Competitive Collaborative Learning

Baruch Awerbuch[1,*] and Robert D. Kleinberg[2,**]

[1] Department of Computer Science, Johns Hopkins University, Baltimore MD 21218, USA
baruch@cs.jhu.edu

[2] Department of Mathematics, Massachusetts Institute of Technology, Cambridge MA 02139, USA
rdk@math.mit.edu

Abstract. We develop algorithms for a community of users to make decisions about selecting products or resources, in a model characterized by two key features:

– The quality of the products or resources may vary over time.
– Some of the users in the system may be dishonest, manipulating their actions in a Byzantine manner to achieve other goals.

We formulate such learning tasks as an algorithmic problem based on the multi-armed bandit problem, but with a set of users (as opposed to a single user), of whom a constant fraction are honest and are partitioned into coalitions such that the users in a coalition perceive the same expected quality if they sample the same resource at the same time. Our main result exhibits an algorithm for this problem which converges in polylogarithmic time to a state in which the average regret (per honest user) is an arbitrarily small constant.

1 Introduction

Only a fool learns from his own mistakes. The wise man learns from the mistakes of others.

— Otto von Bismarck

It is clear that leveraging trust or shared taste enables a community of users to be more productive, as it allows them to repeat each other's good decisions while avoiding unnecessary repetition of mistakes. Systems based on this paradigm are becoming increasingly prevalent in computer networks and the applications they support. Examples include reputation systems in e-commerce (e.g. eBay, where buyers and sellers rank each other), collaborative filtering (e.g. Amazon's recommendation system, where customers recommend books to other customers), and link analysis techniques in web search (e.g., Google's PageRank, based on

* Supported by NSF grants ANIR-0240551 and CCR-0311795.
** Supported by a Fannie and John Hertz Foundation Fellowship.

combining links — i.e. recommendations — of different web sites). Not surprisingly, many algorithms and heuristics for such systems have been proposed and studied experimentally or phenomenologically [5, 11, 12, 13, 15, 16, 17]. Yet well-known algorithms (e.g. eBay's reputation system, the Eigentrust algorithm [10], the PageRank [5, 13] and HITS [11] algorithms for web search) have thus far not been placed on an adequate theoretical foundation.

Our goal in this paper is to provide a theoretical framework for understanding the capabilities and limitations of such systems as a model of distributed computation. We propose a new paradigm for addressing these issues, which is inspired by online learning theory, specifically the multi-armed bandit problem [1]. Our approach aims to highlight the following challenges which confront the users of collaborative decision-making systems such as those cited above.

Malicious users. Since the Internet is open for anybody to join, the above systems are vulnerable to fraudulent manipulation by dishonest ("Byzantine") participants.

Distinguishing tastes. Agents' tastes may differ, so that the advice of one honest agent may not be helpful to another.

Temporal fluctuation. The quality of resources varies of time, so past experience is not necessarily predictive of future performance.

While our learning theory paradigm is different from prior approaches, the resulting algorithms exhibit a resemblance to algorithms previously proposed in the systems and information retrieval literature [5, 10, 11, 13] indicating that our approach may be providing a theoretical framework which sheds light on the efficacy of such algorithms in practice while suggesting potential enhancements to these algorithms.

1.1 Our Approach

The problem we will consider is a generalization of the multi-armed bandit problem studied in [1]. In that problem there is a single learner and a set Y of m *resources*. In each of T consecutive trials, the learner chooses one of the resources while the adversary chooses a cost (taking values in $[0, 1]$) for each resource; after the trial, the cost of the resource chosen by the learner is revealed, and this cost is charged to the learner. We generalize this by considering a set X of n *agents*, some of which (possibly, a majority) may be dishonest. In each trial, each of the n agents chooses a resource, and the adversary chooses a cost for each resource. Each agent then learns the cost of the resource it selected, and this cost is charged to the agent. We assume that the honest agents belong to k *coalitions*, such that agents in the same coalition who choose the same resource at the same time will perceive the same expected cost. All agents may communicate with each other between trials, to exchange information (or possibly disinformation, in the case of dishonest agents) about the costs of resources they have sampled. However, agents are unaware of which coalitions exist and which ones they belong to.

If an agent chooses to ignore the feedback from other agents, and simply runs the multi-armed bandit algorithm by itself, then the classical analysis of the multi-armed bandit algorithm [1] ensures that for any constant $\delta > 0$, if $T = \Omega(m \log m)$, then the expected average cost of the resources chosen by that agent will exceed the average cost of the best resource in hindsight by no more than δ. However, it is possible that the honest agents may require much fewer than $\Omega(m \log m)$ trials to achieve this goal, if they can find a way to pool their information without being fooled by the bad advice from dishonest agents and agents from other coalitions. Here, we show that this is in fact possible, by presenting an algorithm whose convergence time is polynomial in $k \log(n)$, assuming that a constant fraction of the agents are honest and that $m = O(n)$.

