Reliable Multiple-choice Iterative Algorithm for Crowdsourcing Systems

Donghyeon Lee
Seoul National University
Seoul, Korea
donghyeon@snu.ac.kr

Joonyoung Kim
Seoul National University
Seoul, Korea
kimjymcl@snu.ac.kr

Hyunmin Lee
Seoul National University
Seoul, Korea
kakamin0@snu.ac.kr

Kyomin Jung
Seoul National University
Seoul, Korea
kjung@snu.ac.kr

ABSTRACT

The appearance of web-based crowdsourcing systems gives a promising solution to exploiting the wisdom of crowds efficiently in a short time with a relatively low budget. Despite their efficiency, crowdsourcing systems have an inherent problem in that responses from workers can be unreliable since workers are low-paid and have low responsibility. Although simple majority voting can be a solution, various research studies have sought to aggregate noisy responses to obtain greater reliability in results through effective techniques such as Expectation-Maximization (EM) based algorithms. While EM-based algorithms get the limelight in crowdsourcing systems due to their useful inference techniques, Karger et al. [8, 9] made a significant breakthrough by proposing a novel iterative algorithm based on the idea of low-rank matrix approximations and the message passing technique. They showed that the performance of their iterative algorithm is order-optimal, which outperforms majority voting and EM-based algorithms. However, their algorithm is not always applicable in practice since it can only be applied to binary-choice questions. Recently, they devised an inference algorithm for multi-class labeling [10], which splits each task into a bunch of binary-choice questions and exploits their existing algorithm. However, it has difficulty in combining into real crowdsourcing systems since it over-exploits redundancy in that each split question should be queried in multiple times to obtain reliable results.

In this paper, we design an iterative algorithm to infer true answers for multiple-choice questions, which can be directly applied to real crowdsourcing systems. Our algorithm can also be applicable to short-answer questions as well. We analyze the performance of our algorithm, and prove that the error bound decays exponentially. Through extensive experiments, we verify that our algorithm outperforms majority voting and EM-based algorithm in accuracy.

SIGMETRICS'15, June 15–19, 2015, Portland, OR, USA.

http://dx.doi.org/10.1145/2745844.2745871.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Statistical Computing; F.2 [**Analysis of Algorithms and Problem Complexity**]: Miscellaneous

General Terms

Algorithms, Performance

Keywords

Crowdsourcing; Resource Allocation; Multiple-choice; Iterative Learning

1. INTRODUCTION

Crowdsourcing has become one of the cornerstones of research in the development of human computation-based intelligence systems. New web-based services such as Amazon Mechanical Turk have arisen and become popular, as they can provide ideal solutions, gathering enormous responses from widespread crowds in a short time with a relatively low budget [13, 14]. For example, ImageNet, a large-scale image database, was a successful project that exploited the idea of crowdsourcing to label 3.2 million images hierarchically [3].

Despite the innovative framework of crowdsourcing systems, responses from workers can be unreliable [7, 12, 17, 18], since workers hired by crowdsourcing systems are low-paid and have low responsibility. Therefore, extensive works have been proceeded to find reliable solutions that infer the true answers from noisy responses. One natural method for aggregating responses is majority voting. But due to its simplicity, Expectation-Maximization (EM)-based algorithms have become popular. Since EM-based algorithms can deal with inference problems with latent variables and unknown model parameters, researchers applied the EM algorithm to proper graphical models for crowdsourcing systems, and showed that their results generally outperform those of majority voting [16, 20, 21]. Recently, Karger et al. [8, 9] made a significant breakthrough by proposing a novel iterative algorithm based on the idea of low-rank matrix approximations and the message passing technique. They showed that the performance of their iterative algorithm is order-optimal, which outperforms majority voting and EM-based algorithms.

Major research studies in this field have concentrated on cases with binary answers, yes (+1) or no (-1) [9, 21]. One example of such a binary case would be when workers have to determine whether a given image is suitable for children. However, real crowdsourced data posted on Amazon Mechanical Turk usually consists of multiple-choice questions and short-answer questions, so more general inference techniques should be employed.

In this paper, we focus on a more general structure for crowdsourcing systems that can be applied to multiple-choice questions. Note that we consider multiple-choice questions in which all choices are independent from each other. Independent multiple choices differ from linearly ordered choices which are commonly used in rating systems. For example, classifying types of cancers in patients is appropriate for an independent multiple-choice case, whereas determining the stage of a specific cancer of a patient is adequate for a linearly ordered choices case. We are currently focusing on the former case, which has greater applicability in D-ary classification problems. Moreover, we do not make a restriction on variation in the number of choices for each multiple-choice question. In addition, we suggest a method to transform short-answer questions into several multiple-choice questions so that our algorithm can be applied.

Our algorithm iteratively computes relative reliability of each worker in a novel way, where relative reliability is exploited as a weight of the worker's responses. Our algorithm also gets reliable results rapidly with small error compared to majority voting or EM-based algorithms. One of our main contributions is the performance guarantee of our algorithm by proving that the error bound of our algorithm decays exponentially. An interesting aspect of the error bound is its dependency on the negative entropy of workers in a perspective on information theory. Naturally, it is reasonable to assume that the true answers can be revealed by how much information there is in the workers' responses. We verify the performance of our algorithm through numerical experiments on various cases, which is close to that of oracle estimator. We also verify that our algorithm can infer relative reliability of workers almost correctly by experiments

Moreover, we addressed a strategy to gain responses with greater reliability from diligent workers in an adaptive manner. In this strategy, some pilot tasks chosen from whole tasks can be exploited to assess the expertise of the crowds. Note that we consider pilot tasks that differ from golden standard units. The relative reliability of workers can be estimated through given pilot tasks by applying our algorithm. In other words, we can initially assess workers' reliability with a small number of tasks, even if their true answers are unknown. Since the relative reliability of workers are estimated by managing the number of tasks each worker is given, we can expect to get responses with greater reliability for the same budget in an efficient way. Since our algorithm generally converges rapidly, our work can be combined to the context of online learning, which is more realistic setting for crowdsourcing systems.

The paper is organized as follows: We discuss related work in Section 1.1. In Section 2, we make a setup, and we describe our algorithm to infer the true answers for multiple-choice questions in Section 3. Then, we look into some applications in Section 4 and provide performance guarantees for our algorithm in Section 5. In Section 6, we present comparative results through numerical experiments, and we draw conclusions in Section 7.

1.1 Related Work

A common, intuitive strategy for aggregating responses is majority voting, which is widely used in real life due to its simplicity. However, in crowdsourcing systems, this simple inference technique has several limitations, since it assumes all workers have an equal level of expertise, and it gives the same weight to all responses. In general, there are unreliable workers such as novices or free money collectors, and even adversarial workers can be shown, so majority voting has obvious weak points when workers are unreliable [17].

There have been various approaches to trying to improve the reliability of results from unreliable responses. Two key ideas are introducing latent variables and estimating results by an iterative algorithm known as the EM algorithm. Dawid and Skene [2] exploited these ideas when they developed a simple probabilistic model using confusion matrices for each labeler as latent variables. They proposed an iterative algorithm based on EM to infer ground truth from unreliable responses.

Since the EM algorithm has an effective procedure to evaluate missing or hidden data and performs quite well, this model has been generalized and extended by several researchers. The GLAD model [21] combines the implicit characteristics of tasks and workers. Responses from workers are determined by several factors, such as the difficulty of the task, the expertise of the labeler, and the true label. The EM-based model can operate flexibly on various cases by introducing extra latent variables, which can be represented as the natural properties of tasks and workers [20]. Another variant proposed by Raykar et al. [16] considers a proper classifier for crowdsourcing systems, and aims to learn the classifier and the ground truth together.

Despite its popularity, there are some arguments in existing EM algorithms. The main thing is lack of intensive analysis about performance guarantees since their performance is only empirically evaluated in most cases. Another point is that inference techniques based on EM algorithms are not scalable. If the data size increases, EM-based algorithms become inefficient and degenerate, because their time and space requirements grow exponentially. Moreover, designing model-based EM algorithms with greater complexity leads to the introduction of an increased number of latent variables and model parameters. Apart from the computational complexity problem, the performance of EM-based algorithms could degenerate due to the initialization problem, even though it is designed to be a more complex model. Alternative approaches have been suggested by Karger et al. [8] in the context of spectral methods that use low-rank matrix approximations. They treated the data matrix A which involves workers' responses perturbed by a random noise. The true answers can be approximated by a rank-1 matrix, of which the singular vector reflects the correct answer of the tasks. When the spectral radius of the signal matrix outweighs the spectral radius of the random noise matrix, the correct answers can be extracted by the singular vector of the data matrix A. Using the power iteration method, the top singular vector can be obtained more efficiently compared to the computation complexity of EM-based algorithms.

They also proposed a novel iterative learning algorithm [9] that learns the likelihood of candidate answers and the

reliability of workers. It is inspired by the standard *Belief Propagation* (BP) algorithm, which approximates the maximal marginal distribution of variables. This message passing algorithm achieves almost the same results as the previous spectral method, but they provide novel analysis techniques such as *Density Evolution* in coding theory to improve the error bound more tightly, which decays exponentially. Although they did not assume any prior knowledge, Liu et al. [15] shows that choosing a suitable prior can improve the performance via a Bayesian approach.

