Iterative Learning for Reliable Crowdsourcing Systems

David R. Karger **Sewoong Oh** **Devavrat Shah**
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
{karger,swoh,devavrat}@mit.edu

Abstract

Crowdsourcing systems, in which 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 crowdsourcers must devise schemes to increase confidence in their answers, typically by assigning each task multiple times and combining the answers in some way such as 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 significantly outperforms majority voting and, in fact, is asymptotically optimal through comparison to an oracle that knows the reliability of every worker.

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 Turk[1] provide 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).

Since typical crowdsourced tasks are tedious and the reward is small, 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 answers. Because the worker crowd is large, anonymous, and transient, 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 for your future tasks. 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

[1] http://www.mturk.com

[2] For certain high-value tasks, crowdsourcers can use entrance exams to "prequalify" workers and block spammers, but this increases the cost and still provides no guarantee that the prequalified workers will try hard.

voting. For such systems there is a natural core optimization problem to be solved. Assuming the taskmaster wishes to achieve a certain reliability in their answers, how can they do so at minimum cost (which is equivalent to asking how they 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 problems which makes inferences based on a fixed set of signals, our algorithm can choose which signals to measure by deciding which questions to ask which workers.

In the following, we will first define a formal model that captures these aspects of the problem. We will then describe a scheme for deciding which tasks to assign to which workers and introduce a novel iterative algorithm to infer the correct answers from the workers' responses. We show our algorithm is optimal for a broad range of parameters, by comparison to an *oracle* that knows the error probability of workers and can thus make optimal decisions.

Setup. We model a set of m tasks $\{t_i\}_{i\in[m]}$ as each being associated with an unobserved 'correct' answer $s_i \in \{\pm 1\}$. Here and after, we use $[N]$ to denote the set of first N integers. In the earlier image categorization example, each task corresponds to labeling an image as suitable for children $(+1)$ or not (-1). We assign these tasks to n workers from the crowd, which we denote by $\{w_j\}_{j\in[n]}$.

When a task is assigned to a worker, we get a possibly inaccurate answer from the worker. We use $A_{ij} \in \{\pm 1\}$ to denote the answer if task t_i is assigned to worker w_j. Some workers are more diligent or have more expertise than others, while some other workers might be spammers. We choose a simple model to capture this diversity in workers' reliability: we assume that each worker w_j is characterized by a reliability $p_j \in [0, 1]$, and that they make errors randomly on each question they answer. Precisely, if task t_i is assigned to worker w_j then

$$\mathbf{A}_{ij} \quad = \quad \begin{cases} s_i & \text{with probability } p_j \ , \\ -s_i & \text{with probability } 1 - p_j \ , \end{cases}$$

and $\mathbf{A}_{ij} = 0$ if t_i is not assigned to w_j. The random variable $\mathbf{A}_{ij}$ is independent of any other event given p_j. (Throughout this paper, we use boldface characters to denote random variables and random matrices unless it is clear from the context.) 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 further assume that the reliability of workers $\{\mathbf{p}_j\}_{j\in[n]}$ are independent and identically distributed random variables with a given distribution on $[0, 1]$. One example is *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, in which case $\mathbf{p}_j = 1$, and a spammer gives random answers, in which case $\mathbf{p}_j = 1/2$. Given this random variable $\mathbf{p}_j$, we define an important parameter $q \in [0, 1]$, which captures the 'average quality' of the crowd:

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

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. We will see later that our bound on the error rate of our inference algorithm holds for any distribution of $\mathbf{p}_j$ but depends on the distribution only through this parameter q. It is quite realistic to assume the existence of a prior distribution for $\mathbf{p}_j$. The model is therefore quite general: 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. 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 certain reliability.

Under this crowdsourcing model, a taskmaster first decides which tasks should be assigned to which workers, and then estimates the correct solutions $\{s_i\}_{i\in[m]}$ once all the answers $\{\mathbf{A}_{ij}\}$ are submitted. We assume a *one-shot scenario* in which all questions are asked simultaneously and then an estimation is performed after all the answers are obtained. In particular, we do not allow allocating tasks adaptively based on the answers received thus far. Then, assigning tasks to nodes amounts to designing a bipartite graph $G(\{t_i\}_{i\in[m]} \cup \{w_j\}_{j\in[n]}, E)$ with m task and n worker nodes. Each edge $(i,j) \in E$ indicates that task t_i was assigned to worker w_j.

Prior Work. A naive approach to identify the correct answer from multiple workers' responses is to use majority voting. Majority voting simply chooses what the majority of workers agree on. When there are many spammers, majority voting is error-prone since it weights all the workers equally. We will show that majority voting is provably sub-optimal and can be significantly improved upon.