Briefly, our algorithm works by having each agent select a resource in each trial by taking a random walk on a "reputation network" whose vertex set is the set of all agents and resources. Resources are absorbing states of this random walk, while the transition probabilities at an agent x may be interpreted as the probability that x would select a given resource y, or would ask a given other agent x' for advice. When an agent learns the cost of the resource chosen in a given trial, it uses this feedback to update its transition probabilities according to the multi-armed bandit algorithm. In this way, agents will tend to raise the probability of asking for advice from other agents who have given good advice in the past. In particular, though the initial transition probabilities do not reflect the partition of the honest agents into coalitions, over time the honest agents will tend to place greater weight on edges leading to other agents in the same coalition, since the advice they receive from such agents is generally better, on average, than the advice they receive from agents in other coalitions.

1.2 Comparison with Existing Work

Above, we cited the adversarial multi-armed bandit problem [1] which forms the basis for our work, and we have indicated the ways in which our model generalizes the existing multi-armed bandit model to the setting of collaborative learning with dishonest users. Our work is also related to several other topics which we now discuss.

Collaborative filtering — spectral methods: Our problem is similar, at least in terms of motivation, to the problem of designing collaborative filtering or recommendation systems. In such problems, one has a community of users selecting products and giving feedback on their evaluations of these products. The goal is to use this feedback to make recommendations to users, guiding them to subsequently select products which they are likely to evaluate positively. Theoretical work on collaborative filtering has mostly dealt with centralized algorithms for such problems. Typically, theoretical solutions have been considered for specific (e.g., stochastic) input models [7, 8, 9, 14, 4], In such work, the goal is typically to reconstruct the full matrix of user preferences based on small set of potentially noisy samples. This is often achieved using spectral methods. In constrast, we consider a general, i.e. adversarial, input model. Matrix reconstruction techniques do not suffice in our model. Firstly, they are vulnerable to manipulation by dishonest users, as was observed in [3] and [2]. Dishonest users, who may be in the overwhelming majority, may certainly disrupt the low rank assumption which is crucial in matrix reconstruction approaches. Alternatively, they may report phony data so as to perturb the singular vectors of the matrix, directing all the agents to a particularly bad action, e.g. an unscrupulous seller.

Collaborative filtering — random sampling methods: The only known collaborative filtering algorithm which tolerates Byzantine behavior is the "Random Advice Random Sample" algorithm in [2, 3]; it achieves a logarithmic learning time. The model in [2] deals with the static case, in which bad resources are consistently bad and good resources are consistently good; the only changes in the operating environment over time occur when resources arrive or depart. The algorithm in [2] uses the notion of "recommendation": once an agent finds a good resource, it sticks to it forever and recommends it to others. As the time elapses, progressively more agents 'stick" with the good advice. The bounds on regret and convergence time in [2] are analogous to ours, and are in fact poly-logarithmically superior, to those in our Theorem 1. However, [2] does not handle costs which evolve dynamically as a function of time, and is limited to $\{0, 1\}$-valued rather than real-valued costs.

Reputation management in P2P networks: Kamvar et al [10] proposed an algorithm, dubbed *EigenTrust*, for the problem of locating resources in peer-to-peer networks. In this problem, users of a peer-to-peer network wish to select other peers from whom to download files, with the aim of minimizing the number of downloads of inauthentic files by honest users; the problem is made difficult by the presence of malicious peers who may attempt to undermine the algorithm. Like our algorithm, EigenTrust defines reputation scores using a random walk on the set of agents, with time-varying transition probabilities which are updated according to the agents' observations. Unlike our algorithm, they use a different rule for updating the transition probabilities, and they demonstrate the algorithm's robustness against a limited set of malicious exploits, as opposed to the arbitrary adversarial behavior against which our algorithm is provably robust. The problem considered here is less general than the peer-to-peer resource location problem considered in [10]; for instance, we assume that in each trial, any agent may select any resource, whereas they assume that only a subset of the resources are available (namely, those peers who claim to have a copy of the requested file). Despite these differences, we believe that our work may shed light on the efficacy of EigenTrust while suggesting potential enhancements to make it more robust against Byzantine malicious users.

The rest of this paper is organized as follows. In Section 2 we specify our precise models and results. This is followed by a section specifying a general outline of our approach. The precise specification of the main algorithm, TrustFilter, appears in Section 3. The description of the algorithm is complete except for a rather complicated subroutine, BBA, which is specified and analyzed in the following section. Finally, in Section 5, we analyze the main algorithm, modulo a random graph lemma which is proved in the full version of this paper.

2 Statements of the Problem and the Results

The operating environment consists of a set X of n agents and a set Y of m products. A subset $H \subseteq X$ of the agents are *honest*, and the rest are dishonest. Honest agents are assumed to obey the distributed protocol to be specified, and to report their observations truthfully, while dishonest agents may behave in a Byzantine manner, disobeying the protocol or reporting fictitious observations as they wish. We will assume throughout that the number of honest agents is at least αn, where $\alpha > 0$ is a parameter which may be arbitrarily small. The agents do not initially know which ones are honest and which are dishonest, nor are they assumed to know the value of α.

