

Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems

David R. Karger*, Sewoong Oh[†] and Devavrat Shah[‡]

March 27, 2013

Abstract

Crowdsourcing systems, in which numerous tasks are electronically distributed to numerous “information piece-workers”, have emerged as an effective paradigm for human-powered solving of large scale problems in domains such as image classification, data entry, optical character recognition, recommendation, and proofreading. Because these low-paid workers can be unreliable, nearly all such systems must devise schemes to increase confidence in their answers, typically by assigning each task multiple times and combining the answers in an appropriate manner, e.g. majority voting.

In this paper, we consider a general model of such crowdsourcing tasks and pose the problem of minimizing the total price (i.e., number of task assignments) that must be paid to achieve a target overall reliability. We give a new algorithm for deciding which tasks to assign to which workers and for inferring correct answers from the workers’ answers. We show that our algorithm, inspired by belief propagation and low-rank matrix approximation, significantly outperforms majority voting and, in fact, is optimal through comparison to an oracle that knows the reliability of every worker. Further, we compare our approach with a more general class of algorithms which can dynamically assign tasks. By adaptively deciding which questions to ask to the next arriving worker, one might hope to reduce uncertainty more efficiently. We show that, perhaps surprisingly, the minimum price necessary to achieve a target reliability scales in the same manner under both adaptive and non-adaptive scenarios. Hence, our non-adaptive approach is order-optimal under both scenarios. This strongly relies on the fact that workers are fleeting and can not be exploited. Therefore, architecturally, our results suggest that building a reliable worker-reputation system is essential to fully harnessing the potential of adaptive designs.

*Computer Science and Artificial Intelligence Laboratory and Department of EECS at Massachusetts Institute of Technology. Email: karger@mit.edu

[†]Department of Industrial and Enterprise Systems Engineering at University of Illinois at Urbana-Champaign. Email: swoh@illinois.edu

[‡]Laboratory for Information and Decision Systems and Department of EECS at Massachusetts Institute of Technology. Email: devavrat@mit.edu. This work was supported in parts by NSF EMT project, AFOSR Complex Networks project and Army Research Office under MURI Award 58153-MA-MUR.

1 Introduction

Background. Crowdsourcing systems have emerged as an effective paradigm for human-powered problem solving and are now in widespread use for large-scale data-processing tasks such as image classification, video annotation, form data entry, optical character recognition, translation, recommendation, and proofreading. Crowdsourcing systems such as Amazon Mechanical Turk¹, establish a market where a “taskmaster” can submit batches of small tasks to be completed for a small fee by any worker choosing to pick them up. For example a worker may be able to earn a few cents by indicating which images from a set of 30 are suitable for children (one of the benefits of crowdsourcing is its applicability to such highly subjective questions).

Because these crowdsourced tasks are tedious and the pay is low, errors are common even among workers who make an effort. At the extreme, some workers are “spammers”, submitting arbitrary answers independent of the question in order to collect their fee. Thus, all crowdsourcers need strategies to ensure the reliability of their answers. When the system allows the crowdsourcers to identify and reuse particular workers, a common approach is to manage a pool of reliable workers in an explore/exploit fashion. However in many crowdsourcing platforms such as Amazon Mechanical Turk, the worker crowd is large, anonymous, and transient, and it is generally difficult to build up a trust relationship with particular workers.² It is also difficult to condition payment on correct answers, as the correct answer may never truly be known and delaying payment can annoy workers and make it harder to recruit them to your task next time. Instead, most crowdsourcers resort to redundancy, giving each task to multiple workers, paying them all irrespective of their answers, and aggregating the results by some method such as majority voting.

For such systems there is a natural core optimization problem to be solved. Assuming the taskmaster wishes to achieve a certain reliability in her answers, how can she do so at minimum cost (which is equivalent to asking how she can do so while asking the fewest possible questions)?

Several characteristics of crowdsourcing systems make this problem interesting. Workers are neither persistent nor identifiable; each batch of tasks will be solved by a worker who may be completely new and who you may never see again. Thus one cannot identify and reuse particularly reliable workers. Nonetheless, by comparing one worker’s answer to others’ on the same question, it is possible to draw conclusions about a worker’s reliability, which can be used to weight their answers to other questions in their batch. However, batches must be of manageable size, obeying limits on the number of tasks that can be given to a single worker.

Another interesting aspect of this problem is the *choice of task assignments*. Unlike many inference tasks which makes inferences based on a fixed set of signals, our algorithm can choose which signals to measure by deciding which questions to include in which batches. In addition, there are several plausible options: for example, we might choose to ask a few “pilot questions” to each worker (just like a qualifying exam) to decide on the reliability of the worker. Another possibility is to first ask few questions and based on the answers decide to ask more questions or not. We would like to understand the role of all such variations in the overall optimization of budget for reliable task processing.

In the remainder of this section, we will define a formal probabilistic model that captures these aspects of the problem. We consider both a *non-adaptive scenario*, in which all questions are

¹<http://www.mturk.com>

²For certain high-value tasks, crowdsourcers can use entrance exams to “prequalify” workers and block spammers, but this increases the cost of the task and still provides no guarantee that the workers will try hard after qualification.

asked simultaneously and all the responses are collected simultaneously, and an *adaptive scenario*, in which one may adaptively choose which tasks to assign to the next arriving worker based on all the previous answers collected thus far. We provide a non-adaptive task allocation scheme and an inference algorithm based on low-rank matrix approximations and belief propagation. We will then show that our algorithm is order-optimal: for a given target error rate, it spends only a constant factor times the minimum necessary to achieve that error rate. The optimality is established through comparisons to the best possible *non-adaptive* task allocation scheme and an oracle estimator that can make optimal decisions based on extra information provided by an oracle. In particular, we derive a parameter q that characterizes the ‘collective’ reliability of the crowd, and show that to achieve target reliability ε , it is both necessary and sufficient to replicate each task $\Theta(1/q \log(1/\varepsilon))$ times. This leads to the next question of interest: by using adaptive task assignment, can we ask fewer questions and still achieve the same error rate? We, somewhat surprisingly, show that the optimal costs under this adaptive scenario scale in the same manner as the non-adaptive scenario; asking questions adaptively does not help!

Setup. We consider the following probabilistic model for crowdsourcing. There is a set of m binary tasks which is associated with unobserved ‘correct’ solutions: $\{t_i\}_{i \in [m]} \in \{\pm 1\}^m$. Here and after, we use $[N]$ to denote the set of first N integers. In the image categorization example stated earlier, a set of tasks corresponds to labeling m images as suitable for children (+1) or not (-1). We will be interested in finding the true solutions by querying noisy workers who arrive one at a time in an on-line fashion.

An algorithmic solution to crowdsourcing consists of two components: a task allocation scheme and an inference algorithm. At *task allocation phase* queries are made sequentially according to the following rule. At j -th step, the task assignment scheme chooses a subset $T_j \subseteq [m]$ of tasks to be assigned to the next arriving noisy worker. The only constraint on the choice of the batch is that the size $|T_j|$ must obey some limit on the number of tasks that can be given to a single worker. Let r denote such a limit on the number of tasks that can be assigned to a single worker, such that all batches must satisfy $|T_j| \leq r$. Then, a worker j arrives, whose latent reliability is parametrized by $p_j \in [0, 1]$. For each assigned task, this worker gives a noisy answer such that

$$\mathbf{A}_{ij} = \begin{cases} t_i & \text{with probability } p_j, \\ -t_i & \text{otherwise,} \end{cases}$$

and $\mathbf{A}_{ij} = 0$ if $i \notin T_j$. (Throughout this paper, we use boldface characters to denote random variables and random matrices unless it is clear from the context.) The next assignment T_{j+1} can be chosen adaptively, taking into account all of the previous assignments and the answers collected thus far. This process is repeated until the task assignment scheme decides to stop, typically when the total number of queries meet a certain budget constraint. Then, in the subsequent *inference phase*, an inference algorithm makes a final estimation of the true answers.

We say a task allocation scheme is *adaptive* if the choice of T_j depends on the answers collected on previous steps, and it is *non-adaptive* if it does not depend on the answers. In practice, one might prefer using a non-adaptive scheme, since assigning all the batches simultaneously and having all the batches of tasks processed in parallel reduces latency. However, by switching to an adaptive task allocation, one might be able to reduce uncertainty more efficiently. We investigate this possibility in Section 2.4, and show that the gain from adaptation is limited.

Note here that at the time of assigning tasks T_j for a next arriving worker j , the algorithm is not aware of the latent reliability of the worker. This is consistent with how real-world crowdsourcing

works, since taskmasters typically have no choice over which worker is going to pick up which batch of tasks. Further, we make the pessimistic assumption that workers are neither persistent nor identifiable; each batch of tasks T_j will be solved by a worker who may be completely new and who you may never see again. Thus one cannot identify and reuse particularly reliable workers. This is a different setting from adaptive games [LW89], where you have a sequence of trials and a set of predictions is made at each step by a pool of experts. In adaptive games, you can identify reliable experts from their past performance using techniques like multiplicative weights, whereas in crowdsourcing you cannot hope to exploit any particular worker.

The latent variable p_j captures how some workers are more diligent or have more expertise than others, while some other workers might be trying to cheat. The random variable \mathbf{A}_{ij} is independent of any other event given p_j . The underlying assumption here is that the error probability of a worker does not depend on the particular task and all the tasks share an equal level of difficulty. Hence, each worker’s performance is consistent across different tasks. We discuss a possible generalization of this model in Section 2.7.

We further assume that the reliability of workers $\{\mathbf{p}_j\}$ are independent and identically distributed random variables with a given distribution on $[0, 1]$. As one example we define the *spammer-hammer model*, where each worker is either a ‘hammer’ with probability q or is a ‘spammer’ with probability $1 - q$. A hammer answers all questions correctly, meaning $\mathbf{p}_j = 1$, and a spammer gives random answers, meaning $\mathbf{p}_j = 1/2$. It should be noted that the meaning of a ‘spammer’ might be different from its use in other literature. In this model, a spammer is a worker who gives uniformly random labels independent of the true label. In other literature in crowdsourcing, the word ‘spammer’ has been used, for instance, to refer to a worker who always gives ‘+’ labels [RY12]. Another example is the beta distribution with some parameters $\alpha > 0$ and $\beta > 0$ ($f(p) = p^{\alpha-1}(1-p)^{\beta-1}/B(\alpha, \beta)$ for a proper normalization $B(\alpha, \beta)$) [Hol11, RYZ⁺10b]. A distribution of \mathbf{p}_j characterizes a crowd, and the following parameter plays an important role in capturing the ‘collective quality’ of this crowd, as will be clear from our main results:

$$q \equiv \mathbb{E}[(2\mathbf{p}_j - 1)^2].$$

A value of q close to one indicates that a large proportion of the workers are diligent, whereas q close to zero indicates that there are many spammers in the crowd. The definition of q is consistent with use of q in the spammer-hammer model and in the case of beta distribution, $q = 1 - (4\alpha\beta/((\alpha + \beta)(\alpha + \beta + 1)))$. We will see later that our bound on the achievable error rate depends on the distribution only through this parameter q .

When the crowd population is large enough such that we do not need to distinguish whether the workers are ‘sampled’ with or without replacement, then it is quite realistic to assume the existence of a prior distribution for \mathbf{p}_j . In particular, it is met if we simply randomize the order in which we upload our task batches, since this will have the effect of randomizing which workers perform which batches, yielding a distribution that meets our requirements. The model is therefore quite general. On the other hand, it is not realistic to assume that we know what the prior is. To execute our inference algorithm for a given number of iterations, we do not require any knowledge of the distribution of the reliability. However, q is necessary in order to determine how many times a task should be replicated and how many iterations we need to run to achieve a certain target reliability. We discuss a simple way to overcome this limitation in Section 2.2.

The only assumption we make about the distribution is that there is a bias towards the right answer, i.e. $\mathbb{E}[\mathbf{p}_j] > 1/2$. Without this assumption, we can have a ‘perfect’ crowd with $q = 1$, but

everyone is adversarial, $\mathbf{p}_j = 0$. Then, there is no way we can correct for this. Another way to justify this assumption is to define the “ground truth” of the tasks as what the majority of the crowd agrees on. We want to learn this consensus efficiently without having to query everyone in the crowd for every task. If we use this definition of the ground truth, then it naturally follows that the workers are on average more likely to be correct.

Throughout this paper, we are going to assume that there is a fixed cost you need to pay for each response you get regardless of the quality of the response, such that the total cost is proportional to the total number of queries. When we have a given target accuracy we want to achieve, and under the probabilistic crowdsourcing model described in this section, we want to design a task allocation scheme and an inference algorithm that can achieve this target accuracy with minimal cost.

Possible deviations from our model. Some of the main assumptions we make on how crowdsourcing systems work are (a) workers are neither identifiable nor reusable, (b) every worker is paid the same amount regardless of their performance, and (c) each worker completes only one batch and she completes all the tasks in that batch. In this section, we discuss common strategies used in real crowdsourcing that might deviate from these assumptions.

First, there has been growing interest recently in developing algorithms to efficiently identify good workers assuming that worker identities are known and *workers are reusable*. Imagine a crowdsourcing platform where there are a fixed pool of identifiable workers and we can assign the tasks to whichever worker we choose to. In this setting, adaptive schemes can be used to significantly improve the accuracy while minimizing the total number of queries. It is natural to expect that by first exploring to find better workers and then exploiting them in the following rounds, one might be able to improve performance significantly. Donmez et al. [DCS09] proposed IETHresh which simultaneously estimates worker accuracy and actively selects a subset of workers with high accuracy. Zheng et al. [ZSD10] proposed a two-phase approach to identify good workers in the first phase and utilize the best subset of workers in the second phase. Ertekin et al. [EHR11] proposed using a weighted majority voting to better estimate the true labels in CrowdSense, which is then used to identify good workers.

The power of such exploration/exploitation approaches were demonstrated on numerical experiments, however none of these approaches are tested on real-world crowdsourcing. All the experiments are done using *pre-collected* datasets. Given these datasets they simulate a labor market where they can track and reuse any workers they choose to. The reason that the experiments are done on such simulated labor markets, instead of on popular crowdsourcing platforms such as Amazon Mechanical Turk, is that on real-world crowdsourcing platforms it is almost impossible to track workers. Many of the popular crowdsourcing platforms are completely open labor markets where the worker crowd is large and transient. Further, oftentimes it is the workers who choose which tasks they want to work on, hence the taskmaster cannot reuse particular workers. For these reasons, we assume in this paper that the workers are fleeting and provide an algorithmic solution that works even when workers are not reusable. We show that any taskmaster who wishes to outperform our algorithm must adopt complex worker-tracking techniques. Furthermore, no worker-tracking technique has been developed that has been proven to be foolproof. In particular, it is impossible to prevent a worker from starting over with a new account. Many tracking algorithms are susceptible to this attack.

Another important and closely related question that has not been formally addressed in crowd-

sourcing literature is how to *differentiate the payment* based on the inferred accuracy in order to incentivize good workers. Regardless of whether the workers are identifiable or not, when all the tasks are completed we get an estimate of the quality of the workers. It would be desirable to pay the good workers more in order to incentivize them to work for us in the future tasks. For example, bonuses are built into Amazon Mechanical Turk to be granted at the taskmaster’s discretion, but it has not been studied how to use bonuses optimally. This could be an interesting direction for future research.

It has been observed that increasing the cost on crowdsourcing platforms does not directly lead to higher quality of the responses [MW10]. Instead, increasing the cost only leads to faster responses. Mason and Watts [MW10] attributes this counterintuitive findings to an “anchoring” effect. When the (expected) payment is higher, workers perceive the value of their work to be greater as well. Hence, they are no more motivated than workers who are paid less. However, these studies were done in isolated experiments, and the long term effect of taskmasters’ keeping a good reputation still needs to be understood. Workers of Mechanical Turk can manage reputation of the taskmasters using for instance Turkopticon³, a Firefox extension that allows you to rate taskmasters and view ratings from other workers. Another example is Turkernation⁴, an on-line forum where workers and taskmasters can discuss Mechanical Turk and leave feedback.

Finally, in Mechanical Turk, it is typically the workers who *choose which tasks they want to work on* and when they want to stop. Without any regulations, they might respond to multiple batches of your tasks or stop in the middle of a batch. It is possible to systematically prevent the same worker from coming back and repeating more than one batch of your tasks. For example, on Amazon’s Mechanical Turk, a worker cannot repeat the same task more than once. However, it is difficult to guarantee that a worker completes all the tasks in a batch she started on. In practice, there are simple ways to ensure this by, for instance, conditioning the payment on completing all the tasks in a batch.

