Reducing the Cost of Aggregation in Crowdsourcing

Rituraj Singh, Loïc Hélouët, Zoltan Miklos

Reducing the Cost of Aggregation in Crowdsourcing

Singh, Rituraj and Hélouët, Loïc and Miklos, Zoltan

Univ Rennes/INRIA/CNRS/IRISA
rituraj.singh@irisa.fr,loic.helouet@inria.fr, zotlan.miklos@irisa.fr

Abstract. Crowdsourcing is a way to solve problems that need human contribution. Crowdsourcing platforms distribute replicated tasks to workers, pay them for their contribution, and aggregate answers to produce a reliable conclusion. A fundamental problem is to infer a correct answer from the set of returned results. Another challenge is to obtain a reliable answer at a reasonable cost: unlimited budget allows hiring experts or large pools of workers for each task but a limited budget forces to use resources at best.

This paper considers crowdsourcing of simple boolean tasks. We first define a probabilistic inference technique, that considers difficulty of tasks and expertise of workers when aggregating answers. We then propose CrowdInc, a greedy algorithm that reduce the cost needed to reach a consensual answer. CrowdInc distributes resources dynamically to tasks according to their difficulty. We show on several benchmarks that CrowdInc achieves good accuracy, reduces costs, and we compare its performance to existing solutions.

1 Introduction

Crowdsourcing is a way to solve tasks that need human contribution. These tasks include image annotation or classification, polling, etc. Employers publish tasks on an Internet platform, and these tasks are realized by workers in exchange for a small incentive [1]. Workers are very heterogeneous: they have different origins, domains of expertise, and expertise levels. One can even consider malicious workers, that return wrong answers on purpose. To deal with this heterogeneity, tasks are usually replicated: each task is assigned to a *set of workers*. Redundancy is also essential to collect workers opinion: in this setting, work units are the basic elements of a larger task that can be seen as a poll. One can safely consider that each worker executes his assigned task independently, and hence returns his own belief about the answer. As workers can disagree, the role of a platform is then to build a consensual final answer out of the values returned. A natural way to derive a final answer is **Majority Voting** (MV), i.e. choose as conclusion the most represented answer. A limitation of MV is that all answers have equal weight, regardless of expertise of workers. If a crowd is composed of only few experts, and of a large majority of novices, MV favors answers from novices. However, in some domains, an expert worker may give better answer than a novice and his answer should be given more weight. One can easily replace MV by a weighted vote. However, this raises the question of measuring workers expertise, especially when workers competences are not prior known.

Crowdsourcing platforms such as Amazon Mechanical Turk (AMT) do not have prior knowledge about the expertise of their worker. A way to obtain initial measure of workers expertise is to use **Golden Questions** [8]. Several tasks with known *ground truth* are used explicitly or hidden to evaluate workers expertise. As already mentioned, a single answer for a particular task is often not sufficient to obtain a reliable answer, and one has to rely on redundancy, i.e. distribute the same task to several workers and aggregate results to build a final answer. Standard *static* approaches fix prior number of k workers for each task. Each task are displayed at platform and waits for bid by the k workers and is use by AMT. There is no guideline to set the value for k, but two standard situations where k is fixed are frequently met. The first case is when a client has n tasks to complete with total budget B_0 incentive units. Each task can be realized by $k = B_0/n$ workers. The second case is when an initial budget is not known, and the platforms fixes an arbitrary prior level of redundancy. In this case, the number of workers allocated to each tasks is usually between 3 and 10 [6]. It is assumed that the distribution of work is uniform, i.e. that each task is assigned same number of workers, regardless of its difficulty. An obvious drawback of static allocation of workers is that all tasks benefit from the same work power, regardless of their difficulty. Further, the final answer returned by a platform is usually obtained as a consensus among returned answers. Even a simple question where the variance of answers is high calls for sampling of larger size. So, one could expect each task t to be realized by k_t workers, where k_t is a number that guarantee that the likelihood to change the final answer with one additional worker is low. However, without prior knowledge on task's difficulty and on variance in answers, this number k_t cannot be fixed.

In this paper, we address the questions of answers aggregation, task allocation, and we study the cost of crowdsourcing. For simplicity, we consider boolean filtering tasks, i.e. tasks with answers in $\{0, 1\}$, but the setting can be easily extended to tasks with any finite set of answers. These tasks are frequent, for instance to decide whether a particular image belongs or not to a given category of pictures. We consider that each binary task has a *truth label*, i.e. there exists a ground truth for each task. Each worker is asked to answer 0 or 1 to such a task and returns a so-called *observed label*, which may differ from the ground truth. The *difficulty* of a task is a real value in $[0, 1]$. A task with difficulty 0 is a very easy task and a task with difficulty 1 a very complex one. The *expertise* of a worker is modeled in terms of *recall* and *specificity*. **Recall** (also called true positive rate) measures the proportion of correct observed labels given by a worker when the ground truth is 1. On contrary, **specificity** (also called true negative rate) measures the proportion of correct observed labels given by a worker when the ground truth is 0. We propose a generating function to measure the probability of accuracy for each of the truth label (0/1) based on the *observed label, task difficulty, worker expertise*. We rely on Expectation Maximization (EM) based algorithm to maximize the probability of accuracy of ground truth for each task and jointly estimate the difficulty of each of the task as well as expertise of the workers. The algorithm provides a greater weight to the expert worker. In addi-