In each of T consecutive rounds, a cost function $C_t : X \times Y \to [0, 1]$ is given. We think of the cost $C_t(x, y)$ as agent x's perception of the cost of resource y. The costs may be set by an adaptive adversary who is allowed to choose C_t based on the agents' actions in rounds $1, \ldots, t-1$ but not on their random decisions in the present or future rounds; the adversary may also use randomization in determining C_t. Define two agents x_1, x_2 to be *consistent* if the costs $C_t(x_1, y), C_t(x_2, y)$ are random variables with the same expected value (conditional on the choices of all agents in all rounds preceding t), for all $y \in Y, 1 \leq t \leq T$.[1] We will assume that the honest agents may be partitioned into k *coalitions*, such that two agents belonging to the same coalition are consistent; the honest agents do not initially know which coalitions the other honest agents belong to.

At the beginning of each round, each agent $x \in X$ must choose a product $y = y_t(x) \in Y$. Any agent is allowed to choose any product in any round. The cost of the choice is $C_t(x, y)$, and this cost (but not the cost of any other product) is revealed to x. The agents may communicate with each other between rounds, and this communication may influence their decisions in future rounds. To simplify the exposition we will assume all messages are exchanged using a shared, synchronous, public channel. In any round t all agents (including the Byzantine dishonest agents) must commit to their message on this channel before being able to read the messages posted by other agents in round t. This public-channel assumption is for expositional clarity only: in the full version of this paper we will indicate how to achieve the same results (with slightly worse bounds) in a message-passing model where agents may only exchange messages bilaterally on point-to-point communication links, subject to the assumption that all agents can synchronize clocks and have enough time to perform $\Omega(\log n)$ communication rounds in between consecutive decision rounds. (The Byzantine agents may eavesdrop on all such communications, whereas honest agents may not eavesdrop on any message if they are not the sender or receiver.) As might be expected, some subtleties arise in the message-passing model, due to ability of the Byzantine nodes to give differing advice to different parties, and to eavesdrop on others' messages.

[1] The randomness of the variables $C_t(x_1, y), C_t(x_2, y)$ is due to the adversary's potential use of randomness in determining C_t.

The goal of the algorithm is to minimize the total cost incurred by honest agents. As is typical with online decision problems, we will evaluate the algorithm's performance by measuring the expected difference between the algorithm's total cost and the cost of the best assignment in which each agent chooses a single fixed product and selects this product every time. This parameter is called *regret* and will be denoted by R.

$$R = \mathbf{E}\left[\sum_{x\in H}\sum_{t=1}^{T} C_t(x, y_t(x)) - \min_{y:H\to Y}\sum_{x\in H}\sum_{t=1}^{T} C_t(x, y(x))\right]. \quad (1)$$

The following two parameters, closely related to R, are also of interest:

– The *normalized individual regret* $\hat{R} = R/\alpha nT$ is the regret per unit time of the average honest agent. For all of the algorithms we will consider, $\hat{R}$ converges to zero as $T \to \infty$.
– The *δ-convergence time* of such an algorithm, denoted by $T(\delta)$, is defined as the minimum value of T necessary to guarantee that $\hat{R} = O(\delta)$. Here, δ is a positive constant which may be arbitrarily close to zero.

2.1 Our Results

We present a distributed algorithm, named TrustFilter, in Section 3. Let $\beta = 1 + m/n$. We will typically be interested in the case where α, β, δ are all positive constants. For ease of exposition, we will adhere to this assumption when stating the theorems in this section, absorbing such constants into the $O(\cdot)$ notation. See equations (11),(12),(13), (14) in Section 5 for bounds which explicitly indicate the dependence on α, β, and δ; in all cases, this dependence is polynomial.

Theorem 1. *Suppose the set of honest agents may be partitioned into k subsets $S_1, S_2, \ldots, S_k$, such that the agents in each subset are mutually consistent. Then the normalized regret $\hat{R}$ and δ-convergence time $T(\delta)$ of* TrustFilter *satisfy*

$$\hat{R} = O\left(k \cdot \frac{\log^4(n)}{T^{1/4}}\right) \quad (2)$$

$$T(\delta) = O(k^4 \log^{16}(n)). \quad (3)$$

The δ-convergence time bound follows from the regret bound. Typically we are interested in the case where α, β, δ, k are constants, hence we will summarize this result by saying that the algorithm has *polylogarithmic convergence time.* This is the first distributed algorithm with polylogarithmic convergence time in a dynamic environment.

3 The Algorithm TrustFilter

3.1 Intuition

As stated in the introduction, our algorithm is based on a Markov chain representing a random walk in a directed graph, whose vertices represent the set of resources and agents. We refer to this directed graph as the "reputation network." At each time, each agent picks an outgoing edge in the reputation network with appropriate probability, and then traverses this edge. If the edge leads to an agent, "advice" is sought from that agent. Else, if the edge leads to a resource, this resource is selected for sampling. Depending on the observed cost of the sampled resource, the agent updates its transition probabilities.