Recently, Karger et al. [10] focused on multi-class labeling based on their existing novel algorithms, but their strategy for multi-class labeling is well suited to the linearly ordered choices, not independent multiple choices. By converting each multiple-choice question into a bunch of binary-choice questions, they could exploit the existing algorithms to determine true answers of multiple-choice questions. Although this strategy can be extended to independent multiple choices, it overexploits redundancy since each task should be split and queried in multiple times to obtain reliable results. Furthermore, in real crowdsourcing systems, it is natural that workers solve intact multiple-choice questions rather than split binary-choice questions. Therefore it has difficulty in combining into real crowdsourcing systems.

On top of the problem inferring the true answers, proper adaptive strategies are developed to utilize reliable workers when they are reusable. [4, 5, 6, 22] showed that the performance can be significantly improved through exploration/exploitation approaches.

2. SETUP

In this section, we define some variables and notations for problem formulation. Consider a set of m tasks, each of which can be a multiple-choice question that only has one correct answer. The number of choices for task i is denoted D_i. All tasks are distributed to several workers through a proper task allocation strategy.

Suppose that n workers participate to perform m tasks. We consider a probabilistic model to generate responses when workers face tasks. We assume that a worker j is parameterized by a latent variable $p_j \in [0, 1]$, which represents the probability of getting a correct answer. In other words, each worker gives the correct answer with a probability p_j and the wrong answer with probability $1 - p_j$ in the decision-making process. When a worker gives a wrong answer, we can assume that the worker has chosen one of distractors uniformly at random, so the probability of each wrong choice is $\frac{1-p_j}{D_i-1}$. It is reasonable that this latent variable p_j refers to the reliability of the worker, since it captures the ability or diligence of the worker.

In the response process, when a worker j solves an assigned task i, we define the submitted response $\vec{A}^{ij}$ in vector form. The response is represented as a D_i-dimensional binary unit vector $\vec{A}^{ij}$, having 1-of-D_i representation in which the element indicating the chosen answer is equal to 1 and all other elements are equal to 0. The values of A_d^{ij} therefore satisfy $A_d^{ij} \in \{0, 1\}$ and $\sum_d A_d^{ij} = 1$ where A_d^{ij} is the d^{th} component of the response $\vec{A}^{ij}$. For example, when there are three choices, the possible answer forms are $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. Our goal is to determine the correct answer for each task by querying and aggregating all the responses from the workers.

3. ALGORITHM

In this section, we propose our multiple-iterative algorithm with a minimum number of assignments. In advance, using random regular bipartite graph-generating model, we emulate a real crowdsourcing system scenario. Then, the message update rules of our iterative algorithm are explained. In addition, we propose the generalized iterative algorithm for general setting such as a adaptive strategy.

3.1 Task Allocation

To design a graph model for a crowdsourcing system, we use a bipartite graph which consists of two types of node sets. m tasks are defined as the set of nodes $[m]$ at the left side of the graph, and n workers are defined as the set of nodes $[n]$ at the right side respectively. Each edge represents an assignment between a task and a worker and this is determined according to the task assignment method. For simplicity, the i^{th} task and the j^{th} worker are denoted as i and j respectively. Given a bipartite graph $G = \{[m] \cup [n], E\}$ representing the allocation graph between tasks and workers, we connect the edge (i, j) if task i is assigned to worker j. We decide the task node degree l in proportion to the resources we can spend. In addition, the worker node degree r is determined by the work capacity that an individual worker can manage. Since we recruit workers through open-call, the (l, r) regular bipartite graph is adequate for our setting. To generate a (l, r) random regular bipartite graph such that $ml = nr$, we bring a simple random construction model known as the *pairing model*(This is also called a configuration model in [9]). In fact, any arbitrary bipartite graph instance can be used for task allocation. However, we will use the *pairing model* which generates a random bipartite graph with a local tree-like property. Using this property, we prove the tight error bounds of our algorithm in Section 5.3.

3.2 Multiple Iterative Algorithm

In this section, we describe the basic operations of our algorithm and the process of inferring true answers. For each edge (i, j), the response is denoted as $\vec{A}^{ij} \in U = \{\vec{e}_u | u \in [1 : D_i]\}$ which consists of D dimensional binary unit vectors all of whose components are 0 or 1. To extract the true answers from the unreliable responses of workers, we propose an iterative algorithm for multiple-choice questions.

Our algorithm generates two types of messages between task nodes and worker nodes. The first type is the task message $\vec{x}_{i\to j}$, which is denoted as a D_i dimensional vector. Each component of this vector corresponds to the likelihood meaning the possibility being a true answer. The second type is a worker message $y_{j\to i}$ which specifies the reliable worker j. Since these worker messages are strongly correlated with the reliability p_j, our algorithm can assess relative reliability. Hence, we will empirically verify the correlation between $\{y_{j\to i}\}$ and $\{p_j\}$ in section 6. The initial messages of our iterative algorithm are sampled independently from the *Gaussian* distribution with unit mean and variance, i.e., $y_{j\to i}^{(0)} \sim \mathcal{N}(1, 1)$. Unlike EM-based algorithms [2, 21], our approach is not sensitive to initial conditions as long as the consensus of the group of workers is positively biased. Now, we define the adjacent set of task i as ∂i and similarly the adjacent set of worker j is defined as ∂j. Then, at the k^{th} it-

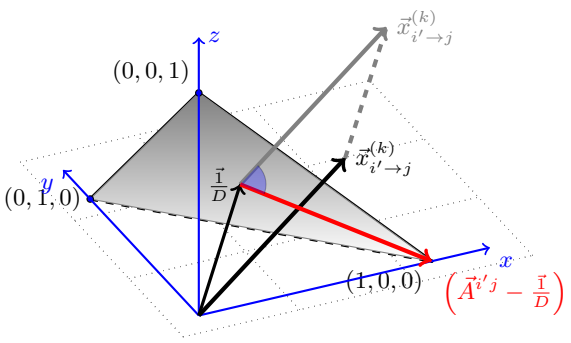


Figure 1: Description of a task message $\vec{x}^{(k)}_{i'\to j}$ and a response vector $\vec{A}^{i'j}$, in the message vector space when $\vec{A}^{i'j}=(1,0,0)$ and $D_{i'}=3$.

eration, both messages are updated using the following rules:

$$\vec{x}^{(k)}_{i\to j}=\sum_{j'\in\partial i\setminus j}\vec{A}^{ij'}y^{(k-1)}_{j'\to i},\qquad \forall(i,j)\in E \quad (1)$$

$$y^{(k)}_{j\to i}=\sum_{i'\in\partial j\setminus i}\left(\vec{A}^{i'j}-\frac{\vec{1}}{D}\right)\cdot\vec{x}^{(k-1)}_{i'\to j},\qquad \forall(i,j)\in E \quad (2)$$

At the task message update process shown in (1), our algorithm gives weight to the answer according to the reliability of a worker. At the worker message update process shown in (2), it gives greater reliability to a worker who strongly follows consensus of other workers.

Figure 1 describes two vectors in the message vector space. As shown above, $(\vec{A}^{i'j}-\frac{\vec{1}}{D})$ represents the difference between response of worker j for task i' and the random answer $\frac{\vec{1}}{D}$. Also, $\vec{x}^{(k-1)}_{i'\to j}$ means the weighted sum of responses of other workers who have solved the task i'. Thus, the inner product of these two vectors in (2) can assess the similarity between the response of worker j for the task i' and sum of those of other workers who have solved the task i'. A larger positive similarity value of the two vectors means that worker j is more reliable. Meanwhile, the negative value specifies that the worker j does not follow the consensus of other workers and our algorithm regards the worker j as unreliable. Specially, when $\vec{x}^{(k-1)}_{i'\to j}$ and $(\vec{A}^{i'j}-\frac{\vec{1}}{D})$ are orthogonal for fixed task i', the inner product of two vector is close to zero. This means that $\vec{x}^{(k-1)}_{i'\to j}$ does not contribute to the message of the worker j. Then, $y^{(k)}_{j\to i}$ is defined as the sum of the inner product from each task message except for that of task i, representing the relative reliability of the worker j. Returning to (1), $\vec{x}^{(k)}_{i\to j}$ is determined by the weighted voting of workers who have solved task i, except for the message from the worker j. The worker j' contributes to the response $\vec{A}^{ij'}$ as much as the weight value $y^{(k-1)}_{j'\to i}$. Thus, $\vec{x}^{(k)}_{i\to j}$ is defined as the sum of $\vec{A}^{ij'}y^{(k-1)}_{j'\to i}$ which represents the estimated true answer for the task i. The following describes the pseudo code of our algorithm.

