T r_t = \widetilde{O}\left(\Big(\sigma\, d + \sqrt{d}\Big)\, \sqrt{m\,T}\right) \,.$$

At the other extreme lies the easy case when we have a single big cluster and many small ones. For instance, $|V_1| = n - m + 1$, and $|V_2| = |V_3| = \ldots |V_m| = 1$, for $m << n$, gives

$$\textstyle\sum_{t=1}^T r_t = \widetilde{O}\Big(\Big(\sigma\, d + \sqrt{d}\Big)\, \sqrt{T}\,\Big(1 + \frac{m}{\sqrt{n}}\Big)\Big) \,.$$

A relevant geometric parameter of the set of $\boldsymbol{u}_j$ is the *sum of distances* $SD(\boldsymbol{u}_j)$ of a given vector $\boldsymbol{u}_j$ w.r.t. the set of vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$, which we define as $SD(\boldsymbol{u}_j) = \sum_{\ell=1}^m ||\boldsymbol{u}_j - \boldsymbol{u}_\ell||$. If it is known that $SD(\boldsymbol{u}_j)$ is small for all j, one can modify the abovementioned independent bandit algorithm, by letting the bandits share signals, as is done, e.g., in [41]. This allows one to exploit the vicinity of

[3] The $\widetilde{O}$-notation hides logarithmic factors.

the $\boldsymbol{u}_j$ vectors, and roughly replace $1 + \sum_{j=1}^{m} \sqrt{\frac{|V_j|}{n}}$ in (2.1) by a quantity also depending on the mutual distances $||\boldsymbol{u}_j - \boldsymbol{u}_{j'}||$ among cluster vectors. However, this improvement is obtained at the cost of a substantial increase of running time [41]. In our analysis (Theorem 1 in Section 4.4), we would like to leverage both the geometry of the clusters, as encoded by vectors $\boldsymbol{u}_j$, and the relative size $|V_j|$ of the clusters, with no prior knowledge of m (or γ), and without too much extra computational burden.

2.3 The Algorithm

Our algorithm, called Cluster of Bandits (CLUB), is described in Figure 2.1. In order to describe the algorithm we find it convenient to re-parameterize the problem and introduce n parameter vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$, one per node, where nodes within the same cluster V_j share the same vector. An illustrative example is given in Figure 2.2.

The algorithm maintains at time t an estimate $\boldsymbol{w}_{i,t}$ for vector $\boldsymbol{u}_i$ associated with user $i \in V$. Vectors $\boldsymbol{w}_{i,t}$ are updated based on the payoff signals, similar to a standard linear bandit algorithm (e.g., [25]) operating on the context vectors contained in C_{i_t}. Every user i in V hosts a linear bandit algorithm like the one described in [41]. One can see that the prototype vector $\boldsymbol{w}_{i,t}$ is the result of a standard linear least-squares approximation to the corresponding unknown parameter vector $\boldsymbol{u}_i$. In particular, $\boldsymbol{w}_{i,t-1}$ is defined through the inverse correlation matrix $M_{i,t-1}^{-1}$, and the additively-updated vector $\mathbf{b}_{i,t-1}$. Matrices $M_{i,t}$ are initialized to the $d \times d$ identity matrix, and vectors $\mathbf{b}_{i,t}$ are initialized to the d-dimensional zero vector. In addition, the algorithm maintains at time t an undirected graph $G_t = (V, E_t)$ whose nodes are precisely the users in V. The algorithm starts off from the complete graph, and progressively erases edges based on the evolution of vectors $\boldsymbol{w}_{i,t}$. The graph is intended to encode the current partition of V by means of the *connected components* of G_t. We denote by $\hat{V}_{1,t}, \hat{V}_{2,t}, \ldots, \hat{V}_{m_t,t}$ the partition of V induced by the connected components of G_t. Initially, we have $m_1 = 1$ and $\hat{V}_{1,1} = V$. The clusters $\hat{V}_{1,1}, \hat{V}_{2,t}, \ldots, \hat{V}_{m_t,t}$ (henceforth called the *current* clusters) are indeed meant to estimate the underlying true partition $V_1, V_2, \ldots, V_m$, henceforth called the *underlying* or *true* clusters.

At each time $t = 1, 2, \ldots$, the algorithm receives the index i_t of the user to serve, and the associated context vectors $\boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,c_t}$ (the set C_{i_t}), and must select one among them. In doing so, the algorithm first determines which cluster (among $\hat{V}_{1,1}, \hat{V}_{2,t}, \ldots, \hat{V}_{m_t,t}$) node i_t belongs to, call this cluster $\hat{V}_{\widehat{j}_t,t}$, then builds the aggregate weight vector $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ by taking prior $\bar{\boldsymbol{x}}_s$, $s < t$, such that $i_s \in \hat{V}_{\widehat{j}_t,t}$, and computing the least squares approximation as if all nodes $i \in \hat{V}_{\widehat{j}_t,t}$ have been collapsed into one. It is weight vector $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ that the algorithm uses to select k_t. In particular,

$$k_t = \operatorname*{argmax}_{k=1,\ldots,c_t} \left(\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^{\top} \boldsymbol{x}_{t,k} + \mathrm{CB}_{\widehat{j}_t,t-1}(\boldsymbol{x}_{t,k}) \right) .$$

```
Input: Exploration parameter $\alpha > 0$; edge deletion parameter $\alpha_2 > 0$
Init:
  • $\mathbf{b}_{i,0} = \mathbf{0} \in \mathbb{R}^d$ and $M_{i,0} = I \in \mathbb{R}^{d\times d}$, $i = 1, \dots n$;
  • Clusters $\hat{V}_{1,1} = V$, number of clusters $m_1 = 1$;
  • Graph $G_1 = (V, E_1)$, $G_1$ is connected over $V$.
for $t = 1, 2, \dots, T$ do
    Set $\boldsymbol{w}_{i,t-1} = M_{i,t-1}^{-1}\mathbf{b}_{i,t-1}, \quad i = 1, \dots, n$;
    Receive $i_t \in V$, and get context $C_{i_t} = \{\boldsymbol{x}_{t,1}, \dots, \boldsymbol{x}_{t,c_t}\}$;
    Determine $\widehat{j}_t \in \{1, \dots, m_t\}$ such that $i_t \in \hat{V}_{\widehat{j}_t,t}$, and set
        $$\bar{M}_{\widehat{j}_t,t-1} = I + \sum_{i\in\hat{V}_{\widehat{j}_t,t}} (M_{i,t-1} - I),$$
        $$\bar{\mathbf{b}}_{\widehat{j}_t,t-1} = \sum_{i\in\hat{V}_{\widehat{j}_t,t}} \mathbf{b}_{i,t-1},$$
        $$\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1} = \bar{M}_{\widehat{j}_t,t-1}^{-1}\bar{\mathbf{b}}_{\widehat{j}_t,t-1}\ ;$$
    Set $k_t = \operatorname*{argmax}_{k=1,\dots,c_t} \left(\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^\top \boldsymbol{x}_{t,k} + \mathrm{CB}_{\widehat{j}_t,t-1}(\boldsymbol{x}_{t,k})\right)$,
        $$\mathrm{CB}_{j,t-1}(\boldsymbol{x}) = \alpha\sqrt{\boldsymbol{x}^\top \bar{M}_{j,t-1}^{-1}\boldsymbol{x}\,\log(t+1)},$$
        $$\bar{M}_{j,t-1} = I + \sum_{i\in\hat{V}_{j,t}} (M_{i,t-1} - I)\,, \quad j = 1, \dots, m_t\ .$$
    Observe payoff $a_t \in [-1, 1]$;
    Update weights:
      •  $M_{i_t,t} = M_{i_t,t-1} + \bar{\boldsymbol{x}}_t\bar{\boldsymbol{x}}_t^\top$,
      •  $\mathbf{b}_{i_t,t} = \mathbf{b}_{i_t,t-1} + a_t\bar{\boldsymbol{x}}_t$,
      •  Set $M_{i,t} = M_{i,t-1}$, $\mathbf{b}_{i,t} = \mathbf{b}_{i,t-1}$ for all $i \neq i_t$ ;
    Update clusters:
      •  Delete from $E_t$ all $(i_t, \ell)$ such that
            $$||\boldsymbol{w}_{i_t,t-1} - \boldsymbol{w}_{\ell,t-1}|| > \widetilde{\mathrm{CB}}_{i_t,t-1} + \widetilde{\mathrm{CB}}_{\ell,t-1}\ ,$$
            $$\widetilde{\mathrm{CB}}_{i,t-1} = \alpha_2\sqrt{\frac{1+\log(1+T_{i,t-1})}{1+T_{i,t-1}}},$$
            $$T_{i,t-1} = |\{s \le t-1\,:\, i_s = i\}|, \qquad i \in V;$$
      •  Let $E_{t+1}$ be the resulting set of edges, set $G_{t+1} = (V, E_{t+1})$, and compute
         associated clusters $\hat{V}_{1,t+1}, \hat{V}_{2,t+1}, \dots, \hat{V}_{m_{t+1},t+1}$ .
end for
```

Figure 2.1: Pseudocode of the CLUB algorithm. The confidence functions $\mathrm{CB}_{j,t-1}$ and $\widetilde{\mathrm{CB}}_{i,t-1}$ are simplified versions of their "theoretical" counterparts $\mathrm{TCB}_{j,t-1}$ and $\widetilde{\mathrm{TCB}}_{i,t-1}$, defined later on. The factors α and α_2 are used here as tunable parameters that bridge the simplified versions to the theoretical ones.

The quantity $\mathrm{CB}_{\widehat{j}_t,t-1}(\boldsymbol{x})$ is a version of the upper confidence bound in the approximation of $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ to a suitable combination of vectors $\boldsymbol{u}_i$, $i \in \hat{V}_{\widehat{j}_t,t}$ – see the supplementary material for details.

Once this selection is done and the associated payoff a_t is observed, the algorithm uses the selected vector $\bar{\boldsymbol{x}}_t$ for updating $M_{i_t,t-1}$ to $M_{i_t,t}$ via a rank-one adjustment, and for turning vector $\mathbf{b}_{i_t,t-1}$ to $\mathbf{b}_{i_t,t}$ via an additive update whose learning rate is precisely a_t. Notice that the update is only performed at node i_t, since for all other $i \neq i_t$ we have $\boldsymbol{w}_{i,t} = \boldsymbol{w}_{i,t-1}$. However, this update at i_t will also implicitly update the aggregate weight vector $\bar{\boldsymbol{w}}_{\widehat{j}_{t+1},t}$ associated with cluster $\hat{V}_{\widehat{j}_{t+1},t+1}$ that node i_t will happen to belong to in the next round. Finally, the cluster structure is possibly modified. At this point CLUB compares, for all existing edges $(i_t,\ell) \in E_t$, the distance $||\boldsymbol{w}_{i_t,t-1} - \boldsymbol{w}_{\ell,t-1}||$ between vectors $\boldsymbol{w}_{i_t,t-1}$ and $\boldsymbol{w}_{\ell,t-1}$ to the quantity $\widetilde{\mathrm{CB}}_{i_t,t-1} + \widetilde{\mathrm{CB}}_{\ell,t-1}$. If the above distance is significantly large (and $\boldsymbol{w}_{i_t,t-1}$ and $\boldsymbol{w}_{\ell,t-1}$ are good approximations to the respective underlying vectors $\boldsymbol{u}_{i_t}$ and $\boldsymbol{u}_\ell$), then this is a good indication that $\boldsymbol{u}_{i_t} \neq \boldsymbol{u}_\ell$ (i.e., that node i_t and node ℓ cannot belong to the same true cluster), so that edge (i_t,ℓ) gets deleted. The new graph G_{t+1}, and the induced partitioning clusters $\hat{V}_{1,t+1}, \hat{V}_{2,t+1}, \ldots, \hat{V}_{m_{t+1},t+1}$, are then computed, and a new round begins.

2.3.1 Implementation

In implementing the algorithm in Figure 2.1, the reader should bear in mind that we are expecting n (the number of users) to be quite large, d (the number of features of each item) to be relatively small, and m (the number of true clusters) to be very small compared to n. With this in mind, the algorithm can be implemented by storing a least-squares estimator $\boldsymbol{w}_{i,t-1}$ at each node $i \in V$, an aggregate least squares estimator $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ for each current cluster $\widehat{j}_t \in \{1,\ldots,m_t\}$, and an extra data-structure which is able to perform decremental dynamic connectivity. Fast implementations of such data-structures are those studied by [100, 55] (see also the research thread referenced therein). One can show (see the supplementary material) that in T rounds we have an overall (expected) running time

$$O\Big(T\left(d^2 + \frac{|E_1|}{n}\,d\right) + m\,(n\,d^2 + d^3) + |E_1| \\ + \min\{n^2, |E_1|\,\log n\} + \sqrt{n\,|E_1|}\,\log^{2.5} n\Big)\,. \qquad (2.2)$$

Notice that the above is $n \cdot \mathrm{poly}(\log n)$, if so is $|E_1|$. In addition, if T is large compared to n and d, the average running time per round becomes $O(d^2 + d \cdot \mathrm{poly}(\log n))$. As for memory requirements, this implementation takes $O(n\,d^2 + m\,d^2 + |E_1|) = O(n\,d^2 + |E_1|)$. Again, this is $n \cdot \mathrm{poly}(\log n)$ if so is $|E_1|$.

2.3.2 Regret Analysis

Our analysis relies on the high probability analysis contained in [1] (Theorems 1 and 2 therein). The analysis (Theorem 1 below) is carried out in the case when the initial graph G_1 is the complete graph. However, if the true clusters are sufficiently large, then we can show (see Remark 4) that a formal statement can be made even if we start off from sparser random graphs, with substantial time and memory savings.

The analysis actually refers to a version of the algorithm where the confidence bound functions $\text{CB}_{j,t-1}(\cdot)$ and $\widetilde{\text{CB}}_{i,t-1}$ in Figure 2.1 are replaced by their "theoretical" counterparts $\text{TCB}_{j,t-1}(\cdot)$, and $\widetilde{\text{TCB}}_{i,t-1}$, respectively,[4] which are defined as follows. Set for brevity

$$A_\lambda(T,\delta)=\left(\frac{\lambda T}{4}-8\log\Big(\frac{T+3}{\delta}\Big)-2\sqrt{T\log\Big(\frac{T+3}{\delta}\Big)}\right)_+$$

where $(x)_+ = \max\{x,0\}$, $x \in \mathbb{R}$. Then, for $j = 1,\dots,m_t$,

$$\text{TCB}_{j,t-1}(\boldsymbol{x}) = \sqrt{\boldsymbol{x}^\top \bar{M}_{j,t-1}^{-1}\boldsymbol{x}}\left(\sigma\sqrt{2\log\frac{|\bar{M}_{j,t-1}|}{\delta/2}}+1\right), \tag{2.3}$$

being $|\cdot|$ the determinant of the matrix at argument, and, for $i \in V$,

$$\widetilde{\text{TCB}}_{i,t-1} = \frac{\sigma\sqrt{2d\,\log t + 2\log(2/\delta)}+1}{\sqrt{1+A_\lambda(T_{i,t-1},\delta/(2nd))}}\,. \tag{2.4}$$

Recall the difference between *true* clusters $V_1,\dots,V_m$ and *current* clusters $\hat{V}_{1,t},\dots,\hat{V}_{m_t,t}$ maintained by the algorithm at time t. Consistent with this difference, we let $G = (V,E)$ be the true underlying graph, made up of the m disjoint cliques over the sets of nodes $V_1,\dots,V_m \subseteq V$, and $G_t = (V,E_t)$ be the one kept by the algorithm – see again Figure 2.2 for an illustration of how the algorithm works. The following is the main theoretical result of this chapter,[5] where additional conditions are needed on the process X generating the context vectors.

Theorem 1. *Let the CLUB algorithm of Figure 2.1 be run on the initial complete graph $G_1 = (V,E_1)$, whose nodes $V = \{1,\dots,n\}$ can be partitioned into m clusters $V_1,\dots,V_m$ where, for each $j = 1,\dots,m$, nodes within cluster V_j host the same vector $\boldsymbol{u}_j$, with $||\boldsymbol{u}_j|| = 1$ for $j = 1,\dots,m$, and $||\boldsymbol{u}_j - \boldsymbol{u}_{j'}|| \ge \gamma > 0$ for any $j \neq j'$. Denote by $v_j = |V_j|$ the cardinality of cluster V_j. Let the $\text{CB}_{j,t}(\cdot)$ function in Figure 2.1 be replaced by the $\text{TCB}_{j,t}(\cdot)$ function defined in (2.3), and $\widetilde{\text{CB}}_{i,t}$ be replaced by $\widetilde{\text{TCB}}_{i,t}$ defined in (2.4). In both $\text{TCB}_{j,t}$ and $\widetilde{\text{TCB}}_{i,t}$,*

[4]Notice that, in all our notations, index i always ranges over nodes, while index j always ranges over clusters. Accordingly, the quantities $\widetilde{\text{CB}}_{i,t}$ and $\widetilde{\text{TCB}}_{i,t}$ are always associates with node $i \in V$, while the quantities $\text{CB}_{j,t-1}(\cdot)$ and $\text{TCB}_{j,t-1}(\cdot)$ are always associates with clusters $j \in \{1,\dots,m_t\}$.

[5] The proof is provided in the supplementary material of chapter 2.

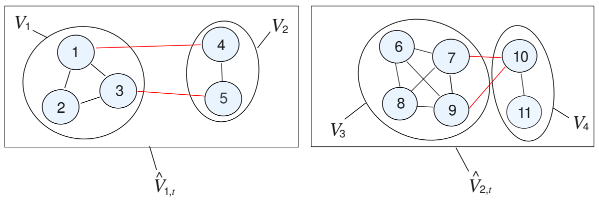


Figure 2.2: A true underlying graph $G = (V, E)$ made up of $n = |V| = 11$ nodes, and $m = 4$ true clusters $V_1 = \{1, 2, 3\}$, $V_2 = \{4, 5\}$, $V_3 = \{6, 7, 8, 9\}$, and $V_4 = \{10, 11\}$. There are $m_t = 2$ current clusters $\hat{V}_{1,t}$ and $\hat{V}_{2,t}$. The black edges are the ones contained in E, while the red edges are those contained in $E_t \setminus E$. The two current clusters also correspond to the two connected components of graph $G_t = (V, E_t)$. Since aggregate vectors $\bar{\boldsymbol{w}}_{j,t}$ are build based on current cluster membership, if for instance, $i_t = 3$, then $\widehat{j}_t = 1$, so $\bar{M}_{1,t-1} = I + \sum_{i=1}^{5}(M_{i,t-1} - I)$, $\bar{\mathbf{b}}_{1,t-1} = \sum_{i=1}^{5} \mathbf{b}_{i,t-1}$, and $\bar{\boldsymbol{w}}_{1,t-1} = \bar{M}_{1,t-1}^{-1}\bar{\mathbf{b}}_{1,t-1}$.

let δ therein be replaced by $\delta/10.5$. Let, at each round t, context vectors $C_{i_t} = \{\boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,c_t}\}$ being generated i.i.d. (conditioned on i_t, c_t and all past indices $i_1, \ldots, i_{t-1}$, payoffs $a_1, \ldots, a_{t-1}$, and sets $C_{i_1}, \ldots, C_{i_{t-1}}$) from a random process X such that $||X|| = 1$, $\mathbb{E}[XX^\top]$ is full rank, with minimal eigenvalue $\lambda > 0$. Moreover, for any fixed unit vector $\boldsymbol{z} \in \mathbb{R}^d$, let the random variable $(\boldsymbol{z}^\top X)^2$ be (conditionally) sub-Gaussian with variance parameter $\nu^2 = \mathbb{V}_t\big[(\boldsymbol{z}^\top X)^2 \,|\, c_t\big] \le \frac{\lambda^2}{8\log(4c)}$, with $c_t \le c$ for all t. Then with probability at least $1 - \delta$ the cumulative regret satisfies

$$\sum_{t=1}^{T} r_t = \widetilde{O}\Bigg((\sigma\sqrt{d}+1)\sqrt{m}\left(\frac{n}{\lambda^2} + \sqrt{T}\Big(1 + \sum_{j=1}^{m}\sqrt{\frac{v_j}{\lambda n}}\Big)\right) + \left(\frac{n}{\lambda^2} + \frac{n\sigma^2 d}{\lambda\gamma^2}\right)\mathbb{E}[SD(\boldsymbol{u}_{i_t})] + m\Bigg)$$
$$= \widetilde{O}\Bigg((\sigma\sqrt{d}+1)\sqrt{mT}\Big(1 + \sum_{j=1}^{m}\sqrt{\frac{v_j}{\lambda n}}\Big)\Bigg), \tag{2.5}$$

as T grows large. In the above, the $\widetilde{O}$-notation hides $\log(1/\delta)$, $\log m$, $\log n$, and $\log T$ factors.

Remark 1. *A close look at the cumulative regret bound presented in Theorem 1 reveals that this bound is made up of three main terms: The first term is of the form*

$$(\sigma\sqrt{dm} + \sqrt{m})\,\frac{n}{\lambda^2} + m\,.$$

This term is constant with T, and essentially accounts for the transient regime due to the convergence of the minimal eigenvalues of $\bar{M}_{j,t}$ and $M_{i,t}$ to the corresponding minimal eigenvalue λ of $\mathbb{E}[XX^\top]$. The second term is of the form

$$\Big(\frac{n}{\lambda^2} + \frac{n\,\sigma^2\,d}{\lambda\gamma^2}\Big)\,\mathbb{E}[SD(\boldsymbol{u}_{i_t})]\ .$$

This term is again constant with T, but it depends through $\mathbb{E}[SD(\boldsymbol{u}_{i_t})]$ on the geometric properties of the set of $\boldsymbol{u}_j$ as well as on the way such $\boldsymbol{u}_j$ interact with the cluster sizes v_j. Specifically,

$$\mathbb{E}[SD(\boldsymbol{u}_{i_t})] = \textstyle\sum_{j=1}^{m} \frac{v_j}{n} \sum_{j'=1}^{m} ||\boldsymbol{u}_j - \boldsymbol{u}_{j'}||\ .$$

Hence this term is small if, say, among the m clusters, a few of them together cover almost all nodes in V (this is a typical situation in practice) and, in addition, the corresponding $\boldsymbol{u}_j$ are close to one another. This term accounts for the hardness of learning the true underlying clustering through edge pruning. We also have an inverse dependence on γ^2, which is likely due to an artifact of our analysis. Recall that γ is not known to our algorithm. Finally, the third term is the one characterizing the asymptotic behavior of our algorithm as $T \to \infty$, its form being just (2.5). It is instructive to compare this term to the reference bound (2.1) obtained by assuming prior knowledge of the cluster structure. Broadly speaking, (2.5) has an extra $\sqrt{m}$ factor,[6] and replaces a factor $\sqrt{d}$ in (2.1) by the larger factor $\sqrt{\frac{1}{\lambda}}$.

Remark 2. *The reader should observe that a similar algorithm as CLUB can be designed that starts off from the empty graph instead, and progressively draws edges (thereby merging connected components and associated aggregate vectors) as soon as two nodes host individual vectors $\boldsymbol{w}_{i,t}$ which are close enough to one another. This would have the advantage to lean on even faster data-structures for maintaining disjoint sets (e.g., [26][Ch. 22]), but has also the significant drawback of requiring prior knowledge of the separation parameter γ. In fact, it would not be possible to connect two previously unconnected nodes without knowing something about this parameter. A regret analysis similar to the one in Theorem 1 exists, though our current understanding is that the cumulative regret would depend linearly on $\sqrt{n}$ instead of $\sqrt{m}$. Intuitively, this algorithm is biased towards a large number of true clusters, rather than a small number.*

Remark 3. *A data-dependent variant of the CLUB algorithm can be designed and analyzed which relies on data-dependent clusterability assumptions of the set of users with respect to a set of context vectors. These data-dependent assumptions allow us to work in a fixed design setting for the sequence of context vectors $\boldsymbol{x}_{t,k}$, and remove the sub-Gaussian and full-rank hypotheses regarding $\mathbb{E}[XX^\top]$. On the other hand, they also require that the power of the adversary generating context vectors be suitably restricted. See the supplementary material for details.*

[6] This extra factor could be eliminated at the cost of having a higher second term in the bound, which does not leverage the geometry of the set of $\boldsymbol{u}_j$.

Remark 4. *Last but not least, we would like to stress that the same analysis contained in Theorem 1 extends to the case when we start off from a p-random Erdos-Renyi initial graph $G_1 = (V, E_1)$, where p is the independent probability that two nodes are connected by an edge in G_1. Translated into our context, a classical result on random graphs due to [58] reads as follows.*

Lemma 2. *Given $V = \{1, \ldots, n\}$, let $V_1, \ldots, V_m$ be a partition of V, where $|V_j| \geq s$ for all $j = 1, \ldots, m$. Let $G_1 = (V, E_1)$ be a p-random Erdos-Renyi graph with $p \geq \frac{12 \log(6n^2/\delta)}{s-1}$. Then with probability at least $1 - \delta$ (over the random draw of edges), all m subgraphs induced by true clusters $V_1, \ldots, V_m$ on G_1 are connected in G_1.*

For instance, if $|V_j| = \beta \frac{n}{m}$, $j = 1, \ldots, m$, for some constant $\beta \in (0, 1)$, then it suffices to have $|E_1| = O\left(\frac{m\, n \log(n/\delta)}{\beta}\right)$. Under these assumptions, if the initial graph G_1 is such a random graph, it is easy to show that Theorem 1 still holds. As mentioned in Section 2.3.1 (Eq. (2.2) therein), the striking advantage of beginning with a sparser connected graph than the complete graph is computational, since we need not handle anymore a (possibly huge) data-structure having n^2-many items. In our experiments, described next, we set $p = \frac{3 \log n}{n}$, so as to be reasonably confident that G_1 is (at the very least) connected.

2.4 Experiments

We tested our algorithm on both artificial and freely available real-world datasets against standard bandit baselines.

2.4.1 Datasets

Artificial datasets. We firstly generated synthetic datasets, so as to have a more controlled experimental setting. We tested the relative performance of the algorithms along different axes: number of underlying clusters, balancedness of cluster sizes, and amount of payoff noise. We set $c_t = 10$ for all $t = 1, \ldots, T$, with time horizon $T = 5,000 + 50,000$, $d = 25$, and $n = 500$. For each cluster V_j of users, we created a random unit norm vector $\boldsymbol{u}_j \in \mathbb{R}^d$. All d-dimensional context vectors $\boldsymbol{x}_{t,k}$ have then been generated uniformly at random on the surface of the Euclidean ball. The payoff value associated with cluster vector $\boldsymbol{u}_j$ and context vector $\boldsymbol{x}_{t,k}$ has been generated by perturbing the inner product $\boldsymbol{u}_j^\top \boldsymbol{x}_{t,k}$ through an additive white noise term ϵ drawn uniformly at random across the interval $[-\sigma, \sigma]$. It is the value of σ that determines the amount of payoff noise. The two remaining parameters are the number of clusters m and the clusters' relative size. We assigned to cluster V_j a number of users $|V_j|$ calculated as[7] $|V_j| = n \frac{j^{-z}}{\sum_{\ell=1}^{m} \ell^{-z}}$, $j = 1, \ldots, m$, with

[7] We took the integer part in this formula, and reassigned the remaining fractionary parts of users to the first cluster.

$z \in \{0, 1, 2, 3\}$, so that $z = 0$ corresponds to equally-sized clusters, and $z = 3$ yields highly unbalanced cluster sizes. Finally, the sequence of served users i_t is generated uniformly at random over the n users.

LastFM & Delicious datasets. These datasets are extracted from the music streaming service Last.fm and the social bookmarking web service Delicious. The LastFM dataset contains $n = 1{,}892$ nodes, and 17,632 items (artists). This dataset contains information about the listened artists, and we used this information to create payoffs: if a user listened to an artist at least once the payoff is 1, otherwise the payoff is 0. Delicious is a dataset with $n = 1{,}861$ users, and 69,226 items (URLs). The payoffs were created using the information about the bookmarked URLs for each user: the payoff is 1 if the user bookmarked the URL, otherwise the payoff is 0.[8] These two datasets are inherently different: on Delicious, payoffs depend on users more strongly than on LastFM, that is, there are more popular artists whom everybody listens to than popular websites which everybody bookmarks. LastFM is a "few hits" scenario, while Delicious is a "many niches" scenario, making a big difference in recommendation practice. Preprocessing was carried out by closely following previous experimental settings, like the one in [41]. In particular, we only retained the first 25 principal components of the context vectors resulting from a tf-idf representation of the available items, so that on both datasets $d = 25$. We generated random context sets C_{i_t} of size $c_t = 25$ for all t by selecting index i_t at random over the n users, then picking 24 vectors at random from the available items, and one among those with nonzero payoff for user i_t.[9] We repeated this process $T = 5,000 + 50,000$ times for the two datasets.

Yahoo dataset. We extracted two datasets from the one adopted by the "ICML 2012 Exploration and Exploitation 3 Challenge"[10] for news article recommendation. Each user is represented by a 136-dimensional binary feature vector, and we took this feature vector as a proxy for the identity of the user. We operated on the first week of data. After removing "empty" users,[11] this gave rise to a dataset of $8, 362, 905$ records, corresponding to $n = 713, 862$ distinct users. The overall number of distinct news items turned out to be 323, c_t changing from round to round, with a maximum of 51, and a median of 41. The news items have no features, hence they have been represented as d-dimensional *versors*, with $d = 323$. Payoff values a_t are either 0 or 1 depending on whether the logged web system which these data refer to has observed a positive (click) or negative (no-click) feedback from the user in round t. We then extracted the two datasets "5k users" and "18k users" by filtering out users that have occurred less than 100 times and less than 50 times, respectively. Since the system's recommendation need not coincide with the recommendation issued by the algorithms we tested, we could only retain

[8] Datasets and their full descriptions are available at `www.grouplens.org/node/462`.

[9] This is done so as to avoid a meaningless comparison: With high probability, a purely random selection would result in payoffs equal to zero for all the context vectors in C_{i_t}.

[10] `https://explochallenge.inria.fr/`

[11] Out of the 136 Boolean features, the first feature is always 1 throughout all records. We call "empty" the users whose only nonzero feature is the first feature.

the records on which the two recommendations were indeed the same. Because records are discarded on the fly, the actual number of retained records changes across algorithms, but it is about $50,000$ for the "5k users" version and about $70,000$ for the "18k users" version.

2.4.2 Algorithms

We compared CLUB with two main competitors: LinUCB-ONE and LinUCB-IND. Both competitors are members of the LinUCB family of algorithms [5, 25, 69, 1, 41]. LinUCB-ONE allocates a single instance of LinUCB across all users (thereby making the same prediction for all users), whereas LinUCB-IND ("LinUCB INDependent") allocates an independent instance of LinUCB to each user, thereby making predictions in a fully personalised fashion. Moreover, on the synthetic experiments, we added two idealized baselines: a GOBLIN-like algorithm [41] fed with a Laplacian matrix encoding the true underlying graph G, and a CLAIRVOYANT algorithm that knows the true clusters a priori, and runs one instance of LinUCB *per cluster*. Notice that an experimental comparison to multitask-like algorithms, like GOBLIN, or to the idealized algorithm that knows all clusters beforehand, can only be done on the artificial datasets, not in the real-world case where no cluster information is available. On the Yahoo dataset, we tested the featureless version of the LinUCB-like algorithm in [41], which is essentially a version of the UCB1 algorithm of [6]. The corresponding ONE and IND versions are denoted by UCB-ONE and UCB-IND, respectively. On this dataset, we also tried a single instance of UCB-V [4] across all users, the winner of the abovementioned ICML Challenge. Finally, all algorithms have also been compared to the trivial baseline (denoted by RAN) that picks the item within C_{i_t} fully at random.

As for parameter tuning, CLUB was run with $p = \frac{3\log n}{n}$, so as to be reasonably confident that the initial graph is at least connected. In fact, after each generation of the graph, we checked for its connectedness, and repeated the process until the graph happened to be connected.[12] All algorithms (but RAN) require parameter tuning: an exploration-exploitation tradeoff parameter which is common to all algorithms (in Figure 2.1, this is the α parameter), and the edge deletion parameter α_2 in CLUB. On the synthetic datasets, as well as on the LastFM and Delicious datasets, we tuned these parameters by picking the best setting (as measured by cumulative regret) after the first $t_0 = 5,000$ rounds, and then sticked to those values for the remaining $T - t_0 = 50,000$ rounds. It is these $50,000$ rounds that our plots refer to. On the Yahoo dataset, this optimal tuning was done within the first $t_0 = 100,000$ records, corresponding to a number of retained records between $4,350$ and $4,450$ across different algorithms.

[12] Our results are averaged over 5 random initial graphs, but this randomness turned out to be a minor source of variance.

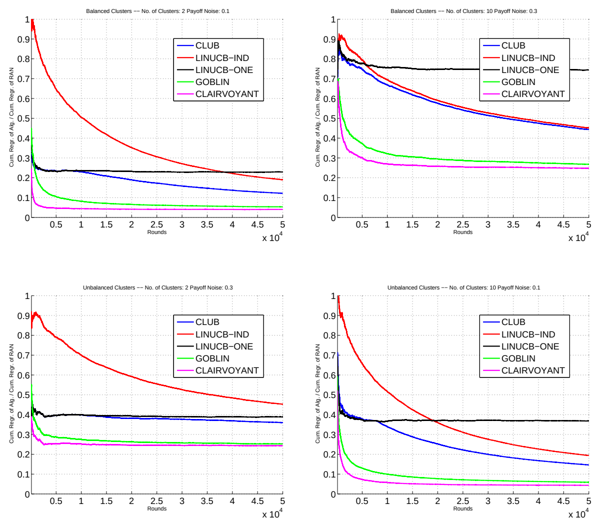


Figure 2.3: Results on synthetic datasets. Each plot displays the behavior of the ratio of the current cumulative regret of the algorithm ("Alg") to the current cumulative regret of RAN, where "Alg" is either "CLUB" or "LinUCB-IND" or "LinUCB-ONE" or "GOBLIN"or "CLAIRVOYANT". In the top two plots cluster sizes are balanced ($z = 0$), while in the bottom two they are unbalanced ($z = 2$).

2.4.3 Results

Our results are summarized in[13] Figures 2.3, 2.4, and 4.5. On the synthetic datasets (Figure 2.3) and the LastFM and Delicious datasets (Figure 2.4) we measured the ratio of the cumulative regret of the algorithm to the cumulative regret of the random predictor RAN (so that the lower the better). On the synthetic datasets, we did so under combinations of number of clusters, payoff noise, and cluster size balancedness. On the Yahoo dataset (Figure 4.5), because the only available payoffs are those associated with the items recommended in the logs, we instead measured the Clickthrough Rate (CTR), i.e., the fraction of times we get $a_t = 1$ out of the number of retained records so far (so the higher the better). This experimental setting is in line with previous ones (e.g., [69]) and, by the way data have been prepared, gives rise to a reliable estimation of actual CTR behavior under the tested experimental conditions [70].

Based on the experimental results, some trends can be spotted: On the **synthetic datasets**, CLUB always outperforms its uninformed competitors LinUCB-IND and LinUCB-ONE, the gap getting larger as we either decrease the number of underlying clusters or we make the clusters sizes more and more unbalanced. Moreover, CLUB can clearly interpolate between these two competitors taking, in a sense, the best of both. On the other hand (and unsurprisingly), the informed competitors GOBLIN and CLEARVOYANT outperform all uninformed ones. On the **"few hits"** scenario of LastFM, CLUB is again outperforming both of its competitors. However, this is not happening in the **"many niches"** case delivered by

[13]Further plots can be found in the supplementary material.

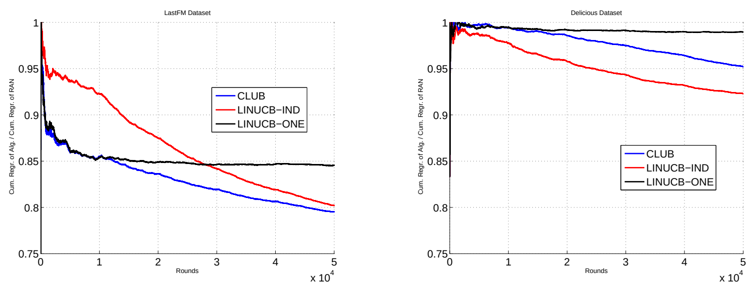


Figure 2.4: Results on the LastFM (left) and the Delicious (right) datasets. The two plots display the behavior of the ratio of the current cumulative regret of the algorithm (“Alg”) to the current cumulative regret of RAN, where “Alg” is either “CLUB” or “LinUCB-IND” or “LinUCB-ONE”.

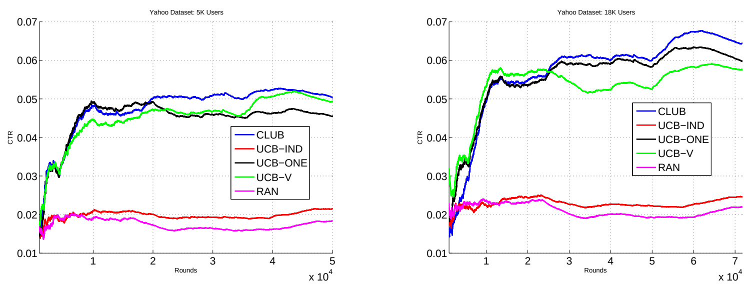


Figure 2.5: Plots on the Yahoo datasets reporting Clickthrough Rate (CTR) over time, i.e., the fraction of times the algorithm gets payoff one out of the number of retained records so far.

the Delicious dataset, where CLUB is clearly outperformed by LinUCB-IND. The proposed alternative of CLUB that starts from an empty graph (Remark 2) might be an effective alternative in this case. On the **Yahoo datasets** we extracted, CLUB tends to outperform its competitors, when measured by CTR curves, thereby showing that clustering users solely based on past behavior can be beneficial. In general, CLUB seems to benefit from situations where it is not immediately clear which is the winner between the two extreme solutions (Lin)UCB-ONE and (Lin)UCB-IND, and an adaptive interpolation between these two is needed.

2.5 Supplementary

This supplementary material contains all proofs and technical details omitted from the main text, along with ancillary comments, discussion about related work, and extra experimental results.

2.5.1 Proof of Theorem 1

The following sequence of lemmas are of preliminary importance. The first one needs extra variance conditions on the process X generating the context vectors.

We find it convenient to introduce the node counterpart to $\mathrm{TCB}_{j,t-1}(\boldsymbol{x})$, and the cluster counterpart to $\widetilde{\mathrm{TCB}}_{i,t-1}$. Given round t, node $i \in V$, and cluster index $j \in \{1, \dots, m_t\}$, we let

$$\mathrm{TCB}_{i,t-1}(\boldsymbol{x}) = \sqrt{\boldsymbol{x}^\top M_{i,t-1}^{-1}\boldsymbol{x}}\,\left(\sigma\sqrt{2\,\log\frac{|M_{i,t-1}|}{\delta/2}}+1\right)$$

$$\widetilde{\mathrm{TCB}}_{j,t-1} = \frac{\sigma\sqrt{2d\,\log t + 2\log(2/\delta)}+1}{\sqrt{1+A_\lambda(\bar{T}_{j,t-1}, \delta/(2^{m+1}d))}}\,,$$

being

$$\bar{T}_{j,t-1} = \sum_{i\in\hat{V}_{j,t}} T_{i,t-1} = |\{s \le t-1\,:\, i_s \in \hat{V}_{j,t}\}|\,,$$

i.e., the number of past rounds where a node lying in cluster $\hat{V}_{j,t}$ was served. From a notational standpoint, notice the difference[14] between $\widetilde{\mathrm{TCB}}_{i,t-1}$ and $\mathrm{TCB}_{i,t-1}(\boldsymbol{x})$, both referring to a single node $i \in V$, and $\widetilde{\mathrm{TCB}}_{j,t-1}$ and $\mathrm{TCB}_{j,t-1}(\boldsymbol{x})$ which refer to an aggregation (cluster) of nodes j among the available ones at time t.

Lemma 3. *Let, at each round t, context vectors $C_{i_t} = \{\boldsymbol{x}_{t,1}, \dots, \boldsymbol{x}_{t,c_t}\}$ being generated i.i.d. (conditioned on i_t, c_t and all past indices $i_1, \dots, i_{t-1}$, rewards $a_1, \dots, a_{t-1}$, and sets $C_{i_1}, \dots, C_{i_{t-1}}$) from a random process X such that $||X|| = 1$, $\mathbb{E}[XX^\top]$ is full rank, with minimal eigenvalue $\lambda > 0$. Let also, for any fixed unit vector $\boldsymbol{z} \in \mathbb{R}^d$, the random variable $(\boldsymbol{z}^\top X)^2$ be (conditionally) sub-Gaussian with variance parameter*[15]

$$\nu^2 = \mathbb{V}_t\big[(\boldsymbol{z}^\top X)^2\,|\,c_t\big] \le \frac{\lambda^2}{8\log(4c_t)} \quad \forall t\,.$$

Then

$$\mathrm{TCB}_{i,t}(\boldsymbol{x}) \le \widetilde{\mathrm{TCB}}_{i,t}$$

[14] Also observe that $2nd$ has been replaced by $2^{m+1}d$ inside the log's.

[15] Random variable $(\boldsymbol{z}^\top X)^2$ is conditionally sub-Gaussian with variance parameter $\sigma^2 > 0$ when $\mathbb{E}_t\big[\exp(\gamma\,(\boldsymbol{z}^\top X)^2)|\,c_t\big] \le \exp(\sigma^2\,\gamma^2/2)$ for all $\gamma \in \mathbb{R}$. The sub-Gaussian assumption can be removed here at the cost of assuming the conditional variance of $(\boldsymbol{z}^\top X)^2$ scales with c_t like $\frac{\lambda^2}{c_t}$, instead of $\frac{\lambda^2}{\log(c_t)}$.

holds with probability at least $1 - \delta/2$*, uniformly over* $i \in V$*,* $t = 0, 1, 2 \ldots$*, and* $\boldsymbol{x} \in \mathbb{R}^d$ *such that* $||\boldsymbol{x}|| = 1$.