As an aid in developing intuition, consider the special case of this algorithm when the Markov chain is based on a random graph. Specifically, each agent picks at random a small subset of other agents and a small subset of the resources, and sets equal transition probabilities to all outgoing edges leading to members of that subset. All other outgoing probabilities are zero. Assume that agents adopt the following simple rule for updating their transition probabilities: if the agent chooses an outgoing edge and it leads to a product with cost 0, assign probability 1 permanently to that edge and probability 0 to all other edges; otherwise leave the transition probabilities unchanged. It is easy to prove, that for the static case with binary resource costs, this algorithm can be viewed as an alternative to Random Advice Random Sample algorithm in [3]; like that algorithm, it achieves logarithmic convergence time. The invariant used in the proof is the fact that the set of agents who recommend the optimal resource is growing exponentially with time. This invariant is proved by induction on time. Indeed, with high probability there is an edge in the reputation network from some honest agent to the optimal resource, and in constant time that neighboring agent will either directly sample this resource, or will stumble on an equivalent resource following advice of others. Consider the set S of honest agents who "saw the light," i.e., discovered the optimal resource. Note that the set N of neighbors of S, namely nodes with outgoing edges leading into S, is at least $|N| = |S| \cdot \rho$ where ρ is the expansion ratio of the underlying random graph. Note that within constant time, a constant fraction of agents in N will also discover the optimal resource by sampling nodes in S or following advice to other equivalent resources. Thus, within expected logarithmic time, all the agents discover the optimal resource.

Our algorithm for the case of dynamic costs looks quite different from the algorithm for static costs presented in the preceding paragraph, but it is based on the same intuition: by structuring the reputation network as a random graph, the set of honest agents who are selecting an optimal or near-optimal resource will grow exponentially over time. The main technical difference is that agents must update their transition probabilities using the multi-armed bandit algorithm, rather than shifting all of their probability mass to one outgoing edge as soon as they discover a resource with zero cost. This modification is necessary in order to deal with the fact that a resource which has zero cost at one time may not have zero cost at future times. More subtly, when agents are using the multi-armed bandit algorithm to update their transition probabilities, they must use a modification of the classical multi-armed bandit algorithm which we denote by BBA. This is because the agents do not know how many other honest agents belong to their coalition, so they must potentially consider all βn

other vertices of the reputation network as potential neighbors. (Recall from Section 2 that $\beta = (m+n)/n$, so that βn is the cardinality $X \cup Y$, the vertex set of the reputation network.) Classical multi-armed bandit algorithms, e.g. Exp3 [1], will have a convergence time of $\Omega(n \log(n))$ in such a scenario, whereas we seek a polylogarithmic convergence time. Accordingly, we present a modified bandit algorithm BBA whose salient feature is that it satisfies a significantly better bound on regret when stopped at times $T < n \log(n)$. The details of this algorithm will be explained in Section 4. For now, it is best for the reader to consider it as a black box (instantiated separately by each agent x) which outputs, at each time t, a probability distribution $\pi_t(x)$ on the set of all agents and resources. We will use the notation $\pi_t(x, y)$ to denote the probability that $\pi_t(x)$ assigns to the element $y \in X \cup Y$.

3.2 The Algorithm

Here we present an algorithm TrustFilter which solves the collaborative learning problem, establishing Theorem 1. We use, as a subroutine, the algorithm BBA whose existence is asserted by Theorem 2. We defer the specification of this subroutine until later.

At the beginning of each round t, each agent x queries its local bandit algorithm BBA(x) to obtain a probability distribution $\pi_t(x)$ on the set of agents and resources, and posts this distribution on the public channel. This enables each agent to construct an $(m+n)$-by-$(m+n)$ matrix M_t whose rows and columns are indexed by the elements of $X \cup Y$, and whose entries are given by:

$$(M_t)_{ij} = \begin{cases} \pi_t(i,j) & \text{if } i \in X \\ 1 & \text{if } i \in Y \text{ and } j = i \\ 0 & \text{if } i \in Y \text{ and } j \neq i. \end{cases}$$

We may think of M_t as the transition matrix for a Markov chain with state space $X \cup Y$, in which elements of Y are absorbing states, and the transition probabilities at an element x of X are determined by the bandit algorithm BBA(x). This Markov chain corresponds to the intuitive notion of "taking a random walk by following the advice of the bandit algorithm at each node."

The random walk starting from $x \in X$ will, with probability 1, be absorbed by some state $y \in Y$; this enables us to define a matrix A_t by

$$(A_t)_{ij} = \Pr(\text{absorbing state is } j \,|\, \text{starting state is } i).$$

Algebraically, A_t satisfies the equations $M_t A_t = A_t$ and $A_t \mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ represents a column vector whose components are all equal to 1.

