Ph.D. DISSERTATION

Fast and Reliable Inference Algorithms in Crowdsourcing Systems

크라우드소싱 시스템에서의 빠르고 신뢰성 높은 추론 알고리즘

BY

DONGHYEON LEE

FEBRUARY 2021

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Fast and Reliable Inference Algorithms in Crowdsourcing Systems

크라우드소싱 시스템에서의 빠르고 신뢰성 높은 추론 알고리즘

BY

DONGHYEON LEE

FEBRUARY 2021

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Fast and Reliable Inference Algorithms in Crowdsourcing Systems

크라우드소싱 시스템에서의 빠르고 신뢰성 높은 추론 알고리즘

지도교수 정 교 민

이 논문을 공학박사 학위논문으로 제출함

2021년 2월

서울대학교 대학원

전기 컴퓨터 공학부

이 동 현

이동현의 공학박사 학위 논문을 인준함

2021년 2월

Abstract

As the need for large scale labeled data grows in various fields, 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 natural solution, various research studies have sought to aggregate noisy responses to obtain greater reliability in results. In this dissertation, we propose novel iterative massage-passing style algorithms to infer the groundtruths from noisy answers, which can be directly applied to real crowdsourcing systems. While EM-based algorithms get the limelight in crowdsourcing systems due to their useful inference techniques, our proposed algorithms draw faster and more reliable answers through an iterative scheme based on the idea of low-rank matrix approximations. We show that the performance of our proposed iterative algorithms are order-optimal, which outperforms majority voting and EM-based algorithms. Unlike other researches solving simple binary-choice questions (yes & no), our studies cover more complex task types which contain multiple-choice questions, short-answer questions, K-approval voting, and real-valued vector regression.

keywords: Crowdsourcing, Message-passing style algorithm, Approximate inference

student number: 2013-23125

Contents

List of Tables

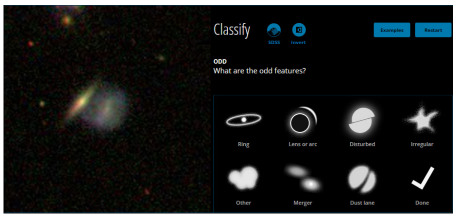
List of Figures

Chapter 1

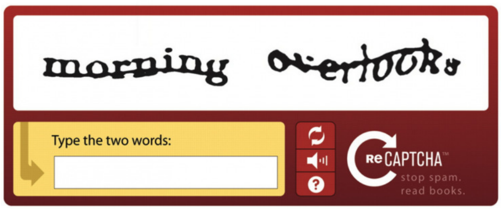
Introduction

Crowdsourcing has become one of the cornerstones of research in the development of human computation based intelligence systems [3, 4, 5] since some of modern machine learning-based approaches are impractical in reality due to several problems. One problem is that they need a lot of labeled data in the training phase. As the big data era begins, we are able to access enormous amount of data, but most of them are not labeled. Through the conventional approach in which a taskmaster distributes tasks to few expert labelers, the labeling task becomes a bottleneck since it is generally too expensive and tardy. Another problem is that there are still a lot of tasks machines cannot do well in compared with humans despite of rapid development of machine learning techniques. To overcome those problems, new web-based services like Amazon Mechanical Turk [6] have arisen and become popular as a consequence [3]. Since crowdsourcing systems are well-combined with the characteristics of Internet such as accessibility, spontaneity, and vastness, those web tools are able to provide ideal solutions for gathering enormous responses from widespread crowds in a short time with a relatively low budget. Some of real crowdsourcing projects successfully labeled by web-based crowdsourcing services are shown in Figure 1.1.

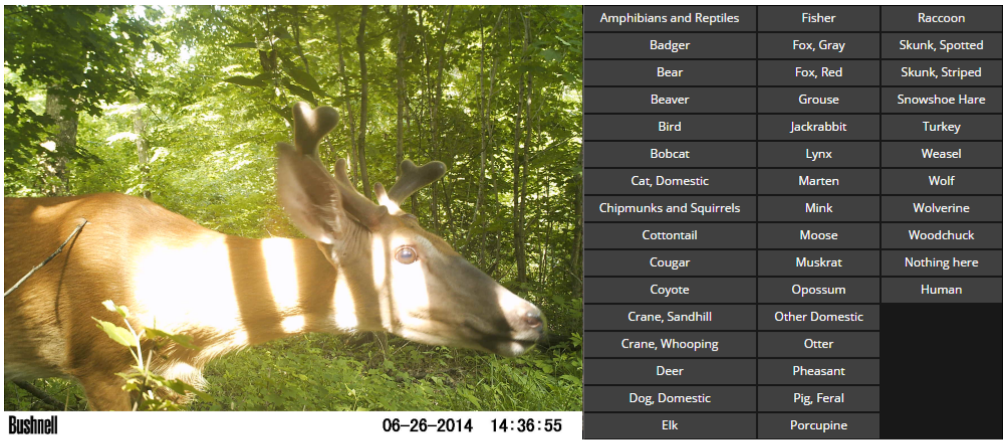
One of the most active field to gather labeled data by web-based crowdsourcing is the computer vision. Developing computer vision algorithms that are able to au-



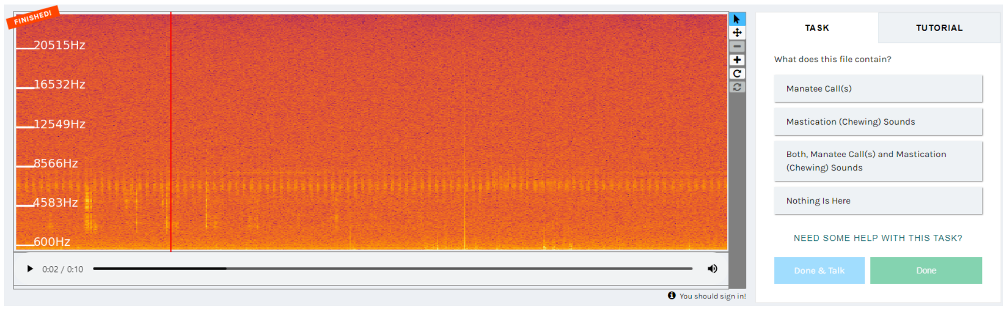
(a) GalaxyZoo [1]: Classifying galaxies according to their shapes.



(b) reCAPTCHA [2]: Typing words for spam protection and a book digitization project.



(c) Wisconsin Wildlife: Classify animals pictured at Wisconsin.


FINISHED!
20515Hz
16532Hz
12549Hz
8566Hz
4583Hz
600Hz
0:02 / 0:10
TASK
TUTORIAL
What does this file contain?
Manatee Call(s)
Mastication (Chewing) Sounds
Both, Manatee Call(s) and Mastication (Chewing) Sounds
Nothing Is Here
NEED SOME HELP WITH THIS TASK?
Done & Talk
Done
You should sign in!

(d) Manatee Chat: Identifying and classifying of manatee calls.

Figure 1.1: Real crowdsourcing projects successfully labeled by web-based crowdsourcing services. (a) GalaxyZoo [1], (b) reCAPTCHA [2], (c) Wisconsin Wildlife, (d) ManateeChat

tomatically distinguish objects or backgrounds requires manually annotating a large collection of images. To properly annotate challenging visual concepts, web-based crowdsourcing platforms offer an inexpensive method to capture human knowledge and understanding. As a consequence, most well-known dataset for computer vision is successfully annotated by web-based crowdsourcing platforms. For examples, well-known crowdsourced datasets are listed as follows: ImageNet [3] for object classification, PASCAL VOC [7] and ILSVRC [8] for object detection, LabelMe [9], and MS-COCO [10] for pixel-level image segmentation, MPII [11], LSP [12], and FLIC [13] for human pose estimations, TUHOI [14], UT Interactee [15], HICO [16] for actions and interactions in images, VATIC [17], YouTube-Objects [18], SegTrack [19], Event-Net [20], Sports-1M [21], THUMOS [22], MultiTHUMos [23], ActivityNet [24], and Hollywood in Homes [25] for detailed video annotation, and INTERACT [26] for abstraction and cartoons.

Despite the innovative framework of crowdsourcing systems, responses from workers can be unreliable [27, 28, 29, 4], 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 [30, 31, 32]. Recently, Karger et al. [33, 34] 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.

Our algorithm iteratively computes relative reliability of each worker in a novel

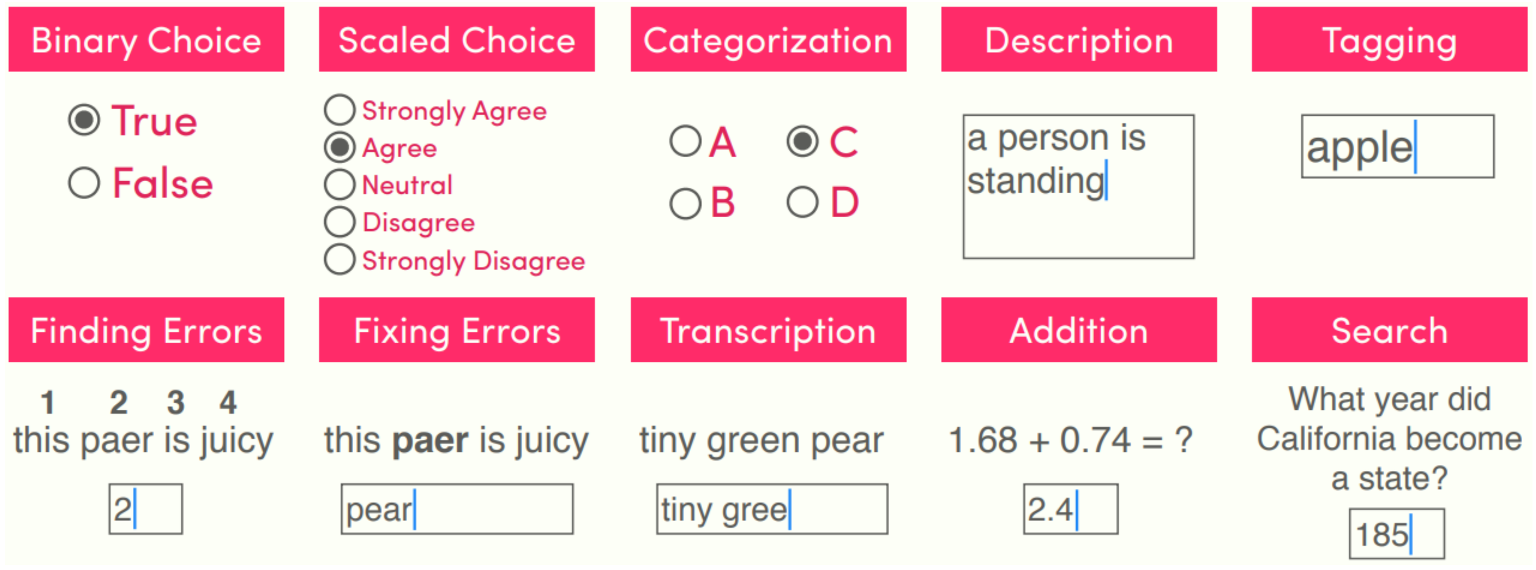


Figure 1.2: Various types of tasks in real crowdsourcing systems.

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. 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.

The motivation of this research is to develop crowdsourcing applications which can be operated in real crowdsourcing systems. In fact, most of major research studies in this field have concentrated on cases with binary answers, yes (+1) or no (-1) [34, 32]. However, in reality, crowdsourcers have been requested various types of tasks in real crowdsourcing as shown in Figure 1.2. Real crowdsourced data posted on Amazon Mechanical Turk usually consists of multiple-choice questions, short-answer questions, K-approval voting, real-valued vector regression, and short-answer questions, so more general inference techniques should be employed.

In this dissertation, we focus on a more general structure for crowdsourcing systems that can be applied to multiple-choice questions, short-answer questions, K-approval voting, and real-valued vector regression. These tasks are fundamental components composing crowdsourcing tasks, but have been not of interest to academic studies in general. Of course, even more complex tasks have been requested in real crowdsourcing systems, but many of complex tasks can be considered as a combination of those fundamental tasks which are stated above. For example, let us consider an Optical Character Reader (OCR)-like task that asks to find a car number plate and extract the car number from an image, shown in Figure 1.3 and 1.4. The corresponding response would be the positions of the car number plate represented as bounding

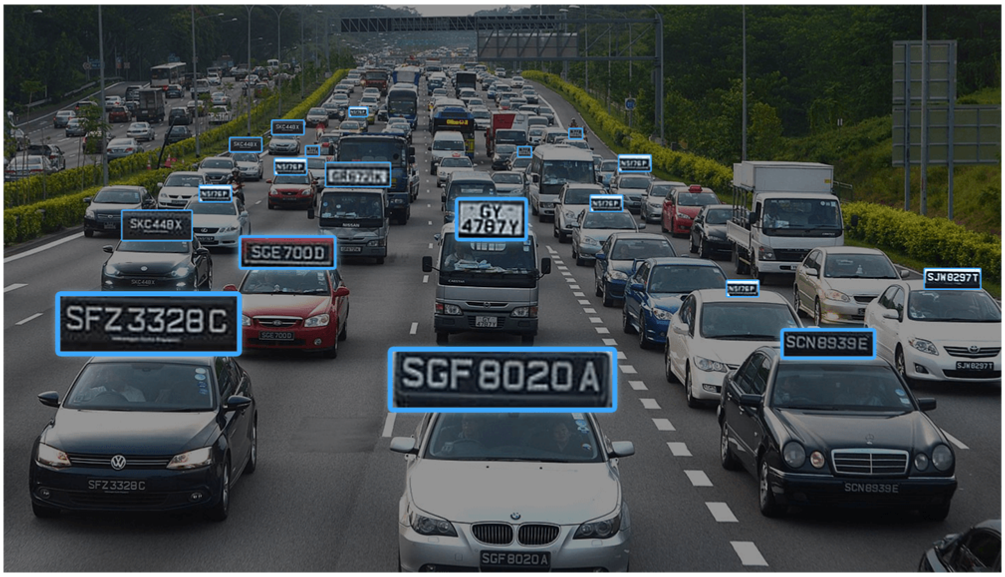


Figure 1.3: Car number plates captured by a traffic camera. These data can be easily labeled by web-based crowdsourcing services.

boxes, and the car number written in some numbers and characters. Then the response can be split into real-valued regression tasks (the position of boxes) and multiple-choice questions (the number of car number plates, the issuing state of the car number plates, the length of the car numbers and each character of the car numbers). We claim that this dissertation should be evaluated as not only research studies but also practical applications since we propose several corresponding inference algorithms for those fundamental tasks that can be directly applied to crowdsourcing systems.

This dissertation is based on our several previous papers for crowdsourcing systems:

- Reliable Multiple-choice Iterative Algorithm for Crowdsourcing Systems. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pages 205–216. ACM, 2015.

- Iterative Learning for K-approval Votes in Crowdsourcing Systems. MDPI Applied Sciences, 2021.

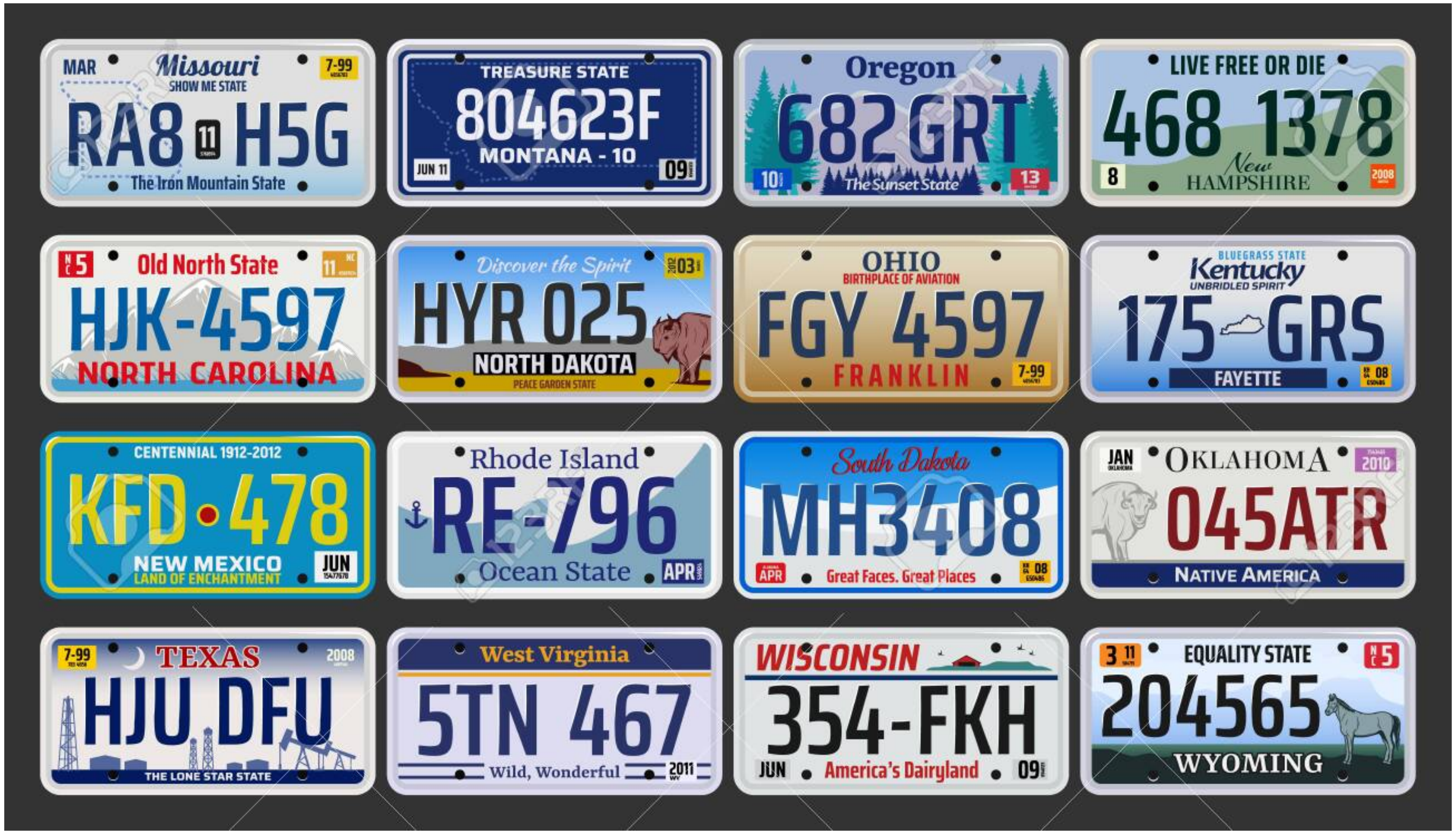


Figure 1.4: Various car number plates in USA. Labeling a car number plate can be split into several real-valued regression tasks for bounding boxes, and several multiple-choice questions for the number of car number plates, the issuing state of the car number plates, the length of the car numbers and each character of the car numbers.

- Reliable Aggregation Method for Vector Regression Tasks in Crowdsourcing. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 261–273. Springer, 2020.

This dissertation is organized as follows: In *Chapter 2*, we define an inference problem for crowdsourcing systems and introduce several methodologies that can deal with binary-choice questions. We study crowdsourcing systems requesting discrete answers for multiple-choice questions and short-answer questions in *Chapter 3*, and for multiple-choice questions with K-approval voting in *Chapter 4*. We further study crowdsourcing systems requesting continuous answers for real-valued vector regressions in *Chapter 5*. Each chapter proposes own algorithm for the corresponding crowdsourcing task, and provide rigorous performance guarantees for each proposed algorithm. Then we present comparative results through numerical experiments. Lastly, we draw conclusions in *Chapter 6*.

Chapter 2

Background

In this chapter, we introduce crowdsourcing systems with mathematical representations where all tasks are comprised of binary-choice questions. This kind of tasks is well studied in various research studies that provide mathematical problem setups and technical solutions to design inference algorithms. Note that mathematical notations introduced in this chapter follow the previous works [34, 35, 36], and can be appropriately changed in other chapters that solves other task types.

2.1 Crowdsourcing Systems for Binary-choice Questions

Assume there are M workers and N tasks with binary labels $\{\pm 1\}$. Denote by $z_i \in \{\pm 1\}$, $i \in [N]$ the true label of task i, where $[N]$ represents the set of first N integers; $\mathcal{N}_j$ is the set of tasks labeled by worker j, and $\mathcal{M}_i$ the workers labeling task i. The task assignment scheme can be represented by a bipartite graph where an edge (i, j) denotes that the task i is labeled by the worker j. The labeling results form a matrix $L \in \{0, \pm 1\}^{N \times M}$, where $L_{ij} \in \{\pm 1\}$ denotes the answer if worker j labels task i, and $L_{ij} = 0$ if otherwise. The goal is to find an optimal estimator $\hat{z}$ of the true labels z given the observation L, minimizing the average bit-wise error rate $\frac{1}{N} \sum_{i \in [N]} \text{prob}[\hat{z}_i \neq z_i]$.

We assume that all the tasks have the same level of difficulty, but that workers may have different predictive abilities. Following Karger et al. [34], we initially assume that the ability of worker j is measured by a single parameter q_j, which corresponds to their probability of correctness: $q_j = \text{prob}[L_{ij} = z_i]$. More generally, the workers' abilities can be measured by a confusion matrix, to which our method can be easily extended.

The values of q_j reflect the abilities of the workers: $q_j \approx 1$ correspond to *experts* that provide reliable answers; $q_j \approx 1/2$ denote *spammers* that give random labels independent of the questions; and $q_j < 1/2$ denote *adversaries* that tend to provide opposite answers. Conceptually, the spammers and adversaries should be treated differently, the spammers provide no useful information and only degrade the results, while the adversaries actually carry useful information, and can be exploited to improve the results if the algorithm can identify them and flip their labels. We assume the q_j of all workers are drawn independently from a common prior $p(q_j|\theta)$, where θ are the hyper-parameters. To avoid the cases when adversaries and/or spammers overwhelm the system, it is reasonable to require that $\mathbb{E}[q_j|\theta] > 1/2$. Typical priors include the Beta prior $p(q_j|\theta) \propto q_j^{\alpha-1}(1-q_j)^{\beta-1}$ and discrete priors, e.g., the *spammer-hammer* model, where $q_j \approx 0.5$ or $q_j \approx 1$ with equal probability.

2.1.1 Majority Voting

The majority voting (MV) method aggregates the workers' labels by

$$\hat{z}_i^{\text{majority}} = \text{sign}[\sum_{j \in \mathcal{M}_i} L_{ij}]. \tag{2.1}$$

The limitation of MV is that it weights all the workers equally, and performs poorly when the qualities of the workers are diverse, especially when adversarial workers exist.