The maximum number of iterations k_{max} is analyzed in section 5.2. In practice, a dozen of iterations is sufficient for the convergence of our algorithm. After k_{max} iterations, our algorithm makes the final estimate vector $\vec{x}_i$ of a task i,

Algorithm 1 Multiple Iterative Algorithm

Input: E, $\{\vec{A}^{ij}\}_{(i,j)\in E}$, k_{max}
Output: Estimation $^{\forall}i\in[m]$, $\hat{t}_i\in\{\vec{e}_{u_i}|u_i\in[1:D]\}$
For $^{\forall}(i,j)\in E$ **do**
  Initialize $y^{(0)}_{j\to i}$ with random $Z_{ij}\sim N(1,1)$;
For $k=1,2\ldots,k_{max}$ **do**
  For $^{\forall}(i,j)\in E$ do $\vec{x}^{(k)}_{i\to j}\leftarrow\sum_{j'\in\partial i\setminus j}\vec{A}^{ij'}y^{(k-1)}_{j'\to i}$;
  For $^{\forall}(i,j)\in E$ do $y^{(k)}_{j\to i}\leftarrow\sum_{i'\in\partial j\setminus i}(\vec{A}^{i'j}-\frac{\vec{1}}{D})\cdot\vec{x}^{(k-1)}_{i'\to j}$;
For $^{\forall}j\in[n]$ **do** $y_j\leftarrow\sum_{i\in\partial j}(\vec{A}^{ij}-\frac{\vec{1}}{D})\cdot\vec{x}^{(k_{max}-1)}_{i\to j}$;
For $^{\forall}i\in[m]$ **do** $\vec{x}_i\leftarrow\sum_{j\in\partial i}\vec{A}^{ij}y^{(k_{max}-1)}_{j\to i}$;
Estimate vector $\hat{t}_i=\vec{e}_{u_i}$ where $u_i=\arg\max_d(\vec{x}_i)$

and each component of the vector represents the possibility of being the true answer. Our algorithm infers the true answer by choosing u_i that has the maximum component among final likelihoods of $\vec{x}_i$. Then, our algorithm outputs the estimate of the true answer denoted as a unit vector, $\vec{e}_{u_i}$.

3.3 Task Allocation for General Setting

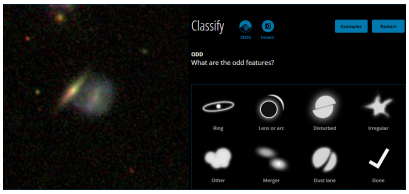
In the previous section, we proposed our iterative algorithm for a bipartite graph according to the *pairing model*. However, the number of workers allocated to each task can differ in cases that are more general. That must bring about the variation of the number of tasks that each worker solves. Hence, we consider a general bipartite graph with various node degrees. To apply our algorithm in this scenario, the update rules of both messages should be slightly changed in terms of the task node degree l_i and the worker node degree r_j. For a task message $\vec{x}^{(k)}_{i\to j}$, we divide each message value by the task node degree (l_i-1) so that tasks with different degrees receive the similar effect from worker nodes. In other words, dividing by (l_i-1) equalizes the task message values. Likewise, a worker message $y^{(k)}_{j\to i}$ is divided by the worker node degree (r_j-1) for general setting.

In addition to the generalization of the degree profile, we consider the various number of choices for each task (For example $^{\forall}i\in[m]$, $D_i\in\{2,3,4\}$). In practice, the number of choice for each task can differ from one another and our Algorithm 2 can cope with this variation. The following describes the pseudo code of our generalized algorithm.

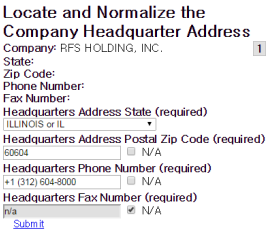
Algorithm 2 Generalized Multiple Iterative Algorithm

Input: E, $\{\vec{A}^{ij}\}_{(i,j)\in E}$, k_{max}
Output: Estimation $^{\forall}i\in[m]$, $\hat{t}_i\in\{\vec{e}_{u_i}|u_i\in[1:D_i]\}$
For $^{\forall}(i,j)\in E$ **do**
  Initialize $y^{(0)}_{j\to i}$ with random $Z_{ij}\sim N(1,1)$;
For $k=1,2\ldots,k_{max}$ **do**
  For $^{\forall}(i,j)\in E$ do $\vec{x}^{(k)}_{i\to j}\leftarrow\sum_{j'\in\partial i\setminus j}\left(\frac{1}{l_i-1}\right)\vec{A}^{ij'}y^{(k-1)}_{j'\to i}$;
  For $^{\forall}(i,j)\in E$ do
    $y^{(k)}_{j\to i}\leftarrow\sum_{i'\in\partial j\setminus i}\left(\frac{1}{r_j-1}\right)(\vec{A}^{i'j}-\frac{\vec{1}}{D_{i'}})\cdot\vec{x}^{(k-1)}_{i'\to j}$;
For $^{\forall}j\in[n]$ **do**
    $y_j\leftarrow\sum_{i\in\partial j}\left(\frac{1}{r_j-1}\right)(\vec{A}^{ij}-\frac{\vec{1}}{D_i})\cdot\vec{x}^{(k_{max}-1)}_{i\to j}$;
For $^{\forall}i\in[m]$ **do** $\vec{x}_i\leftarrow\sum_{j\in\partial i}\left(\frac{1}{l_i-1}\right)\vec{A}^{ij}y^{(k_{max}-1)}_{j\to i}$;
Estimate vector $\hat{t}_i=\vec{e}_{u_i}$ where $u_i=\arg\max_d(\vec{x}_i)$

(a) An independent multiple-choice question: Determining the breed of a dog.



(b) GalaxyZoo project: classifying galaxies according to their shapes.



(c) A real task in Amazon Mechanical Turk: Filling up address information of a given company.

(d) reCAPTCHA: Typing words for spam protection and a book digitization project.

Figure 2: Examples of multiple-choice questions.

Adaptive task allocation method. One of significant points of our algorithm is that worker's relative reliability can be assessed in the course of its iterations. If we use this property, the performance of inferring the true answer can be improved further. Consider the adaptive strategy as an improvement method using the above property. First, a small portion of the tasks is used to infer the reliability of each worker using the iterative algorithm. Then, we select partial workers who have higher worker values to message and let them solve all of the remaining tasks. Although this method gives a larger burden to workers who are more reliable, the total number of edges is maintained. In section 6, the adaptive task allocation method will be explained in detail and we will verify some of the gains of this method through several experiments.

4. APPLICATIONS

We described an algorithmic solution to crowdsourcing systems for multiple-choice questions in the previous section, and we now look into some applications that our algorithm can treat. As we can see in crowdsourcing systems like Amazon Mechanical Turk, tasks are distributed in the form of multiple-choice questions and short-answer questions like entering zip-code. Although previous algorithms like [9, 21] have shown remarkable results in binary cases, a merit of our algorithm is that outstanding results can even be achieved on multiple-choice and short-answer questions that real tasks usually contain. Furthermore, a remarkable characteristic of our model is that the number of choices can vary for each question. This flexibility makes our model more applicable for real crowdsourced data. In this section, we describe some applications in detail that can apply our algorithm.

Labeling or tagging images is a common usage of exploiting crowdsourcing systems, and shows successful results in practice [3]. One of such example is classifying species or breeds of dogs in the images illustrated in Figure 2(a). Such tasks are very tough for machines, and even humans who have no background knowledge of dogs. These tasks are suitable for crowdsourcing materials and have multiple choices that are directly applicable to our algorithm.

Another application of labeling tasks is Galaxy Zoo, one of the well known projects using the wisdom of crowds (cf. Figure 2(b)). Galaxy Zoo has distributed over 300,000 images of galaxies to crowds for classification by their shape. Any volunteer with no prior knowledge can visit the website, where they are presented with an image of a galaxy and instructions of labeling manner. Then they answer a series of questions about the visual form of the galaxy, like whether it has arms or a bulge. Each step consists of multiple-choice questions, and the number of choices varies for each question. Since our algorithm is flexible for the number of choices, the responses of Galaxy Zoo can be easily aggregated using our algorithm.

For short-answer questions, it is hard to aggregate workers' responses in general, because their responses can vary. Our algorithm can settle this problem with the idea of transforming short-answer questions into several multiple-choice questions. When the length of the response to a short-answer question is fixed, short-answer questions can be split into several smaller tasks by considering each character of a response. In other words, each character is treated as one *microtask* in short-answer questions. For example, consider the task of entering a fixed-length answer such as a zip code like 97232. It can be treated as five microtasks, and each of the characters has 10 possible answers, from 0 to 9. Note that in each microtask, we only consider the number of choices as much as the number of candidate answers. For example, if candidate answers for a microtask are "4", "7", and "9", then we set the number of choices to three for this microtask. In addition, we can decide a set of candidate answers as all gathered responses simply, or only responses of top-K likelihood effectively.

Next, we consider when the length of the response varies. We can make another small task that determines the true length of the response and then we can discard the answers whose length is determined as a minor option. In summary, every short-answer question can be decomposed to several microtasks by considering each character of the answer and its length. Characters of the response and its length are transformed into small microtasks, and each microtask is considered a multiple-choice question. Thus, by applying our algorithm, responses to these short-answer questions can be easily aggregated. For a real task in Amazon Mechanical Turk, as illustrated in Figure 2(c), entering zip codes or phone numbers is an example of short-answer questions.