To infer the answers of the tasks and also the reliability of workers, Dawid and Skene [1] proposed an algorithm based on expectation maximization (EM) [2]. This approach has also been applied in classification problems where the training data is annotated by low-cost noisy 'labelers' [3, 4]. In [5] and [6], this EM approach has been applied to more complicated probabilistic models for image labeling tasks. However, the performance of these approaches are only empirically evaluated, and there is no analysis that proves performance guarantees. In particular, EM algorithms require an initial starting point which is typically randomly guessed. The algorithm is highly sensitive to this initialization, making it difficult to predict the quality of the resulting estimate. The advantage of using low-cost noisy 'labelers' has been studied in the context of supervised learning, where a set of labels on a training set is used to find a good classifier. 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 [7] show that getting repeated labeling can give considerable advantage.

Contributions. In this work, we provide a rigorous treatment of designing a crowdsourcing system with the aim of minimizing the budget to achieve completion of a set of tasks with a certain reliability. We provide both an asymptotically optimal graph construction (random regular bipartite graph) and an asymptotically optimal algorithm for inference (iterative algorithm) on that graph. As the main result, we show that our algorithm performs as good as the best possible algorithm. The surprise lies in the fact that the optimality of our algorithm is established by comparing it with the best algorithm, one that is free to choose any graph, regular or irregular, and performs optimal estimation based on the information provided by an oracle about reliability of the workers. Previous approaches focus on developing inference algorithms assuming that a graph is already given. None of the prior work on crowdsourcing provides any systematic treatment of the graph construction. To the best of our knowledge, we are the first to study both aspects of crowdsourcing together and, more importantly, establish optimality.

Another novel contribution of our work is the analysis technique. The iterative 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 (see Section 3 for a definition) using recursion, and compute the parameters in a closed form. This allows us to prove the sharp result on the error rate, and this technique could be of independent interest for analyzing a more general class of algorithms.

2 Main result

Under the crowdsourcing model introduced, we want to design algorithms to assign tasks and estimate the answers. In what follows, we explain how to assign tasks using a random regular graph and introduce a novel iterative algorithm to infer the correct answers. We state the performance guarantees for our algorithm and provide comparisons to majority voting and an oracle estimator.

Task allocation. Assigning tasks amounts to designing a bipartite graph $G\big(\{t_i\}_{i\in[m]} \cup \{w_j\}_{j\in[n]}, E\big)$, where each edge corresponds to a task-worker assignment. The taskmaster makes a choice of how many workers to assign to each task (the left degree l) and how many tasks to assign to each worker (the right degree r). Since the total number of edges has to be consistent, the number of workers n directly follows from $ml = nr$. To generate an (l,r)-regular bipartite graph we use

a random graph generation scheme known as the *configuration model* in random graph literature [8, 9]. In principle, one could use arbitrary bipartite graph G for task allocation. However, as we show later in this paper, random regular graphs are sufficient to achieve order-optimal performance.

Inference algorithm. We introduce a novel iterative algorithm which operates on real-valued task messages $\{x_{i\to j}\}_{(i,j)\in E}$ and worker messages $\{y_{j\to i}\}_{(i,j)\in E}$. The worker messages are initialized as independent Gaussian random variables. At each iteration, the messages are updated according to the described update rule, where ∂i is the neighborhood of t_i. Intuitively, a worker message $y_{j\to i}$ represents our belief on how 'reliable' the worker j is, such that our final estimate is a weighted sum of the answers weighted by each worker's reliability: $\hat{s}_i = \mathrm{sign}(\sum_{j\in\partial i} A_{ij} y_{j\to i})$.

Iterative Algorithm

Input: E, $\{A_{ij}\}_{(i,j)\in E}$, $k_{\max}$
Output: Estimation $\hat{s}\big(\{A_{ij}\}\big)$

1: **For all** $(i,j)\in E$ **do**
  Initialize $y^{(0)}_{j\to i}$ with random $Z_{ij}\sim\mathcal{N}(1,1)$;
2: **For** $k=1,\ldots,k_{\max}$ **do**
  For all $(i,j)\in E$ **do** $x^{(k)}_{i\to j}\leftarrow\sum_{j'\in\partial i\setminus j} A_{ij'} y^{(k-1)}_{j'\to i}$;
  For all $(i,j)\in E$ **do** $y^{(k)}_{j\to i}\leftarrow\sum_{i'\in\partial j\setminus i} A_{i'j} x^{(k)}_{i'\to j}$;
3: **For all** $i\in[m]$ **do** $x_i\leftarrow\sum_{j\in\partial i} A_{ij} y^{(k_{\max}-1)}_{j\to i}$;
4: Output estimate vector $\hat{s}\big(\{A_{ij}\}\big)=[\mathrm{sign}(x_i)]$.