tion, if a worker with high *recall* makes a mistake in the *observed label*, then it increases the difficulty of the task (correspondingly for specificity). Along with, if expert workers fail to return a correct answer, then the task is considered difficult. The EM algorithm converges with a very low error rate and at the end returns the task *difficulty*, worker *expertise* and the *final estimated label* for each of the task based on *observed label*.

Additionally, we propose a dynamic worker allocation algorithm that handles at the same time aggregation of answers, and optimal allocation of a budget to reach a consensus among workers. The algorithm works in two phases. For the initial *Estimation* phase, as we do not have any prior information about the task difficulty and worker expertise, we allocate one third of total budget to inspect the behavior of each task. Based on the answers provided by the human workers for each of the task, we first derive the difficulty of each of the task, final aggregated answer and along with the worker expertise using an EM algorithm. For each task, we estimate the likelihood that the aggregated answer is the ground truth. We terminate tasks which are above the derived threshold at that particular instance. The second phase is named as *Explore*. Based on each of the estimated task difficulty, we start to allocate workers for each of the remaining tasks. The process continues until all task are terminated or the whole budget is consumed.

Related work: Several papers have considered tools such as EM to aggregate answers, or allocate tasks. We do not perform a complete survey of the domain, but only highlight a few works that are close to our approach, and refer interested readers to [19], a survey. Zencrowd [3] considers workers competences in terms of accuracy (ratio of correct answers) and aggregates answers using EM. PM [9] considers an optimization scheme based on Lagrange multipliers. Workers accuracy and ground truth are the hidden variables that must be discovered and optimize in order to minimize the deviations between workers answers and aggregated conclusion. D&S [2] uses EM to synthesize answers that minimize error rates from a set of patient records. It considers recall and specificity, but not difficulty of tasks.

[7] proposes an algorithm to affect tasks to workers, synthesize answers, and reduce the cost of crowdsourcing. They assume that all tasks have the same difficulty, and that worker reliability is a consistent value in $[0, 1]$ (whence considering accuracy as a representation of competences). CrowdBudget [14] is an approach that divides a budget B among K existing tasks to achieve a low error rate, and then uses MV to aggregate answers. Workers answers follow an unknown Bernoulli distribution. The objective is to affect the most appropriate number of workers to each task in order to reduce the estimation error. Aggregation is done using Bayesian classifiers combination (BCC). The approach in [15] extends BCC with communities and is called CBCC. Each worker is supposed to belong to a particular (unknown) community, and to share characteristics of this community (same recall and specificity). This assumption helps improving accuracy of classification. Expectation maximization is used by [13] to improve supervised learning when the ground truth in unknown. This work considers re-

call and specificity of workers and proposes maximum-likelihood estimator that jointly learns a classifier, discover the best experts, and an estimation of ground truth. Most of the works cited above consider expertise of workers but do not address tasks difficulty. An exception is GLAD (Generative model of Labels, Abilities, and Difficulties) [18] that proposes to estimate tasks difficulty as well as workers accuracy to aggregate final answers. The authors recall the EM is an iterative process that stops only after converging, but demonstrate that the EM approach needs only a few minutes to tag a database with 1 million images. Notice that expertise in GLAD is not expressed in terms of recall not specificity. Most of the works in database and machine learning focus on data aggregation techniques and leave budget optimization apart. Raykar et.al [12] introduce sequential crowdsourced labeling, where instead of asking for all the labels in one shot, decides at each step whether evaluation of a task shall be stopped, and which worker should be hired. The model incorporates a Bayesian model for workers (workers are only characterized by their accuracy), and cost. Then, sequential crowdsourced labeling amounts to exploring a (very large) Markov decision process (states contain all pairs of task/label collected at a given instant) with a greedy strategy.

It is usually admitted [19] that recall and specificity gives a finer picture of worker's competence than accuracy. Our work aggregates workers answers using expectation maximization with three parameters : task difficulty, recall and specificity of workers. The CrowdInc algorithm uses this EM aggregation to estimate error and difficulty of tasks. This error allows to compute dynamically a threshold to stop tasks which aggregated answers have reached a reasonable reliability and to allocate more workers to the most difficult tasks, hence saving costs. One can notice that we assign an identical cost to all tasks. This makes sense, as the difficulty of tasks are prior unknown.