A problem with restricting the number of tasks assigned to each worker (as we propose in Section 2.1) is that it might take a long time to have all the batches completed. Letting the workers choose how many tasks they want to complete allows a few eager workers to complete enormous amount of tasks. However, if we restrict the number of tasks assigned to each worker, we might need to recruit more workers to complete all the tasks. This problem of tasks taking long time to finish is not just restricted to our model, but is a very common problem in open crowdsourcing platforms. Ipeirotis [Ipe10] studied the completion time of tasks on Mechanical Turk and observed that it follows a heavy tail distribution according to a power law. Hence, for some tasks it takes significant amount of time to finish. A number of strategies have been proposed to complete tasks on time. This includes optimizing pricing policy [FHI11], continuously posting tasks to stay on the first page [BJJ⁺10, CHMA10], and having a large amount of tasks available [CHMA10]. These strategies are effective in attracting more workers fast. However, in our model, we assume there is no restrictions on the latency and we can wait until all the batches are completed, and if we have good strategies to reduce worker response time, such strategies could be incorporated into our system design.

Prior work. Previous crowdsourcing system designs have focused on developing inference algorithms assuming that the task assignments are fixed and the workers’ responses are already given.

³<http://turkopticon.differenceengines.com>

⁴<http://turkernation.com>

None of the prior work on crowdsourcing provides any systematic treatment of task assignment under the crowdsourcing model considered in this paper. To the best of our knowledge, we are the first to study both aspects of crowdsourcing together and, more importantly, establish optimality.

A naive approach to solve the inference problem, which is widely used in practice, is majority voting. Majority voting simply follows what the majority of workers agree on. When we have many spammers in the crowd, majority voting is error-prone since it gives the same weight to all the responses, regardless of whether they are from a spammer or a diligent workers. We will show in Section 2.3 that majority voting is provably sub-optimal and can be significantly improved upon.

If we know how reliable each worker is, then it is straightforward to find the maximum likelihood estimates: compute the weighted sum of the responses weighted by the log-likelihood. Although, in reality, we do not have this information, it is possible to learn about a worker’s reliability by comparing one worker’s answer to others’. This idea was first proposed by Dawid and Skene, who introduced an iterative algorithm based on expectation maximization (EM) [DS79]. They considered the problem of classifying patients based on labels obtained from multiple clinicians. They introduce a simple probabilistic model describing the clinicians’ responses, and gave an algorithmic solution based on EM. This model, which is described in Section 2.7, is commonly used in modern crowdsourcing settings to explain how workers make mistakes in classification tasks [SPI08].

This heuristic algorithm iterates the following two steps. In the M-step, the algorithm estimates the error probabilities of the workers that maximizes the likelihood using the current estimates of the answers. In the E-step, the algorithm estimates the likelihood of the answers using the current estimates of the error probabilities. More recently, a number of algorithms followed this EM approach based on a variety of probabilistic models [SFB⁺95, WRW⁺09, RYZ⁺10a]. The crowdsourcing model we consider in this paper is a special case of these models, and we discuss their relationship in Section 2.7. The EM approach has also been widely applied in classification problems, where a set of labels from low-cost noisy workers is used to find a good classifier [JG03, RYZ⁺10a]. Given a fixed budget, there is a trade-off between acquiring a larger training dataset or acquiring a smaller dataset but with more labels per data point. Through extensive experiments, Sheng, Provost and Ipeirotis [SPI08] show that getting repeated labeling can give considerable advantage.

Despite the popularity of the EM algorithms, the performance of these approaches are only empirically evaluated and there is no analysis that gives performance guarantees. In particular, EM algorithms are highly sensitive to the initialization used, making it difficult to predict the quality of the resulting estimate. Further, the role of the task assignment is not at all understood with the EM algorithm (or for that matter any other algorithm). We want to address both questions of task allocation and inference together, and devise an algorithmic solution that can achieve minimum error from a fixed budget on the total number of queries. When we have a given target accuracy, such an algorithm will achieve this target accuracy with minimum cost. Further, we want to provide a strong performance guarantee for this approach and show that it is close to a fundamental limit on what the best algorithm can achieve.

Contributions. In this work, we provide the first rigorous treatment of both aspects of designing a reliable crowdsourcing system: task allocation and inference. We provide both an order-optimal task allocation scheme (based on random graphs) and an order-optimal algorithm for inference (based on low-rank approximation and belief propagation) on that task assignment. We show that our algorithm, which is non-adaptive, performs as well (for the worst-case worker distribution) as

the optimal oracle estimator which can use any adaptive task allocation scheme.

Concretely, given a target probability of error ε and a crowd with collective quality q , we show that spending a budget which scales as $O((1/q) \log(1/\varepsilon))$ is sufficient to achieve probability of error less than ε using our approach. We give a task allocation scheme and an inference algorithm with runtime which is linear in the total number of queries (up to a logarithmic factor). Conversely, we also show that using the best adaptive task allocation scheme together with the best inference algorithm, and under the worst-case worker distribution, this scaling of the budget in terms of q and ε is unavoidable. No algorithm can achieve error less than ε with number of queries smaller than $(C/q) \log(1/\varepsilon)$ with some positive universal constant C . This establishes that our algorithm is worst-case optimal up to a constant factor in the required budget.

Our main results show that our non-adaptive algorithm is worst-case optimal and there is no significant gain in using an adaptive strategy. We attribute this limit of adaptation to the fact that, in existing platforms such as Amazon’s Mechanical Turk, the workers are fleeting and the system does not allow for exploiting good workers. Therefore, a positive message of this result is that a good rating system for workers is essential to truly benefit from crowdsourcing platforms using adaptivity.

Another novel contribution of our work is the analysis technique. The iterative inference algorithm we introduce operates on real-valued messages whose distribution is a priori difficult to analyze. To overcome this challenge, we develop a novel technique of establishing that these messages are sub-Gaussian and compute the parameters recursively in a closed form. This allows us to prove the sharp result on the error rate. This technique could be of independent interest in analyzing a more general class of message-passing algorithms.

2 Main result

To achieve a certain reliability in our answers with minimum number of queries, we propose using random regular graphs for task allocation and introduce a novel iterative algorithm to infer the correct answers. While our approach is *non-adaptive*, we show that it is sufficient to achieve an order-optimal performance when compared to the best possible approach using *adaptive* task allocations. Precisely, we prove an upper bound on the resulting error when using our approach and a matching lower bound on the minimax error rate achieved by the best possible adaptive task allocation together with an optimal inference algorithm. This shows that our approach is minimax optimal up to a constant factor: it requires only a constant factor times the minimum necessary budget to achieve a target error rate under the worst-case worker distribution. We then present the intuitions behind our inference algorithm through connections to low-rank matrix approximations and belief propagation.

2.1 Algorithm

Task allocation. We use a non-adaptive scheme which makes all the task assignments before any worker arrives. This amounts to designing a bipartite graph with one type of nodes corresponding to each of the tasks and another set of nodes corresponding to each of the batches. An edge (i, j) indicates that task i is included in batch T_j . Once all T_j ’s are determined according to the graph, these batches are submitted simultaneously to the crowdsourcing platform. Each arriving worker

will pick up one of the batches and complete all the tasks in that batch. We denote by j the worker working on j -th batch T_j .

To design a bipartite graph, the taskmaster first makes a choice of how many workers to assign to each task and how many tasks to assign to each worker. The task degree ℓ is typically determined by how much resources (e.g. money, time, etc.) one can spend on the tasks. The worker degree r is typically determined by how many tasks are manageable for a worker depending on the application. The total number of workers that we need is automatically determined as $n = m\ell/r$, since the total number of edges has to be consistent.

We will show that with such a regular graph, you can achieve probability of error which is quite close to a lower bound on what any inference algorithm can achieve with any task assignment. In particular, this includes all possible graphs which might have irregular degrees or have very large worker degrees (and small number of workers) conditioned on the total number of edges being the same. This suggests that, among other things, there is no significant gain in using an irregular graph.

We assume that the total cost that must be paid is proportional to the total number of edges and not the number of workers. If we have more budget we can increase ℓ . It is then natural to expect the probability of error to decrease, since we are collecting more responses. We will show that the error rate decreases exponentially in ℓ as ℓ grows. However, increasing r does not incur increase in the cost and it is not immediately clear how it affects the performance. We will show that with larger r we can learn more about the workers and the error rate decreases as r increases. However, how much we can gain by increasing the worker degree is limited.

Given the task and worker degrees, there are multiple ways to generate a regular bipartite graph. We want to choose a graph that will minimize the probability of error. Deviating slightly from regular degrees, we propose using a simple random construction known as *configuration model* in random graph literature [RU08, Bol01]. We start with $[m] \times [\ell]$ half-edges for task nodes and $[n] \times [r]$ half-edges for the worker nodes, and pair all the half-edges according to a random permutation of $[m\ell]$. The resulting graph might have multi-edges where two nodes are connected by more than one edges. However, they are very few in thus generated random graph as long as $\ell \ll n$, whence we also have $r \ll m$. Precisely, the number of double-edges in the graph converges in distribution to Poisson distribution with mean $(\ell - 1)(r - 1)/2$ [Bol01, Page 59 Exercise 2.12]. The only property that we need for the main result to hold is that the resulting random graph converges locally to a random tree in probability in the large system limit. This enables us to analyze the performance of our inference algorithm and provide sharp bounds on the probability of error.

The intuition behind why random graphs are good for our inference problem is related to the spectral gap of random matrices. In the following, we will use the (approximate) top singular vector of a weighted adjacency matrix of the random graph to find the correct labels. Since, sparse random graphs are excellent expanders with large spectral gaps, this enables us to reliably separate the low-rank structure from the data matrix which is perturbed by random noise.

Inference algorithm. We are given a task allocation graph $G([m] \cup [n], E)$ where we connect an edge (i, j) if a task i is assigned to a worker j . In the following, we will use indexes i for a i -th task node and j for a j -th worker node. We use ∂i to denote the neighborhood of node i . Each edge (i, j) on the graph G has a corresponding worker response A_{ij} .

To find the correct labels from the given responses of the workers, we introduce a novel iterative algorithm. This algorithm is inspired by the celebrated belief propagation algorithm and low-rank

matrix approximations. The connections are explained in detail in Section 2.5 and 2.6, along with mathematical justifications.

The algorithm operates on real-valued task messages $\{x_{i \rightarrow j}\}_{(i,j) \in E}$ and worker messages $\{y_{j \rightarrow i}\}_{(i,j) \in E}$. A task message $x_{i \rightarrow j}$ represents the log-likelihood of task i being a positive task, and a worker message $y_{j \rightarrow i}$ represents how reliable worker j is. We start with the worker messages initialized as independent Gaussian random variables, although the algorithm is not sensitive to a specific initialization as long as it has a strictly positive mean. We could also initialize all the messages to one, but then we need to add extra steps in the analysis to ensure that this is not a degenerate case. At k -th iteration, the messages are updated according to

$$x_{i \rightarrow j}^{(k)} = \sum_{j' \in \partial i \setminus j} A_{ij'} y_{j' \rightarrow i}^{(k-1)}, \quad \text{for all } (i, j) \in E, \text{ and} \quad (1)$$

$$y_{j \rightarrow i}^{(k)} = \sum_{i' \in \partial j \setminus i} A_{i'j} x_{i' \rightarrow j}^{(k)}, \quad \text{for all } (i, j) \in E, \quad (2)$$

where ∂i is the neighborhood of the task node i and ∂j is the neighborhood of the worker node j . At task update, we are giving more weight to the answers that came from more trustworthy workers. At worker update, we increase our confidence in that worker if the answers she gave on another task, $A_{i'j}$, has the same sign as what we believe, $x_{i' \rightarrow j}$. Intuitively, a worker message represents our belief on how ‘reliable’ the worker is. Hence, our final estimate is a weighted sum of the answers weighted by each worker’s reliability:

$$\hat{t}_i^{(k)} = \text{sign} \left(\sum_{j \in \partial i} A_{ij} y_{j \rightarrow i}^{(k-1)} \right).$$

Iterative Algorithm

Input: $E, \{A_{ij}\}_{(i,j) \in E}, k_{\max}$

Output: Estimate $\hat{t} \in \{\pm 1\}^m$

1: **For all** $(i, j) \in E$ **do**

Initialize $y_{j \rightarrow i}^{(0)}$ with random $Z_{ij} \sim \mathcal{N}(1, 1)$;

2: **For** $k = 1, \dots, k_{\max}$ **do**

For all $(i, j) \in E$ **do** $x_{i \rightarrow j}^{(k)} \leftarrow \sum_{j' \in \partial i \setminus j} A_{ij'} y_{j' \rightarrow i}^{(k-1)}$;

For all $(i, j) \in E$ **do** $y_{j \rightarrow i}^{(k)} \leftarrow \sum_{i' \in \partial j \setminus i} A_{i'j} x_{i' \rightarrow j}^{(k)}$;

3: **For all** $i \in [m]$ **do** $x_i \leftarrow \sum_{j \in \partial i} A_{ij} y_{j \rightarrow i}^{(k_{\max}-1)}$;

4: **Output** estimate vector $\hat{t}^{(k)} = \{\text{sign}(x_i)\}$.

While our algorithm is inspired by the standard Belief Propagation (BP) algorithm for approximating max-marginals [Pea88, YFW03], our algorithm is original and overcomes a few limitations of the standard BP for this inference problem under the crowdsourcing model. First, the iterative algorithm does not require any knowledge of the prior distribution of \mathbf{p}_j , whereas the standard BP requires it as explained in detail in Section 2.6. Second, the iterative algorithm is provably order-optimal for this crowdsourcing problem. We use a standard technique, known as *density evolution*, to analyze the performance of our message-passing algorithm. Although we can write down the density evolution equations for the standard BP for crowdsourcing, it is not trivial to describe or

compute the densities, analytically or numerically. It is also very simple to write down the density evolution equations (cf. (13) and (14)) for our algorithm, but it is not a priori clear how one can analyze the densities in this case either. We develop a novel technique to analyze the densities for our iterative algorithm and prove optimality. This technique could be of independent interest to analyzing a broader class of message-passing algorithms.

2.2 Performance guarantee and experimental results

We provide an upper bound on the probability of error achieved by the iterative inference algorithm and task allocation according to the configuration model. The bound decays as $e^{-C\ell q}$ with a universal constant C . Further, an algorithm-independent lower bound that we establish suggests that such a dependence of the error on ℓq is unavoidable.

2.2.1 Bound on the average error probability

To lighten the notation, let $\hat{\ell} \equiv \ell - 1$ and $\hat{r} \equiv r - 1$, and recall that $q = \mathbb{E}[(2\mathbf{p}_j - 1)^2]$. Using these notations, we define σ_k^2 to be the effective variance in the sub-Gaussian tail of our estimates after k iterations of our inference algorithm:

$$\sigma_k^2 \equiv \frac{2q}{\mu^2(q^2\hat{\ell}\hat{r})^{k-1}} + \left(3 + \frac{1}{q\hat{r}}\right) \frac{1 - (1/q^2\hat{\ell}\hat{r})^{k-1}}{1 - (1/q^2\hat{\ell}\hat{r})}.$$

With this, we can prove the following upper bound on the probability of error when we run k iterations of our inference algorithm with (ℓ, r) -regular assignments on m tasks using a crowd with collective quality q . We refer to Section 3.1 for the proof.

Theorem 2.1. *For fixed $\ell > 1$ and $r > 1$, assume that m tasks are assigned to $n = m\ell/r$ workers according to a random (ℓ, r) -regular graph drawn from the configuration model. If the distribution of the worker reliability satisfies $\mu \equiv \mathbb{E}[2\mathbf{p}_j - 1] > 0$ and $q^2 > 1/(\hat{\ell}\hat{r})$, then for any $t \in \{\pm 1\}^m$, the estimate after k iterations of the iterative algorithm achieves*

$$\frac{1}{m} \sum_{i=1}^m \mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leq e^{-\ell q/(2\sigma_k^2)} + \frac{3\ell r}{m} (\hat{\ell}\hat{r})^{2k-2}. \quad (3)$$

The second term, which is the probability that the resulting graph is not locally tree-like, vanishes for large m . Hence, the dominant term in the error bound is the first term. Further, when $q^2\hat{\ell}\hat{r} > 1$ as per our assumption and when we run our algorithm for large enough number of iterations, σ_k^2 converges linearly to a finite limit $\sigma_\infty^2 \equiv \lim_{k \rightarrow \infty} \sigma_k^2$ such that

$$\sigma_\infty^2 = \left(3 + \frac{1}{q\hat{r}}\right) \frac{q^2\hat{\ell}\hat{r}}{q^2\hat{\ell}\hat{r} - 1}.$$

With linear convergence of σ_k^2 , we only need a small number of iterations to achieve σ_k close to this limit. It follows that for large enough m and k , we can prove an upper bound that does not depend on the problem size or the number of iterations, which is stated in the following corollary.