While our algorithm is inspired by the standard Belief Propagation (BP) algorithm for approximating max-marginals [10, 11], our algorithm is original and overcomes a few critical limitations of the standard BP. First, the iterative algorithm does not require any knowledge of the prior distribution of $\mathbf{p}_j$, whereas the standard BP requires the knowledge of the distribution. Second, there is no efficient way to implement standard BP, since we need to pass sufficeint statistics (or messages) which under our general model are distributions over the reals. On the otherhand, the iterative algorithm only passes messages that are real numbers regardless of the prior distribution of $\mathbf{p}_j$, which is easy to implement. Third, the iterative algorithm is provably asymptotically order-optimal. *Density evolution*, explained in detail in Section 3, is a standard technique to analyze the performance of BP. Although we can write down the density evolution for the standard BP, we cannot analyze the densities, analytically or numerically. It is also very simple to write down the density evolution equations for the iterative algorithm, but it is not trivial to analyze the densities in this case either. We develop a novel technique to analyze the densities and prove optimality of our algorithm.

2.1 Performance guarantee

We state the main analytical result of this paper: for random (l,r)-regular bipartite graph based task assignments with our iterative inference algorithm, the probability of error decays exponentially in lq, up to a universal constant and for a broad range of the parameters l, r and q. With a reasonable choice of $l=r$ and both scaling like $(1/q)\log(1/\epsilon)$, the proposed algorithm is guarantted to achieve error less than ϵ for any $\epsilon\in(0,1/2)$. Further, an algorithm independent lower bound that we establish suggests that such an error dependence on lq is unavoidable. Hence, in terms of the task allocation budget, our algorithm is order-optimal. The precise statements follow next. Let $\mu=\mathbb{E}[2\mathbf{p}_j-1]$ and recall $q=\mathbb{E}[(2\mathbf{p}_j-1)^2]$. To lighten the notation, let $\hat{l}\equiv l-1$ and $\hat{r}\equiv r-1$. Define

$$\rho_k^2 \equiv \frac{2q}{\mu^2(q^2\hat{l}\hat{r})^{k-1}} + \Big(3+\frac{1}{q\hat{r}}\Big)\frac{1-(1/q^2\hat{l}\hat{r})^{k-1}}{1-(1/q^2\hat{l}\hat{r})} .$$

For $q^2\hat{l}\hat{r}>1$, let $\rho_\infty^2\equiv\lim_{k\to\infty}\rho_k^2$ such that

$$\rho_\infty^2 = \big(3+(1/q\hat{r})\big)q^2\hat{l}\hat{r}/(q^2\hat{l}\hat{r}-1) .$$

Then we can show the following bound on the probability of making an error.

Theorem 2.1. *For fixed $l>1$ and $r>1$, assume that m tasks are assigned to $n=ml/r$ workers according to a random (l,r)-regular graph drawn from the configuration model. If the distribution*

of the worker reliaiblity satisfy $\mu \equiv \mathbb{E}[2\mathbf{p}_j - 1] > 0$ *and* $q^2 > 1/(\hat{l}\hat{r})$*, then for any* $s \in \{\pm 1\}^m$*, the estimates from* k *iterations of the iterative algorithm achieve*

$$\lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} \mathbb{P}\Big(s_i \neq \hat{s}_i\big(\{\mathbf{A}_{ij}\}_{(i,j)\in E}\big)\Big) \quad \leq \quad e^{-lq/(2\rho_k^2)} \,. \tag{1}$$

As we increase k, the above bound converges to a non-trivial limit.

Corollary 2.2. *Under the hypotheses of Theorem 2.1,*

$$\lim_{k\to\infty} \lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} \mathbb{P}\Big(s_i \neq \hat{s}_i\big(\{\mathbf{A}_{ij}\}_{(i,j)\in E}\big)\Big) \quad \leq \quad e^{-lq/(2\rho_\infty^2)} \,. \tag{2}$$

Even if we fix the value of $q = \mathbb{E}[(2\mathbf{p}_j - 1)^2]$, different distributions of $\mathbf{p}_j$ can have different values of μ in the range of $[q, \sqrt{q}]$. Surprisingly, the asymptotic bound on the error rate does not depend on μ. Instead, as long as q is fixed, μ only affects how fast the algorithm converges (cf. Lemma 2.3).

