Crowd Learning: Improving Online Decision Making Using Crowdsourced Data

Yang Liu[#] **and Mingyan Liu**[$]
[#] yangl@seas.harvard.edu, Harvard University, Cambridge MA, USA
[$] mingyan@umich.edu, University of Michigan, Ann Arbor MI, USA

Abstract

We analyze an online learning problem that arises in crowdsourcing systems for users facing crowdsourced data: a user at each discrete time step t can choose K out of a total of N options (bandits), and receives randomly generated rewards dependent on user-specific and option-specific statistics unknown to the user. Each user aims to maximize her expected total rewards over a certain time horizon through a sequence of exploration and exploitation steps. Different from the typical regret/bandit learning setting, in this case a user may also exploit crowdsourced information to augment her learning process, i.e., other users' choices or rewards from using these options. We consider two scenarios, one in which only their choices are shared, and the other in which users share full information including their choices and subsequent rewards. In both cases we derive bounds on the *weak regret*, the difference between the user's expected total reward and the reward from a user-specific best single-action policy; and show how they improve over their individual-learning counterpart. We also evaluate the performance of our algorithms using simulated data as well as the real-world movie ratings dataset MovieLens.

1 Introduction

We analyze the following learning problem in the context of crowdsourcing, such as in a recommendation system: M users face N options, such as those in restaurants, movies, etc.; in a discrete time setting, at each step a user chooses K out of the N options, and receives randomly generated rewards, whose statistics depend on the options chosen as well as the user herself, but are unknown to the user. The objective of each user is to maximize her expected total reward (e.g., overall satisfaction of watching movies) over a certain time horizon through an online learning process, i.e., a sequence of exploration (sampling the return of each option) and exploitation (selecting empirically good options) steps. Taken separately, an individual user's learning process may be cast as a standard multi-armed bandit (MAB) problem which has been extensively studied, see e.g., [Anantharam *et al.*, 1986; Auer *et al.*, 2002; Lai and Robbins, 1985; Tekin and Liu, 2012]. Our interest, however, is on how an individual's learning process may be affected by "second-hand learning", i.e., by observing how others in the crowd act and what they recommend. The challenge is that what is considered desirable options for one may be undesirable for another (consider restaurant choices: one Yelp user may favor large establishments with extensive menus while another may favor small out-of-the-way places), and this difference in preference is in general unknown a priori. Moreover, even when two users happen to have the same preference (e.g., they agree one option is better than the other), they may differ in their absolute valuation of each individual option (again, two Yelp users may agree restaurant A is better than B, but one may rate them 5 and 4 stars respectively, while the other 4 and 3, respectively). Consequently if an individual wants to take others' actions into account in her own learning process, she would need to figure out whether their preferences are aligned and whether their valuations are on a similar scale. While it is generally believed such crowdsourced information is useful, this is often argued heuristically. A recent study [Goldstein *et al.*, 2014] shows that crowd wisdom should be selected carefully to reach its potential. This raises the interesting question on how to harness the power of crowdsourcing, and what type of learning algorithms can effectively utilize crowdsourced data in addition to one's direct observations. This is what we aim to address in this paper.

MAB has been employed to analyze crowdsourcing problems, with notable successes [Bresler *et al.*, 2015; Donmez *et al.*, 2009; Ertekin *et al.*, 2012; Ho *et al.*, 2013]. [Donmez *et al.*, 2009; Ertekin *et al.*, 2012; Ho *et al.*, 2013] focus on task assignment on a crowdsourcing market. Our model analyzes another major type of crowdsourcing platform, where users crowdsource their personal experiences such as in a recommendation system. Our motivation is similar to that of [Bresler *et al.*, 2015]; however, [Bresler *et al.*, 2015] studies an item-to-item based collaborative filtering method, while our model and results are user-centric. Another related subject is learning with side information or with metric or spectral structure among arms, see e.g., [chun Wang *et al.*, 2005; Lu *et al.*, 2010; Langford and Zhang, 2007; Gentile and Li, 2014; Valko *et al.*,]; this line of work however assumes additional statistical information relating side information to reward observation or continuity on arms' reward space, both of which are absent in our case.

Our question on how one user can learn via observing others data resembles certain similarity to social learning [Bandura and Walters, 1963; Smith and Sørensen, 2000]. Different from many existing works on agents' behavioral studies (e.g. herding), we concern how one should construct such a learning procedure efficiently when facing crowd information in a bandit setting. [Rosenberg *et al.*, 2007] also set out to study social learning in a bandit setting, where a one-arm bandit question is considered. Our work differs in that agents' observations are not necessarily drawn from the same unknown distribution – this does complicate the learning as users need to learn to distinguish other users that share similar preferences from those that are not. Also we consider both cases that users share partial or full information.