Corollary 2.2. *Under the hypotheses of Theorem 2.1, there exists $m_0 = 3\ell r e^{\ell q/4\sigma_\infty^2} (\hat{\ell}\hat{r})^{2(k-1)}$ and $k_0 = 1 + (\log(q/\mu^2)/\log(\hat{\ell}\hat{r}q^2))$ such that*

$$\frac{1}{m} \sum_{i=1}^m \mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leq 2e^{-\ell q/(4\sigma_\infty^2)}, \quad (4)$$

for all $m \geq m_0$ and $k \geq k_0$.

Proof. For $\hat{\ell}\hat{r}q^2 > 1$ as per our assumption, $k = 1 + \log(q/\mu^2)/\log(\hat{\ell}\hat{r}q^2)$ iterations suffice to ensure that $\sigma_k^2 \leq (2q/\mu^2)(\hat{\ell}\hat{r}q^2)^{-k+1} + q\hat{\ell}(3q\hat{r} + 1)/(q^2\hat{\ell}\hat{r} - 1) \leq 2\sigma_\infty^2$. Also, $m = 3\ell r e^{\ell q/4\sigma_\infty^2} (\hat{\ell}\hat{r})^{2(k-1)}$ suffices to ensure that $(\hat{\ell}\hat{r})^{2k-2}(3\ell r)/m \leq \exp\{-\ell q/(4\sigma_\infty^2)\}$. \square

The required number of iterations k_0 is small (only logarithmic in ℓ , r , q , and μ) and does not depend on the problem size m . On the other hand, the required number of tasks m_0 in our main theorem is quite large. However, numerical simulations suggest that the actual performance of our approach is not very sensitive to the number of tasks and the bound still holds for tasks of small size as well. For example, in Figure 1 (left), we ran numerical experiment with $m = 1000$, $q = 0.3$, and $k = 20$, and the resulting error exhibits exponential decay as predicted by our theorem even for large $\ell = r = 30$. In this case, theoretical requirement on the number of tasks m_0 is much larger than what we used in the experiment.

Consider a set of worker distributions $\{\mathcal{F} \mid \mathbb{E}_{\mathcal{F}}[(2\mathbf{p} - 1)^2] = q\}$ that have the same collective quality q . These distributions that have the same value of q can give different values for μ ranging from q to $q^{1/2}$. Our main result on the error rate suggests that the error does not depend on the value of μ . Hence, the effective second moment q is the right measure of the collective quality of the crowd, and the effective first moment μ only affects how fast the algorithm converges, since we need to run our inference algorithm $k = \Omega(1 + \log(q/\mu^2)/\log(\hat{\ell}\hat{r}q^2))$ iterations to guarantee the error bound.

The iterative algorithm is efficient with run-time comparable to that of majority voting which requires $O(m\ell)$ operations. Each iteration of the iterative algorithm requires $O(m\ell)$ operations, and we need $O(1 + \log(q/\mu^2)/\log(q^2\hat{\ell}\hat{r}))$ iterations to ensure an error bound in (4). By definition, we have $q \leq \mu \leq \sqrt{q}$. The run-time is the worst when $\mu = q$, which happens under the spammer-hammer model, and it is the smallest when $\mu = \sqrt{q}$ which happens if $\mathbf{p}_j = (1 + \sqrt{q})/2$ deterministically. In any case, we only need extra logarithmic factor that does not increase with compared to majority voting, and this Notice that as we increase the number of iterations, the messages converge to an eigenvector of a particular sparse matrix of dimensions $m\ell \times m\ell$. This suggests that we can alternatively compute the messages using other algorithms for computing the top singular vector of large sparse matrices that are known to converge faster (e.g. Lanczos algorithm [Lan50]).

Next, we make a few remarks on the performance guarantee.

First, the assumption that $\mu > 0$ is necessary. If there is no assumption on μ , then we cannot distinguish if the responses came from tasks with $\{t_i\}_{i \in [m]}$ and workers with $\{p_j\}_{j \in [n]}$ or tasks with $\{-t_i\}_{i \in [m]}$ and workers with $\{1 - p_j\}_{j \in [n]}$. Statistically, both of them give the same output. The hypothesis on μ allows us to distinguish which of the two is the correct solution. In the case when we know that $\mu < 0$, we can use the same algorithm changing the sign of the final output and get the same performance guarantee.

Second, our algorithm does not require any information on the distribution of \mathbf{p}_j . However, in order to generate a graph that achieves an optimal performance, we need the knowledge of q for

selecting the degree $\ell = \Theta(1/q \log(1/\varepsilon))$. Here is a simple way to overcome this limitation at the loss of only additional constant factor, i.e. scaling of cost per task still remains $\Theta(1/q \log(1/\varepsilon))$. To that end, consider an incremental design in which at step a the system is designed assuming $q = 2^{-a}$ for $a \geq 1$. At step a , we design two replicas of the task allocation for $q = 2^{-a}$. Now compare the estimates obtained by these two replicas for all m tasks. If they agree amongst $m(1-2\varepsilon)$ tasks, then we stop and declare that as the final answer. Otherwise, we increase a to $a + 1$ and repeat. Note that by our optimality result, it follows that if 2^{-a} is less than the actual q then the iteration must stop with high probability. Therefore, the total cost paid is $\Theta(1/q \log(1/\varepsilon))$ with high probability. Thus, even lack of knowledge of q does not affect the order optimality of our algorithm.

Further, unlike previous approaches based on Expectation Maximization (EM), the iterative algorithm is not sensitive to initialization and converges to a unique solution from a random initialization with high probability. This follows from the fact that the algorithm is essentially computing a leading eigenvector of a particular linear operator.

Finally, we observe a phase transition at $\hat{\ell} \hat{r} q^2 = 1$. Above this phase transition, when $\hat{\ell} \hat{r} q^2 > 1$, we will show that our algorithm is order-optimal and the probability of error is significantly smaller than majority voting. However, perhaps surprisingly, when we are below the threshold, when $\hat{\ell} \hat{r} q^2 < 1$, we empirically observe that our algorithm exhibit a fundamentally different behavior (cf. Figure 1). The error we get after k iterations of our algorithm increases with k . In this regime, we are better off stopping the algorithm after 1 iteration, in which case the estimate we get is essentially the same as the simple majority voting, and we cannot do better than majority voting. This phase transition is universal and we observe similar behavior with other inference algorithms including EM approaches. We provide more discussions on the choice of ℓ and the limitations of having small r in the following section.

2.2.2 Minimax optimality of our approach

For a task master, the natural core optimization problem of her concern is how to achieve a certain reliability in the answers with minimum cost. Throughout this paper, we assume that the cost is proportional to the total number of queries. In this section, we show that if a taskmaster wants to achieve a target error rate of ε , she can do so using our approach with budget per task scaling as $O((1/q) \log(1/\varepsilon))$ for a broad range of worker degree r . Compared to the necessary condition which we provide in Section 2.3, this is within a constant factor from what is necessary using the best *non-adaptive* task assignment and the best inference algorithm. Further, we show in Section 2.4 that this scaling in the budget is still necessary if we allow using the best *adaptive* task assignment together with the best inference algorithm. This proves that our approach is minimax optimal up to a constant factor in the budget.

Assuming for now that there is no restrictions on the worker degree r and we can assign as many tasks to each worker as we want, we can get the following simplified upper bound on the error that holds for all $r \geq 1 + 1/q$. To simplify the resulting bound, let us assume for now that $\hat{\ell} \hat{r} q \geq 2$. Then, we get that $\sigma_\infty^2 \leq 2(3 + 1/\hat{r}q)$. Then from (4), we get the following bound:

$$\frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leq 2e^{-\ell q/32} ,$$

for large enough $m \geq m_0$. In terms of the budget or the number of queries necessary to achieve a target accuracy, we get the following sufficient condition as a corollary.

Corollary 2.3. *Using the non-adaptive task assignment scheme with $r \geq 1 + 1/q$ and the iterative inference algorithm introduced in Section 2.1, it is sufficient to query $(32/q) \log(2/\varepsilon)$ times per task to guarantee that the probability of error is at most ε for any $\varepsilon \leq 1/2$ and for all $m \geq m_0$.*

We provide a matching minimax necessary condition up to a constant factor for non-adaptive algorithms in Section 2.3. When the nature can choose the worst-case worker distributions, no non-adaptive algorithm can achieve error less than ε with budget per task smaller than $(C'/q) \log(1/2\varepsilon)$ with some universal positive constant C' . This establishes that under the non-adaptive scenario, our approach is minimax optimal up to a constant factor for large enough m . With our approach you only need to ask (and pay for) a constant factor more than what is necessary using the best non-adaptive task assignment scheme together with the best inference algorithm under the worst-case worker distribution.

Perhaps surprisingly, we will show in Section 2.4 that the necessary condition does not change even if we allow adaptive task assignments. No algorithm, adaptive or non-adaptive, can achieve error less than ε without asking $(C''/q) \log(1/2\varepsilon)$ queries per task with some universal positive constant C'' . Hence, our non-adaptive approach achieves minimax optimal performance that can be achieved by the best adaptive scheme.

In practice, we might not be allowed to have large r depending on the application. For different regimes of the restrictions on the allowed worker degree r , we need different choices of ℓ . When we have a target accuracy ε , the following corollary establishes that we can achieve probability of error ε with $\ell \geq C(1 + 1/\hat{r}q)(1/q) \log(1/\varepsilon)$ for any value of r .

Corollary 2.4. *Using the non-adaptive task assignment scheme with any r and the iterative inference algorithm introduced in Section 2.1, it is sufficient to query $(24 + 8/\hat{r}q)(1/q) \log(2/\varepsilon)$ times per task to guarantee that the probability of error is at most ε for any $\varepsilon \leq 1/2$ and for all $m \geq m_0$.*

Proof. We will show that for $\ell \geq \max\{1 + 2/(\hat{r}q^2), 8(3 + 1/\hat{r}q)(1/q) \log(1/\varepsilon)\}$, the probability of error is at most ε . Since, $1 + 2/(\hat{r}q^2) \leq 8(3 + 1/\hat{r}q)(1/q) \log(1/\varepsilon)$ for $\varepsilon \leq 1/2$, this proves the corollary. Since $\hat{r}q^2 \geq 2$ from the first condition, we get that $\sigma_\infty^2 \leq 2(3 + 1/\hat{r}q)$. Then, the probability of error is upper bounded by $2 \exp\{-\ell q/(24 + 8/\hat{r}q)\}$. This implies that for $\ell \geq (24 + 8/\hat{r}q)(1/q) \log(2/\varepsilon)$ the probability of error is at most ε . \square

For $r \geq C'/q$, this implies that our approach requires $O((1/q) \log(1/\varepsilon))$ queries and it is minimax optimal. However, for $r = O(1)$, our approach requires $O((1/q^2) \log(1/\varepsilon))$ queries. This is due to the fact that when r is small, we cannot efficiently learn the quality of the workers and need significantly more questions to achieve the accuracy we desire. Hence, in practice, we want to be able to assign more tasks to each worker when we have low-quality workers.

2.2.3 Experimental results

Figure. 1 shows the comparisons between probabilities of error achieved by different inference algorithms, but on the same task assignment using regular bipartite random graphs. We ran 20 iterations of EM and our iterative algorithm, and also the spectral approach of using leading left singular vector of A for estimation. The spectral approach, which we call Singular Vector in the graph, is explained in detail in Section 2.5. The error rates are compared with those of majority voting and the oracle estimator. The oracle estimator performance sets a lower bound on what any inference algorithm can achieve, since it knows all the values of p_j 's. For the numerical simulation

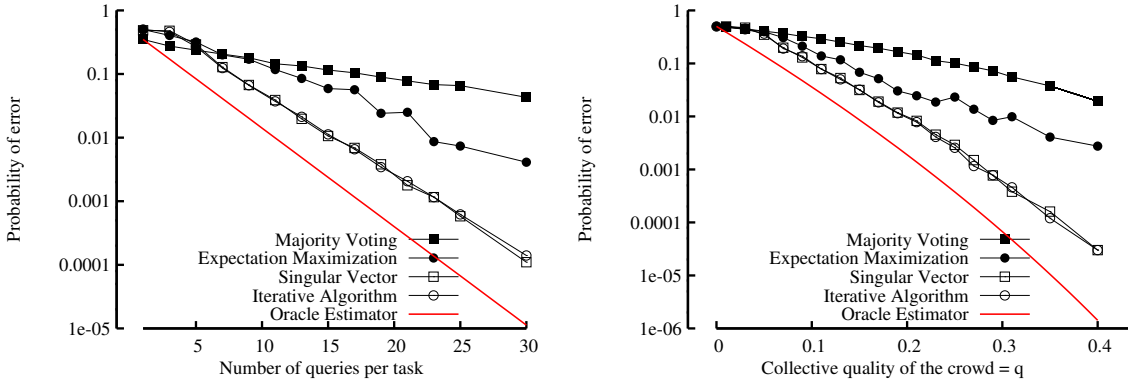


Figure 1: The iterative algorithm improves over majority voting and EM algorithm. Using the top singular vector for inference has similar performance as our iterative approach.

on the left-hand side, we set $m = 1000$, $\ell = r$ and used the spammer hammer model for the distribution of the workers with $q = 0.3$. According to our theorem, we expect a phase transition at $\ell = 1 + 1/0.3 = 4.3333$. From the figure, we observe that the iterative inference algorithm starts to perform better than majority voting at $\ell = 5$. For the figure on the right-hand side, we set $\ell = 25$. For fair comparisons with the EM approach, we used an implementation of the EM approach in Java by Sheng et al. [SPI08], which is publicly available.

We also ran two experiments with real crowd using Amazon Mechanical Turk. In our experiments, we created tasks for comparing colors; we showed three colors on each task, one on the top and two on the bottom. We asked the crowd to indicate “if the color on the top is more similar to the color on the left or on the right”.

The first experiment confirms that the ground truth for these color comparisons tasks are what is expected from pairwise distances in the Lab color space. The distances in the Lab color space between the a pair of colors are known to be a good measure of the perceived distance between the pair [WS67]. To check the validity of this Lab distance we collected 210 responses on each of the 10 color comparison tasks. As shown in Figure. 2, for all 10 tasks, the majority of the 210 responses were consistent with the Lab distance based ground truth.

Next, to test our approach, we created 50 of such similarity tasks and recruited 28 workers to answer all the questions. Once we have this data, we can subsample the data to simulate what would have happened if we collected smaller number of responses per task. The resulting average probability of error is illustrated in Figure. 3. For this crowd from Amazon Mechanical Turk, we can estimate the collective quality from the data, which is about $q \simeq 0.175$. Theoretically, this indicates that phase transition should happen when $(\ell - 1)((50/28)\ell - 1)q^2 = 1$, since we set $r = (50/28)\ell$. With this, we expect phase transition to happen around $\ell \simeq 5$. In Figure. 3, we see that our iterative algorithm starts to perform better than majority voting around $\ell = 8$.

2.3 Fundamental limit under the non-adaptive scenario

Under the non-adaptive scenario, we are allowed to use only non-adaptive task assignment schemes which assign all the tasks a priori and collect all the responses simultaneously. In this section, we

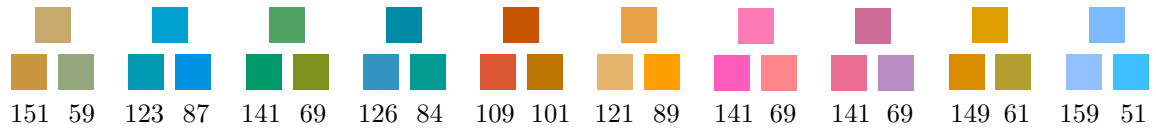


Figure 2: Experimental results on color comparison using real data from Amazon’s Mechanical Turk. The color on the left is closer to the one on the top in Lab distance for each triplet. The votes from 210 workers are shown below each triplet.

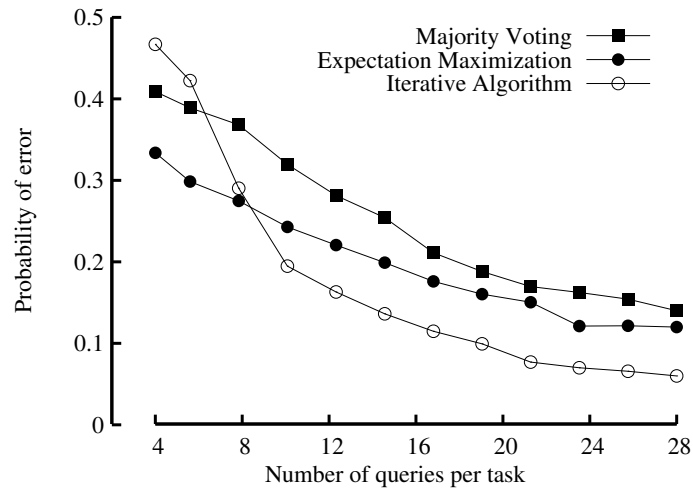


Figure 3: The average probability of error on color comparisons using real data from Amazon’s Mechanical Turk.

investigate the fundamental limit on how small an error can be achieved using the best possible non-adaptive task assignment scheme together with the best possible inference algorithm. In particular, we are interested in the minimax optimality: What is the minimum error that can be achieved under the worst-case worker distribution? To this end, we analyze the performance of an oracle estimator when the workers' latent qualities are drawn from a specific distribution and provide a lower bound on the minimax rate on the probability of error. Compared to our main result, this establishes that our approach is minimax optimal up to a constant factor.