2.1.2 Expectation Maximization

Weighting the workers properly requires estimating their abilities q_j, usually via a maximum *a posteriori* estimator, $\hat{q} = \mathrm{argmax} \log p(q|L,\theta) = \log \sum_z p(q,z|L,\theta)$. This is commonly solved using an EM algorithm treating the z as hidden variables. Assuming a Beta(α,β) prior on q_j, EM is formulated as

$$\text{E-step} : \mu_i(z_i) \propto \prod_{j \in \mathcal{M}_i} \hat{q}_j^{\delta_{ij}} (1-\hat{q}_j)^{1-\delta_{ij}}, \tag{2.2}$$

$$\text{M-step} : \hat{q}_j = \frac{\sum_{i \in \mathcal{N}_j} \mu_i(L_{ij}) + \alpha - 1}{|\mathcal{N}_j| + \alpha + \beta - 2}, \tag{2.3}$$

where $\delta_{ij} = \mathbb{I}[L_{ij} = z_i]$; the $\hat{z}_i$ is then estimated via $\hat{z}_i = \mathrm{argmax}_{z_i} \mu_i(z_i)$. Many approaches have been proposed to improve this simple EM approach, mainly by building more complicated models.

2.1.3 Message Passing

A rather different algorithm in a message-passing style is proposed by Karger, Oh and Shah [34] (referred to as KOS in the sequel). Let $x_{i \to j}$ and $y_{j \to i}$ be *real-valued* messages from tasks to workers and from workers to tasks, respectively. Initializing $y^0_{j \to i}$ randomly from Normal$(1, 1)$ or deterministically by $y^0_{j \to i} = 1$, KOS updates the messages at t-th iteration via

$$x^{t+1}_{i \to j} = \sum_{j' \in \mathcal{M}_{ij}} L_{ij'} y^t_{j' \to i}, \tag{2.4}$$

$$y^{t+1}_{j \to i} = \sum_{i' \in \mathcal{N}_{ji}} L_{i'j} x^{t+1}_{i' \to j}, \tag{2.5}$$

and the labels are estimated via $\hat{s}^t_i = \mathrm{sign}[\hat{x}^t_i]$, where $\hat{x}^t_i = \sum_{j \in \mathcal{M}_i} L_{ij} y^t_{j \to i}$.

Chapter 3

Crowdsourcing Systems for Multiple-choice Questions

In this study, we focus on 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 as considered in [37] 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.

This chapter is organized as follows: We discuss related work in *Section 3.1*. In *Section 3.2*, we make a setup, and we describe our algorithm to infer the true answers for multiple-choice questions in *Section 3.3*. Then, we look into some applications in *Section 3.4* and provide performance guarantees for our algorithm in *Section 3.5*. In *Section 3.6*, we present comparative results through numerical experiments, and we draw conclusions in *Section 3.7*.

3.1 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 [29].

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 [38] 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 [32] 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 [31]. Another variant proposed by Raykar et al. [30] 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. [33] 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 [34] 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. [36] shows that choosing a suitable prior can improve the performance via a Bayesian approach.

Recently, Karger et al. [37] 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. [39, 40, 41, 42] showed that the performance can be significantly improved through exploration/exploitation approaches.

3.2 Problem 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 $\dfrac{1 - p_j}{D_i - 1}$. 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.

3.3 Inference 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.

3.3.1 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 [34]). 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 3.5.3*.

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 3.6*. The initial messages of our iterative algorithm are sampled independently from the *Gaussian* distribution with unit mean and variance, i.e., $y^{(0)}_{j \to i} \sim \mathcal{N}(1, 1)$. Unlike EM-based algorithms [38, 32], 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} iteration, both messages are updated using the following rules:

$$\vec{x}^{(k)}_{i \to j} = \sum_{j' \in \partial i \backslash j} \vec{A}^{ij'} y^{(k-1)}_{j' \to i}, \qquad \forall (i, j) \in E \tag{3.1}$$

$$y^{(k)}_{j \to i} = \sum_{i' \in \partial j \backslash i} \left(\vec{A}^{i'j} - \frac{\vec{1}}{D} \right) \cdot \vec{x}^{(k-1)}_{i' \to j}, \qquad \forall (i, j) \in E \tag{3.2}$$

At the task message update process shown in (3.1), our algorithm gives weight to the answer according to the reliability of a worker. At the worker message update

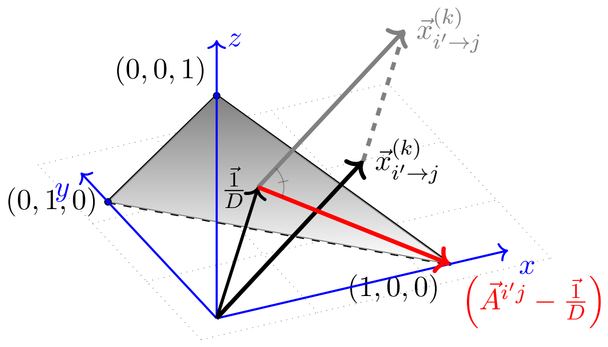

$\vec{x}^{(k)}_{i'\to j}$
z
$(0,0,1)$
$\vec{x}^{(k)}_{i'\to j}$
$\frac{\vec{1}}{D}$
$(0,1,0)$
y
x
$(1,0,0)$
$\left(\vec{A}^{i'j}-\frac{\vec{1}}{D}\right)$

Figure 3.1: 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 $D_{i'}=3$.

process shown in (3.2), it gives greater reliability to a worker who strongly follows consensus of other workers.

Figure 3.1 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 (3.2) 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 (3.1), $\vec{x}^{(k)}_{i\to j}$ 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.

Algorithm 3.1 Multiple Iterative Algorithm

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

The maximum number of iterations k_{max} is analyzed in *Section 3.5.2*. 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, 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, $\vec{e}_{u_i}$.

3.3.3 Task Allocation for General Setting

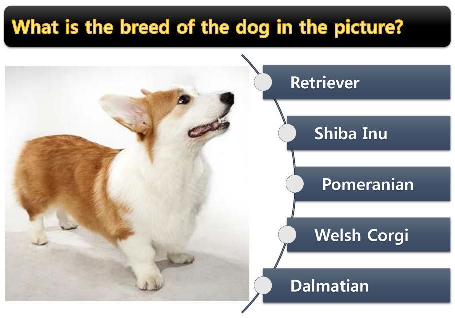
In the previous section, we proposed our iterative algorithm for a bipartite graph according to the *pairing model*. However, the number of workers allocated to each task can differ in cases that are more general. That must bring about the variation of the number of tasks that each worker solves. Hence, 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.

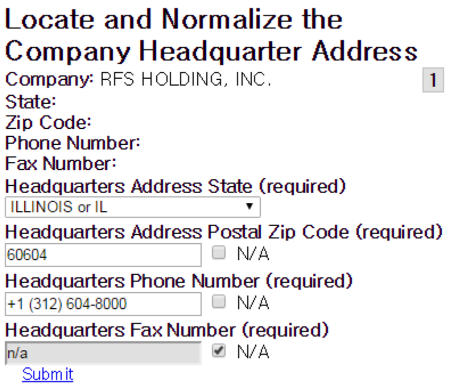
In addition to the generalization of the degree profile, we consider the various number of choices for each task (For example $^\forall i \in [m]$, $D_i \in \{2, 3, 4\}$). In practice, the number of choice for each task can differ from one another and our Algorithm 2 can cope with this variation. The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

Algorithm 3.2 Generalized Multiple Iterative Algorithm

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

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.




Locate and Normalize the Company Headquarter Address

Company: RFS HOLDING, INC. 1

State:

Zip Code:

Phone Number:

Fax Number:

Headquarters Address State (required)

ILLINOIS or IL

Headquarters Address Postal Zip Code (required)

60604 N/A

Headquarters Phone Number (required)

+1 (312) 604-8000 N/A

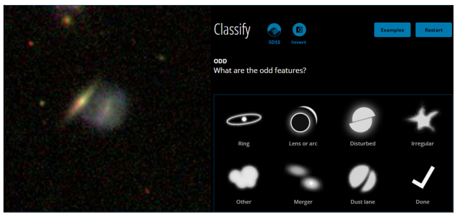
Headquarters Fax Number (required)

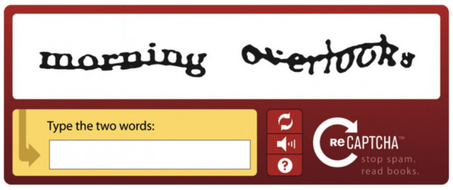
n/a ✓ N/A

Submit

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

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





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

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

Figure 3.2: Examples of various crowdsourcing tasks.

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 3.6*, the adaptive task allocation method will be explained in detail and we will verify some of the gains of this method through several experiments.

3.4 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 [34, 32] 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 3.2(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 3.2(c)). 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 3.2(b), entering zip codes or phone numbers is an example of short-answer questions.

Another popular crowdsourcing application for short-answer questions is reCAPTCHA [43] illustrated in Figure 3.2(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 Recognition (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.

3.5 Analysis of Algorithms

In this section, we provide proof for the performance guarantee of Algorithm 1. In *Theorem 3.1*, 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 $i \in [n]$.

3.5.1 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

$$q = \mathbb{E}\Big[H(p) - \bar{p}\log(\hat{D}) + \log(D)\Big], \tag{3.3}$$

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

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.

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 3.3. 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.

$$q \simeq \tilde{q}_1 = \mathbb{E}\left[\Big(\frac{D}{D-1}\Big)^2\Big(p_j - \frac{1}{D}\Big)^2\right] \tag{3.4}$$

For simplicity, we will use this approximated quality in the following sections. There is one more necessary assumption about worker distribution that workers give

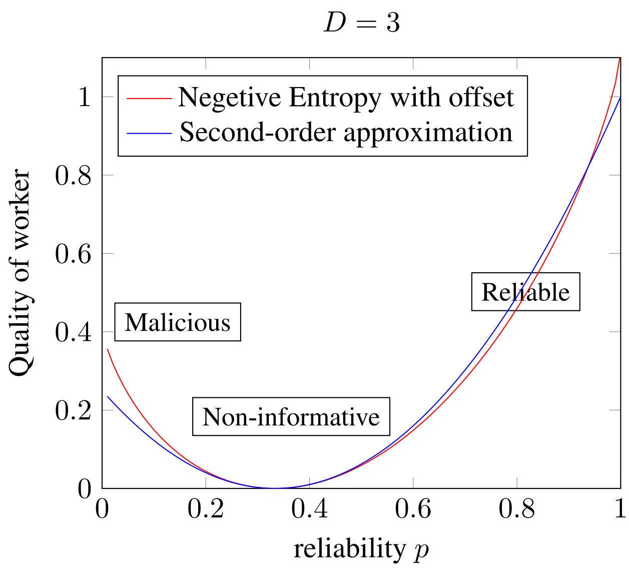


Figure 3.3: Comparison of the quality between negative entropy with offset and second-order polynomial approximation.

the correct answers on average rather than random or adversarial answers, so that $\mathbb{E}[p_j] > \dfrac{1}{D}$. 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 $\mathbb{E}[p_j] > \dfrac{1}{D}$ is inevitably necessary.

3.5.2 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.

$$\sigma_k^2 \equiv \frac{2q}{\mu^2 T^{k-1}} + \left(\frac{D}{D-1}\right)^2 \left(3 + \frac{1}{8q\hat{r}}\right) \left[\frac{1 - 1/T^{k-1}}{1 - 1/T}\right], \tag{3.5}$$

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

Theorem 3.1. *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*

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leqslant (D-1)e^{-lq/(2\sigma_k^2)} + \frac{3lr}{m}(\hat{l}\hat{r})^{2k-2}. \tag{3.6}$$

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 (3.5), $T = 1$ 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

$$\sigma_\infty^2 = \left(3 + \frac{1}{8q\hat{r}}\right)\left(\frac{T}{T-1}\right). \tag{3.7}$$

Thus, the bounds of the first term of (3.6) does not depend on the number of tasks m or the number of iterations k. The following *Corollary 3.1.1* describes an upper bound that only depends on l, q, σ_∞^2, and D.

Corollary 3.1.1. *Under the hypotheses of Theorem 3.1, there exists $m_0 = 3lre^{lq/4\sigma_\infty^2}(\hat{l}\hat{r})^{2(k-1)}$ and $k_0 = 1 + (\log(q/\mu^2)/\log T)$ such that*

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leqslant De^{-lq/(4\sigma_\infty^2)}, \tag{3.8}$$

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 (3.6) 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 $m \geqslant 3lre^{lq/4\sigma_\infty^2}(\hat{l}\hat{r})^{2(k-1)}$ to ensure $\frac{3lr}{m}(\hat{l}\hat{r})^{2k-2} \leqslant e^{-lq/(4\sigma_\infty^2)}$. □

From *Corollary 3.1.1*, 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 3.1.1*, the experimental result in *Section 3.6* shows that the performance of error exhibits exponential decay as stated in (3.8).

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 [35], we get the following bound:

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leqslant De^{-lq/32}. \tag{3.9}$$

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 3.1.2*.

Corollary 3.1.2. *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* $m \geqslant m_0$.

3.5.3 Proof of the Error Bounds

The proof is roughly composed of three parts. First, the second term at the right-hand side of (3.6) is proved using its local tree-like property. Second, the remaining term of the right-hand side of (3.6) 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 [35].

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 3.5.2. If we draw a task $\mathbf{I}$, uniformly at random from the task set, the average probability of error can be denoted as

$$\frac{1}{m}\sum_{i\in[m]}\mathbb{P}(t_i \neq \hat{t}_i^{(k)}) = \mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}), \tag{3.10}$$

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.

$$\begin{aligned}\mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}) &\leqslant \mathbb{P}(\mathrm{G}_{\mathbf{I},2k-1} \text{ is not a tree }) \\ &+ \mathbb{P}(\mathrm{G}_{\mathbf{I},2k-1} \text{ is a tree and } t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}).\end{aligned} \tag{3.11}$$

The following *Lemma 3.2* 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 3.2* is provided [35] (cf. *Karger, Oh and Shah* 2014, *Section 3.2*).

Lemma 3.2. *From a random (l,r)-regular bipartite graph generated according to the pairing model,*

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

From the result of *Lemma 3.2*, we can concentrate directly on the second term of (3.11) 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 3.5.4*. 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:

$$\mathbb{E}[e^{\lambda \mathbf{z}}] \leqslant e^{m\lambda + (1/2)\tilde{\sigma}^2\lambda^2}. \tag{3.12}$$

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

$$\begin{aligned} m_k &\equiv \mu \hat{l} U^{k-1}, \quad ^\forall k \in \mathbb{N} \\ \tilde{\sigma}_k^2 &\equiv 2\hat{l}S^{k-1} + [\mu^2 \hat{l}^2 \hat{r}(3q^2\hat{l}\hat{r} + \hat{l}/8)]U^{2k-4}\left[\frac{1-(1/T)^{k-1}}{1-(1/T)}\right], \\ \text{where } U &= \frac{D-1}{D} q\hat{l}\hat{r}, \quad S = \frac{D^2-D-1}{D^2}\hat{l}\hat{r} \\ T &= \frac{U^2}{S} = \frac{(D-1)^2}{D^2-D-1} q^2 \hat{l}\hat{r} \end{aligned}$$

then

$$\mathbb{E}[e^{\lambda \tilde{\mathbf{x}}_d^{(k)}}] \leqslant e^{m_k\lambda + (1/2)\tilde{\sigma}_k^2\lambda^2}. \tag{3.13}$$

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

$$\mathbb{P}(\hat{\mathbf{x}}_d^{(k)} \leqslant 0) \leqslant \mathbb{E}[e^{\lambda \hat{\mathbf{x}}_d^{(k)}}] \leqslant e^{-lm_k^2/(2\hat{l}\tilde{\sigma}_k^2)}. \tag{3.14}$$

$$\begin{aligned} \mathbb{P}(\cup_{d=2}^{D}\{\hat{\mathbf{x}}_d^{(k)} \leqslant 0\}) &\leqslant \sum_{d=2}^{D} \mathbb{P}(\hat{\mathbf{x}}_d^{(k)} \leqslant 0) \\ &\leqslant (D-1)e^{-lm_k^2/(2\hat{l}\tilde{\sigma}_k^2)}. \end{aligned} \tag{3.15}$$

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 3.1.*

3.5.4 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 $\mathbf{y}_\mathbf{p}$ are stated as

$$\begin{aligned}\mathbb{E}[e^{\lambda \tilde{\mathbf{x}}_d^{(k)}}] &= \Bigg(\mathbb{E}_\mathbf{p}\Bigg[\mathbf{p}\mathbb{E}\Big[e^{\lambda \mathbf{y}_p^{(k-1)}}|\mathbf{p}\Big] + \frac{\bar{\mathbf{p}}}{D-1}\mathbb{E}\Big[e^{-\lambda \mathbf{y}_p^{(k-1)}}|\mathbf{p}\Big] \\ &\quad + \frac{\bar{\mathbf{p}}}{D-1}(D-2)\Bigg]\Bigg)^{\hat{l}}, \end{aligned} \tag{3.16}$$

$$\begin{aligned}\mathbb{E}[e^{\lambda \mathbf{y}_p^{(k)}}] &= \Bigg(p\mathbb{E}\Big[e^{\lambda(\frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}_d^{(k)})}\Big] \\ &\quad + \frac{\bar{p}}{D-1}\sum_{j=2}^{D}\mathbb{E}\Big[e^{\lambda(-\tilde{\mathbf{x}}_j^{(k)} + \frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}_d^{(k)})}\Big]\Bigg)^{\hat{r}}, \end{aligned} \tag{3.17}$$

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

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

$$\begin{aligned}\mathbb{E}[e^{\lambda \tilde{\mathbf{x}}_d^{(1)}}] &= \Bigg(\mathbb{E}_\mathbf{p}\Bigg[\mathbf{p}e^{\lambda+(1/2)\lambda^2} + \Big(\frac{1-\mathbf{p}}{D-1}\Big)e^{-\lambda+(1/2)\lambda^2} \\ &\quad + \Big(\frac{1-\mathbf{p}}{D-1}\Big)(D-2)\Bigg]\Bigg)^{\hat{l}} \\ &\leqslant \Big(\mathbb{E}[a]e^{\lambda} + \Big(\mathbb{E}[\bar{a}]e^{-\lambda}\Big)^{\hat{l}}e^{(1/2)\hat{l}\lambda^2} \\ &\leqslant e^{(\mu\lambda+\lambda^2)\hat{l}}, \end{aligned} \tag{3.18}$$

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

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

$$be^{z} + (1-b)e^{-z} \leqslant e^{(2b-1)z+(1/2)z^2}, \tag{3.19}$$

for any $z \in \mathbb{R}$ and $b \in [0,1]$ (cf. *Alon and Spencer* 2008, *Lemma A.1.5*) [44].

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 (3.19) into (3.17), we have

Lemma 3.3. *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 (3.18)'s process, from (3.16), we get

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

with $2 \leqslant e^{\lambda} + e^{-\lambda}$ for any $\lambda \in \mathbb{R}$.

Substituting the result of *Lemma 3.3* into the above inequality provides

$$\begin{aligned}\mathbb{E}[e^{\lambda \tilde{\mathbf{x}}_d^{(k+1)}}] \leqslant \mathbb{E}_{\mathbf{p}}\Bigg[&a\Big(\mathbf{p}e^{(1/2)m_k\lambda} + \bar{\mathbf{p}}e^{-(1/2)m_k\lambda}\Big)^{\hat{r}} \\ &+\bar{a}\Big(\mathbf{p}e^{(1/2)m_k\lambda} + \bar{\mathbf{p}}e^{-(1/2)m_k\lambda}\Big)^{\hat{r}}\Bigg]^{\hat{l}} \\ &\cdot e^{(\frac{D-2}{2D})\hat{l}\hat{r}m_k\lambda+(\frac{D^2-D-1}{D^2})\hat{l}\hat{r}\tilde{\sigma}_k^2\lambda^2}.\end{aligned} \tag{3.20}$$

Now we are left to bound (3.20) using following *Lemma 3.4*.

Lemma 3.4. *For any $|z| \leqslant 1/(2\hat{r})$ and $p \in [0,1]$, we get*

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

Applying this to (3.20) 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*

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

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

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 $\tilde{\sigma}_k^2$, we obtain

$$\tilde{\sigma}_k^2 = a\tilde{\sigma}_{k-1}^2 + bc^{k-2}, \tag{3.21}$$

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.

$$\tilde{\sigma}_k^2 = \tilde{\sigma}_1^2 a^{k-1} + bc^{k-2}\Big[\frac{1-(a/c)^{k-1}}{1-(a/c)}\Big]. \tag{3.22}$$

This finishes the proof of (3.13).

Proof of *Lemma 3.3* In the $k+1^{th}$ inductive step of *mathematical induction*, we assume that $\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}] \leqslant e^{m_k\lambda+(1/2)\tilde{\sigma}_k^2\lambda^2}$ for any $d \in [2:D]$ with $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$. In other words, all of $\tilde{\mathbf{x}}_d^{(k)}$ follow *sub-Gaussian* distribution with parameters m_k and $\tilde{\sigma}_k^2$. From (3.17), each component at the right-hand side can be represented as the product of several combinations of $[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}]$ 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 $\tilde{\sigma}_k^2$ of each *sub-Gaussian* $\tilde{\mathbf{x}}_d^{(k)}$. Applying *h"older's inequality* to (3.17), the first term at the right-hand side of (3.17) gives

$$\mathbb{E}\left[e^{\lambda(\frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}_d^{(k)})}\right] \leqslant \prod_{d=2}^{D}\left[\mathbb{E}\left(e^{\lambda(1/D)\tilde{\mathbf{x}}_d^{(k)})}\right)^{D-1}\right]^{\frac{1}{D-1}}$$
$$\leqslant e^{(\frac{D-1}{D})m_k\lambda+(\frac{D-1}{2D^2})\tilde{\sigma}_k^2\lambda^2}.$$

For the second term at the right-hand side of (3.17), we have

$$\mathbb{E}\left[e^{\lambda(-\tilde{\mathbf{x}}_j^{(k)}+\frac{1}{D}\sum_{d=2}^{D}\tilde{\mathbf{x}}_d^{(k)})}\right] \leqslant \mathbb{E}\left[e^{-\lambda(\frac{D-1}{D})\tilde{\mathbf{x}}_j^{(k)}}\right]$$
$$\cdot \prod_{d=2,d\neq j}^{D}\left[\mathbb{E}\left(e^{\lambda(1/D)\tilde{\mathbf{x}}_d^{(k)})}\right)^{D-1}\right]^{\frac{1}{D-1}}$$
$$\leqslant e^{(-\frac{1}{D})m_k\lambda+(\frac{D^2-D-1}{2D^2})\tilde{\sigma}_k^2\lambda^2}.$$

Getting these two results together finishes the proof of *Lemma 3.3*.