We will assume that users are heterogeneous in general, i.e., when choosing the same option they obtain rewards driven by different random processes. We consider two scenarios and design crowd-learning algorithms in each case. (1) In the first case users reveal limited information, only their choices but not the rewards obtained; (2) in the second case users crowdsource/exchange full information, not only their choices but also the reward outcomes of those choice. Performance is measured in *weak regret*, the difference between the user's total reward and that from a user-specific best single-action policy (i.e., always selecting the set of options generating the highest mean rewards for this user). We show when limited information is available, the improvement in learning is tied to how much we value others' opinion compared to our own. When complete information is shared, we can achieve an M-fold improvement in the regret bound under certain assumptions on the rewards. To our best knowledge this is the first attempt to analyze online learning with crowdsourced data. By applying our results to the movie ratings dataset MovieLens [KONECT, 2014], we show our algorithms can be used as an online (causal) process to make real-time predictions/recommendations. Due to space limitation, proofs can be found in the full version of our paper.

2 Problem formulation

Consider a network of M users indexed by the set $\mathcal{U} = \{1, 2, ..., M\}$ and a set of available options (also referred to as *arms* following the bandit problem literature), denoted by $\Omega = \{1, 2, ..., N\}$. The system works in discrete time indexed by $t = 1, 2, \cdots$. At each time step a user can choose up to $1 \leq K \leq N$ options. For user i an option k generates an IID reward over time denoted by random variables $\{X_k^i(t)\}$, with a mean reward given by $\mu_k^i := \mathbb{E}[X_k^i]$. We will assume that $\mu_l^i \neq \mu_k^i$ for $l \neq k, \forall i \in \mathcal{U}$, i.e., different options present distinct values to a user. Denote $\mathcal{X}_k^i$ as the support for X_k^i and we further assume finite support over X_k^i, i.e., there are finite positive constants $\bar{X}_k$ such that $X_k^i(\omega) < \bar{X}_k$, $\forall i, k, \omega$, where ω denotes an arbitrary realization. Notice for each option k, $\mathcal{X}_k^i$ could differ from each other. For exmple, one user never rates a restaurant higher than 4 stars, while another one may agree to give out a 5. All these knowledge on the support set remains unknown to users *a priori*. We will denote the set of top K options (in terms of mean rewards) for user i as N_K^i and its complement $\overline{N}_K^i$. Denote by $a^i(t)$ the set of choices made by user i at time t; the sequence $\{a^i(t)\}_{t=1,2,\cdots}$ constitutes user i's policy.

Following the classical regret learning framework, we will adopt the *weak regret* as a performance measure; this is the gap between the total reward (up to some time T) of a given learning algorithm and the total reward of the best single-action policy given a priori knowledge on the average statistics, which in our case is the sum reward generated by the top K options for a user. This is formally given as follows for user i adopting policy a^i:

$$R^{i,a}(T) = T \cdot \sum_{k \in N_K^i} \mu_k^i - \mathbb{E}[\sum_{t=1}^{T} \sum_{k \in a^i(t)} X_k^i]. \qquad (1)$$

The goal of an online learning algorithm, for each user, is to minimize the above regret measure, whose time-average should ideally diminish, i.e., it is desirable to have $R^{i,a}(T) = o(T), \forall i$. In this study we investigate whether the regret performance can be improved by allowing a user i_1 access to observations (or decisions) made by another user i_2, i.e., by letting a user crowdsource data generated by other distinct users.

We will consider two types of information disclosed by the users. Under the first type, users disclose *partial information*, sharing only their decisions, i.e., the set of choices they make, at the beginning of each time step, while withholding the actual observation/reward information following the decisions. This models many real crowdsourcing systems. For example, users check in at a restaurant without leaving a review. In the second case the users announce observations following their decisions, i.e., the actual rewards received from those options, such as perceived quality of a movie. Such announcements may be made at the end of each time step, or may be made periodically at a lesser frequency.

For the partial information scenario we model explicitly the fact that users have different preference orderings over the N options, by assuming that the M users may be classified into G distinct groups, indexed by the set $\mathcal{G} = \{1, 2, ..., G\}$, with users of the same group (say group l) having a unique K-preferred set N_K^l; these preference sets (but not the group membership) are assumed to be public knowledge[1]. Note that even with the same preference set, users may be further distinguished based on

[1] This is not a restrictive assumption as we can simply choose $\mathcal{G}$ to be the complete K-set with $G = \binom{N}{K}$.

the actual ordering of these top K options. Our model essentially bundles these users into the same group, provided their top K choices are the same. This is because as a user is allowed K choices at a time, further distinguishing their preferences within these K options will not add to the performance of an algorithm. In the full information case we do not further differentiate users' preference ordering over the N options as we have access to their direct observations.