In terms of the budget, the natural core optimization problem of our concern is how to achieve a certain reliability in our answers with minimum cost. Let us assume that the cost is proportional to the total number of queries. We show that for a given target error rate ε , the total budget sufficient to achieve this target error rate using our algorithm is within a constant factor from what is necessary using the best non-adaptive task assignment and the best inference algorithm.

Fundamental limit. Consider a crowd characterized by worker distribution \mathcal{F} such that $\mathbf{p}_j \sim \mathcal{F}$. Let \mathcal{F}_q be a set of all distributions on $[0, 1]$, such that the collective quality is parametrized by q :

$$\mathcal{F}_q = \{ \mathcal{F} \mid \mathbb{E}_{\mathcal{F}}[(2\mathbf{p}_j - 1)^2] = q \} .$$

We want to prove a lower bound on the *minimax rate* on the probability of error, which only depends on q and ℓ . Define the minimax rate as

$$\min_{\tau \in \mathcal{T}_\ell, \hat{t}} \max_{t \in \{\pm 1\}^m, \mathcal{F} \in \mathcal{F}_q} \frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i) ,$$

where \hat{t} ranges over all estimators which are measurable functions over the responses, and τ ranges over the set \mathcal{T}_ℓ of all task assignment schemes which are non-adaptive and ask $m\ell$ queries in total. Here the probability is taken over all realizations of \mathbf{p}_j 's, \mathbf{A}_{ij} 's, and the randomness introduced in the task assignment and the inference.

Consider any non-adaptive scheme that assigns ℓ_i workers to the i -th task. The only constraint is that the average number of queries is bounded by $(1/m) \sum_{i \in [m]} \ell_i \leq \ell$. To get a lower bound on the minimum achievable error, we consider an oracle estimator that has access to all the \mathbf{p}_j 's, and hence can make an optimal estimation. Further, since we are proving minimax optimality and not instance-optimality, the worst-case error rate will always be lower bounded by the error rate for any choice of worker distribution. In particular, we prove a lower bound using the spammer-hammer model. Concretely, we assume the \mathbf{p}_j 's are drawn from the spammer-hammer model with perfect hammers:

$$\mathbf{p}_j = \begin{cases} 1/2 & \text{with probability } 1 - q , \\ 1 & \text{otherwise .} \end{cases}$$

Notice that the use of q is consistent with $\mathbb{E}[(2\mathbf{p}_j - 1)^2] = q$. Under the spammer-hammer model, the oracle estimator only makes a mistake on task i if it is only assigned to spammers, in which case we flip a fair coin to achieve error probability of half. Formally,

$$\mathbb{P}(\hat{t}_i \neq t_i) = \frac{1}{2}(1 - q)^{\ell_i} .$$

By convexity and using Jensen's inequality, the average probability of error is lower bounded by

$$\frac{1}{m} \sum_{i \in [m]} \mathbb{P}(\hat{t}_i \neq t_i) \geq \frac{1}{2}(1 - q)^\ell .$$

Since we are interested in how many more queries are necessary as the quality of the crowd deteriorates, we are going to assume $q \leq 2/3$, in which case $(1 - q) \geq e^{-(q+q^2)}$. As long as total $m\ell$ queries are used, this lower bound holds regardless of how the actual tasks are assigned. And since this lower bound holds for a particular choice of \mathcal{F} , it holds for the worst case \mathcal{F} as well. Hence, for the best task assignment scheme and the best inference algorithm, we have

$$\min_{\tau \in \mathcal{T}_\ell, \hat{t}} \max_{t \in \{\pm 1\}^m, \mathcal{F} \in \mathcal{F}_q} \frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i) \geq \frac{1}{2} e^{-(q+q^2)\ell}.$$

This lower bound on the minimax rate holds for any positive integer m , and regardless of the number of workers or the number of queries, r , assigned to each worker. In terms of the average number of queries necessary to achieve a target accuracy of ε , this implies the following necessary condition.

Lemma 2.5. *Assuming $q \leq 2/3$ and the non-adaptive scenario, if the average number of queries per task is less than $(1/2q) \log(1/2\varepsilon)$, then no algorithm can achieve average probability of error less than ε for any m under the worst-case worker distribution.*

To prove this worst-cased bound, we analyzed a specific distribution of the spammer-hammer model. However, the result (up to a constant factor) seems to be quite general and can also be proved using different distributions, e.g. when all workers have the same quality. The assumption on q can be relaxed as much as we want, by increasing the constant in the necessary budget. Compared to the sufficient condition in Corollary 2.3 this establishes that our approach is minimax optimal up to a constant factor. With our approach you only need to ask (and pay for) a constant factor more than what is necessary for any algorithm.

Majority voting. As a comparison, we can do similar analysis for the simple majority voting and show that the performance is significantly worse than our approach. The next lemma provides a bound on the minimax rate of majority voting. A proof of this lemma is provided in Section 3.4.

Lemma 2.6. *For any $C < 1$, there exists a positive constant C' such that when $q \leq C$, the error achieved by majority voting is at least*

$$\min_{\tau \in \mathcal{T}_\ell} \max_{t \in \{\pm 1\}^m, \mathcal{F} \in \mathcal{F}_q} \frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i) \geq e^{-C'(\ell q^2 + 1)}.$$

In terms of the number of queries necessary to achieve a target accuracy ε using majority voting, this implies that we need to ask at least $(c/q^2) \log(c'/\varepsilon)$ queries per task for some universal constants c and c' . Hence, majority voting is significantly more costly than our approach in terms of budget. Our algorithm is more efficient in terms of computational complexity as well. Simple majority voting requires $O((m/q^2) \log(1/\varepsilon))$ operations to achieve target error rate ε in the worst case. From Corollary 2.2, together with $\ell = O((1/q) \log(1/\varepsilon))$ and $\ell r q^2 = \Omega(1)$, we get that our approach requires $O((m/q) \log(1/q) \log(1/\varepsilon))$ operations in the worst case.

2.4 Fundamental limit under the adaptive scenario

In terms of the scaling of the budget necessary to achieve a target accuracy, we established that using a *non-adaptive* task assignment, no algorithm can do better than our approach. One might

prefer a non-adaptive scheme in practice because having all the batches of tasks processed in parallel reduces the latency. This is crucial in many applications, especially in real-time applications such as searching, visual information processing, and document processing [BJJ⁺10, BLM⁺10, YKG10, BBMK11]. However, by switching to an adaptive task assignment, one might hope to be more efficient and still obtain a desired accuracy from fewer questions. On one hand, adaptation can help improve performance. But on the other hand, it can significantly complicate system design due to careful synchronization requirements. In this section, we want to prove an algorithm-independent upper bound on how much one can gain by using an adaptive task allocation.

When the identities of the workers are known, one might be tempted to first identify which workers are more reliable and then assign all the tasks to those workers in an explore/exploit manner. However, in typical crowdsourcing platforms such as Amazon Mechanical Turk, it is unrealistic to assume that we can identify and reuse any particular worker, since typical workers are neither persistent nor identifiable and batches are distributed through an open-call. Hence, exploiting a reliable worker is not possible. However, we can adaptively resubmit batches of tasks; we can dynamically choose which subset of tasks to assign to the next arriving worker. In particular, we can allocate tasks to the next batch based on all the information we have on all the tasks from the responses collected thus far. For example, one might hope to reduce uncertainty more efficiently by adaptively collecting more responses on those tasks that she is less certain about.

Fundamental limit. In this section, we show that, perhaps surprisingly, there is no significant gain in switching from our non-adaptive approach to an adaptive strategy when the workers are *fleeting*. We first prove a lower bound on the minimax error rate: the error that is achieved by the best inference algorithm \hat{t} using the best adaptive task allocation scheme τ under a worst-case worker distribution \mathcal{F} and the worst-case true answers t . Let $\tilde{\mathcal{T}}_\ell$ be the set of all task assignment schemes that use at most $m\ell$ queries in total. Then, we can show the following lower bound on the minimax rate on the probability of error. A proof of this theorem is provided in Section 3.5.

Theorem 2.7. *When $q \leq C$ for any constant $C < 1$, there exists a positive constant C' such that*

$$\min_{\tau \in \tilde{\mathcal{T}}_\ell, \hat{t}} \max_{t \in \{\pm 1\}^m, \mathcal{F} \in \mathcal{F}_q} \frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i) \geq \frac{1}{2} e^{-C' \ell q}, \quad (5)$$

for all m where the task assignment scheme τ ranges over all adaptive schemes that use at most $m\ell$ queries and \hat{t} ranges over all estimators that are measurable functions over the responses.

We cannot avoid the factor of half in the lower bound, since we can always achieve error probability of half without asking any queries (with $\ell = 0$). In terms of the budget required to achieve a target accuracy, the above lower bound proves that no algorithm, adaptive or non-adaptive, can achieve an error rate less than ε with number of queries per task less than $(C'/q) \log(2/\varepsilon)$ in the worst case of worker distribution.

Corollary 2.8. *Assuming $q \leq C$ for any constant $C < 1$ and the iterative scenario, there exists a positive constant C' such that if the average number of queries is less than $(C'/q) \log(1/2\varepsilon)$, then no algorithm can achieve average probability of error less than ε for any m under the worst-case worker distribution.*

Compared to Corollary 2.3, we have a matching sufficient and necessary conditions up to a constant factor. This proves that there is no significant gain in using an adaptive scheme, and our

approach achieves minimax-optimality up to a constant factor with a non-adaptive scheme. This limitation of adaptation strongly relies on the fact that workers are *fleeting* in existing platforms and can not be reused. Therefore, architecturally our results suggest that building a reliable reputation system for workers would be essential to harnessing the potential of adaptive designs.

A counter example for instance-optimality. The above corollary establishes minimax-optimality: for the worst-case worker distribution, no algorithm can improve over our approach other than improving the constant factor in the necessary budget. However, this does not imply instance-optimality. In fact, there exists a family of worker distributions where all non-adaptive algorithms fail to achieve order-optimal performance whereas a trivial adaptive algorithm succeeds. Hence, for particular instances of worker distributions, there exists a gap between what can be achieved using non-adaptive algorithms and adaptive ones.

We will prove this in the case of the spammer-hammer model where each new worker is a hammer ($p_j = 1$) with probability q or a spammer ($p_j = 1/2$) otherwise. We showed in Section 2.3 that no non-adaptive algorithm can achieve an error less than $(1/2)e^{-C'\ell q}$ for any value of m . In particular, this does not vanish even if we increase m . We will introduce a simple adaptive algorithm and show that this algorithm achieves an error probability that goes to zero as m grows.

The algorithm first groups all the tasks into \sqrt{m} disjoint sets of size \sqrt{m} each. Starting with the first group, the algorithm assigns all \sqrt{m} tasks to new arriving workers until it sees two workers who agreed on all \sqrt{m} tasks. It declares those responses as its estimate for this group and moves on to the next group. This process is repeated until it reaches the allowed number of queries. This estimator makes a mistake on a group if (a) there were two spammers who agreed on all \sqrt{m} tasks or (b) we run out of allowed number of queries before we finish the last group. Formally, we can prove the following upper bound on the probability of error.

Lemma 2.9. *Under the spammer-hammer model, when the allowed number of queries per task ℓ is larger than $2/q$, there is an adaptive task allocation scheme and an inference algorithm that achieves average probability of error at most $m\ell^2 2^{-\sqrt{m}} + e^{-(2/\ell)(\ell q - 2)^2 \sqrt{m}}$.*

Proof. Recall that we are only allowed ℓm queries. Since we are allocating \sqrt{m} queries per worker, we can only ask at most $\ell\sqrt{m}$ workers. First, the probability that there is at least one pair of spammers (among all possible pairs from $\ell\sqrt{m}$ workers) who agreed on all \sqrt{m} responses is at most $m\ell^2 2^{-\sqrt{m}}$. Next, given that no pairs of spammers agreed on all their responses, the probability that we run out of all $m\ell$ allowed queries is the probability that the number of hammers in $\ell\sqrt{m}$ workers is strictly less than $2\sqrt{m}$ (which is the number of hammers we need in order to terminate the algorithm, conditioned on that no spammers agree with one another). By standard concentration results, this happens with probability at most $e^{-(2/\ell)(\ell q - 2)^2 \sqrt{m}}$. \square

This proves the existence of an adaptive algorithm which achieves vanishing error probability as m grows for a board range of task degree ℓ . Comparing the above upper bound with the known lower bound for non-adaptive schemes, this proves that non-adaptive algorithms cannot be instance optimal: there is a family of distributions where adaptation can significantly improve performance. This is generally true when there is a strictly positive probability that a worker is a hammer ($p_j = 1$).

One might be tempted to apply the above algorithm in more general settings other than the spammer-hammer model. However, this algorithm fails when there are no perfect workers in the

crowd. If we apply this algorithm in such a general setting, then it produces useless answers: the probability of error approaches half as m grows for any finite ℓ .

2.5 Connections to low-rank matrix approximation

In this section, we first explain why the top singular vector of the data matrix \mathbf{A} reveals the true answers of the tasks, where \mathbf{A} is the $m \times n$ matrix of the responses and we fill in zeros wherever we have no responses collected. This naturally defines a spectral algorithm for inference which we present next. It was proven in [KOS11] that the error achieved by this spectral algorithm is upper bounded by $C/(\ell q)$ with some constant C . But numerical experiments (cf. Figure 1) suggest that the error decays much faster, and that the gap is due to the weakness of the analysis used in [KOS11]. Inspired by this spectral approach, we introduced a novel inference algorithm that performs as well as the spectral algorithm (cf. Figure 1) and proved a much tighter upper bound on the resulting error which scales as $e^{-C'\ell q}$ with some constant C' . Our inference algorithm is based on *power iteration*, which is a well-known algorithm for computing the top singular vector of a matrix, and Figure 1 suggests that both algorithms are equally effective and the resulting errors are almost identical.

The data matrix \mathbf{A} can be viewed as a rank-1 matrix that is perturbed by random noise. Since, $\mathbb{E}[\mathbf{A}_{ij}|t_i, \mathbf{p}_j] = (r/m)t_i(2\mathbf{p}_j - \mathbf{1})$, the conditional expectation of this matrix is

$$\mathbb{E}[\mathbf{A} | t, \mathbf{p}] = \left(\frac{r}{m}\right)t(2\mathbf{p} - \mathbf{1})^T,$$

where $\mathbf{1}$ is the all ones vector, the vector of correct solutions is $t = \{t_i\}_{i \in [m]}$ and the vector of worker reliability is $\mathbf{p} = \{\mathbf{p}_j\}_{j \in [n]}$. Notice that the rank of this conditional expectation matrix is one and this matrix reveals the correct solutions exactly. We can decompose \mathbf{A} into a low-rank expectation plus a random perturbation:

$$\mathbf{A} = \left(\frac{r}{m}\right)t(2\mathbf{p} - \mathbf{1})^T + \mathbf{Z},$$

where $\mathbf{Z} \equiv \mathbf{A} - \mathbb{E}[\mathbf{A} | t, \mathbf{p}]$ is the random perturbation with zero mean. When the spectral radius of the noise matrix \mathbf{Z} is much smaller than the spectral radius of the signal, we can correctly extract most of t using the leading left singular vector of \mathbf{A} .

Under the crowdsourcing model considered in this paper, an inference algorithm using the top left singular vector of \mathbf{A} was introduced and analyzed by Karger et al. [KOS11]. Let u be the top left singular vector of \mathbf{A} . They proposed estimating $\hat{t}_i = \text{sign}(u_i)$ and proved an upper bound on the probability of error that scales as $O(1/\ell q)$. The main technique behind this result is in analyzing the spectral gap of \mathbf{A} . It is not difficult to see that the spectral radius of the conditional expectation matrix is $(r/m)\|t(2\mathbf{p} - \mathbf{1})^T\|_2 = \sqrt{\ell r q}$, where the operator norm of a matrix is denoted by $\|X\|_2 \equiv \max_a \|Xa\|/\|a\|$. Karger et al. proved that the spectral radius of the perturbation $\|\mathbf{Z}\|_2$ is in the order of $(\ell r)^{1/4}$. Hence, when $\ell r q^2 \gg 1$, we expect a separation between the conditional expectation and the noise.