To select a product $y = y_t(x) \in Y$, x uses BBA(x) to choose a strategy $s = s_t(x) \in X \cup Y$. It then samples y randomly using the probability distribution in the row of A_t corresponding to s, learns the cost $C_t(y)$, and returns this feedback score to BBA(x). The probability distribution from which y is drawn can be determined either by computing A_t algebraically, or by simulating the random walk with transition matrix M_t starting from state s until it hits an absorbing state. We call this probability distribution on Y *harmonic measure relative to* x, by analogy with the harmonic measure defined on the boundary of a bounded domain $U \subset \mathbb{R}^d$ according the hitting probability of Brownian motion starting from a point $x \in U$.

4 The Biased Bandit Algorithm BBA

Our multi-agent learning algorithms require each agent to instantiate a single-agent learning algorithm called the *biased bandit algorithm*, or BBA, which we describe in this section. For a multi-armed bandit algorithm with strategy set $S = \{1, 2, \ldots, K\}$ (whose selection at time t is denoted by $i_t \in S$) define its *regret profile* to be the function $R(T, i)$ which specifies the algorithm's regret relative to strategy $i \in S$ at time $T \geq 1$, i.e.

$$R(T, i) = \max \left\{ \mathbf{E} \left(\sum_{t=1}^{T} C_t(i_t) - C_t(i) \right) \right\},$$

the maximum being taken over the set of all adaptive adversarial policies assigning costs C_t in $[0, 1]$. For example, the regret profile of the classical multi-armed bandit algorithm Exp3 is known to satisfy $R(T, i) = O(\sqrt{TK \log(K)})$; we call this a uniform regret profile since the value of $R(T, i)$ does not depend on i. Which *non-uniform* regret profiles are achievable by multi-armed bandit algorithms? The BBA supplies a non-trivial upper bound for this question.

Theorem 2. *Let a strategy set S of size K be given, along with positive real weights $\{w_i\}_{i \in S}$ which sum to 1. There exists a multi-armed bandit algorithm* BBA *whose regret profile satisfies*

$$R(T, i) = O\left(\frac{1}{w_i} \log^2 \left(\frac{1}{w_i} \right) T^{3/4} \right).$$

In fact, the theorem holds even if the feedback for choosing strategy i, rather than being equal to $C_t(i)$, is a random variable $X_t(i)$ (taking values in $[0, 1]$) whose conditional expectation is bounded above by $C_t(i)$. We call this the "noisy feedback model"; see Section 4.1 for details.

When the BBA algorithm is applied as a subroutine in TrustFilter, its strategy set is $S = X \cup Y$, which has $m + n$ elements. The weights assigned to these elements are a random permutation of the set

$$\left\{ \frac{1}{H_{m+n}}, \frac{1}{2H_{m+n}}, \ldots, \frac{1}{(m+n)H_{m+n}} \right\}.$$

(Here H_{m+n} represents the harmonic number $\sum_{i=1}^{m+n} 1/i$.)

One way of interpreting BBA is as an *anytime* version of the multi-armed bandit algorithm, in that it meets a non-trivial performance guarantee when stopped at any time T, even if $T \ll K$. This contrasts with traditional multi-armed bandit algorithms such as Exp3 whose performance at time $T = o(K)$ is generally indistinguishable from random guessing. The anytime guarantee for BBA can be made precise as follows. For any threshold $\lambda > 0$ let $S(\lambda) = \{i \in S : w_i \log^{-3}(1/w_i) > \lambda\}$. Theorem 2 establishes that by time T, the normalized regret of BBA relative to the strategies in $S(\delta^{-1}T^{-1/6})$ is at most δ. This set of strategies may be quite large even when $T \ll K$, and grows to encompass all of S as $T \to \infty$.

We will now describe the algorithm BBA. The high-level idea of the algorithm is to partition the strategy set into two subsets of approximately equal weight, then to further partition each of these two subsets into two pieces of approximately equal weight, and so on, building a tree $\mathfrak{T}$ whose leaves are labeled with elements of the strategy set S. We will use, as a black box, the multi-armed bandit algorithm Exp3 from [1]. An instance of Exp3 at the root of the tree is responsible for deciding whether to select a strategy from the left or right subtree; the task of picking a leaf of this subtree is recursively delegated to the descendants of the root. Each node z of the tree is therefore running an instance of Exp3, but gets feedback only for a random subset of the rounds, namely those in which the chosen leaf lies in its subtree. The analysis of Exp3 in models such as this, where the feedback is noisy or sporadic, is carried out in Appendix 4.1. Applying the relevant bound on Exp3's regret (Theorem 3) at each level of $\mathfrak{T}$, we will obtain the desired global upper bound on regret. A subtlety which arises in designing the algorithm is that we must ensure that each internal node z gets feedback reasonably often, which necessitates devoting a small fraction of the rounds to explicitly sampling a descendant of z.