Notice that the bound in (2) is only meaningful when it is less than a half, whence $\hat{l}\hat{r}q^2 > 1$ and $lq > 6\log(2) > 4$. While as a task master the case of $\hat{l}\hat{r}q^2 < 1$ may not be of interest, for the purpose of completeness we comment on the performance of our algorithm in this 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. The proof is omitted due to a space constraint.

$$\lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} \mathbb{P}\Big(s_i \neq \hat{s}_i\big(\{\mathbf{A}_{ij}\}_{(i,j)\in E}\big)\Big) \quad \leq \quad e^{-l\mu^2/4} \,. \tag{3}$$

2.2 Discussion

Here we make a few comments relating to the execution of the algorithm and the interpretation of the main results. First, the iterative algorithm is efficient with runtime comparable to the simple majority voting which requires $O(ml)$ operations.

Lemma 2.3. *Under the hypotheses of Theorem 2.1, the total* computational cost *sufficient to achieve the bound in Corollary 2.2 up to any constant factor in the exponent is* $O(ml\log(q/\mu^2)/\log(q^2\hat{l}\hat{r}))$.

By definition, we have $q \leq \mu \leq \sqrt{q}$. The runtime is the worst when $\mu = q$, which happens under the spammer-hammer model, and it is the best when $\mu = \sqrt{q}$ which happens if $\mathbf{p}_j = (1 + \sqrt{q})/2$ deterministically. There exists a (non-iterative) polynomial time algorithm with runtime independent of q for computing the estimate which achieves (2), but in practice we expect that the number of iterations needed is small enough that the iterative algorithm will outperform this non-iterative algorithm. Detailed proof of Lemma 2.3 will be skipped here due to a space constraint.

Second, 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 $\{s_i\}_{i\in[m]}$ and workers with $\{p_j\}_{j\in[n]}$ or tasks with $\{-s_i\}_{i\in[m]}$ and workers with $\{1 - p_j\}_{j\in[n]}$. Statistically, both of them give the same output. 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.

Third, our algorithm does not require any information on the distribution of $\mathbf{p}_j$. Further, unlike other EM based algorithms, the iterative algorithm is not sensitive to initialization and with random initialization converges to a unique estimate with high probability. This follows from the fact that the algorithm is essentially computing a leading eigenvector of a particular linear operator.

2.3 Relation to singular value decomposition

The leading singular vectors are often used to capture the important aspects of datasets in matrix form. In our case, the leading left singular vector of A can be used to estimate the correct answers, where $A \in \{0, \pm 1\}^{m\times n}$ is the $m \times n$ adjacency matrix of the graph G weighted by the submitted

answers. We can compute it using power iteration: for $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, starting with a randomly initialized v, power iteration iteratively updates u and v according to

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

It is known that normalized u converges exponentially to the leading left singular vector. This update rule is very similar to that of our iterative algorithm. But there is one difference that is crucial in the analysis: in our algorithm we follow the framework of the celebrated belief propagation algorithm [10, 11] and exclude the incoming message from node j when computing an outgoing message to j. This extrinsic nature of our algorithm and the locally tree-like structure of sparse random graphs [8, 12] allow us to perform asymptotic analysis on the average error rate. In particular, if we use the leading singular vector of A to estimate s, such that $s_i = \text{sign}(u_i)$, then existing analysis techniques from random matrix theory does not give the strong performance guarantee we have. These techniques typically focus on understanding how the subspace spanned by the top singular vector behaves. To get a sharp bound, we need to analyze how each entry of the leading singular vector is distributed. We introduce the iterative algorithm in order to precisely characterize how each of the decision variable x_i is distributed. Since the iterative algorithm introduced in this paper is quite similar to power iteration used to compute the leading singular vectors, this suggests that our analysis may shed light on how to analyze the top singular vectors of a sparse random matrix.

2.4 Optimality of our algorithm

As a taskmaster, the natural core optimization problem of our concern is how to achieve a certain reliability in our answers with minimum cost. Since we pay equal amount for all the task assignments, the cost is proportional to the total number of edges of the graph G. Here we compute the total budget sufficient to achieve a target error rate using our algorithm and show that this is within a constant factor from the necessary budget to achieve the given target error rate using any graph and the best possible inference algorithm. The order-optimality is established with respect to all algorithms that operate in *one-shot*, i.e. all task assignments are done simultaneously, then an estimation is performed after all the answers are obtained. The proofs of the claims in this section are skipped here due to space limitations.