One way to compute the leading singular vector is to use power iteration: for two vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, starting with a randomly initialized v , power iteration iteratively updates u and v by repeating $u = Av$ and $v = A^T u$. It is known that normalized u (and v) converges linearly to the leading left (and right) singular vector. Then we can use the sign of u_i to estimate t_i . Writing the

update rule for each entry, we get

$$u_i = \sum_{j \in \partial i} A_{ij} v_j, \quad v_j = \sum_{i \in \partial j} A_{ij} u_i .$$

Notice that this power iteration update rule is almost identical to those of message passing updates in (1) and (2). The ℓ task messages $\{x_{i \rightarrow j}\}_{j \in \partial i}$ from task i are close in value to the entry u_i of the power iteration. The r worker messages $\{y_{j \rightarrow i}\}_{i \in \partial j}$ from worker j are close in value to the entry v_j of the power iteration. Numerical simulations in Figure 1 suggest that the quality of the estimates from the two algorithms are almost identical. However, the known performance guarantee for the spectral approach is weak. We developed novel analysis techniques to analyze our message passing algorithm, and provide an upper bound on the error that scales as $e^{-C\ell q}$. It might be possible to apply our algorithm, together with the analysis techniques, to other problems where the top singular vector of a data matrix is used for inference.

2.6 Connections to belief propagation

The crowdsourcing model described in this paper can naturally be described using a graphical model. Let $G([m] \times [n], E, A)$ denote the weighted bipartite graph, where $[m]$ is the set of m task nodes, $[n]$ is the set of n worker nodes, E is the set of edges connecting a task to a worker who is assigned that task, and A is the set of weights on those edges according to the responses. Given such a graph, we want to find a set of task answers that maximize the following posterior distribution $F(\hat{t}, p) : \{\pm 1\}^m \times [0, 1]^n \rightarrow \mathbb{R}^+$.

$$\max_{\hat{t}, p} \prod_{a \in [n]} \mathcal{F}(p_a) \prod_{(i, a) \in E} \left\{ p_a \mathbb{I}(\hat{t}_i = A_{ia}) + (1 - p_a) \mathbb{I}(\hat{t}_i \neq A_{ia}) \right\},$$

where with a slight abuse of notation we use $\mathcal{F}(\cdot)$ to denote the prior probability density function of p_a 's and we use i and j to denote task nodes and a and b to denote worker nodes. For such a problem of finding the most probable realization in a graphical model, the celebrated belief propagation (BP) gives a good approximate solution. To be precise, BP is an approximation for maximizing the marginal distribution of each variable, and a similar algorithm known as min-sum algorithm approximates the most probable realization. However, the two algorithms are closely related, and in this section we only present standard BP. There is a long line of literature providing the theoretical and empirical evidences supporting the use BP [Pea88, YFW03].

Under the crowdsourcing graphical model, standard BP operates on two sets of messages: the task messages $\{\tilde{x}_{i \rightarrow a}\}_{(i, a) \in E}$ and the worker messages $\{\tilde{y}_{a \rightarrow i}\}_{(i, a) \in E}$. In our iterative algorithm the messages were scalar variables with real values, whereas the messages in BP are probability density functions. Each task message corresponds to an edge and each worker message also corresponds to an edge. The task node i corresponds to random variable \hat{t}_i , and the task message from task i to worker a , denoted by $\tilde{x}_{i \rightarrow a}$, represents our belief on the random variable \hat{t}_i . Then $\tilde{x}_{i \rightarrow a}$ is a probability distribution over $\{\pm 1\}$. Similarly, a worker node a corresponds to a random variable p_a . The worker message $\tilde{y}_{a \rightarrow i}$ is a probability distribution of p_a over $[0, 1]$. Following the standard BP framework, we iteratively update the messages according to the following rule. We start with

randomly initialized $\tilde{x}_{i \rightarrow a}$'s and at k -th iteration,

$$\begin{aligned} \tilde{y}_{a \rightarrow i}^{(k)}(p_a) &\propto \mathcal{F}(p_a) \prod_{j \in \partial a \setminus i} \left\{ (p_a + \bar{p}_a + (p_a - \bar{p}_a)A_{ja})\tilde{x}_{j \rightarrow a}^{(k)}(+1) + (p_a + \bar{p}_a - (p_a - \bar{p}_a)A_{ja})\tilde{x}_{j \rightarrow a}^{(k)}(-1) \right\}, \\ \tilde{x}_{i \rightarrow a}^{(k+1)}(\hat{t}_i) &\propto \prod_{b \in \partial i \setminus a} \int \left(\tilde{y}_{b \rightarrow i}^{(k)}(p_b) (p_b \mathbb{I}_{(A_{ib}=\hat{t}_i)} + \bar{p}_b \mathbb{I}_{(A_{ib} \neq \hat{t}_i)}) \right) dp_b, \end{aligned}$$

for all $(i, a) \in E$ and for $\bar{p} = 1 - p$. The above update rule only determines the messages up to a scaling, where \propto indicates that the left-hand side is proportional to the right-hand side. The algorithm produces the same estimates in the end regardless of the scaling. After a predefined number of iterations, we make a decision by computing the decision variable

$$\tilde{x}_i(\hat{t}_i) \propto \prod_{b \in \partial i} \int \left(\tilde{y}_{b \rightarrow i}^{(k)}(p_b) (p_b \mathbb{I}_{(A_{ib}=\hat{t}_i)} + \bar{p}_b \mathbb{I}_{(A_{ib} \neq \hat{t}_i)}) \right) dp_b,$$

and estimating $\hat{t}_i = \text{sign}(\tilde{x}_i(+) - \tilde{x}_i(-))$.

In a special case of a Haldane prior, where a worker either always tells the truth or always gives the wrong answer,

$$p_j = \begin{cases} 0 & \text{with probability } 1/2, \\ 1 & \text{otherwise,} \end{cases}$$

the above BP updates boils down to our iterative inference algorithm. Let $x_{i \rightarrow a} = \log(\tilde{x}_{i \rightarrow a}(+)/\tilde{x}_{i \rightarrow a}(-))$ denote the log-likelihood of $\tilde{x}_{i \rightarrow a}(\cdot)$. Under Haldane prior, p_a is also a binary random variable. We can use $y_{a \rightarrow i} = \log(\tilde{y}_{a \rightarrow i}(1)/\tilde{y}_{a \rightarrow i}(0))$ to denote the log-likelihood of $\tilde{y}_{a \rightarrow i}(\cdot)$. After some simplifications, the above BP update boils down to

$$\begin{aligned} y_{a \rightarrow i}^{(k)} &= \sum_{j \in \partial a \setminus i} A_{ja} x_{j \rightarrow a}^{(k-1)}, \\ x_{i \rightarrow a}^{(k)} &= \sum_{b \in \partial i \setminus a} A_{ib} y_{b \rightarrow i}^{(k)}. \end{aligned}$$

This is exactly the same update rule as our iterative inference algorithm (cf. Eqs. (1) and (2)). Thus, our algorithm is belief propagation for a very specific prior. Despite this, it is surprising that it performs near optimally (with random regular graph for task allocation) for all priors. This robustness property is due to the models assumed in this crowdsourcing problem and is not to be expected in general.

2.7 Discussion

In this section, we discuss several implications of our main results and possible future research directions in generalizing the model studied in this paper.

Below phase transition. We first discuss the performance guarantees in the below threshold regime when $\hat{\ell} \hat{r} q^2 < 1$. As we will show, the bound in (4) always holds even when $\hat{\ell} \hat{r} q^2 \leq 1$. However, numerical experiments suggest that we should stop our algorithm at first iteration when we are below the phase transition as discussed in Section 2.2. We provide an upper bound on

the resulting error when only one iteration of our iterative inference algorithm is used (which is equivalent as majority voting algorithm).

Notice that the bound in (4) is only meaningful when it is less than a half. When $\hat{\ell}\hat{r}q^2 \leq 1$ or $\ell q < 24 \log 2$, the right-hand side of inequality (4) is always larger than half. Hence the upper bound always holds, even without the assumption that $\hat{\ell}\hat{r}q^2 > 1$, and we only have that assumption in the statement of our main theorem to emphasize the phase transition in how our algorithm behaves.

However, we can also try to get a tighter bound than a trivial half implied from (4) in the below threshold regime. Specifically, we empirically observe that the error rate increases as the number of iterations k increases. Therefore, it makes sense to use $k = 1$. In which case, the algorithm essentially boils down to the majority rule. We can prove the following error bound which generally holds for any regime of ℓ , r and the worker distribution \mathcal{F} . A proof of this statement is provided in Section 3.6.

Lemma 2.10. *For any value of ℓ , r , and m , and any distribution of workers \mathcal{F} , the estimates we get after first step of our algorithm achieve*

$$\frac{1}{m} \sum_{i=1}^m \mathbb{P}(t_i \neq \hat{t}_i) \leq e^{-\ell\mu^2/4}, \quad (6)$$

where $\mu = \mathbb{E}_{\mathcal{F}}[2\mathbf{p}_j - 1]$.

Since μ is always between q and $q^{1/2}$, the scaling of the above error exponent is always worse than what we have after running our algorithm for a long time (cf. Theorem 2.1). This suggests that iterating our inference algorithm helps when $\hat{\ell}\hat{r}q^2 > 1$ and especially when the gap between μ and q is large. Under these conditions, our approach does significantly better than majority voting (cf. Figure 1). The gain of using our approach is maximized when there exists both good workers and bad workers. This is consistent with our intuition that when there is a variety of workers, our algorithm can identify the good ones and get better estimates.

Golden standard units. Next, consider the variation where we ask questions to workers whose answers are already known (also known as ‘gold standard units’). We can use these to assess the quality of the workers. There are two ways we can use this information. First, we can embed ‘seed gold units’ along with the standard tasks, and use these ‘seed gold units’ in turn to perform more informed inference. However, we can show that there is no gain in using such ‘seed gold units’. The optimal lower bound of $1/q \log(1/\varepsilon)$ essentially utilizes the existence of oracle that can identify the reliability of every worker *exactly*, i.e. the oracle has a lot more information than what can be gained by such embedded golden questions. Therefore, clearly ‘seed gold units’ *do not* help the oracle estimator, and hence the order optimality of our approach still holds even if we include all the strategies that can utilize these ‘seed gold units’. However, in practice, it is common to use the ‘seed gold units’, and this can improve the constant factor in the required budget, but not the scaling.

Alternatively, we can use ‘pilot gold units’ as qualifying or pilot questions that the workers must complete to qualify to participate. Typically a taskmaster do not have to pay for these qualifying questions and this provides an effective way to increase the quality of the participating workers. Our approach can benefit from such ‘pilot gold units’, which has the effect of increasing the effective collective quality of the crowd q . Further, if we can ‘measure’ how the distribution of workers change

when using pilot questions, then our main result fully describes how much we can gain by such pilot questions. In any case, pilot questions only change the distribution of participating workers, and the order-optimality of our approach still holds even if we compare all the schemes that use the same pilot questions.

How to optimize over a multiple choices of crowds. We next consider the scenario where we have a choice over which crowdsourcing platform to use from a set of platforms with different crowds. Each crowd might have different worker distributions with different prices. Specifically, suppose there are K crowds of workers: the k -th crowd has collective quality q_k and requires payment of c_k to perform a task. Now our optimality result suggests that the per-task cost scales as $c_k/q_k \log(1/\varepsilon)$ if we only used workers of class k . More generally, if we use a mix of these workers, say α_k fraction of workers from class k , with $\sum_k \alpha_k = 1$, then the effective parameter $q = \sum_k \alpha_k q_k$. And subject to this, the optimal per task cost scales as $(\sum_k \alpha_k c_k)/(\sum_k \alpha_k q_k) \log(1/\varepsilon)$. This immediately suggests that the optimal choice of fraction α_k must be such that $\alpha_k > 0$ only if $c_k/q_k = \min_i c_i/q_i$. That is, the optimal choice is to select workers only from the classes that have maximal quality per cost ratio of q_k/c_k over $k \in [K]$. One implication of this observation is that it suggests a pricing scheme for crowdsourcing platforms. If you are managing a crowdsourcing platform with the collective quality q and the cost c and there is another crowdsourcing platform with q' and c' , you want to choose the cost such that the quality per cost ratio is at least as good as the other crowd: $q/c \geq q'/c'$.

General crowdsourcing models. Finally, we consider possible generalizations of our model. The model assumed in this paper does not capture several factors: tasks with different level of difficulties or workers who always answer positive or negative. In general, the responses of a worker j to a binary question i may depend on several factors: (i) the correct answer to the task; (ii) the difficulty of the task; (iii) the expertise or the reliability of the worker; (iv) the bias of the worker towards positive or negative answers. Let $t_i \in \{+1, -1\}$ represent the correct answer and $r_i \in [0, \infty)$ represents the level of difficulty. also, let $\alpha_j \in [-\infty, \infty]$ represent the reliability and $\beta_j \in (-\infty, \infty)$ represent the bias of worker j . In formula, a worker j 's response to a binary task i can be modeled as

$$\mathbf{A}_{ij} = \text{sign}(\mathbf{Z}_{i,j}) ,$$

where $\mathbf{Z}_{i,j}$ is a Gaussian random variable distributed as $\mathbf{Z}_{i,j} \sim \mathcal{N}(\alpha_j t_i + \beta_j, r_i)$ and $\text{sign}(\mathbf{Z}) = 1$ almost surely for $\mathbf{Z} \sim \mathcal{N}(\infty, 1)$. A task with $r_i = 0$ is an easy task and large r_i is a difficult task. A worker with large positive α_j is more likely to give the right answer and large negative α_j is more likely to give the wrong answer. When $\alpha_j = 0$, the worker gives independent answers regardless of what the correct answer is. A worker with large β_j is biased towards positive responses and if $\beta_j = 0$ then the worker is unbiased. A similar model with multi-dimensional latent variables was studied in [WBBP10].

Most of the models studied in the crowdsourcing literature can be reduced to a special case of this model. For example, the early patient-classification model introduced by Dawid and Skene [DS79] is equivalent to the above Gaussian model with $r_i = 1$. Each worker is represented by two latent quality parameters p_j^+ and p_j^- , such that

$$\mathbf{A}_{ij} = \begin{cases} t_i & \text{with probability } p_j^{t_i} , \\ -t_i & \text{otherwise .} \end{cases}$$

This model captures the bias of workers. More recently, Whitehill et al. [WRW⁺09] introduced another model where $\mathbb{P}(A_{ij} = t_i | a_i, b_j) = 1/(1 + e^{-a_i b_j})$, with worker reliability a_i and task difficulty b_j . This is again a special case of the above Gaussian model if we set $\beta_j = 0$. The model we study in this paper has an underlying assumption that all the tasks share an equal level of difficulty and the workers are unbiased. It is equivalent to the above Gaussian model with $\beta_j = 0$ and $r_i = 1$. In this case, there is a one-to-one relation between the worker reliability p_j and α_j : $p_j = Q(\alpha_j)$, where $Q(\cdot)$ is the tail probability of the standard Gaussian distribution.

3 Proof of main results

In this section, we provide proofs of the main results.

3.1 Proof of the main result in Theorem 2.1

By symmetry, we can assume that all t_i 's are +1. Let $\hat{t}_i^{(k)}$ denote the resulting estimate of task i after k iterations of our iterative inference algorithm defined in Section 2.1. If we draw a random task \mathbf{I} uniformly in $[m]$, then we want to compute the average error probability, which is the probability that we make an error on this randomly chosen task:

$$\frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i^{(k)}) = \mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}). \quad (7)$$

We will prove an upper bound on the probability of error in two steps. First, we prove that the local neighborhood of a randomly chosen task node \mathbf{I} is a tree with high probability. Then, assuming that the graph is locally tree-like, we provide an upper bound on the error using a technique known as *density evolution*.

We construct a random bipartite graph $\mathbf{G}([m] \cup [n], E)$ according to the configuration model. We start with $[m] \times [\ell]$ half-edges for task nodes and $[n] \times [r]$ half-edges for the worker nodes, and pair all the $m\ell$ task half-edges to the same number of worker half-edges according to a random permutation of $[m\ell]$.

Let $\mathbf{G}_{i,k}$ denote a subgraph of $\mathbf{G}([m] \cup [n], E)$ that includes all the nodes whose distance from the ‘root’ i is at most k . At first iteration of our inference algorithm, to estimate the task i , we only use the responses provided by the workers who were assigned to task i . Hence we are performing inference on the local neighborhood $\mathbf{G}_{i,1}$. Similarly, when we run k iterations of our (message-passing) inference algorithm to estimate a task i , we only run inference on local subgraph $\mathbf{G}_{i,2k-1}$. Since we update both task and worker messages, we need to grow the subgraph by distance two at each iteration. When this local subgraph is a tree, then we can apply density evolution to analyze the probability of error. When this local subgraph is not a tree, we can make a pessimistic assumption that an error has been made to get an upper bound on the actual error probability.

$$\mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}) \leq \mathbb{P}(\mathbf{G}_{\mathbf{I},2k-1} \text{ is not a tree}) + \mathbb{P}(\mathbf{G}_{\mathbf{I},2k-1} \text{ is a tree and } t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}). \quad (8)$$

Next lemma bounds the first term and shows that the probability that a local subgraph is not a tree vanishes as m grows. A proof of this lemma is provided in Section 3.2.