Proof of *Lemma 3.4* From (3.19), we get

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

Applying this result to the original formula, we have

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

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

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

$$
\begin{aligned}
&\leqslant \mathbb{E}\Big[1+\Big(\frac{D-1}{D}\Big)q\hat{r}z+\frac{1}{2}\Big(\frac{D-1}{D}\Big)^2 q\hat{r}^2z^2 \\
&\quad +0.63\Big\{\Big(\frac{D-1}{D}\Big)q\hat{r}z+\frac{1}{2}\Big(\frac{D-1}{D}\Big)^2 q\hat{r}^2z^2\Big\}^2\Big] \\
&\leqslant 1+\Big(\frac{D-1}{D}\Big)q\hat{r}z+\frac{3}{2}\Big(\frac{D-1}{D}\Big)^2 q\hat{r}^2z^2 \\
&\leqslant e^{\left(\frac{D-1}{D}\right)q\hat{r}z+\frac{3}{2}q\hat{r}^2z^2},
\end{aligned}
$$

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

Phase Transition. As shown in (3.22), the performance of our algorithm is only bounded when the condition $T > 1$ is satisfied. Meanwhile, with $T < 1$, $\tilde{\sigma}_k^2$ which means the variance of the $\tilde{\mathbf{x}}_d^{(k)}$ 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 3.6*, we will verify the existence of this critical point at $T = 1$ through several experiments with different conditions.

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

3.6 Experimental Results

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.

3.6.1 Comparison with Other Algorithms

To show the competitiveness of our algorithm, we ran our *multiple iterative algorithm*, majority voting, and the EM approach for 2,000 tasks and 2,000 workers with fixed $D = 2$, 3, 4, and 5 (Figure 3.4 and Figure 3.5). The performance of the oracle estimator is also presented as a lower bound and the EM algorithm is implemented with Dawid and Skene's method [38]. In Figure 3.4, 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 3.5, 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 3.4, 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 3.5. 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 3.4, carrying off the required $r(= l)$ is accomplished easily since each alphabet is considered as a separate question.

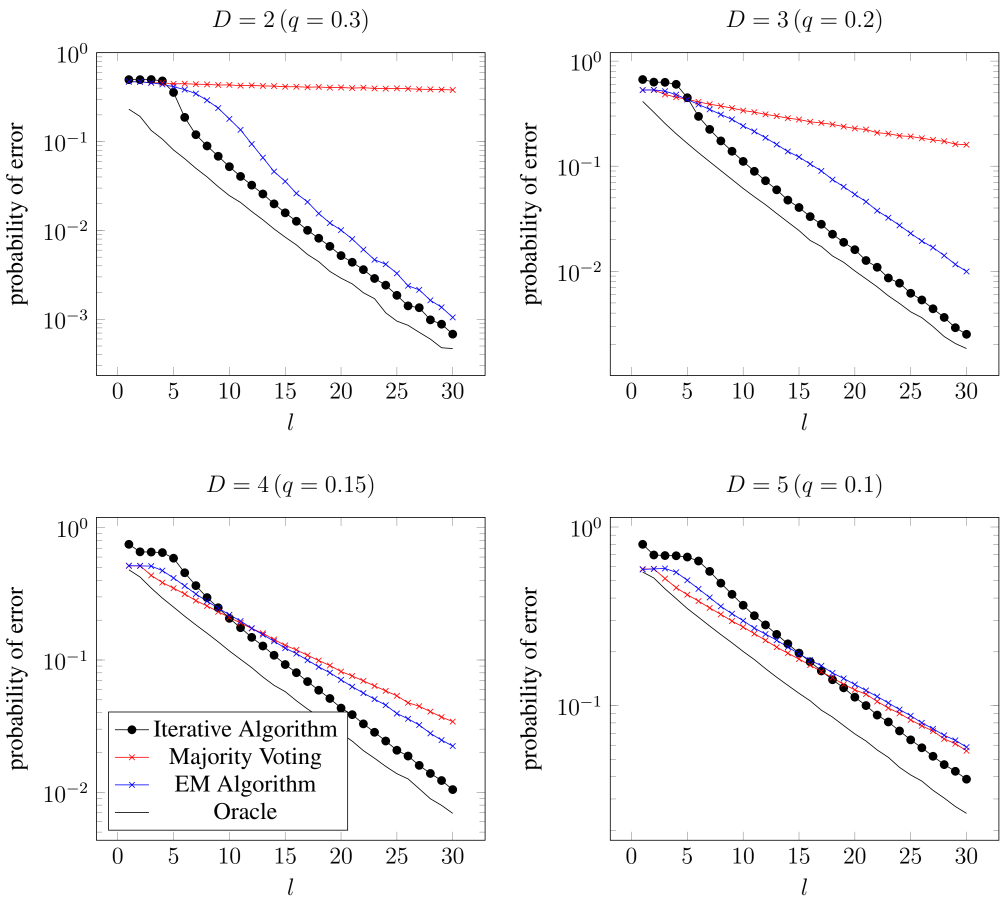


Figure 3.4: Comparisons of probabilities of error between different algorithms varying l values ($m = n = 2000$, $l = r$).

3.6.2 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

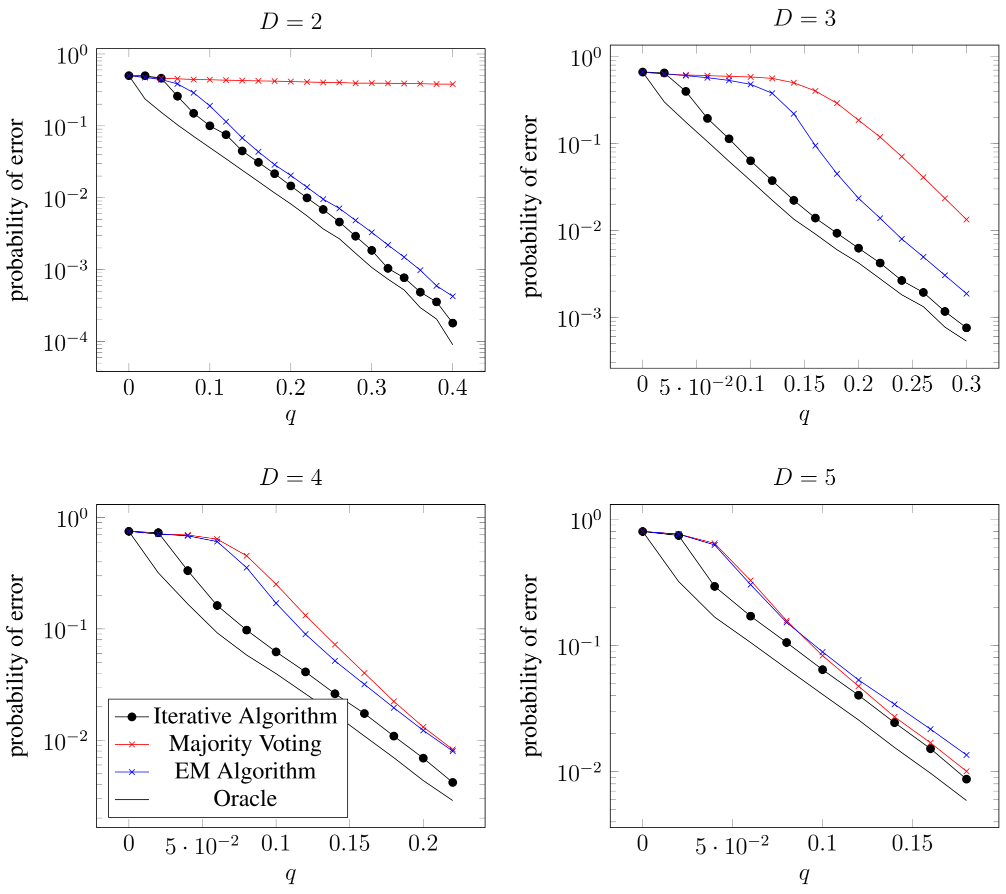


Figure 3.5: Comparisons of probabilities of error between different algorithms varying q values ($m = n = 2000$, $l = r = 25$).

y_j is calculated by the following equation in Algorithm 3.1.

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

Figure 3.6 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

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

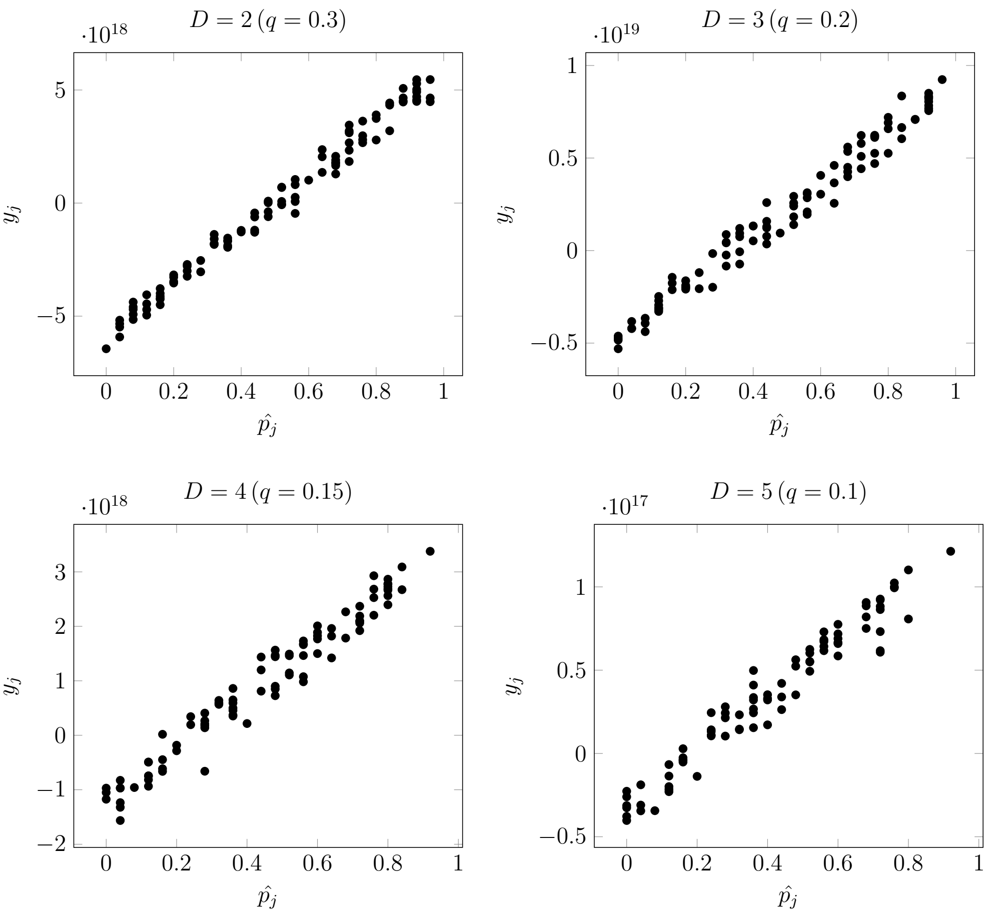


Figure 3.6: Relationship between y_j and $\hat{p_j}$ ($m = n = 2000$, $k = 10$).

non-informative worker's reliability, as expected in Section 3.5.

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 3.6 and Figure 3.3).

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

To show the performance improvements when using the adaptive scenario, we perform experiments with several (m', β) sets. Figure 3.7 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.

3.6.3 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 3.2. In detail, we repeat the same experiments with a question set composed in $1 : 1 : 1$ ratios of tasks

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

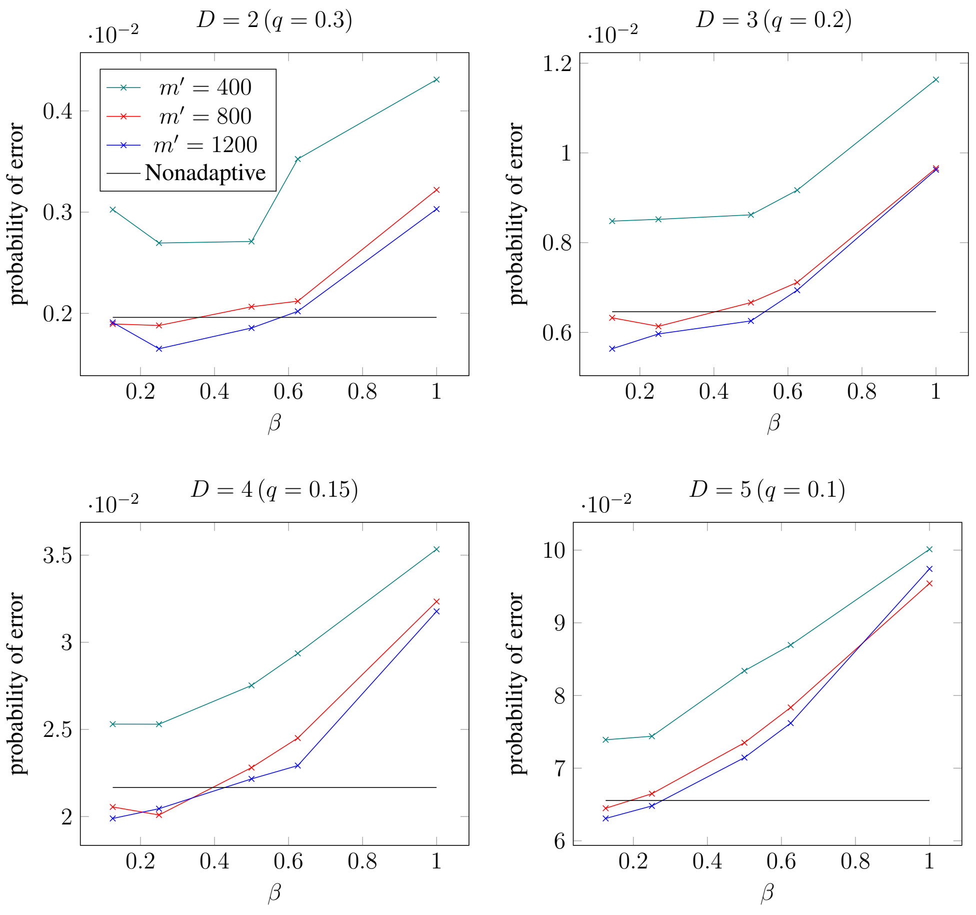


Figure 3.7: Adaptive Scenario ($m = n = 2000$, $l = 25$).

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 3.8. There is also the

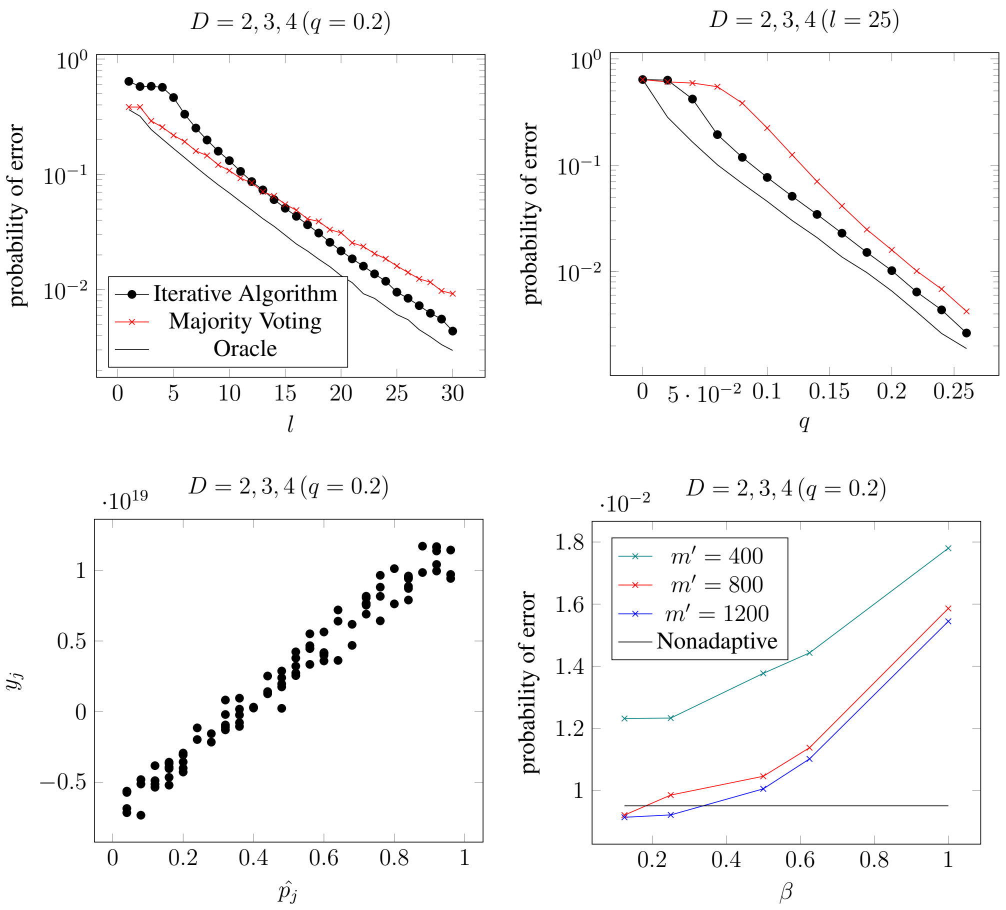


Figure 3.8: Simulations on a set of various D values ($m = n = 2000$, where $D = 2$: 666 / $D = 3$: 667 / $D = 4$: 668).

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.

3.7 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.

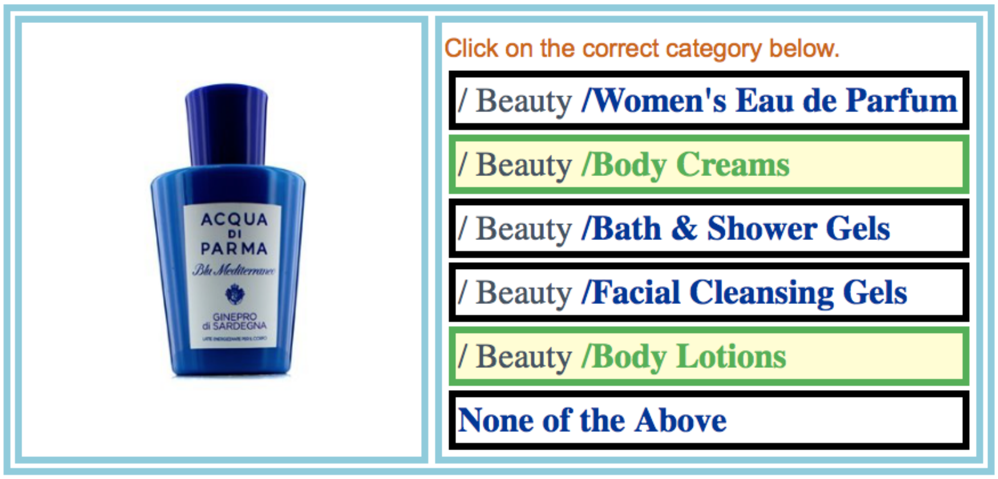
Especially for short-answer questions, we provide a method of transforming original tasks into several microtasks. Therefore, we give a general solution for real crowdsourcing systems to infer the correct answers from unreliable responses. From the performance analysis of our algorithm, we have proved 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.

Chapter 4

Crowdsourcing Systems for Multiple-choice Questions with K-Approval Voting

In eliciting the wisdom of crowds, many crowdsourcing platforms have encourage workers to select top-K alternatives they believe as correct candidates. This voting rule is called "K-approval voting" and its interface provides workers more flexibility to response and takes advantage of even their partial expertise [45, 46]. Due to the above merits, many crowdsourcers have adopt the K-approval voting setup to collect large amount of responses. For real crowdsourcing examples, two tasks described in Figure 4.1, 4.2 are real world crowdsourcing examples of K-approval voting. Figure 4.1 shows a task being distributed on Amazon Mechanical Turk, and the task was requested by one of well-known online shopping mall, Amazon. The goal of the task is to classify the best category of the item in the picture from given alternatives. Figure 4.2 shows another real task named 'Wisconsin Wildlife Watch' on Zooniverse which is another popular crowdsourcing system. The goal of this task is to correctly figure out what animals were pictured at the Wisconsin wildlife.

Although the demand on K-approval voting increases, there is not much work on inferring the correct answer from the collected data via a K-approval voting. One natural method to aggregate responses is the majority voting. However, it is insufficient to


Click on the correct category below.

/ Beauty /Women's Eau de Parfum

/ Beauty /Body Creams

/ Beauty /Bath & Shower Gels

/ Beauty /Facial Cleansing Gels

/ Beauty /Body Lotions

None of the Above

Figure 4.1: A task in the Amazon Mechanical Turk, whose goal is to categorize the item in the picture. The worker is allowed to choose two candidates.

exploit the wisdom of crowds appropriately since it assumes that the level of reliability of the responses of all the workers are the same. In fact it is recently proved that [46] the majority voting is sub-optimal for any K-approval voting systems.

In this study, we design a novel algorithm for K-approval-voting systems which evaluates workers' reliability to infer the correct answers of the tasks more precisely. This work can be generally applicable on real crowdsourcing platforms in practice where tasks are D many multiple-choice questions allowing workers to select top-K alternatives. Moreover, our algorithm can be applied to the case that each problem have different (D, K) value.

To the best of our knowledge, this study is the first work which propose an inference algorithm for K-approval votes. One of main contributions in this study is the performance guarantee of our algorithm. We rigorously prove 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. Also, we verify the performance of our algorithm through numerical experiments on

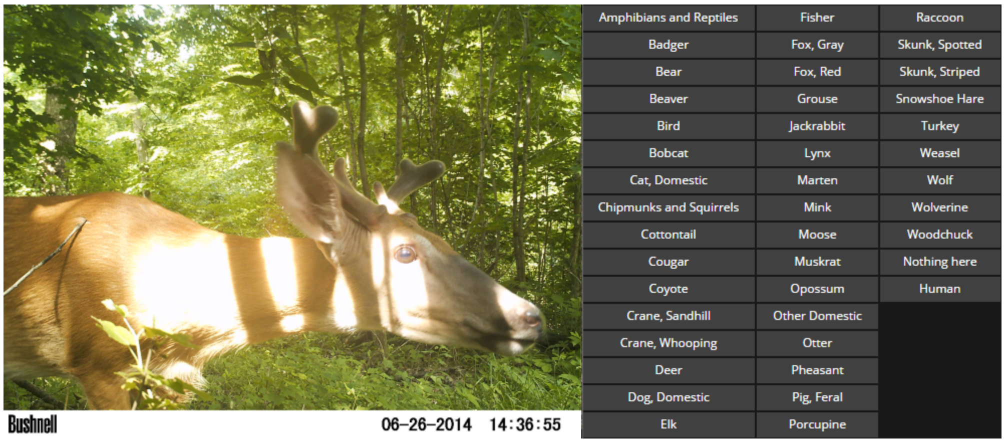


Figure 4.2: A task being distributed on Zooniverse to correctly figure out what animals are pictured at Wisconsin wildlife. The worker is allowed to choose multiple candidates.

various cases including a realistic case containing mixed tasks with various number D of alternatives and K selections. Moreover, through experiments, we show that our algorithm estimates the relative reliabilities of the workers properly.