Another popular crowdsourcing application for short-answer questions is reCAPTCHA [19] illustrated in Figure 2(d). In its original version, CAPTCHA was first introduced to distinguish automatic bots by typing some characters correctly in a given image. It was extended to reCAPTCHA which digitalizes some hard phrases that Optical Character Recog-

nition (OCR) techniques cannot recognize. In this case, the length of responses can vary, so a small task determining the length of response is necessary, as we mentioned. Although discarding the rest of the responses can be viewed as a waste, it is a tolerable loss, since the length of the responses is generally consistent. In addition, we need discuss the number of tasks r, each worker is given. In reCAPTCHA, we can only assign one entering task to each worker, while our algorithm needs sufficient number of tasks for each worker to ensure reliable inference. However, since we split each worker's response into several microtasks, the task size problem is naturally solved.

Another special application of our algorithm is as an adaptive task allocation strategy, since it explicitly computes the relative reliability of the workers, even with no prior knowledge of the worker distribution. If we design a proper adaptive strategy for crowdsourcing systems, we can boost its performance from the perspective of quality control of workers. The best workers can be recruited and exploited to resolve more questions. It can be viewed as a method for finding experts from crowds or filtering out workers who just spam for rewards; therefore, we can exploit reliable workers efficiently under the same budget through an adaptive task allocation strategy. We will examine such an adaptive strategy in the experiment section.

5. ANALYSIS OF ALGORITHMS

In this section, we provide proof for the performance guarantee of Algorithm 1. In *Theorem 1*, we show that the error bound depends on task degree l and the quality of the workers. More precisely, we show that an upper bound on the probability of error decays exponentially. From this section, we assume that $D_i = D$ for all $i \in [n]$.

5.1 Quality of workers

Let $\vec{v}_j$ denote the confusion vector of each worker j. Each component of the vector means the probability that a worker chooses the corresponding choice for a response. For a fixed task i with true answer $\hat{t}_{u_i} \in U$, the confusion vector $\vec{v}_j$ of worker j is defined as follows:

$$v_{jd} = \begin{cases} p_j & \text{if } \hat{t}_{u_i} = \vec{e}_d \\ \frac{1-p_j}{D-1} & \text{otherwise} \end{cases}$$

From an information theoretical perspective, the quality of workers can be defined as negative entropy with an offset and using the above confusion vector, we can define the quality of workers as

$$q = \mathbb{E}\Big[H(p) - \bar{p}\log(\hat{D}) + \log(D)\Big], \tag{3}$$

where $H(p) = p\log p + \bar{p}\log\bar{p}, \quad \bar{p} = 1-p, \quad \hat{D} = D-1.$

According to the quality of each worker, we can divide the workers into three types. At the extreme, workers with a quality close to zero make arbitrary responses. Since, we cannot obtain any information from them, let us define them as "Non-informative workers." At the other extreme, workers with the a quality close to one make almost true answers and we call them "Reliable workers." Lastly, there are workers who make wrong answers on purpose and affect the crowdsourcing system badly; they can be regarded as "Malicious workers." In our algorithm, since the worker message value y_j is related to the quality, workers with negative y_j, positive y_j and y_j close to zero correspond to "Reliable workers," "Malicious workers," and "Non-informative workers," respectively.

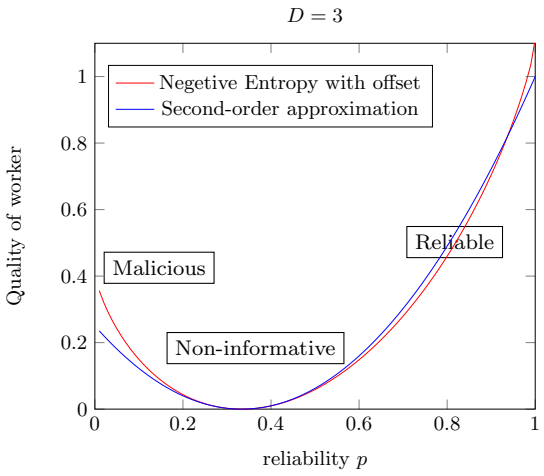


Figure 3: Comparison of the quality between negative entropy with offset and second-order polynomial approximation.

Although the quality of workers theoretically follows negative entropy, we found that a second-order polynomial approximation is sufficient for our analysis as described in Figure 3. As the dimension of the tasks increase, the approximation deviates from the real quality. Nevertheless, second-order approximation fits well to the real quality in the acceptable dimension case that our algorithm targets.

$$q \simeq \tilde{q}_1 = \mathbb{E}\left[\left(\frac{D}{D-1}\right)^2\left(p_j - \frac{1}{D}\right)^2\right] \tag{4}$$

For simplicity, we will use this approximated quality in the following sections. There is one more necessary assumption about worker distribution that workers give the correct answers on average rather than random or adversarial answers, so that $E[p_j] > \frac{1}{D}$. Given only workers' responses, any inference algorithms analogize the true answers from the general or popular choices of crowds. Consider an extreme case in which everyone gives adversarial answers in a binary classification task; no algorithm can correctly infer the reliability of the crowd. Hence, the assumption $E[p_j] > \frac{1}{D}$ is a natural necessary.

5.2 Bound on the Average Error Probability

From now on, let $\hat{l} \equiv l-1$, $\hat{r} \equiv r-1$, and the average quality of workers is defined as $q = \mathbb{E}[(\frac{D}{D-1})^2(p_j - \frac{1}{D})^2]$. Also, σ_k^2 denotes the effective variance in the *sub-Gaussian* tail of the task message distribution after k iterations.

$$\sigma_k^2 \equiv \frac{2q}{\mu^2 T^{k-1}} + \left(\frac{D}{D-1}\right)^2\left(3 + \frac{1}{8q\hat{r}}\right)\left[\frac{1-1/T^{k-1}}{1-1/T}\right], \tag{5}$$

where $T = \dfrac{(D-1)^2}{(D^2-D-1)}q^2\hat{l}\hat{r}.$

THEOREM 1. *For fixed $l > 1$ and $r > 1$, assume that m tasks are assigned to n workers according to a random (l, r)-regular bipartite graph according to the pairing model. If the distribution of the reliability satisfies $\mu \equiv \mathbb{E}[\frac{D}{D-1}(p_j - \frac{1}{D})] > 0$ and $T > 1$, then for any $t \in \{e_i\}^m$, the estimate after k iterations of the iterative algorithm achieves*

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leqslant (D-1)e^{-lq/(2\sigma_k^2)} + \frac{3lr}{m}(\hat{lr})^{2k-2}. \quad (6)$$

The second term of the equation is the upper bound of probability that the graph dose not have a local tree-like structure and it can be quite small as long as we treat a large number of tasks. Therefore, the dominant factor of the upper bound is the first exponential term. As shown in (5), $T = 1$ is the crucial condition and we can satisfy $T > 1$ by using a sufficiently larger l or r. Then, with $T > 1$, σ_k^2 converges to a finite limit σ_∞^2, and we have

$$\sigma_\infty^2 = \left(3 + \frac{1}{8q\hat{r}}\right)\left(\frac{T}{T-1}\right). \quad (7)$$

Thus, the bounds of the first term of (6) does not depend on the number of tasks m or the number of iterations k. The following corollary describes an upper bound that only depends on l, q, σ_∞^2, and D.

COROLLARY 1. *Under the hypotheses of Theorem 1, there exists $m_0 = 3lre^{lq/4\sigma_\infty^2}(\hat{lr})^{2(k-1)}$ and $k_0 = 1+(\log(q/\mu^2)/\log T)$ such that*

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leqslant De^{-lq/(4\sigma_\infty^2)}, \quad (8)$$

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

PROOF. First, we will show that $\sigma_k^2 \leqslant 2\sigma_\infty^2$ for $k \geqslant 1 + (\log(q/\mu^2)/\log T)$. Since $T > 1$, as per our assumption, $\sigma_k^2 = (2q/\mu^2 T^{k-1}) + (\frac{D}{D-1})^2(3 + 1/8q\hat{r})\frac{1-1/T^{k-1}}{T-1} \leqslant 2 + \sigma_\infty^2 \leqslant \sigma_\infty^2 + \sigma_\infty^2 \leqslant 2\sigma_\infty^2$. Therefore, the first term of (6) is bounded like $(D-1)e^{-lq/2\sigma_k^2} \leqslant (D-1)e^{-lq/4\sigma_\infty^2}$. Next, it is sufficient to set $m \geqslant 3lre^{lq/4\sigma_\infty^2}(\hat{lr})^{2(k-1)}$ to ensure $\frac{3lr}{m}(\hat{lr})^{2k-2} \leqslant e^{-lq/(4\sigma_\infty^2)}$. □

From *corollary 1*, we obtained that the required number of iterations k_0, is small in that it is the only logarithmic in l,r,q,μ and D. On the other hand, although the required number of entire tasks m_0, is very large in *corollary 1*, the experimental result in section 6 shows that the performance of error exhibits exponential decay as stated in (8).

Now, if we assume that there are no limitation on worker degree r and $T \geqslant 2$, we can find $\sigma_\infty^2 \leqslant 2(3 + 1/8q\hat{r})$. Then, for all $r \geqslant 1 + 1/8q$, as similar with the [11], we get the following bound:

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leqslant De^{-lq/32}. \quad (9)$$

Also, we can check the following corollary in terms of the number of queries per task l to achieve a target accuracy. Hence, we get the following corollary.