Lemma 3.1. *For a random (ℓ, r) -regular bipartite graph generated according to the configuration model,*

$$\mathbb{P}(\mathbf{G}_{\mathbf{I}, 2k-1} \text{ is not a tree}) \leq ((\ell - 1)(r - 1))^{2k-2} \frac{3\ell r}{m}. \quad (9)$$

Then, to bound the second term of (8), we provide a sharp upper bound on the error probability conditioned on that $\mathbf{G}_{\mathbf{I}, 2k-1}$ is a tree. Let $x_i^{(k)}$ denote the decision variable for task i after k iterations of the iterative algorithm such that $\hat{t}_i^{(k)} = \text{sign}(x_i^{(k)})$. Then, we make an error whenever this decision variable is negative. When this is exactly zero, we make a random decision, in which case we make an error with probability half. Then,

$$\mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)} \mid G_{\mathbf{I}, k} \text{ is a tree}) \leq \mathbb{P}(x_{\mathbf{I}}^{(k)} \leq 0 \mid G_{\mathbf{I}, k} \text{ is a tree}). \quad (10)$$

To analyze the distribution of the decision variable on a locally tree-like graph, we use a standard probabilistic analysis technique known as ‘density evolution’ in coding theory or ‘recursive distributional equations’ in probabilistic combinatorics [RU08, MM09]. Precisely, we use the following equality that

$$\mathbb{P}(x_{\mathbf{I}}^{(k)} \leq 0 \mid G_{\mathbf{I}, k} \text{ is a tree}) = \mathbb{P}(\hat{\mathbf{x}}^{(k)} \leq 0), \quad (11)$$

where $\hat{\mathbf{x}}^{(k)}$ is defined through density evolution equations (13), (14) and (15) in the following. We will prove in the following that when $\hat{\ell} \hat{r} q^2 > 1$,

$$\mathbb{P}(\hat{\mathbf{x}}^{(k)} \leq 0) \leq e^{-\ell q / (2\sigma_k^2)}. \quad (12)$$

Together with equations (11), (10), (9), (8), and (7), this finishes the proof of Theorem 2.1.

Density evolution. At iteration k the algorithm operates on a set of messages $\{x_{i \rightarrow j}^{(k)}\}_{(i,j) \in E}$ and $\{y_{j \rightarrow i}^{(k)}\}_{(i,j) \in E}$. If we chose an edge (i, j) uniformly at random, the values of x and y messages on that randomly chosen edge define random variables whose randomness comes from random choice of the edge, any randomness introduced by the inference algorithm, the graph, and the realizations of \mathbf{p}_j ’s and \mathbf{A}_{ij} ’s. Let $\mathbf{x}^{(k)}$ denote this random variable corresponding to the message $x_{i \rightarrow j}^{(k)}$ and $\mathbf{y}_p^{(k)}$ denote the random variable corresponding to $y_{j \rightarrow i}^{(k)}$ conditioned on the latent worker quality being p for randomly chosen edge (i, j) .

As proved in Lemma 3.1, the (ℓ, r) -regular random graph locally converges in distribution to a (ℓ, r) -regular tree with high probability. On a tree, there is a recursive way of defining the distribution of messages $\mathbf{x}^{(k)}$ and $\mathbf{y}_p^{(k)}$. At initialization, we initialize the worker messages with Gaussian random variable with mean one and variance one. The corresponding random variable $\mathbf{y}_p^{(0)} \sim \mathcal{N}(1, 1)$, which at initial step is independent of the worker quality p , fully describes the distribution of $y_{j \rightarrow i}^{(0)}$ for all (i, j) . At first iteration, the task messages are updated according to $x_{i \rightarrow j}^{(1)} = \sum_{j' \in \partial i \setminus j} \mathbf{A}_{ij'} y_{j' \rightarrow i}^{(0)}$. If we know the distribution of $\mathbf{A}_{ij'}$ ’s and $y_{j' \rightarrow i}^{(0)}$ ’s, we can update the distribution of $x_{i \rightarrow j}^{(1)}$. Since we are assuming a tree, all $x_{i \rightarrow j}^{(1)}$ are independent. Further, because of the symmetry in the way we construct our random graph, all $x_{i \rightarrow j}^{(1)}$ ’s are identically distributed. Precisely, they are distributed according to $\mathbf{x}^{(1)}$ defined in (13). This recursively defines $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ through the *density evolution equations* in (13) and (14) [MM09].

Let us first introduce a few definitions first. Here and after, we drop the superscript k denoting the iteration number whenever it is clear from the context. Let \mathbf{x}_b 's and $\mathbf{y}_{p,a}$'s be independent random variables distributed according to \mathbf{x} and \mathbf{y}_p respectively. Also, $\mathbf{z}_{p,a}$'s and $\mathbf{z}_{p,b}$'s are independent random variables distributed according to \mathbf{z}_p , where

$$\mathbf{z}_p = \begin{cases} +1 & \text{with probability } p, \\ -1 & \text{with probability } 1 - p. \end{cases}$$

This represents the answer given by a worker conditioned on the worker having quality parameter p . Let $\mathbf{p} \sim \mathcal{F}$ be a random variable distributed according to the distribution of the worker's quality \mathcal{F} over $[0, 1]$. Then \mathbf{p}_a 's are independent random variable distributed according to \mathbf{p} . Further, $\mathbf{z}_{p,b}$'s and \mathbf{x}_b 's are independent, and $\mathbf{z}_{\mathbf{p}_a,a}$'s and $\mathbf{y}_{\mathbf{p}_a,a}$'s are conditionally independent conditioned on \mathbf{p}_a .

We initialize \mathbf{y}_p with a Gaussian distribution, whence it is independent of the latent variable p : $\mathbf{y}_p^{(0)} \sim \mathcal{N}(1, 1)$. Let $\stackrel{d}{=}$ denote equality in distribution. Then, for $k \in \{1, 2, \dots\}$, the task messages are distributed as the sum of $\ell - 1$ incoming messages that are independent and identically distributed according to $\mathbf{y}_{\mathbf{p}}^{(k-1)}$ and weighted by i.i.d. responses:

$$\mathbf{x}^{(k)} \stackrel{d}{=} \sum_{a \in [\ell-1]} \mathbf{z}_{\mathbf{p}_a,a} \mathbf{y}_{\mathbf{p}_a,a}^{(k-1)}. \quad (13)$$

Similarly, the worker messages (conditioned on the latent worker quality p) are distributed as the sum of $r - 1$ incoming messages that are independent and identically distributed according to $\mathbf{x}^{(k)}$ and weighted by i.i.d. responses:

$$\mathbf{y}_p^{(k)} \stackrel{d}{=} \sum_{b \in [r-1]} \mathbf{z}_{p,b} \mathbf{x}_b^{(k)}. \quad (14)$$

For the decision variable $x_{\mathbf{I}}^{(k)}$ on a randomly chosen task \mathbf{I} , we have

$$\hat{\mathbf{x}}^{(k)} \stackrel{d}{=} \sum_{i \in [\ell]} \mathbf{z}_{\mathbf{p}_i,i} \mathbf{y}_{\mathbf{p}_i,i}^{(k-1)}. \quad (15)$$

Numerically or analytically computing the densities in (13) and (14) exactly is not computationally feasible when the messages take continuous values as is the case for our algorithm. Typically, heuristics are used to approximate the densities such as quantizing the messages, approximating the density with simple functions, or using Monte Carlo method to sample from the density. A novel contribution of our analysis is that we prove that the messages are sub-Gaussian using recursion, and we provide an upper bound on the parameters in a closed form. This allows us to prove the sharp result on the error bound that decays exponentially.

Mean and variance computation. To give an intuition on how the messages behave, we describe the evolution of the mean and the variance of the random variables in (13) and (14). Let \mathbf{p} be a random variable distributed according to the measure \mathcal{F} . Define $m^{(k)} \equiv \mathbb{E}[\mathbf{x}^{(k)}]$, $\hat{m}_{\mathbf{p}}^{(k)} \equiv \mathbb{E}[\mathbf{y}_{\mathbf{p}}^{(k)} | \mathbf{p}]$, $v^{(k)} \equiv \text{Var}(\mathbf{x}^{(k)})$, and $\hat{v}_{\mathbf{p}}^{(k)} \equiv \text{Var}(\mathbf{y}_{\mathbf{p}}^{(k)} | \mathbf{p})$. Also let $\hat{\ell} = \ell - 1$ and $\hat{r} = r - 1$ to

simplify notation. Then, from (13) and (14) we get that

$$\begin{aligned}
m^{(k)} &= \hat{\ell} \mathbb{E}_{\mathbf{p}}[(2\mathbf{p} - 1)\hat{m}_{\mathbf{p}}^{(k-1)}] , \\
\hat{m}_{\mathbf{p}}^{(k)} &= \hat{r} (2p - 1)m^{(k)} , \\
v^{(k)} &= \hat{\ell} \left\{ \mathbb{E}_{\mathbf{p}}[\hat{v}_{\mathbf{p}}^{(k-1)} + (\hat{m}_{\mathbf{p}}^{(k-1)})^2] - \left(\mathbb{E}_{\mathbf{p}}[(2\mathbf{p} - 1)\hat{m}_{\mathbf{p}}^{(k-1)}] \right)^2 \right\} , \\
\hat{v}_{\mathbf{p}}^{(k)} &= \hat{r} \left\{ v^{(k)} + (m^{(k)})^2 - ((2p - 1)m^{(k)})^2 \right\} .
\end{aligned}$$

Recall that $\mu = \mathbb{E}[2\mathbf{p} - 1]$ and $q = \mathbb{E}[(2\mathbf{p} - 1)^2]$. Substituting $\hat{m}_{\mathbf{p}}$ and $\hat{v}_{\mathbf{p}}$ we get the following evolution of the first and the second moment of the random variable $x^{(k)}$.

$$\begin{aligned}
m^{(k+1)} &= \hat{\ell}\hat{r}qm^{(k)} , \\
v^{(k+1)} &= \hat{\ell}\hat{r}v^{(k)} + \hat{\ell}\hat{r}(m^{(k)})^2(1 - q)(1 + \hat{r}q) .
\end{aligned}$$

Since $\hat{m}_{\mathbf{p}}^{(0)} = 1$ and $\hat{v}^{(0)} = 1$ as per our assumption, we have $m^{(1)} = \mu\hat{\ell}$ and $v^{(1)} = \hat{\ell}(4 - \mu^2)$. This implies that $m^{(k)} = \mu\hat{\ell}(\hat{\ell}\hat{r}q)^{k-1}$, and $v^{(k)} = av^{(k-1)} + bc^{k-2}$, with $a = \hat{\ell}\hat{r}$, $b = \mu^2\hat{\ell}^3\hat{r}(1 - q)(1 + \hat{r}q)$, and $c = (\hat{\ell}\hat{r}q)^2$. After some algebra, it follows that $v^{(k)} = v^{(1)}a^{k-1} + bc^{k-2}\sum_{\ell=0}^{k-2}(a/c)^\ell$.

For $\hat{\ell}\hat{r}q^2 > 1$, we have $a/c < 1$ and

$$v^{(k)} = \hat{\ell}(4 - \mu^2)(\hat{\ell}\hat{r})^{k-1} + (1 - q)(1 + \hat{r}q)\mu^2\hat{\ell}^2(\hat{\ell}\hat{r}q)^{2k-2}\frac{1 - 1/(\hat{\ell}\hat{r}q^2)^{k-1}}{\hat{\ell}\hat{r}q^2 - 1} .$$

The first and second moment of the decision variable $\hat{\mathbf{x}}^{(k)}$ in (15) can be computed using a similar analysis: $\mathbb{E}[\hat{\mathbf{x}}^{(k)}] = (\ell/\hat{\ell})m^{(k)}$ and $\text{Var}(\hat{\mathbf{x}}^{(k)}) = (\ell/\hat{\ell})v^{(k)}$. In particular, we have

$$\frac{\text{Var}(\hat{\mathbf{x}}^{(k)})}{\mathbb{E}[\hat{\mathbf{x}}^{(k)}]^2} = \frac{\hat{\ell}(4 - \mu^2)}{\hat{\ell}\mu^2(\hat{\ell}\hat{r}q^2)^{k-1}} + \frac{\hat{\ell}(1 - q)(1 + \hat{r}q)}{\ell(\hat{\ell}\hat{r}q^2 - 1)} \left(1 - \frac{1}{(\hat{\ell}\hat{r}q^2)^{k-1}} \right) .$$

Applying Chebyshev's inequality, it immediately follows that $\mathbb{P}(\hat{\mathbf{x}}^{(k)} < 0)$ is bounded by the right-hand side of the above equality. This bound is weak compared to the bound in Theorem 2.1. In the following, we prove a stronger result using the sub-Gaussianity of $\mathbf{x}^{(k)}$. But first, let us analyze what this weaker bound gives for different regimes of ℓ , r , and q , which indicates that the messages exhibit a fundamentally different behavior in the regimes separated by a phase transition at $\hat{\ell}\hat{r}q^2 = 1$.

In a 'good' regime where we have $\hat{\ell}\hat{r}q^2 > 1$, the bound converges to a finite limit as the number of iterations k grows. Namely,

$$\lim_{k \rightarrow \infty} \mathbb{P}(\hat{x}^{(k)} < 0) \leq \frac{\hat{\ell}(1 - q)(1 + \hat{r}q)}{\ell(\hat{\ell}\hat{r}q^2 - 1)} .$$

Notice that the upper bound converges to $(1 - q)/(\ell q)$ as $\hat{\ell}\hat{r}q^2$ grows. This scales in the same way as the known bounds for using the left singular vector directly for inference (cf. [KOS11]). In the case when $\hat{\ell}\hat{r}q^2 < 1$, the same analysis gives

$$\frac{\text{Var}(\hat{x}^{(k)})}{\mathbb{E}[\hat{x}^{(k)}]^2} = e^{\Theta(k)} .$$

Finally, when $\hat{\ell}\hat{r}q^2 = 1$, we get $v^{(k)} = (\hat{\ell}\hat{r})^k + \hat{\ell}\hat{r}(1-q)(1+\hat{r}q)(\hat{\ell}\hat{r}q)^{2k-2}$, which implies

$$\frac{\text{Var}(\hat{x}^{(k)})}{\mathbb{E}[\hat{x}^{(k)}]^2} = \Theta(k).$$

Analyzing the density. Our strategy to provide a tight upper bound on $\mathbb{P}(\hat{\mathbf{x}}^{(k)} \leq 0)$ is to show that $\hat{\mathbf{x}}^{(k)}$ is sub-Gaussian with appropriate parameters and use the Chernoff bound. A random variable \mathbf{z} with mean m is said to be *sub-Gaussian* with parameter $\tilde{\sigma}$ if for all $\lambda \in \mathbb{R}$ the following inequality holds:

$$\mathbb{E}[e^{\lambda\mathbf{z}}] \leq e^{m\lambda + (1/2)\tilde{\sigma}^2\lambda^2}.$$

Define

$$\tilde{\sigma}_k^2 \equiv 2\hat{\ell}(\hat{\ell}\hat{r})^{k-1} + \mu^2\hat{\ell}^3\hat{r}(3q\hat{r}+1)(q\hat{\ell}\hat{r})^{2k-4} \frac{1 - (1/q^2\hat{\ell}\hat{r})^{k-1}}{1 - (1/q^2\hat{\ell}\hat{r})},$$

and $m_k \equiv \mu\hat{\ell}(q\hat{\ell}\hat{r})^{k-1}$ for $k \in \mathbb{Z}$. We will first show that, $\mathbf{x}^{(k)}$ is sub-Gaussian with mean m_k and parameter $\tilde{\sigma}_k^2$ for a regime of λ we are interested in. Precisely, we will show that for $|\lambda| \leq 1/(2m_{k-1}\hat{r})$,

$$\mathbb{E}[e^{\lambda\mathbf{x}^{(k)}}] \leq e^{m_k\lambda + (1/2)\tilde{\sigma}_k^2\lambda^2}. \quad (16)$$

By definition, due to distributional independence, we have $\mathbb{E}[e^{\lambda\hat{\mathbf{x}}^{(k)}}] = \mathbb{E}[e^{\lambda\mathbf{x}^{(k)}}]^{(\ell/\hat{\ell})}$. Therefore, it follows from (16) that $\hat{\mathbf{x}}^{(k)}$ satisfies $\mathbb{E}[e^{\lambda\hat{\mathbf{x}}^{(k)}}] \leq e^{(\ell/\hat{\ell})m_k\lambda + (\ell/2\hat{\ell})\tilde{\sigma}_k^2\lambda^2}$. Applying the Chernoff bound with $\lambda = -m_k/(\tilde{\sigma}_k^2)$, we get

$$\mathbb{P}(\hat{\mathbf{x}}^{(k)} \leq 0) \leq \mathbb{E}[e^{\lambda\hat{\mathbf{x}}^{(k)}}] \leq e^{-\ell m_k^2 / (2\hat{\ell}\tilde{\sigma}_k^2)}, \quad (17)$$

Since $m_k m_{k-1} / (\tilde{\sigma}_k^2) \leq \mu^2 \hat{\ell}^2 (q\hat{\ell}\hat{r})^{2k-3} / (3\mu^2 q \hat{\ell}^3 \hat{r}^2 (q\hat{\ell}\hat{r})^{2k-4}) = 1/(3\hat{r})$, it is easy to check that $|\lambda| \leq 1/(2m_{k-1}\hat{r})$. This implies the desired bound in (12).