3 Partial Information (CL-PART)

In the partial information case, the main idea behind our algorithm is to use the shared decision to first obtain any user's frequency of selecting each option. These statistics are indicators of a user's preference, assuming the user is trying to maximize its own reward over time; thus decisions of another reveal preferences and may be exploited. In what follows we first propose a group classification procedure with performance guarantee to help a user distinguish similar users and then show how we can design algorithms using this information.

To differentiate users' preferences, consider the following sample frequency based group identification procedure. Each user keeps the same set of statistics $n^i_k(t)$ as before: the number of times user i is seen using option k. Users then estimate each other's preference by ordering the statistics: at time t user i's preference is estimated to be the set $\tilde{N}^i_K(t)$, which contains options k whose frequency $n^i_k(t)$ is among the K highest of all i's frequencies. User i is then put in a preference group with whose (known) preferred set $N^l_K(t)$ is the closest in distance: assign user i to group $g^i(t)$ if

$$g^i(t) = \text{argmax}_{l\in\mathcal{G}}\, D^{i,l}(t) := \text{argmax}_{l\in\mathcal{G}}\, |\tilde{N}^i_K(t) \cap N^l_K|\ ,$$

with ties broken randomly. The performance of the above classification step will be analyzed later within the context of our learning algorithm.

3.1 Algorithm and performance

After differentiation, denote by $\mathcal{U}_i$ the set of similar users for each user i and denote $M_i = |\mathcal{U}_i|$. There are two options a user can choose to implement an algorithm: (i) to use information from only those identified as having the same preferences, i.e., users in set $\mathcal{U}_i$, and (ii) to use all users' information. We start with the easier case (i). Denote by $n_{k,i}(t)$ the total number of times option k has been selected by the crowd $\mathcal{U}_i$ up to time t, i.e., $n_{k,i}(t) := \sum_{i\in\mathcal{U}_i} n^i_k(t)$. Then define $\beta^i_k(t) := n_{k,i}(t)/\sum_{l\in\Omega} n_{l,i}(t)$ to denote the frequency at which option k is used by the group up to time t. This will be referred to as the group recommendation. Then we rewrite the frequency parameter to take advantage of crowd learning in the following way. Based on $\beta^i_k(t)$ we first order the options in descending order; their rank denoted by $\mathbf{rank}_k$. Denote by $\overline{B}_K(t)$ the non-top K options based on the frequency estimate: $\overline{B}_K(t) := \{k \in \Omega : \mathbf{rank}_k > K\}$. Define user specific frequency estimate for $j \in \mathcal{U}^i$ as $\beta^i_k(j;t) = \max\{1 - \frac{\log t}{\log T}, n^j_k(t)/t\}$. The first constraint $1-\frac{\log t}{\log T}$ is to regulate the change in β over time. Clearly $\beta^i_k(j;t) \le 1, \forall t$. With this now we redefine the frequency parameters as follows: $\tilde{\beta}^i_k(t) := \min_{j\in\mathcal{U}_i} \beta^i_k(j;t)$. With these definitions, we construct the following algorithm CL-PART(I), by biasing towards potentially good options as indicated by the group. Under the CL-PART(I) algorithm, option k is selected at time t if its index value defined below is among the K highest:

$$\text{CL-PART(I)}: r^i_k(t) - \alpha(t)(1-\tilde{\beta}^i_k(t))\sqrt{\frac{2\log t}{n^i_k(t)}} + \sqrt{\frac{2\log t}{n^i_k(t)}},$$

where $\alpha(t) := (1-\gamma) - \epsilon(t)^2$, where $\epsilon(t) := \sqrt{\frac{C_{\text{explore}}\log t}{\min_k(\Delta^i_k)^2 t}}$, with $C_{\text{explore}} > 0$ being a constant. Δ^i_k is defined as $\Delta^i_k := \mu^i_K - \mu^i_k$. $\alpha(t) \in [0,1)$ is a weighting factor over the group recommendation capturing how much user i is valuing recommendation from the group. The convergence term implies with more samples, we are more confident with the recommendation factor. $0 < \gamma < 1$ is a constant that can be arbitrarily small. For technical reason we need $\alpha(t) < 1, \forall t$. γ helps us achieve this. Define $z^* := 1 - \sqrt{3/4}$, we have the following result on the regret performance of CL-PART(I).

Theorem 1 *Under CL-PART(I), we have user i's weak regret is upper bounded by* $R^i_{\textit{CL-PART(I)}}(t) \le \sum_{k\in\overline{N}^i_K} \lceil \frac{8}{\Delta^i_k}(z^* + [\gamma + \epsilon(t) + \frac{\alpha(t)}{M_i}]^2)\log t\rceil + \textit{const.}$

The original UCB1 algorithm has a weak regret upper bounded by ([Auer *et al.*, 2002]) $R^i_{\text{UCB1}}(t) \le \sum_{k\in\overline{N}^i_K} \lceil 8\log t/\Delta^i_k\rceil$ + const. To compare we see when t, M_i are large, $z^* + [\gamma + \epsilon(t) + \frac{\alpha(t)}{M_i}]^2 \ll 1$, and thus a much better performance can be expected.