The rest of the paper is organized as follows. In Section 2.1, we introduce our notations, the factors that influence results during aggregation of answers, and the EM algorithm. In Section 3, we present a model for workers and our EM-based aggregation technique. We detail the CrowdInc algorithm to optimize the cost of crowdsourcing in Section 4. We then give results of experiments with our aggregation technique and with *CrowdInc* in Section 5. Finally we conclude and give future research directions in Section 6.

2 Preliminaries

In the rest of the paper, we will work with discrete variables and discrete probabilities. A random variable is a variable whose value depends on random phenomenon. For a given variable x, we denote by $Dom(x)$ its domain. For a particular value $v \in Dom(x)$ we denote by $x = v$ the event "x has value v". A probability measure $Pr()$ is a function from a domain to interval $[0, 1]$. We denote by $Pr(x = v)$ the probability that event $x = v$ occurs. In the rest of the paper, we mainly consider boolean events, i.e. variables with domain $\{0, 1\}$. A probability of the form $Pr(x = v)$ only considers occurrence of a single event. When considering several events, we define the *joint probability* $Pr(x = v, y = v')$

the probability that the two events occur simultaneously. The notation extends to an arbitrary number of variables. If x and y are independent variables, then $Pr(x = v, y = v') = Pr(x = v) \cdot Pr(y = v')$. Last, we will use conditional probabilities of the form $Pr(x = v \mid y = v')$, that defines the probability for an event $x = v$ when it is known that $y = v'$. We recall that, when $P(y = v') > 0$ $Pr(x = v \mid y = v') = \frac{Pr(x=v, y=v')}{Pr(y=v')}$.

2.1 Factors influencing efficiency of crowdsourcing

During task labeling, several factors can influence the efficiency of crowdsourcing, and the accuracy of aggregated answers. The first of them is **Task difficulty**. Tasks submitted to crowdsourcing platforms by a client address simple questions, but from different domains, that may need expertise. Event within a single domain, the difficulty for the realization of a particular task may vary from one experiment to another: tagging an image can be pretty simple if the worker only has to decide whether the picture contains an animal or an object, or conversely very difficult if the boolean question asks whether a particular insect picture shows an hymenopteran (an order of insects). The difficulty of a task may be known prior by the client, but in most cases, one has no prior measure of the difficulty of a task. Similarly, **Expertise of workers** play a major role in accuracy of aggregated answers. In general, an expert worker performs better on a specialized task than a randomly chosen worker without particular competence in the domain. For example, an entomologist can annotate an insect image more precisely than any random worker.

The technique used for **Amalgamation** also play a major role. Given a set of answers returned for a task t, one can aggregate the results using *majority voting* (MV), or more interesting, as a weighted average answer where individual answers are pondered by workers expertise. However, it is difficult to get a prior measure of workers expertise and of the difficulty of tasks. Many crowdsourcing platforms use MV and ignore difficulty of tasks and expertise of workers to aggregate answers or allocate workers to tasks. We show in Section 5 that MV has a low accuracy.

In Section 3, we propose a technique to estimate the expertise of workers and difficulty of tasks on the fly. Intuitively, one wants to consider a task difficult if even experts fail to provide a correct answer for this task, and consider it easy if even workers with low competence level answer correctly. Similarly, a worker is competent if he answers correctly difficult tasks. Notice however that to measure difficulty of tasks and expertise of workers, one needs to have the final answer for each task. Conversely, to precisely estimate the final answer one needs to have the worker expertise and task difficulty. This is a chicken and egg situation, but we show in section 3 how to get plausible value for both.

The next issue to consider is the **cost** of crowdsourcing. Workers receive incentives for their work, but usually clients have limited budgets. Some task may require a lot of answers to reach the consensus, while some may require only a few answers. Therefore, a challenge is to spend efficiently the budget to get the most accurate answers. In Section 4, we discuss some of the key factors

in budget allocation. Many crowdsourcing platforms do not considers *difficulty*, and allocate the same number of workers to each task. The allocation of many workers to simple tasks is usually not justified and is a waste of budget that would be useful for difficult tasks. Now, tasks difficulty is not prior known. This advocates for on the fly worker allocation once the difficulty of a task can be estimated.

Last, one can stop collecting answers for a task when there is an evidence that enough answers have been collected to reach a consensus of a final answer. A immediate solution is to measure the confidence of final aggregated answer and take as **Stopping Criterion** for a task the fact that this confidence exceeds a choose threshold. However, the criteria do not works well in practice as client usually want high thresholds for all their tasks. This may lead to consuming all available budget without reaching an optimal accuracy. Ideally, we would like to have a stopping criteria that balances both in terms of confidence in the final answers and budget, optimizing the overall accuracy of all the tasks.

2.2 Expectation Maximization

Expectation Maximization [4] is an iterative technique to obtain maximum likelihood estimation of parameter of a statistical model when some parameters are unobserved and *latent*, i.e. they are not directly observed but rather inferred from observed variables. In some sense, the EM algorithm is a way to find the best fit between data samples and parameters. It has many applications in machine learning, data mining and Bayesian statistics.