Now we are left to prove that $\mathbf{x}^{(k)}$ is sub-Gaussian with appropriate parameters. We can write down a recursive formula for the evolution of the moment generating functions of \mathbf{x} and \mathbf{y}_p as

$$\mathbb{E}[e^{\lambda\mathbf{x}^{(k)}}] = \left(\mathbb{E}_{\mathbf{p}} \left[\mathbf{p} \mathbb{E}[e^{\lambda\mathbf{y}_{\mathbf{p}}^{(k-1)}} | \mathbf{p}] + \bar{\mathbf{p}} \mathbb{E}[e^{-\lambda\mathbf{y}_{\mathbf{p}}^{(k-1)}} | \mathbf{p}] \right] \right)^{\hat{\ell}}, \quad (18)$$

$$\mathbb{E}[e^{\lambda\mathbf{y}_p^{(k)}}] = \left(p \mathbb{E}[e^{\lambda\mathbf{x}^{(k)}}] + \bar{p} \mathbb{E}[e^{-\lambda\mathbf{x}^{(k)}}] \right)^{\hat{r}}, \quad (19)$$

where $\bar{p} = 1 - p$ and $\bar{\mathbf{p}} = 1 - \mathbf{p}$. We can prove that these are sub-Gaussian using induction.

First, for $k = 1$, we show that $\mathbf{x}^{(1)}$ is sub-Gaussian with mean $m_1 = \mu\hat{\ell}$ and parameter $\tilde{\sigma}_1^2 = 2\hat{\ell}$, where $\mu \equiv \mathbb{E}[2\mathbf{p} - 1]$. Since \mathbf{y}_p is initialized as Gaussian with unit mean and variance, we have $\mathbb{E}[e^{\lambda\mathbf{y}_p^{(0)}}] = e^{\lambda + (1/2)\lambda^2}$ regardless of p . Substituting this into (18), we get for any λ ,

$$\mathbb{E}[e^{\lambda\mathbf{x}^{(1)}}] = \left(\mathbb{E}[\mathbf{p}]e^{\lambda} + (1 - \mathbb{E}[\mathbf{p}])e^{-\lambda} \right)^{\hat{\ell}} e^{(1/2)\hat{\ell}\lambda^2} \leq e^{\hat{\ell}\mu\lambda + \hat{\ell}\lambda^2}, \quad (20)$$

where the inequality follows from the fact that $ae^z + (1-a)e^{-z} \leq e^{(2a-1)z + (1/2)z^2}$ for any $z \in \mathbb{R}$ and $a \in [0, 1]$ (cf. [AS08, Lemma A.1.5]).

Next, assuming $\mathbb{E}[e^{\lambda \mathbf{x}^{(k)}}] \leq e^{m_k \lambda + (1/2) \tilde{\sigma}_k^2 \lambda^2}$ for $|\lambda| \leq 1/(2m_{k-1} \hat{r})$, we show that $\mathbb{E}[e^{\lambda \mathbf{x}^{(k+1)}}] \leq e^{m_{k+1} \lambda + (1/2) \tilde{\sigma}_{k+1}^2 \lambda^2}$ for $|\lambda| \leq 1/(2m_k \hat{r})$, and compute appropriate m_{k+1} and $\tilde{\sigma}_{k+1}^2$. Substituting the bound $\mathbb{E}[e^{\lambda \mathbf{x}^{(k)}}] \leq e^{m_k \lambda + (1/2) \tilde{\sigma}_k^2 \lambda^2}$ in (19), we get $\mathbb{E}[e^{\lambda \mathbf{y}_p^{(k)}}] \leq (pe^{m_k \lambda} + \bar{p}e^{-m_k \lambda})^{\hat{r}} e^{(1/2) \hat{r} \tilde{\sigma}_k^2 \lambda^2}$. Further applying this bound in (18), we get

$$\mathbb{E}[e^{\lambda \mathbf{x}^{(k+1)}}] \leq \left(\mathbb{E}_{\mathbf{p}} \left[\mathbf{p} (pe^{m_k \lambda} + \bar{p}e^{-m_k \lambda})^{\hat{r}} + \bar{\mathbf{p}} (\bar{p}e^{-m_k \lambda} + pe^{m_k \lambda})^{\hat{r}} \right] \right)^{\hat{\ell}} e^{(1/2) \hat{\ell} \hat{r} \tilde{\sigma}_k^2 \lambda^2}. \quad (21)$$

To bound the first term in the right-hand side, we use the next key lemma. A proof of this lemma is provided in Section 3.3.

Lemma 3.2. *For any $|z| \leq 1/(2\hat{r})$ and $\mathbf{p} \in [0, 1]$ such that $q = \mathbb{E}[(2\mathbf{p} - 1)^2]$, we have*

$$\mathbb{E}_{\mathbf{p}} \left[\mathbf{p} (pe^z + \bar{p}e^{-z})^{\hat{r}} + \bar{\mathbf{p}} (\bar{p}e^z + pe^{-z})^{\hat{r}} \right] \leq e^{q\hat{r}z + (1/2)(3q\hat{r}^2 + \hat{r})z^2}.$$

Applying this inequality to (21) gives

$$\mathbb{E}[e^{\lambda \mathbf{x}^{(k+1)}}] \leq e^{q\hat{\ell}\hat{r}m_k \lambda + (1/2) \left((3q\hat{\ell}\hat{r}^2 + \hat{\ell}\hat{r})m_k^2 + \hat{\ell}\hat{r}\tilde{\sigma}_k^2 \right) \lambda^2},$$

for $|\lambda| \leq 1/(2m_k \hat{r})$. In the regime where $q\hat{\ell}\hat{r} \geq 1$ as per our assumption, m_k is non-decreasing in k . At iteration k , the above recursion holds for $|\lambda| \leq \min\{1/(2m_1 \hat{r}), \dots, 1/(2m_{k-1} \hat{r})\} = 1/(2m_{k-1} \hat{r})$. Hence, we get the following recursion for m_k and $\tilde{\sigma}_k$ such that (16) holds for $|\lambda| \leq 1/(2m_{k-1} \hat{r})$.

$$\begin{aligned} m_{k+1} &= q\hat{\ell}\hat{r}m_k, \\ \tilde{\sigma}_{k+1}^2 &= (3q\hat{\ell}\hat{r}^2 + \hat{\ell}\hat{r})m_k^2 + \hat{\ell}\hat{r}\tilde{\sigma}_k^2. \end{aligned}$$

With the initialization $m_1 = \mu\hat{\ell}$ and $\tilde{\sigma}_1^2 = 2\hat{\ell}$, we have $m_k = \mu\hat{\ell}(q\hat{\ell}\hat{r})^{k-1}$ for $k \in \{1, 2, \dots\}$ and $\tilde{\sigma}_k^2 = a\tilde{\sigma}_{k-1}^2 + bc^{k-2}$ for $k \in \{2, 3, \dots\}$, with $a = \hat{\ell}\hat{r}$, $b = \mu^2\hat{\ell}^2(3q\hat{\ell}\hat{r}^2 + \hat{\ell}\hat{r})$, and $c = (q\hat{\ell}\hat{r})^2$. After some algebra, it follows that $\tilde{\sigma}_k^2 = \tilde{\sigma}_1^2 a^{k-1} + bc^{k-2} \sum_{\ell=0}^{k-2} (a/c)^\ell$. For $\hat{\ell}\hat{r}q^2 \neq 1$, we have $a/c \neq 1$, whence $\tilde{\sigma}_k^2 = \tilde{\sigma}_1^2 a^{k-1} + bc^{k-2} (1 - (a/c)^{k-1}) / (1 - a/c)$. This finishes the proof of (16).

3.2 Proof of Lemma 3.1

Consider the following discrete time random process that generates the random graph $\mathbf{G}_{\mathbf{I}, 2k-1}$ starting from the root \mathbf{I} . At first step, we connect ℓ worker nodes to node \mathbf{I} according to the configuration model, where ℓ half-edges are matches to a randomly chosen subset of nr worker half-edges of size ℓ . Let α_1 denote the probability that the resulting graph is a tree, that is no pair of edges are connected to the same worker node. Since there are $\binom{\ell}{2}$ pairs and each pair of half-edges are connected to the same worker node with probability $(r-1)/(nr-1)$:

$$\alpha_1 \leq \binom{\ell}{2} \frac{r-1}{nr-1}.$$

Similarly, define

$$\begin{aligned} \alpha_t &\equiv \mathbb{P}(\mathbf{G}_{\mathbf{I}, 2t-1} \text{ is not a tree} \mid \mathbf{G}_{\mathbf{I}, 2t-2} \text{ is a tree}), \text{ and} \\ \beta_t &\equiv \mathbb{P}(\mathbf{G}_{\mathbf{I}, 2t-2} \text{ is not a tree} \mid \mathbf{G}_{\mathbf{I}, 2t-3} \text{ is a tree}). \end{aligned}$$

Then,

$$\mathbb{P}(\mathbf{G}_{\mathbf{I},2k-1} \text{ is not a tree}) \leq \alpha_1 + \sum_{t=2}^k (\alpha_t + \beta_t). \quad (22)$$

We can upper bound α_t 's and β_t 's in a similar way. For generating $\mathbf{G}_{\mathbf{I},2t-1}$ conditioned on $\mathbf{G}_{\mathbf{I},2t-2}$ being a tree, there are $\ell(\hat{\ell}\hat{r})^{t-1}$ half-edges, where $\hat{\ell} = \ell - 1$ and $\hat{r} = r - 1$. Among $\binom{\ell(\hat{\ell}\hat{r})^{t-1}}{2}$ pairs of these half-edges, each pair will be connected to the same worker with probability at most $(r-1)/(r(n - \sum_{a=1}^{t-1} \ell(\hat{\ell}\hat{r})^{a-1}) - 1)$, where $\sum_{a=1}^{t-1} \ell(\hat{\ell}\hat{r})^{a-1}$ is the total number of worker nodes that are assigned so far in $\mathbf{G}_{\mathbf{I},2t-2}$. Then,

$$\begin{aligned} \alpha_t &\leq \frac{\ell^2(\hat{\ell}\hat{r})^{2t-2}}{2} \frac{r-1}{r(n - (\ell((\hat{\ell}\hat{r})^{t-2} - 1))/(\hat{\ell}\hat{r} - 1)) - 1} \\ &\leq \frac{\ell^2(\hat{\ell}\hat{r})^{2t-2}}{2(n - \ell(\hat{\ell}\hat{r})^{t-2}/2)} \\ &\leq \frac{\ell^2(\hat{\ell}\hat{r})^{2t-2}}{n} + \frac{\ell(\hat{\ell}\hat{r})^{t-2}}{n} \\ &\leq \frac{3\ell^2(\hat{\ell}\hat{r})^{2t-2}}{2n}, \end{aligned}$$

where the second inequality follows from the fact that $(a-1)/(b-1) \leq a/b$ for all $a \leq b$ and $\hat{\ell}\hat{r} \geq 2$ as per our assumption, and in the third inequality we used the fact that α_t is upper bounded by one and the fact that for $f(x) = b/(x-a)$ which is upper bounded by one, we have $f(x) \leq (2b/x) + (2a/x)$. Similarly, we can show that

$$\beta_t \leq \frac{3\ell^2(\hat{\ell}\hat{r})^{2t-2}}{\hat{\ell}^2 m}.$$

Substituting α_t and β_t into (22), we get that

$$\mathbb{P}(\mathbf{G}_{\mathbf{I},2k-1} \text{ is not a tree}) \leq (\hat{\ell}\hat{r})^{2k-2} \frac{3\ell r}{m}.$$

3.3 Proof of Lemma 3.2

By the fact that $ae^b + (1-a)e^{-b} \leq e^{(2a-1)b+(1/2)b^2}$ for any $b \in \mathbb{R}$ and $a \in [0, 1]$, we have $\mathbf{p}e^z + \bar{\mathbf{p}}e^{-z} \leq e^{(2\mathbf{p}-1)z+(1/2)z^2}$ almost surely. Applying this inequality once again, we get

$$\mathbb{E}\left[\mathbf{p}(\mathbf{p}e^z + \bar{\mathbf{p}}e^{-z})^{\hat{r}} + \bar{\mathbf{p}}(\bar{\mathbf{p}}e^z + \mathbf{p}e^{-z})^{\hat{r}}\right] \leq \mathbb{E}\left[e^{(2\mathbf{p}-1)^2\hat{r}z+(1/2)(2\mathbf{p}-1)^2\hat{r}^2z^2}\right] e^{(1/2)\hat{r}z^2}.$$

Using the fact that $e^a \leq 1 + a + 0.63a^2$ for $|a| \leq 5/8$,

$$\begin{aligned} &\mathbb{E}\left[e^{(2\mathbf{p}-1)^2\hat{r}z+(1/2)(2\mathbf{p}-1)^2\hat{r}^2z^2}\right] \\ &\leq \mathbb{E}\left[1 + (2\mathbf{p}-1)^2\hat{r}z + (1/2)(2\mathbf{p}-1)^2\hat{r}^2z^2 + 0.63((2\mathbf{p}-1)^2\hat{r}z + (1/2)(2\mathbf{p}-1)^2\hat{r}^2z^2)^2\right] \\ &\leq 1 + q\hat{r}z + (3/2)q\hat{r}^2z^2 \\ &\leq e^{q\hat{r}z+(3/2)q\hat{r}^2z^2}, \end{aligned}$$

for $|z| \leq 1/(2\hat{r})$. This proves Lemma 3.2.

3.4 Proof of a bound on majority voting in Lemma 2.6

Majority voting simply follows what the majority of workers agree on. In formula, $\hat{t}_i = \text{sign}(\sum_{j \in W_i} A_{ij})$, where W_i denotes the neighborhood of node i in the graph. It makes a random choice when there is a tie. We want to compute a lower bound on $\mathbb{P}(\hat{t}_i \neq t_i)$. Let $x_i = \sum_{j \in W_i} A_{ij}$. Assuming $t_i = +1$ without loss of generality, the error rate is lower bounded by $\mathbb{P}(x_i < 0)$. After rescaling, $(1/2)(x_i + \ell)$ is a standard binomial random variable $\text{Binom}(\ell, \alpha)$, where ℓ is the number of neighbors of the node i , $\alpha = \mathbb{E}[\mathbf{p}_j]$, and by assumption each A_{ij} is one with probability α .

It follows that $\mathbb{P}(x_i = -\ell + 2k) = ((\ell!)/(\ell - k)!k!) \alpha^k (1 - \alpha)^{\ell - k}$. Further, for $k \leq \alpha\ell - 1$, the probability distribution function is monotonically increasing. Precisely,

$$\frac{\mathbb{P}(x_i = -\ell + 2(k+1))}{\mathbb{P}(x_i = -\ell + 2k)} \geq \frac{\alpha(\ell - k)}{(1 - \alpha)(k+1)} \geq \frac{\alpha(\ell - \alpha\ell + 1)}{(1 - \alpha)\alpha\ell} > 1,$$

where we used the fact that the above ratio is decreasing in k whence the minimum is achieved at $k = \alpha\ell - 1$ under our assumption.