3.2 Leveraging more information

We next consider leveraging more user information beyond the set $\mathcal{U}_i$. Restricting our attention to $\mathcal{U}_i$ previously ensures that users have the same top-K choices. Removing this restriction means that we may have $N^i_K \neq N^j_K$; however, it is possible there exists option $k \in \overline{N}^i_K \cap \overline{N}^j_K$, that is two users may not agree on all option, but on some of them. In this sense, j's sample frequency estimation of k can again be utilized by i.

We re-use the index construction as detailed in CL-PART(I), with the following difference: we discount choices made by users believed to belong to a different group, so as not to be overly influenced by users with different preferences. Specifically, user i assigns the following weight to option k: $\beta^i_k(t) = \min_{j\in\mathcal{U}_i}(\beta^i_k(j;t))^{\omega^{i,j}}$, where weights $\omega^{i,j} = 1$ if i estimates user j to be in the same group as itself, and $\omega^{i,j} < 1$ otherwise; $\omega^{i,j}$ can also be chosen as a function of the difference between different preference groups. This modified index leads

[2] We slightly abuse the notation here as $\alpha(t)$ differs from user to user by a little.

to algorithm CL-PART(II). We then define the following set for each $k \in \overline{N}^i_K$: $\mathcal{U}^i_k = \{j : j \in\in \mathcal{U}, k \in \overline{N}^j_K\}$. And denote $M^i_k = |\mathcal{U}^i_k|$. That is, M^i_k records the number of users who agree k is out of their top choices. Notice $\mathcal{U}^i \subseteq \mathcal{U}^i_k$ and thus $M^i_k \geq M_i$. We can then similarly prove the following result.

Theorem 2 *Under CL-PART(II), at each time t, user i's weak regret is upper bounded by* $R^i_{\textit{CL-PART(II)}}(t) \leq \sum_{k \in \overline{N}^i_K} \lceil \frac{8}{\Delta^i_k}(z^* + [\gamma + \epsilon(t) + \frac{\alpha(t)}{M^i_k}]^2) \log t \rceil + \textit{const.}$

Since $M^i_k \geq M_i$ we have achieved a strictly better bound compared to the one in Theorem 1.

4 Full Information (CL-FULL)

In this case users disclose the reward $X^i_k(t)$ for each option they chose at the end of a time step. The technical challenge here is that the statistics driving the rewards generation is not identical for all users even when using the same option. We note that a user's (say user i) own observed reward of any option k can be generally related to another's (user j) disclosed reward by the following: $X^i_k \overset{d}{=} \mathcal{T}^{i,j}_k(X^j_k)$, where $\mathcal{T}^{i,j}_k$ could be an arbitrary function. In this definition, $\mathcal{T}^{i,j}_k$ captures the potential difference in users i, j's belief over option k, as well as possible discrepancies intentionally introduced by mis-reporting from malicious users (either i or j). For example if a user i learns that whenever she likes an item k and gives it rating higher than 4 (on the scale of {1,2,3,4,5}), a malicious user j always rates the item a 2 or below either or mislead or out of belief, user i can safely convert it as if she observed it directly by choosing $\mathcal{T}^{i,j}_k$ such that $\mathcal{T}^{i,j}_k : \{1, 2\} \rightarrow \{4, 5\}$. It follows that if we are able to estimate $\mathcal{T}^{i,j}_k$, then one user's sample can be safely converted to another. A natural way for estimating such a function is through the Taylor expansion : $X^i_k \approx \sum_{n=0}^{D} b^{i,j}_{k,n} \cdot (X^j_k)^n$, where D is the degree of estimation. When $D = \infty$ the equality holds in distribution. As long as we can estimate the $b_{k,n}$s, we will similarly be able to convert a sample from one user to another. However, there is generally no closed form characterization of the solutions to $b_{k,n}$s. We thus limit our attention to the following simplified and tractable model.

Log-Linear model Consider the case where the rewards received/disclosed by two users from the same option are given by a log-linear relationship: $X^i_k \overset{d}{=} (X^j_k)^{\delta^{i,j}_k}$, $\forall i \neq j \in \mathcal{U}, k \in \Omega$,where $\delta^{i,j}_k$ is a constant[3] and unknown scaling factor also referred to as the *distortion* or *distortion factor* between two users. This relationship implies that one user's perception of a given option is statistically identical to another's to a constant power.