Proof. Fix node $i \in V$ and round t. By the very way the algorithm in Figure 1 is defined, we have

$$M_{i,t} = I + \sum_{s \le t \,:\, i_s = i} \bar{\boldsymbol{x}}_s \bar{\boldsymbol{x}}_s^\top = I + S_{i,t} \,.$$

First, notice that by standard arguments (e.g., [30]) we have

$$\log |M_{i,t}| \le d \, \log(1 + T_{i,t}/d) \le d \, \log(1 + t) \,.$$

Moreover, denoting by $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ the maximal and the minimal eigenvalue of the matrix at argument we have that, for any fixed unit norm $\boldsymbol{x} \in \mathbb{R}^d$,

$$\boldsymbol{x}^\top M_{i,t}^{-1} \boldsymbol{x} \le \lambda_{\max}(M_{i,t}^{-1}) = \frac{1}{1 + \lambda_{\min}(S_{i,t})} \,.$$

Hence, we want to show with probability at least $1 - \delta/(2n)$ that

$$\begin{aligned} \lambda_{\min}(S_{i,t}) \ge \lambda T_{i,t}/4 - 8 \log\left(\frac{T_{i,t} + 3}{\delta/(2nd)}\right) \\ - 2\sqrt{T_{i,t} \log\left(\frac{T_{i,t} + 3}{\delta/(2nd)}\right)} \end{aligned} \tag{2.6}$$

holds for any fixed node i. To this end, fix a unit norm vector $\boldsymbol{z} \in \mathbb{R}^d$, a round $s \le t$, and consider the variable

$$\begin{aligned} V_s &= \boldsymbol{z}^\top \left(\bar{\boldsymbol{x}}_s \bar{\boldsymbol{x}}_s^\top - \mathbb{E}_s[\bar{\boldsymbol{x}}_s \bar{\boldsymbol{x}}_s^\top \,|\, c_s]\right) \boldsymbol{z} \\ &= (\boldsymbol{z}^\top \bar{\boldsymbol{x}}_s)^2 - \mathbb{E}_s[(\boldsymbol{z}^\top \bar{\boldsymbol{x}}_s)^2 \,|\, c_s] \,. \end{aligned}$$

The sequence $V_1, V_2, \ldots, V_{T_{i,t}}$ is a martingale difference sequence, with optional skipping, where $T_{i,t}$ is a stopping time.[16] Moreover, the following claim holds.

Claim 1. *Under the assumption of this lemma,*

$$\mathbb{E}_s[(\boldsymbol{z}^\top \bar{\boldsymbol{x}}_s)^2 \,|\, c_s] \ge \lambda/4 \,.$$

Proof of claim. Let[17] in round s the context vectors be $C_{i_s} = \{\boldsymbol{x}_{s,1}, \ldots, \boldsymbol{x}_{s,c_s}\}$, and consider the corresponding i.i.d. random variables $Z_i = (\boldsymbol{z}^\top \boldsymbol{x}_{s,i})^2 - \mathbb{E}_s[(\boldsymbol{z}^\top \boldsymbol{x}_{s,i})^2 \,|\, c_s]$, $i = 1, \ldots, c_s$. Since by assumption these variables are (zero-mean) sub-Gaussian, we have that (see, e.g., [77][Ch.2])

$$\mathbb{P}_s\left(Z_i < -a \,|\, c_t\right) \le \mathbb{P}_s\left(|Z_i| > a \,|\, c_t\right) \le 2e^{-a^2/2\nu^2}.$$

[16] More precisely, we are implicitly considering the sequence $\eta_{i,1}V_1, \eta_{i,2}V_2, \ldots, \eta_{i,t}V_t$, where $\eta_{i,s} = 1$ if $i_s = i$, and 0 otherwise, with $T_{i,t} = \sum_{s=1}^t \eta_{i,s}$.

[17] This proof is based on standard arguments, and is reported here for the sake of completeness.

holds for any i, where $\mathbb{P}_s(\cdot)$ is the shorthand for the conditional probability

$$\mathbb{P}\left(\,\cdot\,\middle|\,(i_1, C_{i_1}, a_1), \ldots, (i_{s-1}, C_{i_{s-1}}, a_{s-1}), i_s\,\right)\ .$$

The above implies

$$\mathbb{P}_s\left(\min_{i=1,\ldots,c_s} (\boldsymbol{z}^\top \boldsymbol{x}_{s,i})^2 \geq \lambda - a \,\middle|\, c_t\right) \\ \geq \left(1 - 2e^{-a^2/2\nu^2}\right)^{c_s}\ .$$

Therefore

$$\mathbb{E}_s[(\boldsymbol{z}^\top \bar{\boldsymbol{x}}_s)^2 \,|\, c_s] \geq \mathbb{E}_s\left[\min_{i=1,\ldots,c_s} (\boldsymbol{z}^\top \boldsymbol{x}_{s,i})^2 \,\middle|\, c_s\right] \\ \geq (\lambda - a)\left(1 - 2e^{-a^2/2\nu^2}\right)^{c_s}\ .$$

Since this holds for all $a \in \mathbb{R}$, we set $a = \sqrt{2\nu^2 \log(4c_s)}$ to get $\left(1 - 2e^{-a^2/2\nu^2}\right)^{c_s} = (1 - \frac{1}{2c_s})^{c_s} \geq 1/2$ (because $c_s \geq 1$), and $\lambda - a \geq \lambda/2$ (because of the assumption on ν^2). Putting together concludes the proof of the claim. □

We are now in a position to apply a Freedman-like inequality for matrix martingales due to [83, 101] to the (matrix) martingale difference sequence

$$\mathbb{E}_1[\bar{\boldsymbol{x}}_1 \bar{\boldsymbol{x}}_1^\top \,|\, c_1] - \bar{\boldsymbol{x}}_1 \bar{\boldsymbol{x}}_1^\top,\ \mathbb{E}_2[\bar{\boldsymbol{x}}_2 \bar{\boldsymbol{x}}_2^\top \,|\, c_2] - \bar{\boldsymbol{x}}_2 \bar{\boldsymbol{x}}_2^\top, \ldots$$

with optional skipping. Setting for brevity $X_s = \bar{\boldsymbol{x}}_s \bar{\boldsymbol{x}}_s^\top$, and

$$W_t = \sum_{s \leq t \,:\, i_s = i} \left(\mathbb{E}_s[X_s^2 \,|\, c_s] - \mathbb{E}_s^2[X_s \,|\, c_s]\right)\ ,$$

Theorem 1.2 in [101] implies

$$\mathbb{P}\Big(\exists t \,:\, \lambda_{\min}(S_{i,t}) \leq T_{i,t} \lambda_{\min}(\mathbb{E}_1[X_1 \,|\, c_1]) - a, ||W_t|| \leq \sigma^2\Big) \\ \leq d\, e^{-\frac{a^2/2}{\sigma^2 + 2a/3}}\ . \tag{2.7}$$

where $||W_t||$ denotes the operator norm of matrix W_t.

We apply Claim 1, so that $\lambda_{\min}(\mathbb{E}_1[X_1 \,|\, c_1]) \geq \lambda/4$, and proceed as in, e.g., [22]. We set for brevity $A(x, \delta) = 2 \log \frac{(x+1)(x+3)}{\delta}$, and $f(A, r) = 2A + \sqrt{Ar}$.

We can write

$$
\begin{aligned}
&\mathbb{P}\Big(\exists t\,:\,\lambda_{\min}(S_{i,t}) \le \lambda_{\min}T_{i,t}/4 - f(A(||W_t||,\delta),||W_t||)\Big)\\
&\quad\le \sum_{r=0}^{\infty}\mathbb{P}\Big(\exists t\,:\,\lambda_{\min}(S_{i,t}) \le \lambda_{\min}T_{i,t}/4 - f(A(r,\delta),r),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \lfloor||W_t||\rfloor = r\Big)\\
&\quad\le \sum_{r=0}^{\infty}\mathbb{P}\Big(\exists t\,:\,\lambda_{\min}\left(S_{i,t}\right) \le \lambda_{\min}T_{i,t}/4 - f(A(r,\delta),r),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad ||W_t|| \le r+1\Big)\\
&\quad\le d\sum_{r=0}^{\infty} e^{-\frac{f^2(A(r,\delta),r)/2}{r+1+2f(A(r,\delta),r)/3}}\,,
\end{aligned}
$$

the last inequality deriving from (2.7). Because $f(A,r)$ satisfies $f^2(A,r) \ge Ar + A + \frac{2}{3}f(A,r)A$, we have that the exponent in the last exponential is at least $A(r,\delta)/2$, implying

$$
\sum_{r=0}^{\infty} e^{-A(r,\delta)/2} = \sum_{r=0}^{\infty}\frac{\delta}{(r+1)(r+3)} < \delta
$$

which, in turn, yields

$$
\begin{aligned}
\mathbb{P}\Big(\exists t\,:\,\lambda_{\min}(S_{i,t}) &\le T_{i,t}\lambda_{\min}/4\\
&\qquad - f(A(||W_t||,\delta/d),||W_t||)\Big)\\
&\le \delta\,.
\end{aligned}
$$

Finally, observe that

$$
\begin{aligned}
||W_t|| &\le \sum_{s\le t\,:\,i_s=i} ||\mathbb{E}_s[X_s^2\,|\,c_s]||\\
&= \sum_{s\le t\,:\,i_s=i} ||\mathbb{E}_s[X_s\,|\,c_s]||\\
&\le \sum_{s\le t\,:\,i_s=i} \mathbb{E}_s[||X_s\,|\,c_s||]\\
&\le T_{i,t}\,.
\end{aligned}
$$

Therefore we conclude

$$
\begin{aligned}
\mathbb{P}\Big(\forall t\,:\,\lambda_{\min}(S_{i,t}) &\ge \lambda_{\min}T_{i,t}/4 - f(A(T_{i,t},\delta/d),T_{i,t})\Big)\\
&\ge 1-\delta\,.
\end{aligned}
$$

Stratifying over $i \in V$, replacing δ by $\delta/(2n)$ in the last inequality, and overapproximating proves the lemma. □

Lemma 4. *Under the same assumptions as in Lemma 3, we have*

$$||\boldsymbol{u}_i - \boldsymbol{w}_{i,t}|| \le \widetilde{\mathrm{TCB}}_{i,t}$$

holds with probability at least $1-\delta$*, uniformly over* $i \in V$*, and* $t = 0, 1, 2, \ldots$.

Proof. From [1] it follows that

$$|\boldsymbol{u}_i^\top \boldsymbol{x} - \boldsymbol{w}_{i,t}^\top \boldsymbol{x}| \le \mathrm{TCB}_{i,t}(\boldsymbol{x})$$

holds with probability at least $1-\delta/2$, uniformly over $i \in V$, $t = 0, 1, 2, \ldots$. and $\boldsymbol{x} \in \mathbb{R}^d$. Hence,

$$\begin{aligned} ||\boldsymbol{u}_i - \boldsymbol{w}_{i,t}|| &\le \max_{\boldsymbol{x} \in \mathbb{R}^d \,:\, ||\boldsymbol{x}||=1} |\boldsymbol{u}_i^\top \boldsymbol{x} - \boldsymbol{w}_{i,t}^\top \boldsymbol{x}| \\ &\le \max_{\boldsymbol{x} \in \mathbb{R}^d \,:\, ||\boldsymbol{x}||=1} \mathrm{TCB}_{i,t}(\boldsymbol{x}) \\ &\le \widetilde{\mathrm{TCB}}_{i,t}\ , \end{aligned}$$

the last inequality holding with probability $\ge 1-\delta/2$ by Lemma 3. This concludes the proof. □

Lemma 5. *Under the same assumptions as in Lemma 3:*

1. *If* $||\boldsymbol{u}_i - \boldsymbol{u}_j|| \ge \gamma$ *and* $\widetilde{\mathrm{TCB}}_{i,t} + \widetilde{\mathrm{TCB}}_{j,t} < \gamma/2$ *then*

   $$||\boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t}|| > \widetilde{\mathrm{TCB}}_{i,t} + \widetilde{\mathrm{TCB}}_{j,t}$$

   holds with probability at least $1-\delta$*, uniformly over* $i, j \in V$ *and* $t = 0, 1, 2, \ldots$*;*

2. *if* $||\boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t}|| > \widetilde{\mathrm{TCB}}_{i,t} + \widetilde{\mathrm{TCB}}_{j,t}$ *then*

   $$||\boldsymbol{u}_i - \boldsymbol{u}_j|| \ge \gamma$$

   holds with probability at least $1-\delta$*, uniformly over* $i, j \in V$ *and* $t = 0, 1, 2, \ldots$.

Proof. 1. We have

$$\begin{aligned} \gamma &\le ||\boldsymbol{u}_i - \boldsymbol{u}_j|| \\ &= ||\boldsymbol{u}_i - \boldsymbol{w}_{i,t} + \boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t} + \boldsymbol{w}_{j,t} - \boldsymbol{u}_j|| \\ &\le ||\boldsymbol{u}_i - \boldsymbol{w}_{i,t}|| + ||\boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t}|| + ||\boldsymbol{w}_{j,t} - \boldsymbol{u}_j|| \\ &\le \widetilde{\mathrm{TCB}}_{i,t} + ||\boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t}|| + \widetilde{\mathrm{TCB}}_{j,t} \\ &\quad \text{(from Lemma 4)} \\ &\le ||\boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t}|| + \gamma/2, \end{aligned}$$

i.e., $||\boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t}|| \ge \gamma/2 > \widetilde{\mathrm{TCB}}_{i,t} + \widetilde{\mathrm{TCB}}_{j,t}$.

2. Similarly, we have

$$\begin{aligned}\widetilde{\text{TCB}}_{i,t} + \widetilde{\text{TCB}}_{j,t} &< ||\boldsymbol{w}_{i,t} - \boldsymbol{w}_{j,t}|| \\ &\le ||\boldsymbol{u}_i - \boldsymbol{w}_{i,t}|| + ||\boldsymbol{u}_i - \boldsymbol{u}_j|| \\ &\quad + ||\boldsymbol{w}_{j,t} - \boldsymbol{u}_j|| \\ &\le \widetilde{\text{TCB}}_{i,t} + ||\boldsymbol{u}_i - \boldsymbol{u}_j|| + \widetilde{\text{TCB}}_{j,t} \ ,\end{aligned}$$

implying $||\boldsymbol{u}_i - \boldsymbol{u}_j|| > 0$. By the well-separatedness assumption, it must be the case that $||\boldsymbol{u}_i - \boldsymbol{u}_j|| \ge \gamma$.

□

From Lemma 5, it follows that if any two nodes i and j belong to different true clusters and the upper confidence bounds $\widetilde{\text{TCB}}_{i,t}$ and $\widetilde{\text{TCB}}_{j,t}$ are both small enough, then it is very likely that edge (i, j) will get deleted by the algorithm (Lemma 5, Item 1). Conversely, if the algorithm deletes an edge (i, j), then it is very likely that the two involved nodes i and j belong to different true clusters (Lemma 5, Item 2). Notice that, we have $E \subseteq E_t$ with high probability for all t. Because the clusters $\hat{V}_{1,t}, \dots, \hat{V}_{m_t,t}$ are induced by the connected components of $G_t = (V, E_t)$, every true cluster V_i must be entirely included (with high probability) in some cluster $\hat{V}_{j,t}$. Said differently, for all rounds t, the partition of V produced by $V_1, \dots, V_m$ is likely to be a refinement of the one produced by $\hat{V}_{1,t}, \dots, \hat{V}_{m_t,t}$ (in passing, this also shows that, with high probability, $m_t \le m$ for all t). This is a key property to all our analysis. See Figure 2 in the main text for reference.

Lemma 6. *Under the same assumptions as in Lemma 3, if $\widehat{j}_t$ is the index of the current cluster node i_t belongs to, then we have*

$$\text{TCB}_{\widehat{j}_t,t-1}(\boldsymbol{x}) \le \widetilde{\text{TCB}}_{\widehat{j}_t,t-1}$$

holds with probability at least $1 - \delta/2$, uniformly over all rounds $t = 1, 2, \dots$, and $\boldsymbol{x} \in \mathbb{R}^d$ such that $||\boldsymbol{x}|| = 1$.

Proof. The proof is the same as the one of Lemma 3, except that at the very end we need to stratify over all possible shapes for cluster $\hat{V}_{\widehat{j}_t,t}$, rather than over the n nodes. Now, since with high probability (Lemma 5), $\hat{V}_{\widehat{j}_t,t}$ is the union of true clusters, the set of all such unions is with the same probability upper bounded by 2^m. □

The next lemma is a generalization of Theorem 1 in [1], and shows a convergence result for aggregate vector $\bar{\boldsymbol{w}}_{j,t-1}$.

Lemma 7. *Let t be any round, and assume the partition of V produced by true clusters $V_1, \dots, V_m$ is a refinement of the one produced by the current clusters $\hat{V}_{1,t}, \dots, \hat{V}_{m_t,t}$. Let $j = \widehat{j}_t$ be the index of the current cluster node i_t belongs* *to. Let this cluster be the union of true clusters $V_{j_1}, V_{j_2}, \ldots, V_{j_k}$, associated with (distinct) parameter vectors $\boldsymbol{u}_{j_1}, \boldsymbol{u}_{j_2}, \ldots, \boldsymbol{u}_{j_k}$, respectively. Define*

$$\bar{\boldsymbol{u}}_t = \bar{M}_{j,t-1}^{-1} \left(\sum_{\ell=1}^{k} \left(\frac{1}{k} I + \sum_{i \in V_{j_\ell}} (M_{i,t-1} - I) \right) \boldsymbol{u}_{j_\ell} \right) .$$

Then:

1. *Under the same assumptions as in Lemma 3,*

$$||\bar{\boldsymbol{u}}_t - \bar{\boldsymbol{w}}_{j,t-1}|| \le \sqrt{3m}\, \widetilde{\mathrm{TCB}}_{j,t-1}$$

holds with probability at least $1 - \delta$, uniformly over cluster indices $j = 1, \ldots, m_t$, and rounds $t = 1, 2, \ldots$.

2. *For any fixed $\boldsymbol{u} \in \mathbb{R}^d$ we have*

$$||\bar{\boldsymbol{u}}_t - \boldsymbol{u}|| \le 2 \sum_{\ell=1}^{k} ||\boldsymbol{u}_{j_\ell} - \boldsymbol{u}|| \le 2\, SD(\boldsymbol{u}) .$$

Proof. Let $X_{\ell,t-1}$ be the matrix whose columns are the d-dimensional vectors $\bar{\boldsymbol{x}}_s$, for all $s < t \,:\, i_s \in V_{j_\ell}$, $\mathrm{BA}_{\ell,t-1}$ be the column vector collecting all payoffs a_s, $s < t \,:\, i_s \in V_{j_\ell}$, and $\boldsymbol{\eta}_{\ell,t-1}$ be the corresponding column vector of noise values. We have

$$\bar{\boldsymbol{w}}_{j,t-1} = \bar{M}_{j,t-1}^{-1} \bar{\mathbf{b}}_{j,t-1} ,$$

with

$$\begin{aligned}
\bar{\mathbf{b}}_{j,t-1} &= \sum_{\ell=1}^{k} X_{\ell,t-1} \mathrm{BA}_{\ell,t-1} \\
&= \sum_{\ell=1}^{k} X_{\ell,t-1} \left(X_{\ell,t-1}^{\top} \boldsymbol{u}_{j_\ell} + \boldsymbol{\eta}_{\ell,t-1} \right) \\
&= \sum_{\ell=1}^{k} \left(\sum_{i \in V_{j_\ell}} (M_{i,t-1} - I) \boldsymbol{u}_{j_\ell} + X_{\ell,t-1}\, \boldsymbol{\eta}_{\ell,t-1} \right) .
\end{aligned}$$

Thus

$$\bar{\boldsymbol{w}}_{j,t-1} - \bar{\boldsymbol{u}}_t = \bar{M}_{j,t-1}^{-1} \left(\sum_{\ell=1}^{k} \left(X_{\ell,t-1}\, \boldsymbol{\eta}_{\ell,t-1} - \frac{1}{k}\, \boldsymbol{u}_{j_\ell} \right) \right)$$

and, for any fixed $\boldsymbol{x} \in \mathbb{R}^d \, : \, ||\boldsymbol{x}|| = 1$, we have

$$
\begin{aligned}
&\left(\bar{\boldsymbol{w}}_{j,t-1}^\top \boldsymbol{x} - \bar{\boldsymbol{u}}_t^\top \boldsymbol{x}\right)^2 \\
&= \left(\left(\sum_{\ell=1}^k \left(X_{\ell,t-1}\,\boldsymbol{\eta}_{\ell,t-1} - \frac{1}{k}\,\boldsymbol{u}_{j_\ell}\right)\right)^\top \bar{M}_{j,t-1}^{-1}\boldsymbol{x}\right)^2 \\
&\leq \boldsymbol{x}^\top \bar{M}_{j,t-1}^{-1}\boldsymbol{x}\,\left(\sum_{\ell=1}^k \left(X_{\ell,t-1}\,\boldsymbol{\eta}_{\ell,t-1} - \frac{1}{k}\,\boldsymbol{u}_{j_\ell}\right)\right)^\top \bar{M}_{j,t-1}^{-1} \\
&\quad\times \left(\sum_{\ell=1}^k \left(X_{\ell,t-1}\,\boldsymbol{\eta}_{\ell,t-1} - \frac{1}{k}\,\boldsymbol{u}_{j_\ell}\right)\right) \\
&\leq 2\,\boldsymbol{x}^\top \bar{M}_{j,t-1}^{-1}\boldsymbol{x} \\
&\quad\times \left(\Big(\sum_{\ell=1}^k X_{\ell,t-1}\,\boldsymbol{\eta}_{\ell,t-1}\Big)^\top \bar{M}_{j,t-1}^{-1}\Big(\sum_{\ell=1}^k X_{\ell,t-1}\,\boldsymbol{\eta}_{\ell,t-1}\Big)\right. \\
&\qquad\qquad\qquad \left. + \frac{1}{k^2}\Big(\sum_{\ell=1}^k \boldsymbol{u}_{j_\ell}\Big)^\top \bar{M}_{j,t-1}^{-1}\Big(\sum_{\ell=1}^k \boldsymbol{u}_{j_\ell}\Big)\right) \\
&\quad\text{(using } (a+b)^2 \leq 2a^2 + 2b^2\text{)}\,.
\end{aligned}
$$

We focus on the two terms inside the big braces. Because $\hat{V}_{j,t}$ is made up of the union of true clusters, we can stratify over the set of all such unions (which are at most 2^m with high probability), and then apply the martingale result in [1] (Theorem 1 therein), showing that

$$
\begin{aligned}
\left(\sum_{\ell=1}^k X_{\ell,t-1}\,\boldsymbol{\eta}_{\ell,t-1}\right)^\top \bar{M}_{j,t-1}^{-1}\left(\sum_{\ell=1}^k X_{\ell,t-1}\,\boldsymbol{\eta}_{\ell,t-1}\right)& \\
\leq 2\,\sigma^2\left(\log\frac{|\bar{M}_{j,t-1}|}{\delta/2^{m+1}}\right)&
\end{aligned}
$$

holds with probability at least $1 - \delta/2$. As for the second term, we simply write

$$
\frac{1}{k^2}\left(\sum_{\ell=1}^k \boldsymbol{u}_{j_\ell}\right)^\top \bar{M}_{j,t-1}^{-1}\left(\sum_{\ell=1}^k \boldsymbol{u}_{j_\ell}\right) \leq \frac{1}{k^2}\,\Big|\Big|\sum_{\ell=1}^k \boldsymbol{u}_{j_\ell}\Big|\Big|^2 \leq 1\,.
$$

Putting together and overapproximating we conclude that

$$
|\bar{\boldsymbol{w}}_{j,t-1}^\top \boldsymbol{x} - \bar{\boldsymbol{u}}_t^\top \boldsymbol{x}| \leq \sqrt{3m}\,\mathrm{TCB}_{j,t-1}(\boldsymbol{x})
$$

and, since this holds for all unit-norm $\boldsymbol{x}$, Lemma 6 yields

$$
||\bar{\boldsymbol{w}}_{j,t-1} - \bar{\boldsymbol{u}}_t|| \leq \sqrt{3m}\,\widetilde{\mathrm{TCB}}_{j,t-1}\,,
$$

thereby concluding the proof of part 1.

As for part 2, because

$$\bar{M}_{j,t-1} = I + \sum_{\ell=1}^{k} \sum_{i \in V_{j_\ell}} (M_{i,t-1} - I) \,,$$

we can rewrite $\boldsymbol{u}$ as

$$\boldsymbol{u} = \bar{M}_{j,t-1}^{-1} \left(\boldsymbol{u} + \sum_{\ell=1}^{k} \sum_{i \in V_{j_\ell}} (M_{i,t-1} - I)\boldsymbol{u} \right) \,,$$

so that

$$\begin{aligned}\bar{\boldsymbol{u}}_t - \boldsymbol{u} = \bar{M}_{j,t-1}^{-1} \Bigg(& \frac{1}{k} \sum_{\ell=1}^{k} (\boldsymbol{u}_{j_\ell} - \boldsymbol{u}) \\ & + \sum_{\ell=1}^{k} \sum_{i \in V_{j_\ell}} (M_{i,t-1} - I)\,(\boldsymbol{u}_{j_\ell} - \boldsymbol{u}) \Bigg) \,.\end{aligned}$$

Hence

$$\begin{aligned} ||\bar{\boldsymbol{u}}_t - \boldsymbol{u}|| &\le \frac{1}{k} \left|\left| \bar{M}_{j,t-1}^{-1} \sum_{\ell=1}^{k} (\boldsymbol{u}_{j_\ell} - \boldsymbol{u}) \right|\right| \\ &\quad + \sum_{\ell=1}^{k} \left|\left| \bar{M}_{j,t-1}^{-1} \sum_{i \in V_{j_\ell}} (M_{i,t-1} - I)\,(\boldsymbol{u}_{j_\ell} - \boldsymbol{u}) \right|\right| \\ &\le \frac{1}{k} \sum_{\ell=1}^{k} ||\boldsymbol{u}_{j_\ell} - \boldsymbol{u})|| + \sum_{\ell=1}^{k} ||\boldsymbol{u}_{j_\ell} - \boldsymbol{u}|| \\ &\le 2 \sum_{\ell=1}^{k} ||\boldsymbol{u}_{j_\ell} - \boldsymbol{u}|| \,, \end{aligned}$$

as claimed. □

The next lemma gives sufficient conditions on $T_{i,t}$ (or on $\bar{T}_{j,t}$) to insure that $\widetilde{\text{TCB}}_{i,t}$ (or $\widetilde{\text{TCB}}_{j,t}$) is small. We state the lemma for $\widetilde{\text{TCB}}_{i,t}$, but the very same statement clearly holds when we replace $\widetilde{\text{TCB}}_{i,t}$ by $\widetilde{\text{TCB}}_{j,t}$, $T_{i,t}$ by $\bar{T}_{j,t}$, and n by 2^m.

Lemma 8. *The following properties hold for upper confidence bound* $\widetilde{\text{TCB}}_{i,t}$*:*

1. $\widetilde{\text{TCB}}_{i,t}$ *is nonincreasing in* $T_{i,t}$*;*

2. *Let* $A = \sigma\sqrt{2d\,\log(1+t) + 2\log(2/\delta)} + 1$. *Then*

$$\widetilde{\text{TCB}}_{i,t} \le \frac{A}{\sqrt{1+\lambda T_{i,t}/8}}$$

when

$$T_{i,t} \ge \frac{2\cdot 32^2}{\lambda^2}\log\left(\frac{2nd}{\delta}\right)\log\left(\frac{32^2}{\lambda^2}\log\left(\frac{2nd}{\delta}\right)\right)\ ;$$

3. *We have*

$$\widetilde{\text{TCB}}_{i,t} \le \gamma/4$$

when

$$\begin{aligned} T_{i,t} \ge & \frac{32}{\lambda}\max\Bigg\{\frac{A^2}{\gamma^2}, \frac{64}{\lambda}\log\left(\frac{2nd}{\delta}\right) \\ & \qquad\qquad \times \log\left(\frac{32^2}{\lambda^2}\log\left(\frac{2nd}{\delta}\right)\right)\Bigg\}\,. \end{aligned}$$

Proof. The proof follows from simple but annoying calculations, and is therefore omitted. □

We are now ready to combine all previous lemmas into the proof of Theorem 1.

Proof. Let t be a generic round, $\widehat{j}_t$ be the index of the current cluster node i_t belongs to, and j_t be the index of the *true* cluster i_t belongs to. Also, let us define the aggregate vector $\bar{\boldsymbol{w}}_{j_t,t-1}$ as follows :

$$\begin{aligned} \bar{\boldsymbol{w}}_{j_t,t-1} &= \bar{M}^{-1}_{j_t,t-1}\bar{\mathbf{b}}_{j_t,t-1}, \\ \bar{M}_{j_t,t-1} &= I + \sum_{i\in V_{j_t}}(M_{i,t-1} - I), \\ \bar{\mathbf{b}}_{j_t,t-1} &= \sum_{i\in V_{j_t}}\mathbf{b}_{i,t-1}\ . \end{aligned}$$

Assume Lemma 5 holds, implying that the current cluster $\hat{V}_{\widehat{j}_t,t}$ is the (disjoint) union of true clusters, and define the aggregate vector $\bar{\boldsymbol{u}}_t$ accordingly, as in the statement of Lemma 7. Notice that $\bar{\boldsymbol{w}}_{j_t,t-1}$ is the true cluster counterpart to $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$, that is, $\bar{\boldsymbol{w}}_{j_t,t-1} = \bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ if $V_{j_t} = \hat{V}_{\widehat{j}_t,t}$. Also, observe that $\bar{\boldsymbol{u}}_t = \boldsymbol{u}_{i_t}$ when $V_{j_t} = \hat{V}_{\widehat{j}_t,t}$. Finally, set for brevity

$$\boldsymbol{x}_t^* = \operatorname*{argmax}_{\boldsymbol{x}\in C_{i_t}} \boldsymbol{u}_{i_t}^\top \boldsymbol{x}$$

We can rewrite the time-t regret r_t as follows:

$$\begin{aligned} r_t &= \boldsymbol{u}_{i_t}^\top \boldsymbol{x}_t^* - \boldsymbol{u}_{i_t}^\top \bar{\boldsymbol{x}}_t \\ &= \boldsymbol{u}_{i_t}^\top \boldsymbol{x}_t^* - \bar{\boldsymbol{w}}_{j_t,t-1}^\top \boldsymbol{x}_t^* + \bar{\boldsymbol{w}}_{j_t,t-1}^\top \boldsymbol{x}_t^* - \bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^\top \boldsymbol{x}_t^* \\ &\quad + \bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^\top \boldsymbol{x}_t^* - \bar{\boldsymbol{w}}_{j_t,t-1}^\top \bar{\boldsymbol{x}}_t + \bar{\boldsymbol{w}}_{j_t,t-1}^\top \bar{\boldsymbol{x}}_t - \boldsymbol{u}_{i_t}^\top \bar{\boldsymbol{x}}_t \,. \end{aligned}$$

Combined with

$$\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^\top \boldsymbol{x}_t^* + \text{TCB}_{\widehat{j}_t,t-1}(\boldsymbol{x}_t^*) \le \bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^\top \bar{\boldsymbol{x}}_t + \text{TCB}_{\widehat{j}_t,t-1}(\bar{\boldsymbol{x}}_t),$$

and rearranging gives

$$\begin{aligned} r_t &\le \boldsymbol{u}_{i_t}^\top \boldsymbol{x}_t^* - \bar{\boldsymbol{w}}_{j_t,t-1}^\top \boldsymbol{x}_t^* - \text{TCB}_{\widehat{j}_t,t-1}(\boldsymbol{x}_t^*) && (2.8) \\ &\quad + \bar{\boldsymbol{w}}_{j_t,t-1}^\top \bar{\boldsymbol{x}}_t - \boldsymbol{u}_{i_t}^\top \bar{\boldsymbol{x}}_t + \text{TCB}_{\widehat{j}_t,t-1}(\bar{\boldsymbol{x}}_t) && (2.9) \\ &\quad + (\bar{\boldsymbol{w}}_{j_t,t-1} - \bar{\boldsymbol{w}}_{\widehat{j}_t,t-1})^\top (\boldsymbol{x}_t^* - \bar{\boldsymbol{x}}_t) \,. && (2.10) \end{aligned}$$

We continue by bounding with high probability the three terms (2.8), (2.9), and (2.10).

As for (2.8), and (2.9), we simply observe that Lemma 4 allows[18] us to write

$$\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_t^* - \bar{\boldsymbol{w}}_{j_t,t-1}^\top \boldsymbol{x}_t^* \le ||\boldsymbol{u}_{i_t} - \bar{\boldsymbol{w}}_{j_t,t-1}|| \le \widetilde{\text{TCB}}_{j_t,t-1} \,,$$

and

$$\bar{\boldsymbol{w}}_{j_t,t-1}^\top \bar{\boldsymbol{x}}_t - \boldsymbol{u}_{i_t}^\top \bar{\boldsymbol{x}}_t \le ||\boldsymbol{u}_{i_t} - \bar{\boldsymbol{w}}_{j_t,t-1}|| \le \widetilde{\text{TCB}}_{j_t,t-1} \,.$$

Moreover,

$$\begin{aligned} \text{TCB}_{\widehat{j}_t,t-1}(\bar{\boldsymbol{x}}_t) &\le \widetilde{\text{TCB}}_{\widehat{j}_t,t-1} \\ &\qquad \text{(by Lemma 6)} \\ &\le \widetilde{\text{TCB}}_{j_t,t-1} \\ &\qquad \text{(by Lemma 5 and the definition of } \widehat{j}_t). \end{aligned}$$

Hence,

$$(2.8) + (2.9) \le 3\widetilde{\text{TCB}}_{j_t,t-1} \qquad (2.11)$$

holds with probability at least $1 - 2\delta$, uniformly over t.

As for (2.10), letting $\{\cdot\}$ be the indicator function of the predicate at argument,

[18] This lemma applies here since, by definition, $\bar{\boldsymbol{w}}_{j_t,t-1}$ is built only from payoffs from nodes in V_{j_t}, sharing the common unknown vector $\boldsymbol{u}_{i_t}$.

we can write

$$\begin{aligned}
&(\bar{\boldsymbol{w}}_{j_t,t-1} - \bar{\boldsymbol{w}}_{\widehat{j}_t,t-1})^\top(\boldsymbol{x}_t^* - \bar{\boldsymbol{x}}_t)\\
&\quad = (\bar{\boldsymbol{w}}_{j_t,t-1} - \boldsymbol{u}_{i_t})^\top(\boldsymbol{x}_t^* - \bar{\boldsymbol{x}}_t) + (\boldsymbol{u}_{i_t} - \bar{\boldsymbol{u}}_t)^\top(\boldsymbol{x}_t^* - \bar{\boldsymbol{x}}_t)\\
&\qquad + (\bar{\boldsymbol{u}}_t - \bar{\boldsymbol{w}}_{\widehat{j}_t,t-1})^\top(\boldsymbol{x}_t^* - \bar{\boldsymbol{x}}_t)\\
&\quad \le 2\,\widetilde{\mathrm{TCB}}_{j_t,t-1} + 2\,||\boldsymbol{u}_{i_t} - \bar{\boldsymbol{u}}_t|| + 2\sqrt{3m}\,\widetilde{\mathrm{TCB}}_{\widehat{j}_t,t-1}\\
&\qquad \text{(using Lemma 4, } ||\boldsymbol{x}_t^* - \bar{\boldsymbol{x}}_t|| \le 2\text{, and Lemma 7, part 1)}\\
&\quad = 2\,\widetilde{\mathrm{TCB}}_{j_t,t-1} + 2\,\{V_{j_t} \ne \hat{V}_{\widehat{j}_t,t}\}\,||\boldsymbol{u}_{i_t} - \bar{\boldsymbol{u}}_t||\\
&\qquad\qquad + 2\sqrt{3m}\,\widetilde{\mathrm{TCB}}_{\widehat{j}_t,t-1}\\
&\quad \le 2(1+\sqrt{3m})\,\widetilde{\mathrm{TCB}}_{j_t,t-1} + 4\,\{V_{j_t} \ne \hat{V}_{\widehat{j}_t,t}\}\,SD(\boldsymbol{u}_{i_t})\\
&\qquad \text{(by Lemma 5, and Lemma 7, part 2)}\ .
\end{aligned}$$

Piecing together we have so far obtained

$$\begin{aligned}
r_t &\le (5 + 2\sqrt{3m})\,\widetilde{\mathrm{TCB}}_{j_t,t-1}\\
&\quad + 4\,\{V_{j_t} \ne \hat{V}_{\widehat{j}_t,t}\}\,SD(\boldsymbol{u}_{i_t})\ .
\end{aligned} \tag{2.12}$$

We continue by bounding $\{V_{j_t} \ne \hat{V}_{\widehat{j}_t,t}\}$. From Lemma 5, we clearly have

$$\begin{aligned}
&\{V_{j_t} \ne \hat{V}_{\widehat{j}_t,t}\}\\
&\quad \le \{\exists i \in V_{j_t}, \exists j \notin V_{j_t}\ :\ (i,j) \in E_t\}\\
&\quad \le \Big\{\exists i \in V_{j_t}, \exists j \notin V_{j_t}\ :\ \forall s < t\big((i_s \ne i)\\
&\qquad \vee (i_s = i, ||\boldsymbol{w}_{i,s-1} + \boldsymbol{w}_{j,s-1}|| \le \widetilde{\mathrm{TCB}}_{i,s-1} + \widetilde{\mathrm{TCB}}_{j,s-1})\big)\Big\}\\
&\quad \le \{\exists i \in V_{j_t}\ :\ \forall s < t\ i_s \ne i\}\\
&\qquad + \Big\{\exists i \in V_{j_t}, \exists j \notin V_{j_t}\ :\\
&\qquad\quad \forall s < t\ ||\boldsymbol{w}_{i,s-1} + \boldsymbol{w}_{j,s-1}|| \le \widetilde{\mathrm{TCB}}_{i,s-1} + \widetilde{\mathrm{TCB}}_{j,s-1}\Big\}\\
&\quad \le \{\exists i \in V_{j_t}\ :\ \forall s < t\ i_s \ne i\}\\
&\qquad + \{\exists i \in V_{j_t}, \exists j \notin V_{j_t}\ :\\
&\qquad\qquad\qquad \forall s < t\ \widetilde{\mathrm{TCB}}_{i,s-1} + \widetilde{\mathrm{TCB}}_{j,s-1} \ge \gamma/2\}\\
&\quad \le \{\exists i \in V_{j_t}\ :\ \forall s < t\ i_s \ne i\}\\
&\qquad + \{\exists i \in V\ :\ \forall s < t\ \widetilde{\mathrm{TCB}}_{i,s-1} \ge \gamma/4\}\ .
\end{aligned}$$

At this point, we apply Lemma 8 to $\widetilde{\mathrm{TCB}}_{i,t}$ with

$$\begin{aligned}
A^2 &= \Big(\sigma\sqrt{2d\,\log(1+T) + 2\log(2/\delta)} + 1\Big)^2\\
&\le 4\sigma^2(d\,\log(1+T) + \log(2/\delta)) + 2,
\end{aligned}$$

and set for brevity

$$\begin{aligned}
B &= \frac{32}{\lambda}\,\max\left\{\frac{A^2}{\gamma^2}, \frac{64}{\lambda}\,\log\left(\frac{2nd}{\delta}\right)\right.\\
&\qquad\qquad\left.\times \log\left(\frac{32^2}{\lambda^2}\log\left(\frac{2nd}{\delta}\right)\right)\right\},\\
C &= \frac{2\cdot 32^2}{\lambda^2}\,\log\left(\frac{2^{m+1}d}{\delta}\right)\,\log\left(\frac{32^2}{\lambda^2}\log\left(\frac{2^{m+1}d}{\delta}\right)\right)\,.
\end{aligned}$$

We can write

$$\begin{aligned}
&\{\exists i \in V \,:\, \forall s < t\ \widetilde{\mathrm{TCB}}_{i,s-1} \ge \gamma/4\}\\
&\quad \le \{\exists i \in V \,:\, \widetilde{\mathrm{TCB}}_{i,t-2} \ge \gamma/4\}\\
&\quad \le \{\exists i \in V \,:\, T_{i,t-2} \le B\}\,.
\end{aligned}$$

Moreover,

$$\begin{aligned}
&\{\exists i \in V_{j_t} \,:\, \forall s < t\ i_s \neq i\}\\
&\quad \le \{\exists i \in V_{j_t} \setminus \{i_t\} \,:\, T_{i,t-1} = 0\}\\
&\quad \le \{\exists i \in V \,:\, T_{i,t-1} = 0\}\,.
\end{aligned}$$

That is,

$$\begin{aligned}
\{V_{j_t} \neq \hat{V}_{\hat{j}_t,t}\} &\le \{\exists i \in V \,:\, T_{i,t-2} \le B\}\\
&\qquad + \{\exists i \in V \,:\, T_{i,t-1} = 0\}\,.
\end{aligned}$$

Further, using again Lemma 8 (applied this time to $\widetilde{\mathrm{TCB}}_{j,t}$) combined with the fact that $\widetilde{\mathrm{TCB}}_{j,t} \le A$ for all j and t, we have

$$\widetilde{\mathrm{TCB}}_{j_t,t-1} = A\,\{\bar{T}_{j_t,t-1} < C\} + \frac{A}{\sqrt{1+\lambda\,\bar{T}_{j_t,t-1}/8}}\,,$$

where

$$\bar{T}_{j_t,t-1} = \sum_{i\in V_{j_t}} T_{i,t-1} = |\{s \le t-1 \,:\, i_s \in V_{j_t}\}|\,.$$

Putting together as in (2.12), and summing over $t = 1,\ldots,T$, we have shown so far that with probability at least $1 - 7\delta/2$,

$$\begin{aligned}\sum_{t=1}^{T} r_t \leq (5+2\sqrt{3m})A &\sum_{t=1}^{T}\{\bar{T}_{j_t,t-1} < C\}\\ &+ (5+2\sqrt{3m})A \sum_{t=1}^{T} \frac{1}{\sqrt{1+\lambda \bar{T}_{j_t,t-1}/8}}\\ &+ 4\sum_{t=1}^{T} SD(\boldsymbol{u}_{i_t})\ \{\exists i \in V\ :\ T_{i,t-2} \leq B\}\\ &+ 4\sum_{t=1}^{T} SD(\boldsymbol{u}_{i_t})\,\{\exists i \in V\ :\ T_{i,t-1} = 0\}\ ,\end{aligned}$$

with $T_{i,t} = 0$ if $t \leq 0$.

We continue by upper bounding with high probability the four terms in the right-hand side of the last inequality. First, observe that for any fixed i and t, $T_{i,t}$ is a binomial random variable with parameters t and $1/n$, and $\bar{T}_{j_t,t-1} = \sum_{i\in V_{j_t}} T_{i,t-1}$ which, for fixed i_t, is again binomial with parameters t, and $\frac{v_{j_t}}{n}$, where v_{j_t} is the size of the true cluster i_t falls into. Moreover, for any fixed t, the variables $T_{i,t}$, $i \in V$ are indepedent.

To bound the third term, we use a standard Bernstein inequality twice: first, we apply it to sequences of independent Bernoulli variables, whose sum $T_{i,t-2}$ has average $\mathbb{E}[T_{i,t-2}] = \frac{t-2}{n}$ (for $t \geq 3$), and then to the sequence of variables $SD(\boldsymbol{u}_{i_t})$ whose average $\mathbb{E}[SD(\boldsymbol{u}_{i_t})] = \frac{1}{n}\sum_{i\in V} SD(\boldsymbol{u}_i)$ is over the random choice of i_t.