COROLLARY 2. *Using the task assignment scheme according to pairing model with $r \geqslant 1 + 1/8q$ and the iterative algorithm, it is sufficient to query $(32/q)log(D/\epsilon)$ times per task to guarantee that the error bound is at most ϵ for any $\epsilon \leqslant 1/2$ and for all $m \geqslant m_0$.*

5.3 Proof of the Theorem 1

The proof is roughly composed of three parts. First, the second term at the right-hand side of (6) is proved using its local tree-like property. Second, the remaining term of the right-hand side of (6) is verified using *Chernoff bound* in the assumption that the estimates of the task message follow *sub-Gaussian* distribution. Lastly, we prove that the assumption of the second part is true within certain parameters. To apply *density evolution* with multi-dimensional vector form is difficult in that the cross term of each components are generated. Therefore our proof can be differentiated from binary setting in [11].

Without a loss of generality, it is possible to assume that the true answer of each task, for any $i \in [m]$, $t_i = \vec{e}_1$. Let $\hat{t}_i^{(k)}$ denote the estimated answer of task i defined in section 5.2. If we draw a task $\mathbf{I}$, uniformly at random from the task set, the average probability of error can be denoted as

$$\frac{1}{m}\sum_{i\in[m]}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) = \mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}), \quad (10)$$

Let $\mathrm{G}_{\mathbf{I},k}$ denote a subgraph of G that consists of all the nodes whose distance from the node '$\mathbf{I}$' is at most k. After k iterations, the local graph with root '$\mathbf{I}$' is $\mathrm{G}_{\mathbf{I},2k-1}$, since the update process operates twice for each iteration. To take advantage of *density evolution*, the full independence of each branch is needed. Thus, we bound the probability of error with two terms, one that represents the probability that subgraph $\mathrm{G}_{\mathbf{I},2k-1}$ is not a tree and the other, which represents the probability that $\mathrm{G}_{\mathbf{I},2k-1}$ is a tree with a wrong answer.

$$\begin{aligned}\mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}) &\leqslant \mathbb{P}(\mathrm{G}_{\mathbf{I},2k-1} \text{ is not a tree }) \\ &+ \mathbb{P}(\mathrm{G}_{\mathbf{I},2k-1} \text{ is a tree and } t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}). \quad (11)\end{aligned}$$

The following lemma bounds the first term and proves that the probability that a local subgraph is not a tree vanishes as m grows. A proof of Lemma 1 is provided [11] (cf. *Karger, Oh and Shah* 2011, section 3.2).

LEMMA 1. *From a random (l,r)-regular bipartite graph generated according to the pairing model,*

$$\mathbb{P}(\mathrm{G}_{\mathbf{I},2k-1} \textit{ is not a tree }) \leqslant \left(\hat{lr}\right)^{(2k-2)}\frac{3lr}{m}.$$

From the result of Lemma 1, we can concentrate directly on the second term of (11) and define the pairwise difference of task messages as $\tilde{\mathbf{x}}_d^{(k)} = \mathbf{x}_1^{(k)} - \mathbf{x}_d^{(k)}$ for $^\forall d \in [2:D]$.

$$\begin{aligned}\mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)} | \mathrm{G}_{\mathbf{I},k} \text{ is a tree}) &\leqslant \mathbb{P}(\cup_{d=2}^{D}\{\tilde{x}_{\mathbf{I}}^{(k)} \leqslant 0\} | \mathrm{G}_{\mathbf{I},k} \text{ is a tree}) \\ &\leqslant \mathbb{P}(\cup_{d=2}^{D}\{\tilde{x}_{\mathbf{I}} \leqslant 0\}).\end{aligned}$$

To obtain a tight upper bound on $\mathbb{P}(\cup_{d=2}^{D}\{\hat{\mathbf{x}}_d^{(k)} \leqslant 0\})$ of our iterative algorithm, we assume that $\tilde{\mathbf{x}}_d^{(k)}$ follow *sub-Gaussian* distribution for any $d \in [2:D]$ and prove these in section 5.4. Then, *Chernoff bound* is applied to the independent message branches and this brings us the tight bound of our algorithm. A random variable $\mathbf{z}$ with mean m is said to be *sub-Gaussian* with parameter $\tilde{\sigma}$ if for any $\lambda \in \mathbb{R}$ the following inequality holds:

$$\mathbb{E}[e^{\lambda \mathbf{z}}] \leqslant e^{m\lambda + (1/2)\tilde{\sigma}^2\lambda^2}. \quad (12)$$

We will first show that for $^{\forall}d \in [2:D]$, $\tilde{\mathbf{x}}_d^{(k)}$ is *sub-Gaussian* with mean m_k and parameter $\tilde{\sigma}_k^2$ for specific region of λ, precisely for $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$. Now we define

$$m_k \equiv \mu \hat{l} U^{k-1}, \quad {}^{\forall}k \in \mathbb{N}$$

$$\tilde{\sigma}_k^2 \equiv 2\hat{l}S^{k-1} + [\mu^2\hat{l}^2\hat{r}(3q^2\hat{l}\hat{r} + \hat{l}/8)]U^{2k-4}\left[\frac{1-(1/T)^{k-1}}{1-(1/T)}\right],$$

$$\text{where } U = \frac{D-1}{D}q\hat{l}\hat{r}, \quad S = \frac{D^2-D-1}{D^2}\hat{l}\hat{r}$$

$$T = \frac{U^2}{S} = \frac{(D-1)^2}{D^2-D-1}q^2\hat{l}\hat{r}$$

then

$$\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}] \leqslant e^{m_k\lambda + (1/2)\tilde{\sigma}_k^2\lambda^2}. \tag{13}$$

The locally tree-like property of a sparse random graph provides the distributional independence among incoming messages, that is $\mathbb{E}[e^{\lambda\hat{\mathbf{x}}_d^{(k)}}] = \mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}]^{(l/\hat{l})}$. Thus, $\hat{\mathbf{x}}_d^{(k)}$ satisfies $\mathbb{E}[e^{\lambda\hat{\mathbf{x}}_d^{(k)}}] \leqslant e^{(l/\hat{l})m_k\lambda + ((l/2\hat{l}))\tilde{\sigma}_k^2\lambda^2}$ for all $d \in [2:D]$. Because of full independence of each branch, we can apply *Chernoff bound* with $\lambda = -m_k/(\tilde{\sigma_k^2})$, and then we obtain

$$\mathbb{P}(\hat{\mathbf{x}}_d^{(k)} \leqslant 0) \leqslant \mathbb{E}[e^{\lambda\hat{\mathbf{x}}_d^{(k)}}] \leqslant e^{-lm_k^2/(2\hat{l}\tilde{\sigma}_k^2)}. \tag{14}$$

$$\begin{aligned}\mathbb{P}(\cup_{d=2}^{D}\{\hat{\mathbf{x}}_d^{(k)} \leqslant 0\}) &\leqslant \sum_{d=2}^{D}\mathbb{P}(\hat{\mathbf{x}}_d^{(k)} \leqslant 0) \\ &\leqslant (D-1)e^{-lm_k^2/(2\hat{l}\tilde{\sigma}_k^2)}. \end{aligned}\tag{15}$$

Since $m_k m_{k-1}/(\tilde{\sigma}_k^2) \leqslant 1/(3\hat{r})$, we can easily check $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$. This finalizes the *Proof of the Theorem 1.*

5.4 Proof of Sub-Gaussianity

Now we prove that for all $d \in [2:D]$, $\tilde{\mathbf{x}}_d^{(k)}$ is *sub-Gaussian* with some m_k and $\tilde{\sigma}_k^2$. Recurrence relation of the evolution of the MGFs(moment generating functions) on $\tilde{\mathbf{x}}_d$ and $\mathbf{y_p}$ are stated as

$$\begin{aligned}\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}] &= \left(\mathbb{E}_{\mathbf{p}}\left[\mathbf{p}\mathbb{E}\left[e^{\lambda\mathbf{y}_p^{(k-1)}}|\mathbf{p}\right] + \frac{\bar{\mathbf{p}}}{D-1}\mathbb{E}\left[e^{-\lambda\mathbf{y}_p^{(k-1)}}|\mathbf{p}\right]\right.\right. \\ &\quad \left.\left. + \frac{\bar{\mathbf{p}}}{D-1}(D-2)\right]\right)^{\hat{l}}, \end{aligned}\tag{16}$$

$$\begin{aligned}\mathbb{E}[e^{\lambda\mathbf{y}_p^{(k)}}] &= \left(p\mathbb{E}\left[e^{\lambda(\frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}_d^{(k)})}\right]\right. \\ &\quad \left. + \frac{\bar{p}}{D-1}\sum_{j=2}^{D}\mathbb{E}\left[e^{\lambda(-\tilde{\mathbf{x}}_j^{(k)} + \frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}_d^{(k)})}\right]\right)^{\hat{r}}, \end{aligned}\tag{17}$$

$$\text{where} \quad \bar{p} = 1-p \text{ and } \bar{\mathbf{p}} = 1-\mathbf{p}.$$

Using above MGFs and *mathematical induction*, we can prove that $\tilde{\mathbf{x}}_d^{(k)}$ are *sub-Gaussian*, for all $d \in [2:D]$.