To specify the tree $\mathfrak{T}$, we may assume without loss of generality that the weights w_i are powers of 2, say $w_i = 2^{-d_i}$. (If not, we may round each w_i down to the next-lowest power of 2, then round some of them up to restore the property that their sum is 1.) Now define $\mathfrak{T}$ to be the Huffman tree of the distribution $\{w_i\}$ [6]. This tree has the property that for any node at depth d, the combined weight of all leaves in its subtree is 2^{-d}. For a node z of depth d in $\mathfrak{T}$, let $\tilde{w}(z) = 2^{-d} \cdot d^{-2}$; note that if z is a leaf corresponding to an element i, then

$$\frac{1}{\tilde{w}(z)} = O\left(\frac{1}{w_i} \log^2\left(\frac{1}{w_i}\right)\right).$$

In the BBA algorithm, each internal node z of $\mathfrak{T}$ maintains an instance Exp3(z) of the multi-armed bandit algorithm, with a two-element strategy set identified with the two children, z_L and z_R, of z in $\mathfrak{T}$. In each round t, each internal node z chooses a child $\chi_t(z)$ according to the probability distribution supplied by Exp3(z). These choices define a mapping ℓ_t from the nodes of $\mathfrak{T}$ to the leaves of $\mathfrak{T}$, defined recursively by the rule that $\ell_t(i) = i$ for a leaf i, and $\ell_t(z) = \ell_t(\chi_t(z))$ for an internal node z. A random node $z_t \in \mathfrak{T}$ is sampled in round t according to the distribution which assigns probability $\rho(z) = T^{-1/4}\tilde{w}(z)$ to each node z other than the root r, and assigns all remaining probability mass to r. The algorithm BBA chooses strategy $i_t = \ell_t(z_t)$. Let P_t denote the path in $\mathfrak{T}$ from

z_t to i_t. After learning the cost $C_t(i_t)$, each internal node z updates $\mathsf{Exp3}(z)$ by attributing a feedback value $X_t(z')$ to its child $z' = \chi_t(z)$ as follows.

$$X_t(z') = \begin{cases} C_t(\ell_t(z')) & \text{if } z = z_t \\ 0 & \text{otherwise,} \end{cases}$$

4.1 Analysis of BBA

To prove Theorem 2 we must first recall some properties of the multi-armed bandit algorithm Exp3 from [1] and extend the analysis of this algorithm to a slightly more general setting, which we call the "noisy feedback model."

Theorem 3 ([1]). *For any $\varepsilon > 0$, there is a multi-armed bandit algorithm* Exp3 *with strategy set $S = \{1, 2, \ldots, K\}$ whose regret relative to strategy $i \in S$, i.e. the number*

$$R = \mathbf{E}\left(\sum_{t=1}^{T} C_t(i_t) - \sum_{t=1}^{T} C_t(i)\right),$$

satisfies $R = O(\sqrt{TK\log(K)})$.

For the applications in this paper, we actually need to work with a slight generalization of the model considered in [1]. This generalization, which we call the "noisy feedback" model, is described as follows. In each round t, in addition to specifying a cost function $C_t : S \to [0,1]$, the adversary specifies, for each $i \in S$, a random variable $X_t(i)$ which depends only on $i_1, i_2, \ldots, i_{t-1}$ and on some random bits independent of the algorithm's random bits. This random variable takes values in $[0,1]$ and satisfies $\mathbf{E}[X_t(i) \,\|\, \mathcal{F}_{<t}] = C_t(i)$, where $\mathcal{F}_{<t}$ denotes the σ-field generated by all random variables revealed by the algorithm and adversary prior to time t. Rather than receiving $C_t(i_t)$ as feedback, the algorithm's feedback is $X_t(i_t)$. The following easy proposition, whose proof appears in the full version of this paper, demonstrates that the regret of algorithm Exp3 is unaffected by the noisy feedback.

Proposition 1. *In the noisy feedback model, the regret R experienced by algorithm* Exp3 *relative to strategy i still satisfies $R = O(\sqrt{TK\log K})$. This bound holds regardless of whether R is defined as $R_X := \mathbf{E}\left(\sum_{t=1}^{T} X_t(i_t) - C_t(i)\right)$ or as $R_C := \mathbf{E}\left(\sum_{t=1}^{T} C_t(i_t) - C_t(i)\right)$.*

We are now ready to finish the analysis of the BBA algorithm.