Formally, consider a scenario where there are m tasks to complete and a target accuracy $\epsilon \in (0, 1/2)$. To measure accuracy, we use the average probability of error per task denoted by $d_m(s, \hat{s}) \equiv (1/m) \sum_{i \in [m]} \mathbb{P}(s_i \neq \hat{s}_i)$. We will show that $\Omega\big((1/q)\log(1/\epsilon)\big)$ assignments per task is necessary and sufficient to achieve the target error rate: $d_m(s, \hat{s}) \leq \epsilon$. To establish this fundamental limit, we use the following minimax bound on error rate. Consider the case where nature chooses a set of correct answers $s \in \{\pm 1\}^m$ and a distribution of the worker reliability $\mathbf{p}_j \sim f$. The distribution f is chosen from a set of all distributions on $[0, 1]$ which satisfy $\mathbb{E}_f[(2\mathbf{p}_j - 1)^2] = q$. We use $\mathcal{F}(q)$ to denote this set of distributions. Let $\mathcal{G}(m, l)$ denote the set of all bipartite graphs, including irregular graphs, that have m task nodes and ml total number of edges. Then the minimax error rate achieved by the best possible graph $G \in \mathcal{G}(m, l)$ using the best possible inference algorithm is at least

$$\inf_{ALGO, G \in \mathcal{G}(m,l)} \sup_{s, f \in \mathcal{F}(q)} d_m(s, \hat{s}_{G,ALGO}) \ \geq \ (1/2) e^{-(lq + O(lq^2))} \ , \tag{4}$$

where $\hat{s}_{G,ALGO}$ denotes the estimate we get using graph G for task allocation and algorithm $ALGO$ for inference. This minimax bound is established by computing the error rate of an oracle esitimator, which makes an optimal decision based on the information provided by an oracle who knows how reliable each worker is. Next, we show that the error rate of majority voting decays significantly slower: the leading term in the error exponent scales like $-lq^2$. Let $\hat{s}_{\text{MV}}$ be the estimate produced by majority voting. Then, for $q \in (0, 1)$, there exists a numerical constant C_1 such that

$$\inf_{G \in \mathcal{G}(m,l)} \sup_{s, f \in \mathcal{F}(q)} d_m(s, \hat{s}_{\text{MV}}) \quad = \quad e^{-(C_1 lq^2 + O(lq^4 + 1))} \ . \tag{5}$$

The lower bound in (4) does not depend on how many tasks are assigned to each worker. However, our main result depends on the value of r. We show that for a broad range of parameters l, r, and q our algorithm achieves optimality. Let $\hat{s}_{\text{Iter}}$ be the estimate given by random regular graphs and the iterative algorithm. For $\hat{l}q \geq C_2$, $\hat{r}q \geq C_3$ and $C_2 C_3 > 1$, Corollary 2.2 gives

$$\lim_{m \to \infty} \sup_{s, f \in \mathcal{F}(q)} d_m(s, \hat{s}_{\text{Iter}}) \quad \leq \quad e^{-C_4 lq} \ . \tag{6}$$

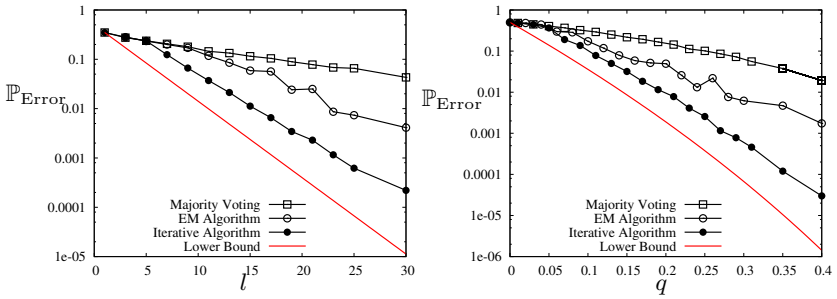

Figure 1: The iterative algorithm improves over majority voting and EM algorithm [7].

This is also illustrated in Figure 1. We ran numerical experiments with 1000 tasks and 1000 workers from the spammer-hammer model assigned according to random graphs with $l = r$ from the configuration model. For the left figure, we fixed $q = 0.3$ and for the right figure we fixed $l = 25$.