This simplification gives us significant computational advantage – as long as we can learn a single parameter between any pair of users over each option, we will be able to map samples from one to another. While certainly a simplification, this concrete model leads to closed-form characterizations of the regret bounds, which sheds light on the effect of various problem parameters, and it proves to work well with the real dataset MovieLens. It's also worth noting that this log-linear relationship applies to exponential family distributions such as Log-normal, Pareto, and Weibull, when used to model user observations. Also it is worth noting that the above model is chosen to solve more challenging scenarios when $\mathcal{X}^i_k$ has a large state space or even infinite space (continuous random variable).

4.1 Crowd-Learning with Full Information

For simplicity of presentation, we normalized all $X^i_k, \forall i, k$ onto the support $(0, 1]$, i.e., $\mathcal{X}^i_k \subseteq (0, 1]$. Consider two users i and j, and option k. Up to time t, all quantities $\{X^i_{a^i(t)}(t)\}_t$ are not only available to user i, but also to all other users $j \in \mathcal{U}\backslash\{i\}$ due to the full information disclosure, and vice versa. Denote by $\hat{r}^i_k(t)$ the sample mean of log-reward $\{\log X^i_k(t)\}_t$ collected by i from option k. Considering continuous support for each observation, since $\log X^i_k(t) = \log X^j_k(t) \cdot \delta^{i,j}_k$, user i then estimates the distortion between herself and user j by calculating the following $\tilde{\delta}^{i,j}_k(t) = \hat{r}^i_k(t)/\hat{r}^j_k(t)$,$\forall i \neq j \in \mathcal{U}, k \in \Omega$. With the above quantity we then make the following simple modification to the well-known UCB1 algorithm [Auer *et al.*, 2002]. In the original UCB1 (or rather, a trivial multiple-play extension of it), user i's decision $a^i(t)$ at time t is entirely based on her own observations. Specifically, denote by $n^i_k(t)$ the number of times user i has selected option k up to time t. The original UCB1 then selects option k at time t, if its index value given below is among the K highest: $r^i_k(t) + \sqrt{2 \log t/n^i_k(t)}$. Our modified algorithm takes this index as a baseline and makes the following changes: option k is selected at time t if its index is among the K highest CL-FULL index:

$$\frac{r^i_k(t) \cdot n^i_k(t) + \sum_{j \neq i} \Lambda^{i,j}(r^j_k(t)) \cdot n^j_k(t)}{\sum_{j \in \mathcal{U}} n^j_k(t)} + \sqrt{\frac{2 \log t}{\sum_{j \in \mathcal{U}} n^j_k(t)}},$$

where the operator $\Lambda^{i,j}(r^j_k(t))$ is defined as: $\Lambda^{i,j}(r^j_k(t)) = \sum_{s=1}^{t}(X^j_k(s))^{\tilde{\delta}^{i,j}_k(t)} \cdot 1\{k \in a^j(s)\}/n^j_k(t)$.

4.2 Performance analysis of CL-FULL

By default $\delta^{i,i}_k = 1$, and we denote $\delta^{i,*}_k := \max_{j \in \mathcal{U}} \delta^{i,j}_k$, $\delta^{i,*} := \max_{k \in \Omega} \delta^{i,*}_k$, $\bar{X}_k = \max_{i,\omega} X^i_k(\omega)$. The following series of results characterize the performance of CL-FULL. They are organized based on the nature of $\delta^{i,j}_k$. We shall start with the simplest case when $\delta^{i,j}_k$ is option independent: $\delta^{i,j}_1 = \delta^{i,j}_2 = ... = \delta^{i,j}_K$, followed by k-dependent $\delta^{i,j}_k$s.

Option independent $\delta^{i,j}$ In this case, for user i to estimate $\delta^{i,j}$, she can simply choose the option that has the largest number of collected samples (by users i and

[3] This assumption can be relaxed to the case that $\delta^{i,j}_k$ is generated with certain randomness (e.g., noise) , instead of being fixed as a constant.

j) for the purpose of calculation. We assume in this section for each pair of (i, j) we have $N_K^i \cap N_K^j \neq \emptyset$. This mild assumption is to ensure for any pair of users (i, j) we can find a common good option. This assumption can be removed by ignoring the users $j \neq i$ for each user i such that $N_K^i \cap N_K^j = \emptyset$. This can be done via frequency counting and matching (For details please refer to Section 3). Practically if a user found another sharing no option in her top K preferred set, she may have a strong reason to believe this user's data may not be useful for her and will then discard. We have the following theorem characterizing the performance of CL-FULL.