Proof (Theorem 2). The analysis of BBA depends on a reduction to the noisy-feedback bandit problem defined above. If z is a node of $\mathfrak{T}$ and z' is one of its two children, define:

$$\tilde{C}_t(z') = \rho(z) C_t(\ell_t(z')).$$

Then $\tilde{C}_t(z'), X_t(z')$ take values in $[0,1]$, and they are independent of $\mathsf{Exp3}(z)$'s random choices at times $t, t+1, \ldots, T$. Moreover, recalling that $\rho(z) = \Pr(z =$

$z_t \parallel \mathcal{F}_{<t}$), we have $\mathbf{E}[X_t(z') \parallel \mathcal{F}_{<t}] = \tilde{C}_t(z')$. Therefore, Exp3($z$) is following the algorithm Exp3 in the noisy feedback model with cost functions $\tilde{C}_t$ and random feedback variables $X_t(z')$. Applying Proposition 1 with $K = 2$,

$$\rho(z)\mathbf{E}\left(\sum_{t=1}^{T} C_t(\ell_t(z)) - C_t(\ell_t(z'))\right) = O(\sqrt{T}).$$

This inequality holds for every edge (z, z') in $\mathfrak{T}$. Rescaling and summing over all the edges on the path P from r to i, we obtain:

$$\begin{aligned}\mathbf{E}\left(\sum_{t=1}^{T}(C_t(\ell_t(r)) - C_t(i))\right) &= \sum_{(z,z')\in P} \mathbf{E}\left(\sum_{t=1}^{T}(C_t(\ell_t(z)) - C_t(\ell_t(z')))\right)\\ &= O\left(\sum_{z\in P} \rho(z)^{-1}T^{1/2}\right)\\ &= O\left(\sum_{z\in P} \tilde{w}(z)^{-1}T^{3/4}\right)\\ &= O\left(\tilde{w}(i)^{-1}T^{3/4}\right)\\ &= O\left(\frac{1}{w_i}\log^2\left(\frac{1}{w_i}\right)T^{3/4}\right). \qquad (4)\end{aligned}$$

Finally, we may account for the cost of the steps in which $z_t \neq r$ as follows:

$$\begin{aligned}\mathbf{E}\left(\sum_{t=1}^{T} C_t(i_t) - C_t(\ell_t(r))\right) &\le \sum_{t=1}^{T} \Pr(i_t \neq \ell_t(r))\\ &\le \sum_{t=1}^{T} \Pr(z_t \neq r)\\ &= T \cdot \sum_{z\neq r} \rho(z) = O(T^{3/4}). \qquad (5)\end{aligned}$$

Summing the bounds (4) and (5) we obtain the desired bound on the regret of BBA:

$$\begin{aligned}R &= \mathbf{E}\left(\sum_{t=1}^{T} C_t(i_t) - C_t(\ell_t(r))\right) + \mathbf{E}\left(\sum_{t=1}^{T} C_t(\ell_t(r)) - C_t(i)\right)\\ &= O(T^{3/4}) + O\left(\frac{1}{w_i}\log^2\left(\frac{1}{w_i}\right)T^{3/4}\right)\\ &= O\left(\frac{1}{w_i}\log^2\left(\frac{1}{w_i}\right)T^{3/4}\right).\end{aligned}$$

5 Analysis of Algorithm TrustFilter

In this section we complete the analysis of algorithm TrustFilter by proving Theorem 1.

Proof (Theorem 1.). For $x \in X, s \in X \cup Y$, let

$$\tilde{C}_t(x,s) = \begin{cases} C_t(x,s) & \text{if } s \in Y \\ \mathbf{E}[C_t(x,y_t(s))] & \text{if } s \in X. \end{cases}$$

From the standpoint of agent x, the bandit algorithm $\mathsf{BBA}(x)$ is running in the noisy feedback model with cost functions $\tilde{C}_t(x,\cdot)$ and random feedback variables $X_t(s)$ distributed according to the cost $(C_t(x,y))$ of a random product $y \in Y$ sampled according to the harmonic measure relative to s. It follows from the analysis of BBA that for each pair of elements u, v in $H \cup Y$,

$$\mathbf{E}\left(\sum_{t=1}^{T}(\tilde{C}_t(u,u) - \tilde{C}_t(u,v))\right) = O\left(\frac{1}{w(u,v)}\log^2\left(\frac{1}{w(u,v)}\right)T^{3/4}\right). \tag{6}$$

Here $w(u,v)$ denotes the random weight assigned to strategy v by $\mathsf{BBA}(u)$ at initialization time. Using the fact that $1/w(u,v) = O(\beta n \log(\beta n))$, and that $\tilde{C}(u,v) = \tilde{C}(v,v)$ when u, v are consistent, we may rewrite (6) as

$$\mathbf{E}\left[\left(\sum_{t=1}^{T}\tilde{C}_t(u,u)\right) - \left(\sum_{t=1}^{T}\tilde{C}_t(v,v)\right)\right] = O\left(\frac{1}{w(u,v)}T^{3/4}\log^2(\beta n)\right), \tag{7}$$

provided that u and v are consistent. Let's introduce the following notations:

$$\bar{C}(u) = \mathbf{E}\left(\frac{1}{T}\sum_{t=1}^{T}\tilde{C}_t(u,u)\right)$$
$$B = \log^3(\beta n)T^{-1/4}$$
$$d(u,v) = (H_{m+n}w(u,v))^{-1}.$$