This chapter is organized as follows: In *Section 4.2*, we make a setup, and in *Section 4.3*, we describe our algorithm to infer the correct answers for K-approval votes. Then, we provide performance guarantees for our algorithm in *Section 4.4*. In *Section 4.5*, we present comparative results through numerical experiments, and we draw conclusions in *Section 4.6*.

4.1 Related Work

Recently, there has been a few researches about K-approval votes in crowdsourcing field. These works aim to elicit the mode of worker's belief and propose a new paradigm of amassing high quality responses. Specially, In [45], they endeavor to obtain higher-quality labels with a new incentive mechanism which encourages the good workers with additional payments. In addition, [46] prove that simple majority voting is sub-optimal and they bring up a conversation topic of necessity of probabilistic inference algorithm which can be applied to K-approval voting.

In single choice voting, there have been various approaches to get reliable results from unreliable responses. The simplest one is majority voting. However, it is insufficient to get reliable results since it regards the expertise of each worker as equal and gives the same weight to every worker. Typically, the level of expertise are very different from experts to novices. free money collectors, and even adversarial workers [29]. In order to exploit difference of expertise among workers, EM-based algorithms are suggested with latent variables and unknown model parameters [38, 32, 31, 30, 47, 48, 49].

Alternative approaches for the single voting for the binary questions have been suggested in [33] in the context of spectral methods that use low-rank matrix approximations. Also, they proposed a novel iterative learning algorithm [34, 33, 37] inspired by the standard *Belief Propagation* (BP) algorithm. Although they did not assume any prior knowledge, Liu et al. [36] shows that choosing a suitable prior can improve the performance via a Bayesian approach. Lately, [50, 51] proposed an iterative learning algorithm for single voting on multiple-choice questions and real-valued vector regressions respectively. However, their algorithm cannot be applied to the K-approval voting.

In recent years, there have been some researches that build models jointly combined with human and computer vision model. [52] propose online collaboration crowdsourcing system between crowdsourcing platform and computer vision machines with Bayesian predictive model. [53, 54] develops the combining system using confidence modeling and applied these systems to various applications in binary and multiclass setting.

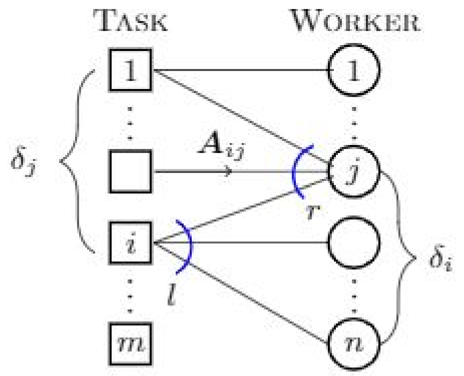


Figure 4.3: System model for task-worker assignments.

4.2 Problem Setup

4.2.1 Problem Definition

In this section, we define the setup of the problem. Suppose a crowdsourcing system where there are m tasks to be solved and n workers to participate in. For convenience, tasks and workers are indexed by $i \in [m]$ and $j \in [n]$ respectively where $[m]$ means a set of integers from 1 to m such that $\{1, ..., m\}$.

Each task is a multiple-choice question and consists of D_i alternatives (or options) with only one correct answer (correct alternative). Each task is duplicated l times to make a redundancy for boosting performance, which is a common strategy in crowdsourcing systems. Therefore, in total, ml tasks are to be distributed. Since we use a random and equal assignment strategy to distribute tasks, each worker gets to solve r tasks such that $ml = nr$. If we draw this scheme as a graph, a random (l, r)-regular bipartite graph can be made by the pairing model with m task nodes, n worker nodes, and $ml(= nr)$ edges representing tasks assigned to workers.

Each task allows K_i-approval votes when workers make their own responses. Here, we need to consider a worker model how workers make their responses when solving tasks. For a simple probabilistic approach, we assume that worker j solves task i, and gives a response including the correct answer with a probability $p_{ij} \in [0, 1]$. This probability should be considered as a value which depends on the type of given task such as D_i and K_i, and how reliable the worker is. We will discuss its dependency on the task and worker in *Section 4.2.2*. We also assume that distractors (a set of alternatives except the correct answer) are independent from each other, and their levels of difficulty are same. Thus, in the K-approval voting system, a response include the correct answer with probability of p_{ij}, and the rest alternatives in the response are randomly selected from the distractors.

The goal of the problem is to infer the correct answer of tasks when workers' responses are given. The error performance can be easily measured by the ratio of the tasks incorrectly inferred such that $\frac{1}{m}\sum_{i\in[m]}\mathbb{I}(t_i \neq \hat{t}_i)$ where t_i and $\hat{t}_i$ are the correct and inferred answer of task i respectively, and $\mathbb{I}$ is an indicator function.

4.2.2 Worker Model for Various (D, K)

In our model, responses are generated by a simple probabilistic approach, so reliability should be calculated for every task and worker. To verify this, we can consider two easy examples. Suppose that there is an worker who only gives a random response, and two tasks are given to the worker where both (D, K)-pairs of the tasks are $(2, 1)$, $(3, 1)$ respectively. Then we can easily calculate the probability of getting a response including the correct answer from the worker. Since the worker picks a random choice, reliabilities for two tasks should be set to 0.5 and 0.33 respectively. For another example, suppose that the same worker solves other two tasks whose (D, K)-pairs are $(3, 1)$, $(3, 2)$ respectively. Then, reliabilities for two tasks should be set to 0.33 and 0.67 respectively. Therefore we conclude that reliability should be dependent on D_i and K_i.

Here we assume that each worker has an intrinsic value q_j called quality of the worker, which represents inherent diligence or expertise of the worker. This means that the quality of the worker should be fixed whatever types of tasks are given. From an information theoretical perspective, the quality of a worker can be defined as negative entropy with an offset which makes the amount set to zero when the worker gives random responses. This is reasonable since no information can be obtained from random responses. We define quality of workers more precisely and relationship between p_{ij} and q_j in Chapter 4.4.2 and skip mathematical analysis here to take a look at our algorithm first.

4.3 Inference Algorithm

In this section, we explain the design of our algorithm and the process to estimate the correct answers. To avoid confusion in notation, we use the upper case K to represent the number of selections and the lower case k to represent the iteration number of our algorithm. In our setting, workers whom task i is assigned to are allowed to vote K_i alternatives.

For each edge (i, j), a response is denoted as

$$\vec{A}_{ij} \in \mathrm{A} = \left\{ \sum_{s=1}^{K_i} \vec{e}_s \middle| \vec{e}_s \in U_{D_i} \right\}$$

where $\mathbb{U}_{D_i} = \{\vec{e}_u | u \in [1 : D_i]\}$ comprising D_i dimensional unit vectors. To extract the correct answers from unreliable responses of workers, we propose an iterative algorithm for K-approval voting systems. Our algorithm takes an advantage of two types of messages between a task node and a worker node. A task message is denoted as a D_i dimensional vector, $\vec{x}_{i\to j}$. Each component of this vector corresponds to the likelihood meaning the possibility being the correct answer of task i. A worker message, $y_{j\to i}$, represents how reliable the worker j is. Since these worker messages are strongly correlated with the reliability p_{ij}, our algorithm can assess relative reliability. We will empirically verify the correlation between $\{y_{j\to i}\}$ and $\{p_{ij}\}$ in *Section 4.5*.

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 [38, 32], our approach is not sensitive to initial conditions as

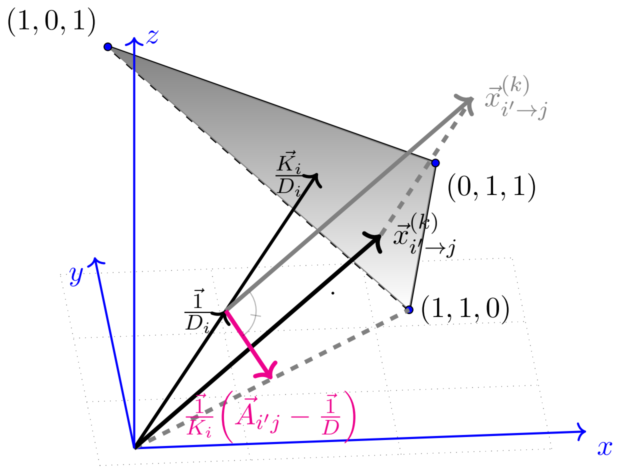

$(1,0,1)$
z
$\vec{x}^{(k)}_{i'\to j}$
$\frac{\vec{K}_i}{D_i}$
$(0,1,1)$
$\vec{x}^{(k)}_{i'\to j}$
y
$\frac{\vec{1}}{D_i}$
$(1,1,0)$
$\frac{1}{K_i}\left(\vec{A}_{i'j} - \frac{\vec{1}}{D}\right)$
x

Figure 4.4: 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 $\vec{A}_{i'j} = (1,1,0)$ and $D = 3$, $K = 2$.

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} iteration, both messages are updated using the following rules: For $\forall (i,j) \in E$,

$$\vec{x}^{(k)}_{i\to j} = \sum_{j' \in \partial i \backslash j} \frac{1}{K_i}\left(\vec{A}_{ij'} y^{(k-1)}_{j'\to i}\right), \tag{4.1}$$

$$y^{(k)}_{j\to i} = \sum_{i' \in \partial j \backslash i} \frac{1}{K_{i'}}\left(\vec{A}_{i'j} - \left(\frac{K_{i'}}{D_{i'}}\right)\vec{1}\right) \cdot \vec{x}^{(k-1)}_{i'\to j}. \tag{4.2}$$

At the task message update process shown in (4.1), our algorithm gives a weight to the response according to the relative reliability of a worker. At the worker message update process shown in (4.2), it gives greater relative reliability to a worker who strongly follows the consensus of other workers.

Figure 4.4 describes two vectors in the message vector space. As shown above, $\frac{1}{K_{i'}}\left(\vec{A}_{i'j} - (\frac{K_{i'}}{D_{i'}})\vec{1}\right)$ represents the difference between the response of worker j solving task i' and the expectation of a random response $\left(\frac{K_{i'}}{D_{i'}}\right)\vec{1}$ with normalizing factor $\frac{1}{K_{i'}}$. Also, $\vec{x}^{(k-1)}_{i'\to j}$ means weighted sum of responses of other workers who have solved the task i'. Thus, the inner product of these two vectors in (4.2) 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 means that the worker j does not follow the consensus of other workers, and our algorithm regards the worker j unreliable. Specially, when $\vec{x}^{(k-1)}_{i'\to j}$ and $\frac{1}{K_{i'}}\big(\vec{A}_{i'j} - (\frac{K_{i'}}{D_{i'}})\vec{1}\big)$ 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 inner products from each task message except for that of task i, representing the relative reliability of the worker j. Returning to (4.1), $\vec{x}^{(k)}_{i\to j}$ 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 of $y^{(k-1)}_{j'\to i}$. 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 likelihood on the correct answer for the task i.

The following describes the pseudo code of our algorithm.

In practice, a dozen of iterations are sufficient for the convergence of our algorithm. After k_{max} iterations, our algorithm makes the final estimate vector $\vec{x}_i$ of the task i, and each component of the vector represents the possibility of being the correct answer. Our algorithm infers the correct answer by choosing u_i that has the maximum value among final likelihoods of $\vec{x}_i$. Then, our algorithm outputs the estimate of the correct answer denoted as a unit vector, $\vec{e}_{u_i}$.

4.4 Analysis of Algorithms

In this section, we verify the performance on the average error of Algorithm 4.1. In *Theorem 4.1*, we show that the error bound depends on the task degree l and the quality of workers q. Furthermore, we provide that an upper bound on the probability of error decays exponentially as the quality of workers increases. Here, we assume that $D_i = D$ and $K_i = K$ for all task $i \in [n]$ and accordingly $p_{ij} = p_j$. However, we will show

Algorithm 4.1 K-approval Iterative Algorithm

```
1:  Input: $E$, $\{\vec{A}_{ij}\}_{(i,j)\in E}$, $k_{max}$
2:  Output: Estimation $^{\forall}i \in [m]$ , $\hat{t}_i \in \{\vec{e}_{u_i} | u_i \in [1:D]\}$
3:  for $^{\forall}(i,j) \in E$ do
4:      Initialize $y^{(0)}_{j\to i}$ with random $Z_{ij} \sim N(1,1)$;
5:  end for
6:  repeat
7:      for $^{\forall}(i,j) \in E$ do
8:          $\vec{x}^{(k)}_{i\to j} \leftarrow \sum_{j'\in\partial i\setminus j} \frac{1}{K_i}\vec{A}_{ij'} y^{(k-1)}_{j'\to i}$;
9:          $y^{(k)}_{j\to i} \leftarrow \sum_{i'\in\partial j\setminus i} \frac{1}{K_i}(\vec{A}_{i'j} - (\frac{K_i}{D_i})\vec{1}) \cdot \vec{x}^{(k-1)}_{i'\to j}$;
10:     end for
11: until $k \leq k_{max}$
12: Final Estimation
13: for $^{\forall}i \in [m]$ do
14:         $\vec{x}_i \leftarrow \sum_{j\in\partial i} \frac{1}{K_i}\vec{A}_{ij} y^{(k_{max}-1)}_{j\to i}$;
15: end for
16: for $^{\forall}j \in [n]$ do
17:         $y_j \leftarrow \sum_{i\in\partial j} \frac{1}{K_i}(\vec{A}_{ij} - (\frac{K_i}{D_i})\vec{1}) \cdot \vec{x}^{(k_{max}-1)}_{i\to j}$;
18: end for
19: Estimated answer: $\hat{t}_i = \vec{e}_{u_i}$ where $u_i = \arg\max_d(\vec{x}_i)$
```

the performance of our algorithm with general scenarios in *Section 4.5*.

4.4.1 Worker Model

We can assume several worker models which reflect how workers behave when they get tasks to solve. Common methods to generate their responses are simple probabilistic approaches. Basic assumption is that a worker has latent variables which influences on generating worker's responses. Latent variables usually represent diligence or expertise of workers, or level of difficulty of given tasks. Recent model [46] uses a noise model which assumes that worker's responses are considered corrupted by a noise from true answer.

Since our interest is on approval voting, we use a simple probabilistic approach to generate workers' responses. If alternatives are independent from each other, and each distractor has same difficulty level to distinguish the correct answer, then it is reasonable to assume that there are only two probabilities to be determined. One is the probability when the correct answer exists in the worker's response, and another is the probability when worker cannot find the correct answer. Since probabilities should be sum to one in an event, just one probability is left to be determined. Although this assumption is too simple, it would be good enough to represent worker's behavior in that we should reduce parameters because we cannot gather much responses from workers. It is hard to learn all features each alternative have as we don't have much money to get enough information.

Additionally, for the same reason, we assume that all given tasks have same level of difficulty when the number of alternatives are same. On the other hand, if the number of alternatives varies from each task, it is reasonable to assume that the level of difficulty is not same. Suppose that there is an worker who only picks a random choice, and two tasks are given where one task has two alternatives and another has three alternatives. We can easily calculate the expectation of getting a response including the correct answer from the worker. The former task gives expectation of 0.5 and the latter task gives expectation of 0.33. This means that a task with higher number of alternatives are hard to solve than one with lower number of alternatives. Thus we have to determine the proper probabilities when workers give the correct answer in their responses according to tasks given to them.

Furthermore, the number allowed on approval votes also influences on the expectation of getting a response including the correct answer. Suppose that there are two tasks having three alternatives where they permits 1 and 2-approval votes respectively, and the tasks are given to the same worker who only picks a random choice. Calculating the expectation same as we did above, the former task gives 0.33, and the latter task gives 0.67. Therefore we also need to consider the effects of the number allowed on approval votes.

4.4.2 Quality of Workers

We can assume several worker models which reflect how workers behave when they get tasks to solve. Common methods to generate their responses are simple probabilistic approaches. Basic assumption is that a worker has latent variables which influences on generating worker's responses. Latent variables usually represent diligence or expertise of workers, or level of difficulty of given tasks. Recent model [46] uses a noise model which assumes that worker's responses are considered corrupted by a noise from true answer.

Here and after, we use $\mathbb{U}_D$ to denote a set of standard D-dimensional unit vectors. For a fixed (D, K), we model a task i associated with an unobserved "correct" solution $s_i \in \mathbb{U}_D$. $\vec{A}_{ij}$ denotes the response vector of each worker j and can be represented by sum of K number of vectors in $\mathbb{U}_D$ (K-approval setting). Each component of the response vector $\vec{A}_{ij}$ is binary (1 or 0). The worker j with reliability p_j has ${}_DC_K$ choices for a response, and those choices are divided to only two types as follows:

$$\vec{A}_{ij}(d) : d^{th} \text{ component of } \vec{A}_{ij},\ {}^{\forall}d \in [1 : D]$$

$$\vec{A}_{ij} \cdot s_i = \begin{cases} 1 & \text{with probability } \left(\frac{p_j}{{}_{D-1}C_{K-1}}\right) \\ 0 & \text{with probability } \left(\frac{1-p_j}{{}_{D-1}C_K}\right) \end{cases}$$

From an information theoretical perspective, the quality of workers can be defined as negative entropy with an offset. This offset makes the quality of worker with random response to set to zero. Using the probabilities above, we can express negative entropy and define the quality of worker j with reliability p_j as

$$q_j(p_j) = Q(p_j) - Q\Big(\frac{K}{D}\Big), \tag{4.3}$$

$$\text{where } Q(p) = p \log\left[\frac{p}{{}_{D-1}C_{K-1}}\right] + (1-p)\log\left[\frac{1-p}{{}_{D-1}C_K}\right].$$

According to the quality of each worker, we can divide the workers into three types. “Reliable workers” are workers with the a quality close to one make almost correct answers. At the extreme, workers with a quality close to zero make arbitrary responses and we define them as “Non-informative workers”. At the other extreme, 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 positive y_j, negative y_j and y_j close to zero correspond to “Reliable workers”, “Malicious workers”, and “Non-informative workers” respectively.

Although the quality of workers theoretically follows negative entropy, we found that a fourth order polynomial approximation is sufficient for our analysis as described in Figure 4.5. As the number of alternatives D increases, the approximation deviates from the real quality. Nevertheless, fourth order approximation fits well to the real quality in the acceptable D case in which our algorithm targets in general.

$$q \simeq \tilde{q} = \mathbb{E}\Big[f(p_j) + f(p_j)^2\Big], \tag{4.4}$$

$$\text{where } f(p) = \left[\Big(\frac{D}{D-1}\Big)^2\Big(p - \frac{K}{D}\Big)^2\right].$$

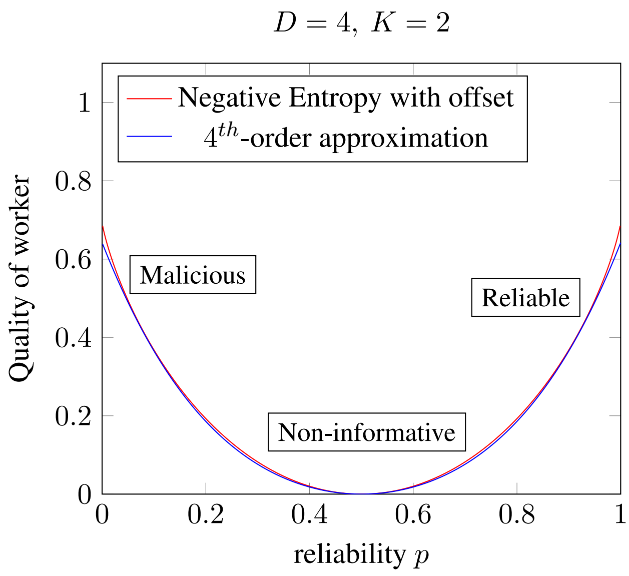


Figure 4.5: Comparison of the quality between negative entropy with offset and 4^{th}-order polynomial approximation.

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 $\mathbb{E}\left[p_j\right] > \dfrac{K}{D}$. Given only workers' responses, any inference algorithms analogize the correct 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 reliability of the crowds. Hence, the assumption $\mathbb{E}\left[p_j\right] > \dfrac{K}{D}$ is inevitably necessary.

4.4.3 Bound on the Average Error Probability

In this section, we show the error bound of our algorithm. *Theorem 4.1* claims that the probability of error decays exponentially as the quality of workers q or the degree of worker nodes l increases.

From now on, let $\hat{l} \equiv l - 1$, $\hat{r} \equiv r - 1$, and we use the quality q as defined in (4.4).

Also, σ_k^2 denotes an effective variance in the *sub-Gaussian* tail of the task message distribution after k iterations.

$$\sigma_k^2 \equiv \frac{2q}{\mu^2 T^{k-1}} + \frac{5}{8}\left(1 + \frac{5}{16}\left(\frac{D}{D-1}\right)^2 \frac{1}{q\hat{r}}\right)\left[\frac{1 - 1/T^{k-1}}{1 - 1/T}\right], \tag{4.5}$$

$$\text{where} \quad T = \frac{32}{25}\left(\frac{D-1}{D-K}\right)^2 q^2 \hat{l}\hat{r}.$$

Theorem 4.1. *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 generated according to the pairing model. If the distribution of the reliability satisfies $\mu \equiv \mathbb{E}[\frac{D}{D-1}(p_j - \frac{K}{D})] > 0$ and $T > 1$, then for any $t \in \{e_i\}^m$, the estimate after k iterations of the iterative algorithm achieves*

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{P}(t_i \neq \hat{t}_i^{(k)}) \leqslant (D-1)e^{-lq/(2\sigma_k^2)} + \frac{3lr}{m}(\hat{l}\hat{r})^{2k-2}. \tag{4.6}$$

The second term of the equation is the upper bound of probability that the graph dose not have a locally 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 (4.5), $T = 1$ is the crucial condition and we can satisfy $T > 1$ by using a sufficiently large l or r. Then, with $T > 1$, σ_k^2 converges to a finite limit σ_∞^2, and we have

$$\sigma_\infty^2 = \frac{5}{8}\left(1 + \frac{5}{16}\left(\frac{D}{D-1}\right)^2 \frac{1}{q\hat{r}}\right)\left[\frac{T}{T-1}\right]. \tag{4.7}$$

Thus, the bounds of the first term of (4.6) does not depend on the number of tasks m or the number of iterations k.

4.4.4 Proof of the Error Bounds

The proof is roughly composed of three parts. First, the second term at the right-hand side of (4.6) is proved using its locally tree-like property. Second, the remaining term of the right-hand side of (4.6) 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.

Without a loss of generality, it is possible to assume that the correct 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 4.4.3*. If we draw a task **I**, uniformly at random from the task set, the average probability of error can be denoted as

$$\frac{1}{m} \sum_{i \in [m]} \mathbb{P}(t_i \neq \hat{t}_i^{(k)}) = \mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}). \tag{4.8}$$

Let $\mathrm{G}_{\mathbf{I},k}$ denote a subgraph of the random (l, r)-regular bipartite graph that consists of all the nodes whose distance from the node '**I**' is at most k. After k iterations, the local graph with root '**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.

$$\begin{aligned} \mathbb{P}(t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}) \quad &\leqslant \quad \mathbb{P}(\mathrm{G}_{\mathbf{I},2k-1} \text{ is not a tree }) \\ &+ \quad \mathbb{P}(\mathrm{G}_{\mathbf{I},2k-1} \text{ is a tree \& } t_{\mathbf{I}} \neq \hat{t}_{\mathbf{I}}^{(k)}). \end{aligned} \tag{4.9}$$

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 4.2* is provided in [35] (cf. *Karger, Oh and Shah* 2011, *Section 3.2*).