Let us assume that ℓ is even, so that x_i take even values. When ℓ is odd, the same analysis works, but x_i takes odd values. Our strategy is to use a simple bound: $\mathbb{P}(x_i < 0) \geq k\mathbb{P}(x_i = -2k)$. By assumption that $\alpha = \mathbb{E}[\mathbf{p}_j] \geq 1/2$, For an appropriate choice of $k = \sqrt{\ell}$, the right-hand side closely approximates the error probability. By definition of x_i , it follows that

$$\mathbb{P}(x_i = -2\sqrt{\ell}) = \binom{\ell}{\ell/2 + \sqrt{\ell}} \alpha^{\ell/2 - \sqrt{\ell}} (1 - \alpha)^{\ell/2 + \sqrt{\ell}}. \quad (23)$$

Applying Stirling's approximation, we can show that

$$\binom{\ell}{\ell/2 + \sqrt{\ell}} \geq \frac{C_2}{\sqrt{\ell}} 2^\ell, \quad (24)$$

for some positive constant C_2 . We are interested in the case where worker quality is low, that is α is close to $1/2$. Accordingly, for the second and third terms in (23), we expand in terms of $2\alpha - 1$.

$$\begin{aligned} & \log \left(\alpha^{\ell/2 - \sqrt{\ell}} (1 - \alpha)^{\ell/2 + \sqrt{\ell}} \right) \\ &= \left(\frac{\ell}{2} - \sqrt{\ell} \right) \left(\log(1 + (2\alpha - 1)) - \log(2) \right) + \left(\frac{\ell}{2} + \sqrt{\ell} \right) \left(\log(1 - (2\alpha - 1)) - \log(2) \right) \\ &= -\ell \log(2) - \frac{\ell(2\alpha - 1)^2}{2} + O(\sqrt{\ell}(2\alpha - 1)^4). \end{aligned} \quad (25)$$

We can substitute (24) and (25) in (23) to get the following bound:

$$\mathbb{P}(x_i < 0) \geq \exp \left\{ -C_3(\ell(2\alpha - 1)^2 + 1) \right\}, \quad (26)$$

for some positive constant C_3 .

Now, let ℓ_i denote the degree of task node i , such that $\sum_i \ell_i = \ell m$. Then for any $\{t_i\} \in \{\pm 1\}^m$, any distribution of \mathbf{p} such that $\mu = \mathbb{E}[2\mathbf{p} - 1] = 2\alpha - 1$, and any non-adaptive task assignment for m tasks, the following lower bound is true.

$$\begin{aligned} \frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i) &\geq \frac{1}{m} \sum_{i=1}^m e^{-C_3(\ell_i \mu^2 + 1)} \\ &\geq e^{-C_3(\ell \mu^2 + 1)}, \end{aligned}$$

where the last inequality follows from convexity of the exponential function. Under the spammer-hammer model, where $\mu = q$ this gives

$$\min_{\tau \in \mathcal{T}_\ell} \max_{t \in \{\pm 1\}^m, \mathcal{F} \in \mathcal{F}_q} \frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i) \geq e^{-C_3(\ell q^2 + 1)}.$$

This finishes the proof of lemma.

3.5 Proof of a bound on the adaptive schemes in Theorem 2.7

In this section, we prove that, even with the help of an oracle, the probability of error cannot decay faster than $e^{-C\ell q}$. We consider an labeling algorithm which has access to an oracle that knows the reliability of every worker (all the p_j 's). At k -th step, after the algorithm assign T_k and all the $|T_k|$ answers are collected from the k -th worker, the oracle provides the algorithm with p_k . Using all the previously collected answers $\{A_{ij}\}_{j \leq k}$ and the worker reliability $\{p_j\}_{j \leq k}$, the algorithm makes a decision on the next task assignment T_{k+1} . This process is repeated until a stopping criterion is met, and the algorithm outputs the optimal estimate of the true labels. The algorithm can compute the maximum likelihood estimates, which is known to minimize the probability of making an error. Let W_i be the set of workers assigned to task i , then

$$\hat{t}_i = \text{sign} \left(\sum_{j \in W_i} \log \left(\frac{p_j}{1 - p_j} \right) A_{ij} \right). \quad (27)$$

We are going to show that there exists a family of distributions \mathcal{F} such that for any stopping rule and any task assignment scheme, the probability of error is lower bounded by $e^{-C\ell q}$. We define the following family of distributions according to the spammer-hammer model with imperfect hammers. We assume that $q \leq a^2$ and

$$p_j = \begin{cases} 1/2 & \text{with probability } 1 - (q/a^2), \\ 1/2(1 + a) & \text{with probability } q/a^2, \end{cases}$$

such that $\mathbb{E}[(2p_j - 1)^2] = q$.

Let W_i denote the set of workers assigned to task i when the algorithm has stopped. Then $|W_i|$ is a random variable representing the total number of workers assigned to task i . The oracle estimator knows all the values necessary to compute the error probability of each task. Let $\mathcal{E}_i = \mathbb{E}[\mathbb{I}(t_i \neq \hat{t}_i) | \{A_{ij}\}, \{p_j\}]$ be the random variable representing the error probability as computed by the oracle estimator, conditioned on the $|W_i|$ responses we get from the workers and their reliability p_j 's. We are interested in identifying how the average budget $(1/m) \sum_i \mathbb{E}[|W_i|]$ depends on the achieve average error rate $(1/m) \sum_i \mathbb{E}[\mathcal{E}_i]$. In the following we will show that for any task i , independent of which task allocation scheme is used, it is necessary that

$$\mathbb{E}[|W_i|] \geq \frac{0.27}{q} \log \left(\frac{1}{2\mathbb{E}[\mathcal{E}_i]} \right). \quad (28)$$

By convexity of $\log(1/x)$ and Jensen's inequality, this implies that

$$\frac{1}{m} \sum_{i=1}^m \mathbb{E}[|W_i|] \geq \frac{0.27}{q} \log \left(\frac{1}{2(1/m) \sum_{i=1}^m \mathbb{E}[\mathcal{E}_i]} \right).$$

Since the total number of queries has to be consistent, we have $\sum_j |T_j| = \sum_i |W_i| \leq m\ell$. Also, by definition $\mathbb{E}[\mathcal{E}_i] = \mathbb{P}(t_i \neq \hat{t}_i)$. Then, from the above inequality, we get $(1/m) \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i) \geq (1/2)e^{-(1/0.27)q\ell}$, which finishes the proof of the theorem. Note that this bound holds for any value of m .

Now, we are left to prove that the inequality (28) holds. Focusing on a single task i , since we know who the spammers are and spammers give us no information about the task, we only need the responses from the reliable workers in order to make an optimal estimate as per (27). The conditional error probability \mathcal{E}_i of the optimal estimate depends on the realizations of the answers $\{A_{ij}\}_{j \in W_i}$ and the worker reliability $\{p_j\}_{j \in W_i}$. The following lower bound on the error only depends on the number of reliable workers, which we denote by ℓ_i .

Without loss of generality, let $t_i = +1$. Then, if all the reliable workers provide ‘-’ answers, the maximum likelihood estimation would be ‘-’ for this task. This leads to an error. Therefore,

$$\begin{aligned} \mathcal{E}_i &\geq \mathbb{P}(\text{all } \ell_i \text{ reliable workers answered } -) \\ &= \frac{1}{2} \left(\frac{1-a}{2} \right)^{\ell_i}, \end{aligned}$$

for all the realizations of $\{A_{ij}\}$ and $\{p_j\}$. The scaling by half ensures that the above inequality holds even when $\ell_i = 0$. By convexity and Jensen’s inequality, it follows that

$$\mathbb{E}[\ell_i] \geq \frac{\log(2\mathbb{E}[\mathcal{E}_i])}{\log((1-a)/2)}.$$

When we recruit $|W_i|$ workers, we see $\ell_i = (q/a^2)|W_i|$ reliable ones on average. Formally, we have $\mathbb{E}[\ell_i] = (q/a^2)\mathbb{E}[|W_i|]$. Again applying Jensen’s inequality, we get

$$\mathbb{E}[|W_i|] \geq \frac{1}{q} \frac{a^2}{\log((1-a)/2)} \log(2\mathbb{E}[\mathcal{E}_i]).$$

Maximizing over all choices of $a \in (0, 1)$, we get

$$\mathbb{E}[|W_i|] \geq -\log(2\mathbb{E}[\mathcal{E}_i]) \frac{0.27}{q},$$

which in particular is true with $a = 0.8$. For this choice of a , the result holds in the regime where $q \leq 0.64$. Notice that by changing the constant in the bound, we can ensure that the result holds for any values of q . This finishes the proof of (28).

3.6 Proof of a bound with one iteration in Lemma 2.10

The probability of making an error after one iteration of our algorithm for node i is $\mathbb{P}(t_i \neq \hat{t}_i^{(1)}) \leq \mathbb{P}(\hat{\mathbf{x}}_i \leq 0)$, where $\hat{\mathbf{x}}_i = \sum_{j \in \partial i} \mathbf{A}_{ij} \mathbf{y}_{j \rightarrow i}^{(1)}$. Assuming $t_i = +$, without loss of generality, \mathbf{A}_{ij} is $+1$ with probability $\mathbb{E}[\mathbf{p}]$ and -1 otherwise. All $\mathbf{y}_{j \rightarrow i}^{(1)}$ ’s are initialized as Gaussian random variables with mean one and variance one. All these random variables are independent of one another at this initial step. Hence, the resulting random variable $\hat{\mathbf{x}}_i$ is a sum of a shifted binomial random variable $2(\text{Binom}(\ell, \mathbb{E}[\mathbf{p}]) - \ell)$ and a zero-mean Gaussian random variable $\mathcal{N}(0, \ell)$. From calculations similar to (20), it follows that

$$\mathbb{E}\left[e^{\lambda \hat{\mathbf{x}}^{(1)}}\right] \leq e^{\ell\mu\lambda + \ell\lambda^2} \leq e^{-(1/4)\ell\mu^2},$$

where we choose $\lambda = -\mu/2$. By Chernoff’s inequality, this implies the lemma for any value of m .

4 Conclusion

We conclude with some limitations of our results and interesting research directions.

1. *More general models.* In this paper, we provided an order-optimal task assignment scheme and an order-optimal inference algorithm for that task assignment assuming a probabilistic crowdsourcing model. In this model, we assumed that each worker makes a mistake randomly according to a worker specific quality parameter. Two main simplifications we make here is that, first, the worker’s reliability does not depend on whether the task is a positive task or a negative task, and second, all the tasks are equally easy or difficult. The main remaining challenges in developing inference algorithms for crowdsourcing is how to develop a solution for more generic models formally described in Section 2.7. When workers exhibit bias and can have heterogeneous quality parameters depending on the correct answer to the task, spectral methods using low-rank matrix approximations nicely generalize to give an algorithmic solution. Also, it would be interesting to find algorithmic solutions with performance guarantees for the generic model where tasks difficulties are taken into account.

2. *Improving the constant.* We prove our approach is minimax optimal up to a constant factor. However, there might be another algorithm with better constant factor than our inference algorithm. Some modification of the expectation maximization or the belief propagation might achieve a better constant compared to our inference algorithm. It is an interesting research direction to find such an algorithm and give an upper bound on the error probability that is smaller than what we have in our main theorem.

3. *Instance-optimality.* The optimality of our approach is proved under the worst-case worker distribution. However, it is not known whether our approach is instance-optimal or not under the non-adaptive scenario. It would be important to prove lower bounds for all worker distributions or to find a counter example where another algorithm achieves a strictly better performance for a particular worker distribution in terms of the scaling of the required budget.

4. *Phase transition.* We empirically observe that there is a phase transition around $\hat{\ell}\hat{r}q^2 = 1$. Below this, no algorithm can do better than majority voting. This phase transition seems to be an algorithm-independent and fundamental property of the problem (and the random graph). It might be possible to formally prove the fundamental difference in the way information propagates under the crowdsourcing model. Such phase transition has been studied for a simpler model of broadcasting on trees in information theory and statistical mechanics [EKPS00].

References

- [AS08] N. Alon and J. H. Spencer, *The probabilistic method*, John Wiley, 2008.
- [BBMK11] M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger, *Crowds in two seconds: enabling realtime crowd-powered interfaces*, Proceedings of the 24th annual ACM symposium on User interface software and technology, UIST ’11, 2011, pp. 33–42.
- [BJJ⁺10] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh, *Vizwiz: nearly real-time answers to visual questions*, Proceedings of the 23rd annual ACM symposium on User interface software and technology, UIST ’10, 2010, pp. 333–342.

- [BLM⁺10] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, *Soylent: a word processor with a crowd inside*, Proceedings of the 23rd annual ACM symposium on User interface software and technology (New York, NY, USA), ACM UIST, 2010, pp. 313–322.
- [Bol01] B. Bollobás, *Random Graphs*, Cambridge University Press, January 2001.
- [CHMA10] L. B. Chilton, J. J. Horton, R. C. Miller, and S. Azenkot, *Task search in a human computation market*, Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP '10, 2010, pp. 1–9.
- [DCS09] P. Donmez, J. G. Carbonell, and J. Schneider, *Efficiently learning the accuracy of labeling sources for selective sampling*, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 259–268.
- [DS79] A. P. Dawid and A. M. Skene, *Maximum likelihood estimation of observer error-rates using the em algorithm*, Journal of the Royal Statistical Society. Series C (Applied Statistics) **28** (1979), no. 1, 20–28.
- [EHR11] Ş. Ertekin, H. Hirsh, and C. Rudin, *Approximating the wisdom of the crowd*, Proceedings of the Second Workshop on Computational Social Science and the Wisdom of Crowds (NIPS 2011), 2011.
- [EKPS00] W. Evans, C. Kenyon, Y. Peres, and L. J. Schulman, *Broadcasting on trees and the ising model*, The Annals of Applied Probability **10** (2000), no. 2, pp. 410–433.
- [FHI11] S. Faradani, B. Hartmann, and P. G. Ipeirotis, *What’s the right price? pricing tasks for finishing on time*, Human Computation’11, 2011.
- [Hol11] S. Holmes, *Crowd counting a crowd*, 2011, March 2011, Statistics Seminar, Stanford University.
- [Ipe10] P. G. Ipeirotis, *Analyzing the amazon mechanical turk marketplace*, XRDS **17** (2010), no. 2, 16–21.
- [JG03] R. Jin and Z. Ghahramani, *Learning with multiple labels*, Advances in neural information processing systems, 2003, pp. 921–928.
- [KOS11] D. R. Karger, S. Oh, and D. Shah, *Budget-optimal crowdsourcing using low-rank matrix approximations*, Proc. of the Allerton Conf. on Commun., Control and Computing, 2011.
- [Lan50] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of The National Bureau Of Standards **45** (1950), no. 4, 255–282.
- [LW89] N. Littlestone and M. K. Warmuth, *The weighted majority algorithm*, Foundations of Computer Science, 1989., 30th Annual Symposium on, oct 1989, pp. 256 –261.
- [MM09] M. Mezard and A. Montanari, *Information, physics, and computation*, Oxford University Press, Inc., New York, NY, USA, 2009.

- [MW10] W. Mason and D. J. Watts, *Financial incentives and the “performance of crowds”*, SIGKDD Explor. Newsl. **11** (2010), no. 2, 100–108.
- [Pea88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann Publ., San Mateo, California, 1988.
- [RU08] T. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, march 2008.
- [RY12] V. C. Raykar and S. Yu, *Eliminating spammers and ranking annotators for crowd-sourced labeling tasks*, J. Mach. Learn. Res. **13** (2012), 491–518.
- [RYZ⁺10a] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, *Learning from crowds*, J. Mach. Learn. Res. **99** (2010), 1297–1322.
- [RYZ⁺10b] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, L. Moy, and D. Blei, *Learning from crowds*, Journal of Machine Learning Research (2010), no. 11, 1297–1322.
- [SFB⁺95] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi, *Inferring ground truth from subjective labelling of venus images*, Advances in neural information processing systems, 1995, pp. 1085–1092.
- [SPI08] V. S. Sheng, F. Provost, and P. G. Ipeirotis, *Get another label? improving data quality and data mining using multiple, noisy labelers*, Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, ACM, 2008, pp. 614–622.
- [WBBP10] P. Welinder, S. Branson, S. Belongie, and P. Perona, *The multidimensional wisdom of crowds*, Advances in Neural Information Processing Systems, 2010, pp. 2424–2432.
- [WRW⁺09] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan, *Whose vote should count more: Optimal integration of labels from labelers of unknown expertise*, Advances in Neural Information Processing Systems, vol. 22, 2009, pp. 2035–2043.
- [WS67] G. Wyszecki and W. S. Stiles, *Color science: Concepts and methods, quantitative data and formulae*, Wiley-Interscience, 1967.
- [YFW03] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Understanding belief propagation and its generalizations*, pp. 239–269, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
- [YKG10] T. Yan, V. Kumar, and D. Ganesan, *Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones*, Proceedings of the 8th international conference on Mobile systems, applications, and services, MobiSys '10, 2010, pp. 77–90.
- [ZSD10] Y. Zheng, S. Scott, and K. Deng, *Active learning from multiple noisy labelers with varied costs*, Data Mining (ICDM), 2010 IEEE 10th International Conference on, dec. 2010, pp. 639–648.