Now, let Δ_{LB} be the minimum cost per task *necessary* to achieve a target accuracy $\epsilon \in (0, 1/2)$ using any graph and any possible algorithm. Then (4) implies $\Delta_{\text{LB}} \sim (1/q)\log(1/\epsilon)$, where $x \sim y$ indicates that x scales as y. Let Δ_{Iter} be the minimum cost per task *sufficient* to achieve a target accuracy ϵ using our proposed algorithm. Then from (6) we get $\Delta_{\text{Iter}} \sim (1/q)\log(1/\epsilon)$. This establishes the order-optimality of our algorithm. It is indeed surprising that regular graphs are sufficient to achieve this optimality. Further, let Δ_{Majority} be the minimum cost per task *necessary* to achieve a target accuracy ϵ using the Majority voting. Then $\Delta_{\text{Majority}} \sim (1/q^2)\log(1/\epsilon)$, which significantly more costly than the optimal scaling of $(1/q)\log(1/\epsilon)$ of our algorithm.

3 Proof of Theorem 2.1

By symmetry, we can assume all s_i's are $+1$. If I is a random integer drawn uniformly in $[m]$, then $(1/m)\sum_{i\in[m]} \mathbb{P}(s_i \neq \hat{s}_i) \leq \mathbb{P}(x_I^{(k)} \leq 0)$, where $x_i^{(k)}$ denotes the decision variable for task i after k iterations of the iterative algorithm. Asymptotically, for a fixed k, l and r, the local neighborhood of x_I converges to a regular tree. To analyze $\lim_{m\to\infty} \mathbb{P}(x_I^{(k)} \leq 0)$, we use a standard probabilistic analysis technique known as 'density evolution' in coding theory or 'recursive distributional equations' in probabilistic combinatorics [8, 12]. Precisely, we use the following equality that in the large system limit,

$$\lim_{m\to\infty} \mathbb{P}(x_I^{(k)} \leq 0) = \mathbb{P}(\hat{\mathbf{x}}^{(k)} \leq 0)\ , \tag{7}$$

where $\hat{\mathbf{x}}^{(k)}$ is defined through density evolution equations (8) and (9) in the following.

Density evolution. In the large system limit as $m \to \infty$, the (l, r)-regular random graph locally converges in distribution to a (l, r)-regular tree. Therefore, for a randomly chosen edge (i, j), the messages $x_{i\to j}$ and $y_{j\to i}$ converge in distribution to $\mathbf{x}$ and $\mathbf{y}_{p_j}$ defined in the following *density evolution equations* (8). Here and after, we drop the superscript k denoting the iteration number whenever it is clear from the context. We initialize $\mathbf{y}_p$ with a Gaussian distribution independent of p: $\mathbf{y}_p^{(0)} \sim \mathcal{N}(1, 1)$. Let $\stackrel{d}{=}$ denote equality in distribution. Then, for $k \in \{1, 2, \ldots\}$,

$$\mathbf{x}^{(k)} \stackrel{d}{=} \sum_{i\in[l-1]} \mathbf{z}_{\mathbf{p}_i,i}\mathbf{y}_{\mathbf{p}_i,i}^{(k-1)}\ , \quad \mathbf{y}_p^{(k)} \stackrel{d}{=} \sum_{j\in[r-1]} \mathbf{z}_{p,j}\mathbf{x}_j^{(k)}\ , \tag{8}$$

where $\mathbf{x}_j$'s, $\mathbf{p}_i$'s, and $\mathbf{y}_{p,i}$'s are independent copies of $\mathbf{x}$, $\mathbf{p}$, and $\mathbf{y}_p$, respectively. Also, $\mathbf{z}_{p,i}$'s and $\mathbf{z}_{p,j}$'s are independent copies of $\mathbf{z}_p$. $\mathbf{p} \in [0, 1]$ is a random variable distributed according to the distribution of the worker's quality. $\mathbf{z}_{p,j}$'s and $\mathbf{x}_j$'s are independent. $\mathbf{z}_{\mathbf{p}_i,i}$'s and $\mathbf{y}_{\mathbf{p}_i,i}$'s are conditionally independent conditioned on $\mathbf{p}_i$. Finally, $\mathbf{z}_p$ is a random variable which is $+1$ with probability p and -1 with probability $1 - p$. Then, for a randomly chosen I, the decision variable $x_I^{(k)}$ converges in distribution to

$$\hat{\mathbf{x}}^{(k)} \stackrel{d}{=} \sum_{i\in[l]} \mathbf{z}_{\mathbf{p}_i,i}\mathbf{y}_{\mathbf{p}_i,i}^{(k-1)}\ . \tag{9}$$