Let $\mathcal{M}$ be a model which generates a set $\mathcal{X}$ of observed data, a set of missing latent data $\mathcal{Y}$, and a vector of unknown parameters θ, along with a likelihood function $L(\theta \mid \mathcal{X}, \mathcal{Y}) = p(\mathcal{X}, \mathcal{Y} \mid \theta)$. In this paper, observed data $\mathcal{X}$ represents the answers provided by the crowd, $\mathcal{Y}$ depicts the *final answers* which needs to be estimated and is hidden, and parameters in θ are the *difficulty* of tasks and the *expertise* of workers. The *maximum likelihood estimate* (MLE) of the unknown parameters is determined by maximizing the marginal likelihood of the observed data. We have $L(\theta \mid \mathcal{X}) = p(\mathcal{X} \mid \theta) = \int p(\mathcal{X}, \mathcal{Y} \mid \theta) d\mathcal{Y}$. The EM algorithm computes iteratively MLE, and proceeds in two steps. At the k^{th} iteration of the algorithm, we let θ^k denote the estimate of parameters θ. At the first iteration of the algorithm, θ^0 is randomly chosen.

E-Step: In the E step, the missing data are estimated given observed data and current estimate of parameters. The E-step computes the expected value of $L(\theta \mid \mathcal{X}, \mathcal{Y})$ given the observed data $\mathcal{X}$ and the current parameter θ^k. We define

$$Q(\theta \mid \theta^k) = \mathbb{E}_{\mathcal{Y} \mid \mathcal{X}, \theta^k}[L(\theta \mid \mathcal{X}, \mathcal{Y})] \tag{1}$$

In the crowdsourcing context, we use the E-Step to compute the probability of occurrence of $\mathcal{Y}$ that is the *final answer* for a each task, given the observed data $\mathcal{X}$ and parameters θ^k obtained at k^{th} iteration.

M-Step: The M-step finds parameters θ that maximize the expectation computed in equation. 1.

$$\theta^{k+1} = \arg\max_{\theta} Q(\theta \mid \theta^k) \tag{2}$$

Here, with respect to estimated probability for $\mathcal{Y}$ for *final answers* from the last E-Step, we maximize the joint log likelihood of the observed data $\mathcal{X}$ (answer provided by the crowd), hidden data $\mathcal{Y}$ (final answers), to estimate the new value of θ^{k+1} i.e. the *difficulty* of tasks and the *expertise* of workers. The E and M steps are repeated until the value of θ^k converges. A more general version of the algorithm is presented in algorithm 1.

Algorithm 1: General EM Algorithm

Data: Observed Data $\mathcal{X}$
Result: Parameter values θ, Hidden data $\mathcal{Y}$
Initialize parameters in θ^0 to some random values.
while $||\theta^k - \theta^{k-1}|| > \epsilon$ **do**
    Compute the expected possible value of $\mathcal{Y}$, given θ^k and observed data $\mathcal{X}$
    Use $\mathcal{Y}$ to compute the values of θ that maximize $Q(\theta \mid \theta^k)$.
end
return parameter θ^k, Hidden data $\mathcal{Y}$

3 The Aggregation model

We address the problem of evaluation of binary properties of samples in a dataset by aggregation of answers returned by participants in a crowdsourcing system. This type of application is frequently met: one can consider for instance a database of n images, for which workers have to decide whether each image is clear or blur, whether a cat appears on the image, etc. The evaluated property is binary, i.e. workers answers can be represented as a label in $\{0, 1\}$. From now, we will consider that tasks are elementary work units which objective is to associate a binary label to a particular input object. For each task, an actual ground truth exists, but it is not known by the system. We assume a set of k independent workers, which role is to realize a task, i.e. return an *observed label* in $\{0, 1\}$ according to their perception of a particular sample. We consider a set of tasks $T = \{t_1, \dots t_n\}$ that need to be evaluated. For a task $t_j \in T$ the observed label given by worker $1 \leq i \leq k$ is denoted by l_{ij}. We let y_j denote the *final label* of an image j obtained by aggregating the answers of all workers. $L_j = \bigcup_{i \in 1..k} l_{ij}$ denotes the set of all labels returned by workers for task j, L denotes the set of all observed labels, $L = \bigcup_{j \in 1..n} L_j$. The goal is to estimate the ground truth by synthesizing a set of *final label* $Y = \{y_j, 1 \leq j \leq n\}$ from the set of *observed label* $L = \{L_j\}$ for all tasks.