Theorem 3 *Under CL-FULL, $\exists C_1 > 0$ such that user i's weak regret is upper bounded by: $R^i_{CL\text{-}FULL}(t) \leq \sum_{k \in \overline{N}^i_K} \lceil \frac{8\Delta^i_k}{M \cdot [\Delta^i_k - 2\bar{X}^2_k \epsilon_k(t)]^2} \log t \rceil + const.$ where $\epsilon_k(t) = 2\delta^{i,*}_k / (c_k - 2\sqrt{\frac{\log t}{C_1 t}}) \cdot \sqrt{\log t / C_1 t}$, and $0 < c_k \leq 1$ is an option dependent constant.*

Compare with the bound in original UCB1, we note that in our result, the term $2\delta^{i,*}_k / (c_k - 2\sqrt{\log t / C_1 t}) \cdot \sqrt{\log t / C_1 t} \rightarrow 0$ at approximately the rate of $O(\sqrt{\log t / t})$; if we ignore this term the constant in front of $\log t$ becomes $8 / M \cdot \Delta^i_k$ which shows a M-fold performance improvement.

4.3 Option dependent $\delta^{i,j}_k$

With option dependent $\delta^{i,j}_k$, we need to bound the error ϵ in estimating $\delta^{i,j}_k$ for each option k. We have the following result.

Theorem 4 *Under CL-FULL, user i's weak regret is upper bounded by, $R^i_{CL\text{-}FULL}(t) \leq \sum_{k \in \overline{N}^i_K} \lceil \frac{8}{\Delta^i_k} (M^{-1/2} + \frac{2\sqrt{2}\bar{X}^2_k \delta^{i,*}_k}{c_k})^2 \log t \rceil + const.$*

A joint estimation of $\delta^{i,j}_k$ Note the extra learning error in the bound of Theorem 4 is independent of M, which does not decrease when the number of users increases; this is a potentially worrisome bottleneck. Below we examine ways to mitigate this. We start by explaining why we do not have an M-fold scaling in this additional error term. Statistically speaking, since we use the converted samples to estimate the sample mean of each option, there are M-fold number of samples (though noisy) compared to individual learning. However, this is not true when learning the δs, as we need to use a user's own data to make such pairwise calculations (one-fold for each pair). This motivates us to consider using all samples to estimate δ simultaneously to achieve an M-fold speed-up. Specifically, instead of pair-wise estimation, $\delta^{i,j}_k$ can be estimated as follows:

$$\tilde{\delta}^{i,j}_k = (\hat{r}^i_k(t) + \sum_{l \neq i} \hat{r}^l_k(t) \tilde{\delta}^{i,l}_k) / (\hat{r}^j_k(t) + \sum_{l \neq j} \hat{r}^l_k(t) \tilde{\delta}^{j,l}_k) \ .$$

which is a quadratic equation of $\delta^{i,j}_k$s. Denote $\Phi_k := [\tilde{\delta}^{i,j}_k]_{i,j}$, we have the following Quadratic Matrix Equation (QME) whose solution leads to solutions for δ:

$$\sum_{i=1}^{M} \sum_{j=1}^{M} E_{j,i} (\Phi_k)^T E_{j,i} \Phi_k \tilde{A}_k E_{i,j} - \Phi_k \tilde{A}_k = 0 \ , \qquad (2)$$

where $E_{j,i}(q, l) = 0$ when $(q, l) \neq (j, i)$, and $E_{j,i}(j, i) = 1$, and some $\tilde{A}_k$. When the solution for such a QME satisfies certain perturbation bounds, we prove:

Theorem 5 *Under CL-FULL, $\exists$ a constant $C_3 > 0$ s.t. user i's weak regret is upper bounded by $R^i_{CL\text{-}FULL}(t) \leq \sum_{k \in \overline{N}^i_K} \lceil \frac{8}{\Delta^i_k} (1 + 2\sqrt{2}\bar{X}^2_k \delta^{i,*}_k C_3)^2 / M \log t \rceil + const.$*

5 Numerical Experiment

In our simulation we have ten users with five options; each user targets the top three options at each time, i.e., $M = 10, K = 3, N = 5$. Furthermore, for each option the reward is given by a truncated exponentially distributed random variable (bounded). The distortion factor between each pair of users for each option is generated according to a Gaussian random variable with mean 1 and variance 1. We use "crowd regret" to denote the sum of regrets from all users. The regret results are averaged over 50 sample realizations.

The performance of CL-PART is compared to UCB-IND under different parameter γ in Figure 1. We see that CL-PART consistently outperforms UCB-IND. A smaller γ gives better performance, which confirms our analytical results. Also, as we noted in our analysis and see here, a smaller K results in less regret. Also we compare CL-PART(I) and CL-PART(II). From results in Figure 1:Right we observe that with more appropriately designed algorithm and leveraging more data, a (much) better performance can be achieved.