Analyzing the density. Our strategy to provide an 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 σ if for all $\lambda \in \mathbb{R}$ the following inequality holds: $\mathbb{E}[e^{\lambda \mathbf{z}}] \leq e^{m\lambda + (1/2)\sigma^2\lambda^2}$. Define $\sigma_k^2 \equiv 2\hat{l}(\hat{l}\hat{r})^{k-1} + \mu^2\hat{l}^3\hat{r}(3q\hat{r}+1)(q\hat{l}\hat{r})^{2k-4}(1-(1/q^2\hat{l}\hat{r})^{k-1})/(1-(1/q^2\hat{l}\hat{r}))$ and $m_k \equiv \mu\hat{l}(q\hat{l}\hat{r})^{k-1}$ for $k \in \mathbb{Z}$. We will first show that, $\mathbf{x}^{(k)}$ is sub-Gaussian with mean m_k and parameter σ_k^2 for a regime of λ we are interested in. Precisely, we will show that for $|\lambda| \leq 1/(2m_{k-1}\hat{r})$,

$$\mathbb{E}[e^{\lambda \mathbf{x}^{(k)}}] \;\leq\; e^{m_k\lambda + (1/2)\sigma_k^2\lambda^2} \;. \tag{10}$$

By definition, due to distributional independence, we have $\mathbb{E}[e^{\lambda\hat{\mathbf{x}}^{(k)}}] = \mathbb{E}[e^{\lambda\mathbf{x}^{(k)}}]^{(l/\hat{l})}$. Therefore, it follows from (10) that $\hat{\mathbf{x}}^{(k)}$ satisfies $\mathbb{E}[e^{\lambda\hat{\mathbf{x}}^{(k)}}] \leq e^{(l/\hat{l})m_k\lambda + (l/2\hat{l})\sigma_k^2\lambda^2}$. Applying the Chernoff bound with $\lambda = -m_k/(\sigma_k^2)$, we get

$$\mathbb{P}\big(\hat{\mathbf{x}}^{(k)} \leq 0\big) \;\leq\; \mathbb{E}\big[e^{\lambda\hat{\mathbf{x}}^{(k)}}\big] \;\leq\; e^{-l\,m_k^2/(2\,\hat{l}\,\sigma_k^2)} \;, \tag{11}$$

Since $m_k m_{k-1}/(\sigma_k^2) \leq \mu^2\hat{l}^2(q\hat{l}\hat{r})^{2k-3}/(3\mu^2 q\hat{l}^3\hat{r}^2(q\hat{l}\hat{r})^{2k-4}) = 1/(3\hat{r})$, it is easy to check that $|\lambda| \leq 1/(2m_{k-1}\hat{r})$. Substituting (11) in (7), this finishes the proof of Theorem 2.1.

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

$$\mathbb{E}\big[e^{\lambda\mathbf{x}^{(k)}}\big] \;=\; \Big(\mathbb{E}_{\mathbf{p}}\Big[\mathbf{p}\mathbb{E}[e^{\lambda\mathbf{y}_{\mathbf{p}}^{(k-1)}}|\mathbf{p}] + \bar{\mathbf{p}}\mathbb{E}[e^{-\lambda\mathbf{y}_{\mathbf{p}}^{(k-1)}}|\mathbf{p}]\Big]\Big)^{\hat{l}} \;, \tag{12}$$

$$\mathbb{E}\big[e^{\lambda\mathbf{y}_p^{(k)}}\big] \;=\; \Big(p\mathbb{E}\big[e^{\lambda\mathbf{x}^{(k)}}\big] + \bar{p}\mathbb{E}\big[e^{-\lambda\mathbf{x}^{(k)}}\big]\Big)^{\hat{r}} \;, \tag{13}$$

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 $m_1 = \mu\hat{l}$ and parameter $\sigma_1^2 = 2\hat{l}$, 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 (12), we get for any λ, $\mathbb{E}[e^{\lambda\mathbf{x}^{(1)}}] = (\mathbb{E}[\mathbf{p}]e^{\lambda} + (1-\mathbb{E}[\mathbf{p}])e^{-\lambda})^{\hat{l}}e^{(1/2)\hat{l}\lambda^2} \leq e^{\hat{l}\mu\lambda + \hat{l}\lambda^2}$, 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. [13, Lemma A.1.5]).