Setting for brevity

$$D(B) = 2n\left(B + \frac{5}{3}\log(Tn/\delta)\right) + 2,$$

where B has been defined before, we can write

$$\begin{aligned}&\sum_{t=1}^{T} SD(\boldsymbol{u}_{i_t})\ \{\exists i \in V\ :\ T_{i,t-2} \leq B\}\\ &\quad= \sum_{t\leq D(B)} SD(\boldsymbol{u}_{i_t})\ \{\exists i \in V\ :\ T_{i,t-2} \leq B\}\\ &\qquad+ \sum_{t> D(B)} SD(\boldsymbol{u}_{i_t})\ \{\exists i \in V\ :\ T_{i,t-2} \leq B\}\\ &\quad\leq \sum_{t\leq D(B)} SD(\boldsymbol{u}_{i_t})\\ &\qquad+ m \sum_{t> D(B)} \{\exists i \in V\ :\ T_{i,t-2} \leq B\}\ .\end{aligned}$$

Then from Bernstein's inequality,

$$\mathbb{P}\left(\exists i \in V\, \exists t > D(B)\,:\, T_{i,t-2} \le B\right) \le \delta\,,$$

and

$$\mathbb{P}\left(\sum_{t \le D(B)} SD(\boldsymbol{u}_{i_t}) \ge \frac{3}{2}\, D(B)\, \mathbb{E}[SD(\boldsymbol{u}_{i_t})] + \frac{5}{3}\, m\, \log(1/\delta)\right) \le \delta\,.$$

Thus with probability $\ge 1 - 2\delta$

$$\sum_{t=1}^{T} SD(\boldsymbol{u}_{i_t})\, \{\exists i \in V\,:\, T_{i,t-2} \le B\} \le \frac{3}{2}\, D(B)\, \mathbb{E}[SD(\boldsymbol{u}_{i_t})] + \frac{5}{3}\, m\, \log(1/\delta)\,.$$

Similarly, to bound the fourth term we have, with probability $\ge 1 - 2\delta$,

$$\sum_{t=1}^{T} SD(\boldsymbol{u}_{i_t})\, \{\exists i \in V\,:\, T_{i,t-1} = 0\} \le \frac{3}{2}\, D(0)\, \mathbb{E}[SD(\boldsymbol{u}_{i_t})] + \frac{5}{3}\, m\, \log(1/\delta)\,.$$

Next, we crudely upper bound the first term as

$$(5+2\sqrt{3m})A \sum_{t=1}^{T} \{\bar{T}_{j_t,t-1} < C\} \le (5+2\sqrt{3m})A \sum_{t=1}^{T} \{T_{i_t,t-1} < C\}\,,$$

and then apply a very similar argument as before to show that with probability $\ge 1 - \delta$,

$$\sum_{t=1}^{T} \{T_{i_t,t-1} < C\} \le n\left(C + \frac{5}{3}\log\left(\frac{T}{\delta}\right)\right) + 1\,.$$

Finally, we are left to bound the second term. The following is a simple property of binomial random variables we be useful.

Claim 2. *Let X be a binomial random variable with parameters n and p, and $\lambda \in (0,1)$ be a constant. Then*

$$\mathbb{E}\left[\frac{1}{\sqrt{1+\lambda X}}\right] \le \begin{cases} \frac{3}{\sqrt{1+\lambda\, n\, p}} & \textit{if } np \ge 10\,; \\ 1 & \textit{if } np < 10\,. \end{cases}$$

Proof of claim. The second branch of the inequality is clearly trivial, so we focus on the first one under the assumption $np \geq 10$. Let then $\beta \in (0, 1)$ be a parameter that will be set later on. We have

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{\sqrt{1+\lambda X}}\right] &\leq \mathbb{P}(X \leq (1-\beta)\,n\,p) \\
&\quad + \frac{1}{\sqrt{1+\lambda\,(1-\beta)\,n\,p}}\,\mathbb{P}(X \geq (1-\beta)\,n\,p) \\
&\leq e^{-\beta^2\,n\,p/2} + \frac{1}{\sqrt{1+\lambda\,(1-\beta)\,n\,p}}\,,
\end{aligned}$$

the last inequality following from the standard Chernoff bounds. Setting $\beta = \sqrt{\frac{\log(1+\lambda\,n\,p)}{n\,p}}$ gives

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{\sqrt{1+\lambda X}}\right] &\leq \frac{1}{\sqrt{1+\lambda\,n\,p}} \\
&\quad + \frac{1}{\sqrt{1+\lambda\,(np - \sqrt{np}\log(1+\lambda np))}} \\
&\leq \frac{1}{\sqrt{1+\lambda\,n\,p}} + \frac{1}{\sqrt{1+\lambda\,n\,p/2}} \\
&\quad \text{(using } np \geq 10\text{)} \\
&\leq \frac{3}{\sqrt{1+\lambda\,n\,p}}\,,
\end{aligned}$$

i.e., the claimed inequality □

Now,

$$\mathbb{E}_{t-1}\left[\frac{1}{\sqrt{1+\lambda\,\bar{T}_{j_t,t-1}/8}}\right] = \sum_{j=1}^{m}\frac{v_j}{n}\,\frac{1}{\sqrt{1+\lambda\,\bar{T}_{j,t-1}/8}}\,,$$

being $\bar{T}_{j,t-1} = |\{s < t \,:\, i_s \in V_j\}|$ a binomial variable with parameters $t-1$ and $\frac{v_j}{n}$, where $v_j = |V_j|$. By the standard Hoeffding-Azuma inequality

$$\begin{aligned}
\sum_{t=1}^{T}\frac{1}{\sqrt{1+\lambda\,\bar{T}_{j_t,t-1}/8}} &\leq \sum_{t=1}^{T}\sum_{j=1}^{m}\frac{v_j}{n}\,\frac{1}{\sqrt{1+\lambda\,\bar{T}_{j,t-1}/8}} \\
&\quad + \sqrt{2T\,\log(1/\delta)}
\end{aligned}$$

holds with probability at least $1-\delta$, In turn, from Bernstein's inequality, we have

$$\mathbb{P}\left(\exists t\,\exists j \,:\, \bar{T}_{j,t-1} \leq \frac{t-1}{2n}\,v_j - \frac{5}{3}\,\log(Tm/\delta)\right) \leq \delta\,.$$

Therefore, with probability at least $1 - 2\delta$,

$$
\begin{aligned}
&\sum_{t=1}^{T} \frac{1}{\sqrt{1+\lambda\,\bar{T}_{j_t,t-1}/8}} \\
&\leq \sum_{t=1}^{T}\sum_{j=1}^{m} \frac{v_j}{n}\,\frac{1}{\sqrt{1+\frac{\lambda}{8}\,\left(\frac{t-1}{2n}\,v_j - \frac{5}{3}\,\log(Tm/\delta)\right)_+}} \\
&\quad + \sqrt{2T\,\log(1/\delta)} \\
&\leq \sum_{j=1}^{m}\frac{v_j}{n}\left(4n\,\frac{5}{3}\,\log(Tm/\delta) + 1 + \sum_{t=1}^{T}\frac{1}{\sqrt{1+\frac{\lambda}{8}\,\frac{t-1}{4n}\,v_j}}\right) \\
&\quad + \sqrt{2T\,\log(1/\delta)} \\
&= 4n\,\frac{5}{3}\,\log(Tm/\delta) + 1 + \sum_{j=1}^{m}\frac{v_j}{n}\,\sum_{t=1}^{T}\frac{1}{\sqrt{1+\frac{\lambda}{8}\,\frac{t-1}{4n}\,v_j}} \\
&\quad + \sqrt{2T\,\log(1/\delta)}\,.
\end{aligned}
$$

If we set for brevity $r_j = \frac{\lambda}{8}\,\frac{v_j}{4n}$, $j = 1, \dots, m$, we have

$$
\begin{aligned}
\sum_{t=1}^{T}\frac{1}{\sqrt{1+\frac{\lambda}{8}\,\frac{t-1}{4n}\,v_j}} &\leq \int_0^T \frac{dx}{\sqrt{1+(x-1)r_j}} \\
&= \frac{2}{r_j}\left(\sqrt{1+T\,r_j - r_j} - \sqrt{1-r_j}\right) \\
&\leq 2\sqrt{\frac{T}{r_j}}\,,
\end{aligned}
$$

so that

$$
\begin{aligned}
\sum_{t=1}^{T} \frac{1}{\sqrt{1+\lambda\,\bar{T}_{j_t,t-1}/8}} &\leq 4n\,\frac{5}{3}\,\log(Tm/\delta) + 1 \\
&\quad + \sqrt{2T\,\log(1/\delta)} + 8\sum_{j=1}^{m}\sqrt{\frac{2Tv_j}{\lambda n}}\,.
\end{aligned}
$$

Finally, we put all pieces together. In order for all claims to hold simultaneously with probability at least $1 - \delta$, we need to replace δ throughout by $\delta/10.5$. Then we switch to a $\widetilde{O}$-notation, and overapproximate once more to conclude the proof. $\square$

2.5.2 Implementation

As we said in the main text, in implementing the algorithm in Figure 1, the reader should keep in mind that it is reasonable to expect n (the number of users) to be quite large, d (the number of features of each item) to be relatively small, and m (the number of true clusters) to be very small compared to n. Then the algorithm can be implemented by storing a least-squares estimator $\boldsymbol{w}_{i,t-1}$ at each node $i \in V$, an aggregate least squares estimator $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ for each current cluster $\widehat{j}_t \in \{1, \ldots, m_t\}$, and an extra data-structure which is able to perform decremental dynamic connectivity. Fast implementations of such data-structures are those studied by [100, 55] (see also the research thread referenced therein). In particular, in [100] (Theorem 1.1 therein) it is shown that a randomized construction exists that maintains a spanning forerst which, given an initial undirected graph $G_1 = (V, E_1)$, is able to perform edge deletions and answer connectivity queries of the form "Is node i connected to node j" in expected total time $O\left(\min\{|V|^2, |E_1| \log |V|\} + \sqrt{|V|\,|E_1|}\, \log^{2.5} |V|\right)$ for $|E_1|$ deletions. Connectivity queries and deletions can be interleaved, the former being performed in *constant* time. Notice that when we start off from the full graph, we have $|E_1| = O(|V|^2)$, so that the expected amortized time per query becomes constant. On the other hand, if our initial graph has $|E_1| = O(|V| \log |V|)$ edges, then the expected amortized time per query is $O(\log^2 |V|)$. This becomes $O(\log^{2.5} |V|)$ if the initial graph has $|E_1| = O(|V|)$. In addition, we maintain an n-dimensional vector ClusterIndices containing, for each node $i \in V$, the index j of the current cluster i belongs to.

With these data-structures handy, we can implement our algorithm as follows. After receiving i_t, computing j_t is $O(1)$ (just by accessing ClusterIndices). Then, computing k_t can be done in time $O(d^2)$ (matrix-vector multiplication, executed c_t times, assuming c_t is a constant). Then the algorithm directly updates $\mathbf{b}_{i_t,t-1}$ and $\bar{\mathbf{b}}_{\widehat{j}_t,t-1}$, as well as the inverses of matrices $M_{i_t,t-1}$ and $\bar{M}_{\widehat{j}_t,t-1}$, which is again $O(d^2)$, using standard formulas for rank-one adjustment of inverse matrices. In order to prepare the ground for the subsequent edge deletion phase, it is convenient that the algorithm also stores at each node i matrix $M_{i,t-1}$ (whose time-t update is again $O(d^2)$).

Let DELETE(i, ℓ) and IS-CONNECTED(i, ℓ) be the two operations delivered by the decremental dynamic connectivity data-structure. Edge deletion at time t corresponds to cycling through all nodes ℓ such that (i_t, ℓ) is an existing edge. The number of such edges is on average equal to the average degree of node i_t, which is $O\left(\frac{|E_1|}{n}\right)$, where $|E_1|$ is the number of edges in the initial graph G_1. Now, if (i_t, ℓ) has to be deleted (each the deletion test being $O(d)$), then we invoke DELETE(i_t, ℓ), and then IS-CONNECTED(i_t, ℓ). If IS-CONNECTED(i_t, ℓ) = "no", this means that the current cluster $\hat{V}_{j_t,t-1}$ has to split into two new clusters as a consequence of the deletion of edge (i_t, ℓ). The set of nodes contained in these two clusters correspond to the two sets

$$\{k \in V \,:\, \text{IS-CONNECTED}(i_t, k) = \text{``}yes\text{''}\},$$
$$\{k \in V \,:\, \text{IS-CONNECTED}(\ell, k) = \text{``}yes\text{''}\}\text{`},$$

whose expected amortized computation *per node* is $O(1)$ to $O(\log^{2.5} n)$ (depending on the density of the initial graph G_1). We modify the CLUSTERINDICES vector accordingly, but also the aggregate least squares estimators. This is because $\bar{\boldsymbol{w}}_{\widehat{\jmath}_t,t-1}$ (represented through $\bar{M}^{-1}_{\widehat{\jmath}_t,t}$ and $\bar{\mathbf{b}}_{\widehat{\jmath}_t,t}$) has to be spread over the two new-born clusters. This operation can be performed by adding up all matrices $M_{i,t}$ and all $\mathbf{b}_{i,t}$, over all i belonging to each of the two new clusters (it is at this point that we need to access $M_{i,t}$ for each i), and then inverting the resulting aggregate matrices. This operation takes $O(n\,d^2 + d^3)$. However, as argued in the comments following Lemma 5, with high probability the number of current clusters m_t can never exceed m, so that with the same probability this operation is only performed at most m times throughout the learning process. Hence in T rounds we have an overall (expected) running time

$$O\Biggl(T\,\left(d^2 + \frac{|E_1|}{n}\,d\right) + m\,(n\,d^2 + d^3) + |E_1| \\ + \min\{n^2, |E_1|\,\log n\} + \sqrt{n\,|E_1|}\,\log^{2.5} n\Biggr)\,.$$

Notice that the above is $n \cdot \mathrm{poly}(\log n)$, if so is $|E_1|$. In addition, if T is large compared to n and d, the average running time per round becomes $O(d^2 + d \cdot \mathrm{poly}(\log n))$.

As for memory requirements, we need to store two $d \times d$ matrices and one d-dimensional vector at each node, one $d \times d$ matrix and one d-dimensional vector for each current cluster, vector CLUSTERINDICES, and the data-structures allowing for fast deletion and connectivity tests. Overall, these data-structures do not require more than $O(|E_1|)$ memory to be stored, so that this implementation takes $O(n\,d^2 + m\,d^2 + |E_1|) = O(n\,d^2 + |E_1|)$, where we again relied upon the $m_t \le m$ condition. Again, this is $n \cdot \mathrm{poly}(\log n)$ if so is $|E_1|$.

2.5.3 More Plots

This section contains a more thorough set of comparative plots on the synthetic datasets described in the main text. See Figure 2.6 and Figure 2.7.

2.5.4 Reference Bounds

We now provide a proof sketch of the reference bounds mentioned in Section 2 of the main text.

Let us start off from the *single user* bound for LINUCB (either ONE or IND) one can extract from [1]. Let $\boldsymbol{u}_j \in \mathbb{R}^d$ be the profile vector of this user. Then, with

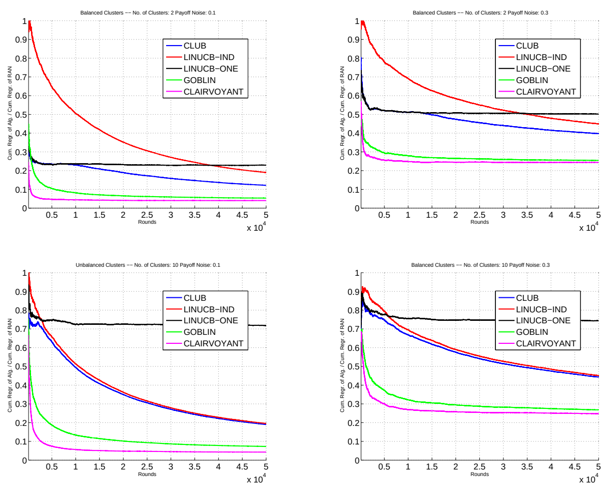


Figure 2.6: Results on synthetic datasets. Each plot displays the behavior of the ratio of the current cumulative regret of the algorithm (“Alg”) to the current cumulative regret of RAN, where where “Alg” is either “CLUB” or “LinUCB-IND” or “LinUCB-ONE” or “GOBLIN”or “CLAIRVOYANT”. The cluster sizes are balanced ($z = 0$). From left to right, payoff noise steps from 0.1 to 0.3, and from top to bottom the number of clusters jumps from 2 to 10.

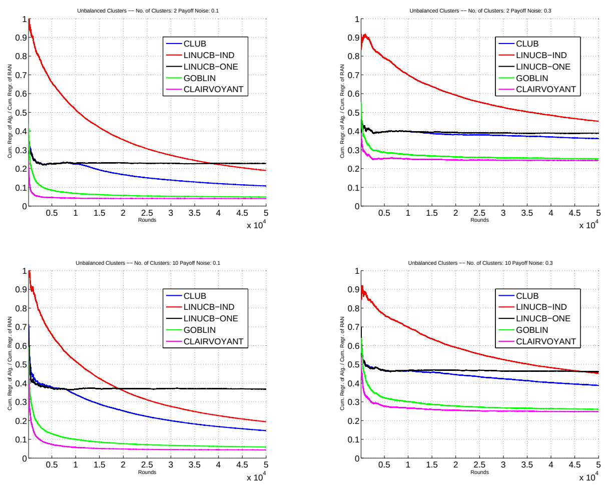


Figure 2.7: Results on synthetic datasets in the case of unbalanced ($z = 2$) cluster sizes. The rest is the same as in Figure 2.6.

probability at least $1 - \delta$, we have

$$\begin{aligned}\sum_{t=1}^{T} r_t &= O\left(\sqrt{T\left(\sigma^2\, d\,\log T + \sigma^2\,\log\frac{1}{\delta} + ||\boldsymbol{u}_i||^2\right)\, d\,\log T}\right)\\ &= \widetilde{O}\left(\sqrt{T\,(\sigma^2\, d + ||\boldsymbol{u}_j||^2)\, d}\right)\\ &= \widetilde{O}\left((\sigma\, d + \sqrt{d})\sqrt{T}\right),\end{aligned}$$

the last line following from assuming $||\boldsymbol{u}_j|| = 1$.

Then, a straightforward way of turning this bound into a bound for the CLEARVOYANT algorithm that knows all clusters $V_1, \ldots, V_m$ ahead of time and runs one instance of LINUCB per cluster is to sum the regret contributed by each cluster throughout the T rounds. Letting $T_{j,T}$ denote the set of rounds t such that $i_t \in V_j$, we can write

$$\sum_{t=1}^{T} r_t = \widetilde{O}\left((\sigma\, d + \sqrt{d}) \sum_{j=1}^{m} \sqrt{T_{j,T}} \right) .$$

However, because i_t is drawn uniformly at random over V, we also have $\mathbb{E}[T_{j,T}] = T\frac{|V_j|}{n}$, so that we essentially have with high probability

$$\sum_{t=1}^{T} r_t = \widetilde{O}\left((\sigma\, d + \sqrt{d})\sqrt{T} \left(1 + \sum_{j=1}^{m} \sqrt{\frac{|V_j|}{n}} \right) \right) ,$$

i.e., Eq. (1) in the main text.

2.5.5 Further Thoughts

As we said in Remark 3, a data-dependent variant of the CLUB algorithm can be designed and analyzed which relies on data-dependent clusterability assumptions of the set of users with respect to a set of context vectors. These data-dependent assumptions allow us to work in a fixed design setting for the sequence of context vectors $\boldsymbol{x}_{t,k}$, and remove the sub-Gaussian and full-rank hypotheses regarding $\mathbb{E}[XX^\top]$. To make this more precise, consider an adversary that generates (unit norm) context vectors in a (possibly adaptive) way that *for all* $\boldsymbol{x}$ so generated $|\boldsymbol{u}_j^\top \boldsymbol{x} - \boldsymbol{u}_{j'}^\top \boldsymbol{x}| \geq \gamma$, whenever $j \neq j'$. In words, the adversary's power is restricted in that it cannot generate two distict context vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ such that $|\boldsymbol{u}_j^\top \boldsymbol{x} - \boldsymbol{u}_{j'}^\top \boldsymbol{x}|$ is small and $|\boldsymbol{u}_j^\top \boldsymbol{x}' - \boldsymbol{u}_{j'}^\top \boldsymbol{x}'|$ is large. The two quantities must either be both zero (when $j = j'$) or both bounded away from 0 (when $j \neq j'$). Under this assumption, one can show that a modification to the $\text{TCB}_{i,t}(\boldsymbol{x})$ and $\text{TCB}_{j,t}(\boldsymbol{x})$ functions exists that makes the CLUB algorithm in Figure 1 achieve a cumulative regret bound similar to the one in (5), where the $\sqrt{\frac{1}{\lambda}}$ factor therein is turned back into $\sqrt{d}$, as in the reference bound (1), but with a worse dependence on the geometry of the set of $\boldsymbol{u}_j$, as compared to $\mathbb{E}[SD(\boldsymbol{u}_{i_t})]$. The analysis goes along the very same lines as the one of Theorem 1.

2.5.6 Related Work

The most closely related papers are [34, 7, 14, 76].

In [7], the authors define a transfer learning problem within a stochastic multi-armed bandit setting, where a prior distribution is defined over the set of possible models over the tasks. More similar in spirit to our paper is the recent work [14] that relies on clustering Markov Decision Processes based on their model parameter similarity. A paper sharing significant similarities with ours, in terms of both setting and technical tools is the very recent paper [76] that came to our attention at the time of writing ours. In that paper, the authors analyze a noncontextual stochastic bandit problem where model parameters can indeed be clustered in a few (unknown) types, thereby requiring the algorithm to learn the clusters rather than learning the parameters in isolation. Yet, the provided algorithmic solutions are completely different from ours. Finally, in [34], the authors work under the assumption that users are defined using a context vector, and try to learn a low-rank subspace under the assumption that variation across users is low-rank. The paper combines low-rank matrix recovery with high-dimensional Gaussian Process Bandits, but it gives rise to algorithms which do not seem easy to use in large scale practical scenarios.

2.5.7 Discussion

This work could be extended along several directions. First, we may rely on a softer notion of clustering than the one we adopted here: a cluster is made up of nodes where the "within distance" between associated profile vectors is smaller than their "between distance". Yet, this is likely to require prior knowledge of either the distance threshold or the number of underlying clusters, which are assumed to be unknown in this paper. Second, it might be possible to handle partially overlapping clusters. Third, CLUB can clearly be modified so as to cluster nodes through off-the-shelf graph clustering techniques (mincut, spectral clustering, etc.). Clustering via connected components has the twofold advantage of being computationally faster and relatively easy to analyze. In fact, we do not know how to analyze CLUB when combined with alternative clustering techniques, and we suspect that Theorem 1 already delivers the sharpest results (as $T \to \infty$) when clustering is indeed based on connected components only. Fourth, from a practical standpoint, it would be important to incorporate further side information, like must-link and cannot-link constraints. Fifth, in recommender systems practice, it is often relevant to provide recommendations to new users, even in the absence of past information (the so-called "cold start" problem). In fact, there is a way of tackling this problem through the machinery we developed here (the idea is to duplicate the newcomer's node as many times as the current clusters are, and then treat each copy as a separate user). This would potentially allow CLUB to work even in the presence of (almost) idle users. We haven't so far collected any experimental evidence on the effectiveness of this strategy. Sixth, following the comments we made in Remark 3, we are trying to see if the i.i.d. and other statistical assumptions we made in Theorem 1 could be removed.

Chapter 3

Decentralized Clustering Bandits

3.1 Introduction

Bandits are a class of classic optimisation problems that are fundamental to several important application areas. The most prominent of these is recommendation systems, and they can also arise more generally in networks (see, e.g., [74, 45]).

We consider settings where a network of agents are trying to solve collaborative linear bandit problems. Sharing experience can improve the performance of both the whole network and each agent simultaneously, while also increasing robustness. However, we want to avoid putting too much strain on communication channels. Communicating every piece of information would just overload these channels. The solution we propose is a gossip-based information sharing protocol which allows information to diffuse across the network at a small cost, while also providing robustness.

Such a set-up would benefit, for example, a small start-up that provides some recommendation system service but has limited resources. Using an architecture that enables the agents (the client's devices) to exchange data between each other directly and to do all the corresponding computations themselves could significantly decrease the infrastructural costs for the company. At the same time, without a central server, communicating all information instantly between agents would demand a lot of bandwidth.

Multi-Agent Linear Bandits In the simplest setting we consider, all the agents are trying to solve the same underlying linear bandit problem. In particular, we have a set of *nodes* V, indexed by i, and representing a finite set of *agents*. At each time, t:

- a set of *actions* (equivalently, the *contexts*) arrives for each agent i, $\mathcal{D}_t^i \subset \mathcal{D}$ and we assume the set $\mathcal{D}$ is a subset of the unit ball in $\mathbb{R}^d$;
- each agent, i, chooses an action (context) $x_t^i \in \mathcal{D}_t^i$, and receives a *reward*

$$r_t^i = (x_t^i)^\intercal \theta + \xi_t^i,$$

where θ is some unknown *coefficient vector*, and ξ_t^i is some zero mean, R-subGaussian noise;

- last, the agents can share information according to some protocol across a communication channel.

We define the *instantaneous regret* at each node i, and, respectively, the *cumulative regret* over the whole network to be:

$$\rho_t^i := \left(x_t^{i,*}\right)^{\intercal} \theta - \mathbb{E} r_t^i, \quad \text{and} \quad \mathcal{R}_t := \sum_{k=1}^{t} \sum_{i=1}^{|V|} \rho_t^i,$$

where $x_t^{i,*} := \operatorname{argmax}_{x \in \mathcal{D}_t^i} x^{\intercal}\theta$. The aim of the agents is to minimise the rate of increase of cumulative regret. We also wish them to use a sharing protocol that does not impose much strain on the information-sharing communication channel.

Gossip protocol In a gossip protocol (see, e.g., [61, 104, 49, 50]), in each round, an overlay protocol assigns to every agent another agent, with which it can share information. After sharing, the agents aggregate the information and, based on that, they make their corresponding decisions in the next round. In many areas of distributed learning and computation gossip protocols have offered a good compromise between low-communication costs and algorithm performance. Using such a protocol in the multi-agent bandit setting, one faces two major challenges.

First, information sharing is not perfect, since each agent acquires information from only one other (randomly chosen) agent per round. This introduces a bias through the unavoidable doubling of data points. The solution is to mitigate this by using a delay (typically of $O(\log t)$) on the time at which information gathered is used. After this delay, the information is sufficiently mixed among the agents, and the bias vanishes.

Second, in order to realize this delay, it is necessary to store information in a buffer and only use it to make decisions after the delay has been passed. In [96] this was achieved by introducing an epoch structure into their algorithm, and emptying the buffers at the end of each epoch.

The Distributed Confidence Ball Algorithm (DCB) We use a gossip-based information sharing protocol to produce a distributed variant of the generic Confidence Ball (CB) algorithm, [1, 29, 69]. Our approach is similar to [96] where the authors produced a distributed ϵ-greedy algorithm for the simpler multi-armed bandit problem. However their results do not generalise easily, and thus significant new analysis is needed. One reason is that the linear setting introduces serious complications in the analysis of the delay effect mentioned in the previous paragraphs. Additionally, their algorithm is epoch-based, whereas we are using a more natural and simpler algorithmic structure. The downside is that the size of the buffers of our algorithm grow with time. However, our analyses easily transfer to the epoch approach too. As the rate of growth is logarithmic, our algorithm is still efficient over a very long time-scale.

The simplifying assumption so far is that all agents are solving the same underlying bandit problem, i.e. finding the same unknown θ-vector. This, however, is often unrealistic, and so we relax it in our next setup. While it may have uses in special cases, DCB and its analysis can be considered as a base for providing an algorithm in this more realistic setup, where some variation in θ is allowed across the network.

Clustered Linear Bandits Proposed in [41, 71, 75], this has recently proved to be a very successful model for recommendation problems with massive numbers of users. It comprises a multi-agent linear bandit model agents' θ-vectors are allowed to vary across a clustering. This clustering presents an additional challenge to find the groups of agents sharing the same underlying bandit problem before information sharing can accelerate the learning process. Formally, let $\{U^k\}_{k=1,\ldots,M}$ be a clustering of V, assume some coefficient vector θ^k for each k, and let for agent $i \in U^k$ the reward of action x^i_t be given by

$$r^i_t = (x^i_t)^\intercal \theta^k + \xi^i_t.$$

Both clusters and coefficient vectors are assumed to be initially unknown, and so need to be learnt on the fly.

The Distributed Clustering Confidence Ball Algorithm (DCCB) The paper [41] proposes the initial centralised approach to the problem of clustering linear bandits. Their approach is to begin with a single cluster, and then incrementally prune edges when the available information suggests that two agents belong to different clusters. We show how to use a gossip-based protocol to give a distributed variant of this algorithm, which we call DCCB.

Our main contributions In Theorems 9 and 14 we show our algorithms DCB and DCCB achieve, in the multi-agent and clustered setting, respectively, near-optimal improvements in the regret rates. In particular, they are of order almost $\sqrt{|V|}$ better than applying CB without information sharing, while still keeping communication cost low. And our findings are demonstrated by experiments on real-world benchmark data.

3.2 Linear Bandits and the DCB Algorithm

The generic Confidence Ball (CB) algorithm is designed for a single agent linear bandit problem (i.e. $|V| = 1$). The algorithm maintains a confidence ball $C_t \subset \mathbb{R}^d$ within which it believes the true parameter θ lies with high probability. This confidence ball is computed from the observation pairs, $(x_k, r_k)_{k=1,\ldots,t}$ (for the sake of simplicity, we dropped the agent index, i). Typically, the covariance matrix $A_t = \sum_{k=1}^t x_k x_k^\intercal$ and b-vector, $b_t = \sum_{k=1}^t r_k x_k$, are sufficient statistics to characterise this confidence ball. Then, given its current action set, $\mathcal{D}_t$, the agent selects the optimistic action, assuming that the true parameter sits in C_t, i.e. $(x_t, \sim) = \text{argmax}_{(x,\theta')\in\mathcal{D}_t\times C_t}\{x^\intercal\theta'\}$. Pseudo-code for CB is given in the Appendix 3.5.1.

Gossip Sharing Protocol for DCB We assume that the agents are sharing across a peer to peer network, i.e. every agent can share information with every other agent, but that every agent can communicate with only one other agent per round. In our algorithms, each agent, i, needs to maintain

(1) a *buffer* (an ordered set) $\mathcal{A}_t^i$ of covariance matrices and an *active* covariance matrix $\widetilde{A}_t^i$,
(2) a *buffer* $\mathcal{B}_t^i$ of b-vectors and an *active* b-vector $\widetilde{b}_t^i$,

Initially, we set, for all $i \in V$, $\widetilde{A}_0^i = I$, $\widetilde{b}_0^i = 0$. These *active* objects are used by the algorithm as sufficient statistics from which to calculate confidence balls, and summarise only information gathered before or during time $\tau(t)$, where τ is an arbitrary monotonically increasing function satisfying $\tau(t) < t$. The buffers are initially set to $\mathcal{A}_0^i = \emptyset$, and $\mathcal{B}_0^i = \emptyset$. For each $t > 1$, each agent, i, shares and updates its buffers as follows:

(1) a random permutation, σ, of the numbers $1, \ldots, |V|$ is chosen uniformly at random in a decentralised manner among the agents,[1]
(2) the buffers of i are then updated by averaging its buffers with those of $\sigma(i)$, and then extending them using their current observations[2]

$$\mathcal{A}_{t+1}^i = \left(\left(\tfrac{1}{2}(\mathcal{A}_t^i + \mathcal{A}_t^{\sigma(i)})\right) \circ \left(x_{t+1}^i \, (x_{t+1}^i)^\mathsf{T}\right)\right),$$
$$\mathcal{B}_{t+1}^i = \left(\left(\tfrac{1}{2}(\mathcal{B}_t^i + \mathcal{B}_t^{\sigma(i)})\right) \circ \left(r_{t+1}^i x_{t+1}^i\right)\right),$$

$\widetilde{A}_{t+1}^i = \widetilde{A}_t^i + \widetilde{A}_t^{\sigma(i)}$, and $\widetilde{b}_{t+1}^i = \widetilde{b}_t^i + \widetilde{b}_t^{\sigma(i)}$.

(3) if the length $|\mathcal{A}_{t+1}^i|$ exceeds $t - \tau(t)$, the first element of $\mathcal{A}_{t+1}^i$ is added to $\widetilde{A}_{t+1}^i$ and deleted from $\mathcal{A}_{t+1}^i$. $\mathcal{B}_{t+1}^i$ and $\widetilde{b}_{t+1}^i$ are treated similarly.

In this way, each buffer remains of size at most $t - \tau(t)$, and contains only information gathered after time $\tau(t)$. The result is that, after t rounds of sharing, the current covariance matrices and b-vectors used by the algorithm to make decisions have the form:

$$\widetilde{A}_t^i := I + \sum_{t'=1}^{\tau(t)} \sum_{i'=1}^{|V|} w_{i,t}^{i',t'} x_{t'}^{i'} x_{t'}^{i'\mathsf{T}},$$
$$\text{and } \widetilde{b}_t^i := \sum_{t'=1}^{\tau(t)} \sum_{i'=1}^{|V|} w_{i,t}^{i',t'} r_{t'}^{i'} x_{t'}^{i'}.$$

where the weights $w_{i,t}^{i',t'}$ are random variables which are unknown to the algorithm. Importantly for our analysis, as a result of the overlay protocol's uniformly random choice of σ, they are identically distributed ($i.d.$) for each fixed pair (t, t'), and $\sum_{i' \in V} w_{i,t}^{i',t'} = |V|$. If information sharing was perfect at each time step, then the

[1]This can be achieved in a variety of ways.

[2]The $\circ$ symbol denotes the concatenation operation on two ordered sets: if $x = (a, b, c)$ and $y = (d, e, f)$, then $x \circ y = (a, b, c, d, e, f)$, and $y \circ x = (d, e, f, a, b, c)$.

current covariance matrix could be computed using all the information gathered by all the agents, and would be:

$$A_t := I + \sum_{i'=1}^{|V|} \sum_{t'=1}^{t} x_{t'}^{i'} \left(x_{t'}^{i'}\right)^{\mathsf{T}}. \tag{3.1}$$

DCB algorithm The OFUL algorithm [1] is an improvement of the confidence ball algorithm from [29], which assumes that the confidence balls C_t can be characterised by A_t and b_t. In the DCB algorithm, each agent $i \in V$ maintains a confidence ball C_t^i for the unknown parameter θ as in the OFUL algorithm, but calculated from $\widetilde{A}_t^i$ and $\widetilde{b}_t^i$. It then chooses its action, x_t^i, to satisfy $(x_t^i, \theta_t^i) = \operatorname{argmax}_{(x,\theta)\in\mathcal{D}_t^i \times C_t^i} x^{\mathsf{T}}\theta$, and receives a reward r_t^i. Finally, it shares its information buffer according to the sharing protocol above. Pseudo-code for DCB is given in Appendix 3.5.1, and in Algorithm 1.

3.2.1 Results for DCB

Theorem 9. *Let $\tau(\cdot) : t \to 4\log(|V|^{\frac{3}{2}}t)$. Then, with probability $1-\delta$, the regret of DCB is bounded by*

$$\begin{aligned}\mathcal{R}_t \leq & \left(N(\delta)|V| + \nu(|V|, d, t)\right) \|\theta\|_2 \\ & + 4e^2 \left(\beta(t) + 4R\right) \sqrt{|V| t \ln\left((1 + |V|t/d)^d\right)},\end{aligned}$$

where $\nu(|V|, d, t) := (d+1)d^2(4|V|\ln(|V|^{\frac{3}{2}}t))^3$, $N(\delta) := \sqrt{3}/((1-2^{-\frac{1}{4}})\sqrt{\delta})$, and

$$\beta(t) := R\sqrt{\ln\left(\frac{(1+|V|t/d)^d}{\delta}\right)} + \|\theta\|_2. \tag{3.2}$$

The term $\nu(t, |V|, d)$ describes the loss compared to the centralised algorithm due to the delay in using information, while $N(\delta)|V|$ describes the loss due to the incomplete mixing of the data across the network.

If the agents implement CB independently and do not share any information, which we call CB-*NoSharing*, then it follows from the results in [1], the equivalent regret bound would be

$$\mathcal{R}_t \leq |V|\beta(t)\sqrt{t \ln\left((1+t/d)^d\right)} \tag{3.3}$$

Comparing Theorem 9 with (3.3) tells us that, after an initial "burn in" period, the gain in regret performance of DCB over CB-*NoSharing* is of order almost $\sqrt{|V|}$.

Corollary 10. *We can recover a bound in expectation from Theorem 9, by using the value* $\delta = 1/\sqrt{|V|t}$*:*

$$\begin{aligned}\mathbb{E}[\mathcal{R}_t] &\leq O(t^{\frac{1}{4}}) + \sqrt{|V|t}\|\theta\|_2 \\ &+ 4e^2\left(R\sqrt{\ln\left((1+|V|t/d)^d\sqrt{|V|t}\right)} + \|\theta\|_2 + 4R\right) \\ &\qquad \times \sqrt{|V|t\ln\left((1+|V|t/d)^d\right)}.\end{aligned}$$

This shows that DCB exhibits asymptotically optimal regret performance, up to log factors, in comparison with any algorithm that can share its information perfectly between agents at each round.

Communication Complexity

If the agents communicate their information to each other at each round without a central server, then every agent would need to communicate their chosen action and reward to every other agent at each round, giving a communication cost of order $d|V|^2$ per-round. We call such an algorithm CB-*InstSharing*. Under the gossip protocol we propose each agent requires at most $O(log_2(|V|t)d^2|V|)$ bits to be communicated per round. Therefore, a significant communication cost reduction is gained when $log(|V|t)d \ll |V|$.

Using an epoch-based approach, as in [96], the per-round communication cost of the gossip protocol becomes $O(d^2|V|)$. This improves efficiency over any horizon, requiring only that $d \ll |V|$, and the proofs of the regret performance are simple modifications of those for DCB. However, in comparison with growing buffers this is only an issue after $O(\exp(|V|))$ number of rounds, and typically $|V|$ is large.

While the DCB has a clear communication advantage over CB-*InstSharing*, there are other potential approaches to this problem. For example, instead of randomised neighbour sharing one can use a deterministic protocol such as *Round-Robin* (RR), which can have the same low communication costs as DCB. However, the regret bound for RR suffers from a naturally larger delay in the network than DCB. Moreover, attempting to track potential doubling of data points when using a gossip protocol, instead of employing a delay, leads back to a communication cost of order $|V|^2$ per round. More detail is included in Appendix 3.5.2.

Proof of Theorem 9

In the analysis we show that the bias introduced by imperfect information sharing is mitigated by delaying the inclusion of the data in the estimation of the parameter θ. The proof builds on the analysis in [1]. The emphasis here is to show how to handle the extra difficulty stemming from imperfect information sharing, which results in the influence of the various rewards at the various peers being unbalanced and appearing with a random delay. Proofs of the Lemmas 11 and 12, and of Proposition 1 are crucial, but technical, and are deferred to Appendix 3.5.3.

Step 1: Define modified confidence ellipsoids. First we need a version of the confidence ellipsoid theorem given in [1] that incorporates the bias introduced by the random weights:

Proposition 1. *Let* $\delta > 0$, $\widetilde{\theta}_t^i := (\widetilde{A}_t^i)^{-1}\widetilde{b}_t^i$, $W(\tau) := \max\{w_{i,t}^{i',t'} : t, t' \leq \tau,\ i, i' \in V\}$, *and let*

$$C_t^i := \Bigg\{ x \in \mathbb{R}^d : \|\widetilde{\theta}_t^i - x\|_{\widetilde{A}_t^i} \leq \|\theta\|_2 \tag{3.4}$$
$$+ W(\tau(t)) R \sqrt{2 \log \left(\det(\widetilde{A}_t^i)^{\frac{1}{2}} / \delta \right)} \Bigg\}.$$

Then with probability $1 - \delta$, $\theta \in C_t^i$.

In the rest of the proof we assume that $\theta \in C_t^i$.

Step 2: Instantaneous regret decomposition. Denote by $(x_t^i, \theta_t^i) = \operatorname{argmax}_{x \in D_t^i, y \in C_t^i} x^\intercal y$. Then we can decompose the instantaneous regret, following a classic argument (see the proof of Theorem 3 in [1]):

$$\begin{aligned} \rho_t^i &= \left(x_t^{i,*}\right)^\intercal \theta - (x_t^i)^\intercal \theta \leq \left(x_t^i\right)^\intercal \theta_t^i - (x_t^i)^\intercal \theta \\ &= \left(x_t^i\right)^\intercal \left[\left(\theta_t^i - \widetilde{\theta}_t^i\right) + \left(\widetilde{\theta}_t^i - \theta\right) \right] \\ &\leq \|x_t^i\|_{\left(\widetilde{A}_t^i\right)^{-1}} \left[\left\| \theta_t^i - \widetilde{\theta}_t^i \right\|_{\widetilde{A}_t^i} + \left\| \widetilde{\theta}_t^i - \theta \right\|_{\widetilde{A}_t^i} \right] \end{aligned} \tag{3.5}$$

Step 3: Control the bias. The norm differences inside the square brackets of the regret decomposition are bounded through (3.4) in terms of the matrices $\widetilde{A}_t^i$. We would like, instead, to have the regret decomposition in terms of the matrix A_t (which is defined in (3.1)). To this end, we give some lemmas showing that using the matrices $\widetilde{A}_t^i$ is almost the same as using A_t. These lemmas involve elementary matrix analysis, but are crucial for understanding the impact of imperfect information sharing on the final regret bounds.

Step 3a: Control the bias coming from the weight imbalance.