Lemma 4.2. *From a random (l,r)-regular bipartite graph generated according to the pairing model,*

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

From the result of *Lemma 4.2*, we can concentrate on the second term of (4.9) 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 $\tilde{\mathbf{x}}_d^{(k)}$ follows *sub-Gaussian* distribution for any $d \in [2:D]$. 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:

$$\mathbb{E}[e^{\lambda \mathbf{z}}] \leqslant e^{m\lambda + (1/2)\tilde{\sigma}^2\lambda^2}. \tag{4.10}$$

We 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 a specific region of λ, precisely for $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$. Its proof is in the Supplementary material. Now we define

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

$$\tilde{\sigma}_k^2 \equiv 2\hat{l}S^{k-1} + \mu^2\hat{l}^3\hat{r}\left[\frac{2}{5}\Big(\frac{D-1}{D}\Big)^2 q\hat{r} + \frac{1}{8}\right] \cdot U^{2k-4}\left[\frac{1-(1/T)^{k-1}}{1-(1/T)}\right],$$

where $U = \frac{4}{5}\Big(\frac{D-1}{D}\Big)q\hat{l}\hat{r}, \quad S = \frac{1}{2}\Big(\frac{D-K}{D}\Big)^2\hat{l}\hat{r}$

$$T = \frac{U^2}{S} = \frac{32}{25}\Big(\frac{D-1}{D-K}\Big)^2 q^2\hat{l}\hat{r}$$

then

$$\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}] \leqslant e^{m_k\lambda + (1/2)\tilde{\sigma}_k^2\lambda^2}. \tag{4.11}$$

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

$$\mathbb{P}(\hat{\mathbf{x}}_d^{(k)} \leqslant 0) \leqslant \mathbb{E}[e^{\lambda\hat{\mathbf{x}}_d^{(k)}}] \leqslant e^{-lm_k^2/(2\hat{l}\tilde{\sigma}_k^2)}. \tag{4.12}$$

$$\begin{aligned}\mathbb{P}(\cup_{d=2}^{D}\{\hat{\mathbf{x}}_d^{(k)} \leqslant 0\}) &\leqslant \sum_{d=2}^{D}\mathbb{P}(\hat{\mathbf{x}}_d^{(k)} \leqslant 0)\\ &\leqslant (D-1)e^{-lm_k^2/(2\hat{l}\tilde{\sigma}_k^2)}. \qquad (4.13)\end{aligned}$$

Since $m_k m_{k-1}/(\tilde{\sigma}_k^2) \leqslant 1/(3\hat{r})$, we can check $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$. This finalizes the Proof of the *Theorem 4.1*.

4.4.5 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 $\mathbf{y}_\mathbf{p}$ are stated as

$$\begin{aligned}\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k)}}] &= \Bigg(\mathbb{E}_\mathbf{p}\Bigg[\mathbf{p}^{+}\mathbb{E}\Bigg[e^{\lambda\mathbf{y}_p^{(k-1)}}|\mathbf{p}\Bigg] + \mathbf{p}^{-}\mathbb{E}\Bigg[e^{-\lambda\mathbf{y}_p^{(k-1)}}|\mathbf{p}\Bigg]\\ &\quad +(1-\mathbf{p}^{+}-\mathbf{p}^{-})\Bigg]\Bigg)^{\hat{l}}, \qquad (4.14)\end{aligned}$$

$$\begin{aligned}\mathbb{E}[e^{\lambda\mathbf{y}_p^{(k)}}] &= \Bigg(p\cdot\mathbb{E}\Bigg[e^{\lambda(\frac{D-K}{D})\tilde{\mathbf{x}}_d^{(k)}}\Bigg]\\ &\quad +(1-p)\cdot\mathbb{E}\Bigg[e^{-\lambda(\frac{K}{D})\tilde{\mathbf{x}}_d^{(k)}}\Bigg]\Bigg)^{\hat{r}}, \qquad (4.15)\end{aligned}$$

$$\text{where}\quad \mathbf{p}^{+} = p\cdot\frac{D-K}{D-1} \text{ and } \mathbf{p}^{-} = (1-p)\cdot\frac{K}{D-1}.$$

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

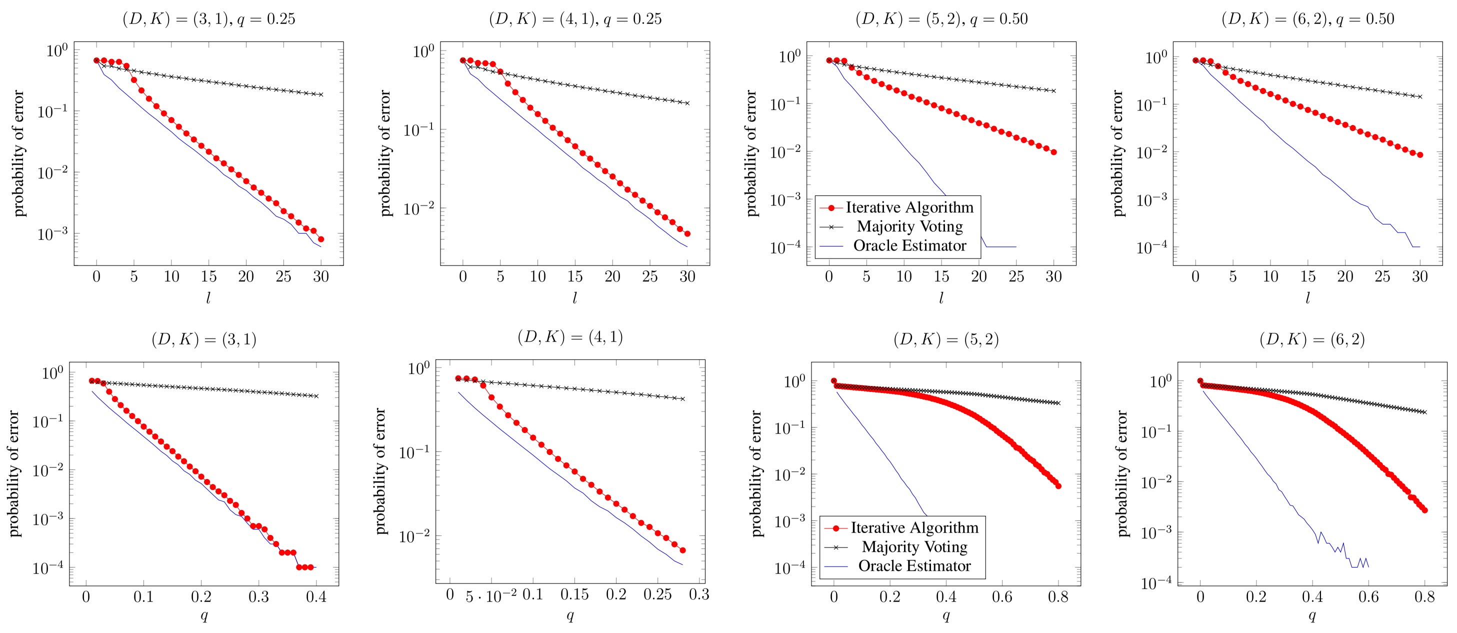


Figure 4.6: Comparisons of probabilities of error between different algorithms varying l (first row) and q (second row) values ($m = n = 2000$, $l = r = 25$).

of p. Substituting this into (4.13), we have

$$\begin{aligned}
\mathbb{E}[e^{\lambda \tilde{\mathbf{x}}_d^{(1)}}] &= \Bigg(\mathbb{E}_{\mathbf{p}}\Bigg[\mathbf{p}^{+} e^{\lambda+(1/2)\lambda^2} + \mathbf{p}^{-} e^{-\lambda+(1/2)\lambda^2} \\
&\qquad + (1-\mathbf{p}^{+}-\mathbf{p}^{-})\Bigg]\Bigg)^{\hat{l}} \\
&\leqslant \Big(\mathbb{E}[a]e^{\lambda} + \Big(\mathbb{E}[\bar{a}]e^{-\lambda}\Big)^{\hat{l}} e^{(1/2)\hat{l}\lambda^2} \\
&\leqslant e^{(\mu\lambda+\lambda^2)\hat{l}}, \qquad (4.16)
\end{aligned}$$

$$\text{where} \quad a = \frac{1+\mathbf{p}^{+}-\mathbf{p}^{-}}{2}, \quad \bar{a} = 1-a = \frac{1-\mathbf{p}^{+}+\mathbf{p}^{-}}{2}.$$

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

$$be^{z} + (1-b)e^{-z} \leqslant e^{(2b-1)z+(1/2)z^2}, \qquad (4.17)$$

for any $z \in \mathbb{R}$ and $b \in [0,1]$ (cf. *Alon and Spencer* 2008, *Lemma A.1.5*) [44].

From k^{th} inductive hypothesis, we can assume that $\mathbb{E}[e^{\lambda \tilde{\mathbf{x}}_d^{(k)}}] \leqslant e^{m_k\lambda+(1/2)\tilde{\sigma}_k^2\lambda^2}$ 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 $|\lambda| \leqslant 1/(2m_k\hat{r})$.

Lemma 4.3. *For any* $|\lambda| \leqslant 1/(2m_k\hat{r})$ *and* $p \in [0,1]$, *we get*

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

Similar to (4.16)'s process, from (4.14), we get

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

with $2 \leqslant e^{\lambda} + e^{-\lambda}$ for any $\lambda \in \mathbb{R}$.

Substituting the result of *Lemma 4.3* into the above inequality provides

$$\begin{aligned}\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k+1)}}] \;\leqslant\; & \mathbb{E}_{\mathbf{p}}\Bigg[a\Big(\mathbf{p}e^{(1/2)m_k\lambda} + \bar{\mathbf{p}}e^{-(1/2)m_k\lambda}\Big)^{\hat{r}} \\ & +\bar{a}\Big(\mathbf{p}e^{(1/2)m_k\lambda} + \bar{\mathbf{p}}e^{-(1/2)m_k\lambda}\Big)^{\hat{r}}\Bigg]^{\hat{l}} \\ & \cdot\, e^{(\frac{D-2K}{2D})\hat{l}\hat{r}m_k\lambda + \frac{1}{2}(\frac{D-K}{D})^2\hat{l}\hat{r}\tilde{\sigma}_k^2\lambda^2}. \end{aligned} \tag{4.18}$$

Now we are left to bound (4.18) using following *Lemma 4.4*.

Lemma 4.4. *For any* $|z| \leqslant 1/(2\hat{r})$ *and* $p \in [0,1]$, *we get*

$$\begin{aligned}\mathbb{E}_{\mathbf{p}}\Bigg[a\Big(\mathbf{p}e^{\frac{1}{2}z} + \bar{\mathbf{p}}e^{-\frac{1}{2}z}\Big)^{\hat{r}} + \bar{a}\Big(\mathbf{p}e^{-\frac{1}{2}z} + \bar{\mathbf{p}}e^{\frac{1}{2}z}\Big)^{\hat{r}}\Bigg]^{\hat{l}} \cdot\; e^{(\frac{D-2K}{2D})\hat{l}\hat{r}z} \\ \leqslant e^{\frac{4}{5}(\frac{D-1}{D})q\hat{l}\hat{r}z + \left[\frac{2}{5}(\frac{D-1}{D})^2q\hat{r}+\frac{1}{8}\right]\hat{l}\hat{r}z^2}.\end{aligned}$$

Applying this to (4.18) gives

$$\mathbb{E}[e^{\lambda\tilde{\mathbf{x}}_d^{(k+1)}}] \leqslant e^{(\frac{D-1}{D})q\hat{l}\hat{r}m_k\lambda + \frac{1}{2}\left[\left((\frac{D-1}{D})^2q\hat{r}+\frac{1}{4}\right)m_k^2 + (\frac{D-K}{D})^2\tilde{\sigma}_k^2\right]\hat{l}\hat{r}\lambda^2},$$

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*

$$m_{k+1} = \Big[\frac{4}{5}\Big(\frac{D-1}{D}\Big)q\hat{l}\hat{r}\Big]m_k,$$

$$\tilde{\sigma}_{k+1}^2 = \left[\left\{\frac{2}{5}\Big(\frac{D-1}{D}\Big)^2q\hat{r} + \frac{1}{8}\right\}m_k^2 + \frac{1}{2}\Big(\frac{D-K}{D}\Big)^2\tilde{\sigma}_k^2\right]\hat{l}\hat{r}$$

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

$$m_k = \mu\hat{l}\Big[\frac{4}{5}\Big(\frac{D-1}{D}\Big)q\hat{l}\hat{r}\Big]^{k-1},\ {}^{\forall}k \in \mathbb{N}$$

Substituting m_k into $\tilde{\sigma}_k^2$, we obtain

$$\tilde{\sigma}_k^2 = A\tilde{\sigma}_{k-1}^2 + B\cdot C^{k-2}, \tag{4.19}$$

where

$$A = \frac{1}{2}\Big(\frac{D-K}{D}\Big)^2 \hat{l}\hat{r}, \quad B = \mu^2 \hat{l}^3 \hat{r}\Big[\frac{2}{5}\Big(\frac{D-1}{D}\Big)^2 q\hat{r} + \frac{1}{8}\Big],$$

$$C = \Big[\frac{4}{5}\Big(\frac{D-1}{D}\Big) q\hat{l}\hat{r}\Big]^2.$$

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

$$\tilde{\sigma}_k^2 = \tilde{\sigma}_1^2 A^{k-1} + BC^{k-2}\Big[\frac{1-(A/C)^{k-1}}{1-(A/C)}\Big]. \tag{4.20}$$

This finishes the proof of (4.11).

Proof of *Lemma 4.3*. In the $k+1^{th}$ step of *mathematical induction*, we assume that for any $d \in [2:D]$ with $|\lambda| \leqslant 1/(2m_{k-1}\hat{r})$.

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

In other words, all of $\tilde{\mathbf{x}}_d^{(k)}$ follow *sub-Gaussian* distribution with parameters m_k and $\tilde{\sigma}_k^2$. From (4.15), each component at the right-hand side can be represented as the product of several combinations of $[e^{\lambda \tilde{\mathbf{x}}_d^{(k)}}]$ and the product of variables means a linear combination in the exponential field. We verify that the linear transformation of *sub-Gaussian* random variables follows also *sub-Gaussian* distribution with some parameters. Moreover, these parameters are determined by D, K, mean m_k and variance $\tilde{\sigma}_k^2$ of each *sub-Gaussian* $\tilde{\mathbf{x}}_d^{(k)}$. In (4.15), the first term is bounded as

$$\mathbb{E}\Bigg[e^{\lambda(\frac{D-K}{D})\tilde{\mathbf{x}}_d^{(k)}}\Bigg] \leqslant \Bigg(e^{\frac{1}{2}m_k\lambda}\Bigg) \cdot e^{\frac{D-2K}{2D}m_k\lambda + \frac{1}{2}(\frac{D-K}{D})^2\tilde{\sigma}_k^2\lambda^2},$$

and the second term is bounded as

$$\begin{aligned}\mathbb{E}\Bigg[e^{\lambda(-\frac{K}{D})\tilde{\mathbf{x}}_d^{(k)}}\Bigg] &\leqslant \Bigg(e^{-\frac{1}{2}m_k\lambda}\Bigg) \cdot e^{\frac{D-2K}{2D}m_k\lambda + \frac{1}{2}(-\frac{K}{D})^2\tilde{\sigma}_k^2\lambda^2} \\ &\leqslant \Bigg(e^{-\frac{1}{2}m_k\lambda}\Bigg) \cdot e^{\frac{D-2K}{2D}m_k\lambda + \frac{1}{2}(\frac{D-K}{D})^2\tilde{\sigma}_k^2\lambda^2}.\end{aligned}$$

The second inequality holds since the case $D \geqslant 2K$ is natural for K-approval setting, otherwise the quality of response is too worthless to inference the ground truth accurately. Getting these two results together finishes the proof of *Lemma 4.3*.

Proof of *Lemma 4.4*. From (4.17), we get

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

Applying this result to the original formula and we have

$$\begin{aligned}
&\mathbb{E}_{\mathbf{p}}\Big[a\Big(\mathbf{p}e^{\frac{1}{2}z} + \bar{\mathbf{p}}e^{-\frac{1}{2}z}\Big)^{\hat{r}} + \bar{a}\Big(\mathbf{p}e^{-\frac{1}{2}z} + \bar{\mathbf{p}}e^{\frac{1}{2}z}\Big)^{\hat{r}}\Big]^{\hat{l}} \cdot\ e^{(\frac{D-2K}{2D})\hat{l}\hat{r}z} \\
&\leqslant \mathbb{E}\Big[e^{(\frac{D}{D-1})(\mathbf{p}-\frac{K}{D})^2\hat{r}z+\frac{1}{2}(\mathbf{p}-\frac{K}{D})^2\hat{r}^2z^2}\Big] \cdot e^{\frac{1}{8}\hat{r}z^2}.
\end{aligned}$$

In this point, we bring (4.4), the definition of worker quality and the fact that $e^b \leqslant 1 + 0.764 * b + b^2$ for $|b| \leqslant 5/8$. We get

$$\begin{aligned}
&\mathbb{E}\Big[e^{(\frac{D}{D-1})(\mathbf{p}-\frac{K}{D})^2\hat{r}z+\frac{1}{2}(\mathbf{p}-\frac{K}{D})^2\hat{r}^2z^2}\Big] \\
&\leqslant \mathbb{E}\Big[1 + 0.764\Big\{\Big(\frac{D-1}{D}\Big)\hat{r}z + \frac{1}{2}\Big(\frac{D-1}{D}\Big)^2\hat{r}^2z^2\Big\}f(\mathbf{p}) \\
&\qquad + \Big\{\Big(\frac{D-1}{D}\Big)\hat{r}z + \frac{1}{2}\Big(\frac{D-1}{D}\Big)^2\hat{r}^2z^2\Big\}^2 f(\mathbf{p})^2\Big] \\
&\leqslant 1 + \frac{4}{5}\Big[\Big(\frac{D-1}{D}\Big)q\hat{r}z + \frac{1}{2}\Big(\frac{D-1}{D}\Big)^2 q\hat{r}^2z^2\Big] \\
&\leqslant e^{\frac{4}{5}\left[\left(\frac{D-1}{D}\right)q\hat{r}z+\frac{1}{2}\left(\frac{D-1}{D}\right)^2q\hat{r}^2z^2\right]},
\end{aligned}$$

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

4.4.6 Phase Transition

As shown in (4.5), the performance of our algorithm is only bounded when the condition $T > 1$ is satisfied. Meanwhile, with $T < 1$, $\tilde{\sigma}_k^2$ which means the variance of the $\tilde{\mathbf{x}}_d^{(k)}$ 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 majority voting. Note that except for the extreme case such as when using very low quality workers and the deficient assignments, $T > 1$ is easily satisfied and our performance guarantee becomes valid. In *Section 4.5*, we will verify the existence of this critical point at $T = 1$ through several experiments with different conditions.

$$T \equiv \frac{C}{A} = \frac{32}{25}\Big(\frac{D-1}{D-K}\Big)^2 q^2 \hat{l}\hat{r}.$$

4.5 Experimental Results

This section verify the performance of the *K-approval iterative algorithm* discussed in the previous *Section 4.3* with different sets of simulations. First, we check the performance on the average error of our algorithm exhibits exponential decay as degree of worker node l or quality of workers q increases. In addition, we show that our algorithm achieves a better performance than that of the majority voting above a phase transition of $T = 1$. Next simulation investigates the linear relationship between y_j value and the ratio of the number of responses including the correct answer to r_j for each worker. Then, we consult experiments with a task set consisting of various (D, K) pairs which mean the number of alternatives D and selection K.

4.5.1 Performance on the Average Error with q and l

To show the competitiveness of our algorithm, we ran our *K-approval iterative algorithm*, majority voting, and the oracle estimator for 2000 tasks and 2000 workers with a fixed (D, K) pair in Figure 4.6. Here, the oracle estimator can give each worker appropriate weight since we assume the oracle estimator knows quality of every worker. The performance on the average error of the oracle estimator is presented as a lower bound. In Figure 4.6 (top), we can check that the probability of error decays exponentially as l increases, and is lower than that of the majority voting above the phase transition point $T = 1$. In addition, Figure 4.6 (bottom) presents the probabilities of error decays as q increases.

In *Section 4.4*, we expect a phase transition at $T = \frac{32}{25}\big(\frac{D-1}{D-K}\big)^2 q^2 \hat{l}\hat{r} = 1$ or $l = \frac{4\sqrt{2}}{5}\frac{(D-K)}{(D-1)} \cdot \frac{1}{q}$. If we follow the result, we can expect transitions to happen around

$l = 4.52$ for $(3,1),(4,1)$, $l = 3.39$ for $(5,2)$ and $l = 3.62$ for $(6,2)$. From the experiments in Figure 4.6 (top), we find that iterative algorithm starts to perform better than majority voting around $l = 5,\ 6,\ 3,\ 4$ for each (D,K) pair. Note that these values are very similar with the above theoretical values. It follows from the fact the error of our method increases with k when $T < 1$ as stated in *Section 4.4*. As can clearly be seen from the simulation results, we can be sure that the l values required for achieving $T > 1$ are not large.

4.5.2 Relationship between Reliability and y-message.

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

$$\hat{p_j} = \frac{\#\, of\, responses\, including\, correct\, answer}{r_j}.$$

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 4.1.

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

Figure 4.7 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 0.991, 0.992, 0.904, 0.954 for each $(D,K) = (3,1),(4,1),(5,2)$, and $(6,2)$, We can also check that the line passes approximately the point of $(\frac{K}{D}, 0)$, where $p_j = \frac{K}{D}$ is close to the reliability of non-informative worker who gives random responses, as expected in *Section 4.4*.