Next, assuming $\mathbb{E}[e^{\lambda\mathbf{x}^{(k)}}] \leq e^{m_k\lambda + (1/2)\sigma_k^2\lambda^2}$ for $|\lambda| \leq 1/(2m_{k-1}\hat{r})$, we show that $\mathbb{E}[e^{\lambda\mathbf{x}^{(k+1)}}] \leq e^{m_{k+1}\lambda + (1/2)\sigma_{k+1}^2\lambda^2}$ for $|\lambda| \leq 1/(2m_k\hat{r})$, and compute appropriate m_{k+1} and σ_{k+1}^2. Substituting the bound $\mathbb{E}[e^{\lambda\mathbf{x}^{(k)}}] \leq e^{m_k\lambda + (1/2)\sigma_k^2\lambda^2}$ in (13), we get $\mathbb{E}[e^{\lambda\mathbf{y}_p^{(k)}}] \leq (pe^{m_k\lambda} + \bar{p}e^{-m_k\lambda})^{\hat{r}}e^{(1/2)\hat{r}\sigma_k^2\lambda^2}$. Further applying this bound in (12), we get

$$\mathbb{E}\Big[e^{\lambda\mathbf{x}^{(k+1)}}\Big] \;\leq\; \Big(\mathbb{E}_{\mathbf{p}}\Big[\mathbf{p}(\mathbf{p}e^{m_k\lambda} + \bar{\mathbf{p}}e^{-m_k\lambda})^{\hat{r}} + \bar{\mathbf{p}}(\mathbf{p}e^{-m_k\lambda} + \bar{\mathbf{p}}e^{m_k\lambda})^{\hat{r}}\Big]\Big)^{\hat{l}}\, e^{(1/2)\hat{l}\hat{r}\sigma_k^2\lambda^2} \;. \tag{14}$$

To bound the first term in the right-hand side, we use the next key lemma.

Lemma 3.1. *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*

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

For the proof, we refer to the journal version of this paper. Applying this inequality to (14) gives $\mathbb{E}[e^{\lambda\mathbf{x}^{(k+1)}}] \leq e^{q\hat{l}\hat{r}m_k\lambda + (1/2)\left((3q\hat{l}\hat{r}^2+\hat{l}\hat{r})m_k^2 + \hat{l}\hat{r}\sigma_k^2\right)\lambda^2}$, for $|\lambda| \leq 1/(2m_k\hat{r})$. In the regime where $q\hat{l}\hat{r} \geq 1$ as per our assumption, m_k is non-decreasing in k. At iteration k, the above recursion holds for $|\lambda| \leq \min\{1/(2m_1\hat{r}), \ldots, 1/(2m_{k-1}\hat{r})\} = 1/(2m_{k-1}\hat{r})$. Hence, we get a recursion for m_k and σ_k such that (10) holds for $|\lambda| \leq 1/(2m_{k-1}\hat{r})$:

$$m_{k+1} \;=\; q\hat{l}\hat{r}m_k \;, \qquad \sigma_{k+1}^2 \;=\; (3q\hat{l}\hat{r}^2 + \hat{l}\hat{r})m_k^2 + \hat{l}\hat{r}\sigma_k^2 \;.$$

With the initialization $m_1 = \mu\hat{l}$ and $\sigma_1^2 = 2\hat{l}$, we have $m_k = \mu\hat{l}(q\hat{l}\hat{r})^{k-1}$ for $k \in \{1,2,\ldots\}$ and $\sigma_k^2 = a\sigma_{k-1}^2 + bc^{k-2}$ for $k \in \{2,3,\ldots\}$, with $a = \hat{l}\hat{r}$, $b = \mu^2\hat{l}^2(3q\hat{l}\hat{r}^2 + \hat{l}\hat{r})$, and $c = (q\hat{l}\hat{r})^2$. After some algebra, it follows that $\sigma_k^2 = \sigma_1^2a^{k-1} + bc^{k-2}\sum_{\ell=0}^{k-2}(a/c)^{\ell}$. For $\hat{l}\hat{r}q^2 \neq 1$, we have $a/c \neq 1$, whence $\sigma_k^2 = \sigma_1^2a^{k-1} + bc^{k-2}(1-(a/c)^{k-1})/(1-a/c)$. This finishes the proof of (10).

References

[1] 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(1):20–28, 1979.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38, 1977.

[3] R. Jin and Z. Ghahramani. Learning with multiple labels. *Advances in neural information processing systems*, pages 921–928, 2003.

[4] 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.*, 99:1297–1322, August 2010.

[5] 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*, 22:2035–2043, 2009.

[6] P. Welinder, S. Branson, S. Belongie, and P. Perona. The Multidimensional Wisdom of Crowds. pages 2424–2432, 2010.

[7] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 614–622. ACM, 2008.

[8] T. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, march 2008.

[9] B. Bollobás. *Random Graphs*. Cambridge University Press, January 2001.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publ., San Mateo, Califonia, 1988.

[11] J. S. Yedidia, W. T. Freeman, and Y. Weiss. *Understanding belief propagation and its generalizations*, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[12] M. Mezard and A. Montanari. *Information, Physics, and Computation*. Oxford University Press, Inc., New York, NY, USA, 2009.

[13] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley, 2008.