Lemma 11 (Bound on the influence of general weights)**.** *For all* $i \in V$ *and* $t > 0$,

$$\|x_t^i\|^2_{\left(\widetilde{A}_t^i\right)^{-1}} \leq e^{\sum_{t'=1}^{\tau(t)} \sum_{i'=1}^{|V|} \left| w_{i,t}^{i',t'} - 1 \right|} \|x_t^i\|^2_{\left(A_{\tau(t)}\right)^{-1}},$$
$$\text{and } \det\left(\widetilde{A}_t^i\right) \leq e^{\sum_{t'=1}^{\tau(t)} \sum_{i'=1}^{|V|} \left| w_{i,t}^{i',t'} - 1 \right|} \det\left(A_{\tau(t)}\right).$$

Using Lemma 4 in [96], by exploiting the random weights are identically distributed ($i.d.$) for each fixed pair (t, t'), and $\sum_{i' \in V} w_{i,t}^{i',t'} = |V|$ under our gossip protocol, we can control the random exponential constant in Lemma 11, and the upper bound $W(T)$ using the Chernoff-Hoeffding bound:

Lemma 12 (Bound on the influence of weights under our sharing protocol)**.** *Fix some constants* $0 < \delta_{t'} < 1$. *Then with probability* $1 - \sum_{t'=1}^{\tau(t)} \delta_{t'}$

$$\sum_{i'=1}^{|V|} \sum_{t'=1}^{\tau(t)} \left| w_{i,t}^{i',t'} - 1 \right| \leq |V|^{\frac{3}{2}} \sum_{t'=1}^{\tau(t)} \left(2^{(t-t')} \delta_{t'} \right)^{-\frac{1}{2}},$$

$$\text{and } W(T) \leq 1 + \max_{1 \leq t' \leq \tau(t)} \left\{ |V|^{\frac{3}{2}} \left(2^{(t-t')} \delta_{t'} \right)^{-\frac{1}{2}} \right\}.$$

In particular, for any $\delta \in (0,1)$, choosing $\delta_{t'} = \delta 2^{\frac{t'-t}{2}}$, with probability $1 - \delta/(|V|^3 t^2 (1-2^{-1/2}))$ we have

$$\sum_{i'=1}^{|V|} \sum_{t'=1}^{\tau(t)} \left| w_{i,t}^{i',t'} - 1 \right| \leq \frac{1}{(1-2^{-\frac{1}{4}}) t \sqrt{\delta}},$$

$$\text{and } W(\tau(t)) \leq 1 + \frac{|V|^{\frac{3}{2}}}{t\sqrt{\delta}}. \tag{3.6}$$

Thus Lemma 11 and 12 give us control over the bias introduced by the imperfect information sharing. Combining them with Equations (3.4) and (3.5) we find that with probability $1 - \delta/(|V|^3 t^2 (1-2^{-1/2}))$:

$$\rho_t^i \leq 2e^{C(t)} \|x_t^i\|_{\left(A_{\tau(t)}^i\right)^{-1}} (1 + C(t)) \tag{3.7}$$
$$\times \left[R\sqrt{2 \log \left(e^{C(t)} \det \left(A_{\tau(t)} \right)^{\frac{1}{2}} \delta^{-1} \right)} + \|\theta\| \right]$$

where $C(t) := 1/(1-2^{-1/4}) t \sqrt{\delta}$

Step 3b: Control the bias coming from the delay. Next, we need to control the bias introduced from leaving out the last $4 \log(|V|^{3/2} t)$ time steps from the confidence ball estimation calculation:

Proposition 2. *There can be at most*

$$\nu(k) := (4|V| \log(|V|^{3/2} k))^3 (d+1) d (tr(A_0) + 1) \tag{3.8}$$

pairs $(i,k) \in 1, \ldots, |V| \times \{1, \ldots, t\}$ *for which one of*

$$\|x_k^i\|^2_{A_{\tau(k)}^{-1}} \geq e \|x_k^i\|^2_{\left(A_{k-1} + \sum_{j=1}^{i-1} x_k^j (x_k^j)^\intercal\right)^{-1}},$$

$$\textit{or } \det \left(A_{\tau(k)} \right) \geq e \det \left(A_{k-1} + \sum_{j=1}^{i-1} x_k^j (x_k^j)^\intercal \right) \textit{ holds.}$$

Step 4: Choose constants and sum the simple regret. Defining a constant

$$N(\delta) := \frac{1}{(1-2^{-\frac{1}{4}}) \sqrt{\delta}},$$

we have, for all $k \geq N(\delta)$, $C(k) \leq 1$, and so, by (3.7) with probability $1 - (|V|k)^{-2}\delta/(1-2^{-1/2})$

$$\begin{aligned}\rho_k^i \leq & 2e\|x_k^i\|_{A_{\tau(k)}^{-1}} \\ & \times \left[2R\sqrt{2\log\left(\frac{e\det\left(A_{\tau(k)}\right)^{\frac{1}{2}}}{\delta}\right)} + \|\theta\|_2 \right].\end{aligned} \tag{3.9}$$

Now, first applying Cauchy-Schwarz, then step 3b from above together with (3.9), and finally Lemma 11 from [1] yields that, with probability $1 - \left(1 + \sum_{t=1}^{\infty}(|V|t)^{-2}/(1-2^{-1/2})\right)\delta \geq 1 - 3\delta$,

$$\begin{aligned}\mathcal{R}_t \leq & N(\delta)|V|\|\theta\|_2 + \left[|V|t \sum_{t'=N(\delta)}^{t} \sum_{i=1}^{|V|} \left(\rho_{t'}^i\right)^2 \right]^{\frac{1}{2}} \\ \leq & \left(N(\delta)|V| + \nu(|V|, d, t)\right)\|\theta\|_2 \\ & + 4e^2\left(\beta(t) + 2R\right)\left[|V|t \sum_{t'=1}^{t}\sum_{i=1}^{M} \|x_t^i\|^2_{(A_t)^{-1}} \right]^{\frac{1}{2}} \\ \leq & \left(N(\delta)|V| + \nu(|V|, d, t)\right)\|\theta\|_2 \\ & + 4e^2\left(\beta(t) + 2R\right)\sqrt{|V|t\left(2\log\left(\det\left(A_t\right)\right)\right)},\end{aligned}$$

where $\beta(\cdot)$ is as defined in (3.2). Replacing δ with $\delta/3$ finishes the proof.

Proof of Proposition 2

This proof forms the major innovation in the proof of Theorem 9. Let $(y_k)_{k\geq 1}$ be any sequence of vectors such that $\|y_k\|_2 \leq 1$ for all k, and let $B_n := B_0 + \sum_{k=1}^{n} y_k y_k^{\intercal}$, where B_0 is some positive definite matrix.

Lemma 13. *For all $t > 0$, and for any $c \in (0,1)$, we have*

$$\begin{aligned}&\left|\left\{k \in \{1,2,\dots\} : \|y_k\|^2_{B_{k-1}^{-1}} > c\right\}\right| \\ &\qquad \leq (d+c)d(tr(B_0^{-1}) - c)/c^2,\end{aligned}$$

Proof. We begin by showing that, for any $c \in (0,1)$

$$\|y_k\|^2_{B_{k-1}^{-1}} > c \tag{3.10}$$

can be true for only $2dc^{-3}$ different k.

Indeed, let us suppose that (3.10) is true for some k. Let $(e_i^{(k-1)})_{1\leq i\leq d}$ be the orthonormal eigenbasis for B_{k-1}, and, therefore, also for B_{k-1}^{-1}, and write $y_k =$

$\sum_{i=1}^{d} \alpha_i e_i$. Let, also, $(\lambda_i^{(k-1)})$ be the eigenvalues for B_{k-1}. Then,

$$\begin{aligned} c < y_k^\intercal B_{k-1}^{-1} y_k = \sum_{i=1}^{d} \tfrac{\alpha_i^2}{\lambda_i^{(k-1)}} &\leq tr(B_{k-1}^{-1}), \\ \implies \exists j \in \{1, \dots, d\} : \tfrac{\alpha_j^2}{\lambda_j^{(k-1)}}, \tfrac{1}{\lambda_j^{(k-1)}} &> \tfrac{c}{d}, \end{aligned}$$

where we have used that $\alpha_i^2 < 1$ for all i, since $\|y_k\|_2 < 1$. Now,

$$\begin{aligned} & tr(B_{k-1}^{-1}) - tr(B_k^{-1}) \\ & \quad = tr(B_{k-1}^{-1}) - tr((B_{k-1} + y_k y_k^\intercal)^{-1}) \\ & \quad > tr(B_{k-1}^{-1}) - tr((B_{k-1} + \alpha_j^2 e_j e_j^\intercal)^{-1}) \\ & \quad = \tfrac{1}{\lambda_j^{(k-1)}} - \tfrac{1}{\lambda_j^{(k-1)} + \alpha_j^2} = \tfrac{\alpha_j^2}{\lambda_j^{(k-1)}(\lambda_j^{(k-1)} + \alpha_j^2)} \\ & \quad > \left(d^2 c^{-2} + d c^{-1}\right)^{-1} > \tfrac{c^2}{d(d+c)} \end{aligned}$$

So we have shown that (3.10) implies that

$$tr(B_{k-1}^{-1}) > c \text{ and } tr(B_{k-1}^{-1}) - tr(B_k^{-1}) > \frac{c^2}{d(d+c)}.$$

Since $tr(B_0^{-1}) \geq tr(B_{k-1}^{-1}) \geq tr(B_k^{-1}) \geq 0$ for all k, it follows that (3.10) can be true for at most $(d+c)d(tr(B_0^{-1}) - c)c^{-2}$ different k. □

Now, using an argument similar to the proof of Lemma 11, for all $k < t$

$$\begin{aligned} \|y_{k+1}\|_{B_{\tau(k)}^{-1}} &\leq e^{\sum_{s=\tau(k)+1}^{k} \|y_{s+1}\|_{B_s^{-1}}} \|y_{k+1}\|_{B_k^{-1}}, \\ \text{and } \det\left(B_{\tau(t)}\right) &\leq e^{\sum_{k=\tau(t)+1}^{t} \|y_k\|_{B_k^{-1}}^2} \det\left(B_t\right). \end{aligned}$$

Therefore,

$$\begin{aligned} \|y_{k+1}\|_{B_{\tau(k)}^{-1}} \geq c \|y_{k+1}\|_{B_k^{-1}} &\text{ or } \det(B_{\tau(k)}) \geq c \det(B_k) \\ \implies \sum_{s=\tau(k)}^{k-1} \|y_{s+1}\|_{B_s^{-1}} &\geq \ln(c) \end{aligned}$$

However, according to Lemma 13, there can be at most

$$\nu(t) := \left(d + \tfrac{\ln(c)}{\Delta(t)}\right) d \left(tr\left(B_0^{-1}\right) - \tfrac{\ln(c)}{\Delta(t)}\right) \left(\tfrac{\Delta(t)}{\ln(c)}\right)^2$$

times $s \in \{1, \dots, t\}$, such that $\|y_{s+1}\|_{B_s^{-1}} \geq \ln(c)/\Delta(t)$, where $\Delta(t) := \max_{1 \leq k \leq t}\{k - \tau(k)\}$. Hence $\sum_{s=\tau(j)+1}^{k} \|y_{s+1}\|_{B_s}^{-1} \geq \ln(c)$ is true for at most $\Delta(t)\nu(|V|, d, t)$ indices $k \in \{1, \dots, t\}$.

Finally, we finish by setting $(y_k)_{k \geq 1} = \circ_{t \geq 1}(x_t^i)_{i=1}^{|V|}$.

3.3 Clustering and the DCCB Algorithm

We now incorporate distributed clustering into the DCB algorithm. The analysis of DCB forms the backbone of the analysis of DCCB.

Algorithm 1 Distributed Clustering Confidence Ball

```
Input: Size of network $|V|, \tau : t \to t - 4\log_2 t, \alpha, \lambda$
Initialization: $\forall i \in V$, set $\widetilde{A}_0^i = I_d, \widetilde{b}_0^i = \mathbf{0}, \mathcal{A}_0^i = \mathcal{B}_0^i = \emptyset$, and $V_0^i = V$.
for $t = 0, \dots \infty$ do
    Draw a random permutation $\sigma$ of $\{1, \dots, V\}$ respecting the current local clusters
    for $i = 1, \dots, |V|$ do
        Receive action set $\mathcal{D}_t^i$ and construct the confidence ball $C_t^i$ using $\widetilde{A}_t^i$ and $\widetilde{b}_t^i$
        Choose action and receive reward:
        Find $(x_{t+1}^i, *) = \text{argmax}_{(x,\widetilde{\theta}) \in \mathcal{D}_t^i \times C_t^i} x^\mathsf{T}\widetilde{\theta}$, and get reward $r_{t+1}^i$ from context $x_{t+1}^i$.
        Share and update information buffers:
        if $\|\hat{\theta}_{local}^i - \hat{\theta}_{local}^j\| > c_\lambda^{thresh}(t)$
        Update local cluster: $V_{t+1}^i = V_t^i \setminus \{\sigma(i)\}, V_{t+1}^{\sigma(i)} = V_t^{\sigma(i)} \setminus \{i\}$, and reset according to (3.13)
        elseif $V_t^i = V_t^{\sigma(i)}$
        Set $\mathcal{A}_{t+1}^i = \left(\frac{1}{2}(\mathcal{A}_t^i + \mathcal{A}_t^{\sigma(i)})\right) \circ (x_{t+1}^i (x_{t+1}^i)^\mathsf{T})$ and $\mathcal{B}_{t+1}^i = \left(\frac{1}{2}(\mathcal{B}_t^i + \mathcal{B}_t^{\sigma(i)})\right) \circ (r_{t+1}^i x_{t+1}^i)$
        else
        Update: Set $\mathcal{A}_{t+1}^i = \mathcal{A}_t^i \circ (x_{t+1}^i (x_{t+1}^i)^\mathsf{T})$ and $\mathcal{B}_{t+1}^i = \mathcal{B}_t^i \circ (r_{t+1}^i x_{t+1}^i)$
        endif
        Update local estimator: $A_{local,t+1}^i = A_{local,t}^i + x_{t+1}^i (x_{t+1}^i)^\mathsf{T}$, $b_{local,t+1}^i = b_{local,t}^i + r_{t+1}^i x_{t+1}^i$, and $\hat{\theta}_{local,t+1} = \left(A_{local,t+1}^i\right)^{-1} b_{local,t+1}^i$
        if $|\mathcal{A}_{t+1}^i| > t - \tau(t)$ set $\widetilde{A}_{t+1}^i = \widetilde{A}_t^i + \mathcal{A}_{t+1}^i(1)$, $\mathcal{A}_{t+1}^i = \mathcal{A}_{t+1}^i \setminus \mathcal{A}_{t+1}^i(1)$. Similarly for $\mathcal{B}_{t+1}^i$.
    end for
end for
```

DCCB Pruning Protocol In order to run DCCB, each agent i must maintain some local information buffers in addition to those used for DCB. These are:

(1) a local covariance matrix $A_{local}^i = A_{local,t}^i$, a local b-vector $b_{local}^i = b_{local,t}^i$,

(2) and a local neighbour set V_t^i.

The local covariance matrix and b-vector are updated as if the agent was applying the generic (single agent) confidence ball algorithm: $A^i_{local,0} = A_0$, $b^i_{local,0} = 0$,

$$A^i_{local,t} = x^i_t (x^i_t)^\intercal + A^i_{local,t-1},$$
$$\text{and } b^i_{local,t} = r^i_t x^i_t + b^i_{local,t-1}.$$

DCCB Algorithm Each agent's local neighbour set V^i_t is initially set to V. At each time step t, agent i contacts one other agent, j, at random from V^i_t, and both decide whether they do or do not belong to the same cluster. To do this they share local estimates, $\hat{\theta}^i_t = {A^i_{local,t}}^{-1} b^i_{local,t}$ and $\hat{\theta}^j_t = {A^j_{local,t}}^{-1} b^j_{local,t}$, of the unknown parameter of the bandit problem they are solving, and see if they are further apart than a threshold function $c = c^{thresh}_\lambda(t)$, so that if

$$\|\hat{\theta}^i_t - \hat{\theta}^j_t\|_2 \geq c^{thresh}_\lambda(t), \tag{3.11}$$

then $V^i_{t+1} = V^i_t \setminus \{j\}$ and $V^j_{t+1} = V^j_t \setminus \{i\}$. Here λ is a parameter of an extra assumption that is needed, as in [41], about the process generating the context sets $\mathcal{D}^i_t$:

(A) Each context set $\mathcal{D}^i_t = \{x_k\}_k$ is finite and contains $i.i.d.$ random vectors such that for all, k, $\|x_k\| \leq 1$ and $\mathbb{E}(x_k x_k^\intercal)$ is full rank, with minimal eigenvalue $\lambda > 0$.

We define $c^{thresh}_\lambda(t)$, as in [41], by

$$c^{thresh}_\lambda(t) := \frac{R\sqrt{2d\log(t) + 2\log(2/\delta)} + 1}{\sqrt{1 + \max\{A_\lambda(t, \delta/(4d)), 0\}}} \tag{3.12}$$

where $A_\lambda(t,\delta) := \frac{\lambda t}{\delta} - 8\log\frac{t+3}{\delta} - 2\sqrt{t\log\frac{t+3}{\delta}}$.

The DCCB algorithm is pretty much the same as the DCB algorithm, except that it also applies the pruning protocol described. In particular, each agent, i, when sharing its information with another, j, has three possible actions:

(1) if (3.11) is not satisfied and $V^i_t = V^j_t$, then the agents share simply as in the DCB algorithm;

(2) if (3.11) is not satisfied but $V^i_t \neq V^j_t$, then no sharing or pruning occurs.

(3) if (3.11) is satisfied, then both agents remove each other from their neighbour sets and reset their buffers and active matrices so that

$$\mathcal{A}^i = (0, 0, \ldots, A^i_{local}), \mathcal{B}^i = (0, 0, \ldots, b^i_{local}),$$
$$\text{and } \widetilde{A}^i = A^i_{local}, \widetilde{b}^i = b^i_{local}, \tag{3.13}$$

and similarly for agent j.

It is proved in the theorem below, that under this sharing and pruning mechanism, in high probability after some finite time each agent i finds its true cluster, i.e. $V^i_t = U^k$. Moreover, since the algorithm resets to its local information each time a pruning occurs, once the true clusters have been identified, each cluster shares only information gathered within that cluster, thus avoiding introducing a bias by sharing information gathered from outside the cluster before the clustering has been identified. Full pseudo-code for the DCCB algorithm is given in Algorithm 1, and the differences with the DCB algorithm are highlighted in blue.

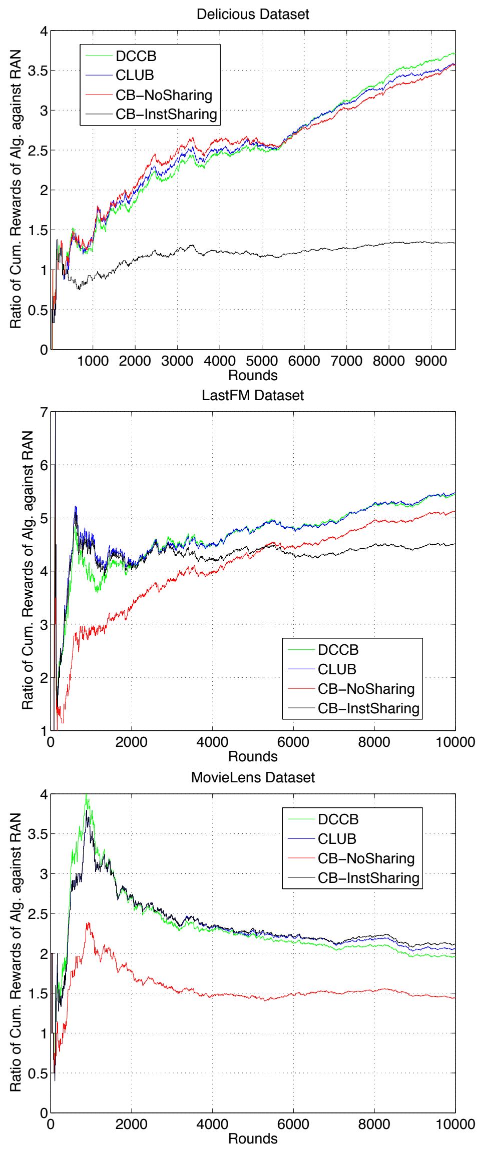


Figure 3.1: Here we plot the performance of DCCB in comparison to CLUB, CB-*NoSharing* and CB-*InstSharing*. The plots show the ratio of cumulative rewards achieved by the algorithms to the cumulative rewards achieved by the random algorithm.

3.3.1 Results for DCCB

Theorem 14. *Assume that (A) holds, and let γ denote the smallest distance between the bandit parameters θ^k. Then there exists a constant $C = C(\gamma, |V|, \lambda, \delta)$, such that with probability $1-\delta$ the total cumulative regret of cluster k when the agents employ DCCB is bounded by*

$$\begin{aligned}\mathcal{R}_t \leq \Bigg[\max\left\{\sqrt{2}N(\delta), C + 4\log_2(|V|^{\frac{3}{2}}C)\right\} |U^k| \\ + \nu(|U^k|, d, t)\Bigg] \|\theta\|_2 \\ + 4e\left(\beta(t) + 3R\right)\sqrt{|U^k| t \ln\left((1 + |U^k|t/d)^d\right)},\end{aligned}$$

where N and ν are as defined in Theorem 9, and $\beta(t) := R\sqrt{2\ln\left((1+|U^k|t/d)^d\right)} + \|\theta\|_2$.

The constant $C(\gamma, |V|, \lambda, \delta)$ is the time that you have to wait for the true clustering to have been identified,

The analysis follows the following scheme: When the true clusters have been correctly identified by all nodes, within each cluster the algorithm, and thus the analysis, reduces to the case of Section 3.2.1. We adapt results from [41] to show how long it will be before the true clusters are identified, in high probability. The proof is deferred to Appendices 3.5.4 and 3.5.5.

3.4 Experiments and Discussion

Experiments We closely implemented the experimental setting and dataset construction principles used in [71, 75], and for a detailed description of this we refer the reader to [71]. We evaluated DCCB on three real-world datasets against its centralised counterpart CLUB, and against the benchmarks used therein, CB-*NoSharing*, and CB-*InstSharing*. The LastFM dataset comprises of 91 users, each of which appear at least 95 times. The Delicious dataset has 87 users, each of which appear at least 95 times. The MovieLens dataset contains 100 users, each of which appears at least 250 times. The performance was measured using the ratio of cumulative reward of each algorithm to that of the predictor which chooses a random action at each time step. This is plotted in in Figure 3.1. From the experimental results it is clear that DCCB performs comparably to CLUB in practice, and both outperform CB-*NoSharing*, and CB-*InstSharing*.

Relationship to existing literature There are several strands of research that are relevant and complimentary to this work. First, there is a large literature on single agent linear bandits, and other more, or less complicated bandit problem settings. There is already work on distributed approaches to multi-agent, multi-armed bandits, not least [96] which examines ϵ-greedy strategies over a peer to peer network, and provided an initial inspiration for this current work. The paper [54] examines the extreme case when there is no communication channel across which the agents can communicate, and all communication must be performed through observation of action choices alone. Another approach to the multi-armed bandit case, [81], directly incorporates the communication cost into the regret.

Second, there are several recent advances regarding the state-of-the-art methods for clustering of bandits. The work [71] is a faster variant of [41] which adopt the strategy of boosted training stage. In [75] the authors not only cluster the users, but also cluster the items under collaborative filtering case with a sharp regret analysis.

Finally, the paper [99] treats a setting similar to ours in which agents attempt to solve contextual bandit problems in a distributed setting. They present two algorithms, one of which is a distributed version of the approach taken in [94], and show that they achieve at least as good asymptotic regret performance in the distributed approach as the centralised algorithm achieves. However, rather than sharing information across a limited communication channel, they allow each agent only to ask another agent to choose their action for them. This difference in our settings is reflected worse regret bounds, which are of order $\Omega(T^{2/3})$ at best.

Discussion Our analysis is tailored to adapt proofs from [1] about generic confidence ball algorithms to a distributed setting. However many of the elements of these proofs, including Propositions 1 and 2 could be reused to provide similar asymptotic regret guarantees for the distributed versions of other bandit algorithms, e.g., the Thompson sampling algorithms, [2, 59, 88].

Both DCB and DCCB are synchronous algorithms. The work on distributed computation through gossip algorithms in [12] could alleviate this issue. The current pruning algorithm for DCCB guarantees that techniques from [96] can be applied to our algorithms. However the results in [12] are more powerful, and could be used even when the agents only identify a sub-network of the true clustering.

Furthermore, there are other existing interesting algorithms for performing clustering of bandits for recommender systems, such as COFIBA in [75]. It would be interesting to understand how general the techniques applied here to CLUB are.

3.5 Supplementary

3.5.1 Pseudocode of the Algorithms CB and DCB

Algorithm 2 Confidence Ball

```
**Initialization:** Set $A_0 = I$ and $b_0 = 0$.
**for** $t = 0, \dots \infty$ **do**
    Receive action set $\mathcal{D}_t$
    Construct the confidence ball $C_t$ using $A_t$ and $b_t$
    **Choose action and receive reward:**
        Find $(x_t, *) = \mathrm{argmax}_{(x,\widetilde{\theta}) \in \mathcal{D}_t \times C_t} x^\intercal \widetilde{\theta}$
        Get reward $r_t^i$ from context $x_t^i$
        Update $A_{t+1} = A_t + x_t x_t^\intercal$ and $b_{t+1} = b_t + r_t x_t$
**end for**
```

Algorithm 3 Distributed Confidence Ball

```
**Input:** Network $V$ of agents, the function $\tau : t \to t - 4\log_2(|V|^{\frac{3}{2}} t)$.
**Initialization:** For each $i$, set $\widetilde{A}_0^i = I_d$ and $\widetilde{b}_0^i = \mathbf{0}$, and the buffers $\mathcal{A}_0^i = \emptyset$ and $\mathcal{B}_0^i = \emptyset$.
**for** $t = 0, \dots \infty$ **do**
    Draw a random permutation $\sigma$ of $\{1, \dots, |V|\}$
    **for each agent** $i \in V$ **do**
        Receive action set $\mathcal{D}_t^i$ and construct the confidence ball $C_t^i$ using $\widetilde{A}_t^i$ and $\widetilde{b}_t^i$
        **Choose action and receive reward:**
            Find $(x_{t+1}^i, *) = \mathrm{argmax}_{(x,\widetilde{\theta}) \in \mathcal{D}_t^i \times C_t^i} x^\intercal \widetilde{\theta}$
            Get reward $r_{t+1}^i$ from context $x_{t+1}^i$.
        **Share and update information buffers:**
            Set $\mathcal{A}_{t+1}^i = \left(\frac{1}{2}(\mathcal{A}_t^i + \mathcal{A}_t^{\sigma(i)})\right) \circ (x_{t+1}^i (x_{t+1}^i)^\intercal)$ and $\mathcal{B}_{t+1}^i = \left(\frac{1}{2}(\mathcal{B}_t^i + \mathcal{B}_t^{\sigma(i)})\right) \circ (r_{t+1}^i x_{t+1}^i)$
            **if** $|\mathcal{A}_{t+1}^i| > t - \tau(t)$ set $\widetilde{A}_{t+1}^i = \widetilde{A}_t^i + \mathcal{A}_{t+1}^i(1)$ and $\mathcal{A}_{t+1}^i = \mathcal{A}_{t+1}^i \setminus \mathcal{A}_{t+1}^i(1)$. Similary for $\mathcal{B}_{t+1}^i$.
    **end for**
**end for**
```

3.5.2 More on Communication Complexity

First, recall that if the agents want to communicate their information to each other at each round without a central server, then every agent would need to communicate their chosen action and reward to every other agent at each round, giving a communication cost of $O(d|V|^2)$ bits per-round. Under DCB each agent requires at most $O(log_2(|V|t)d^2|V|)$ bits to be communicated per round. Therefore, a significant communication cost reduction is gained when $log(|V|t)d \ll |V|$.

Recall also that using an epoch-based approach, as in [96], we reduce the per-round communication cost of the gossip-based approach to $O(d^2|V|)$. This makes the algorithm more efficient over any time horizon, requiring only that $d \ll |V|$, and the proofs of the regret performance are simple modifications of the proofs for DCB. In comparison with growing buffers this is only an issue after $O(\exp(|V|))$ number of rounds, and typically $|V|$ is large. This is why we choose to exhibit the growing-buffer approach in this current work.

Instead of relying on the combination of the diffusion and a delay to handle the potential doubling of data points under the randomised gossip protocol, we could attempt to keep track which observations have been shared with which agents, and thus simply stop the doubling from occurring. However, the per-round communication complexity of this is at least quadratic in $|V|$, whereas our approach is linear. The reason for the former is that in order to be efficient, any agent j, when sending information to an agent i, needs to know for each k which are the latest observations gathered by agent k that agent i already knows about. The communication cost of this is of order $|V|$. Since every agent shares information with somebody in each round, this gives per round communication complexity of order $|V|^2$ in the network.

A simple, alternative approach to the gossip protocol is a *Round-Robin* (RR) protocol, in which each agent passes the information it has gathered in previous rounds to the next agent in a pre-defined permutation. Implementing a RR protocol leads to the agents performing a distributed version of the CB-*InstSharing* algorithm, but with a delay that is of size at least linear in $|V|$, rather than the logarithmic dependence on this quantity that a gossip protocol achieves. Indeed, at any time, each agent will be lacking $|V|(|V|-1)/2$ observations. Using this observation, a cumulative regret bound can be achieved using Proposition 2 which arrives at the same asymptotic dependence on $|V|$ as our gossip protocol, but with an additive constant that is worse by a multiplicative factor of $|V|$. This makes a difference to the performance of the network when $|V|$ is very large. Moreover, RR protocols do not offer the simple generalisability and robustness that gossip protocols offer.

Note that the pruning protocol for DCCB only requires sharing the estimated θ-vectors between agents, and adds at most $O(d|V|)$ to the communication cost of the algorithm. Hence the per-round communication cost of DCCB remains $O(log_2(|V|t)d^2|V|)$.

3.5.3 Proofs of Intermediary Results for DCB

Proof of Proposition 1. This follows the proof of Theorem 2 in [1], substituting appropriately weighted quantities.

Algorithm	Regret Bound	Per-Round Communication Complexity
CB-*NoSharing*	$O(\|V\|\sqrt{t})$	0
CB-*InstSharing*	$O(\sqrt{\|V\|t})$	$O(d\|V\|^2)$
DCB	$O(\sqrt{\|V\|t})$	$O(log_2(\|V\|t)d^2\|V\|)$
DCCB	$O(\sqrt{\|U^k\|t})$	$O(log_2(\|V\|t)d^2\|V\|)$

Figure 3.2: This table gives a summary of theoretical results for the multi-agent linear bandit problem. Note that CB with no sharing cannot benefit from the fact that all the agents are solving the same bandit problem, while CB with instant sharing has a large communication-cost dependency on the size of the network. DCB succesfully achieves near-optimal regret performance, while simultaneously reducing communication complexity by an order of magnitude in the size of the network. Moreover, DCCB generalises this regret performance at not extra cost in the order of the communication complexity.

For ease of presentation, we define the shorthand

$$\widetilde{X} := (\sqrt{w_1}y_1, \dots, \sqrt{w_n}y_n) \text{ and } \widetilde{\eta} = (\sqrt{w_1}\eta_1, \dots, \sqrt{w_n}\eta_n)^\intercal,$$

where the y_i are vectors with norm less than 1, the η_i are R-subgaussian, zero mean, random variables, and the w_i are positive real numbers. Then, given samples $(\sqrt{w_1}y_1, \sqrt{w_1}(\theta y_1+\eta_1)), \dots, (\sqrt{w_n}y_n, \sqrt{w_n}(\theta y_n+\eta_n))$, the maximum likelihood estimate of θ is

$$\begin{aligned}\widetilde{\theta} :&= (\widetilde{X}\widetilde{X}^\intercal + I)^{-1}\widetilde{X}(\widetilde{X}^\intercal\theta + \widetilde{\eta})\\ &= (\widetilde{X}\widetilde{X}^\intercal + I)^{-1}\widetilde{X}\widetilde{\eta} + (\widetilde{X}\widetilde{X}^\intercal + I)^{-1}(\widetilde{X}\widetilde{X}^\intercal + I)\theta - (\widetilde{X}\widetilde{X}^\intercal + I)^{-1}\theta\\ &= (\widetilde{X}\widetilde{X}^\intercal + I)^{-1}\widetilde{X}\widetilde{\eta} + \theta - (\widetilde{X}\widetilde{X}^\intercal + I)^{-1}\theta\end{aligned}$$

So by Cauchy-Schwarz, we have, for any vector x,

$$\begin{aligned}x^\intercal(\widetilde{\theta} - \theta) &= \langle x, \widetilde{X}\widetilde{\eta}\rangle_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}} - \langle x, \theta\rangle_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}} &(3.14)\\ &\leq \|x\|_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}}\left(\|\widetilde{X}\widetilde{\eta}\|_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}} + \|\theta\|_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}}\right) &(3.15)\end{aligned}$$

Now from Theorem 1 of [1], we know that with probability $1 - \delta$

$$\|\widetilde{X}\widetilde{\eta}\|^2_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}} \leq W^2R^2 2\log\sqrt{\frac{\det(\widetilde{X}\widetilde{X}^\intercal + I)}{\delta^2}}.$$

where $W = \max_{i=1,\dots,n} w_i$. So, setting $x = (\widetilde{X}\widetilde{X}^\intercal + I)^{-1}(\widetilde{\theta} - \theta)$, we obtain that with probability $1 - \delta$

$$\|\widetilde{\theta} - \theta\|_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}} \leq WR\left(2\log\sqrt{\frac{\det(\widetilde{X}\widetilde{X}^\intercal + I)}{\delta^2}}\right)^{\frac{1}{2}} + \|\theta\|_2$$

since [3]

$$\begin{aligned}\|x\|_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}}\|\theta\|_{(\widetilde{X}\widetilde{X}^\intercal+I)^{-1}} &\le \|x\|_2\lambda_{\min}^{-1}(\widetilde{X}\widetilde{X}^\intercal+I)\|\theta\|_2\lambda_{\min}^{-1}(\widetilde{X}\widetilde{X}^\intercal+I)\\ &\le \|x\|_2\|\theta\|_2.\end{aligned}$$

Conditioned on the values of the weights, the statement of Proposition 1 now follows by substituting appropriate quantities above, and taking the probability over the distribution of the subGaussian random rewards. However, since this statement holds uniformly for any values of the weights, it holds also when the probability is taken over the distribution of the weights. □

Proof of Lemma 11. Recall that $\widetilde{A}_t^i$ is constructed from the contexts chosen from the first $\tau(t)$ rounds, across all the agents. Let i' and t' be arbitrary indices in V and $\{1,\dots,\tau(t)\}$, respectively.

(i) We have

$$\begin{aligned}\det\left(\widetilde{A}_t^i\right) &= \det\left(\widetilde{A}_t^i - \left(w_{i,t}^{i',t'}-1\right)x_{t'}^{i'}\left(x_{t'}^{i'}\right)^\intercal + \left(w_{i,t}^{i',t'}-1\right)x_{t'}^{i'}\left(x_{t'}^{i'}\right)^\intercal\right)\\ &= \det\left(\widetilde{A}_t^i - \left(w_{i,t}^{i',t'}-1\right)x_{t'}^{i'}\left(x_{t'}^{i'}\right)^\intercal\right)\\ &\quad\cdot\left(1+\left(w_{i,t}^{i',t'}-1\right)\|x_{t'}^{i'}\|_{\left(\widetilde{A}_t^i-\left(w_{i,t}^{i',t'}-1\right)x_{t'}^{i'}\left(x_{t'}^{i'}\right)^\intercal\right)^{-1}}\right)\end{aligned}$$

The second equality follows using the identity $\det(I+cB^{1/2}xx^\intercal B^{1/2}) = (1+c\|x\|_B)$, for any matrix B, vector x, and scalar c. Now, we repeat this process for all $i'\in V$ and $t'\in\{1,\dots,\tau(t)\}$ as follows. Let $(t_1,i_1),\dots,(t_{|V|\tau(t)},i_{|V|\tau(t)})$ be an arbitrary enumeration of $V\times\{1,\dots,\tau(t)\}$, let $B_0=\widetilde{A}_t^i$, and $B_s = B_{s-1}-(w_{i,t}^{i_s,t_s}-1)x_{t_s}^{i_s}\left(x_{t_s}^{i_s}\right)^\intercal$ for $s=1,\dots,|V|\tau(t)$. Then $B_{|V|\tau(t)} = A_{\tau(t)}$, and by the calculation above we have

$$\begin{aligned}\det\left(\widetilde{A}_t^i\right) &= \det\left(A_{\tau(t)}\right)\prod_{s=1}^{|V|\tau(t)}\left(1+\left(w_{i,t}^{i_s,t_s}-1\right)\|x_{t_s}^{i_s}\|_{(B_s)^{-1}}\right)\\ &\le \det\left(A_{\tau(t)}\right)\exp\left(\sum_{s=1}^{|V|\tau(t)}\left(w_{i,t}^{i_s,t_s}-1\right)\|x_{t_s}^{i_s}\|_{(B_s)^{-1}}\right)\\ &\le \exp\left(\sum_{t'=1}^{\tau(t)}\sum_{i'=1}^{|V|}\left|w_{i,t}^{i',t'}-1\right|\right)\det\left(A_{\tau(t)}\right)\end{aligned}$$

[3] $\lambda_{\min}(\,\cdot\,)$ denotes the smallest eigenvalue of its argument.

(ii) Note that for vectors x, y and a matrix B, by the Sherman-Morrison Lemma, and Cauchy-Schwarz inequality we have that:

$$\begin{aligned} x^\intercal(B+yy^\intercal)^{-1}x = x^\intercal B^{-1}x - \frac{x^\intercal B^{-1}yy^\intercal B^{-1}x}{1+y^\intercal B^{-1}y} &\geq x^\intercal B^{-1}x - \frac{x^\intercal B^{-1}xy^\intercal B^{-1}y}{1+y^\intercal B^{-1}y} \\ &= x^\intercal B^{-1}x(1+y^\intercal B^{-1}y)^{-1} \end{aligned} \tag{3.16}$$

Taking

$$B = \left(\widetilde{A}_t^i - \left(w_{i,t}^{i',t'} - 1\right) x_{t'}^{i'} \left(x_{t'}^{i'}\right)^\intercal\right) \text{ and } y = \sqrt{w_{i,t}^{i',t'} - 1}x_{t'}^{i'},$$

and using that $y^\intercal B^{-1}y \leq \lambda_{min}(B)^{-1}y^\intercal y$, by construction, we have that, for any $t' \in \{1, \ldots, \tau(t)\}$ and $i' \in V$,

$$x^\intercal \left(\widetilde{A}_t^i\right)^{-1} x \geq x^\intercal \left(\widetilde{A}_t^i - \left(w_{i,t}^{i',t'} - 1\right) x_{t'}^{i'} \left(x_{t'}^{i'}\right)^\intercal\right)^{-1} x(1+|w_{i,t}^{i',t'} - 1|)^{-1}.$$

Performing this for each $i' \in V$ and $t' \in \{1, \ldots, \tau(t)\}$, taking the exponential of the logarithm and using that $\log(1+a) \leq a$ like in the first part finishes the proof.

□

3.5.4 Proof of Theorem 14

Throughout the proof let i denote the index of some arbitrary but fixed agent, and k the index of its cluster.

Step 1: Show the true clustering is obtained in finite time. First we prove that with probability $1-\delta$, the number of times agents in different clusters share information is bounded. Consider the statements

$$\forall i, i' \in V, \ \forall t, \ \left(\|\hat{\theta}_{local,t}^i - \hat{\theta}_{local,t}^{i'}\| > c_\lambda^{thresh}(t)\right) \implies i' \notin U^k \tag{3.17}$$

and,

$$\forall t \geq C(\gamma, \lambda, \delta) = c_\lambda^{thresh^{-1}}\left(\frac{\gamma}{2}\right), \ i' \notin U^k, \ \|\hat{\theta}_{local,t}^i - \hat{\theta}_{local,t}^{i'}\| > c_\lambda^{thresh}(t). \tag{3.18}$$

where c_λ^{thresh} and A_λ are as defined in the main paper. Lemma 4 from [41] proves that these two statements hold under the assumptions of the theorem with probability $1-\delta/2$.

Let i be an agent in cluster U^k. Suppose that (3.17) and (3.18) hold. Then we know that at time $t = \lceil C(\gamma, \lambda, \delta) \rceil$, $U^k \subset V_t^i$. Moreover, since the sharing protocol chooses an agent uniformly at random from V_t^i independently from the history before time t, it follows that the time until $V_t^i = U^k$ can be upper bounded by a constant $C = C(|V|, \delta)$ with probability $1 - \delta/2$. So it follows that there exists a constant $C = C(|V|, \gamma, \lambda, \delta)$ such that the event

$$E := \{(3.17) \text{ and } (3.18) \text{ hold, and } (t \geq C(|V|, \gamma, \lambda, \delta) \implies V_t^i = U^k)\}$$

holds with probability $1 - \delta$.

Step 2: Consider the properties of the weights after clustering. On the event E, we know that each cluster will be performing the algorithm DCB within its own cluster for all $t > C(\gamma, |V|)$. Therefore, we would like to directly apply the analysis from the proof of Theorem 9 from this point. In order to do this we need to show that the weights, $w_{i,t}^{i',t'}$, have the same properties after time $C = C(\gamma, |V|, \lambda, \delta)$ that are required for the proof of Theorem 9.

Lemma 15. *Suppose that agent i is in cluster U^k. Then, on the event E,*

(i) for all $t > C(|V|, \gamma, \lambda, \delta)$ and $i' \in V \setminus U^k$, $w_{i,t}^{i',t'} = 0$;

(ii) for all $t' \geq C(|V|, \gamma, \lambda, \delta)$ and $i' \in U^k$, $\sum_{i \in U^k} w_{i,C(|V|,\gamma)}^{i',t'} = |U^k|$;

(iii) for all $t \geq t' \geq C(|V|, \gamma, \lambda, \delta)$ and $i' \in U^k$, the weights $w_{i,t}^{i',t'}$, $i \in U^k$, are i.d..

Proof. See Appendix 3.5.5. □

We must deal also with what happens to the information gathered before the cluster has completely discovered itself. To this end, note that we can write, supposing that $\tau(t) \geq C(|V|, \gamma, \lambda, \delta)$,

$$\widetilde{A}_t^i := \sum_{i' \in U^k} \frac{w_{i,t}^{i',C}}{|U^k|} \widetilde{A}_C^{i'} + \sum_{t'=C+1}^{\tau(t)} \sum_{i' \in U^k} w_{i,t}^{i',t'} x_{t'}^{i'} \left(x_{t'}^{i'}\right)^{\mathsf{T}}. \tag{3.19}$$

Armed with this observation we show that the fact that sharing within the appropriate cluster only begins properly after time $C = C(|V|, \gamma, \lambda, \delta)$ the influence of the bias is unchanged:

Lemma 16 (Bound on the influence of general weights). *On the event E, for all $i \in V$ and t such that $T(t) \geq C(|V|, \gamma, \lambda, \delta)$,*

(i) $\det\left(\widetilde{A}_t^i\right) \leq \exp\left(\sum_{t'=C}^{\tau(t)} \sum_{i' \in U^k} \left|w_{i,t}^{i',t'} - 1\right|\right) \det\left(A_{\tau(t)}^k\right)$,

(ii) and $\|x_t^i\|^2_{\left(\widetilde{A}_t^i\right)^{-1}} \leq \exp\left(\sum_{t'=C}^{\tau(t)} \sum_{i' \in U^k} \left|w_{i,t}^{i',t'} - 1\right|\right) \|x_t^i\|^2_{\left(A_{\tau(t)}^k\right)^{-1}}$.

Proof. See Appendix 3.5.5. □

The final property of the weights required to prove Theorem 9 is that their variance is diminishing geometrically with each iteration. For the analysis of DCB this is provided by Lemma 4 of [96], and, using Lemma 15, we can prove the same result for the weights after time $C = C(|V|, \gamma, \lambda, \delta)$:

Lemma 17. *Suppose that agent i is in cluster U^k. Then, on the event E, for all $t \geq C = C(|V|, \gamma, \lambda, \delta)$ and $t' < t$, we have*

$$\mathbb{E}\left((w_{i,t}^{j,t'} - 1)^2\right) \leq \frac{|U^k|}{2^{t - \max\{t', C\}}}.$$

Proof. Given the properties proved in Lemma 15, the proof is identical to the proof of Lemma 4 of [96]. □

Step 3: Apply the results from the analysis of DCB. We can now apply the same argument as in Theorem 9 to bound the regret after time $C = C(\gamma, |V|, \lambda, \delta)$. The regret before this time we simply upper bound by $|U^k| C(|V|, \gamma, \lambda, \delta)\|\theta\|$. We include the modified sections bellow as needed.