First, for $k=1$, we prove that all of $\tilde{\mathbf{x}}_d^{(1)}$ are *sub-Gaussian* random variables with mean $m_1 = \mu\tilde{l}$ and variance $\tilde{\sigma}_1^2 = 2\tilde{l}$, where $\mu \equiv \mathbb{E}[\frac{D}{D-1}(p_j - \frac{1}{D})]$. Using *Gaussian* initialization of $\mathbf{y}_p \sim \mathcal{N}(1,1)$, we obtain $\mathbb{E}[e^{\lambda\mathbf{y}_p^{(0)}}] = e^{\lambda + (1/2)\lambda^2}$ regardless of p. Substituting this into equation (13), we have

$$\begin{aligned}\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(1)}}] &= \left(\mathbb{E}_{\mathbf{p}}\left[\mathbf{p}e^{\lambda+(1/2)\lambda^2} + \left(\frac{1-\mathbf{p}}{D-1}\right)e^{-\lambda+(1/2)\lambda^2}\right.\right. \\ &\quad \left.\left. + \left(\frac{1-\mathbf{p}}{D-1}\right)(D-2)\right]\right)^{\hat{l}} \\ &\leqslant \left(\mathbb{E}[a]e^{\lambda} + \left(\mathbb{E}[\bar{a}]e^{-\lambda}\right)^{\hat{l}}e^{(1/2)\hat{l}\lambda^2}\right. \\ &\leqslant e^{(\mu\lambda+\lambda^2)\hat{l}}, \end{aligned}\tag{18}$$

$$\text{where} \quad a = \frac{Dp+D-2}{2(D-1)}, \quad \bar{a} = 1-a = \frac{D(1-p)}{2(D-1)}$$

where the first inequality follows from the fact that $2 \leqslant e^{\lambda} + e^{-\lambda}$ for any $\lambda \in \mathbb{R}$, and the second inequality follows from that

$$be^{z} + (1-b)e^{-z} \leqslant e^{(2b-1)z + (1/2)z^2}, \tag{19}$$

for any $z \in \mathbb{R}$ and $b \in [0,1]$ (cf. *Alon and Spencer* 2008, Lemma A.1.5) [1].

From k^{th} inductive hypothesis, we have $\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}] \leqslant e^{m_k\lambda + (1/2)\tilde{\sigma}_k^2\lambda^2}$ for $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$. Now, we will show $\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k+1)}}] \leqslant e^{m_{k+1}\lambda + (1/2)\tilde{\sigma}_{k+1}^2\lambda^2}$ for $|\lambda| \leqslant 1/(2m_k\hat{r})$. In advance, substituting (19) into (17), we have

LEMMA 2. *For any $|\lambda| \leqslant 1/(2m_k\hat{r})$ and $p \in [0,1]$, we get*

$$\mathbb{E}[e^{\lambda\mathbf{y}_p^{(k)}}] \leqslant \left[\left(pe^{(1/2)m_k\lambda} + \bar{p}e^{-(1/2)m_k\lambda}\right)\right]^{\hat{r}} \cdot e^{(\frac{D-2}{2D})\hat{r}m_k\lambda + (\frac{D^2-D-1}{D^2})\hat{r}\tilde{\sigma}_k^2\lambda^2}.$$

Similar to (18)'s process, from (16), we get

$$\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k+1)}}] \leqslant \mathbb{E}_{\mathbf{p}}\left(a\mathbb{E}\left[e^{\lambda\mathbf{y}_p^{(k)}}\right] + \bar{a}\mathbb{E}\left[e^{-\lambda\mathbf{y}_p^{(k)}}\right]\right)^{\hat{l}}.$$

with $2 \leqslant e^{\lambda} + e^{-\lambda}$ for any $\lambda \in \mathbb{R}$.
Substituting the result of Lemma 2 into the above inequality provides

$$\begin{aligned}\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k+1)}}] &\leqslant \mathbb{E}_{\mathbf{p}}\left[a\left(\mathbf{p}e^{(1/2)m_k\lambda} + \bar{\mathbf{p}}e^{-(1/2)m_k\lambda}\right)^{\hat{r}}\right. \\ &\quad \left. + \bar{a}\left(\mathbf{p}e^{(1/2)m_k\lambda} + \bar{\mathbf{p}}e^{-(1/2)m_k\lambda}\right)^{\hat{r}}\right]^{\hat{l}} \\ &\quad \cdot e^{(\frac{D-2}{2D})\hat{l}\hat{r}m_k\lambda + (\frac{D^2-D-1}{D^2})\hat{l}\hat{r}\tilde{\sigma}_k^2\lambda^2}. \end{aligned}\tag{20}$$

Now we are left to bound (20) using following Lemma 3.

LEMMA 3. *For any $|z| \leqslant 1/(2\hat{r})$ and $p \in [0,1]$, we get*

$$\mathbb{E}_{\mathbf{p}}\left[a\left(\mathbf{p}e^{\frac{D-1}{D}z} + \bar{\mathbf{p}}e^{-\frac{1}{D}z}\right) + \bar{a}\left(\mathbf{p}e^{-\frac{D-1}{D}z} + \bar{\mathbf{p}}e^{\frac{1}{D}z}\right)\right]^{\hat{r}} \leqslant e^{\frac{D-1}{D}q\hat{r}z + \left(\frac{3}{2}q\hat{r} + \frac{1}{8}\right)\hat{r}z^2}.$$

Applying this to (20) gives

$$\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k+1)}}] \leqslant e^{\frac{D-1}{D}q\hat{l}\hat{r}m_k\lambda + \left[\left(\frac{3}{2}q\hat{r} + \frac{1}{8}\right)m_k^2 + \left(\frac{D^2-D-1}{D^2}\right)\tilde{\sigma}_k^2\right]\hat{l}\hat{r}},$$

for $|\lambda| \leqslant 1/(2m_k\hat{r})$.

From the result of *mathematical induction*, we can obtain the recurrence relations of two parameters of the *sub-Gaussians*

$$m_{k+1} = \Big[\frac{D-1}{D} q\hat{l}\hat{r}\Big] m_k,$$

$$\tilde{\sigma}^2_{k+1} = \Big[\Big(\frac{3}{2}q\hat{r} + \frac{1}{8}\Big)m_k^2 + \Big(\frac{D^2-D-1}{D^2}\Big)\tilde{\sigma}^2_k\Big]\hat{l}\hat{r},$$

with $\frac{D-1}{D} q\hat{l}\hat{r} \geqslant 1$, where m_k is increasing geometric series. Thus, the above recursions hold for $|\lambda| \leqslant 1/(2m_k\hat{r})$ and we get

$$m_k = \mu\hat{l}\Big[\frac{D-1}{D} q\hat{l}\hat{r}\Big]^{k-1},$$

for all $k \in \mathbb{N}$. Substituting m_k into $\tilde{\sigma}^2_k$, we obtain

$$\tilde{\sigma}^2_k = a\tilde{\sigma}^2_{k-1} + bc^{k-2}, \tag{21}$$

where

$$a = \frac{D^2-D-1}{D^2}\hat{l}\hat{r}, \quad b = \mu^2\hat{l}^3\hat{r}\Big(\frac{3}{2}q\hat{r}+\frac{1}{8}\Big) \quad c = \Big[\frac{D-1}{D}q\hat{l}\hat{r}\Big]^2$$

For $T \neq 1$, This type of recurrence relation can be represented as the following closed formula.

$$\tilde{\sigma}^2_k = \tilde{\sigma}^2_1 a^{k-1} + bc^{k-2}\Big[\frac{1-(a/c)^{k-1}}{1-(a/c)}\Big]. \tag{22}$$

This finishes the proof of (13).

Proof of Lemma 2. In the $k+1^{th}$ inductive step of *mathematical induction*, we assume that $\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}^{(k)}_d}] \leqslant e^{m_k\lambda+(1/2)\tilde{\sigma}^2_k\lambda^2}$ for any $d \in [2:D]$ with $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$. In other words, all of $\tilde{\mathbf{x}}^{(k)}_d$ follow *sub-Gaussian* distribution with parameters m_k and $\tilde{\sigma}^2_k$. From (17), each component at the right-hand side can be represented as the product of several combinations of $[e^{\lambda\tilde{\mathbf{x}}^{(k)}_d}]$ and the product of variables means a linear combination in the exponential field. Using *hölder's inequality*, we prove that the linear transformation of *sub-Gaussian* random variables follows also *sub-Gaussian* distribution with some parameters. Moreover, these parameters are determined by D, mean m_k and variance $\tilde{\sigma}^2_k$ of each *sub-Gaussian* $\tilde{\mathbf{x}}^{(k)}_d$. Applying *h"older's inequality* to (17), the first term at the right-hand side of (17) gives

$$\mathbb{E}\Big[e^{\lambda(\frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}^{(k)}_d)}\Big] \leqslant \prod_{d=2}^{D}\Big[\mathbb{E}\Big(e^{\lambda(1/D)\tilde{\mathbf{x}}^{(k)}_d)}\Big)^{D-1}\Big]^{\frac{1}{D-1}}$$

$$\leqslant e^{(\frac{D-1}{D})m_k\lambda+(\frac{D-1}{2D^2})\tilde{\sigma}^2_k\lambda^2}.$$

For the second term at the right-hand side of (17), we have

$$\mathbb{E}\Big[e^{\lambda(-\tilde{\mathbf{x}}^{(k)}_j+\frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}^{(k)}_d)}\Big] \leqslant \mathbb{E}\Big[e^{-\lambda(\frac{D-1}{D})\tilde{\mathbf{x}}^{(k)}_j}\Big]$$

$$\cdot \prod_{d=2,d\neq j}^{D}\Big[\mathbb{E}\Big(e^{\lambda(1/D)\tilde{\mathbf{x}}^{(k)}_d)}\Big)^{D-1}\Big]^{\frac{1}{D-1}}$$

$$\leqslant e^{(-\frac{1}{D})m_k\lambda+(\frac{D^2-D-1}{2D^2})\tilde{\sigma}^2_k\lambda^2}.$$

Getting these two results together finishes the proof of Lemma 2.