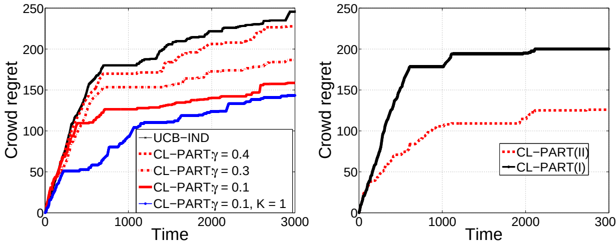

Figure 1: Performance of CL-PART. Left: with different γ parameters. Right: CL-PART(I) v.s. CL-PART(II)

We do performance evaluation for CL-FULL. From Figure 2 we see with full information exchange the crowd learning algorithm significantly outperforms individual learning. Moreover, its performance is comparable to a centralized scheme (denoted as UCB Centralized in the figure), whereby the M users are centrally controlled and coordinated in their learning using UCB1, allowing simultaneous selection of the same options by multiple users, and each receiving MK samples at each time step.

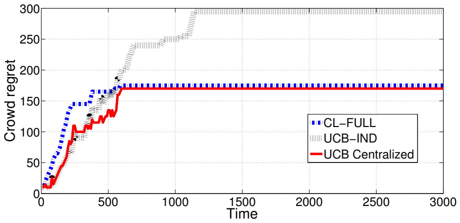

Figure 2: CL-FULL v.s. UCB-IND v.s. UCB Central.

6 An Empirical Study Using MovieLens

We now apply the idea of crowd-learning to the MovieLens data [KONECT, 2014] collected via a movie recommendation system. We will use MovieLens-1M dataset. Obviously the decision aspect of the learning algorithm cannot be verified using this dataset as we have no information on the users' actual decision process. Thus our goal is to use the MovieLens data to verify whether our crowd-learning algorithm can help us predict how a user is going to rate movies (indexes in our algorithm) in the future given this and other users' reviews in the past.

6.1 Experiment design

We explain the experiment in the context of this dataset. We first note that in this case time is no longer discrete, as reviews arrive as an arbitrary arrival process in continuous time. Therefore the discrete time steps used in the algorithm is replaced by a clock driven by this arrival process, i.e., one tick for each arrival.

Secondly, the reviews tend to arrive in clusters upon new movie releases, and it may be hard to find any review for a movie that was released some time ago. This means that if we are to treat each movie as a separate option (or arm) then these options are not simultaneously available at all times as movies come and go, and along with them their corresponding reviews. We thus bundle these movies into categories so that over a long period of time there are always reviews available for a genre. Adopting the classification given in [KONECT, 2014], we will bundle movies by their genres, e.g., Action, Adventure, Comedy, etc., resulting in 18 categories/genres; each is regarded as an option/arm for our algorithm.

6.2 Online prediction result

The prediction performance is measured by the average error $\mathcal{E}_A(T)$ and squared error $\mathcal{E}_S(T)$ (of our predicted rating), both averaged over the number of samples in the data. We start by plotting $\mathcal{E}_A(T)$ as a function of T; this is shown in Figure 4 where we have used $K = 3$, i.e., a user's top 3 preferences (unordered) determine her similar group. The data suggests that a user rarely reviews more than 6 different categories of movies, thus the choice of $K = 3$ is a reasonable trade-off between too restrictive a definition of similarity group (a large K and a very small similarity group) and an overly liberal one (a small K and a very large similarity group). As expected we see a downward trend as the prediction becomes more accurate with more past samples and crowd learning clearly outperforms individual learning. This trend is not exactly monotonic primarily because the arrivals of new movies which can give rise to increased error within the corresponding category.

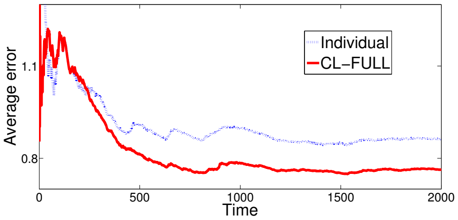

Figure 3: Convergence of regret in time.

Cate. k	1	2	3	4	5	6
Indv.	0.4853	0.6343	0.6620	0.6868	0.7134	0.7278
FULL ($K = 3$)	0.4618	0.5173	0.5356	0.6371	0.6551	0.6529
FULL ($K = 6$)	0.4826	0.6249	0.6543	0.6776	0.7084	0.7263
Indv.	0.5376	0.7367	0.8251	0.6551	0.7503	0.7574
FULL ($K = 3$)	0.6002	0.7195	0.7997	0.7933	0.8648	0.8389
FULL ($K = 6$)	0.7071	0.8438	0.8773	0.9276	0.9685	0.9997

Table 1: $\mathcal{E}_A(T)$ (top 3 rows) and $\mathcal{E}_S(T)$ (bottom 3)

Furthermore, to see more clearly the effect of the parameter choice K, we compare the error performance by categories. Specifically, we measure the prediction error of each user for movies in her most preferred category and average this over all users in Tables 2. For each user the preference ordering of categories is determined using her average rating for each category over the entire data trace. Due to our choice of $K = 3$, we see a clear degradation in the average error performance under the crowd learning algorithm when we go from $k = 3$ to $k = 4$, although the latter still outperforms individual learning. This is to be expected because in making prediction for one's top 3 categories we have the advantage of using the similarity group for help; this may not apply to the next 3 categories as members of the similarity group may not share the same preference over the these 3. We also added the performance when choosing $K = 6$: the prediction gap between the top 3 and latter categories goes away.