4.5.3 Performance on the Average Error with Various (D,K) Pairs.

To show the performance of the *K-approval iterative algorithm*, we do simulations on a task set consisting of various (D,K) pairs with Algorithm 4.1. In detail, we repeat

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

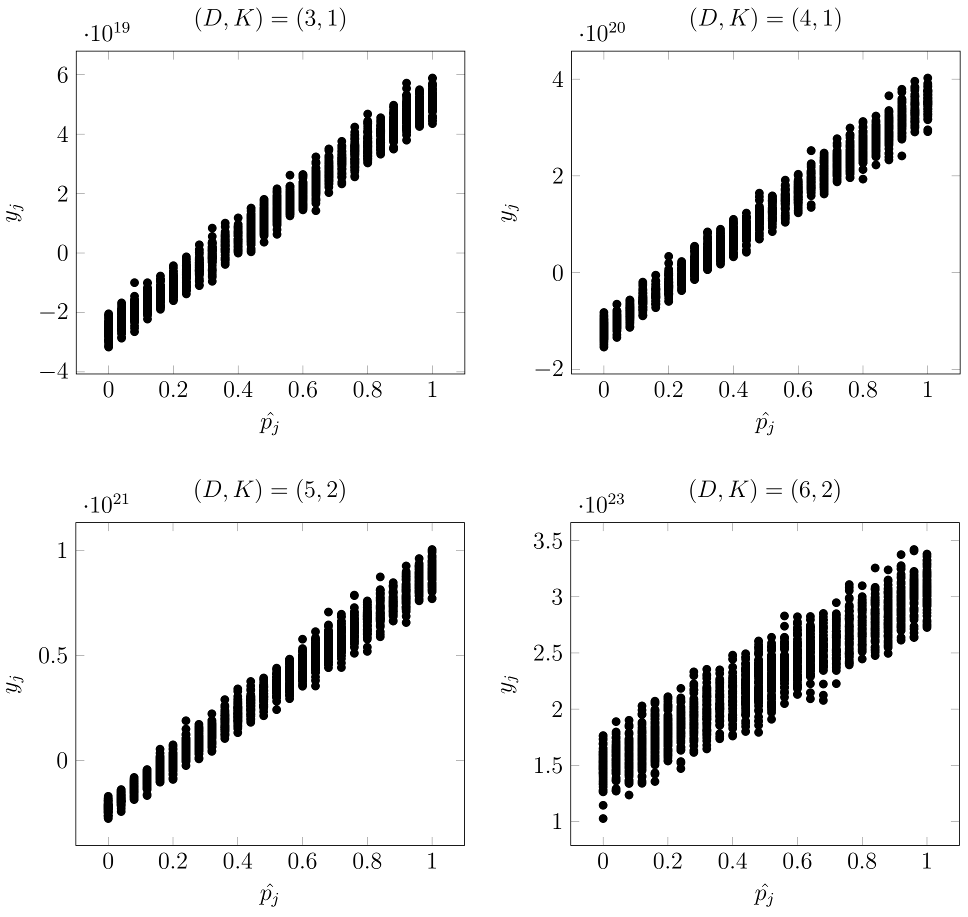


Figure 4.7: Relationship between y_j and $\hat{p_j}$ ($m = n =$ 2000, $k =$ 10).

the same experiments with a question set composed in equal ratio of each task type whose (D, K) are $(3, 1)$, $(4, 1)$, $(5, 2)$ and $(6, 2)$ respectively. Then, we have to investigate for the general case that q_j is calculated with general version of equation (4.4) in *Section 4.4.2* as follows:

$$q_j(p_j) = Q(p_j) - Q\Big(\frac{K_i}{D_i}\Big), \tag{4.21}$$

$$\text{where } Q(p) = p \log\Big[\frac{p}{{}_{D_i-1}C_{K_i-1}}\Big] + (1-p)\log\Big[\frac{1-p}{{}_{D_i-1}C_{K_i}}\Big].$$

We define q_j as an individual inherent quality of the worker j. To perform simulations and to analyze the results, we have to make an assumption that a worker with an

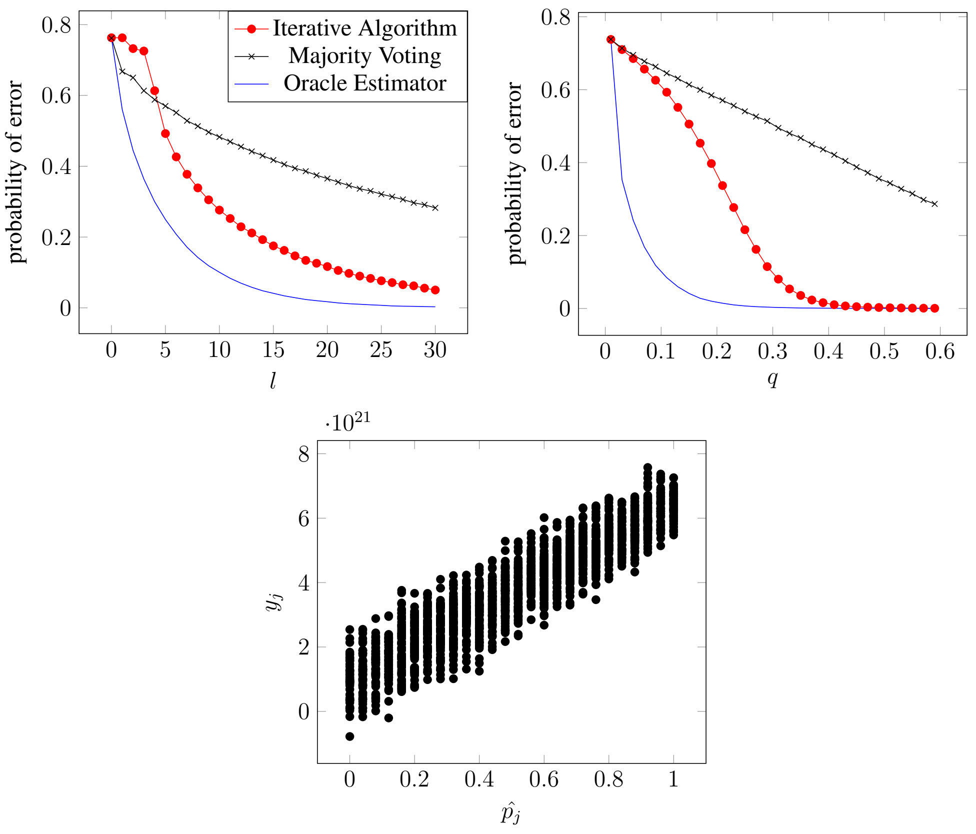


Figure 4.8: Simulations on a set of various (D, K) pairs ($m = n = 2000$, mixed task types : $(D, K) = (3, 1), (4, 1), (5, 2), (6, 2)$) in an equal ratio.

individual quality q_j solves question with a reliability p_{ij} for each (D_i, K_i). Thus for each (D_i, K_i)-task, the corresponding reliability p_{ij} is determined by applying Newton's method on a graph of the quality function.

We can check that the same tendencies found in previous simulations also appear in Figure 4.8. There is also the strong correlation between y_j and $\hat{p_j}$ as 0.929. This result is notable in that in the real world, there are many more cases where questions have various (D, K) pairs than fixed ones.

4.6 Conclusion

We have proposed an iterative algorithm that can handle top-K selection questions which are general types of questions in real crowdsourcing systems. Moreover, our scheme covers the case with a task set consisting of various (D, K) pairs. From the performance analysis of our algorithm, we rigorously proved that an upper bound on the probability of error decays exponentially. Through numerical experiments, we have verified that our algorithm outperforms majority voting and come close to an oracle estimator who knows quality of every worker.

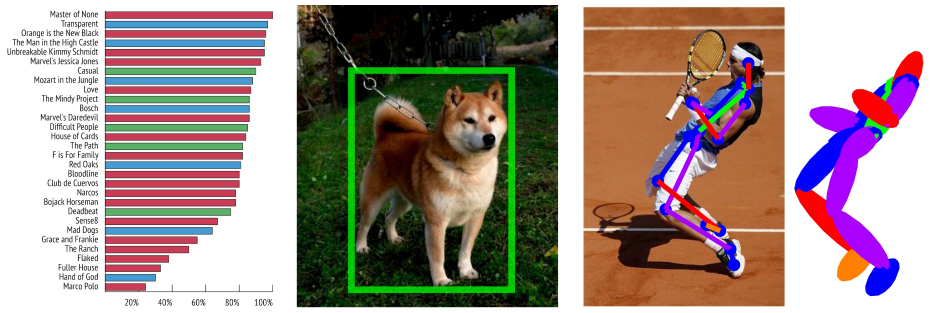
Chapter 5

Crowdsourcing Systems for Real-valued Vector Regression

Over the years, several papers have proposed aggregation methods and verified theoretical bounds for binary-choice tasks [34, 36, 55] and discrete multiple-choice tasks [38, 32, 50]. However, most of recent crowdsourcing tasks ask workers to solve a problem with vectors. Actually, in web-based crowdsourcing platforms such as Amazon Mechanical Turk and CrowdFlower, a considerable number of requesters ask workers to solve vector regression tasks. (ex Monthly statistics for June 2019, about 22%) As described in Figure 5.1, the examples of vector regression tasks are as follow: (1) Rating movies or items, (2) Finding the location of an object in an image, and (3) Estimating a human posture in an image.

There have been studies to devise an inference algorithm for regression tasks. [30] extended their binary classification model to learn a simple linear regressor. As for Expectation Maximization (EM) methods, [56] and [48] proposed a probabilistic graphical model for image object localization. However, those models have a difficulty in learning parameters with relatively small number of responses.

In this paper, we propose an iterative algorithm for inferring true answers from noisy responses in vector regression tasks. As in many previous works[56, 32, 48, 34],



(a) Movie rating (b) Object localization (c) Pose estimation

Figure 5.1: Applications of the regression tasks in crowdsourcing. (a) movie rating : to score movies from 0 to 100. (b) image object localization: to draw a tight bounding box capturing the target object. (c) pose estimation: to find the proper positions of the skeleton's joints.

we also consider the "*reliability*" of a worker represented by a parameter indicating the worker's expertise level and ability. Our algorithm computes two types of messages alternately. First, the worker message estimates the reliability of each worker, and the task message computes the weighted averages of their responses using those reliabilities as weights. These processes contribute to infer more accurate answers by sorting the order of responses by importance. Then we prove the error bound of our algorithm's average performance based on a probabilistic crowd model. This result shows our algorithm achieves better performance than other existing algorithms with a small number of queries and comparatively low average quality of the crowd. Furthermore, we provide that under a certain condition, the ℓ_2 error performance of ours is close to that of an oracle estimator which knows the reliability of every worker. Through extensive experiments, we empirically verify that our algorithm outperforms other existing algorithms for both real world datasets crowd-sourced from *CrowdFlower*, and synthetic datasets.

Source	**Binary**	**Multi-class**	**Regression**
Dawid and Skene[38]	✓	✓	
Whitehill et al. [32]	✓		
Welinder et al. [56]	✓		✓
Raykar et al. [30]	✓	✓	✓
Karger et al. [34]	✓		
Liu et al. [36]	✓		
Dalvi et al. [55]	✓		
Salek et al. [48]			✓
Karger et al. [37]	✓	✓	
Zhang et al. [47]	✓	✓	
Lee et al. [50]	✓	✓	

Table 5.1: Comparisons of the types of tasks covered by well-known crowdsourcing algorithms

5.1 Related Work

For aggregation methods, majority voting is a widely used for its simplicity and intuitiveness. [57] shows majority voting can effectively reduce the error in the attribute-based setting. However, it regards every worker as equally reliable and gives an identical weight to all responses. Therefore, the performance of majority voting suffers even with a small number of erroneous responses [29]. To overcome this limitation, there have been several approaches for improving the inference performance from unreliable responses. [38, 56, 32] adopt Expectation and Maximization (EM) to evaluate the implicit characteristics of tasks and workers. Also, [47] improves this EM approach using a spectral method with performance guarantees. However, in practice, there is a difficulty in parameter estimation since these EM approaches are aimed at estimating a huge confusion matrix from relatively few responses.

[34, 36] proposed Belief Propagation(BP)-based iterative algorithms and proved that their error performances are bounded by worker quality and the number of queries in binary-choice tasks. Furthermore, there are several researches for crowdsourcing systems with multiple-choice tasks. [37] focused on multi-class labeling using a spectral method with low rank approximation, [49] proposed an aggregating method with minimax conditional entropy and [58] suggested an aggregation method using a decoding algorithm of coding theory. In addition, [50] exploits a inner product method($\mathcal{IP}$) for evaluating similarity measures between an answer from a worker and the group consensus.

There have been studies to target vector regression tasks: [59] and the DALE model in [48], which focus on finding the location of a bounding box in an image. The former suggests a simple serial task assignment method for a quality-controlled crowdsourcing system with no theoretical guarantee. The latter proposes a probabilistic graphical model for image object localization and inference method with expectation propagation. However, the worker model assumption in these papers has two limitations; it strictly divides the workers' expertise level and ignores the order of selection when a crowd divides a length into multiple segments. Also, the latter graphical model has too many parameters to learn from relatively small number of responses.

On the other hand, there are *outlier rejection* methods that can be used to filter unreliable responses without a graphical model. For non-parametric setting, *mean shift* and *top-k selection* are typically used as classical methods. *mean shift* is the technique for locating the maxima of a density function and *top-k selection* picks k most reliable responses based on distances between the mean vector and each response itself. For parametric setting, *RANSAC*(random sample consensus) is widely used. it is an iterative method to estimate parameters of a mathematical model from a set of responses that contains outliers, when they are to be accorded no influence on the values of the estimates.

While most of the papers mentioned above assume random regular task assignments, [55, 60] proposed inference methods in irregular task assignments. Also, [37, 50, 61] suggested the adaptive task assignment which gives more tasks to more reliable workers in order to infer more accurate answers given a limited budget.

5.2 Problem Setup

In this section, we describe a problem setup with variables and notations. First, we assume that there are m tasks in total and each task i is assigned to distinct l_i workers. Similarly, there are n workers in total and each worker j solves different r_j tasks. Here and after, we use $[N]$ to denote the set of first N integers. If we regard tasks and workers as set of vertices and connect the edge $(i, j) \in E$ when the task i is assigned to the worker j, our system can be described as a bipartite graph $G = \{[m], [n], E\}$ in Figure 5.2.

Our crowdsourcing system considers a specific type of task whose answer space spans a finite continuous domain. If a task asks D number of real values, a response $\tilde{\boldsymbol{A}}$ is a D-dimensional vector. On one task node i, given all of responses $\big\{\tilde{\boldsymbol{A}}_{ij} | (i, j) \in E\big\}$, we transform them to $\boldsymbol{A}$ subject to $\|\boldsymbol{A}_{ij}\|_1 = 1$ by the min-max normalization since each task can have a different domain length.

For a simple example, in an image object localization regression task, a response is a bounding box to capture the target object. Considering the x axis only for brevity, the box coordinate is $\tilde{\boldsymbol{A}} = [x_{tl}, x_{br}]$, where x_{tl} and x_{br} stand for the top-left and bottom-right coordinates. Then it can be transformed as

$$\boldsymbol{A} = \big(x_{tl}, x_{br} - x_{tl}, x_{max} - x_{br}\big)/x_{max},$$

where x_{max} represents the width of the image. Since images have different size of width and height, all responses are transformed to have the same domain length.

In summary, when the worker j solves the task i, the response is denoted as $\tilde{\boldsymbol{A}}_{ij} \in \mathbb{R}^D$ and transformed to $\boldsymbol{A}_{ij} \in \mathbb{R}^{D+1}$ with respect to $\|\boldsymbol{A}_{ij}\|_1 = 1$. For convenience, δ_i and δ_j denotes the group of workers who give responses to the task i and

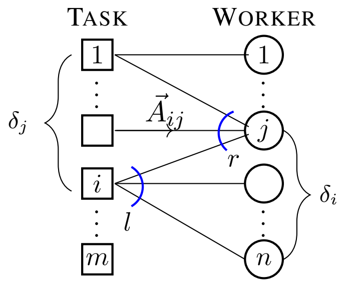


Figure 5.2: System model for task-worker assignments.

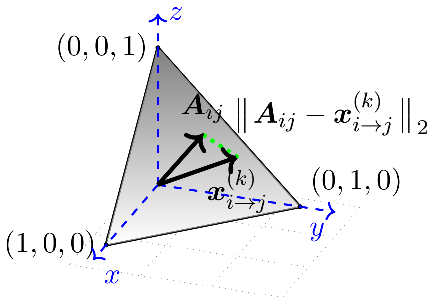

z
$(0,0,1)$
$\boldsymbol{A}_{ij}$
$\| \boldsymbol{A}_{ij} - \boldsymbol{x}^{(k)}_{i\to j} \|_2$
$\boldsymbol{x}^{(k)}_{i\to j}$
$(0,1,0)$
y
$(1,0,0)$
x

Figure 5.3: Distance between answer $\boldsymbol{A}_{ij}$ and x message $\boldsymbol{x}_{i\to j}$ in the standard 2-dimensional simplex space when $D_i = 2$.

the group of tasks which are assigned to worker j respectively.

Majority Voting ($\mathcal{MV}$). The simplest method in response aggregation is majority voting, well-known sub-optimal estimator, which computes the centroid of responses. However, its performance can be easily degraded whether there exist a few adversarial workers or spammers who give wrong answers intentionally or random answers respectively.

Majority voting method gives the identical weight to every worker who annotates the task for fixed task i.

$$\hat{\boldsymbol{t}}_i^{(\mathcal{MV})} = \sum_{j\in\delta_i} \frac{1}{l_i} \boldsymbol{A}_{ij}. \tag{5.1}$$

5.3 Inference Algorithm

In this section, we propose a message-passing algorithm for vector regression tasks. Our iterative algorithm alternatively estimates two types of messages: (1) task messages $\boldsymbol{x}_{i\to j}$, and worker messages $\boldsymbol{y}_{j\to i}$. This updating process estimates the ground truth of each task and the reliability of each worker respectively. From now on, $\hat{l}_i$ and

$\hat{r_j}$ denote $(l_i - 1)$ and $(r_j - 1)$ respectively for brevity.

5.3.1 Task Message

We first describe a task message that estimates the current candidate of a ground truth. It simply computes the centroid of weighted responses from the workers assigned to the task. Thus, it can be viewed as a simple estimator of weighted voting in that those weights are computed according to how workers are reliable. Note that a task message $\boldsymbol{x}_{i\to j}$ averages weighted responses from workers assigned to a task i except for the response from worker j. This helps to block any correlation between the task message and the responses from worker j.

$$\boldsymbol{x}_{i\to j}^{(k)} = \sum_{j'\in\delta_i\backslash j} \left(\frac{y_{j'\to i}^{(k-1)}}{y_{\delta_i\backslash j}^{(k-1)}} \right) \boldsymbol{A}_{ij'}, \tag{5.2}$$

where $y_{\delta_i\backslash j}^{(k-1)} = \sum_{j'\in\delta_i\backslash j} y_{j'\to i}^{(k-1)}$.

Algorithm 5.1 Inference Algorithm

Input: $G = \{[m], [n], E\}$, $\{\boldsymbol{A}_{ij}\}_{(i,j)\in E}$, k_{max}

Output: Estimated truths $\hat{\boldsymbol{t}}_i$, $^\forall i \in [m]$

```
 1: Initialization
 2: for $^\forall(i,j) \in E$ do
 3:         $y^{(0)}_{j\to i} \leftarrow \mathcal{N}(0,1)$
 4: end for
 5: Iteration Step
 6: for $k = 1$ to $k_{max}$ do
 7:     for $^\forall(i,j) \in E$ do
 8:         Update task message, $\boldsymbol{x}^{(k)}_{i\to j}$ using Eq. 5.2
 9:     end for
10:     for $^\forall(i,j) \in E$ do
11:         Update worker message, $y^{(k)}_{j\to i}$ using Eq. 5.3
12:     end for
13: end for
14: Final Estimation
15: for $^\forall j \in [n]$ do
16:         $y_j \leftarrow \left(\frac{1}{\hat{r_j}}\sum_{i\in\delta_j}(\|\boldsymbol{A}_{ij} - \boldsymbol{x}^{(k_{max})}_{i\to j}\|_2)^2\right)^{-1}$
17: end for
18: for $^\forall i \in [m]$ do
19:         $\boldsymbol{x}_i \leftarrow \sum_{j\in\delta_i}\left(\frac{y_j}{y_{\delta_i}}\right)\boldsymbol{A}_{ij}$
20: end for
21: return $\hat{\boldsymbol{t}}^{(\mathcal{ALG})}_i \leftarrow \boldsymbol{x}_i$, $^\forall i \in [m]$
```

5.3.2 Worker Message

The next step is to compute worker messages $y_{j\to i}$ which represents the importance of response $\boldsymbol{A}_{ij}$. These worker messages are used as weights in the weighted voting process in task messages update. Since it is desirable to give a higher weight to more reliable workers, each worker's reliability should be evaluated as the similarity between his response and the task message which indicates the consensus of other workers' responses. In our algorithms, it takes advantage of the reciprocal of the summation of the euclidean distance between the response and the task message as a similarity measure. In analysis section, our analysis verify that this measure is proper to estimate weights of workers' responses. Note that a worker message $y_{j\to i}$ represents the average of similarities between worker j's responses and the average response of other workers' responses in the same task.

$$y^{(k)}_{j\to i} = \left(\frac{1}{\hat{r_j}} \sum_{i' \in \delta_j \setminus i} \left(\|\boldsymbol{A}_{i'j} - \boldsymbol{x}^{(k)}_{i'\to j}\|_2)^2 \right) \right)^{-1}. \tag{5.3}$$

In the worker message update (5.3), we adopt the reciprocal of ℓ_2 norm in the vector space as a similarity measure. However, our algorithm can be generalized with any metric induced by other norm and similarity function which is continuous and monotonically decreasing.

5.4 Analysis of Algorithms

In this section, We first propose a general worker model for real-valued segmentation tasks. Our model follows the *Dirichlet* distribution to describe how reliable each worker is. Under this model, we proceed with theoretical analysis on the performance of our algorithm. As a main result, in *Theorem 5.2*, we will prove a tight error bound which depends on task degree l and average quality q of workers. Then, we compare ℓ_2 error performance of our algorithm with those of the baseline algorithms.

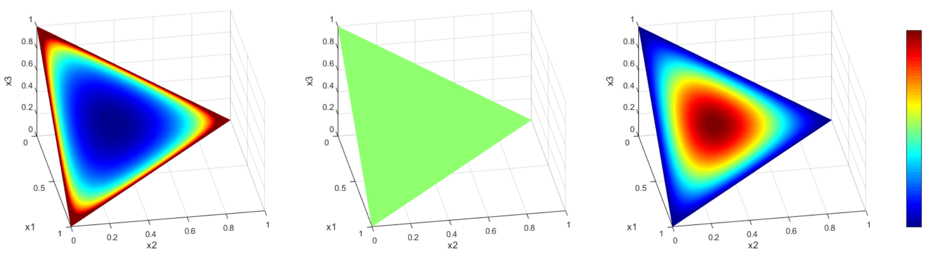
5.4.1 Worker Model

Here we propose a worker model that is well-suited for our crowdsourcing system. We adopt Dirichlet distribution rather than multivariate Gaussian one. The former is more appropriate to capture worker's behaviors in a restricted simplex domain, while the latter cannot represent *Spammer*'s behavior since the latter spans infinite continuous domain. We first review the Dirichlet distribution and its parameters, and then propose a probabilistic worker model.