Proof of Lemma 3. From (19), we get

$$\Big(\mathbf{p}e^{\frac{D-1}{D}z} + \bar{\mathbf{p}}e^{-\frac{1}{D}z}\Big) \leqslant e^{(\mathbf{p}-\frac{1}{D})z+\frac{1}{8}z^2}.$$

Applying this result to the original formula, we have

$$\mathbb{E}_{\mathbf{p}}\Big[a\Big(\mathbf{p}e^{\frac{D-1}{D}z} + \bar{\mathbf{p}}e^{-\frac{1}{D}z}\Big) + \bar{a}\Big(\mathbf{p}e^{-\frac{D-1}{D}z} + \bar{\mathbf{p}}e^{\frac{1}{D}z}\Big)\Big]^{\hat{r}}$$

$$\leqslant \mathbb{E}\Big[e^{\frac{D}{D-1}(\mathbf{p}-\frac{1}{D})\hat{r}z+\frac{1}{2}(\mathbf{p}-\frac{1}{D})^2\hat{r}^2z^2}\Big]\cdot e^{\frac{1}{8}\hat{r}z^2}.$$

In this point, we bring the fact that $e^a \leqslant 1+a+0.63a^2$ for $|a| \leqslant 5/8$

$$\mathbb{E}\Big[e^{\frac{D}{D-1}(\mathbf{p}-\frac{1}{D})\hat{r}z+\frac{1}{2}(\mathbf{p}-\frac{1}{D})^2\hat{r}^2z^2}\Big]$$

$$\leqslant \mathbb{E}\Big[1+\Big(\frac{D-1}{D}\Big)q\hat{r}z + \frac{1}{2}\Big(\frac{D-1}{D}\Big)^2 q\hat{r}^2z^2$$

$$+0.63\Big\{\Big(\frac{D-1}{D}\Big)q\hat{r}z + \frac{1}{2}\Big(\frac{D-1}{D}\Big)^2 q\hat{r}^2z^2\Big\}^2\Big]$$

$$\leqslant 1+\Big(\frac{D-1}{D}\Big)q\hat{r}z + \frac{3}{2}\Big(\frac{D-1}{D}\Big)^2 q\hat{r}^2z^2$$

$$\leqslant e^{\left(\frac{D-1}{D}\right)q\hat{r}z+\frac{3}{2}q\hat{r}^2z^2},$$

for $|z| \leqslant 1/(2\hat{r})$ and $D \leqslant 2$.

Phase Transition. As shown in (22), the performance of our algorithm is only bounded when the condition $T > 1$ is satisfied. Meanwhile, with $T < 1$, $\tilde{\sigma}^2_k$ which means the variance of the $\tilde{\mathbf{x}}^{(k)}_d$ diverges as the number of iteration k increases. In this case, our performance guarantee is no longer valid and the performance becomes worse compared to other algorithms such as EM and majority voting. Note that except for extreme case such as when using very low quality workers and the deficient assignments, $T > 1$ is easily satisfied and our performance guarantee is valid. In section 6, we will verify the existence of this critical point at $T = 1$ through several experiments with different conditions.

$$\frac{a}{c} = \frac{(D-1)^2}{(D^2-D-1)}q^2\hat{l}\hat{r} = T.$$

6. EXPERIMENTS

In this section, we verify the performance of the *multiple iterative algorithm* discussed in the previous sections with different sets of simulations. First, we check that the error of the iterative algorithm exhibits exponential decay as l increases or q increases. In addition, we show that our algorithm achieves a better performance than that of the majority voting and EM approach above a phase transition of $T = 1$. Next simulation investigates the linear relationship between y_j value and the ratio of *the number of correct answers* to r_j for each worker. Then, we do experiments on the adaptive scenario by varying the proportion of pilot tasks and selected reliable workers. Finally, we do simulations on the experiments introduced above with a task set consisting of various D values.

Comparison with other algorithms. To show the competitiveness of our algorithm, we ran our *multiple iterative algorithm*, majority voting, and the EM approach for 2000 tasks and 2000 workers with fixed $D = 2$, 3, 4, and 5 (Figure 4 and Figure 5). The performance of the oracle estimator

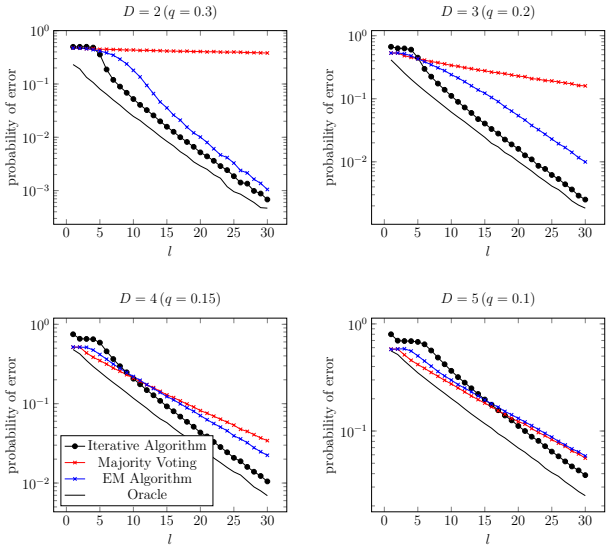


Figure 4: Comparisons of probabilities of error between different algorithms varying l values ($m = n = 2000$, $l = r$).

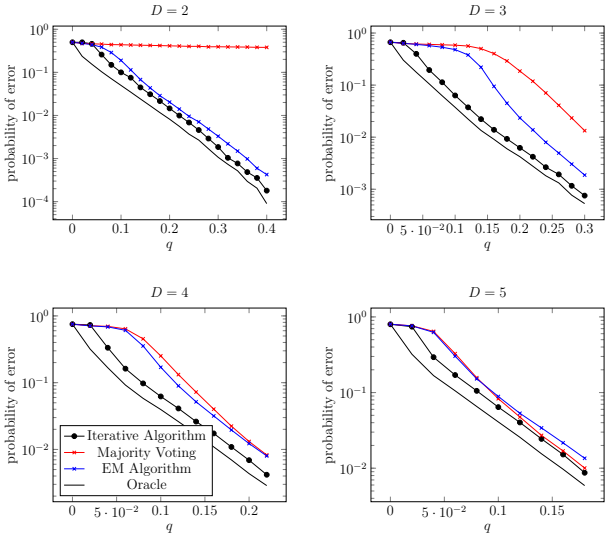


Figure 5: Comparisons of probabilities of error between different algorithms varying q values ($m = n = 2000$, $l = r = 25$).

is also presented as a lower bound and the EM algorithm is implemented with Dawid and Skene's method [2]. In Figure 4, we can check that the probability of error decays exponentially as l increases, and is lower than that of the majority voting and EM approach above the phase transition $T = 1$. In addition, in Figure 5, we find the probabilities of error decays as q increases.

We expect a phase transition at $T = \frac{(D-1)^2}{(D^2-D-1)} q^2 \hat{l}\hat{r} = 1$ or $l = 1 + \sqrt{\frac{(D^2-D-1)}{(D-1)^2}} \frac{1}{q}$ when $l = r$ according to our theorem. With this, we can expect transitions to happen around $l = 4.33$ for $D = 2(q = 0.3)$, $l = 6.59$ for $D = 3(q = 0.2)$, $l = 8.37$ for $D = 4(q = 0.15)$, and $l = 11.89$ for $D = 5(q = 0.1)$. From the experiments in Figure 4, we see that iterative algorithm starts to perform better than majority voting around $l = 5, 6, 10, 18$ for each D. Note that these values are very similar with the theoretical values. It follows from the fact the error of our method increases with k when $T < 1$ as stated in Section 5. As can clearly be seen from the simulation results, we can check that the l values required for achieving $T > 1$ are not large. For example, if we consider dealing with short-answer questions like reCAPTCHA which is introduced in Section 4, carrying off the required $r(= l)$ is accomplished easily since each alphabet is considered as a separate question.