7 Conclusion

In this paper we considered a crowd-learning problem in the context of online crowdsourcing systems, and analyzed two cases, where users share partial or full information respectively. We constructed UCB1-like index algorithms and derived bounds on their weak regret. These bounds reveal interesting insights that are helpful for algorithm/policy design, and generally see a multi-fold improvement due to crowdsourcing and learning. Numerical experiments include both synthetic data and the MovieLens 1M dataset. In the latter case we show that our algorithm can achieve an improvement in movie recommendation quality in an online manner.

References

[Agrawal, 1995] Rajeev Agrawal. The Continuum-Armed Bandit Problem. *SIAM journal on control and optimization*, 33(6):1926–1951, 1995.

[Anantharam *et al.*, 1986] Venkat Anantharam, Pravin Varaiya, and Jean Walrand. Asymptotically Efficient Allocation Rules for the Multiarmed Bandit Problem with Multiple Plays Part I: I.I.D. Rewards, Part I-I: Markovian Rewards. Technical Report UCB/ERL M86/62, EECS Department, University of California, Berkeley, 1986.

[Audibert *et al.*, 2009] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[Auer *et al.*, 2002] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.*, 47:235–256, May 2002.

[Bandura and Walters, 1963] Albert Bandura and Richard H Walters. *Social learning and personality development*, volume 14. JSTOR, 1963.

[Bresler *et al.*, 2015] Guy Bresler, Devavrat Shah, and Luis F Voloch. Collaborative filtering with low regret. *arXiv preprint arXiv:1507.05371*, 2015.

[Candès and Recht, 2009] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[chun Wang *et al.*, 2005] Chih chun Wang, Student Member, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50:338–355, 2005.

[Donmez *et al.*, 2009] Pinar Donmez, Jaime G Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268. ACM, 2009.

[Ertekin *et al.*, 2012] Seyda Ertekin, Haym Hirsh, and Cynthia Rudin. Learning to predict the wisdom of crowds. *arXiv preprint arXiv:1204.3611*, 2012.

[Gentile and Li, 2014] Claudio Gentile and Shuai Li. Online clustering of bandits. In *ICML*, 2014.

[Goldstein *et al.*, 2014] Daniel G Goldstein, Randolph Preston McAfee, and Siddharth Suri. The wisdom of smaller, smarter crowds. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 471–488. ACM, 2014.

[Ho *et al.*, 2013] Chien-Ju Ho, Shahin Jabbari, and Jennifer W Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 534–542, 2013.

[KONECT, 2014] KONECT. Movielens 1m network dataset – KONECT, August 2014.

[Konstantinov *et al.*, 2003] M. Konstantinov, D.W. Gu, V. Mehrmann, and P. Petkov. *Perturbation Theory for Matrix Equations*. Studies in Computational Mathematics. Elsevier Science, 2003.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[Lai and Robbins, 1985] T. L. Lai and H. Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[Langford and Zhang, 2007] John Langford and Tong Zhang. The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information. In *NIPS*, 2007.

[Liu and Aberer, 2013] Xin Liu and Karl Aberer. Soco: A social network aided context-aware recommender system. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 781–802, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[Lu *et al.*, 2010] Tyler Lu, Dvid Pl, and Martin Pal. Contextual multi-armed bandits. *Journal of Machine Learning Research*, 9:485–492, 2010.

[Rosenberg *et al.*, 2007] Dinah Rosenberg, Eilon Solan, and Nicolas Vieille. Social learning in one-arm bandit problems. *Econometrica*, 75(6):1591–1611, 2007.

[Smith and Sørensen, 2000] Lones Smith and Peter Sørensen. Pathological outcomes of observational learning. *Econometrica*, 68(2):371–398, 2000.

[Tekin and Liu, 2012] Cem Tekin and Mingyan Liu. Online Learning of Rested and Restless Bandits. *Information Theory, IEEE Transactions on*, 58(8):5588–5611, 2012.

[Valko *et al.*,] Michal Valko, Rémi Munos, Branislav Kveton, and Tomas Kocak. Spectral bandits for smooth graph functions.

[Zhong *et al.*, 2012] ErHeng Zhong, Wei Fan, and Qiang Yang. Contextual collaborative filtering via hierarchical matrix factorization. In *SDM'12*, pages 744–755, 2012.