Then (7) may be rewritten as

$$\bar{C}(u) - \bar{C}(v) = d(u,v)\cdot O(B) \tag{8}$$

Note that for a product $y \in Y$, $\bar{C}(y)$ is simply the average cost of y, and for an agent $x \in H$, $\bar{C}(x)$ is the average cost of the products sampled by x. Let S be a consistent cluster containing x, and let $\alpha(S) = |S|/n$. Letting y^* denote a product with minimum average cost for members of S, and letting P denote a shortest path from x to y^* in the directed graph with vertex set $S \cup Y$ and edge lengths given by $d(\cdot,\cdot)$, we may sum up the bounds (8) over the edges of P to obtain

$$\bar{C}(x) - \bar{C}(y^*) = O(\mathit{length}(P)\cdot B) \tag{9}$$

Observe that the left side is the expected normalized regret of agent x. The random edge lengths $d(u, v)$ on the $m + n$ outgoing edges from u are simply the numbers $\{1, 2, \ldots, m + n\}$ in a random permutation. For graphs with random edge lengths specified according to this distribution, the expected distance between two given vertices is $O((\beta/\alpha) \log n)$.[2] We may conclude that the expectation of the right side of (9) is

$$O((\beta/\alpha(S)) \log(n)B) = O((\beta/\alpha(S)) \log^4(\beta n)T^{-1/4}). \tag{10}$$

It follows that the normalized regret and δ-convergence time for agents in the cluster S satisfy

$$\hat{R} = O\left(\left(\frac{\beta}{\alpha(S)}\right) \log^4(\beta n)T^{-1/4}\right) \tag{11}$$

$$T(\delta) = O\left(\left(\frac{\beta}{\alpha(S)\delta}\right)^4 \log^{16}(\beta n)\right). \tag{12}$$

Note that (12) can be interpreted as saying that the large consistent clusters learn to approximate the cost of the best resource much more rapidly than do the small clusters, which accords with one's intuition about collaborative learning. To obtain Theorem 1, we must average over the k consistent clusters $S_1, \ldots, S_k$. We may multiply the regret bound for a cluster S in (10) by the size of S, to obtain an upper bound of $O(\beta n \log^4(\beta n)T^{-1/4})$ on the aggregate regret of users in S. Summing over k such clusters, the cumulative regret of all honest users is $O(k\beta n \log^4(\beta n)T^{-1/4})$, so the normalized regret and convergence time satisfy:

$$\hat{R} = O\left(k \cdot \left(\frac{\beta}{\alpha}\right) \log^4(\beta n)T^{-1/4}\right) \tag{13}$$

$$T(\delta) = O\left(k^4 \cdot \left(\frac{\beta}{\alpha\delta}\right)^4 \log^{16}(\beta n)\right). \tag{14}$$

6 Open Problems

In this paper we have introduced and analyzed an algorithm for a simple model of collaborative learning. A key feature of our model is the presence of a large number of dishonest agents who are assumed to behave in an arbitrary Byzantine manner. However, other aspects of our model are quite idealized, and there are some very natural extensions of the model which more closely reflect the reality of collaborative learning systems such as eBay's reputation system and peer-to-peer resource discovery systems. It would be desirable to identify algorithms for some of the following extensions.

[2] A proof of this random graph lemma appears in the full version of this paper.

1. Study *asynchronous* collaborative learning, in which only a subset of the agents act as decision-makers in each round and the rest are inactive.
2. Study cases in which agents are constrained to choose from a proper subset of the resources, e.g. because the set of available resources is changing over time or because of limitations on the set of resources that a given agent is *ever* allowed to select.
3. Consider stronger models of collaborative filtering, by relaxing the consistency condition for two agents x_1, x_2 to belong to the same cluster. For example, consider the case where x_1, x_2 are consistent if $|C_t(x_1, y) - C_t(x_2, y)| < \varepsilon$ for all y, t, or consider the *mixture model* as in [9].
4. Study more structured collaborative decision-making problems, e.g. selecting routing paths in a network, some of whose nodes are identified with the agents.

References

1. Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, Los Alamitos, CA, 1995.
2. Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Mark Tuttle. Collaboration of untrusting peers. In *Proc. of ACM conference on Electronic Commerce (EC)*, May 2004.
3. Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Mark Tuttle. Improved recommendation systems. In *Proc. of ACM SIAM Conference on Discreet Algorithms (SODA)*, January 2005.
4. Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proc. 33rd Ann. ACM Symp. on Theory of Computing (STOC)*, pages 619–626, 2001.
5. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
6. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms. 2nd Edition.* MIT Press, 2001.
7. Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *Proc. 34th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 82–90, 2002.
8. David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.
9. Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, pages 688–693, 1999.
10. Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proc. 12th Int. World Wide Web Conference (WWW)*, 2003.
11. Jon Kleinberg. Finding authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

12. Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–91, 1999.
13. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
14. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175 – 186, October 1994.
15. Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *Cooperative Information Agents*, pages 154–165, 2000.
16. Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms in electronic marketplaces. In *HICSS*, 1999.
17. Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.