Using Lemma 17, we can control the random exponential constant in Lemma 16, and the upper bound $W(T)$:

Lemma 18 (Bound in the influence of weights under our sharing protocol)**.** *Assume that $t \geq C(\gamma, |V|, \lambda\delta)$. Then on the event E, for some constants $0 < \delta_{t'} < 1$, with probability $1 - \sum_{t'=1}^{\tau(t)} \delta_{t'}$*

$$\sum_{t'=C}^{\tau(t)} \sum_{i' \in U^k} \left|w_{i,t}^{i',t'} - 1\right| \leq |U^k|^{\frac{3}{2}} \sum_{t'=C}^{\tau(t)} \sqrt{\frac{2^{-(t - \max\{t', C\})}}{\delta_{t'}}},$$

$$\textit{and } W(\tau(t)) \leq 1 + \max_{C \leq t' \leq \tau(t)} \left\{ |U^k|^{\frac{3}{2}} \sqrt{\frac{2^{-(t - \max\{t', C\})}}{\delta_{t'}}} \right\}.$$

In particular, for any $1 > \delta > 0$, choosing $\delta_{t'} = \delta 2^{-(t - \max\{t', C\})/2}$, and $\tau(t) = t - c_1 \log_2 c_2 t$ we conclude that with probability $1 - (c_2 t)^{-c_1/2} \delta / (1 - 2^{-1/2})$, for any $t > C + c_1 \log_2(c_2 C)$,

$$\sum_{i' \in U^k} \sum_{t'=C}^{\tau(t)} \left|w_{i,t}^{i',t'} - 1\right| \leq \frac{|U^k|^{\frac{3}{2}} (c_2 t)^{-\frac{c_1}{4}}}{(1 - 2^{-\frac{1}{4}})\sqrt{\delta}}, \text{ and } W(\tau(t)) \leq 1 + \frac{|U^k|^{\frac{3}{2}} (c_2 t)^{-\frac{c_1}{4}}}{\sqrt{\delta}}. \tag{3.20}$$

Thus lemmas 16 and 18 give us control over the bias introduced by the imperfect information sharing. Applying lemmas 16 and 18, we find that with probability

$1-(c_2t)^{-c_1/2}\delta/(1-2^{-1/2})$:

$$\rho_t^i \le 2\exp\left(\frac{|U^k|^{\frac{3}{2}}}{(1-2^{-\frac{1}{4}})c_2^{\frac{c_1}{4}}t^{\frac{c_1}{4}}\sqrt{\delta}}\right)\|x_t^i\|_{\left(A^i_{\tau(t)}\right)^{-1}} \tag{3.21}$$
$$\cdot\left[\left(1+\frac{|U^k|^{\frac{3}{2}}}{(1-2^{-\frac{1}{4}})c_2^{\frac{c_1}{4}}t^{\frac{c_1}{4}}\sqrt{\delta}}\right)\left[R\sqrt{2\log\left(\exp\left(\frac{|U^k|^{\frac{3}{2}}}{(1-2^{-\frac{1}{4}})c_2^{\frac{c_1}{4}}t^{\frac{c_1}{4}}\sqrt{\delta}}\right)\frac{\det\left(A_{\tau(t)}\right)^{\frac{1}{2}}}{\delta}\right)}+\|\theta\|\right]\right].$$

Step 4: Choose constants and sum the simple regret. Choosing again $c_1 = 4$, $c_2 = |V|^{\frac{3}{2}}$, and setting $N_\delta = \frac{1}{(1-2^{-\frac{1}{4}})\sqrt{\delta}}$, we have on the event E, for all $t \ge \max\{N_\delta, C + 4\log_2(|V|^{\frac{3}{2}}C)\}$, with probability $1-(|V|t)^{-2}\delta/(1-2^{-1/2})$

$$\rho_t^i \le 4e\|x_t^i\|_{\left(A^k_{t-1}+\sum_{i'=1}^{i-1}x_t^{i'}\left(x_t^{i'}\right)^\intercal\right)^{-1}}\left(\beta(t)+R\sqrt{2}\right),$$

where $\beta(\cdot)$ is as defined in the theorem statement. Now applying Cauchy-Schwarz, and Lemma 11 from [1] yields that on the event E, with probability $1-\left(1+\sum_{t=1}^{\infty}(|V|t)^{-2}/(1-2^{-1/2})\right)\delta \ge 1-3\delta$,

$$\mathcal{R}_t \le \left(\max\{N_\delta, C+4\log_2(|V|^{\frac{3}{2}}C)\}+2\left(4|V|d\log\left(|V|t\right)\right)^3\right)\|\theta\|_2$$
$$+4e\left(\beta(t)+R\sqrt{2}\right)\sqrt{|U^k|t\left(2\log\left(\det\left(A_t^k\right)\right)\right)}.$$

Replacing δ with $\delta/6$, and combining this result with Step 1 finishes the proof.

3.5.5 Proofs of Intermediary Results for DCCB

Proof of Lemma 15. Recall that whenever the pruning procedure cuts an edge, both agents reset their buffers to their local information, scaled by the size of their current neighbour sets. (It does not make a difference practically whether or not they scale their buffers, as this effect is washed out in the computation of the confidence bounds and the local estimates. However, it is convenient to assume that they do so for the analysis.) Furthermore, according to the pruning procedure, no agent will share information with another agent that does not have the same local neighbour set.

On the event E, there is a time for each agent, i, before time $C = C(\gamma, |V|, \lambda\delta)$ when the agent resets its information to their local information, and their local neighbour set becomes their local cluster, i.e. $V_t^i = U^k$. After this time, this agent will only share information with other agents that have also set their local neighbour set to their local cluster. This proves the statement of part (i).

Furthermore, since on event E, after agent i has identified its local neighbour set, i.e. when $V_t^i = U^k$, the agent only shares with members of U^k, the statements of parts (ii) and (iii) hold by construction of the sharing protocol. □

Proof of Lemma 16. The result follows the proof of Lemma 11. For the the iterations until time $C = C(\gamma, |V|, \lambda\delta)$ is reached, we apply the argument there. For the final step we require two further inequalities.

First, to finish the proof of part (i) we note that,

$$\det\left((A_T^k - A_C^k) + \sum_{i'\in U^k} \frac{w_{i,t}^{i',C(\gamma,|V|)}}{|U^k|}\widetilde{A}_C^{i'}\right) = \det\left(A_T^k + \sum_{i'\in U^k} \frac{w_{i,t}^{i',C} - 1}{|U^k|}\widetilde{A}_C^{i'}\right)$$
$$= \det\left(A_T^k\right)\det\left(I + \sum_{i'\in U^k} \frac{w_{i,t}^{i',C} - 1}{|U^k|} {A_T^k}^{-\frac{1}{2}}\widetilde{A}_C^{i'}{A_T^k}^{-\frac{1}{2}}\right)$$
$$\leq \det\left(A_T^k\right)\det\left(I + \left[\sum_{i'\in U^k}\left|w_{i,t}^{i',C} - 1\right|\right]{A_T^k}^{-\frac{1}{2}}\sum_{i'\in U^k}\frac{\widetilde{A}_C^{i'}}{|U^k|}{A_T^k}^{-\frac{1}{2}}\right)$$
$$\leq \det\left(A_T^k\right)\left(1 + \sum_{i'\in U^k}\left|w_{i,t}^{i',C} - 1\right|\right).$$

For the first equality we have used that $|U^k|A_C^k = \sum_{i'\in U^k}\widetilde{A}_C^{i'}$; for the first inequality we have used a property of positive definite matrices; for the second inequality we have used that 1 upper bounds the eigenvalues of ${A_T^k}^{-1/2}A_C^k{A_T^k}^{-1/2}$.

Second, to finish the proof of part (ii), we note that, for any vector x,

$$x^\intercal\left(A_{\tau(t)}^k + \sum_{i'\in U^k}\frac{w_{i,t}^{i',C} - 1}{|U^k|}\widetilde{A}_C^{i'}\right)^{-1} x$$
$$= \left({A_{\tau(t)}^k}^{-\frac{1}{2}}x\right)^\intercal\left(I + \sum_{i'\in U^k}\frac{w_{i,t}^{i',C} - 1}{|U^k|}{A_{\tau(t)}^k}^{-\frac{1}{2}}\widetilde{A}_C^{i'}{A_{\tau(t)}^k}^{-\frac{1}{2}}\right)^{-1}\left({A_{\tau(t)}^k}^{-\frac{1}{2}}x\right)$$
$$\geq \left({A_{\tau(t)}^k}^{-\frac{1}{2}}x\right)^\intercal\left(I + \sum_{i'\in U^k}\frac{\left|w_{i,t}^{i',C} - 1\right|}{|U^k|}{A_{\tau(t)}^k}^{-\frac{1}{2}}\widetilde{A}_C^{i'}{A_{\tau(t)}^k}^{-\frac{1}{2}}\right)^{-1}\left({A_{\tau(t)}^k}^{-\frac{1}{2}}x\right)$$
$$\geq \left(1 + \sum_{i'\in U^k}\left|w_{i,t}^{i',C} - 1\right|\right)^{-1} x^\intercal {A_{\tau(t)}^k}^{-1}x.$$

The first inequality here follows from a property of positive definite matrices, and the other steps follow similarly to those in the inequality that finished part (i) of the proof. □

Chapter 4

Collaborative Clustering Bandits

4.1 Introduction

Recommender Systems are an essential part of many successful on-line businesses, from e-commerce to on-line streaming, and beyond [44, 46]. Moreover, Computational Advertising can be seen as a recommendation problem where the user preferences highly depend on the current *context*. In fact, many recommendation domains such as Youtube video recommendation or news recommendation *do not* fit the classical description of a recommendation scenario, whereby a set of users with essentially fixed preferences interact with a fixed set of items. In this classical setting, the well-known *cold-start* problem, namely, the lack of accumulated interactions by users on items, needs to be addressed, for instance, by turning to *hybrid* recommendation methods (e.g., [45]). In practice, many relevant recommendation domains are dynamic, in the sense that user preferences and the set of active users change with time. Recommendation domains can be distinguished by how much and how often user preferences and content universe change (e.g., [74]). In highly dynamic recommendation domains, such as news, ads and videos, active users and user preferences are fluid, hence classical collaborative filtering-type methods, such as Matrix or Tensor-Factorization break down. In these settings, it is essential for the recommendation method to adapt to the shifting preference patterns of the users.

Exploration-exploitation methods, a.k.a. the multi-armed bandits, which have been shown to be an excellent solution for these dynamic domains (see, e.g., [72, 73]). While effective, standard contextual bandits do not take collaborative information into account, that is, users who have interacted with similar items in the past will not be deemed to have similar taste based on this fact alone, while items that have been chosen by the same group of users will also not be considered as similar. It is this significant limitation in the current bandit methodology that we try to address in this work. Past efforts on this problem were based on using online clustering-like algorithms on the graph or network structure of the data in conjunction with multi-armed bandit methods (see Section 4.3).

Commercial large scale search engines and information retrieval systems are examples of highly dynamic environments where users and items could be described in terms of their membership in some preference cluster. For instance, in a music recommendation scenario, we may have groups of listeners (the users) clustered around music genres, with the clustering changing across different genres. On the other hand, the individual songs (the items) could naturally be grouped by sub-genre or performer based on the fact that they tend to be preferred by the same group of users. Evidence has been collected which suggests that, at least in specific recommendation scenarios, like movie recommendation, data are well modeled by clustering at both user and item sides (e.g., [95]).

In this paper, we introduce a Collaborative Filtering based stochastic multi-armed bandit method that allows for a flexible and generic integration of information of users and items interaction data by alternatively clustering over both user and item sides. Specifically, we describe and analyze an adaptive and efficient clustering of bandit algorithm that can perform collaborative filtering, named COFIBA (pronounced as "coffee bar"). Importantly enough, the clustering performed by our algorithm relies on sparse graph representations, avoiding expensive matrix factorization techniques. We adapt COFIBA to the standard setting of sequential content recommendation known as (contextual) multi-armed bandits (e.g., [5]) for solving the canonical exploration vs. exploitation dilemma.

Our algorithm works under the assumption that we have to serve content to users in such a way that each content *item* determines a clustering over users made up of relatively few groups (compared to the total number of users), within which users tend to react similarly when that item gets recommended. However, the clustering over users need not be the same across different items. Moreover, when the universe of items is large, we also assume that the items might be clustered as a function of the clustering they determine over users, in such a way that the number of *distinct* clusterings over users induced by the items is also relatively small compared to the total number of available items.

Our method aims to exploit collaborative effects in a bandit setting in a way akin to the way co-clustering techniques are used in batch collaborative filtering. Bandit methods also represent one of the most promising approaches to the research community of recommender systems, for instance in tackling the cold-start problem (e.g., [97]), whereby the lack of data on new users leads to suboptimal recommendations. An exploration approach in these cases seems very appropriate.

We demonstrate the efficacy of our dynamic clustering algorithm on three benchmark and real-world datasets. Our algorithm is scalable and exhibits significant increased prediction performance over the state-of-the-art of clustering bandits. We also provide a regret analysis of the $\sqrt{T}$-style holding with high probability in a standard stochastically linear noise setting.

4.2 Learning Model

We assume that the user behavior similarity is encoded by a family of clusterings depending on the specific feature (or context, or item) vector $\boldsymbol{x}$ under consideration. Specifically, we let $\mathcal{U} = \{1, \ldots, n\}$ represent the set of n users. Then, given $\boldsymbol{x} \in \mathbb{R}^d$, set $\mathcal{U}$ can be partitioned into a small number $m(\boldsymbol{x})$ of clusters $U_1(\boldsymbol{x}), U_2(\boldsymbol{x}), \ldots, U_{m(\boldsymbol{x})}(\boldsymbol{x})$, where $m(\boldsymbol{x})$ is upper bounded by a constant m, independent of $\boldsymbol{x}$, with m being much smaller than n. (The assumption $m << n$ is not strictly required but it makes our algorithms more effective, and this is actually what we expect our datasets to comply with.) The clusters are such that users belonging to the same cluster $U_j(\boldsymbol{x})$ tend to have similar behavior w.r.t. feature vector $\boldsymbol{x}$ (for instance, they both like or both dislike the item represented by $\boldsymbol{x}$), while users lying in different clusters have significantly different behavior. The mapping $\boldsymbol{x} \to \{U_1(\boldsymbol{x}), U_2(\boldsymbol{x}), \ldots, U_{m(\boldsymbol{x})}(\boldsymbol{x})\}$ specifying the actual partitioning of the set of users $\mathcal{U}$ into the clusters determined by $\boldsymbol{x}$ (including the number of clusters $m(\boldsymbol{x})$ and its upper bound m), as well as the common user behavior within each cluster are *unknown* to the learning system, and have to be inferred based on user feedback.

For the sake of simplicity, this paper takes the simple viewpoint that clustering over users is determined by linear functions $\boldsymbol{x} \to \boldsymbol{u}_i^\top \boldsymbol{x}$, each one parameterized by an unknown vector $\boldsymbol{u}_i \in \mathbb{R}^d$ hosted at user $i \in \mathcal{U}$, in such a way that if users i and i' are in the same cluster w.r.t. $\boldsymbol{x}$ then $\boldsymbol{u}_i^\top \boldsymbol{x} = \boldsymbol{u}_{i'}^\top \boldsymbol{x}$, while if i and i' are in different clusters w.r.t. $\boldsymbol{x}$ then $|\boldsymbol{u}_i^\top \boldsymbol{x} - \boldsymbol{u}_{i'}^\top \boldsymbol{x}| \geq \gamma$, for some (unknown) gap parameter $\gamma > 0$, independent of $\boldsymbol{x}$.[1] As in the standard linear bandit setting (e.g., [5, 69, 25, 1, 27, 64, 91, 106, 34, 41], and references therein), the unknown vector $\boldsymbol{u}_i$ determines the (average) behavior of user i. More concretely, upon receiving context vector $\boldsymbol{x}$, user i "reacts" by delivering a payoff value

$$a_i(\boldsymbol{x}) = \boldsymbol{u}_i^\top \boldsymbol{x} + \epsilon_i(\boldsymbol{x}) ,$$

where $\epsilon_i(\boldsymbol{x})$ is a conditionally zero-mean and bounded variance noise term so that, conditioned on the past, the quantity $\boldsymbol{u}_i^\top \boldsymbol{x}$ is indeed the expected payoff observed at user i for context vector $\boldsymbol{x}$. Notice that the unknown parameter vector $\boldsymbol{u}_i$ we associate with user i is supposed to be time invariant in this model.[2]

Since we are facing sequential decision settings where the learning system needs to continuously adapt to the newly received information provided by users, we assume that the learning process is broken up into a discrete sequence of rounds: In round $t = 1, 2, \ldots$, the learner receives a user index $i_t \in \mathcal{U}$ to serve content to, hence the user to serve may change at every round, though the same user can recur many times. We assume the sequence of users $i_1, i_2, \ldots$ is determined by an exogenous process that places nonzero and independent probability to each user

[1] As usual, this assumption may be relaxed by assuming the existence of two thresholds, one for the within-cluster distance of $\boldsymbol{u}_i^\top \boldsymbol{x}$ to $\boldsymbol{u}_{i'}^\top \boldsymbol{x}$, the other for the between-cluster distance.

[2] It would in fact be possible to lift this whole machinery to time-drifting user preferences by combining with known techniques (e.g., [21, 79]).

being the next one to serve. Together with i_t, the system receives in round t a set of feature vectors $C_{i_t} = \{\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \boldsymbol{x}_{t,c_t}\} \subseteq \mathbb{R}^d$ encoding the content which is currently available for recommendation to user i_t. The learner is compelled to pick some $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_{t,k_t} \in C_{i_t}$ to recommend to i_t, and then observes i_t's feedback in the form of payoff $a_t \in \mathbb{R}$ whose (conditional) expectation is $\boldsymbol{u}_{i_t}^\top \bar{\boldsymbol{x}}_t$. The goal of the learning system is to maximize its total payoff $\sum_{t=1}^T a_t$ over T rounds. When the user feedback at our disposal is only the click/no-click behavior, the payoff a_t is naturally interpreted as a binary feedback, so that the quantity $\frac{\sum_{t=1}^T a_t}{T}$ becomes a clickthrough rate (CTR), where $a_t = 1$ if the recommended item was clicked by user i_t, and $a_t = 0$, otherwise. CTR is the measure of performance adopted by our comparative experiments in Section 4.5.

From a theoretical standpoint (Section 4.6), we are instead interested in bounding the cumulative *regret* achieved by our algorithms. More precisely, let the regret r_t of the learner at time t be the extent to which the average payoff of the best choice in hindsight at user i_t exceeds the average payoff of the algorithm's choice, i.e.,

$$r_t = \left(\max_{\boldsymbol{x} \in C_{i_t}} \boldsymbol{u}_{i_t}^\top \boldsymbol{x}\right) - \boldsymbol{u}_{i_t}^\top \bar{\boldsymbol{x}}_t \,.$$

We are aimed at bounding with high probability the cumulative regret $\sum_{t=1}^T r_t$, the probability being over the noise variables $\epsilon_{i_t}(\bar{\boldsymbol{x}}_t)$, and any other possible source of randomness, including i_t – see Section 4.6.

The kind of regret bound we would like to contrast to is one where the latent clustering structure over $\mathcal{U}$ (w.r.t. the feature vectors $\boldsymbol{x}$) is somehow known beforehand (see Section 4.6 for details). When the content universe is large but known a priori, as is frequent in many collaborative filtering applications, it is often desirable to also group the items into clusters based on similarity of user preferences, i.e., two items are similar if they are preferred by many of the same users. This notion of "two-sided" clustering is well known in the literature; when the clustering process is simultaneously grouping users based on similarity at the item side and items based on similarity at the user side, it goes under the name of "co-clustering" (see, e.g., [32, 33]). Here, we consider a computationally more affordable notion of collaborate filtering based on adaptive two-sided clustering.

Unlike previous existing clustering techniques on bandits (e.g., [41, 82]), our clustering setting only applies to the case when the content universe is large but known a priori (yet, see the end of Section 4.4). Specifically, let the content universe be $\mathcal{I} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|\mathcal{I}|}\}$, and $P(\boldsymbol{x}_h) = \{U_1(\boldsymbol{x}_h), U_2(\boldsymbol{x}_h), \ldots, U_{m(\boldsymbol{x}_h)}(\boldsymbol{x}_h)\}$ be the partition into clusters over the set of users $\mathcal{U}$ induced by item $\boldsymbol{x}_h$. Then items $\boldsymbol{x}_h, \boldsymbol{x}_{h'} \in \mathcal{I}$ belong to the same cluster (over the set of items $\mathcal{I}$) if and only if they induce the same partition of the users, i.e., if $P(\boldsymbol{x}_h) = P(\boldsymbol{x}_{h'})$. We denote by g the number of distinct partitions so induced over $\mathcal{U}$ by the items in $\mathcal{I}$, and work under the assumption that g is *unknown* but significantly smaller than $|\mathcal{I}|$. (Again, the assumption $g << |\mathcal{I}|$ is not strictly needed, but it both makes our algorithms effective and is expected to be satisfied in relevant practical scenarios.)

Finally, in all of the above, an important special case is when the items to be recommended do not possess specific features (or do not possess features having significant predictive power). In this case, it is common to resort to the more classical non-contextual stochastic multiarmed bandit setting (e.g., [6, 4]), which is recovered from the contextual framework by setting $d = |\mathcal{I}|$, and assuming the content universe $\mathcal{I}$ is made up of the d-dimensional vectors $\boldsymbol{e}_h, h = 1, \ldots, d$, of the canonical basis of $\mathbb{R}^d$, As a consequence, the expected payoff of user i on item h is simply the h-th component of vector $\boldsymbol{u}_i$, and two users i and i' belong to the same cluster w.r.t. to h if the h-th component of $\boldsymbol{u}_i$ equals the h-th component of $\boldsymbol{u}_{i'}$. Because the lack of useful annotation on data was an issue with all datasets at our disposal, it is this latter modeling assumption that motivates the algorithm we actually implemented for the experiments reported in Section 4.5.

4.3 Related Work

Batch collaborative filtering neighborhood methods rely on finding similar groups of users and items to the target user-item pair, e.g., [103], and thus in effect rely on a dynamic form of grouping users and items. Collaborative Filtering-based methods have also been integrated with co-clustering techniques, whereby preferences in each co-cluster are modeled with simple statistics of the preference relations in the co-cluster, e.g., rating averages [42].

Beyond the general connection to co-clustering (e.g., [32, 33]), our paper is related to the research on multi-armed bandit algorithms for trading off exploration and exploitation through dynamic clustering. We are not aware of any specific piece of work that combines bandits with co-clustering based on the scheme of collaborative filtering; the papers which are most closely related to ours are [34, 76, 82, 13, 65, 41, 63]. In [34], the authors work under the assumption that users are defined using a feature vector, and try to learn a low-rank hidden subspace assuming that variation across users is low-rank. The paper combines low-rank matrix recovery with high-dimensional Gaussian Process Bandits, but it gives rise to algorithms which do not seem practical for sizeable problems. In [76], the authors analyze a non-contextual stochastic bandit problem where model parameters are assumed to be clustered in a few (unknown) types. Yet, the provided solutions are completely different from ours. The work [82] combines (k-means-like) online clustering with a contextual bandit setting, but clustering is only made at the user side. The paper [13] also relies on bandit clustering at the user side (as in [76, 82]), with an emphasis on diversifying recommendations to the same user over time. In [65], the authors propose cascading bandits of user behavior to identify the k most attractive items, and formulate it as a stochastic combinatorial partial monitoring problem. Finally, the algorithms in [41, 63, 71] can be seen as a special case of COFIBA when clustering is done only at the user side, under centralized [41, 71] or decentralized [63] environments.

Similar in spirit are also [7, 14, 18, 60]: In [7], the authors define a transfer learning problem within a stochastic multi-armed bandit setting, where a prior distribution is defined over the set of possible models over the tasks; in [14], the authors rely on clustering Markov Decision Processes based on their model parameter similarity. In [18], the authors discuss how to choose from n unknown distributions the k ones whose means are largest by a certain metric; in [60] the authors study particle Thompson sampling with Rao-Blackwellization for online matrix factorization, exhibiting a regret bound in a very specific case of $n \times m$ rank-1 matrices. Yet, in none of above cases did the authors make a specific effort towards item-dependent clustering models applied to stochastic multi-armed bandits.

Further work includes [97, 98]. In [97], an ensemble of contextual bandits is used to address the cold-start problem in recommender systems. A similar approach is used in [98] to deal with cold-start in recommender systems but based on the probability matching paradigm in a parameter-free bandit strategy, which employs online bootstrap to derive the distribution of the estimated models. In contrast to our work, in neither [97] nor [98] are collaborative effects explicitly taken into account.

4.4 The Algorithm

COFIBA, relies on upper-confidence-based tradeoffs between exploration and exploitation, combined with adaptive clustering procedures at both the user and the item sides. COFIBA stores in round t an estimate $\boldsymbol{w}_{i,t}$ of vector $\boldsymbol{u}_i$ associated with user $i \in \mathcal{U}$. Vectors $\boldsymbol{w}_{i,t}$ are updated based on the payoff feedback, as in a standard linear least-squares approximation to the corresponding $\boldsymbol{u}_i$. Every user $i \in \mathcal{U}$ hosts such an algorithm which operates as a linear bandit algorithm (e.g., [25, 1, 41]) on the available content C_{i_t}. More specifically, $\boldsymbol{w}_{i,t-1}$ is determined by an inverse correlation matrix $M_{i,t-1}^{-1}$ subject to rank-one adjustments, and a vector $\mathbf{b}_{i,t-1}$ subject to additive updates. Matrices $M_{i,t}$ are initialized to the $d \times d$ identity matrix, and vectors $\mathbf{b}_{i,t}$ are initialized to the d-dimensional zero vector. Matrix $M_{i,t-1}^{-1}$ is also used to define an upper confidence bound $\mathrm{CB}_{i,t-1}(\boldsymbol{x})$ in the approximation of $\boldsymbol{w}_{i,t-1}$ to $\boldsymbol{u}_i$ along direction $\boldsymbol{x}$. Based on the local information encoded in the weight vectors $\boldsymbol{w}_{i,t-1}$ and the confidence bounds $\mathrm{CB}_{i,t-1}(\boldsymbol{x})$, the algorithm also maintains and updates a family of clusterings of the set of users $\mathcal{U}$, and a single clustering over the set of items $\mathcal{I}$. On both sides, such clusterings are represented through connected components of undirected graphs (this is in the same vein as in [41]), where nodes are either users or items. A pseudocode description of our algorithm is contained in Figures 4.1, 4.2, and 4.3, while Figure 4.4 illustrates the algorithm's behavior through a pictorial example.

At time t, COFIBA receives the index i_t of the current user to serve, along with the available item vectors $\boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,c_t}$, and must select one among them. In order to do so, the algorithm computes the c_t *neighborhood sets* N_k, one per

```
Input:
  • Set of users $\mathcal{U} = \{1, \ldots, n\}$;
  • set of items $\mathcal{I} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\mathcal{I}|}\} \subseteq \mathbb{R}^d$;
  • exploration parameter $\alpha > 0$, and edge deletion parameter $\alpha_2 > 0$.
Init:
  • $\mathbf{b}_{i,0} = \mathbf{0} \in \mathbb{R}^d$ and $M_{i,0} = I \in \mathbb{R}^{d \times d}$, $i = 1, \ldots n$;
  • User graph $G^U_{1,1} = (\mathcal{U}, E^U_{1,1})$, $G^U_{1,1}$ is connected over $\mathcal{U}$;
  • Number of user graphs $g_1 = 1$;
  • No. of user clusters $m^U_{1,1} = 1$;
  • Item clusters $\hat{I}_{1,1} = \mathcal{I}$, no. of item clusters $g_1 = 1$;
  • Item graph $G^I_1 = (\mathcal{I}, E^I_1)$, $G^I_1$ is connected over $\mathcal{I}$.
for $t = 1, 2, \ldots, T$ do
    Set
        $\boldsymbol{w}_{i,t-1} = M^{-1}_{i,t-1}\mathbf{b}_{i,t-1}, \qquad i = 1, \ldots, n$ ;
    Receive $i_t \in \mathcal{U}$, and get items $C_{i_t} = \{\boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,c_t}\} \subseteq \mathcal{I}$;
    For each $k = 1, \ldots, c_t$, determine which cluster (within the current user clustering
    w.r.t. $\boldsymbol{x}_{t,k}$) user $i_t$ belongs to, and denote this cluster by $N_k$;
    Compute, for $k = 1, \ldots, c_t$, aggregate quantities
        $\bar{M}_{N_k,t-1} = I + \sum_{i \in N_k} (M_{i,t-1} - I)$,
        $\bar{\mathbf{b}}_{N_k,t-1} = \sum_{i \in N_k} \mathbf{b}_{i,t-1}$,
        $\bar{\boldsymbol{w}}_{N_k,t-1} = \bar{M}^{-1}_{N_k,t-1}\bar{\mathbf{b}}_{N_k,t-1}$ ;
    Set $k_t = \operatorname{argmax}_{k=1,\ldots,c_t} \left(\bar{\boldsymbol{w}}^\top_{N_k,t-1}\boldsymbol{x}_{t,k} + \mathrm{CB}_{N_k,t-1}(\boldsymbol{x}_{t,k})\right)$ ,
    where $\mathrm{CB}_{N_k,t-1}(\boldsymbol{x}) = \alpha\sqrt{\boldsymbol{x}^\top \bar{M}^{-1}_{N_k,t-1}\boldsymbol{x}\,\log(t+1)}$;
    Set for brevity $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_{t,k_t}$;
    Observe payoff $a_t \in \mathbb{R}$, and update weights $M_{i,t}$ and $\mathbf{b}_{i,t}$ as follows:
      • $M_{i_t,t} = M_{i_t,t-1} + \bar{\boldsymbol{x}}_t\bar{\boldsymbol{x}}_t^\top$,
      • $\mathbf{b}_{i_t,t} = \mathbf{b}_{i_t,t-1} + a_t\bar{\boldsymbol{x}}_t$,
      • Set $M_{i,t} = M_{i,t-1}$, $\mathbf{b}_{i,t} = \mathbf{b}_{i,t-1}$ for all $i \neq i_t$ ,
    Determine $\widehat{h}_t \in \{1, \ldots, g_t\}$ such that $k_t \in \hat{I}_{\widehat{h}_t,t}$;
    Update user clusters at graph $G^U_{t,\widehat{h}_t} = (\mathcal{U}, E^U_{t,\widehat{h}_t})$ by performing the steps in Figure
    4.2;
    For all $h \neq \widehat{h}_t$, set $G^U_{t+1,h} = G^U_{t,h}$;
    Update item clusters at graph $G^I_t = (\mathcal{I}, E^I_t)$ by performing the steps in Figure 4.3 .
end for
```

Figure 4.1: The COFIBA algorithm.

item $\boldsymbol{x}_{t,k} \in C_{i_t}$ based on the current aggregation of users (clusters "at the user side") w.r.t. item $\boldsymbol{x}_{t,k}$. Set N_k should be regarded as the current approximation to the cluster (over the users) i_t belongs to when the clustering criterion is defined by item $\boldsymbol{x}_{t,k}$. Each neighborhood set then defines a compound weight vector $\bar{\boldsymbol{w}}_{N_k,t-1}$ (through the aggregation of the corresponding matrices $M_{i,t-1}$ and vectors $\mathbf{b}_{i,t-1}$) which, in turn, determines a compound confidence bound[3] $\mathrm{CB}_{N_k,t-1}(\boldsymbol{x}_{t,k})$. Vector $\bar{\boldsymbol{w}}_{N_k,t-1}$ and confidence bound $\mathrm{CB}_{N_k,t-1}(\boldsymbol{x}_{t,k})$ are combined through an upper-confidence exploration-exploitation scheme so as to commit to the specific item $\bar{\boldsymbol{x}}_t \in C_{i_t}$ for user i_t. Then, the payoff a_t is received, and the algorithm uses $\bar{\boldsymbol{x}}_t$ to update $M_{i_t,t-1}$ to $M_{i_t,t}$ and $\mathbf{b}_{i_t,t-1}$ to $\mathbf{b}_{i_t,t}$. Notice that the update is only performed at user i_t, though this will affect the calculation of neighborhood sets and compound vectors for other users in later rounds.

After receiving payoff a_t and computing $M_{i_t,t}$ and $\mathbf{b}_{i_t,t}$, COFIBA updates the clusterings at the user side and the (unique) clustering at the item side. In round t, there are multiple graphs $G^U_{t,h} = (\mathcal{U}, E^U_{t,h})$ at the user side (hence many clusterings over $\mathcal{U}$, indexed by h), and a single graph $G^I_t = (\mathcal{I}, E^I_t)$ at the item side (hence a single clustering over $\mathcal{I}$). Each *clustering* at the user side corresponds to a single *cluster* at the item side, so that we have g_t *clusters* $\hat{I}_{1,t}, \ldots, \hat{I}_{g_t,t}$ over items and g_t *clusterings* over users. See Figure 4.4 for an example where $\mathcal{U} = \{1, \ldots 6\}$ and $\mathcal{I} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_8\}$ (the items are depicted here as $1, 2, \ldots, 8$). **(a)** At the beginning we have $g_1 = 1$, with a single item cluster $\hat{I}_{1,1} = \mathcal{I}$ and, correspondingly, a single (degenerate) clustering over $\mathcal{U}$, made up of the unique cluster $\mathcal{U}$. **(b)** In round t we have the $g_t = 3$ item clusters $\hat{I}_{1,t} = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, $\hat{I}_{2,t} = \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$, $\hat{I}_{3,t} = \{\boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8\}$. Corresponding to each one of them are the three clusterings over $\mathcal{U}$ depicted on the left, so that $m^U_{t,1} = 3$, $m^U_{t,2} = 2$, and $m^U_{t,3} = 4$. In this example, $i_t = 4$, and $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_5$, hence $\widehat{h}_t = 2$, and we focus on graph $G^U_{t,2}$, corresponding to user clustering $\{\{1, 2, 3\}, \{4, 5, 6\}\}$. Suppose in $G^U_{t,2}$ the only neighbors of user 4 are 5 and 6. When updating such user clustering, the algorithm considers therein edges $(4, 5)$ and $(4, 6)$ to be candidates for elimination. Suppose edge $(4, 6)$ is eliminated, so that the new clustering over $\mathcal{U}$ induced by the updated graph $G^U_{t+1,2}$ becomes $\{\{1, 2, 3\}, \{4, 5\}, \{6\}\}$. After user graph update, the algorithm considers the item graph update. Suppose $\boldsymbol{x}_5$ is only connected to $\boldsymbol{x}_4$ and $\boldsymbol{x}_3$ in G^I_t, and that $\boldsymbol{x}_4$ is not connected to $\boldsymbol{x}_3$, as depicted. Both edge $(\boldsymbol{x}_5, \boldsymbol{x}_4)$ and edge $(\boldsymbol{x}_5, \boldsymbol{x}_3)$ are candidates for elimination. The algorithm computes the neighborhood N of $i_t = 4$ according to $G^U_{t+1,2}$, and compares it to the the neighborhoods $N^U_{\ell,t+1}(i_t)$, for $\ell = 3, 4$. Assume $N \neq N^U_{3,t+1}(i_t)$, because the two neighborhoods of user 4 are now different, the algorithm deletes edge $(\boldsymbol{x}_5, \boldsymbol{x}_3)$ from the item graph, splitting the item cluster $\{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$ into the two clusters $\{\boldsymbol{x}_3\}$ and $\{\boldsymbol{x}_4, \boldsymbol{x}_5\}$, hence allocating a new cluster at the item side corresponding to a new degenerate clustering $\{\{1, 2, 3, 4, 5, 6\}\}$ at the user side. **(c)** The resulting clusterings at the beginning of

[3] The one given in Figure 4.1 is the confidence bound we use in our experiments. In fact, the theoretical counterpart to CB is significantly more involved, same efforts can also be found in order to close the gap, e.g., in [4, 41].

Update *user* clusters at graph $G^U_{t,\widehat{h}_t}$ as follows:

- Delete from $E^U_{t,\widehat{h}_t}$ all (i_t, j) such that
$$|\boldsymbol{w}_{i_t,t}^\top \bar{\boldsymbol{x}}_t - \boldsymbol{w}_{j,t}^\top \bar{\boldsymbol{x}}_t| > \text{CB}_{i_t,t}(\bar{\boldsymbol{x}}_t) + \text{CB}_{j,t}(\bar{\boldsymbol{x}}_t)\ ,$$
where $\text{CB}_{i,t}(\boldsymbol{x}) = \alpha_2 \sqrt{\boldsymbol{x}^\top M_{i,t}^{-1}\boldsymbol{x}\,\log(t+1)}$;
- Let $E^U_{t+1,\widehat{h}_t}$ be the resulting set of edges, set $G^U_{t+1,\widehat{h}_t} = (\mathcal{U}, E^U_{t+1,\widehat{h}_t})$, and compute associated clusters $\hat{U}_{1,t+1,\widehat{h}_t}, \hat{U}_{2,t+1,\widehat{h}_t}, \ldots, \hat{U}_{m^U_{t+1,\widehat{h}_t},t+1,\widehat{h}_t}$ as the connected components of $G^U_{t+1,\widehat{h}_t}$.

Figure 4.2: User cluster update in the COFIBA

Update *item* clusters at graph G^I_t as follows:

- For all ℓ such that $(\bar{\boldsymbol{x}}_t, \boldsymbol{x}_\ell) \in E^I_t$ build neighborhood $N^U_{\ell,t+1}(i_t)$ as:
$$N^U_{\ell,t+1}(i_t) = \Big\{ j\ :\ j \neq i_t\,, |\boldsymbol{w}_{i_t,t}^\top \boldsymbol{x}_\ell - \boldsymbol{w}_{j,t}^\top \boldsymbol{x}_\ell| \le \text{CB}_{i_t,t}(\boldsymbol{x}_\ell) + \text{CB}_{j,t}(\boldsymbol{x}_\ell) \Big\}\ ;$$
- Delete from E^I_t all $(\bar{\boldsymbol{x}}_t, \boldsymbol{x}_\ell)$ such that $N^U_{\ell,t+1}(i_t) \neq N^U_{k_t,t+1}(i_t)$, where $N^U_{k_t,t+1}(i_t)$ is the *neighborhood* of node i_t w.r.t. graph $G^U_{t+1,\widehat{h}_t}$;
- Let E^I_{t+1} be the resulting set of edges, set $G^I_{t+1} = (\mathcal{I}, E^I_{t+1})$, compute associated *item* clusters $\hat{I}_{1,t+1}, \hat{I}_{2,t+1}, \ldots, \hat{I}_{g_{t+1},t+1}$ through the connected components of G^I_{t+1};
- For each new item cluster created, allocate a new connected graph over users representing a single (degenerate) cluster $\mathcal{U}$.

Figure 4.3: Item cluster update in the COFIBA

round $t+1$ (In this picture it is assumed that edge $(\boldsymbol{x}_5, \boldsymbol{x}_4)$ was not deleted from the item graph at time t).

On both user and item sides, updates take the form of edge deletions. Updates at the user side are only performed on the graph $G^U_{t,\widehat{h}_t}$ pointed to by the selected item $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_{t,k_t}$. Updates at the item side are only made if it is likely that the neighborhoods of user i_t has significantly changed when considered w.r.t. two previously deemed similar items. Specifically, if item $\boldsymbol{x}_h$ was directly connected to item $\bar{\boldsymbol{x}}_t$ at the beginning of round t and, as a consequence of edge deletion at the user side, the set of users that are now likely to be close to i_t w.r.t. $\boldsymbol{x}_h$ is no longer the same as the set of users that are likely to be close to i_t w.r.t. $\bar{\boldsymbol{x}}_t$, then this is taken as a good indication that item $\boldsymbol{x}_h$ is not inducing the same partition over users as $\bar{\boldsymbol{x}}_t$ does, hence edge $(\bar{\boldsymbol{x}}_t, \boldsymbol{x}_h)$ gets deleted. Notice that this need not imply that, as a result of this deletion, the two items are now belonging to different clusters over $\mathcal{I}$, since these two items may still be indirectly connected.

It is worth stressing that a naive implementation of COFIBA would require memory allocation for maintaining $|\mathcal{I}|$-many n-node graphs, i.e., $\mathcal{O}(n^2\,|\mathcal{I}|)$. Because this would be prohibitive even for moderately large sets of users, we make full usage of the approach of [41], where instead of starting off with complete

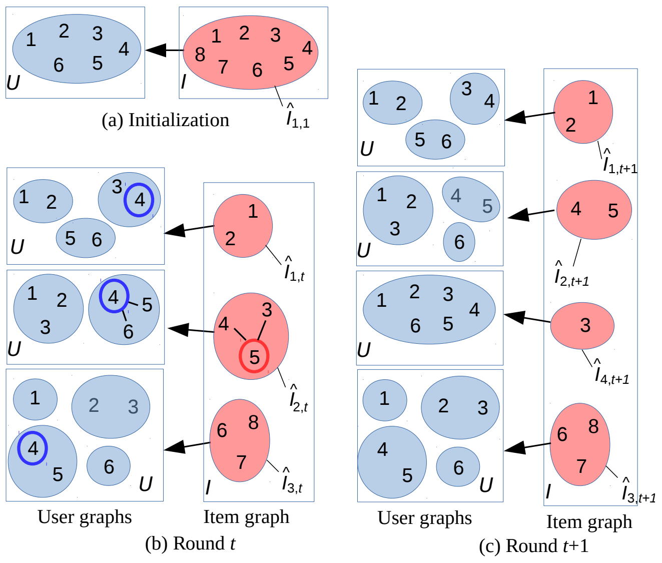


Figure 4.4: Illustration example

graphs over users each time a new cluster over items is created, we randomly sparsify the complete graph by drawing an Erdos-Renyi initial graph, still retaining with high probability the underlying clusterings $\{U_1(\boldsymbol{x}_h), \ldots, U_{m(\boldsymbol{x}_h)}(\boldsymbol{x}_h)\}$, $h = 1, \ldots, |\mathcal{I}|$, over users. This works under the assumption that the latent clusters $U_i(\boldsymbol{x}_h)$ are not too small – see the argument in [41], where it is shown that in practice the initial graphs can have $\mathcal{O}(n \log n)$ edges instead of $\mathcal{O}(n^2)$. Moreover, because we modify the item graph by edge deletions only, one can show that with high probability (under the modeling assumptions of Section 4.2) the number g_t of clusters over items remains upper bounded by g throughout the run of COFIBA, so that the actual storage required by the algorithm is indeed $\mathcal{O}(ng \log n)$. This also brings a substantial saving in running time, since updating connected components scales with the number of edges of the involved graphs. It is this graph sparsification techniques that we used and tested along the way in our experimentation parts.

Finally, despite we have described in Section 4.2 a setting where $\mathcal{I}$ and $\mathcal{U}$ are known a priori (the analysis in Section 4.6 currently holds only in this scenario), nothing prevents in practice to adapt COFIBA to the case when new content or new users show up. This essentially amounts to adding new nodes to the graphs at either the item or the user side, by maintaining data-structures via dynamic memory allocation. In fact, this is precisely how we implemented our algorithm in the case of very big item or user sets (e.g., the Telefonica and the Avazu dataset in the next section).