Dirichlet distribution The Dirichlet distribution is a well-known multivariate generalization of *beta* distribution and is often used as a conjugate prior to the multinomial distribution. A Dirichlet distribution on the standard D-dimensional simplex is parametrized by a $(D+1)$-dimensional positive real valued vector $\boldsymbol{\alpha}$. Its probability density function is given by

$$f(x_1, ..., x_{D+1}; \alpha_1, ..., \alpha_{D+1}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{d=1}^{D+1} {x_d}^{(\alpha_d - 1)}, \tag{5.4}$$

on the standard D-dimensional simplex which is defined by $\boldsymbol{x} \in \mathbb{R}^{(D+1)}$, where $x_d > 0$ for $^\forall d$ and $\|\boldsymbol{x}\|_1 = 1$. The parameter vector $\boldsymbol{\alpha}$ can be decomposed into the mean vector $\boldsymbol{s}$ with a scalar α_0 called a precision (or concentration parameter), i.e. $\boldsymbol{\alpha} = \alpha_0 \boldsymbol{s}$, where $\alpha_0 = \sum \alpha_d$. The precision parameter generally controls the sparsity of samples from the Dirichlet distribution and it is inversely proportional to its variance.



(a) Adversarial (b) Spammer (c) Hammer

Figure 5.4: Three types of crowds in the standard 2-dimensional simplex space.

Note that when every $\alpha_i > 1$, the probability density function of a Dirichlet distribution attains unimodality and it has a bell shape around its mean. In other condition when any $\alpha_i < 1$, it has no mode and most of its density is concentrated on a region outward from its mean.

Dirichlet crowd model For given ground truths on the standard D-dimensional simplex, we can consider each response as a sample from a Dirichlet distribution governed by corresponding task and worker. Since responses are commonly drawn around the ground truth, we can model the mean of this Dirichlet distribution as the task's ground truth t_i. The expected error between the ground truth and the response then comes from the variance, which is determined by the location of ground truth $\boldsymbol{t_i}$ and the worker's reliability w_j. ($\boldsymbol{\alpha} = w_j \boldsymbol{t_i}$) The expected error in responses is determined by the worker reliability which measures the ability, diligence, and precision of the worker. We parametrize this as a positive continuous value $w_j \in [0, \infty]$ where a higher w_j means that worker j gives a closer answer to ground truth. As Figure 5.4 illustrates, three types of worker which are adversarial worker, Spammer and Hammer [62] can be well-described according to precision w.

Here we assume that workers follow the Dirichlet crowd model. In addition, the task assignment is denoted as a random (l, r)-regular bipartite graph $G([m], [n], \boldsymbol{A})$. Then we define expected error from the individual worker j's response in task i by the distance between the answer $\boldsymbol{A}_{ij}$ and the ground truth $\boldsymbol{t}_i$. If we use the square of ℓ_2 norm as an error measure, then the individual error is represented as

$$e_{ij} = (\|\boldsymbol{A}_{ij} - \boldsymbol{t}_i\|_2)^2. \tag{5.5}$$

Due to each answer A_{ij} follows Dirichlet crowd model, the expected error is represented as follows

$$\mathbb{E}[e_{ij}] = \Big(\frac{1}{w_j + 1}\Big) \cdot T_i, \qquad T_i = \boldsymbol{t}_i \cdot (\boldsymbol{1} - \boldsymbol{t}_i). \tag{5.6}$$

The error depend on $(w_j + 1)^{-1}$ and T_i which represent worker j's precision and the effect of ground truth respectively. Now we define the average quality of workers as

$$q^{-1} = \mathbb{E}_{\mathbb{W}}\Big[\frac{1}{\mathrm{w}+1}\Big]. \tag{5.7}$$

A higher value of q indicates that workers are more reliable, whereas a value q close to one means many workers behave maliciously. We will show that the error bound of our algorithm depends on this parameter q. The expected error of our algorithm is defined as

$$\mathrm{E}_{\mathcal{ALG}} := \lim_{k_{\max}\to\infty} \frac{1}{m} \sum_{i\in[m]} \mathbb{E}\Big[\Big(\|\boldsymbol{t}_i - \hat{\boldsymbol{t}}_i\big(\{\boldsymbol{A}_{ij}\}_{(i,j)\in E}\big)\|_2\Big)^2\Big]. \tag{5.8}$$

5.4.2 Oracle Estimator

Under a probabilistic worker model, we can construct the best inference algorithm as obtained by an oracle who knows every worker's reliability. Given every worker's reliability, the oracle can estimate the ground truth by minimizing the expected error between the ground truth and an estimated location. Therefore its performance can be considered as a lower bound of the expected error rate.

$$\hat{t}_i^{(\mathcal{OC})} = \sum_{j\in\delta_i} \Big\{ \frac{w_j + 1}{\sum_{j\in\delta_i}(w_j+1)} \Big\} \boldsymbol{A}_{ij}. \tag{5.9}$$

Secondly, we prove the performance of an oracle estimator which gives the theoretical lower error bound. As described in Section 5.4.2, the oracle estimator knows every worker's reliability. Thus, it gives the optimal weight to each worker as (5.9) and the expected error of oracle estimator can be described as

$$\mathrm{E}_{\mathcal{OC}} = \Bigg(\frac{1}{\mathbb{E}_{\mathbb{W}}\big[\mathrm{w}+1\big]}\Bigg) \cdot \frac{1}{lm} \sum_{i\in[m]} T_i. \tag{5.10}$$

Then, we look into the expected errors of baseline algorithms. As mentioned above, majority voting gives the same weight for each worker. The expected error of majority voting is simply represented as

$$\mathrm{E}_{\mathcal{MV}} = \Big(\frac{1}{qlm}\Big) \sum_{i\in[m]} T_i. \tag{5.11}$$

From above results, we always have

$$\mathrm{E}_{\mathcal{OC}} \leqslant \mathrm{E}_{\mathcal{MV}} \quad (\because \textit{Jensen's inequality}), \tag{5.12}$$

and the equality holds when the distribution of worker reliability $\mathbb{W}$ follows a degenerate distribution.

Optimality of Oracle Estimator. In this part, we prove why the oracle estimator is optimal and it gives worker j the weight in proportion to $(w_j + 1)$. The result of (6) implies that the expected error of oracle estimator is weighted average of $(w_j + 1)^{-1}$ excluding the effect of the ground truth. For fixed task i and given l number of worker batch δ_i, the problem of finding the oracle estimator's optimal weights is formulated as the following convex optimization problem. The object function $G(\boldsymbol{v})$ represents the sum of expected errors from workers in the batch δ_i.

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & G(\boldsymbol{v}) = \sum_{j=1}^{l} \frac{v_j^2}{w_j + 1} \\ \text{subject to} \quad & \sum_{j=1}^{l} v_j = \boldsymbol{v}^T \mathbf{1} = 1. \quad v_j \geqslant 0^{\forall} j \end{aligned}$$

This problem is equivalent to minimize a l-dimensional ellipsoid's radius with a constraint on the subspace $\boldsymbol{v}^T\mathbf{1} = 1$. The optimal point $\boldsymbol{v}^*$ is on the point of contact between the l-dimensional ellipsoid and the subspace. Thus, we can obtain $\nabla G(\boldsymbol{v}^*) = b\mathbf{1}$ for some constant b and $(\boldsymbol{v}^*)^T\mathbf{1} = 1$. In addition, the problem is also convex problem with strong duality (zero dual gap). Therefore, points are optimal if and only if they satisfy the KKT condition:

$$\boldsymbol{v}_j^* \geqslant 0, \ {}^{\forall}j, \ \boldsymbol{v}^T\mathbf{1} = 1,$$

$$\psi_j \geqslant 0, \ \psi_j v_j = 0 \ {}^{\forall}j,$$

$$\nabla H(\boldsymbol{v}) = 0.$$

Once you solve the above equations, we obtain the optimal weight which the oracle estimator gives to worker j. In fact, the optimal weight of the worker j is in proportion to $(w_j + 1)$.

$$\boldsymbol{v}_j^* = \frac{(w_j + 1)}{\sum_{j=1}^{l}(w_j + 1)}. \quad \square$$

5.4.3 Bound on the Average Error Probability

Here we state the main analytic result of our algorithm: for a random (l, r)-regular bipartite graph from configuration model, we prove that the average error performance is bounded by the formula which depends on the average quality q, the number of queries l and some constant related to the distribution of ground truth.

Although our algorithm works in a general bipartite graph, here we assume a regular bipartite graph (i.e. $^{\forall}i, j$, $l_i = l$ and $r_j = r$ solely for a mathematical proof which were commonly used in previous works [34, 50]. This is an ordinary setting when we intend to assign our budget to each task equally. If the crowdsourcing system creates every batch arbitrarily, our setting can be represented as a random (l, r)-regular bipartite graph. To generate a random regular bipartite graph, we bring a simple construction model known as the *configuration model* [63, 64].

Our strategy to provide the upper bound is to find the $\mathbb{E}\big[y_{j\to i}^{(\infty)}\big]$ which means the value of worker message as $k_{max} \to \infty$. In task message update step, our algorithm normalizes the worker message values in order to make each task message on the simplex. Since we assume the sparse bipartite graph, the task-worker graph G is locally tree-like. Therefore, the messages and responses are uncorrelated with high probability.

Lemma 5.1. *For each task i, worker j, If the average quality satisfies $q > (1 + (D + 1)/\hat{l}\hat{r})$, then when $k \to \infty$ the expected worker message of our algorithm converges such that*

$$\mathbb{E}\Big[y_{j\to i}^{(\infty)}\Big] \cong \frac{1}{\frac{1}{(w_j+1)} + \frac{1}{\hat{l}\hat{r}q}} \cdot T_i. \tag{5.13}$$

This result implies our algorithm gradually gives worker j a weight proportional to $(w_j + 1)$ like the oracle estimator as degree l or the worker quality q grows. In the other extreme case, the our algorithm gives the constant weight to each worker like the majority voting (i.e., $(w_j + 1) \gg lq$). In the next part, we will prove that the oracle estimator assigns a worker j the optimal weight in proportion to $(w_j + 1)$. Thus, the average error increases as the KL divergence between the vector of weights and that of oracle estimator's weight increases. Since the weights of workers who annotate the task i are normalized by their sum, we can determine those weights precisely in terms of their ratios mentioned in *Lemma 5.1*. Then, we can verify that Algorithm 1 is gradually getting closer to oracle estimator as $k \to \infty$. In initialization, since Algorithm 1 gives same weight to each worker, it behaves as the majority voting. After a few iterations, each weight vector follows (5.13) similar to oracle estimator which is the theoretical lower bound in ℓ_2 norm.

Proof of *Lemma 5.1* The average value of worker message converges proportionally to $\big((w_j + 1)^{-1} + (lq)^{-1}\big)^{-1}$ as $k \to \infty$ in *Lemma 5.1*. For the proof, we describe a distance of responses in advance. Since we adopt the euclidean distance as a metric in Section 2, the distance between the single response $\boldsymbol{A}_{ij}$ and the task message $\boldsymbol{x}^{(k)}_{i\to j}$ is denoted as

$$d^{(k)}_{j\to i} = (\|\boldsymbol{A}_{ij} - \boldsymbol{x}^{(k)}_{i\to j}\|_2)^2.$$

Then, the expected distance is represented as

$$\mathbb{E}\big[d^{(k)}_{j\to i}\big] = \Bigg(\frac{1}{(w_j+1)} + \underbrace{\sum_{j'\in\delta_i\setminus j} \mathbb{E}\Bigg[\Big(\frac{y^{(k-1)}_{j'\to i}}{y^{(k-1)}_{\delta_i\setminus j'}}\Big)^2\Bigg]\frac{1}{(w_{j'}+1)}}_{p^k_{ij}}\Bigg)T_i.$$

Since our algorithm initializes each worker message value equally and we assume average quality satisfies the condition, $q > (1 + D/\hat{l}\hat{r})$, p^k_{ij} value decreases as $k \to \infty$.

In particular,

$$\mathbb{E}\left[\frac{1}{\hat{r}}\sum_{i'\in\delta_j\backslash i} p_{ij}^{k}\right] \leqslant \frac{1}{\hat{l}\hat{r}q}$$

On the other hand, the estimation process of worker messages is described as

$$y_{j\to i}^{(k)} = \frac{1}{\frac{1}{\hat{r}}\sum_{i'\in\delta_j\backslash i}\left(d_{j\to i'}^{(k)}\right)}. \tag{5.14}$$

In section C, we will prove that the oracle estimator gives each worker j the optimal weight in proportion to $(w_j + 1)$. The expected distance and (5.14) implies that our algorithm gives each worker j similar weight but not equal. Thus, we can obtain the expected worker message after k iteration.

$$\mathbb{E}[y_{j\to i}^{(k)}] \cong \frac{1}{\frac{1}{\hat{r}}\sum_{i'\in\delta_j\backslash i}\mathbb{E}[d_{j\to i'}^{(k)}]}. \tag{5.15}$$

Therefore, as $k \to \infty$, the expected worker message satisfies

$$\mathbb{E}\left[y_{j\to i}^{(\infty)}\right] \cong \frac{1}{\frac{1}{(w_j+1)} + \frac{1}{\hat{l}\hat{r}q}} \cdot T_i. \tag{5.16}$$

Theorem 5.2. *For fixed $l > 1$, $r > 1$ and dimension $D \geqslant 1$, assume that m tasks are assigned to n workers according to a random (l, r)-regular bipartite graph. If the average quality satisfies $q > (1 + (D+1)/\hat{l}\hat{r})$, then when $k \to \infty$ the average error of the our algorithm achieves*

$$\mathrm{E}_{\mathcal{ALG}} \leqslant \left(\frac{(1+1/\hat{l}\hat{r})^2}{(\sqrt{2}+1)q\hat{r}}\right) \cdot \frac{1}{\hat{l}m}\sum_{i\in[m]} T_i. \tag{5.17}$$

This result implies that we can control the error performance by adjusting the average quality of workers and the number of queries assigned to each task. As q and lr increase, the upper bound of our algorithm becomes lower.

Proof of *Theorem 5.2* Here the detailed proof of *Theorem 5.2* is provided using the result of *Lemma 5.1*. In Section 2, we adopt a random (l, r)-regular bipartite graph as a task assignment. If this sparse task-worker graph is assumed ($|m| \gg r$, $|n| \gg l$), we can claim that the graph has locally-tree like structure. This property implies that the response A_{ij} is uncorrelated with the task message $x^{(k)}_{i\to j}$. Then, for each task i, the expected error of our algorithm E_{iter} is represented as

$$\mathrm{E}_{\mathcal{ALG}} = \frac{1}{m}\sum_{i\in[m]} \underbrace{\sum_{j\in\delta_i} \mathbb{E}\Big[\Big(\frac{y^{\infty}_{j\to i}}{y^{\infty}_{\delta_i}}\Big)^2\Big] \cdot \frac{1}{w_j+1}}_{s_i} \cdot T_i.$$

In this point, we can obtain the expected worker message using *Lemma 5.1*,

$$\mathbb{E}\Big[y^{(\infty)}_{j\to i}\Big] \cong \frac{(w_j+1)}{1+\frac{(w_j+1)}{lq}} \cdot T_i. \tag{5.18}$$

According to the definition of the worker message, the weight is described as

$$\frac{y^{\infty}_{j\to i}}{y^{\infty}_{\delta_i}} = \frac{y^{\infty}_{j\to i}}{\sum_{j\in\delta_i} y^{\infty}_{j\to i}}. \tag{5.19}$$

Using (5.16), (5.19) and arithmetic-geometric mean inequality with $w_j > 0$, the s_i is bounded as

$$s_i \leqslant \frac{(1+1/\hat{l}\hat{r})^2}{(\sqrt{2}+1)q\hat{l}\hat{r}}. \tag{5.20}$$

The second inequality comes from the definition of average quality. Thus, we obtain the error bound U_{iter}

$$\mathrm{E}_{\mathcal{ALG}} \leqslant \mathrm{U}_{\mathcal{ALG}} = \left(\frac{(1+1/\hat{l}\hat{r})^2}{(\sqrt{2}+1)q\hat{r}}\right) \cdot \frac{1}{\hat{l}m}\sum_{i\in[m]} T_i. \quad \square$$

Next we compare the upper bound of our algorithm to the error performance of majority voting.

Corollary 5.2.1. *Under the hypotheses of Theorem 5.2,*

$$\mathrm{E}_{\mathcal{ALG}} \leqslant \mathrm{U}_{\mathcal{ALG}} \leqslant \mathrm{E}_{\mathcal{MV}} \tag{5.21}$$

Proof of *Corollary 5.2.1* From the results of (9) and (13), we always have

$$1 > \frac{(1+1/\hat{l}\hat{r})^2}{(\sqrt{2}+1)\hat{l}\hat{r}},$$

given the fixed average quality q if the size of batch satisfies $\hat{l}\hat{r} > (\sqrt{2}+1)$ and $l > 1, r > 1$. Even if we assume the sparse random bipartite graph (i.e., $|m| \gg l$, $|n| \gg r$), only a few number of task allocation is sufficient to outperform majority voting. However, the performance gap is further increases as batch size increases. □

Corollary 5.2.2. *Under the hypotheses of Theorem 5.2, if the distribution of the reliability satisfies*

$$\mathbb{P}\Big((\mathrm{w}+1) \geqslant 2\mu_w\Big) \leqslant \frac{(\sqrt{2}+1)\hat{l}\hat{r}}{l(1+1/(\hat{l}\hat{r}))^2},$$

and symmetrical, then the upper bound of $\mathrm{E}_{\mathcal{ALG}}$ *is close to the oracle estimator's average performance.*

$$\mathrm{U}_{\mathcal{ALG}} \rightarrow \mathrm{E}_{\mathcal{OC}}. \tag{5.22}$$

Proof of *Corollary 5.2.2* Here, we compare the average error of our algorithm with that of the oracle estimator. From (7) and (9), the gap between $\mathrm{E}_{\mathcal{ALG}}$ and $\mathrm{E}_{\mathcal{OC}}$ is as follows.

$$\varDelta = \underbrace{\left\{\frac{l(1+1/\hat{l}\hat{r})^2}{(\sqrt{2}+1)q\hat{l}\hat{r}} - \frac{1}{\mathbb{E}_{\mathbb{W}}\big[\mathrm{w}+1\big]}\right\}}_{\varDelta_w} \cdot \frac{1}{lm}\sum_{i\in[m]} T_i. \tag{5.23}$$

To show concisely, the mean and variance of the reliability distribution are denoted as μ_w and σ_w^2 respectively.

Using (5.23) with the definition of $\varDelta_w$, we are left to find the condition that satisfies $\varDelta_w = 0$. If the distribution of worker reliability $\mathbb{W}$ is degenerate distribution with zero variance, the level of each worker is equal. Thus, it is not meaningful to distinguish the quality of the worker and for some $l > 1$and the error bound of *Theorem 5.2* is loose. As mentioned in (11), equal weighting is the best in degenerate distribution. However, when the distribution of worker reliability $\mathbb{W}$ follows certain probabilistic distribution rather than degenerate, we can approximate the quality as

$$q^{-1} = \mathbb{E}_{\mathbb{W}}\left[\frac{1}{\mathrm{w}+1}\right] \cong \frac{1}{\mathbb{E}_{\mathbb{W}}\left[\mathrm{w}+1\right]}\frac{\mathbb{E}_{\mathbb{W}}\left[(w+1)^2\right]}{\mathbb{E}_{\mathbb{W}}\left[w+1\right]^2} \tag{5.24}$$

using third order *Taylor* expansion around $\mathbb{E}_{\mathbb{W}}[w+1]$. It is known that this approximation is quite accurate when the distribution of worker reliability is symmetric. Substituting (5.24) into (5.23) provides $\varDelta_w = 0$ is equivalent to

$$\frac{\sigma_w^2}{\mu_w^2} = \left(\frac{(\sqrt{2}+1)\hat{l}\hat{r}}{l(1+1/\hat{l}\hat{r})^2} - 1\right).$$

Then, applying *Chebyshev*'s inequality, it immediately follows

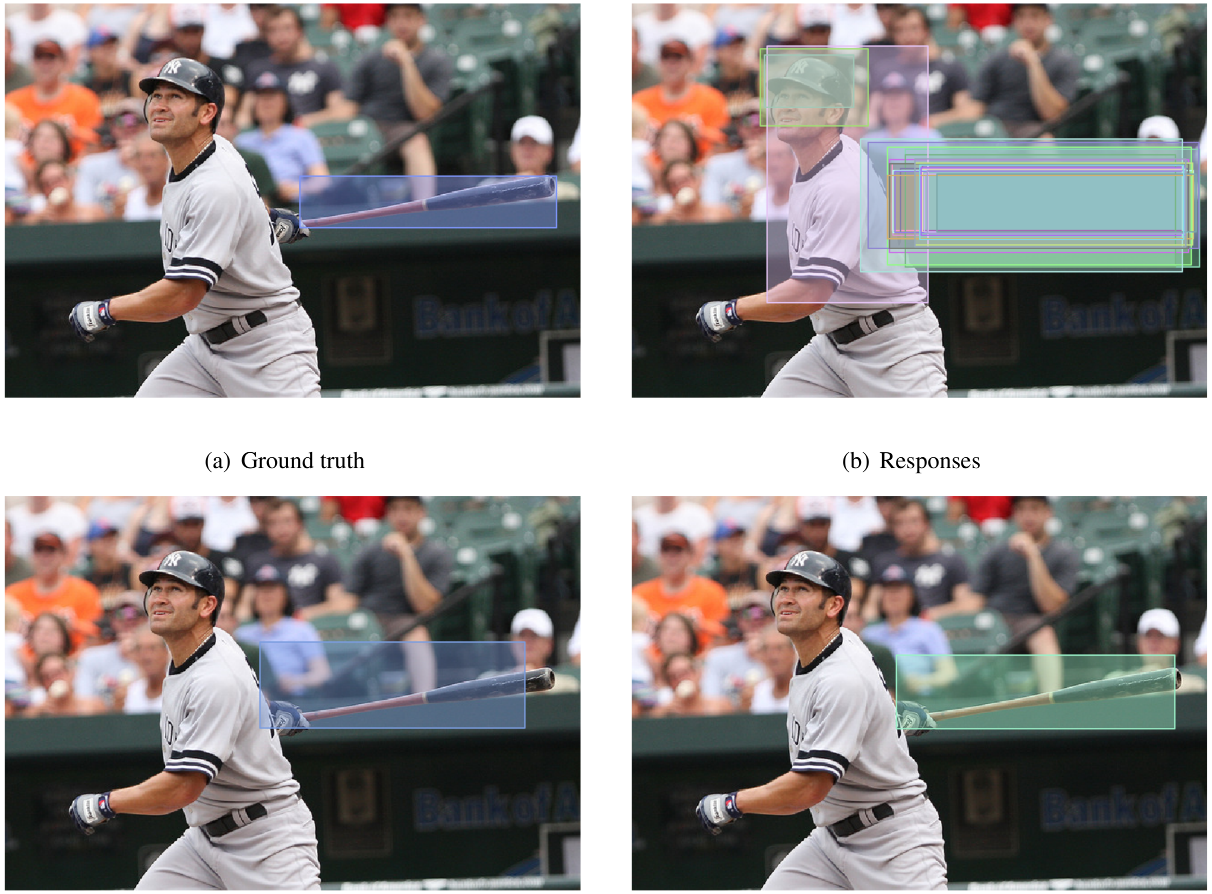
$$\mathbb{P}\Big((\mathrm{w}+1) \geqslant 2\mu_w\Big) \leqslant \frac{(\sqrt{2}+1)\hat{l}\hat{r}}{l(1+1/\hat{l}\hat{r})^2}. \quad \square$$

5.5 Experimental Results

In our experiments, we have evaluated the performance of our algorithm with two popular benchmarks, MSCOCO [10] and the Leeds Sports Pose Extended Training (LSPET) datasets. We compare our algorithm with baselines algorithms which are majority voting ($\mathcal{MV}$) and weighted voting ($\mathcal{WV}$) whose weights are externally given by web-based crowdsourcing platform. We also implemented several state-of-the art which are inner-product method ($\mathcal{IP}$) [50], Welinder's EM model [56], DALE model [48], and outlier rejection methods which are *Mean shift* and *Top-K* selection.