Despite the apparent simplicity of the problem, crowdsourcing binary tagging tasks hides several difficulties, originating from unknown parameters. These parameters are the difficulty of each task, and the expertise of each worker. The difficulty of task t_j is modeled by a parameter $d_j \in (0, 1)$. Here value 0 means that the task is very easy, and can be performed successfully by any worker. On the other hand, $d_j = 1$ means that task t_j is very difficult. A standard way to measure expertise is to define workers accuracy as a pair $\xi_i = \{\alpha_i, \beta_i\}$, where

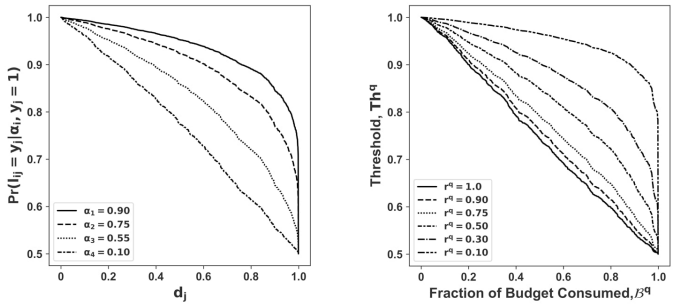

Fig. 1: (left) Generative function for the probability to get $l_{ij} = 1$, given $y_j = 1$, for growing values of task difficulty. The curves represent different recall for the considered workers.(right) The threshold values based on current estimate on consumed budget and fraction of task remaining at the beginning of a round.

α_i is called the *recall* of worker i and β_i the *specificity* of worker i. The **recall** is the probability that worker i annotates an image j with label 1 when the ground truth is 1, i.e. $\alpha_i = Pr(l_{ij} = 1|y_j = 1)$. The **specificity** of worker i is the probability that worker i annotates an image j with 0 when the ground truth is 0, i.e. $\beta_i = Pr(l_{ij} = 0|y_j = 0)$.

In literature,[19] the expertise of workers is often quantified in terms of *accuracy*, i.e. $Pr(l_{ij} = y_j)$. However, if the data samples are unbalanced, i.e. the number of samples with actual ground truth 1 (respectively 0) is much larger than the number of samples with ground truth 0 (respectively 1), defining competences in terms of *accuracy* leads to bias. Indeed, a worker who is good in classifying images with ground truth 1 can obtain bad scores when classifying image with ground truth 0, and yet get a good accuracy (this can be the case of a worker that always answers 1 when tagging a task). *Recall* and *Specificity* overcomes the problem of bias and separates the worker expertise, considering their ability to answer correctly when the ground truth is 0 and when it is 1, and hence give a more precise representation of workers competences.

Another advantage of knowing recall and specificity is that it allows to build a probabilistic model (a generative model) for workers answers. We assume that workers have constant behaviors and are faithful, i.e. do not return wrong answers intentionally. We also assume that workers do not collaborate (their answers are independent variables). Under these assumptions, knowing the recall α_i and specificity β_i of a worker i, we build a model that generates the probability that he returns an *observed label* l_{ij} for a task j with difficulty d_j:

$$Pr(l_{ij} = y_j|d_j, \alpha_i, y_j = 1) = \frac{1 + (1 - d_j)^{(1-\alpha_i)}}{2} \tag{3}$$

$$Pr(l_{ij} = y_j|d_j, \beta_i, y_j = 0) = \frac{1 + (1 - d_j)^{(1-\beta_i)}}{2} \tag{4}$$

Figure 1-(left) shows the probability of associating label 1 to an image for which the ground truth is 1 when the difficulty of the tagging task varies, and for different values of recall. The range of task difficulty is $[0, 1]$. The vertical axis is the probability of getting $l_{ij} = 1$. One can notice that this probability takes values between 0.5 and 1. Indeed, if a task is too difficult, then returning a value is close to making a random guess of a binary value. Unsurprisingly, as the difficulty of tasks increases, the probability of correctly annotating the image decreases. This generative function applies for every worker. For a fixed difficulty of task, workers with higher recalls have higher probability to correctly label an image. Also, note that when the difficulty of a task approaches 1, the probability of annotating image as $l_{ij} = 1$ decreases for every worker. However, for workers with high recall, the probability of a correct annotation is always greater than with a smaller recall. Hence, the probability of correct answer depends both on the difficulty of task and on expertise of the worker realizing the task.

3.1 Aggregating Answers

For a given task j, with unknown difficulty d_j, the answers returned by k workers (observed data) is a set $L_j = \{l_{1j}, \ldots, l_{kj}\}$, where l_{ij} is the answer of worker i to task j. In addition, workers expertise are vectors of parameters $\alpha = \{\alpha_1, \ldots \alpha_k\}$ and $\beta = \{\beta_1, \ldots \beta_k\}$ and are also unknown. The goal is to infer the final label y_j, and to derive the most probable values for d_j, α_i, β_i, given the observed answers of workers. We use a standard EM approach to infer the most probable actual answer $Y = \{y_1, \ldots y_n\}$ along with the hidden parameters $\Theta = \{d_j, \alpha_i, \beta_i\}$. Let us consider the E and M phases of the algorithm.