4.5 Experiments

We compared our algorithm to standard bandit baselines on three real-world datasets: one canonical benchmark dataset on news recommendations, one advertising dataset from a living production system, and one publicly available advertising dataset. In all cases, no features on the items have been used. We closely followed the same experimental setting as in previous work [25, 41], thereby evaluating prediction performance by click-through rate.

4.5.1 Datasets

Yahoo!. The first dataset we use for the evaluation is the freely available benchmark dataset which was released in the "ICML 2012 Exploration & Exploitation Challenge"[4]. The aim of the challenge was to build state-of-the-art news article recommendation algorithms on Yahoo! data, by building an algorithm that learns efficiently a policy to serve news articles on a web site. The dataset is made up of random traffic records of user visits on the "Today Module" of Yahoo!, implying that both the visitors and the recommended news article are selected randomly. The available options (the items) correspond to a set of news articles available for recommendation, one being displayed in a small box on the visited web page. The aim is to recommend an interesting article to the user, whose interest in a given piece of news is asserted by a click on it. The data has 30 million visits over a two-week time stretch. Out of the logged information contained in each record, we used the user ID in the form of a 136-dimensional boolean vector containing his/her features (index i_t), the set of relevant news articles that the system can recommend from (set C_{i_t}); a randomly recommended article during the visit; a boolean value indicating whether the recommended article was clicked by the visiting user or not (payoff a_t). Because the displayed article is chosen uniformly at random from the candidate article pool, one can use an unbiased off-line evaluation method to compare bandit algorithms in a reliable way. We refer the reader to [41] for a more detailed description of how this dataset was collected and extracted. We picked the larger of the two datasets considered in [41], resulting in $n \approx 18K$ users, and $d = 323$ distinct items. The number of records ended up being $2.8M$, out of which we took the first $300K$ for parameter tuning, and the rest for testing.

Telefonica. This dataset was obtained from Telefonica S.A., which is the number one Spanish broadband and telecommunications provider, with business units in Europe and South America. This data contains clicks on ads displayed to user on one of the websites that Telefonica operates on. The data were collected from the back-end server logs, and consist of two files: the first file contains the ads interactions (each record containing an impression timestamp, a user-ID, an action, the ad type, the order item ID, and the click timestamp); the second file contains the ads metadata as item-ID, type-ID, type, order-ID, creative type, mask, cost, creator-ID, transaction key, cap type. Overall, the number n of users was in the

[4]https://explochallenge.inria.fr/category/challenge

scale of millions, while the number d of items was approximately 300. The data contains $15M$ records, out of which we took the first $1, 5M$ for parameter tuning, and the rest for testing. Again, the only available payoffs are those associated with the items served by the system. Hence, in order to make the procedure be an effective estimator in a sequential decision process (e.g., [27, 36, 41, 69]), we *simulated* random choices by the system by generating the available item sets C_{i_t} as follows: At each round t, we stored the ad served to the current user i_t and the associated payoff value a_t (1 ="clicked", 0 ="not clicked"). Then we created C_{i_t} by including the served ad along with 9 extra items (hence $c_t = 10 \, \forall t$) which were drawn uniformly at random in such a way that, for any item $\boldsymbol{e}_h \in \mathcal{I}$, if $\boldsymbol{e}_h$ occurs in some set C_{i_t}, this item will be the one served by the system $1/10$ of the times. The random selection was done independent of the available payoff values a_t. All our experiments on this dataset were run on a machine with 64GB RAM and 32 Intel Xeon cores.

Avazu. This dataset was prepared by Avazu Inc,[5] which is a leading multinational corporation in the digital advertising business. The data was provided for the challenge to predict the click-through rate of impressions on mobile devices, i.e., whether a mobile ad will be clicked or not. The number of samples was around $40M$, out of which we took the first $4M$ for parameter tuning, and the remaining for testing. Each line in the data file represents the event of an ad impression on the site or in a mobile application (app), along with additional context information. Again, payoff a_t is binary. The variables contained in the dataset for each sample are the following: ad-ID; timestamp (date and hour); click (boolean variable); device-ID; device IP; connection type; device type; ID of visited App/Website; category of visited App/Website; connection domain of visited App/Website; banner position; anonymized categorical fields (C1, C14-C21). We pre-processed the dataset as follows: we cleaned up the data by filtering out the records having missing feature values, and removed outliers. We identified the user with device-ID, if it is not null. The number of users on this dataset is in the scale of millions. Similar to the Telefonica dataset, we generated recommendation lists of length $c_t = 20$ for each distinct timestamp. We used the first $4M$ records for tuning parameters, and the remaining $36M$ for testing. All data were transferred to Amazon S3, and all jobs were run through the Amazon EC2 Web Service.

4.5.2 Algorithms

We compared COFIBA to a number of state-of-the-art bandit algorithms:

- LINUCB-ONE is a single instance of the UCB1 [6] algorithm, which is a very popular and established algorithm that has received a lot of attention in the research community over the past years;
- DYNUCB is the dynamic UCB algorithm of [82]. This algorithm adopts a "K-means"-like clustering technique so as to dynamically re-assign the

[5]https://www.kaggle.com/c/avazu-ctr-prediction

clusters on the fly based on the changing contexts and user preferences over time;

- LINUCB-IND [41] is a set of independent UCB1 instances, one per user, which provides a fully personalized recommendation for each user;
- CLUB [41] is the state-of-the-art online clustering of bandits algorithm that dynamically cluster users based on the confidence ellipsoids of their models;
- LINUCB-V [4] is also a single instance of UCB1, but with a more sophisticated confidence bound; this algorithm turned out to be the winner of the "ICML 2012 Challenge" where the Yahoo! dataset originates from.

We tuned the optimal parameters in the training set with a standard grid search as indicated in [27, 41], and used the test set to evaluate the predictive performance of the algorithms. Since the system's recommendation need not coincide with the recommendation issued by the algorithms we tested, we only retained the records on which the two recommendations were indeed the same. Because records are discarded on the fly, the actual number T of retained records ("Rounds" in the plots of the next subsection) changes slightly across algorithms; T was around $70K$ for the Yahoo! data, $350K$ for the Telefonica data, and $900K$ for the Avazu data. All experimental results we report were averaged over 3 runs (but in fact the variance we observed across these runs was fairly small).

4.5.3 Results

Our results are summarized in Figures 4.5, 4.6, and 4.7. Further evidence is contained in Figure 4.8. In Figures 4.5–4.7, we plotted click-through rate ("CTR") vs. retained records so far ("Rounds"). All these experiments are aimed at testing the performance of the various bandit algorithms in terms of prediction performance, also in cold-start regimes (i.e., the first relatively small fraction of the time horizon in the x-axis). Our experimental setting is in line with previous ones (e.g., [25, 41]) and, by the way the data have been prepared, gives rise to a reliable estimation of actual CTR behavior under the same experimental conditions as in [25, 41]. Figure 4.8 is aimed at supporting the theoretical model of Section 4.2, by providing some evidence on the kind of clustering statistics produced by COFIBA at the end of its run.

Whereas the three datasets we took into consideration are all generated by real online web applications, it is worth pointing out that these datasets are indeed different in the way customers consume the associated content. Generally speaking, the longer the lifecycle of one item the fewer the items, the higher the chance that users with similar preferences will consume it, and hence the bigger the collaborative effects contained in the data. It is therefore reasonable to expect that our algorithm will be more effective in datasets where the collaborative effects are indeed strong.

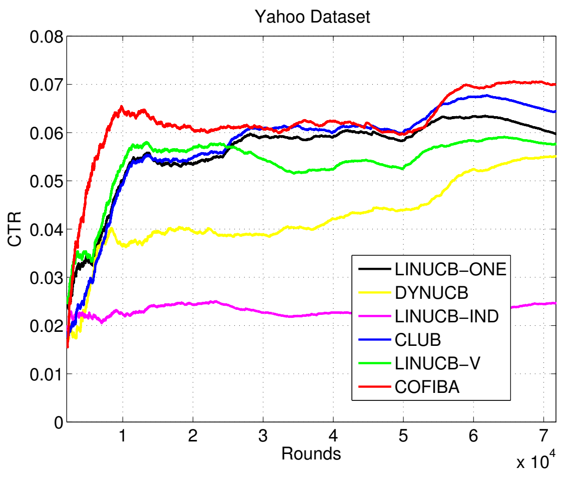


Figure 4.5: Results on the Yahoo dataset.

The users in the Yahoo! data (Figure 4.5), are likely to span a wide range of demographic characteristics; on top of this, this dataset is derived from the consumption of news that are often interesting for large portions of these users and, as such, do not create strong polarization into subcommunities. This implies that more often than not, there are quite a few specific hot news that all users might express interest in, and it is natural to expect that these pieces of news are intended to reach a wide audience of consumers. Given this state of affairs, it is not surprising that on the Yahoo! dataset both LINUCB-ONE and LINUCB-V (serving the same news to all users) are already performing quite well, thereby making the clustering-of-users effort somewhat less useful. This also explains the poor performance of LINUCB-IND, which is not performing any clustering at all. Yet, even in this non-trivial case, COFIBA can still achieve a significant increased prediction accuracy compared, e.g., to CLUB, thereby suggesting that simultaneous clustering at both the user and the item (the news) sides might be an even more effective strategy to earn clicks in news recommendation systems.

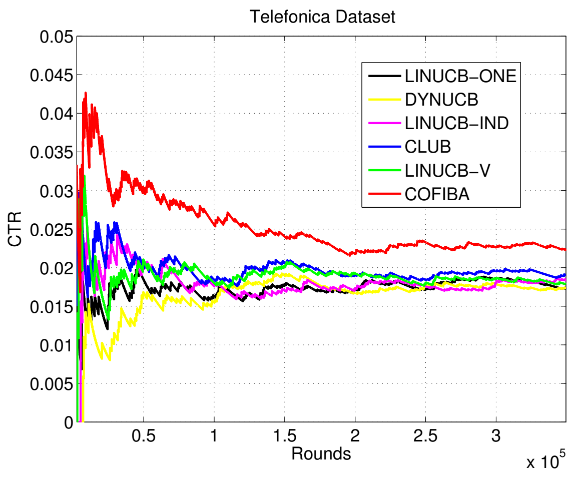


Figure 4.6: Results on the Telefonica dataset.

Most of the users in the Telefonica data are from a diverse sample of people in Spain, and it is easy to imagine that this dataset spans a large number of communities across its population. Thus we can assume that collaborative effects will be much more evident, and that COFIBA will be able to leverage these effects efficiently. In this dataset, CLUB performs well in general, while DYNUCB deteriorates in the initial stage and catches-up later on. COFIBA seems to surpass all other algorithms, especially in the cold-start regime, all other algorithms being in the same ballpark as CLUB. Finally, the Avazu data is furnished from its professional digital advertising solution platform, where the customers click the ad impressions via the iOS/Android mobile apps or through websites, serving either the publisher or the advertiser which leads to a daily high volume internet traffic. In this dataset, neither LINUCB-ONE nor LINUCB-IND displayed a competitive cold-start performance. DYNUCB is underperforming throughout, while LINUCB-V demonstrates a relatively high CTR. CLUB is strong at the beginning, but then its CTR performance degrades. On the other hand, COFIBA seems to work extremely well during the cold-start, and comparatively best in all later stages.

In Figure 4.8 we give a typical distribution of cluster sizes produced by COFIBA after at the end of its run.[6] The emerging pattern is always the same:

[6] Without loss of generality, we take the first Yahoo dataset to provide statistics, for similar shapes of the bar plots can be established for the remaining ones.

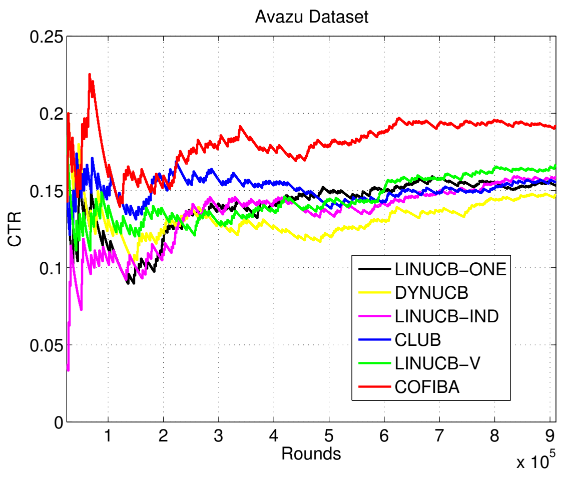


Figure 4.7: Results on the Avazu dataset.

we have few clusters over the items with very unbalanced sizes and, corresponding to each item cluster, we have few clusters over the users, again with very unbalanced sizes. This recurring pattern is in fact the motivation behind our theoretical assumptions (Section 4.2), and a property of data that the COFIBA algorithm can provably take advantage of (Section 4.6). These bar plots, combined with the comparatively good performance of COFIBA, suggest that our datasets do actually possess clusterability properties at both sides.

To summarize, despite the differences in the three datasets, the experimental evidence we collected on them is quite consistent, in that in all the three cases COFIBA significantly outperforms all other competing methods we tested. This is especially noticeable during the cold-start period, but the same relative behavior essentially shows up during the whole time window of our experiments. COFIBA is a bit involved to implement, as contrasted to its competitors, and is also somewhat slower to run (unsurprisingly slower than, say, LINUCB-ONE and LINUCB-IND). On the other hand, COFIBA is far more effective in exploiting the collaborative effects embedded in the data, and still amenable to be run on large datasets.

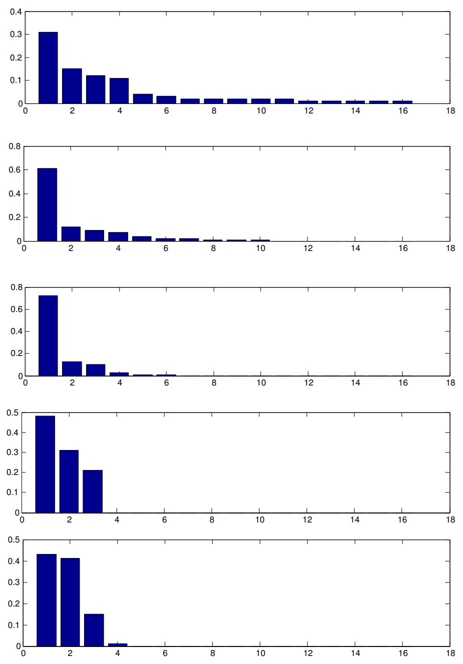


Figure 4.8: A typical distribution of cluster sizes over users for the Yahoo dataset. Each bar plot corresponds to a cluster at the item side. We have 5 plots since this is the number of clusters over the items that COFIBA ended up with after sweeping once over this dataset in the run at hand. Each bar represents the fraction of users contained in the corresponding cluster. For instance, the first cluster over the items generated 16 clusters over the users (bar plot on top), with relative sizes 31%, 15%, 12%, etc. The second cluster over the items generated 10 clusters over the users (second bar plot from top) with relative sizes 61%, 12%, 9%, etc. The relative size of the 5 clusters over the items is as follows: 83%, 10%, 4%, 2%, and 1%, so that the clustering pattern depicted in the top plot applies to 83% of the items, the second one to 10% of the items, and so on.

4.6 Regret Analysis

The following theorem is the theoretical guarantee of COFIBA, where we relate the cumulative regret of COFIBA to the clustering structure of users $\mathcal{U}$ w.r.t. items $\mathcal{I}$. For simplicity of presentation, we formulate our result in the one-hot encoding case, where $\boldsymbol{u}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$, and $\mathcal{I} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\}$. In fact, a more general statement can be proven which holds in the case when $\mathcal{I}$ is a generic set of feature vectors $\mathcal{I} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\mathcal{I}|}\}$, and the regret bound depends on the geometric properties of such vectors.[7]

In order to obtain a provable advantage from our clusterability assumptions, extra conditions are needed on the way i_t and C_{i_t} are generated. The clusterability assumptions we can naturally take advantage of are those where, for most partitions $P(\boldsymbol{e}_h)$, the relative sizes of clusters over users are highly unbalanced. Translated into more practical terms, cluster unbalancedness amounts to saying that the universe of items $\mathcal{I}$ tends to influence users so as to determine a small number of major common behaviors (which need neither be the same nor involve the same users across items), along with a number of minor ones. As we saw in our experiments, this seems like a frequent behavior of users in some practical scenarios.

Theorem 19. *Let the* COFIBA *algorithm of Figure 4.1 be run on a set of users* $\mathcal{U} = \{1, \ldots, n\}$ *with associated profile vectors* $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathbb{R}^d$*, and set of items* $\mathcal{I} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\}$ *such that the h-th induced partition* $P(\boldsymbol{e}_h)$ *over* $\mathcal{U}$ *is made up of* m_h *clusters of cardinality* $v_{h,1}, v_{h,2}, \ldots, v_{h,m_h}$*, respectively. Moreover, let* g *be the number of* distinct *partitions so obtained. At each round* t*, let* i_t *be generated uniformly at random*[8] *from* $\mathcal{U}$*. Once* i_t *is selected, the number* c_t *of items in* C_{i_t} *is generated arbitrarily as a function of past indices* $i_1, \ldots, i_{t-1}$*, payoffs* $a_1, \ldots, a_{t-1}$*, and sets* $C_{i_1}, \ldots, C_{i_{t-1}}$*, as well as the current index* i_t*. Then the sequence of items in* C_{i_t} *is generated i.i.d. (conditioned on* i_t*,* c_t *and all past indices* $i_1, \ldots, i_{t-1}$*, payoffs* $a_1, \ldots, a_{t-1}$*, and sets* $C_{i_1}, \ldots, C_{i_{t-1}}$*) according to a given but unknown distribution* $\mathcal{D}$ *over* $\mathcal{I}$*. Let payoff* a_t *lie in the interval* $[-1, 1]$*, and be generated as described in Section 4.2 so that, conditioned on history, the expectation of* a_t *is* $\boldsymbol{u}_{i_t}^\top \bar{\boldsymbol{x}}_t$*. Finally, let parameters* α *and* α_2 *be suitable functions of* $\log(1/\delta)$*. If* $c_t \le c\ \forall t$ *then, as* T *grows large, with probability at least* $1 - \delta$ *the cumulative regret satisfies*[9]

$$\sum_{t=1}^{T} r_t = \widetilde{\mathcal{O}}\left(\left(\mathbb{E}[S] + \sqrt{c\sqrt{mn}\,\textsc{var}(S)} + 1\right)\sqrt{\frac{dT}{n}}\right),$$

where $S = S(h) = \sum_{j=1}^{m_h} \sqrt{v_{h,j}}$*,* h *is a random index such that* $\boldsymbol{e}_h \sim \mathcal{D}$*, and* $\mathbb{E}[\cdot]$ *and* $\textsc{var}(\cdot)$ *denote, respectively, the expectation and the variance w.r.t. this random index.*

[7] In addition, the function CB should be modified so as to incorporate these properties.

[8] Any distribution having positive probability on each $i \in \mathcal{U}$ would suffice here.

[9] The $\widetilde{\mathcal{O}}$-notation hides logarithmic factors in n, m, g, T, d, $1/\delta$, as well as terms which are independent of T.

To get a feeling of how big (or small) $\mathbb{E}[S]$ and $\text{VAR}[S]$ can be, let us consider the case where each partition over users has a single big cluster and a number of small ones. To make it clear, consider the extreme scenario where each $P(\boldsymbol{e}_h)$ has one cluster of size $v_{h,1} = n - (m-1)$, and $m - 1$ clusters of size $v_{h,j} = 1$, with $m < \sqrt{n}$. Then it is easy to see that $\mathbb{E}[S] = \sqrt{n-(m-1)} + m - 1$, and $\text{VAR}(S) = 0$, so that the resulting regret bound essentially becomes $\widetilde{O}(\sqrt{dT})$, which is the standard regret bound one achieves for learning a *single* d-dimensional user (aka, the standard noncontextual bandit bound with d actions and no gap assumptions among them). At the other extreme lies the case when each partition $P(\boldsymbol{e}_h)$ has n-many clusters, so that $\mathbb{E}[S] = n$, $\text{VAR}(S) = 0$, and the resulting bound is $\widetilde{O}(\sqrt{dnT})$. Looser upper bounds can be achieved in the case when $\text{VAR}(S) > 0$, where also the interplay with c starts becoming relevant. Finally, observe that the number g of distinct partitions influences the bound only indirectly through $\text{VAR}(S)$. Yet, it is worth repeating here that g plays a crucial role in the computational (both time and space) complexity of the whole procedure.

Proof of Theorem 19. The proof sketch builds on the analysis in [41]. Let the true underlying clusters over the users be $V_{h,1}, V_{h,2}, \ldots, V_{h,m_h}$, with $|V_{h,j}| = v_{h,j}$. In [41], the authors show that, because each user i has probability $1/n$ to be the one served in round t, we have, with high probability, $\boldsymbol{w}_{i,t} \to \boldsymbol{u}_i$ for all i, as t grows large. Moreover, because of the gap assumption involving parameter γ, all edges connecting users belonging to different clusters at the user side will eventually be deleted (again, with high probability), after each user i is served at least $O(\frac{1}{\gamma^2})$ times. By the way edges are disconnected at the item side, the above is essentially independent (up to log factors due to union bounds) of which graph at the user side we are referring to. In turn, this entails that the current user clusters encoded by the connected components of graph $G^U_{t,h}$ will eventually converge to the m_h true user clusters (again, independent of h, up to log factors), so that the aggregate weight vectors $\bar{\boldsymbol{w}}_{N_k,t-1}$ computed by the algorithm for trading off exploration vs. exploitation in round t will essentially converge to $\boldsymbol{u}_{i_t}$ at a rate of the form[10]

$$\mathbb{E}\left[\frac{1}{\sqrt{1+T_{h_t,j_t,t-1}/d}}\right], \tag{4.1}$$

where h_t is the index of the true cluster over *items* that $\bar{\boldsymbol{x}}_t$ belongs to, j_t is the index of the true cluster over *users* that i_t belongs to (according to the partition of $\mathcal{U}$ determined by h_t), $T_{h_t,j_t,t-1}$ is the number of rounds so far where we happened to "hit" cluster V_{h_t,j_t}, i.e.,

$$T_{h_t,j_t,t-1} = |\{s \le t-1 \,:\, i_s \in V_{h_t,j_t}\}| ,$$

[10] Because $\mathcal{I} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\}$, the minimal eigenvalue λ of the process correlation matrix $\mathbb{E}[X\,X^\top]$ in [41] is here $1/d$. Moreover, compared to [41], we do not strive to capture the geometry of the user vectors $\boldsymbol{u}_i$ in the regret bound, hence we do not have the extra $\sqrt{m}$ factor occurring in their bound.

and the expectation is w.r.t. both the (uniform) distribution of i_t, and distribution $\mathcal{D}$ generating the items in C_{i_t}, conditioned on all past events. Since, by the Azuma-Hoeffding inequality, $T_{h_t,j_t,t-1}$ concentrates as

$$T_{h_t,j_t,t-1} \approx \frac{t-1}{n}\, v_{h_t,j_t} \ ,$$

we have

$$(4.1) \approx \mathbb{E}_{\mathcal{D}}\left[\sum_{j=1}^{m_{h_t}} \frac{v_{h_t,j}}{n}\, \frac{1}{\sqrt{1+\frac{t-1}{dn}\, v_{h_t,j}}}\right] .$$

It is the latter expression that rules the cumulative regret of COFIBA in that, up to log factors:

$$\sum_{t=1}^{T} r_t \approx \sum_{t=1}^{T} \mathbb{E}_{\mathcal{D}}\left[\sum_{j=1}^{m_{h_t}} \frac{v_{h_t,j}}{n}\, \frac{1}{\sqrt{1+\frac{t-1}{dn}\, v_{h_t,j}}}\right] . \qquad (4.2)$$

Eq. (4.2) is essentially (up to log factors and omitted additive terms) the regret bound one would obtain by knowning beforehand the latent clustering structure over $\mathcal{U}$.

Because $h_t \in C_{i_t}$ is itself a function of the items in C_{i_t}, we can eliminate the dependence on h_t by the following simple stratification argument. First of all, notice that

$$\sum_{j=1}^{m_{h_t}} \frac{v_{h_t,j}}{n}\, \frac{1}{\sqrt{1+\frac{t-1}{dn}\, v_{h_t,j}}} \approx \sqrt{\frac{d}{nt}} \sum_{j=1}^{m_{h_t}} \sqrt{v_{h_t,j}}\ .$$

Then, we set for brevity $S(h) = \sum_{j=1}^{m_h} \sqrt{v_{h,j}}$, and let $h_{t,k}$ be the index of the true cluster over items that $\boldsymbol{x}_{t,k}$ belongs to (recall that $h_{t,k}$ is a random variable since so is $\boldsymbol{x}_{t,k}$). Since $S(h_{t,k}) \le \sqrt{mn}$, a standard argument shows that

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[S(h_t)\right] &\le \mathbb{E}_{\mathcal{D}}\left[\max_{k=1,\ldots,c_t} S(h_{t,k})\right]\\ &\le \mathbb{E}_{\mathcal{D}}[S(h_{t,1})] + \sqrt{c\sqrt{mn}\,\text{VAR}_{\mathcal{D}}(S(h_{t,1}))} + 1\ ,\end{aligned}$$

so that, after some overapproximations, we conclude that $\sum_{t=1}^{T} r_t$ is upper bounded with high probability by

$$\widetilde{\mathcal{O}}\left(\left(\mathbb{E}_{\mathcal{D}}[S(h)] + \sqrt{c\sqrt{mn}\,\text{VAR}_{\mathcal{D}}(S(h))} + 1\right)\sqrt{\frac{dT}{n}}\right)\ ,$$

the expectation and the variance being over the random index h such that $\boldsymbol{e}_h \sim \mathcal{D}$. □

4.7 Conclusions

We have initiated an investigation of collaborative filtering bandit algorithms operating in relevant scenarios where multiple users can be grouped by behavior similarity in different ways w.r.t. items and, in turn, the universe of items can possibly be grouped by the similarity of clusterings they induce over users. We carried out an extensive experimental comparison with very encouraging results, and have also given a regret analysis which operates in a simplified scenario. Our algorithm can in principle be modified so as to be combined with any standard clustering (or co-clustering) technique. However, one advantage of encoding clusters as connected components of graphs (at least at the user side) is that we are quite effective in tackling the so-called *cold start* problem, for the newly served users are more likely to be connected to the old ones, which makes COFIBA in a position to automatically propagate information from the old users to the new ones through the aggregate vectors $\bar{\boldsymbol{w}}_{N_k,t}$. In fact, so far we have not seen any other way of adaptively clustering users and items which is computationally affordable on sizeable datasets and, at the same time, amenable to a regret analysis that takes advantage of the clustering assumption.

All our experiments have been conducted in the setup of one-hot encoding, since the datasets at our disposal did not come with reliable/useful annotations on data. Yet, the algorithm we presented can clearly work when the items are accompanied by (numerical) features. One direction of our future research is to compensate for the lack of features in the data by first *inferring* features during an initial training phase through standard matrix factorization techniques, and subsequently applying our algorithm to a universe of items $\mathcal{I}$ described through such inferred features. Another line of experimental research would be to combine different bandit algorithms (possibly at different stages of the learning process) so as to roughly get the best of all of them in all stages. This would be somewhat similar to the meta-bandit construction described in [97]. Another one would be to combine with matrix factorization techniques as in, e.g., [60].

Chapter 5

Showcase in the Quantification Problem

5.1 Introduction

Quantification [39] is defined as the task of estimating the prevalence (i.e., relative frequency) of the classes of interest in an unlabeled set, given a training set of items labeled according to the same classes. Quantification finds its natural application in contexts characterized by *distribution drift*, i.e., contexts where the training data may not exhibit the same class prevalence pattern as the test data. This phenomenon may be due to different reasons, including the inherent non-stationary character of the context, or class bias that affects the selection of the training data.

A naïve way to tackle quantification is via the "classify and count" (CC) approach, i.e., to classify each unlabeled item independently and compute the fraction of the unlabeled items that have been attributed to each class. However, a good classifier does not necessarily lead to a good quantifier: assuming the binary case, even if the sum (FP + FN) of the false positives and false negatives is comparatively small, bad quantification accuracy might result if FP and FN are significantly different (since perfect quantification coincides with the case FP = FN). This has led researchers to study quantification as a task in its own right, rather than as a byproduct of classification.

The fact that quantification is not just classification in disguise can also be seen by the fact that evaluation measures different from those for classification (e.g., F_1, AUC) need to be employed. Quantification actually amounts to computing how well an estimated class distribution $\hat{p}$ fits an actual class distribution p (where for any class $c \in \mathcal{C}$, $p(c)$ and $\hat{p}(c)$ respectively denote its true and estimated prevalence); as such, the natural way to evaluate the quality of this fit is via a function from the class of *f-divergences* [28], and a natural choice from this class (if only for the fact that it is the best known f-divergence) is the *Kullback-Leibler Diver-*

gence (KLD), defined as

$$\mathrm{KLD}(p, \hat{p}) = \sum_{c \in \mathcal{C}} p(c) \log \frac{p(c)}{\hat{p}(c)} \tag{5.1}$$

Indeed, KLD is the most frequently used measure for evaluating quantification (see e.g., [9, 38, 39, 40]). Note that KLD is non-decomposable, i.e., the error we make by estimating p via $\hat{p}$ cannot be broken down into item-level errors. This is not just a feature of KLD, but an inherent feature of *any* measure for evaluating quantification. In fact, how the error made on a given unlabeled item impacts the overall quantification error depends on how the other items have been classified[1]; e.g., if FP $>$ FN for the other unlabeled items, then generating an additional false negative is actually *beneficial* to the overall quantification accuracy, be it measured via KLD or via any other function.

The fact that KLD is the measure of choice for quantification and that it is non-decomposable, has lead to the use of structured output learners, such as SVM^{perf} [51], that allow a direct optimization of non-decomposable functions; the approach of Esuli and Sebastiani [37, 38] is indeed based on optimizing KLD using SVM^{perf}. However, that minimizing KLD (or |FP $-$ FN|, or any "pure" quantification measure) should be the only objective for quantification regardless of the value of FP $+$ FN (or any other classification measure), is fairly paradoxical. Some authors [9, 78] have observed that this might lead to the generation of unreliable quantifiers (i.e., systems with good quantification accuracy but bad or very bad classification accuracy), and have, as a result, championed the idea of optimizing "multi-objective" measures that combine quantification accuracy with classification accuracy. Using a decision-tree-like approach, [78] minimizes |FP^2 $-$ FN^2|, which is the product of |FN$-$FP|, a measure of quantification error, and (FN$+$FP), a measure of classification error; [9] also optimizes (using SVM^{perf}) a measure that combines quantification and classification accuracy.

While SVM^{perf} does provide a recipe for optimizing general performance measures, it has serious limitations. SVM^{perf} is not designed to directly handle applications where large streaming data sets are the norm. SVM^{perf} also does not scale well to multi-class settings, and the time required by the method is exponential in the number of classes.

In this paper we develop stochastic methods for optimizing a large family of popular quantification performance measures. Our methods can effortlessly work with streaming data and scale to very large datasets, offering training times up to an order of magnitude faster than other approaches such as SVM^{perf}.

[1]For the sake of simplicity, we assume here that quantification is to be tackled in an *aggregative* way, i.e., the classification of individual items is a necessary intermediate step for the estimation of class prevalences. Note however that this is not necessary; non-aggregative approaches to quantification may be found in [43, 62].

5.2 Related Work

Quantification methods. The quantification methods that have been proposed over the years can be broadly classified into two classes, namely aggregative and non-aggregative methods. While *aggregative* approaches perform quantification by first classifying individual items as an intermediate step, *non-aggregative* approaches do not require this step, and estimate class prevalences holistically. Most methods, such as those of [9, 10, 38, 39, 78], fall in the former class, while the latter class has few representatives [43, 62].

Within the class of aggregative methods, a further distinction can be made between methods, such as those of [10, 39], that first use general-purpose learning algorithms and then post-process their prevalence estimates to account for their estimation biases, and methods (which we have already hinted at in Section 5.1) that instead use learning algorithms explicitly devised for quantification [9, 38, 78]. In this paper we focus the latter class of methods.

Applications of quantification. From an application perspective, quantification is especially useful in fields (such as social science, political science, market research, and epidemiology) which are inherently interested in aggregate data, and care little about individual cases. Aside from applications in these fields [48, 62], quantification has also been used in contexts as diverse as natural language processing [24], resource allocation [39], tweet sentiment analysis [40], and the veterinary sciences [43]. Quantification has independently been studied within statistics [48, 62], machine learning [8, 35, 89], and data mining [38, 39]. Unsurprisingly, given this varied literature, quantification also goes under different names, such as *counting* [68], *class probability re-estimation* [3], *class prior estimation* [24], and *learning of class balance* [35].

In some applications of quantification, the estimation of class prevalences is not an end in itself, but is rather used to improve the accuracy of other tasks such as classification. For instance, Balikas et al. [8] use quantification for model selection in supervised learning, by tuning hyperparameters that yield the best quantification accuracy on validation data; this allows hyperparameter tuning to be performed without incurring the costs inherent to k-fold cross-validation. Saerens et al. [89], followed by other authors [3, 105, 108], apply quantification to customize a trained classifier to the class prevalence exhibited in the test set, with the goal of improving classification accuracy on unlabeled data exhibiting a class distribution different from that of the training set. The work of Chan and Ng [24] may be seen as a direct application of this notion, as they use quantification to tune a word sense disambiguator to the estimated sense priors of the test set. Their work can also be seen as an instance of *transfer learning* (see e.g., [84]), since their goal is to adapt a word sense disambiguation algorithm to a domain different from the one the algorithm was trained upon.

Stochastic optimization. As discussed in Section 5.1, our goal in this paper is to perform quantification by directly optimizing, in an online stochastic setting, specific performance measures for the quantification problem. While recent advances have seen much progress in efficient methods for online learning and optimization in full information and bandit settings [23, 41, 47, 92], these works frequently assume that the optimization objective, or the notion of regret being considered is decomposable and can be written as a sum or expectation of losses or penalties on individual data points. However, performance measures for quantification have a multivariate and complex structure, and do not have this form.

There has been some recent progress [56, 80] towards developing stochastic optimization methods for such non-decomposable measures. However, these approaches do not satisfy the needs of our problem. The work of Kar et al. [56] addresses the problem of optimizing structured SVM^{perf}-style objectives in a streaming fashion, but requires the maintenance of large buffers and, as a result, offers poor convergence. The work of Narasimhan et al. [80] presents online stochastic methods for optimizing performance measures that are concave or pseudo-linear in the canonical confusion matrix of the predictor. However, their method requires the computation of gradients of the Fenchel dual of the performance measures, which is difficult for the quantification performance measures that we study, that have a nested structure. Our methods extend the work of [80] and provide convenient routines for optimizing the more complex performance measures used for evaluating quantification.

5.3 Problem Setting

For the sake of simplicity, in this paper we will restrict our analysis to binary classification problems and linear models. We will denote the space of feature vectors by $\mathcal{X} \subset \mathbb{R}^d$ and the label set by $\mathcal{Y} = \{-1, +1\}$. We shall assume that data points are generated according to some fixed but unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. We will denote the proportion of positives in the population by $p := \underset{(\mathbf{x},y)\sim\mathcal{D}}{\Pr}[y = +1]$. Our algorithms, at training time, will receive a set of T training points sampled from $\mathcal{D}$, which we will denote by $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$.

As mentioned above, we will present our algorithms and analyses for learning a linear model over $\mathcal{X}$. We will denote the model space by $\mathcal{W} \subseteq \mathbb{R}^d$ and let $R_{\mathcal{X}}$ and $R_{\mathcal{W}}$ denote the radii of domain $\mathcal{X}$ and model space $\mathcal{W}$, respectively. However, we note that our algorithms and analyses can be extended to learning non-linear models by use of kernels, as well as to multi-class quantification problems. However, we postpone a discussion of these extensions to an expanded version of this paper.

Our focus in this work shall be the optimization of quantification-specific performance measures in online stochastic settings. We will concentrate on performance measures that can be represented as functions of the confusion matrix of the classifier. In the binary setting, the confusion matrix can be completely described in terms of the true positive rate (TPR) and the true negative rate (TNR) of the classifier. However, initially we will develop algorithms that use *reward functions* as surrogates of the TPR and TNR values. This is done to ease algorithm design and analysis, since the TPR and TNR values are count-based and form non-concave and non-differentiable estimators. The surrogates we will use will be concave and almost-everywhere differentiable. More formally, we will use a reward function r that assigns a *reward* $r(\hat{y}, y)$ to a prediction $\hat{y} \in \mathbb{R}$ for a data point, when the true label for that data point is $y \in \mathcal{Y}$. Given a reward function r, a model $\mathbf{w} \in \mathcal{W}$, and a data point $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$, we will use

$$
\begin{aligned}
r^+(\mathbf{w}; \mathbf{x}, y) &= \frac{1}{p} \cdot r(\mathbf{w}^\top \mathbf{x}, y) \cdot \mathbf{1}(y = 1) \\
r^-(\mathbf{w}; \mathbf{x}, y) &= \frac{1}{1-p} \cdot r(\mathbf{w}^\top \mathbf{x}, y) \cdot \mathbf{1}(y = -1)
\end{aligned}
$$

to calculate rewards on positive and negative points. The average or expected value of these rewards will be treated as surrogates of TPR and TNR respectively. Note that since $\underset{(\mathbf{x},y)}{\mathbb{E}} [\![r^+(\mathbf{w};\, \mathbf{x}, y)]\!] = \underset{(\mathbf{x},y)}{\mathbb{E}} [\![r(\mathbf{w}^\top \mathbf{x}, y) | y = 1]\!]$, setting r to $r^{\text{0-1}}(\hat{y}, y) = \mathbb{I}\left[y \cdot \hat{y} > 0\right]$, i.e. the classification accuracy function, yields $\underset{(\mathbf{x},y)}{\mathbb{E}} [\![r^+(\mathbf{w};\, \mathbf{x}, y)]\!] = \text{TPR}(\mathbf{w})$. Here $\mathbb{I}\left[\cdot\right]$ denotes the indicator function. For the sake of convenience we will use $P(\mathbf{w}) = \underset{(\mathbf{x},y)}{\mathbb{E}} [\![r^+(\mathbf{w}; \mathbf{x}, y)]\!]$ and $N(\mathbf{w}) = \underset{(\mathbf{x},y)}{\mathbb{E}} [\![r^-(\mathbf{w}; \mathbf{x}, y)]\!]$ to denote population averages of the reward functions. We shall assume that our reward function r is concave, L_r-Lipschitz, and takes values in a bounded range $[-B_r, B_r]$.

Examples of Surrogate Reward Functions Some examples of reward functions that are surrogates for the classification accuracy indicator function $\mathbb{I}\left[y\hat{y} > 0\right]$ are the inverted hinge loss function

$$r_{\text{hinge}}(\hat{y}, y) = \max\left\{1, y \cdot \hat{y}\right\}$$

and the inverted logistic regression function

$$r_{\text{logit}}(\hat{y}, y) = 1 - \ln(1 + \exp(-y \cdot \hat{y}))$$

We will also experiment with non-surrogate (dubbed NS) versions of our algorithms which use TPR and TNR values directly. These will be discussed in Section 5.5.

5.3.1 Performance Measures

The task of quantification requires estimating the distribution of unlabeled items across a set $\mathcal{C}$ of available classes, with $|\mathcal{C}| = 2$ in the binary setting. In our work we will target quantification performance measures as well as “hybrid” classification-quantification performance measures. We discuss them in turn.

KLD: Kullback-Leibler Divergence In recent years this performance measure has become a standard in the quantification literature, in the evaluation of both binary and multiclass quantification [9, 38, 40]. We redefine KLD below for convenience.

$$\mathrm{KLD}(p, \hat{p}) = \sum_{c \in \mathcal{C}} p(c) \log \frac{p(c)}{\hat{p}(c)} \tag{5.2}$$

For distributions p, p' over $\mathcal{C}$, the values $\mathrm{KLD}(p, p')$ can range between 0 (perfect quantification) and $+\infty$.[2]. Note that since KLD is a distance function and all our algorithms will be driven by reward maximization, for uniformity we will, instead of trying to minimize KLD, try to maximize $-\mathrm{KLD}$; we will call this latter NegKLD.

NSS: Normalized Squared Score This measure of quantification accuracy was introduced in [9], and is defined as $\mathrm{NSS} = 1 - (\frac{\mathrm{FN}-\mathrm{FP}}{\max\{p,(1-p)\}|\mathcal{S}|})^2$. Ignoring normalization constants, this performance measure attempts to reduce $|\mathrm{FN} - \mathrm{FP}|$, a direct measure of quantification error.

We recall from Section 5.1 that several works have advocated the use of hybrid, "multi-objective" performance measures, that try to balance quantification and classification performance. These measures typically take a quantification performance measure such as KLD or NSS, and combine it with a classification performance measure. Typically, a classification performance measure that is sensitive to class imbalance [80] is chosen, such as Balanced Accuracy $\mathrm{BA} = \frac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$ [9], F-measure, or G-mean [80]. Two such hybrid performance measures that are discussed in literature are presented below.

CQB: Classification-Quantification Balancing The work of [78] introduced this performance measure in an attempt to compromise between classification and

[2]KLD is not a particularly well-behaved performance measure, since it is capable of taking unbounded values within the compact domain of the unit simplex. This poses a problem for optimization algorithms from the point of view of convergence, as well as numerical stability. To solve this problem, while computing KLD for two distributions p and $\hat{p}$, we can perform an *additive smoothing* of both $p(c)$ and $\hat{p}(c)$ by computing

$$p_s(c) = \frac{\epsilon + p(c)}{\epsilon|\mathcal{C}| + \sum_{c \in \mathcal{C}} p(c)}$$

where $p_s(c)$ denotes the smoothed version of $p(c)$. The denominator here is just a normalizing factor. The quantity $\epsilon = \frac{1}{2|\mathcal{S}|}$ is often used as a smoothing factor, and is the one we adopt here. The smoothed versions of $p(c)$ and $\hat{p}(c)$ are then used in place of the non-smoothed versions in Equation 5.1. We can show that, as a result, KLD is always bounded by $\mathrm{KLD}(p_s, \hat{p}_s) \leq \mathcal{O}\left(\log \frac{1}{\epsilon}\right)$ However, we note that the smoothed KLD still returns a value of 0 when p and $\hat{p}$ are identical distributions.

quantification accuracy. As discussed in Section 5.1, this performance measure is defined as

$$\mathrm{CQB} = |\mathrm{FP}^2 - \mathrm{FN}^2| = |\mathrm{FP} - \mathrm{FN}| \cdot (\mathrm{FP} + \mathrm{FN}),$$

i.e. a product of $|\mathrm{FN} - \mathrm{FP}|$, a measure of quantification error, and $(\mathrm{FN} + \mathrm{FP})$, a measure of classification error.