Adaptive Scenario. The inference of workers' relative reliability in the course of iterations is one of the algorithm's most important aspects. Now, we define $\hat{p_j}$ for each worker j as following:

$$\hat{p_j} = \frac{the\ number\ of\ correct\ answers}{r_j}.$$

After k_{max} iterations, we can find reliable workers by the value of worker message y_j since this value is proportional to $\hat{p_j}$, which is influenced by p_j. Relative reliability y_j is calculated by the following equation in Algorithm 1.

$$y_j \leftarrow \sum_{i \in \partial j} \left(\vec{A}^{ij} - \frac{\vec{1}}{D} \right) \cdot \vec{x}_{i \to j}^{(k_{max}-1)}$$

Figure 6 shows that there are strong correlations between y_j and $\hat{p_j}$. In one simulation, the correlation coefficients[1] between y_j and $\hat{p_j}$ are measured as 0.993, 0.989, 0.968, 0.938 for each $D = 2$, 3, 4, and 5, which are significantly large values. We can also check that the line passes approximately the point of $(\frac{1}{D}, 0)$, which represents a non-informative worker's reliability, as expected in Section 5.

One of the utilizations of this correlation property is the *adaptive scenario*, which extracts more reliable workers from the crowds after the inference of pilot tasks, and lets them solve the remaining tasks. We can improve the performance of our algorithm further with the scenario. The strategy consists of two steps in detail. In the first step, we use $m' = \alpha m$ pilot tasks to infer the relative reliability of workers using the iterative algorithm.

$$m' = \alpha m, n' = n$$
$$l' = l, r' = \alpha r$$

In the second step, we select βn workers who have higher $|y_j|$ values after the first step, and each worker solves $\frac{m-m'}{\beta m} r$ tasks out of the remaining $m - m'$ tasks. We sort them out with higher $|y_j|$ values since we can gain less information from workers who have lower $|y_j|$ values, which means that their reliability is closer to $1/D$ than those of the others (Figure 6 and Figure 3).

$$m'' = m - m', n'' = \beta n$$
$$l'' = l, r'' = \frac{m - m'}{\beta m} r$$

[1]Pearson product-moment correlation coefficient(PPMCC) is used for evaluation.

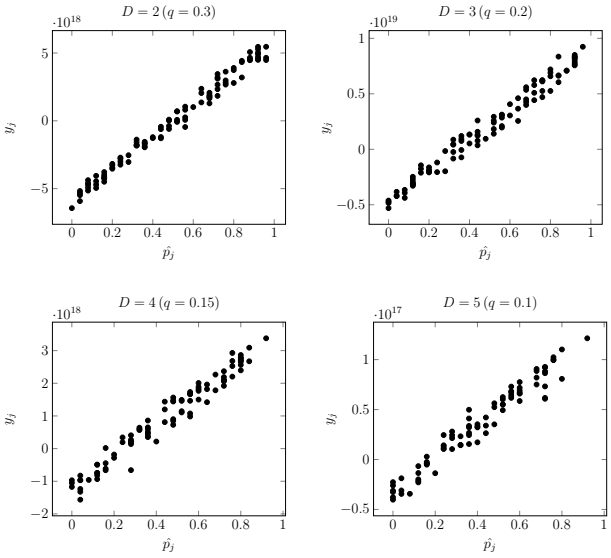


Figure 6: Relationship between y_j and $\hat{p_j}$ ($m = n = 2000$, $k = 10$).

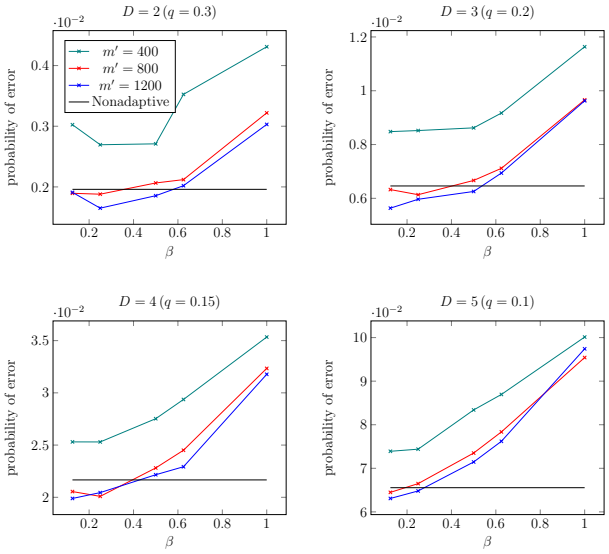


Figure 7: Adaptive Scenario ($m = n = 2000$, $l = 25$).

To show the performance improvements when using the adaptive scenario, we perform experiments with several (m', β) sets. Figure 7 shows that the probability of error is smaller than for the non-adaptive scenario when proper m' and β are used. Specifically, as β decreases, the error tends to decrease since fewer, but more reliable, workers then solve the rest of the questions. However, we have to consider each worker's inherent capacity[2] when choosing an appropriate β. With limited capacity, we cannot use an unreasonably low β, since it places too high a burden on each worker. In addition, we have to take enough m' pilot tasks to guarantee the accuracy of the relative reliability, which are inferred in the first step.

Simulations on a set of various D values. To show the performance of the *generalized multiple iterative algorithm*, we do simulations on a task set consisting of various D values with Algorithm 2. In detail, we repeat the same experiments with a question set composed in $1:1:1$ ratios of tasks which D are 2, 3, 4 respectively. Then, we have to investigate for the general case that q is calculated with the following equation.

$$q = \mathbb{E}[q_j] = \mathbb{E}\left[\left(\frac{D_i}{D_i - 1}\right)^2 \left(p_{ij} - \frac{1}{D_i}\right)^2\right]$$

We define q_j as an individual quality of the worker j. To perform simulations and to analyze the results, we have to make an assumption that a worker with individual quality q_j solves question with a reliability p_{ij} for each D_i. We can check that the same tendencies found in previous simulations also appear in Figure 8. There is also the strong correlation between y_j and $\hat{p_j}$ as 0.960. This result is notable in that in the real world, there are many more cases where questions have varying number of choices than fixed ones.

[2]The number of possible questions that each worker can manage to solve in one transaction.

7. CONCLUSION

We have proposed an iterative algorithm that can handle multiple-choice and short-answer questions which are general types of questions in real crowdsourcing systems. Especially for short-answer questions, we provide a method of transforming original tasks into several microtasks. Therefore, we give a general solution for real crowdsourcing systems to infer the correct answers from unreliable responses. From the performance analysis of our algorithm, we have proved that an upper bound on the probability of error decays exponentially and we have verified that our algorithm outperforms majority voting and EM-based algorithm through numerical experiments.

For the future works, our work can be combined to the context of online learning, or we can think of another model for multiple-choice questions that have multiple correct answers, not just one correct answer.

Acknowledgments

This research was in parts funded by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2012R1A1A1014965), and in parts funded by the Brain Korea 21 Plus Project in 2015. This work was also supported by Automation and System Research Institute (ASRI), SNU, and Institute of New Media and Communications (INMC), SNU.

8. REFERENCES

[1] N. Alon and J. H. Spencer. *The probabilistic method*. John Wiley & Sons, Hoboken, NJ, 2008.

[2] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern*

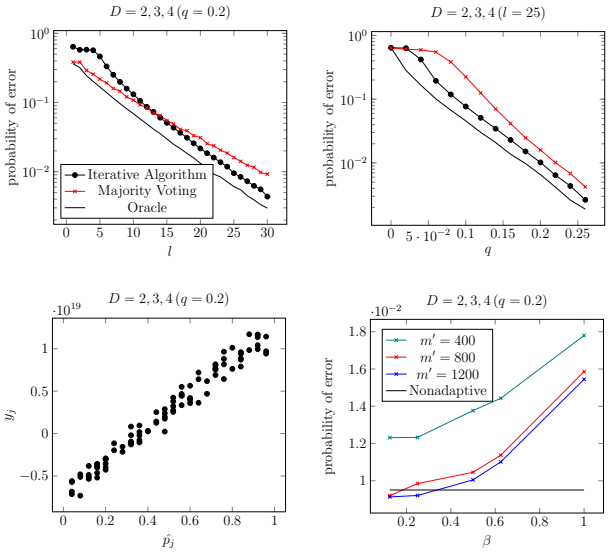


Figure 8: Simulations on a set of various D values ($m = n = 2000\,(D = 2:\ 666\,/\,D = 3:\ 667\,/\,D = 4:\ 668)$).

Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009.

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

[5] S. Ertekin, H. Hirsh, and C. Rudin. Approximating the wisdom of the crowd. In *Proceedings of the Workshop on Computational Social Science and the Wisdom of Crowds*, 2011.

[6] C.-J. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 534–542, 2013.

[7] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

[8] D. R. Karger, S. Oh, and D. Shah. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 284–291. IEEE, 2011.

[9] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961, 2011.

[10] D. R. Karger, S. Oh, and D. Shah. Efficient crowdsourcing for multi-class labeling. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 81–92. ACM, 2013.

[11] D. R. Karger, S. Oh, and D. Shah. Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems. *Operations Research*, 62(1):1–24, Feb. 2014.

[12] G. Kazai, J. Kamps, and N. Milic-Frayling. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval*, 16(2):138–178, 2013.

[13] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.

[14] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, et al. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.

[15] Q. Liu, J. Peng, and A. Ihler. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 692–700, 2012.

[16] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322, 2010.

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

[18] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics, 2008.

[19] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[20] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[21] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

[22] Y. Zheng, S. Scott, and K. Deng. Active learning from multiple noisy labelers with varied costs. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 639–648. IEEE, 2010.