E Step: We assume that all the answers L are independently given by the workers as there is no collaboration between them. So, in every $L_j = \{l_{1j}, \ldots, l_{kj}\}$, l_{ij}'s are independently sampled variables. We compute the posterior probability of $y_j \in \{0, 1\}$ for a given task j given the difficulty of task d_j, worker expertise $\alpha_i, \beta_i, i \leq k$ and the worker answers $L_j = \{l_{ij} \mid i \in 1..k\}$. Using Bayes' theorem, considering a particular value $\lambda \in \{0, 1\}$ we have:

$$Pr[y_j = \lambda | L_j, \alpha, \beta, d_j] = \frac{Pr(L_j | y_j = \lambda, \alpha, \beta, d_j) \cdot Pr(y_j = \lambda | \alpha, \beta, d_j)}{Pr(L_j | \alpha, \beta, d_j)} \quad (5)$$

One can remark that y_j and α, β, d_j are independent variables. We assume that both values of y_j are equiprobable, i.e. $Pr(y_j = 0) = Pr(y_j = 1) = \frac{1}{2}$. We hence get:

$$Pr[y_j = \lambda | L_j, \alpha, \beta, d_j] = \frac{Pr(L_j | y_j = \lambda, \alpha, \beta, d_j) \cdot Pr(y_j = \lambda)}{Pr(L_j | \alpha, \beta, d_j)} = \frac{Pr(L_j | y_j = \lambda, \alpha, \beta, d_j) \cdot \frac{1}{2}}{Pr(L_j | \alpha, \beta, d_j)} \quad (6)$$

Similarly, the probability to obtain a particular set of labels is given by:

$$Pr(L_j \mid \alpha, \beta, d_j) = \frac{1}{2} \cdot Pr(L_j \mid y_j = 0, \alpha, \beta, d_j) + \frac{1}{2} \cdot Pr(L_j \mid y_j = 1, \alpha, \beta, d_j) \quad (7)$$

Overall we obtain:

$$Pr[y_j = \lambda | L_j, \alpha, \beta, d_j] = \frac{Pr(L_j | y_j = \lambda, \alpha, \beta, d_j)}{Pr(L_j | y_j = 0, \alpha, \beta, d_j) + Pr(L_j | y_j = 1 \alpha, \beta, d_j)} \quad (8)$$

Let us consider one of these terms, and let us assume that every l_{ij} in L_j takes a value λ_p . We have

$$Pr(L_j \mid y_j = \lambda, \alpha, \beta, d_j) = \prod_{i=1}^{k} Pr(l_{ij} = \lambda_p \mid \alpha_i, \beta_i, d_j, y_j = \lambda) \tag{9}$$

If $\lambda_p = 0$ then $Pr(l_{ij} = \lambda_p \mid \alpha_i, \beta_i, d_j, y_j = 0)$ is the probability to classify correctly a 0 as 0, as defined in equation 4 denoted by $\delta_{ij} = \frac{1+(1-d_j)^{(1-\beta_i)}}{2}$. Similarly, if $\lambda_p = 1$ then $Pr(l_{ij} = \lambda_p \mid \alpha_i, \beta_i, d_j, y_j = 1)$ is the probability to classify correctly a 1 as 1, expressed in equation 3 and denoted by $\gamma_{ij} = \frac{1+(1-d_j)^{(1-\alpha_i)}}{2}$. Then the probability to classify $y_j = 1$ as $\lambda_p = 0$ is $(1 - \gamma_{ij})$ and the probability to classify $y_j = 1$ as $\lambda_p = 0$ is $(1 - \delta_{ij})$. We hence have $Pr(l_{ij} = \lambda_p \mid \alpha_i, \beta_i, d_j, y_j = 0) = (1 - \lambda_p) \cdot \delta_{ij} + \lambda_p \cdot (1 - \gamma_{ij})$. Similarly, we can write $Pr(l_{ij} = \lambda_p \mid \alpha_i, \beta_i, d_j, y_j = 1) = \lambda_p \cdot \gamma_{ij} + (1 - \lambda_p) \cdot (1 - \delta_{ij})$. So equation 8 rewrites as :

$$\begin{aligned} Pr[y_j = \lambda | L_j, \alpha, \beta, d_j] &= \frac{\prod_{i=1}^{k} Pr(l_{ij} = \lambda_p \mid y_j = \lambda_p), \alpha_i, \beta_i, d_j}{Pr(L_j \mid y_j = 0, \alpha, \beta, d_j) + Pr(L_j \mid y_j = 1, \alpha, \beta, d_j)} \\ &= \frac{\prod_{i=1}^{k} (1 - \lambda_p).[(1 - \lambda_p)\delta_{ij} + \lambda_p(1 - \gamma_{ij})] + \lambda_p.[\lambda_p.\gamma_{ij} + (1 - \lambda_p)(1 - \delta_{ij})]}{Pr(L_j \mid y_j = 0, \alpha, \beta, d_j) + Pr(L_j \mid y_j = 1, \alpha, \beta, d_j)} \\ &= \frac{\prod_{i=1}^{k} (1 - \lambda_p).[(1 - \lambda_p)\delta_{ij} + \lambda_p(1 - \gamma_{ij})] + \lambda_p.[\lambda_p.\gamma_{ij} + (1 - \lambda_p)(1 - \delta_{ij})]}{\prod_{i=1}^{k} (1 - \lambda_p)\delta_{ij} + \lambda_p(1 - \gamma_{ij}) + \prod_{i=1}^{k} \lambda_p.\gamma_{ij} + (1 - \lambda_p)(1 - \delta_{ij})} \end{aligned} \tag{10}$$