QMeasure The work of Barranquero et al. [9] introduced a generic scheme for constructing hybrid performance measures, using the so-called Q-measure defined as

$$Q_\beta = (1 + \beta^2) \cdot \frac{\mathcal{P}_{\text{class}} \cdot \mathcal{P}_{\text{quant}}}{\beta^2 \mathcal{P}_{\text{class}} + \mathcal{P}_{\text{quant}}}, \tag{5.3}$$

that is, a weighted combination of a measure of classification accuracy $\mathcal{P}_{\text{class}}$ and a measure of quantification accuracy $\mathcal{P}_{\text{quant}}$. For the sake of simplicity, in our experiments we will adopt $\mathrm{BA} = \frac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$ as our $\mathcal{P}_{\text{class}}$ and NSS as our $\mathcal{P}_{\text{quant}}$. However, we stress that our methods can be suitably adapted to work with other choices of $\mathcal{P}_{\text{class}}$ and $\mathcal{P}_{\text{quant}}$.

We also introduce three new hybrid performance measures in this paper as a way of testing our optimization algorithms. We define these below and refer the reader to Tables 5.1 and 5.2 for details.

BAKLD This hybrid performance measure takes a weighted average of BA and NegKLD; i.e. $\mathrm{BAKLD} = C \cdot \mathrm{BA} + (1 - C) \cdot (-\mathrm{KLD})$. This performance measure gives the user a strong handle on how much emphasis should be placed on quantification and how much on classification performance. We will use BAKLD in our experiments to show that our methods offer an attractive tradeoff between the two.

We now define two hybrid performance measures that are constructed by taking the *ratio* of a classification and a quantification performance measures. The aim of this exercise is to obtain performance measures that mimic the F-measure, which is also a pseudolinear performance measure [80]. The ability of our methods to directly optimize such complex performance measures will be indicative of their utility in terms of the freedom they allow the user to design objectives in a data- and task-specific manner.

CQReward and BKReward These hybrid performance measures are defined as $\mathrm{CQReward} = \frac{\mathrm{BA}}{2 - \mathrm{NSS}}$ and $\mathrm{BKReward} = \frac{\mathrm{BA}}{1 + \mathrm{KLD}}$. Notice that both performance measures are optimized when the numerator i.e. BA is large, and the denominator is small which translates to NSS being large for CQReward and KLD being small for BKReward. Clearly, both performance measures encourage good performance with respect to both classification and quantification and penalize a predictor which either neglects quantification to get better classification performance, or the other way round.

The past section has seen us introduce a wide variety of quantification and hybrid performance measures. Of these, the NegKLD, NSS, and Q-measure were already prevalent in quantification literature and we introduced BAKLD, CQReward and BKReward. As discussed before, the aim of exploring such a large variety of performance measures is to both demonstrate the utility of our methods with respect to the quantification problem, and present newer ways of designing hybrid performance measures that give the user more expressivity in tailoring the performance measure to the task at hand.

We also note that these performance measures have extremely diverse and complex structures. We can show that NegKLD, Q-measure, and BAKLD are nested concave functions, more specifically, concave functions of functions that are themselves concave in the confusion matrix of the predictor. On the other hand, CQReward and BKReward turn out to be pseudo-concave functions of the confusion matrix. Thus, we are working with two very different families of performance measures here, each of which has different properties and requires different optimization techniques.

In the following section, we introduce two novel methods to optimize these two families of performance measures.

5.4 Stochastic Optimization Methods for Quantification

The previous discussion in Sections 5.1 and 5.2 clarifies two aspects of efforts in the quantification literature. Firstly, specific performance measures have been developed and adopted for evaluating quantification performance including KLD, NSS, Q-measure etc. Secondly, algorithms that *directly* optimize these performance measures are desirable, as is evidenced by recent works [9, 37, 38, 78].

The works mentioned above make use of tools from optimization literature to learn linear (e.g. [38]) and non-linear (e.g. [78]) models to perform quantification. The state of the art efforts in this direction have adopted the structural SVM approach for optimizing these performance measures with great success [9, 38]. However, this approach comes with severe drawbacks.

The structural SVM [51], although a significant tool that allows optimization of arbitrary performance measures, suffers from two key drawbacks. Firstly, the structural SVM surrogate is not necessarily a tight surrogate for all performance measures, something that has been demonstrated in past literature [57, 80], which can lead to poor training. But more importantly, optimizing the structural SVM surrogate requires the use of expensive cutting plane methods which are known to scale poorly with the amount of training data, as well as are unable to handle streaming data.

Table 5.1: A list of nested concave performance measures and their canonical expressions in terms of the confusion matrix $\Psi(P, N)$ where P and N denote the TPR, TNR values and p and n denote the proportion of positives and negatives in the population. The 4th, 6th and 8th columns give the closed form updates used in steps 15-17 in Algorithm 4.

Name	Expression	$\Psi(x, y)$	$\boldsymbol{\gamma}(\mathbf{q})$	$\zeta_1(P, N)$	$\boldsymbol{\alpha}(\mathbf{r})$	$\zeta_2(P, N)$	$\boldsymbol{\beta}(\mathbf{r})$
NegKLD [9, 38]	$-\text{KLD}(p,\hat{p})$	$p{\cdot}x+n{\cdot}y$	(p,n)	$\log(pP+n(1-N))$	$\left(\frac{1}{r_1},\frac{1}{r_2}\right)$	$\log(nN+p(1-P))$	$\left(\frac{1}{r_1},\frac{1}{r_2}\right)$
QMeasure$_\beta$[9]	$\frac{(1+\beta^2)\cdot\text{BA}\cdot\text{NSS}}{\beta^2\cdot\text{BA}+\text{NSS}}$	$\frac{(1+\beta^2)\cdot x\cdot y}{\beta^2\cdot x+y}$	$(1+\beta^2)\cdot\left(z^2,\frac{(1-z)^2}{\beta^2}\right)$ $z=\frac{q_2}{\beta^2 q_1+q_2}$	$\frac{P+N}{2}$	$(\frac{1}{2},\frac{1}{2})$	$1-(p(1-P)-n(1-N))^2$	$2(z,-z)$ $z=r_2-r_1$
BAKLD	$C{\cdot}\text{BA}+(1-C)\cdot(-\text{KLD})$	$C{\cdot}x+(1-C){\cdot}y$	$(C,1-C)$	$\frac{P+N}{2}$	$(\frac{1}{2},\frac{1}{2})$	$-\text{KLD}(P,N)$	see above

To alleviate these problems, we propose stochastic optimization algorithms that *directly* optimize a large family of quantification performance measures. Our methods come with sound theoretical convergence guarantees, are able to operate with streaming data sets and, as our experiments will demonstrate, offer much faster and accurate quantification performance on a variety of data sets.

Our optimization techniques introduce crucial advancements in the field of stochastic optimization of *multivariate performance measures* and address the two families of performance measures discussed while concluding Section 5.3 – 1) nested concave performance measures and 2) pseudo-concave performance measures. We describe these in turn below.

5.4.1 Nested Concave Performance Measures

The first class of performance measures that we deal with are concave combinations of concave performance measures. More formally, given three concave functions $\Psi, \zeta_1, \zeta_2 : \mathbb{R}^2 \to \mathbb{R}$, we can define a performance measure

$$\mathcal{P}_{(\Psi,\zeta_1,\zeta_2)}(\mathbf{w}) = \Psi(\zeta_1(\mathbf{w}), \zeta_2(\mathbf{w})),$$

where we have

$$\zeta_1(\mathbf{w}) = \zeta_1(P(\mathbf{w}), N(\mathbf{w}))$$
$$\zeta_2(\mathbf{w}) = \zeta_2(P(\mathbf{w}), N(\mathbf{w})),$$

where $P(\mathbf{w})$ and $N(\mathbf{w})$ can respectively denote, either the TPR and TNR values or surrogate reward functions therefor. Examples of such performance measures include the negative KLD performance measure and the QMeasure which are described in Section 5.3.1. Table 5.1 describes these performance measures in canonical form i.e. their expressions in terms of TPR and TNR values.

Before describing our algorithm for nested concave measures, we recall the notion of concave Fenchel conjugate of concave functions. For any concave function

$f : \mathbb{R}^2 \to \mathbb{R}$ and any $(u, v) \in \mathbb{R}^2$, the (concave) Fenchel conjugate of f is defined as

$$f^*(u, v) = \inf_{(x,y)\in\mathbb{R}^2} \{ux + vy - f(x, y)\}.$$

Clearly, f^* is concave. Moreover, it follows from the concavity of f that for any $(x, y) \in \mathbb{R}^2$,

$$f(x, y) = \inf_{(u,v)\in\mathbb{R}^2} \{xu + yv - f^*(u, v)\}.$$

Below we state the properties of strong concavity and smoothness. These will be crucial in our convergence analysis.

Definition 1 (Strong Concavity and Smoothness)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be α-strongly concave and γ-smooth if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have*

$$-\frac{\gamma}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \leq f(\mathbf{x}) - f(\mathbf{y}) - \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \leq -\frac{\alpha}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 .$$

We will assume that the functions Ψ, ζ_1, and ζ_2 defining our performance measures are γ-smooth for some constant $\gamma > 0$. This is true of all functions, save the $\log$ function which is used in the definition of the KLD quantification measure. However, if we carry out the smoothing step pointed out in Section 5.3.1 with some $\epsilon > 0$, then it can be shown that the KLD function does become $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$-smooth. An important property of smooth functions, that would be crucial in our analyses, is a close relationship between smooth and strongly convex functions

Theorem 2 ([107])**.** *A closed, concave function f is β smooth iff its (concave) Fenchel conjugate f^* is $\frac{1}{\beta}$-strongly concave.*

We are now in a position to present our algorithm **NEMSIS** for stochastic optimization of nested concave functions. Algorithm 4 gives an outline of the technique. We note that a direct application of traditional stochastic optimization techniques [93] to such nested performance measures as those considered here is not possible as discussed before. **NEMSIS**, overcomes these challenges by exploiting the nested dual structure of the performance measure by carefully balancing updates at the inner and outer levels.

At every time step, **NEMSIS** performs four very cheap updates. The first update is a *primal* ascent update to the model vector which takes a weighted stochastic gradient descent step. Note that this step involves a *projection* step to the set of model vectors $\mathcal{W}$ denoted by $\Pi_{\mathcal{W}}(\cdot)$. In our experiments $\mathcal{W}$ was defined to be the set of all Euclidean norm-bounded vectors so that projection could be effected using Euclidean normalization which can be done in $\mathcal{O}(d)$ time if the model vectors are d-dimensional.

The weights of the descent step are decided by the dual parameters of the functions Ψ, ζ_1, and ζ_2. Then **NEMSIS** updates the dual variables in three simple steps. In fact line numbers 15-17 can be executed in closed form (see Table 5.1) for all the performance measures we see here which allows for very rapid updates. See Appendix 5.7 for the simple derivations.

Algorithm 4 NEMSIS: NEsted priMal-dual StochastIc updateS

```
Require: Outer wrapper function $\Psi$, inner performance measures $\zeta_1, \zeta_2$, step sizes $\eta_t$,
    feasible sets $\mathcal{W}, \mathcal{A}$
Ensure: Classifier $\mathbf{w} \in \mathcal{W}$
 1: $\mathbf{w}_0 \leftarrow \mathbf{0}, t \leftarrow 0, \{\mathbf{r}_0, \mathbf{q}_0, \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0\} \leftarrow (0, 0)$
 2: while data stream has points do
 3:     Receive data point $(\mathbf{x}_t, y_t)$
 4:     //Perform primal ascent
 5:     if $y_t > 0$ then
 6:         $\mathbf{w}_{t+1} \leftarrow \Pi_{\mathcal{W}}\left(\mathbf{w}_t + \eta_t(\gamma_{t,1}\alpha_{t,1} + \gamma_{t,2}\beta_{t,1})\nabla_{\mathbf{w}} r^+(\mathbf{w}_t; \mathbf{x}_t, y_t)\right)$
 7:         $\mathbf{q}_{t+1} \leftarrow t \cdot \mathbf{q}_t + (\alpha_{t,1}, \beta_{t,1}) \cdot r^+(\mathbf{w}_t; \mathbf{x}_t, y_t)$
 8:     else
 9:         $\mathbf{w}_{t+1} \leftarrow \Pi_{\mathcal{W}}\left(\mathbf{w}_t + \eta_t(\gamma_{t,1}\alpha_{t,2} + \gamma_{t,2}\beta_{t,2})\nabla_{\mathbf{w}} r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)\right)$
10:         $\mathbf{q}_{t+1} \leftarrow t \cdot \mathbf{q}_t + (\alpha_{t,2}, \beta_{t,2}) \cdot r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)$
11:     end if
12:     $\mathbf{r}_{t+1} \leftarrow (t+1)^{-1}\left(t \cdot \mathbf{r}_t + (r^+(\mathbf{w}_t; \mathbf{x}_t, y_t), r^-(\mathbf{w}_t; \mathbf{x}_t, y_t))\right)$
13:     $\mathbf{q}_{t+1} \leftarrow (t+1)^{-1}\left(\mathbf{q}_{t+1} - (\zeta_1^*(\boldsymbol{\alpha}_t), \zeta_2^*(\boldsymbol{\beta}_t))\right)$
14:     //Perform dual updates
15:     $\boldsymbol{\alpha}_{t+1} = \arg\min_{\boldsymbol{\alpha}} \{\boldsymbol{\alpha} \cdot \mathbf{r}_{t+1} - \zeta_1^*(\boldsymbol{\alpha})\}$
16:     $\boldsymbol{\beta}_{t+1} = \arg\min_{\boldsymbol{\beta}} \{\boldsymbol{\beta} \cdot \mathbf{r}_{t+1} - \zeta_2^*(\boldsymbol{\beta})\}$
17:     $\boldsymbol{\gamma}_{t+1} = \arg\min_{\boldsymbol{\gamma}} \{\boldsymbol{\gamma} \cdot \mathbf{q}_{t+1} - \Psi^*(\boldsymbol{\gamma})\}$
18:     $t \leftarrow t + 1$
19: end while
20: return $\overline{\mathbf{w}} = \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{w}_\tau$
```

Below we state the convergence proof for **NEMSIS**. We note that despite the complicated nature of the performance measures being tackled, **NEMSIS** is still able to recover the optimal rate of convergence known for stochastic optimization routines. We refer the reader to Appendix 5.8 for a proof of this theorem. The proof requires a careful analysis of the primal and dual update steps at different levels and tying the updates together by taking into account the nesting structure of the performance measure.

Theorem 3. *Suppose we are given a stream of random samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ drawn from a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Let Algorithm 4 be executed with step sizes $\eta_t = \Theta(1/\sqrt{t})$ with a nested concave performance measure $\Psi(\zeta_1(\cdot), \zeta_2(\cdot))$. Then, for some universal constant C, the average model $\overline{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$ output by the algorithm satisfies, with probability at least $1 - \delta$,*

$$\mathcal{P}_{(\Psi,\zeta_1,\zeta_2)}(\overline{\mathbf{w}}) \geq \sup_{\mathbf{w}^* \in \mathcal{W}} \mathcal{P}_{(\Psi,\zeta_1,\zeta_2)}(\mathbf{w}^*) - C_{\Psi,\zeta_1,\zeta_2,r} \cdot \left(\frac{\log \frac{1}{\delta}}{\sqrt{T}}\right),$$

where $C_{\Psi,\zeta_1,\zeta_2,r} = C(L_\Psi(L_r + B_r)(L_{\zeta_1} + L_{\zeta_2}))$ for a universal constant C and L_g denotes the Lipschitz constant of the function g.

Table 5.2: List of pseudo-concave performance measures and their canonical expressions in terms of the confusion matrix $\Psi(P, N)$. Note that p and n denote the proportion of positives and negatives in the population.

Name	Expression	$\mathcal{P}_{\text{quant}}(P, N)$	$\mathcal{P}_{\text{class}}(P, N)$
CQReward	$\frac{\text{BA}}{2-\text{NSS}}$	$1+(p(1-P)-n(1-N))^2$	$\frac{P+N}{2}$
BKReward	$\frac{\text{BA}}{1+\text{KLD}}$	KLD: see Table 5.1	$\frac{P+N}{2}$

Related work of Narasimhan et al : Narasimhan *et al* [80] proposed an algorithm **SPADE** which offered stochastic optimization of concave performance measures. We note that although the performance measures considered here are indeed concave, it is difficult to apply **SPADE** to them directly since **SPADE** requires computation of gradients of the Fenchel dual of the function $\mathcal{P}_{(\Psi,\zeta_1,\zeta_2)}$ which are difficult to compute given the nested structure of this function. **NEMSIS**, on the other hand, only requires the duals of the individual functions Ψ, ζ_1, and ζ_2 which are much more accessible. Moreover, **NEMSIS** uses a much simpler dual update which does not involve any parameters and, in fact, has a closed form solution in all our cases. **SPADE**, on the other hand, performs dual gradient descent which requires a fine tuning of yet another step length parameter. A third benefit of **NEMSIS** is that it achieves a logarithmic regret with respect to its dual updates (see the proof of Theorem 3) whereas **SPADE** incurs a polynomial regret due to its gradient descent-style dual update.

5.4.2 Pseudo-concave Performance Measures

The next class of performance measures we consider can be expressed as a ratio of a quantification and a classification performance measure. More formally, given a *convex* quantification performance measure $\mathcal{P}_{\text{quant}}$ and a *concave* classification performance measure $\mathcal{P}_{\text{class}}$, we can define a performance measure

$$\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}(\mathbf{w}) = \frac{\mathcal{P}_{\text{class}}(\mathbf{w})}{\mathcal{P}_{\text{quant}}(\mathbf{w})},$$

We assume that both the performance measures, $\mathcal{P}_{\text{quant}}$ and $\mathcal{P}_{\text{class}}$, are positive valued. Such performance measures can be very useful in allowing a system designer to balance classification and quantification performance. Moreover, the form of the measure allows an enormous amount of freedom in choosing the quantification and classification performance measures. Examples of such performance measures include the CQReward and the BKReward measures. These were introduced in Section 5.3.1 and are represented in their canonical forms in Table 5.2.

Performance measures, constructed the way described above, with a ratio of a concave over a convex measures, are called *pseudo-concave* measures. This is because, although these functions are not concave, their level sets are still convex which makes it possible to optimize them efficiently. To see the intuition behind this, we need to introduce the notion of the *valuation function* corresponding to

Algorithm 5 CAN: Concave AlternatioN

```
Require: Objective $\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}$, model space $\mathcal{W}$, tolerance $\epsilon$
Ensure: An $\epsilon$-optimal classifier $\mathbf{w} \in \mathcal{W}$
1: Construct the valuation function $V$
2: $\mathbf{w}_0 \leftarrow \mathbf{0}, t \leftarrow 1$
3: while $v_t > v_{t-1} + \epsilon$ do
4:     $\mathbf{w}_{t+1} \leftarrow \arg\max_{\mathbf{w}\in\mathcal{W}} V(\mathbf{w}, v_t)$
5:     $v_{t+1} \leftarrow \arg\max_{v>0} v$ such that $V(\mathbf{w}_{t+1}, v) \geq v$
6:     $t \leftarrow t + 1$
7: end while
8: return $\mathbf{w}_t$
```

the performance measure. As a passing note, we remark that because of the non-concavity of these performance measures, **NEMSIS** cannot be applied here.

Definition 4 (Valuation Function). *The valuation of a pseudo-concave performance measure $\mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w}) = \frac{\mathcal{P}_{class}(\mathbf{w})}{\mathcal{P}_{quant}(\mathbf{w})}$ at any level $v > 0$, is defined as*

$$V(\mathbf{w}, v) = \mathcal{P}_{class}(\mathbf{w}) - v \cdot \mathcal{P}_{quant}(\mathbf{w})$$

It can be seen that the valuation function defines the level sets of the performance measure. To see this, notice that due to the positivity of the functions $\mathcal{P}_{\text{quant}}$ and $\mathcal{P}_{\text{class}}$, we can have $\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}(\mathbf{w}) \geq v$ iff $V(\mathbf{w}, v) \geq 0$. However, since $\mathcal{P}_{\text{class}}$ is concave, $\mathcal{P}_{\text{quant}}$ is convex, and $v > 0$, $V(\mathbf{w}, v)$ is a concave function of $\mathbf{w}$.

This close connection between the level sets and notions of valuation functions have been exploited before to give optimization algorithms for pseudo-linear performance measures such as the F-measure [80, 85]. These approaches treat the valuation function as some form of proxy or surrogate for the original performance measure and optimize it in hopes of making progress with respect to the original measure.

Taking this approach with our performance measures yields a very natural algorithm for optimizing pseudo-concave measures which we outline in the **CAN** algorithm Algorithm 5. **CAN** repeatedly trains models to optimize their valuations at the current level, then upgrades the level itself. Notice that step 4 in the algorithm is a concave maximization problem over a convex set, something that can be done using a variety of methods – in the following we will see how **NEMSIS** can be used to implement this step. Also notice that step 5 can, by the definition of the valuation function, be carried out by simply setting $v_{t+1} = \mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}(\mathbf{w}_{t+1})$.

It turns out that **CAN** has a linear rate of convergence for well-behaved performance measures. The next result formalizes this statement. We note that this result is similar to the one arrived by [80] but only for *pseudo-linear* functions.

Theorem 5. *Suppose we execute Algorithm 5 with a pseudo-concave performance measure $\mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}$ such that the quantification performance measure always takes values in the range $[m, M]$, where $M > m > 0$. Let $\mathcal{P}^* :=$*

$\sup_{\mathbf{w}\in\mathcal{W}} \mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w})$ *be the optimal performance level and* $\Delta_t = \mathcal{P}^* - \mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w}_t)$ *be the excess error for the model* $\mathbf{w}_t$ *generated at time* t*. Then, for every* $t > 0$*, we have* $\Delta_t \leq \Delta_0 \cdot \left(1 - \frac{m}{M}\right)^t$*.*

We refer the reader to Appendix 5.9 for a proof of this theorem. This theorem generalizes the result of [80] to the more general case of pseudo-concave functions. Note that for the pseudo-concave functions defined in Table 5.2, care is taken to ensure that the quantification performance measure satisfies $m > 0$.

A drawback of **CAN** is that it cannot operate in streaming data settings and requires a concave optimization oracle. However, we notice that for the performance measures in Table 5.2, the valuation function is always at least a nested concave function. This motivates us to use **NEMSIS** to solve the inner optimization problems in an online fashion. Combining this with an online technique to approximately execute step 5 of of the **CAN** and gives us the **SCAN** algorithm, outlined in Algorithm 6.

Thoerem 6 shows that **SCAN** enjoys a convergence rate similar to that of **NEMSIS**. Indeed, **SCAN** is able to guarantee an ϵ-approximate solution after witnessing $\widetilde{\mathcal{O}}\left(1/\epsilon^2\right)$ samples which is equivalent to a convergence rate of $\widetilde{\mathcal{O}}\left(1/\sqrt{T}\right)$. The proof of this result is obtained by showing that **CAN** is robust to approximate solutions to the inner optimization problems. We refer the reader to Appendix 5.10 for a proof of this theorem.

Theorem 6. *Suppose we execute Algorithm 6 with a pseudo-concave performance measure* $\mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}$ *such that* $\mathcal{P}_{quant}$ *always takes values in the range* $[m, M]$ *with* $m > 0$*, with epoch lengths* $s_e, s'_e = \frac{C_{\Psi,\zeta_1,\zeta_2,r}}{m^2}\left(\frac{M}{M-m}\right)^{2e}$ *following a geometric rate of increase, where the constant* $C_{\Psi,\zeta_1,\zeta_2,r}$ *is the effective constant for the* ***NEMSIS*** *analysis (Theorem 3) for the inner invocations of* ***NEMSIS*** *in* ***SCAN****. Also let the excess error for the model* $\mathbf{w}_e$ *generated after* e *epochs be denoted by* $\Delta_e = \mathcal{P}^* - \mathcal{P}_{(\mathcal{P}_{quant},\mathcal{P}_{class})}(\mathbf{w}_e)$*. Then after* $e = \mathcal{O}\left(\log\left(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\right)\right)$ *epochs, we can ensure with probability at least* $1-\delta$ *that* $\Delta_e \leq \epsilon$*. Moreover, the number of samples consumed till this point, ignoring universal constants, is at most* $\frac{C^2_{\Psi,\zeta_1,\zeta_2,r}}{\epsilon^2}\left(\log\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\right)\log^4\frac{1}{\epsilon}$*.*

Related work of Narasimhan et al : Narasimhan *et al* [80] also proposed two algorithms **AMP** and **STAMP** which seek to optimize *pseudo-linear* performance measures. However, neither those algorithms nor their analyses transfer directly to the pseudo-concave setting. This is because, by exploiting the pseudo-linearity of the performance measure, **AMP** and **STAMP** are able to convert their problem to a sequence of cost-weighted optimization problems which are very simple to solve. This convenience is absent here and as mentioned above, even after creation of the valuation function, **SCAN** still has to solve a possibly nested concave minimization problem which it does by invoking the **NEMSIS** procedure on this inner problem. The proof technique used in [80] for analyzing **AMP** also makes heavy

Algorithm 6 SCAN: Stochastic Concave AlternatioN

```
Require: Objective $\mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}$, model space $\mathcal{W}$, step sizes $\eta_t$, epoch lengths $s_e, s'_e$
Ensure: Classifier $\mathbf{w} \in \mathcal{W}$
 1: $v_0 \leftarrow 0, t \leftarrow 0, e \leftarrow 0, \mathbf{w}_0 \leftarrow \mathbf{0}$
 2: repeat
 3:     //Learning phase
 4:     $\widetilde{\mathbf{w}} \leftarrow \mathbf{w}_e$
 5:     while $t < s_e$ do
 6:         Receive sample $(\mathbf{x}, y)$
 7:         //NEMSIS update with $V(\cdot, v_e)$ at time $t$
 8:         $\mathbf{w}_{t+1} \leftarrow$ NEMSIS $(V(\cdot, v_e), \mathbf{w}_t, (\mathbf{x}, y), t)$
 9:         $t \leftarrow t + 1$
10:     end while
11:     $t \leftarrow 0, e \leftarrow e + 1, \mathbf{w}_{e+1} \leftarrow \widetilde{\mathbf{w}}$
12:     //Level estimation phase
13:     $v_+ \leftarrow 0, v_- \leftarrow 0$
14:     while $t < s'_e$ do
15:         Receive sample $(\mathbf{x}, y)$
16:         $v_y \leftarrow v_y + r^y(\mathbf{w}_e; \mathbf{x}, y)$        //Collect rewards
17:         $t \leftarrow t + 1$
18:     end while
19:     $t \leftarrow 0, v_e \leftarrow \frac{\mathcal{P}_{\text{class}}(v_+, v_-)}{\mathcal{P}_{\text{quant}}(v_+, v_-)}$
20: until stream is exhausted
21: return $\mathbf{w}_e$
```

use of pseudo-linearity. The convergence proof of **CAN**, on the other hand, is more general and yet guarantees a linear convergence rate.

5.5 Experimental Results

We carried out an extensive set of experiments on diverse set of benchmark and real-world data to compare our proposed methods with other state-of-the-art approaches.

Data sets: We used the following benchmark data sets from the UCI machine learning repository : a) IJCNN, b) Covertype, c) Adult, d) Letters, and e) Cod-RNA. We also used the following three real-world data sets: a) Cheminformatics, a drug discovery data set from [53], b) 2008 KDD Cup challenge data set on breast cancer detection, and c) a data set pertaining to a protein-protein interaction (PPI) prediction task [86]. In each case, we used 70% of the data for training and the remaining for testing.

Methods: We compares our proposed **NEMSIS** and **SCAN** algorithms[3] against the state-of-the-art one-pass mini-batch stochastic gradient method

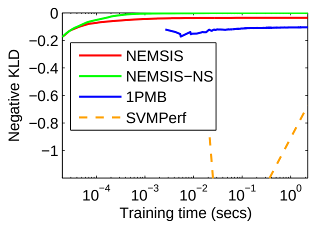
[3]We will make code for our methods available publicly.

Table 5.3: Statistics of data sets used.

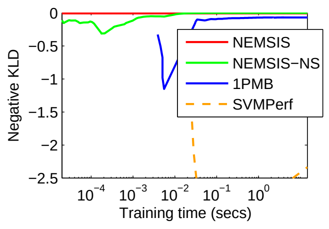
Data Set	**Data Points**	**Features**	**Positives**
KDDCup08	102,294	117	0.61%
PPI	240,249	85	1.19%
CoverType	581,012	54	1.63%
ChemInfo	2142	55	2.33%
Letter	20,000	16	3.92%
IJCNN-1	141,691	22	9.57%
Adult	48,842	123	23.93%
Cod-RNA	488,565	8	33.3%

(**1PMB**) of [56] and the SVM^{perf} technique of [52]. Both these techniques are capable of optimizing structural SVM surrogates of arbitrary performance measures and we modified their implementations to suitably adapt them to the performance measures considered here. The **NEMSIS** and **SCAN** implementations used the hinge-based concave surrogate.

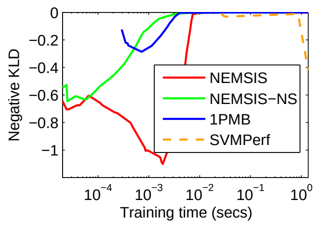
Non-surrogate NS Approaches: We also experimented with a variant of the **NEMSIS** and **SCAN** algorithms, where the dual updates were computed using original count based TPR and TNR values, rather than surrogate reward functions. We refer to this version as **NEMSIS-NS**. We also developed a similar version of **SCAN** called **SCAN-NS** where the level estimation was performed using 0-1 TPR/TNR values. We empirically observed these non-surrogate versions of the algorithms to offer superior and more stable performance than the surrogate versions.



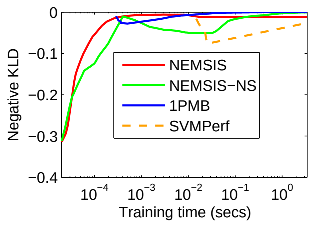
(a) Adult



(b) Cod-RNA

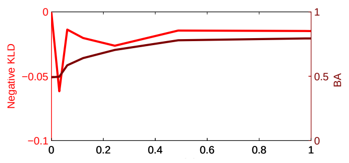


(c) KDD08

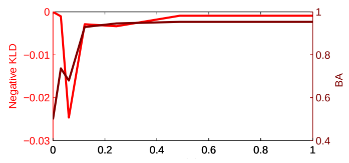


(d) PPI

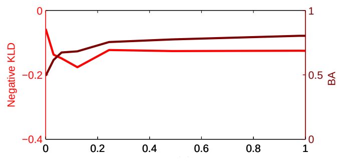
Figure 5.1: Experiments with **NEMSIS** on NegKLD: Plot of NegKLD as a function of training time.



(a) Adult



(b) Cod-RNA



(c) Covtype

Figure 5.2: Experiments on **NEMSIS** with BAKLD: Plots of quantification and classification performance as CWeight is varied.

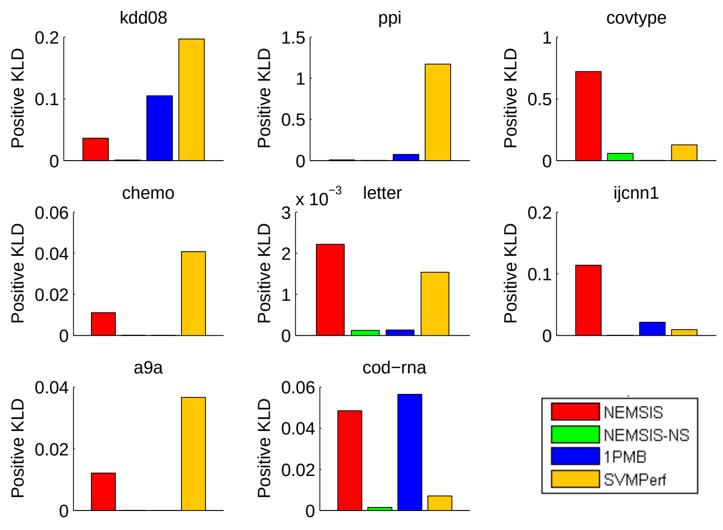
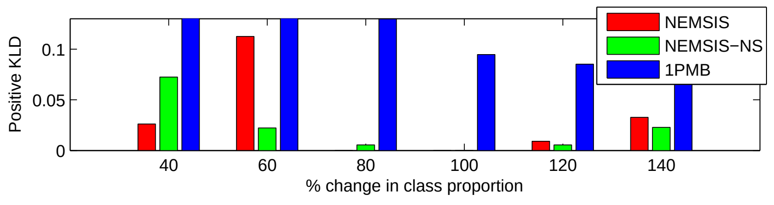
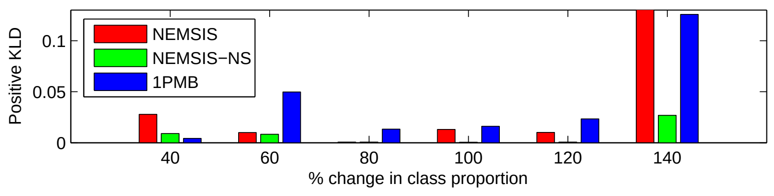


Figure 5.3: A comparison of the KLD performance of various methods on data sets with varying class proportions (see Table 5.4.2).

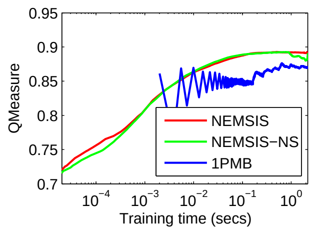


(a) Adult

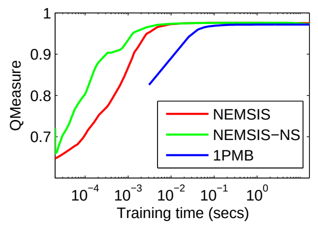


(b) Letter

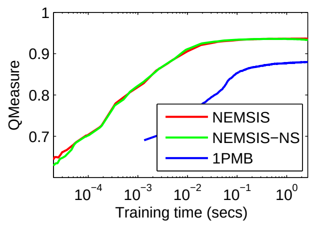
Figure 5.4: A comparison of the KLD performance of various methods when distribution drift is introduced in the test sets.



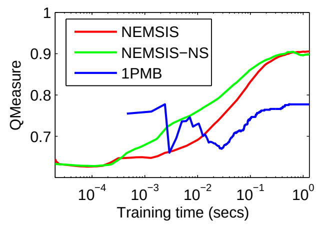
(a) Adult



(b) Cod-RNA

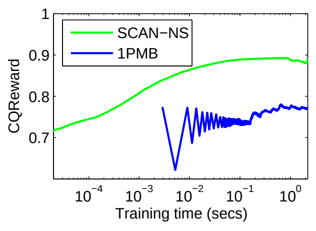


(c) IJCNN1

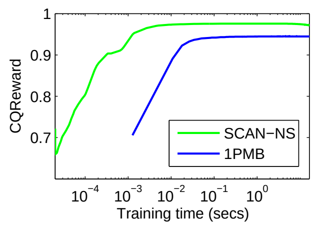


(d) KDD08

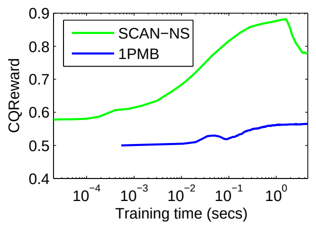
Figure 5.5: Experiments with **NEMSIS** on Q-measure: Plot of Q-measure performance as a function of time.



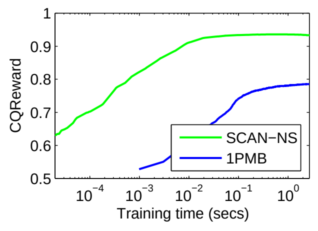
(a) Adult



(b) Cod-RNA



(c) CovType



(d) IJCNN1

Figure 5.6: Experiments with **SCAN** on CQreward: Plot of CQreward performance as a function of time.

Parameters: All parameters including step sizes, upper bounds on reward functions, regularization parameters, and projection radii were tuned from the values $\{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$ using a held-out portion of the training set treated as a validation set. For step sizes, the base step length η_0 was tuned from the above set and the step lengths were set to $\eta_t = \eta_0/\sqrt{t}$. In **1PMB**, we mimic the parameter setting in [56], setting the buffer size to 500 and the number of passes to 25.

Comparison on NegKLD: We first compare **NEMSIS-NS** and **NEMSIS** against the baselines **1PMB** and SVM^{perf} on several data sets on the negative KLD measure. The results are presented in Figure 5.1. It is clear that the proposed algorithms have comparable performance with significantly faster rate of convergence. Since SVM^{perf} is a batch/off-line method, it is important to clarify how it was compared against the other online methods. In this case, timers were embedded inside the SVM^{perf} code, and at regular intervals, the performance of the current model vector was evaluated. It is clear that SVM^{perf} is significantly slower and its behavior is quite erratic. The proposed methods are often faster than **1PMB**. On three of the four data sets **NEMSIS-NS** achieves a faster rate of convergence compared to **NEMSIS**.

Comparison on BAKLD: We also used the BAKLD performance measure to evaluate the trade-off **NEMSIS** offers between quantification and classification performance. The weighting parameter C in BAKLD (see Table 5.1), denoted here by CWeight to avoid confusion, was varied from 0 to 1 across a fine grid; for each value, **NEMSIS** was used to optimize BAKLD and its performance on BA and KLD were noted separately. In the results presented in Figure 5.2 for three data sets, notice that there is a sweet spot where the two tasks, i.e. quantification and classification simultaneously have good performance.

Comparison under varying class proportions: We next evaluated the robustness of the algorithms across data sets with varying different class proportions (see Table 5.4.2 for the dataset label proportions). In Figure 5.3, we plot positive KLD (smaller values are better) for the proposed and baseline methods for these diverse datasets. Again, it is clear that the **NEMSIS** family of algorithms of has better KLD performance compared to the baselines, demonstrating their versatility across a range of class distributions.

Comparison under varying drift: Next, we test the performance of the **NEMSIS** family of methods when there are drifts in class proportions between the train and test sets. In each case, we retain the original class proportion in the train set, and vary the class proportions in the test set, by suitably sampling from the original set of positive and negative test instances.[4] We have not included SVM^{perf}

[4]More formally, we consider a setting where both the train and test sets are generated using the same conditional class distribution $\mathbf{P}(Y = 1\,|X)$, but with different marginal distributions over instances $\mathbf{P}(X)$, and thus, have different class proportions. Further, in these experiments, we made a simplistic assumption that there is no label noise; hence for any instance $\mathbf{x}$, $\mathbf{P}(Y = 1\,|\,X = \mathbf{x}) = 1$ or 0. Thus, we generated our test set with class proportion p' by simply setting $\mathbf{P}(X = \mathbf{x})$ to the following distribution: with probability p', sample a point uniformly from all points with $\mathbf{P}(Y = 1\,|\,X = \mathbf{x}) = 1$, and with probability $1 - p'$, sample a point uniformly from all points with

in these experiments as it took an inordinately long time to complete. As seen in Figure 5.4, on the Adult and Letter data set the **NEMSIS** family is fairly robust to small class drifts. As expected, when the class proportions change by a large amount in the test set (over 100 percent), all algorithms perform poorly.

Comparison on hybrid performance measures: Finally, we tested our methods in optimizing composite performance measures that strike a more nuanced trade-off between quantification and classification performance. Figures 5.5 contains results for the **NEMSIS** methods while optimizing Q-measure (see Table 5.1), and Figure 5.6 contains results for **SCAN-NS** while optimizing CQReward (see Table 5.2). The proposed methods are often significantly better than the baseline **1PMB** in terms of both accuracy and running time.

5.6 Conclusion

Quantification, the task of estimating class prevalence in problem settings subject to distribution drift, has emerged as an important problem in machine learning and data mining. Our discussion justified the necessity to design algorithms that exclusively solve the quantification task, with a special emphasis on performance measures such as the Kullback-Leibler divergence that is considered a de facto standard in the literature.

In this paper we proposed a family of algorithms **NEMSIS**, **CAN**, **SCAN**, and their non-surrogate versions, to address the online quantification problem. By abstracting NegKLD and other hybrid performance measures as nested concave or pseudo concave functions we designed provably correct and efficient algorithms for optimizing these performance measures in an online stochastic setting.

We validated our algorithms on several data sets under varying conditions, including class imbalance and distribution drift. The proposed algorithms demonstrate the ability to jointly optimize both quantification and classification tasks. To the best of our knowledge this is the first work which directly addresses the online quantification problem and as such, opens up novel application areas.

5.7 Deriving Updates for NEMSIS

The derivation of the closed form updates for steps 15-17 in the **NEMSIS** algorithm (see Algorithm 4) starts with the observation that in all the nested concave performance measures considered here, the outer and the inner concave functions, namely Ψ, ζ_1, ζ_2 are concave, continuous, and differentiable. The logarithm function is non-differentiable at 0 but the smoothing step (see Section refformulation) ensures that we will never approach 0 in our analyses or the execution of the algorithm. The derivations hinge on the following basic result from convex analysis [107]:

$\mathbf{P}(Y = 1 \,|\, X = \mathbf{x}) = 0$.

Lemma 7. *Let f be a closed, differentiable and concave function and f^* be its concave Fenchel dual. Then $\nabla f^* = (\nabla f)^{-1}$ i.e. for any $\mathbf{x} \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{X}^*$ (the space of all linear functionals over $\mathcal{X}$), we have $\nabla f^*(\mathbf{u}) = \mathbf{x}$ iff $\nabla f(\mathbf{x}) = \mathbf{u}$.*

Using this result, we show how to derive the updates for $\boldsymbol{\gamma}$. The updates for $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ follow similarly. We have

$$\boldsymbol{\gamma}_t = \arg\min_{\boldsymbol{\gamma}} \{\boldsymbol{\gamma} \cdot \mathbf{q}_t - \Psi^*(\boldsymbol{\gamma})\}$$

By first order optimality conditions, we get that γ_t can minimize the function $h(\gamma) = \gamma \cdot \mathbf{q}_t - \Psi^*(\gamma)$ only if $\mathbf{q}_t = \nabla\Psi^*(\gamma_t)$. Using Lemma 7, we get $\gamma_t = \nabla\Psi(\mathbf{q}_t)$. Using this technique, all the closed form expressions can be readily derived.

For the derivations of α, β for NegKLD, and the derivation of β for Q-measure, the derivations follow when we work with definitions of these performance measures with the TP and TN *counts* or cumulative surrogate reward values, rather than the TPR and TNR values and the average surrogate rewards.

5.8 Proof of Theorem 3

We begin by observing the following general lemma regarding the follow the leader algorithm for strongly convex losses. This will be useful since steps 15-17 of Algorithm 4 are essentially executing follow the leader steps to decide the best value for the dual variables.

Lemma 8. *Suppose we have an action space $\mathcal{X}$ and execute the follow the leader algorithm on a sequence of loss functions $\ell_t : \mathcal{X} \to \mathbb{R}$, each of which is α-strongly convex and L-Lipschitz, then we have*

$$\sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \inf_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} \ell_t(\mathbf{x}) \leq \frac{L^2 \log T}{\alpha},$$

where $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{\tau=1}^{t} \ell_\tau(\mathbf{x})$ are the FTL plays.