5.5.1 Real Crowdsourcing Data

We crowdsourced two types of tasks in CrowdFlower. One is for image object localization in which the task is to draw a bounding box on the specified object as tightly as possible. The other one is for human pose estimation, where the task is to construct a skeleton-like structure of a human in a given image.



(a) Ground truth
(b) Responses
(c) MV
(d) Ours

Figure 5.5: Drawing a bounding box task on the ‘bat’. (a) the ground truth (b) bounding boxes drawn by 25 workers. (c) estimated answer of majority voting. (d) estimated answer of our algorithm.

Bounding box on MSCOCO dataset. In this task, we randomly chose 2,000 arbitrary images from MSCOCO dataset, and each image was distributed to 25 distinct workers, so there were 50,000 tasks to be solved in total. Total 618 workers were employed, and each worker solved 10 (min) to 100 (max) tasks. We exclude some invalid responses (no box, box over out of bounds [0, image size]). Note that a general bipartite graph is created with different node degrees l_i and r_j, which is not a regular bipartite graph. We measured algorithms' performances by the average error in the ℓ_2 norm and the Intersection over Union (IoU), which is another standard measure for object localization computed by a ratio of intersection area to union area of two bounding boxes. In this experiment, DALE model does not converge due to its complex graphical model raising an out of memory error.

Dataset	MSCOCO		LSPET	
Type	Box(ℓ_2)	Box(IoU)	Joints	Angles
$\mathcal{WV}$	0.22227	0.89593	0.15877	0.10524
$\mathcal{MV}$	0.22090	0.89666	0.15858	0.10462
$\mathcal{IP}$	0.22026	0.89712	0.15483	0.10462
Welinder	0.21886	0.89821	N/A	N/A
DALE	0.21834	0.89914	N/A	N/A
Top-K	0.18869	0.91250	0.12222	0.10051
MeanShift	0.18034	0.92150	0.11812	0.09962
Ours	**0.14837**	**0.93445**	**0.09308**	**0.09941**

Table 5.2: An error table of experimental results on real crowdsourced data where the tasks are (1st column) an object detection on MSCOCO dataset, (2nd column) same task with Intersection of Union measure (3rd column) a human joints estimation and (4th column) an angle segmentation by neck and adjacent human joints on LSPET dataset. For Top-K selection, we choose K as a half of the task degree l.

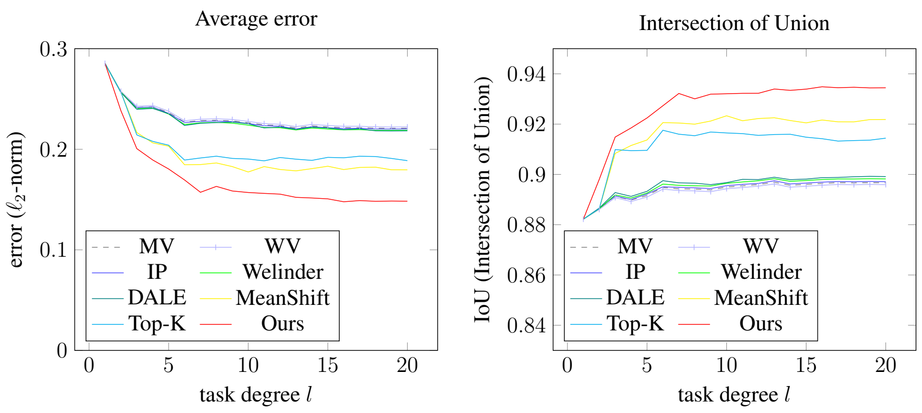


Figure 5.6: Comparisons of error and IoU between different algorithms with varying task degree l.

To measure the performance of DALE model in smaller data, we collected a dedicated dataset of 100 images each of which was assigned to 20 distinct workers. Results are listed in Table 5.2 with two evaluation metric Euclidean distance(ℓ_2) and Intersection over Union(IoU). Our algorithm significantly outperforms others and, even with small number of iterations, can reduce errors rapidly. Empirically, our algorithm converges in less than 20 iterations as plotted in Figure 5.7.

Varying degree on MSCOCO dataset. Here we show how the performances of different algorithms vary with task degree l. We made a number of task-worker bipartite graphs by randomly dropping some edges to make degree l for each task. As expected, the average error of each algorithm decreases as the task degree l increases. Even when the degree value falls until 5, ours can still keep the large gap among other algorithms. In other words, our algorithm needs less budget to get same error rate. The results are listed in Figure 5.6.

Robustness. Since it is well known that message-passing algorithms suffers from the initialization issue in general, we tested robustness of our algorithm by initializing workers' weights to be sampled from proper distributions with moderate hyperparameters. Here we used *Beta* distribution with (α, β), and *Gaussian* distribution with

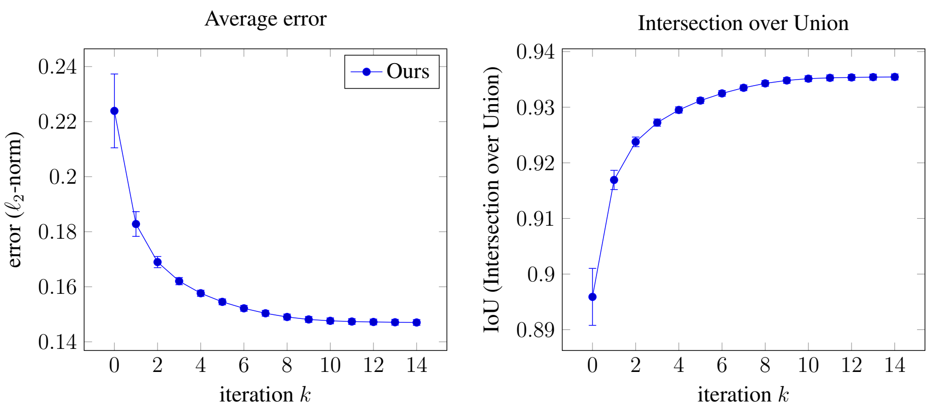


Figure 5.7: Error bar plots of our algorithm for the initialization issue on 2k-edge bounding box task.

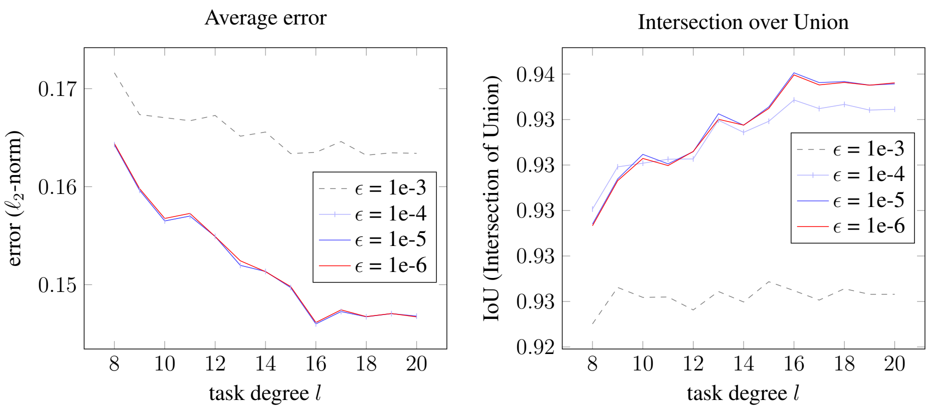


Figure 5.8: The influence of ϵ on error and IoU when computing y-messages with varying task degree l.

(μ, σ^2) sampled from uniform distribution $\mathcal{U}$. The result is shown by error bar plots in Figure 5.7 which represents the deviation reduces rapidly. This result shows that our algorithm is robust to the initialization of workers' weights.

When the number of edges are not sufficient to estimate worker message, our algorithm can diverge as iteration progresses since worker message is computed by the reciprocal of the summation between the response and the task message. It can be resolved by adding a very small positive constant ϵ on the summation before computing the reciprocal.

$$y^{(k)}_{j\to i} = \left(\frac{1}{\hat{r}} \sum_{i' \in \delta_{j \setminus i}} \left(\|\boldsymbol{A}_{ij} - \boldsymbol{x}^{(k)}_{i\to j}\|_2)^2 \right) + \epsilon \right)^{-1}. \tag{5.25}$$

We investigate the influence of ϵ in Figure 5.8. This result shows our algorithm works well when $\epsilon \leq 10^{-5}$.

Human pose estimation. We collected the human pose estimation data of 1,000 images chosen from LSPET dataset using CrowdFlower platform. Each image was distributed to ten distinct workers who were asked to mark dots on the 14 human joints (head, neck, left/right shoulders, elbows, wrists, hips, knees, and ankles). In this experiment, we aggregated their answers to estimate the point of each human joint. Moreover, we estimated angles from the neck and adjacent joints (head, shoulders, hips) as another task which is also important in pose estimation. Estimating angles can be viewed as dividing angle task whose domain is $[0, 2\pi]$. As shown in Table 5.2, our algorithm outperforms others on both joint and angle estimation tasks.

5.5.2 Verification of the Error Bounds with Synthetic data

In order to empirically verify the correctness of the analysis, experiments were performed with synthetic dataset. Assuming hypothetical 2,000 workers and 2,000 tasks with two dimensions ($D = 2, 5$), task assignment follows regular bipartite graph. The performance of the oracle estimator is presented as a theoretical lower bound. Also, each result is averaged of 20 experiments by changing the initial value.

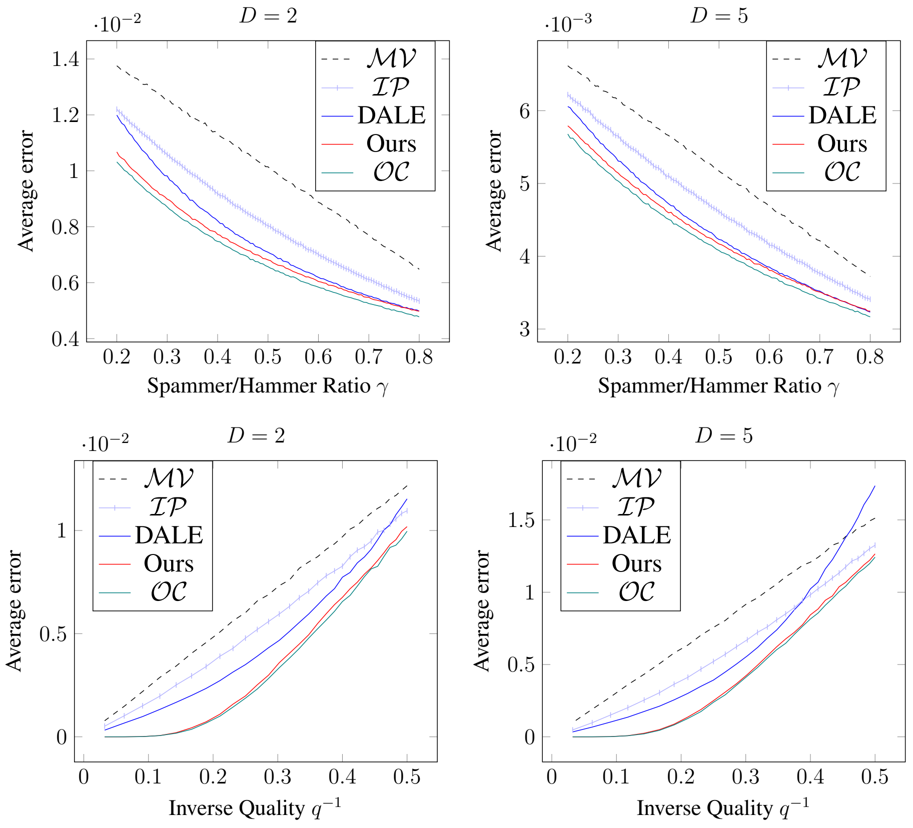


Figure 5.9: Comparison of average errors between different algorithms with $D = (2, 5)$: (*top*) varying γ ($w_s = 0.5$, $w_h = 5$), (*bottom*) varying q.

Spammer/Hammer ratio. In this experiment, we assume the *Spammer/Hammer* scenario which means that each worker is randomly sampled from a Spammer ($w_s = 0.5$) or a Hammer ($w_h = 5$); the response of a Hammer is much closer to the ground truth than that of a Spammer. The ratio γ denotes the Hammer proportion of all workers. Figure 5.9 (left) shows that our algorithm can distinguish Hammer from Spammer much better than others.

Quality. According to the definition of (5.7), the reliability of each worker was drawn from *Beta* distribution $\big($i.e., $(1 + w)^{-1} \sim$ *Beta*$(\alpha, \beta)\big)$. In Figure 5.9 (right), our algorithm shows a large performance gap when the quality is sufficiently high. The average errors of the five algorithms are indistinguishable when the quality is low, but our algorithm is better at estimating the workers' reliabilities if the quality is sufficiently high. Since our algorithm regards the average response of other workers as approximated true answers, high quality promotes its performance.

5.6 Conclusion

In this study, we have proposed an iterative algorithm for vector regression tasks. We observed the considerable gains with both real and synthetic datasets through various experiments. In the theoretical analysis, we proved that the error bound depends on the average worker quality and the number of queries batch achieving near-optimal performance in the probabilistic worker model. Our work can be easily generalized to many image processing tasks such as 3D image processing and multiple object detection. Also, it can be exploited for estimating the precise level of workers in an adaptive manner.

Chapter 6

Conclusions

In this dissertation, we propose iterative algorithms to infer groundtruths by aggregating noisy answers from crowdsourcing workers. We explore several types of crowdsourcing tasks that contains multiple-choice question, short-answer questions, K-approval voting, and real-valued vector regression. These tasks are fundamental components composing complex crowdsourcing tasks, but have been not of interest to academic studies in general. Our proposed algorithms provide reliable solutions for those tasks with performance guarantees. In particular, our proposed algorithms achieve order-optimal performance where the optimal performance comes from oracle estimator who knows every worker's reliability. We give a rigorous theoretical analysis for the proposed algorithms and verify their remarkable performance by numerous experiments in both real and synthetic crowdsourcing datasets. We hope that our work would be of practical use in web-based crowdsourcing services.

Bibliography

[1] Chris J Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan 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.

[2] TechCrunch. recaptcha: Using captchas to digitize books. `https://techcrunch.com/2007/09/16/recaptcha-using-captchas-to-digitize-books/`, 2007. [Online; September 16, 2007].

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[4] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew 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.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[7] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[9] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[11] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[12] Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. In *CVPR 2011*, pages 1465–1472. IEEE, 2011.

[13] Benjamin Sapp and Ben Taskar. Modec: Multimodal decomposable models for human pose estimation. In *In Proc. CVPR*, 2013.

[14] Dieu-Thu Le, Jasper Uijlings, and Raffaella Bernardi. Tuhoi: Trento universal human object interaction dataset. In *Proceedings of the Third Workshop on Vision and Language*, pages 17–24, 2014.

[15] Chao-Yeh Chen and Kristen Grauman. Predicting the location of "interactees" in novel human-object interactions. In *Asian conference on computer vision*, pages 351–367. Springer, 2014.

[16] Yu-Wei Chao, Zhan Wang, Yugeng He, Jiaxuan Wang, and Jia Deng. Hico: A benchmark for recognizing human-object interactions in images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1017–1025, 2015.

[17] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International journal of computer vision*, 101(1):184–204, 2013.

[18] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3282–3289. IEEE, 2012.

[19] David Tsai, Matthew Flagg, Atsushi Nakazawa, and James M Rehg. Motion coherent tracking using multi-label mrf optimization. *International journal of computer vision*, 100(2):190–202, 2012.

[20] Guangnan Ye, Yitong Li, Hongliang Xu, Dong Liu, and Shih-Fu Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 471–480, 2015.

[21] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neu-

ral networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[22] Yu-Gang Jiang, Jingen Liu, A Roshan Zamir, George Toderici, Ivan Laptev, Mubarak Shah, and Rahul Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014.

[23] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 2017.

[24] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015.

[25] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[26] Stanislaw Antol, C Lawrence Zitnick, and Devi Parikh. Zero-shot learning via visual abstraction. In *European conference on computer vision*, pages 401–416. Springer, 2014.

[27] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

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

[29] Victor S Sheng, Foster Provost, and Panagiotis 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.

[30] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322, 2010.

[31] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[32] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul 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.

[33] David R Karger, Sewoong Oh, and Devavrat 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.

[34] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961, 2011.

[35] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems. *Operations Research*, 62(1):1–24, February 2014.

[36] Qiang Liu, Jian Peng, and Alex Ihler. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 692–700, 2012.

[37] David R Karger, Sewoong Oh, and Devavrat 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.

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

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

[40] 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.

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

[42] Yaling Zheng, Stephen Scott, and Kun 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.

[43] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[44] Noga Alon and Joel H. Spencer. *The probabilistic method*. John Wiley & Sons, Hoboken, NJ, 2008.

[45] Nihar B Shah, Dengyong Zhou, and Yuval Peres. Approval voting and incentives in crowdsourcing. *arXiv preprint arXiv:1502.05696*, 2015.

[46] Ariel D Procaccia and Nisarg Shah. Is approval voting optimal given approval votes? In *Advances in Neural Information Processing Systems*, pages 1792–1800, 2015.

[47] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pages 1260–1268, 2014.

[48] Mahyar Salek and Yoram Bachrach. Hotspotting-a probabilistic graphical model for image object localization through crowdsourcing. In *AAAI*, 2013.

[49] Dengyong Zhou, Qiang Liu, John C Platt, and Christopher Meek. Aggregating ordinal labels from crowds by minimax conditional entropy. In *ICML*, volume 14, pages 262–270, 2014.

[50] Donghyeon Lee, Joonyoung Kim, Hyunmin Lee, and Kyomin Jung. Reliable multiple-choice iterative algorithm for crowdsourcing systems. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 205–216. ACM, 2015.

[51] Joonyoung Kim, Donghyeon Lee, and Kyomin Jung. Reliable aggregation method for vector regression tasks in crowdsourcing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 261–273. Springer, 2020.

[52] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[53] Steve Branson, Grant Van Horn, and Pietro Perona. Lean crowdsourcing: Combining humans and machines in an online system. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017.

[54] Grant Van Horn, Steve Branson, Scott Loarie, Serge Belongie, and Pietro Perona. Lean multiclass crowdsourcing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2714–2723, 2018.

[55] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pages 285–294. ACM, 2013.

[56] Peter Welinder and Pietro Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 25–32. IEEE, 2010.

[57] Asif R Khan and Hector Garcia-Molina. Attribute-based crowd entity resolution. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 549–558. ACM, 2016.

[58] A. Vempaty, L.R. Varshney, and P.K. Varshney. Reliable crowdsourcing for multi-class labeling using coding theory. *Selected Topics in Signal Processing, IEEE Journal of*, 8(4):667–679, August 2014.

[59] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, volume 1, 2012.

[60] Yao Ma, Alex Olshevsky, Venkatesh Saligrama, and Csaba Szepesvari. Crowdsourcing with sparsely interacting workers. *arXiv preprint arXiv:1706.06660*, 2017.

[61] Chenxi Qiu, Anna C Squicciarini, Barbara Carminati, James Caverlee, and Dev Rishi Khare. Crowdselect: increasing accuracy of crowdsourcing tasks through behavior prediction and user selection. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 539–548. ACM, 2016.

[62] Yao Zhou, Lei Ying, and Jingrui He. Multic2: an optimization framework for learning from task and worker dual heterogeneity. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 579–587. SIAM, 2017.

[63] Béla Bollobás. Random graphs. In *Modern graph theory*, pages 215–252. Springer, 1998.

[64] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge university press, 2008.

초 록

다양한 분야에서 라벨된 빅데이터를 필요로 하는 현재, 웹 기반 크라우드소싱 서비스들이 출범하며 상대적으로 적은 예산과 짧은 시간에도 효율적으로 사람들의 지혜를 활용할 수 있는 방법들이 제시되고 있다. 이러한 방법들의 효율성에도 불구하고, 크라우드소싱 시스템의 선천적인 문제점은 일을 맡은 사람들의 적은 보상 및 책임감 결여로 인해 그들의 응답을 완전히 신뢰할 수 없다는 점에 있다. 이에 다수결 방식이 자연스러운 해법으로 사용되지만, 보다 신뢰 높은 답을 얻어내기 위해 많은 연구들이 진행되고 있다. 본 박사학위 논문에서는 크라우드소싱 시스템에서 수많은 사람들로부터 받은 응답들을 모아 신뢰성 높은 응답을 추론하는 반복적 메세지전달 형태의 알고리즘들을 제시한다. 본 알고리즘들은 낮은랭크근사에 기반한 반복 추론 방법으로, 기존에 각광받던 EM 알고리즘들에 비해 더 빠르고 신뢰적인 정답을 추론해낸다. 더불어 본 알고리즘들의 추론 정확도가 최적에 매우 근접하며 다수결 방식 및 EM 알고리즘들의 정확도를 상회한다는 것을 이론적 증명 및 실험적 결과를 통해 제시한다. 본 연구는 실제 크라우드소싱에서 대다수의 응답 유형을 차지하는 객관식 응답, 주관식 응답, 복수 선택 응답, 및 실수 값 응답의 추론 문제를 다루며, 기존 양자택일 응답 추론 문제만을 다루는 기존 연구들과 큰 차별성을 가진다.

주요어: 크라우드소싱, 메세지전달 형태 알고리즘, 근사 추론

학번: 2013-23125

ACKNOWLEGEMENT

I would like to say a special thank you to my supervisor, Kyomin Jung. His support, guidance and overall insights in this field have made this an inspiring experience for me. I would also like to thank all of MILaboratory colleagues, understanding and help throughout my research and projects, especially to Joonyoung Kim for our numerous works as a team. Finally, I would like to say my biggest thanks to my family for all the unconditional support through my long academic years.