Proof. By the standard forward regret analysis, we get

$$\sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \inf_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} \ell_t(\mathbf{x}) \leq \sum_{t=1}^{T} \ell_t(\mathbf{x}_t) - \sum_{t=1}^{T} \ell_t(\mathbf{x}_{t+1})$$

Now, by using the strong convexity of the loss functions, and the fact that the strong convexity property is additive, we get

$$\sum_{\tau=1}^{t-1} \ell_\tau(\mathbf{x}_{t+1}) \geq \sum_{\tau=1}^{t-1} \ell_\tau(\mathbf{x}_t) + \frac{\alpha(t-1)}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2^2$$
$$\sum_{\tau=1}^{t} \ell_\tau(\mathbf{x}_t) \geq \sum_{\tau=1}^{t} \ell_\tau(\mathbf{x}_{t+1}) + \frac{\alpha t}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2^2,$$

which gives us

$$\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x}_{t+1}) \geq \frac{\alpha}{2}(2t-1) \cdot \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2^2 .$$

However, we get $\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x}_{t+1}) \leq L \cdot \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2$ by invoking Lipschitz-ness of the loss functions. This gives us

$$\|\mathbf{x}_t - \mathbf{x}_{t+1}\|_2 \leq \frac{2L}{\alpha(2t-1)}.$$

This, upon applying Lipschitz-ness again, gives us

$$\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x}_{t+1}) \leq \frac{2L^2}{\alpha(2t-1)}.$$

Summing over all the time steps gives us the desired result. □

For the rest of the proof, we shall use the shorthand notation that we used in Algorithm 4, i.e.

$$\begin{aligned}
\boldsymbol{\alpha}_t &= (\alpha_{t,1}, \alpha_{t,2}), && \text{(the dual variables for } \zeta_1\text{)}\\
\boldsymbol{\beta}_t &= (\beta_{t,1}, \beta_{t,2}), && \text{(the dual variables for } \zeta_2\text{)}\\
\boldsymbol{\gamma}_t &= (\gamma_{t,1}, \gamma_{t,2}), && \text{(the dual variables for } \Psi\text{)}
\end{aligned}$$

We will also use additional notation

$$\begin{aligned}
\mathbf{s}_t &= (r^+(\mathbf{w}_t; \mathbf{x}_t, y_t), r^-(\mathbf{w}_t; \mathbf{x}_t, y_t)),\\
\mathbf{p}_t &= \left(\boldsymbol{\alpha}_t^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha}_t), \boldsymbol{\beta}_t^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta}_t)\right),\\
\boldsymbol{\ell}_t(\mathbf{w}) &= (r^+(\mathbf{w}; \mathbf{x}_t, y_t), r^-(\mathbf{w}; \mathbf{x}_t, y_t))\\
R(\mathbf{w}) &= (P(\mathbf{w}), N(\mathbf{w}))
\end{aligned}$$

Note that $\boldsymbol{\ell}_t(\mathbf{w}_t) = \mathbf{s}_t$. We now define a quantity that we shall be analyzing to obtain the convergence bound

$$(A) = \sum_{t=1}^{T} (\boldsymbol{\gamma}_t^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma}_t))$$

Now, since Ψ is β_Ψ-smooth and concave, by Theorem 2, we know that Ψ^* is $\frac{1}{\beta}$-strongly concave. However that means that the loss function $g_t(\boldsymbol{\gamma}) := \boldsymbol{\gamma}^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma})$ is $\frac{1}{\beta}$-strongly convex. Now Algorithm 4 (step 17) implements

$$\boldsymbol{\gamma}_t = \arg\min_{\boldsymbol{\gamma}} \left\{\boldsymbol{\gamma} \cdot \mathbf{q}_t - \Psi^*(\boldsymbol{\gamma})\right\},$$

where $\mathbf{q}_t = \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{p}_\tau$ (see steps 7, 10, 13 that update $\mathbf{q}_t$). Notice that this is identical to the FTL algorithm with the losses $g_t(\boldsymbol{\gamma}) = \mathbf{p}_t \cdot \boldsymbol{\gamma} - \Psi^*(\boldsymbol{\gamma})$ which are strongly convex, and can be shown to be $(B_r(L_{\zeta_1} + L_{\zeta_2}))$-Lipschitz, neglecting universal constants. Thus, by an application of Lemma 8, we get, upto universal constants

$$(A) \leq \inf_{\boldsymbol{\gamma}} \left\{ \sum_{t=1}^{T} (\boldsymbol{\gamma}^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma})) \right\} + \beta_\Psi (B_r(L_{\zeta_1} + L_{\zeta_2}))^2 \log T$$

The same technique, along with the observation that steps 15 and 16 of Algorithm 4 also implement the FTL algorithm, can be used to get the following results upto universal constants

$$\sum_{t=1}^{T} (\boldsymbol{\alpha}_t^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha}_t)) \leq \inf_{\boldsymbol{\alpha}} \left\{ \sum_{t=1}^{T} (\boldsymbol{\alpha}^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha})) \right\} + \beta_{\zeta_1} (B_r L_{\zeta_1})^2 \log T,$$

and

$$\sum_{t=1}^{T} (\boldsymbol{\beta}_t^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta}_t)) \leq \inf_{\boldsymbol{\beta}} \left\{ \sum_{t=1}^{T} (\boldsymbol{\beta}^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta})) \right\} + \beta_{\zeta_2} (B_r L_{\zeta_2})^2 \log T.$$

This gives us, for

$$\Delta_1 = \beta_\Psi (B_r(L_{\zeta_1} + L_{\zeta_2}))^2 + \beta_{\zeta_1} (B_r L_{\zeta_1})^2 + \beta_{\zeta_2} (B_r L_{\zeta_2})^2,$$

$$\begin{aligned}
(A) &\leq \inf_{\boldsymbol{\gamma}} \left\{ \sum_{t=1}^{T} (\boldsymbol{\gamma}^\top \mathbf{p}_t - \Psi^*(\boldsymbol{\gamma})) \right\} + \Delta_1 \log T \\
&= \inf_{\boldsymbol{\gamma}} \left\{ \gamma_1 \sum_{t=1}^{T} (\boldsymbol{\alpha}_t^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_2 \sum_{t=1}^{T} (\boldsymbol{\beta}_t^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta}_t)) - \Psi^*(\boldsymbol{\gamma}) \right\} + \Delta_1 \log T \\
&\leq \inf_{\boldsymbol{\gamma}, \boldsymbol{\alpha}, \boldsymbol{\beta}} \left\{ \gamma_1 \sum_{t=1}^{T} (\boldsymbol{\alpha}^\top \mathbf{s}_t - \zeta_1^*(\boldsymbol{\alpha})) + \gamma_2 \sum_{t=1}^{T} (\boldsymbol{\beta}^\top \mathbf{s}_t - \zeta_2^*(\boldsymbol{\beta})) - \Psi^*(\boldsymbol{\gamma}) \right\} + \Delta_1 \log T
\end{aligned}$$

Now, because of the stochastic nature of the samples, we have

$$\mathbb{E}\mathbf{s}_t | \{(\mathbf{x}_\tau, y_\tau)\}_{\tau=1}^{t-1} = \mathbb{E}\boldsymbol{\ell}_t(\mathbf{w}_t) | \{(\mathbf{x}_\tau, y_\tau)\}_{\tau=1}^{t-1} = R(\mathbf{w}_t)$$

$$\begin{aligned}
\frac{(A)}{T} &\leq \inf_{\boldsymbol{\gamma}} \left\{ \gamma_1 \inf_{\boldsymbol{\alpha}} \left\{ \boldsymbol{\alpha}^\top \left(\frac{1}{T} \sum_{t=1}^{T} R(\mathbf{w}_t) \right) - \zeta_1^*(\boldsymbol{\alpha}) \right\} \right. \\
&\qquad \left. + \left\{ \gamma_2 \inf_{\boldsymbol{\beta}} \left\{ \boldsymbol{\beta}^\top \left(\frac{1}{T} \sum_{t=1}^{T} R(\mathbf{w}_t) \right) - \zeta_2^*(\boldsymbol{\beta}) \right\} - \Psi^*(\boldsymbol{\gamma}) \right\} \right. \\
&\quad + \frac{\Delta_1}{T} \left(\log T + \log \frac{1}{\delta} \right) \\
&= \inf_{\boldsymbol{\gamma}} \left(\gamma_1 \zeta_1 \left(\frac{1}{T} \sum_{t=1}^{T} R(\mathbf{w}_t) \right) + \gamma_2 \zeta_2 \left(\frac{1}{T} \sum_{t=1}^{T} R(\mathbf{w}_t) \right) - \Psi^*(\boldsymbol{\gamma}) \right) \\
&\quad + \frac{\Delta_1}{T} \left(\log T + \log \frac{1}{\delta} \right) \\
&\leq \inf_{\boldsymbol{\gamma}} \left(\gamma_1 \zeta_1 (R(\overline{\mathbf{w}})) + \gamma_2 \zeta_2 (R(\overline{\mathbf{w}})) - \Psi^*(\boldsymbol{\gamma}) \right) + \frac{\Delta_1 \log \frac{T}{\delta}}{T} \\
&= \Psi(\zeta_1(R(\overline{\mathbf{w}})), \zeta_2(R(\overline{\mathbf{w}}))) + \frac{\Delta_1 \log \frac{T}{\delta}}{T},
\end{aligned}$$

where the second last step follows from the Jensen's inequality, the concavity of the functions $P(\mathbf{w})$ and $N(\mathbf{w})$, and the assumption that ζ_1 and ζ_2 are increasing functions of both their arguments. Thus, we have, with probability at least $1 - \delta$,

$$(A) \leq T \cdot \Psi(\zeta_1(R(\overline{\mathbf{w}})), \zeta_2(R(\overline{\mathbf{w}}))) + \Delta_1 \log \frac{T}{\delta}$$

Note that this is a much stronger bound than what Narasimhan *et al* [80] obtain for their gradient descent based dual updates. This, in some sense, establishes the superiority of the follow-the-leader type algorithms used by **NEMSIS**.

$$h_t(\mathbf{w}) = \gamma_{t,1}(\boldsymbol{\alpha}_t^\top \boldsymbol{\ell}_t(\mathbf{w}) - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_{t,2}(\boldsymbol{\beta}_t^\top \boldsymbol{\ell}_t(\mathbf{w}) - \zeta_2^*(\boldsymbol{\beta}_t)) - \Psi^*(\boldsymbol{\gamma}_t)$$

Since the functions $h_t(\cdot)$ are concave and $(L_\Psi L_r (L_{\zeta_1} + L_{\zeta_2}))$-Lipschitz (due to assumptions on the smoothness and values of the reward functions), the standard regret analysis for online gradient ascent (for example [109]) gives us the following bound on (A), ignoring universal constants

$$\begin{aligned}
(A) &= \sum_{t=1}^{T} h_t(\mathbf{w}_t) \\
&= \sum_{t=1}^{T} \gamma_{t,1}(\boldsymbol{\alpha}_t^\top \boldsymbol{\ell}_t(\mathbf{w}_t) - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_{t,2}(\boldsymbol{\beta}_t^\top \boldsymbol{\ell}_t(\mathbf{w}_t) - \zeta_2^*(\boldsymbol{\beta}_t)) - \Psi^*(\boldsymbol{\gamma}_t) \\
&\geq \sum_{t=1}^{T} \gamma_{t,1}(\boldsymbol{\alpha}_t^\top \boldsymbol{\ell}_t(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha}_t)) + \gamma_{t,2}(\boldsymbol{\beta}_t^\top \boldsymbol{\ell}_t(\mathbf{w}^*) - \zeta_2^*(\boldsymbol{\beta}_t)) \\
&\quad - \Psi^*(\boldsymbol{\gamma}_t) - \Delta_2 \sqrt{T},
\end{aligned}$$

where $\Delta_2 = (L_\Psi L_r(L_{\zeta_1} + L_{\zeta_2}))$. Note that the above results hold since we used step lengths $\eta_t = \Theta(1/\sqrt{t})$. To achieve the above bounds precisely, η_t will have to be tuned to the Lipschitz constant of the functions $h_t(\cdot)$ and for sake of simplicity we assume that the step lengths are indeed tuned so. We also assume, to get the above result , without loss of generality of course, that the model space $\mathcal{W}$ is the unit norm ball in $\mathbb{R}^d$. Applying a standard online-to-batch conversion bound (for example [23]), then gives us, with probability at least $1-\delta$,

$$\begin{aligned}\frac{(A)}{T} \geq\ & \underbrace{\frac{1}{T}\sum_{t=1}^{T}\gamma_{t,1}(\boldsymbol{\alpha}_t^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha}_t))}_{(B)} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}\gamma_{t,2}(\boldsymbol{\beta}_t^\top R(\mathbf{w}^*) - \zeta_2^*(\boldsymbol{\beta}_t))}_{(C)} \\ & - \frac{1}{T}\sum_{t=1}^{T}\Psi^*(\boldsymbol{\gamma}_t) - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}},\end{aligned}$$

where $\Delta_3 = \Delta_2 + L_\Psi B_r(L_{\zeta_1} + L_{\zeta_2})$. Analyzing the expression (B) gives us

$$\begin{aligned}(B) &= \frac{1}{T}\sum_{t=1}^{T}\gamma_{t,1}(\boldsymbol{\alpha}_t^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha}_t)) \\ &= \frac{\sum_{t=1}^T \gamma_{t,1}}{T}\left(\left(\sum_{t=1}^{T}\frac{\gamma_{t,1}\boldsymbol{\alpha}_t}{\sum_{t=1}^T\gamma_{t,1}}\right)^\top R(\mathbf{w}^*) - \sum_{t=1}^{T}\frac{\gamma_{t,1}}{\sum_{t=1}^T\gamma_{t,1}}\zeta_1^*(\boldsymbol{\alpha}_t)\right) \\ &\geq \frac{\sum_{t=1}^T \gamma_{t,1}}{T}\left(\left(\sum_{t=1}^{T}\frac{\gamma_{t,1}\boldsymbol{\alpha}_t}{\sum_{t=1}^T\gamma_{t,1}}\right)^\top R(\mathbf{w}^*) - \zeta_1^*\left(\sum_{t=1}^{T}\frac{\gamma_{t,1}\boldsymbol{\alpha}_t}{\sum_{t=1}^T\gamma_{t,1}}\right)\right) \\ &\geq \frac{\sum_{t=1}^T \gamma_{t,1}}{T}\min_{\boldsymbol{\alpha}}\left\{\boldsymbol{\alpha}^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha})\right\} \\ &= \bar{\gamma}_1\min_{\boldsymbol{\alpha}}\left\{\boldsymbol{\alpha}^\top R(\mathbf{w}^*) - \zeta_1^*(\boldsymbol{\alpha})\right\} = \bar{\gamma}_1\zeta_1(R(\mathbf{w}^*))\end{aligned}$$

A similar analysis for (C) follows and we get, ignoring universal constants,

$$\begin{aligned}\frac{(A)}{T} &\geq \bar{\gamma}_1\zeta_1(R(\mathbf{w}^*)) + \bar{\gamma}_2\zeta_2(R(\mathbf{w}^*)) - \frac{1}{T}\sum_{t=1}^{T}\Psi^*(\boldsymbol{\gamma}_t) - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}} \\ &\geq \bar{\gamma}_1\zeta_1(R(\mathbf{w}^*)) + \bar{\gamma}_2\zeta_2(R(\mathbf{w}^*)) - \Psi^*(\bar{\boldsymbol{\gamma}}) - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}} \\ &\geq \min_{\boldsymbol{\gamma}}\left\{\gamma_1\zeta_1(R(\mathbf{w}^*)) + \gamma_1\zeta_2(R(\mathbf{w}^*)) - \Psi^*(\boldsymbol{\gamma})\right\} - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}} \\ &= \Psi(\zeta_1(R(\mathbf{w}^*)), \zeta_2(R(\mathbf{w}^*))) - \Delta_3\frac{\log\frac{1}{\delta}}{\sqrt{T}}\end{aligned}$$

Thus, we have with probability at least $1-\delta$,

$$(A) \geq T \cdot \Psi(\zeta_1(R(\mathbf{w}^*)), \zeta_2(R(\mathbf{w}^*))) - \Delta_3 \log\frac{1}{\delta}\sqrt{T}$$

Combining the upper and lower bounds on (A) finishes the proof since $\Delta_3 \log\frac{1}{\delta}\sqrt{T}$ overwhelms the term $\Delta_1 \log\frac{T}{\delta}$.

5.9 Proof of Theorem 5

We will prove the result by proving a sequence of claims. The first claim ensures that the distance to the optimum performance value is bounded by the performance value we obtain in terms of the valuation function at any step. For notational simplicity, we will use the shorthand $\mathcal{P}(\mathbf{w}) := \mathcal{P}_{(\mathcal{P}_{\text{quant}},\mathcal{P}_{\text{class}})}(\mathbf{w})$.

Claim 9. *$\mathcal{P}^* := \sup_{\mathbf{w}\in\mathcal{W}} \mathcal{P}(\mathbf{w})$ be the optimal performance level. Also, define $e_t = V(\mathbf{w}_{t+1}, v_t)$. Then, for any t, we have*

$$\mathcal{P}^* - \mathcal{P}(\mathbf{w}_t) \leq \frac{e_t}{m}$$

Proof. We will prove the result by contradiction. Suppose $\mathcal{P}^* > \mathcal{P}(\mathbf{w}_t) + \frac{e_t}{m}$. Then there must exist some $\widetilde{\mathbf{w}} \in \mathcal{W}$ such that

$$\mathcal{P}(\widetilde{\mathbf{w}}) = \frac{e_t}{m} + \mathcal{P}(\mathbf{w}_t) + e' = \frac{e_t}{m} + v_t + e' =: v',$$

where $e' > 0$. Note that the above uses the fact that we set $v_t = \mathcal{P}(\mathbf{w}_t)$. Then we have

$$V(\widetilde{\mathbf{w}}, v_t) - e_t = \mathcal{P}_{\text{class}}(\widetilde{\mathbf{w}}) - v_t \cdot \mathcal{P}_{\text{quant}}(\widetilde{\mathbf{w}}) - e_t.$$

Now since $\mathcal{P}(\widetilde{\mathbf{w}}) = v'$, we have $\mathcal{P}_{\text{class}}(\widetilde{\mathbf{w}}) - v' \cdot \mathcal{P}_{\text{quant}}(\widetilde{\mathbf{w}}) = 0$ which gives us

$$V(\widetilde{\mathbf{w}}, v_t) - e_t = \left(\frac{e_t}{m} + e'\right)\mathcal{P}_{\text{quant}}(\widetilde{\mathbf{w}}) - e_t \geq \left(\frac{e_t}{m} + e'\right)m - e_t > 0.$$

But this contradicts the fact that $\max_{\mathbf{w}\in\mathcal{W}} V(\mathbf{w}, v_t) = e_t$ which is ensured by step 4 of Algorithm 5. This completes the proof. □

The second claim then establishes that in case we do get a large performance value in terms of the valuation function at any time step, the next iterate will have a large leap in performance in terms of the original performance function $\mathcal{P}$.

Claim 10. *For any time instant t we have*

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq \mathcal{P}(\mathbf{w}_t) + \frac{e_t}{M}$$

Proof. By our definition, we have $V(\mathbf{w}_{t+1}, v_t) = e_t$. This gives us

$$\begin{aligned}\frac{\mathcal{P}_{\text{class}}(\mathbf{w}_{t+1})}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})} - \left(v_t + \frac{e_t}{M}\right) &\geq \frac{\mathcal{P}_{\text{class}}(\mathbf{w}_{t+1})}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})} - \left(v_t + \frac{e_t}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})}\right) \\ &= \frac{v_t \cdot \mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})}{\mathcal{P}_{\text{quant}}(\mathbf{w}_{t+1})} - v_t = 0,\end{aligned}$$

which proves the result. □

We are now ready to establish the convergence proof. Let $\Delta_t = \mathcal{P}^* - \mathcal{P}(\mathbf{w}_t)$. Then we have, by Claim 9

$$e_t \geq m \cdot \Delta_t,$$

and also

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq \mathcal{P}(\mathbf{w}_t) + \frac{e_t}{M},$$

by Claim 10. Subtracting both sides of the above equation from $\mathcal{P}^*$ gives us

$$\begin{aligned}\Delta_{t+1} &= \Delta_t - \frac{e_t}{M} \\ &\leq \Delta_t - \frac{m}{M} \cdot \Delta_t = \left(1 - \frac{m}{M}\right) \cdot \Delta_t,\end{aligned}$$

which concludes the convergence proof.

5.10 Proof of Theorem 6

To prove this theorem, we will first show that the **CAN** algorithm is robust to imprecise updates. More precisely, we will assume that Algorithm 5 only ensures that in step 4 we have

$$V(\mathbf{w}_{t+1}, v_t) = \max_{\mathbf{w} \in \mathcal{W}} V(\mathbf{w}, v_t) - \epsilon_t,$$

where $\epsilon_t > 0$ and step 5 only ensures that

$$v_t = \mathcal{P}(\mathbf{w}_t) + \delta_t,$$

where δ_t may be positive or negative. For this section, we will redefine

$$e_t = \max_{\mathbf{w} \in \mathcal{W}} V(\mathbf{w}, v_t)$$

since we can no longer assume that $V(\mathbf{w}_{t+1}, v_t) = e_t$. Note that if v_t is an unrealizable value, i.e. for no predictor $\mathbf{w} \in \mathcal{W}$ is $\mathcal{P}(\mathbf{w}) \geq v_t$, then we have $e_t < 0$. Having this we establish the following results:

Lemma 11. *Given the previous assumptions on the imprecise execution of Algorithm 5, the following is true*

1. *If* $\delta_t \leq 0$ *then* $e_t \geq 0$
2. *If* $\delta_t > 0$ *then* $e_t \geq -\delta_t \cdot M$
3. *We have* $\mathcal{P}^* < v_t$ *iff* $e_t < 0$
4. *If* $e_t \geq 0$ *then* $e_t \geq m(\mathcal{P}^* - v_t)$
5. *If* $e_t < 0$ *then* $e_t \geq M(\mathcal{P}^* - v_t)$
6. *If* $V(\mathbf{w}, v) = e$ *for* $e \geq 0$ *then* $\mathcal{P}(\mathbf{w}) \geq v + \frac{e}{M}$
7. *If* $V(\mathbf{w}, v) = e$ *for* $e < 0$ *then* $\mathcal{P}(\mathbf{w}) \geq v + \frac{e}{m}$

Proof. We prove the parts separately below

1. Since $\delta_t < 0$, there exists some $\mathbf{w} \in \mathcal{W}$ such that $\mathcal{P}(\mathbf{w}) > v_t$. The result then follows.
2. If $v_t = \mathcal{P}(\mathbf{w}_t) + \delta_t$ then $V(\mathbf{w}_t, v_t) \geq -\delta_t \cdot M$ .The result then follows.
3. Had $e_t \geq 0$ been the case, we would have had, for some $\mathbf{w} \in \mathcal{W}$, $V(\mathbf{w}, v_t) \geq 0$ which would have implied $\mathcal{P}(\mathbf{w}) \geq v_t$ which contradicts $\mathcal{P}^* < v_t$. For the other direction, suppose $\mathcal{P}^* = \mathcal{P}(\mathbf{w}^*) = v_t + e'$ with $e' > 0$. Then we have $e_t = V(\mathbf{w}^*, v_t) > 0$ which contradicts $e_t < 0$.
4. Observe that the proof of Claim 9 suffices, by simply replacing $\mathcal{P}(\mathbf{w}_t)$ with v_t in the statement.
5. Assume the contrapositive that for some $\mathbf{w} \in \mathcal{W}$, we have $\mathcal{P}(\mathbf{w}) = v_t + \frac{e_t}{M} + e'$ where $e' > 0$. We can then show that $V(\mathbf{w}, v_t) = \left(\frac{e_t}{M} + e'\right)\mathcal{P}_{\text{quant}}(\mathbf{w}) \geq e_t + e' \cdot \mathcal{P}_{\text{quant}}(\mathbf{w}) > e_t$ which contradicts the definition of e_t. Note that since $e_t < 0$, we have $\frac{e_t}{M} \geq \frac{e_t}{\mathcal{P}_{\text{quant}}(\mathbf{w})}$ and we have $\mathcal{P}_{\text{quant}}(\mathbf{w}) \geq m > 0$.
6. Observe that the proof of Claim 10 suffices, by simply replacing $\mathcal{P}(\mathbf{w}_t)$ with v_t in the statement.
7. We have $\mathcal{P}_{\text{class}}(\mathbf{w}) - v \cdot \mathcal{P}_{\text{quant}}(\mathbf{w}) = e$. Dividing throughout by $\mathcal{P}_{\text{quant}}(\mathbf{w}) > 0$ and using $\frac{e}{\mathcal{P}_{\text{quant}}(\mathbf{w})} \geq \frac{e}{m}$ since $e < 0$ gives us the result.

This finishes the proofs. □

Using these results, we can now make the following claim on the progress made by **CAN** with imprecise updates.

Lemma 12. *Even if **CAN** is executed with noisy updates, at any time step t, we have*

$$\Delta_{t+1} \leq \left(1 - \frac{m}{M}\right)\Delta_t + \frac{M}{m} \cdot |\delta_t| + \frac{\epsilon_t}{m}.$$

Proof. We analyze time steps when $\delta_t \leq 0$ separately from time steps when $\delta_t < 0$.

Case 1: $\boldsymbol{\delta_t \leq 0}$ In these time steps, the method underestimates the performance of the current predictor but gives a legal i.e. realizable value of v_t. We first deduce that for these time steps, using Lemma 11 part 1, we have $e_t \geq 0$ and then using part 4, we have $e_t \geq m(\mathcal{P}^* - v_t)$. This combined with the identity $\mathcal{P}^* - v_t = \Delta_t - \delta_t$, gives us

$$e_t \geq m(\Delta_t - \delta_t)$$

Now we have, by definition, $V(\mathbf{w}_{t+1}, v_t) = e_t - \epsilon_t$ (note that both $e_t, \epsilon \geq 0$ in this case). The next steps depend on whether this quantity is positive or negative. If $\epsilon_t \leq e_t$, we apply Lemma 11 part 6 to get

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq v_t + \frac{e_t - \epsilon_t}{M},$$

which gives us upon using $\mathcal{P}^* - v_t = \Delta_t - \delta_t$ and $e_t \geq m(\Delta_t - \delta_t)$,

$$\Delta_{t+1} \leq \left(1 - \frac{m}{M}\right)\Delta_t - \left(1 - \frac{m}{M}\right)\delta_t + \frac{\epsilon_t}{M}$$

Otherwise if $\epsilon_t > e_t$ then we have actually made negative progress at this time step since $V(\mathbf{w}_{t+1}, v_t) < 0$. To safeguard us against how much we go back in terms of progress, we us Lemma 11 part 7 to guarantee

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq v_t + \frac{e_t - \epsilon_t}{m},$$

which gives us upon using $\mathcal{P}^* - v_t = \Delta_t - \delta_t$ and $e_t \geq m(\Delta_t - \delta_t)$,

$$\Delta_{t+1} \leq \frac{\epsilon_t}{m},$$

Note however, that we are bound by $\epsilon_t > e_t$ in the above statement. We now move on to analyze the second case.

Case 2: $\boldsymbol{\delta_t > 0}$ In these time steps, the method is overestimating the performance of the current predictor and runs a risk of giving a value of v_t that is unrealizable. We cannot hope to make much progress in these time steps. The following analysis simply safeguards us against too much deterioration. There are two subcases we explore here: first we look at the case where $v_t \leq \mathcal{P}^*$ i.e. v_t is still a legal, realizable performance value. In this case we continue to have $e_t \geq 0$ and the analysis of the previous case (i.e. $\delta_t \leq 0$) continues to apply.

However, if $v_t > \mathcal{P}^*$, we are setting an unrealizable value of v_t. Using Lemma 11 part 3 gives us $e_t < 0$ which, upon using part 5 of the lemma gives us

$$e_t \geq M(\mathcal{P}^* - v_t).$$

In this case, we have $V(\mathbf{w}_{t+1}, v_t) = e_t - \epsilon_t < 0$ since $e_t < 0$ and $\epsilon_t > 0$. Thus, using Lemma 11 part 7 gives us

$$\mathcal{P}(\mathbf{w}_{t+1}) \geq v_t + \frac{e_t - \epsilon_t}{m}$$

which upon manipulation, as before, gives us

$$\Delta_{t+1} \leq \left(1 - \frac{M}{m}\right)\Delta_t + \left(\frac{M}{m} - 1\right)\delta_t + \frac{\epsilon_t}{m} \leq \left(\frac{M}{m} - 1\right)\delta_t + \frac{\epsilon_t}{m},$$

where the last step uses the fact that $\Delta_t \geq 0$ and $M \geq m$. Putting all these cases together and using the fact that the quantities $\Delta_t, \epsilon_t, |\delta_t|$ are always positive gives us

$$\begin{aligned}\Delta_{t+1} &\leq \left(1 - \frac{m}{M}\right)\Delta_t + \left(\frac{M^2 - m^2}{mM}\right)|\delta_t| + \frac{\epsilon_t}{m} \\ &\leq \left(1 - \frac{m}{M}\right)\Delta_t + \frac{M}{m} \cdot |\delta_t| + \frac{\epsilon_t}{m},\end{aligned}$$

which finishes the proof. □

From hereon simple manipulations similar to those used to analyze the **STAMP** algorithm in [80] can be used, along with the guarantees provided by Theorem 3 for the **NEMSIS** analysis to finish the proof of the result. We basically have to use the fact that the **NEMSIS** invocations in **SCAN** (Algorithm 6 line 8), as well as the performance estimation steps (Algorithm 6 lines 14-19) can be seen as executing noisy updates for the original **CAN** algorithm.

Bibliography

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Proc. NIPS*, 2011.

[2] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *ICML*, 2013.

[3] Rocío Alaíz-Rodríguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift. *Neurocomputing*, 74(16):2614–2623, 2011.

[4] J.-Y. Audibert, R. Munos, and C. Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[5] P. Auer. Using confidence bounds for exploration-exploitation trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 2001.

[7] M. G. Azar, A. Lazaric, and E. Brunskill. Sequential transfer in multi-armed bandit with finite set of models. In *NIPS*, pages 2220–2228, 2013.

[8] Georgios Balikas, Ioannis Partalas, Eric Gaussier, Rohit Babbar, and Massih-Reza Amini. Efficient model selection for regularized classification by exploiting unlabeled data. In *Proceedings of the 14th International Symposium on Intelligent Data Analysis (IDA 2015)*, pages 25–36, Saint Etienne, FR, 2015.

[9] José Barranquero, Jorge Díez, and Juan José del Coz. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition*, 48(2):591–604, 2015.

[10] Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. Quantification via probability estimators. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010)*, pages 737–742, Sydney, AU, 2010.

[11] T. Bogers. Movie recommendation using random walks over the contextual graph. In *CARS'10: Proc. 2nd Workshop on Context-Aware Recommender Systems*, 2010.

[12] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2508–2530, 2006.

[13] G. Bresler, G. Chen, and Shah D. A latent source model for online collaborative filtering. In *NIPS*. MIT Press, 2014.

[14] E. Brunskill and L. Li. Sample complexity of multi-task reinforcement learning. In *UAI*, 2013.

[15] S. Buccapatnam, A. Eryilmaz, and N.B. Shroff. Multi-armed bandits in the presence of side observations in social networks. In *Proc. 52nd IEEE Conference on Decision and Control*, 2013.

[16] R. Burke. Hybrid systems for personalized recommendations. In *Proc. of the 2003 ITWP*, pages 133–152, 2005.

[17] G. Buscher, R. W. White, S. Dumais, and J. Huang. Large-scale analysis of individual and task differences in search result page examination strategies. In *Proc. 5th ACM WSDM*, pages 373–382, 2012.

[18] W. Cao, J. Li, Y. Tao, and Z. Li. On top-k selection in multi-armed bandits and hidden bipartite graphs. In *Proc. NIPS*, 2015.

[19] S. Caron and S. Bhagat. Mixing bandits: A recipe for improved cold-start recommendations in a social network. In *SNA-KDD, 7th Workshop on Social Network Mining and Analysis*, 2013.

[20] S. Caron, B. Kveton, M. Lelarge, and S. Bhagat. Leveraging side observations in stochastic bandits. In *Proc. UAI*, pages 142–151, 2012.

[21] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69/2:143–167, 2007.

[22] N. Cesa-Bianchi and C. Gentile. Improved risk tail bounds for on-line algorithms. *IEEE Trans. on Information Theory*, 54(1):386–390, 2008.

[23] Nicoló Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. In *Proceedings of the 15th Annual Conference on Neural Information Processing Systems (NIPS 2001)*, pages 359–366, Vancouver, USA, 2001.

[24] Yee Seng Chan and Hwee Tou Ng. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006)*, pages 89–96, Sydney, AU, 2006.

[25] W. Chu, L. Li, L. Reyzin, and R. E Schapire. Contextual bandits with linear payoff functions. In *Proc. AISTATS*, 2011.

[26] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. McGraw Hill, 1990.

[27] K. Crammer and C. Gentile. Multiclass classification with bandit feedback using adaptive regularization. In *Proc. ICML*, 2011.

[28] Imre Csiszár and Paul C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004.

[29] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, pages 355–366, 2008.

[30] O. Dekel, C. Gentile, and K. Sridharan. Robust selective sampling from single and multiple teachers. In *COLT*, pages 346–358, 2010.

[31] J. Delporte, A. Karatzoglou, T. Matuszczyk, and S. Canu. Socially enabled preference learning from implicit feedback data. In *Proc. ECML/PKDD*, pages 145–160, 2013.

[32] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. 7th KDD*, pages 269–274. ACM, 2001.

[33] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proc. 9th KDD*, pages 89–98, New York, NY, USA, 2003. ACM.

[34] J. Djolonga, A. Krause, and V. Cevher. High-dimensional gaussian process bandits. In *NIPS*, pages 1025–1033, 2013.

[35] Marthinus C. du Plessis and Masashi Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, UK, 2012.

[36] M. Dudik, D. Erhan, J. Langford, and L. Li. Sample-efficient nonstationary-policy evaluation for contextual bandits. In *UAI*, 2012.

[37] Andrea Esuli and Fabrizio Sebastiani. Sentiment quantification. *IEEE Intelligent Systems*, 25(4):72–75, 2010.

[38] Andrea Esuli and Fabrizio Sebastiani. Optimizing text quantifiers for multivariate loss functions. *ACM Transactions on Knowledge Discovery and Data*, 9(4):Article 27, 2015.

[39] George Forman. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206, 2008.

[40] Wei Gao and Fabrizio Sebastiani. Tweet sentiment: From classification to quantification. In *Proceedings of the 7th International Conference on Advances in Social Network Analysis and Mining (ASONAM 2015)*, pages 97–104, Paris, FR, 2015.

[41] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, Bejing, CN, 2014.

[42] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proc. 5th ICDM*, pages 625–628. IEEE Computer Society, 2005.

[43] Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. Class distribution estimation based on the Hellinger distance. *Information Sciences*, 218:146–164, 2013.

[44] Huijuan Guo, Yi Feng, Fei Hao, Shentong Zhong, and Shuai Li. Dynamic fuzzy logic control of genetic algorithm probabilities. *Journal of Computers*, 9(1):22–27, 2014.

[45] Fei Hao, Shuai Li, Geyong Min, Hee-Cheol Kim, Stephen Yau, and Laurence Yang. An efficient approach to generating location-sensitive recommendations in ad-hoc social network environments. *IEEE Transactions on Services Computing*, 8(3):520–533, 2015.

[46] Fei Hao, Doo-Soon Park, Shuai Li, and Hwa Min Lee. Mining -maximal cliques from a fuzzy graph. *Journal of Sustainability*, 8(6):553, 2016.

[47] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic Regret Algorithms for Online Convex Optimization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006)*, pages 499–513, Pittsburgh, USA, 2006.

[48] Daniel J. Hopkins and Gary King. A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229–247, 2010.

[49] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. on Computer Systems*, 23(3):219–252, August 2005.

[50] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems*, 25(3):8, 2007.

[51] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 377–384, Bonn, DE, 2005.

[52] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[53] Robert N. Jorissen and Michael K. Gilson. Virtual screening of molecular databases using a support vector machine. *Jounal of Chemical Information Modelling*, 45(3):549–561, 2005.

[54] Dileep Kalathil, Naumaan Nayyar, and Rahul Jain. Decentralized learning for multiplayer multiarmed bandits. *IEEE Transactions on Information Theory*, 60(4):2331–2345, 2014.

[55] Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proc. SODA*, pages 1131–1142, 2013.

[56] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Online and stochastic gradient methods for non-decomposable loss functions. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS 2014)*, pages 694–702, Montreal, USA, 2014.

[57] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 189–198, Lille, FR, 2015.

[58] D. R. Karger. Random sampling in cut, flow, and network design problems. In *Proc. STOC*, 1994.

[59] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

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

[61] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491. IEEE Computer Society, 2003.

[62] Gary King and Ying Lu. Verbal autopsy methods with multiple causes of death. *Statistical Science*, 23(1):78–91, 2008.

[63] Nathan Korda, Balázs Szörényi, and Shuai Li. Distributed clustering of linear bandits in peer-to-peer networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, New York, US, 2016.

[64] A. Krause and C.S. Ong. Contextual gaussian process bandit optimization. In *Proc. 25th NIPS*, 2011.

[65] B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *Proc. ICML*, 2015.

[66] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[67] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proc. NIPS*, 2007.

[68] David D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1995)*, pages 246–254, Seattle, USA, 1995.

[69] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. WWW*, pages 661–670, 2010.

[70] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. WSDM*, 2011.

[71] Shuai Li, Claudio Gentile, and Alexandros Karatzoglou. Graph clustering bandits for recommendation. *CoRR:1605.00596*, 2016.

[72] Shuai Li, Claudio Gentile, Alexandros Karatzoglou, and Giovanni Zappella. Data-dependent clustering in exploration-exploitation algorithms. In *arXiv*, 2015.

[73] Shuai Li, Claudio Gentile, Alexandros Karatzoglou, and Giovanni Zappella. Online context-dependent clustering in recommendations based on exploration-exploitation algorithms. In *arXiv*, 2015.

[74] Shuai Li, Fei Hao, Mei Li, and Hee-Cheol Kim. Medicine rating prediction and recommendation in mobile social networks. In *International Conference on Green and Pervasive Computing*, pages 216–223, 2013.

[75] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative filtering bandits. In *Proceedings of the 39th ACM International Conference on Research and Development in Information Retrieval (SIGIR 2016)*, Pisa, IT, 2016.

[76] O. Maillard and S. Mannor. Latent bandits. In *ICML*, 2014.

[77] P. Massart. *Concentration Inequalities and Model Selection*. Volume 1896 of Lecture Notes in Mathematics. Springer, Berlin, 2007.

[78] Letizia Milli, Anna Monreale, Giulio Rossetti, Fosca Giannotti, Dino Pedreschi, and Fabrizio Sebastiani. Quantification trees. In *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM 2013)*, pages 528–536, Dallas, USA, 2013.

[79] E. Moroshko, N. Vaits, and K. Crammer. Second-order non-stationary online learning for regression. *Journal of Machine Learning Research*, 16:1481–1517, 2015.

[80] Harikrishna Narasimhan, Purushottam Kar, and Prateek Jain. Optimizing non-decomposable performance measures: A tale of two classes. In *proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 199–208, Lille, FR, 2015.

[81] Naumaan Nayyar, Dileep Kalathil, and Rahul Jain. On regret-optimal learning in decentralized multi-player multi-armed bandits. *CoRR:1505.00553*, 2015.

[82] Trong T. Nguyen and Hady W. Lauw. Dynamic clustering of contextual multi-armed bandits. In *Proc. 23rd CIKM*, pages 1959–1962. ACM, 2014.

[83] R.I. Oliveira. Concentration of the adjacency matrix and of the laplacian in random graphs with independent edges. *arXiv preprint arXiv:0911.0600*, 2010.

[84] Weike Pan, Erheng Zhong, and Qiang Yang. Transfer learning for text mining. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 223–258. Springer, Heidelberg, DE, 2012.

[85] Shameem P. Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing F-Measures by cost-sensitive classification. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS 2014)*, pages 2123–2131, Montreal, USA, 2014.

[86] Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins*, 63:490–500, 2006.

[87] A. M. Rashid, S.K. Lam, G. Karypis, and J. Riedl. Clustknn: a highly scalable hybrid model-& memory-based cf algorithm. In *Proc. WebKDD-06, KDD Workshop on Web Mining and Web Usage Analysis*, 2006.

[88] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[89] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41, 2002.

[90] J.B. Schafer, J.A. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proc. EC*, pages 158–166, 1999.

[91] Y. Seldin, P. Auer, F. Laviolette, J. Shawe-Taylor, and R. Ortner. Pac-bayesian analysis of contextual bandits. In *NIPS*, pages 1683–1691, 2011.

[92] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic Convex Optimization. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009)*, Montreal, CA, 2009.

[93] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. PEGASOS: Primal Estimated sub-GrAdient SOlver for SVM. *Mathematical Programming, Series B*, 127(1):3–30, 2011.

[94] Aleksandrs Slivkins. Contextual bandits with similarity information. *JMLR*, 2014.

[95] I. Sutskever, R. Salakhutdinov, and J. Tenenbaum. Modelling relational data using bayesian clustered tensor factorization. In *NIPS*, pages 1821–1828. MIT Press, 2009.

[96] Balazs Szorenyi, Robert Busa-Fekete, Istvan Hegedus, Robert Ormandi, Mark Jelasity, and Balazs Kegl. Gossip-based distributed stochastic bandit algorithms. In *ICML*, pages 19–27, 2013.

[97] L. Tang, Y. Jiang, L. Li, and T. Li. Ensemble contextual bandits for personalized recommendation. In *Proc. RecSys*, 2014.

[98] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li. Personalized recommendation via parameter-free contextual bandits. In *Proc. SIGIR*. ACM, 2015.

[99] Cem Tekin and Mihaela van der Schaar. Distributed online learning via cooperative contextual bandits. *IEEE Trans. Signal Processing*, 2013.

[100] M. Thorup. Decremental dynamic connectivity. In *Proc. SODA*, pages 305–313, 1997.

[101] J. Tropp. Freedman's inequality for matrix martingales. *arXiv preprint arXiv:1101.3039v1*, 2011.

[102] M. Valko, R. Munos, B. Kveton, and T. Kocák. Spectral Bandits for Smooth Graph Functions. In *31th International Conference on Machine Learning*, 2014.

[103] K. Verstrepen and B. Goethals. Unifying nearest neighbors collaborative filtering. In *Proc. RecSys*, 2014.

[104] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, January 2007.

[105] Jack Chongjie Xue and Gary M. Weiss. Quantification and semi-supervised classification methods for handling changes in class distribution. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2009)*, pages 897–906, Paris, FR, 2009.

[106] Y. Yue, S. A. Hong, and C. Guestrin. Hierarchical exploration for accelerating contextual bandits. In *ICML*, 2012.

[107] Constantin Zalinescu. *Convex Analysis in General Vector Spaces*. River Edge, NJ: World Scientific Publishing, 2002.

[108] Zhihao Zhang and Jie Zhou. Transfer estimation of evolving class priors in data stream classification. *Pattern Recognition*, 43(9):3151–3161, 2010.

[109] Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *20th International Conference on Machine Learning (ICML)*, pages 928–936, 2003.