In the E step, as every α_i, β_i, d_j is fixed, one can compute $\mathbb{E}[y_j | L_j, \alpha_i, \beta_i, d_j]$ and also choose as final value for y_j the value $\lambda \in \{0, 1\}$ such that $Pr[y_j = \lambda | L_j, \alpha_i, \beta_i, d_j] > Pr[y_j = (1 - \lambda_p) | L_j, \alpha_i, \beta_i, d_j]$. We can also estimate the likelihood for the values of variables $P(L \cup Y \mid \theta)$ for parameters $\theta = \{\alpha, \beta, d\}$, as $Pr(y_j = \lambda, L \mid \theta) = Pr(y_j = \lambda_p, L).Pr(L_j \mid y_j = \lambda_p, \theta) = Pr(y_j = \lambda_p).Pr(L_j \mid y_j = \lambda_p, \theta)$

M Step: With respect to the estimated posterior probabilities of Y computed during the E phase of the algorithm, we compute the parameters θ that maximize $Q(\theta, \theta^t)$. Let θ^t be the value of parameters computed at step t of the algorithm. We use the observed values of L, and the previous expectation for Y. We maximize $Q'(\theta, \theta^t) = \mathbb{E}[logPr(L, Y \mid \theta) \mid L, \theta^t]$ (we refer interested readers to [5]-Chap. 9 and [4] for explanations showing why this is equivalent to maximizing $Q(\theta, \theta^t)$). We can hence compute the next value as: $\theta^{t+1} = \arg\max_{\theta} Q'(\theta, \theta^t)$. Here in our context the values of θ are α_i, β_i, d_j. We maximize $Q'(\theta, \theta^t)$ using bounded optimization techniques, truncated Newton algorithm [10] provided by the standard scipy[1] implementation. We iterate E and M steps, computing at each iteration t the posterior probability and the parameters θ^t that maximize $Q'(\theta, \theta^t)$. The algorithm converges, and stops when the improvement (difference between two successive joint log-likelihood values) is below a threshold, fixed in our case to $1e^{-7}$.

[1] docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

4 Cost Model

A drawback of many crowdsourcing approaches is that task distribution is static, i.e. tasks are distributed to prior fixed number of workers, without considering their difficulty, nor checking if a consensus can be reached with fewer workers. Consider again the simple boolean tagging setting, but where each task realization are paid, and with a fixed total budget B_0 provided by the client. For simplicity, we assume that all workers receive 1 unit of credit for each realized task. Hence, to solve n boolean tagging tasks, one can hire only n/B_0 workers per task. In this section, we show a worker allocation algorithm that builds on collected answers and estimated difficulty to distribute tasks to worker at run time, and show its efficiency w.r.t. other approaches.

Our algorithm works in rounds. At each round, only a subset $T_{avl} \subseteq T$ of the initial tasks remain to be evaluated. We collect labels produced by workers for these tasks. We aggregate answers using the EM approach described in Section 3. We denote by y_j^q as the final aggregated answer for task j at round q, d_j^q is the current *difficulty* of task and α_i^q, β_i^q denotes the estimated *expertise* of a worker i at round q. We let $D^q = \{d_1^q \dots d_j^q\}$ denote the set of all difficulties estimated as round q. We fix a maximal step size $\tau \geq 1$, that is the maximal of workers that can be hired during a round for a particular task. For every task $t_j \in T_{avl}$ with difficulty d_j^q at round q, we allocate $\mathbf{a}_j^q = \left\lceil \left(\frac{d_j^q}{\max D^q} \right) \times \tau \right\rceil$ workers for the next round. Once all answers for a task have been received, the EM aggregation can compute final label $y_j^q \in \{0, 1\}$, difficulty of task d_j^q, expertise of all workers $\alpha_1^q, \dots, \alpha_k^q, \beta_1^q, \dots, \beta_k^q$. Now, it remains to decide whether the confidence in answer y_j^q obtained at round q is sufficient (in which case, we do not allocate workers to this task in the next rounds). Let k_j^q be the number of answers obtained for task j at round q. The *confidence* $\hat{c}_j^q$ in a final label y_j^q is defined as follows:

$$\hat{c}_j^q(y_j^q = 1) = \frac{1}{k_j^q} \cdot \sum_{i=1}^{k_j^q} \left\{ l_{ij} \times \left(\frac{1+(1-d_j^q)^{(1-\alpha_i^q)}}{2}\right) + (1 - l_{ij}) \times \left(1 - \frac{1+(1-d_j^q)^{(1-\alpha_i^q)}}{2}\right) \right\} \quad (11)$$

$$\hat{c}_j^q(y_j^q = 0) = \frac{1}{k_j^q} \cdot \sum_{i=1}^{k_j^q} \left\{ (1 - l_{ij}) \times \left(\frac{1+(1-d_j^q)^{(1-\beta_i^q)}}{2}\right) + (l_{ij}) \times \left(1 - \frac{1+(1-d_j^q)^{(1-\beta_i^q)}}{2}\right) \right\} \quad (12)$$

Intuitively, each worker adds its probability of doing an error, which depends on the final label y_j^q estimated at round q and on his competences, i.e. on the probability to choose $l_{ij} = y_j^q$. Let us now show when to stop the rounds of our evaluation algorithm. We start with n tasks, and let T_{avl} denote the set of remaining tasks at round q. We define $r^q \in [0, 1]$ as the fraction of task that are available at the current instance as compared to total number of task, i.e. $r^q = \frac{|T_{avl}|}{n}$. We start with an initial budget B_0, and denote by B_c^q the total budget consumed at round q. We denote by $\mathcal{B}^q$ the the fraction of budget consumed at that current instance, $\mathcal{B}^q = \frac{B_c^q}{B_0}$. We define the stopping threshold $Th^q \in [0.5, 1.0]$ as $Th^q = \frac{1+(1-\mathcal{B}^q)^{r^q}}{2}$.

The intuition behind this function is simple: when the number of remaining tasks decreases, one can afford a highest confidence threshold. Similarly, as the budget decreases, one shall derive a final answer for tasks faster, possibly with a poor confidence, as the remaining budget does not allow hiring many workers. Figure 1-(right) shows the different threshold based on the current estimate of the budget on the horizontal axis. Each line depicts the corresponding fraction of task available in the considered round. Observe that when r^q approaches 1, the threshold value falls rapidly, as large number of tasks remain without definite final answer, and have to be evaluated with the remaining budget. On the other hand, when there are less tasks (e.g. when $r^q = 0.10$), the threshold Th^q decreases slowly.

We can now define a crowdsourcing algorithm (Crowdinc) with a dynamic worker allocation strategy to optimize cost and accuracy. This strategy allocates workers depending on current confidence on final answers, and available resources. Crowdinc is decomposed in two phases, *Estimation* and *Convergence*.

Estimation: As *difficulty* of tasks is not known a priori, the first challenge is to estimate it. To get an a priori measure of difficulties, each task needs to be answered by a set of workers. Now, as each worker receives an incentive for a task, this preliminary evaluation has a cost, and finding an optimal number of workers for *difficulty* estimation is a fundamental issue. The initial budget gives some flexibility in the choice of an appropriate number of workers for preliminary evaluation of difficulty. Choosing a random number of workers per task does not seem a wise choice. We choose to devote a fraction of the initial budget to this estimation phase. We devote one third of the total budget ($B_0/3$) to the estimation phase. It leaves a sufficient budget ($2B_0/3$) for the convergence phase. Experiments in the next Section show that this seems a sensible choice. After collection of answers for each task, we apply the EM based aggregation technique of Section 3 to estimate the *difficulty* of each task as well as the *expertise* of each worker. Considering this as an initial round $q = 0$, we let d_j^0 denote the initially estimated difficulty of each task j, and α_i^0, β_i^0 denote the expertise of each worker and y_j^0 denote the final aggregated answer. Note that if the difficulty of some tasks is available a priori and is provided by the client, we may skip the estimation step. However, in general clients do not posses such information and this initial step is crucial in estimation of parameters. After this initial estimation, one can already compute Th^0 and decide to stop evaluation of task with a sufficient confidence level.

Convergence: The *difficulty* of task d_j^q and the set of remaining tasks T_{avl} are used to start the convergence phase. Now as the difficulty of each task is estimated, we can use the estimated difficulty d_j^q to allocate the workers dynamically. The number of workers allocated at round $q > 0$ follows a difficulty aware *worker allocation* policy. At each round, we allocate $\mathbf{a}_j^q$ workers to remaining task t_j. This allocation policy guarantees that each remaining task is allocated at least one worker, at most τ workers, and that the tasks which are more difficulty (i.e. have the more disagreement) are allocated more workers than easier tasks.

Algorithm 2: CrowdInc

Data: A set of tasks $T = \{t_1, \ldots, t_n\}$, a budget = B_0
Result: Final Answer: $Y = y_1, \ldots, y_n$, Difficulty: d_j,Expertise: α_i, β_i

Initialization : Set every d_j, α_i, β_i to a random value in $[0, 1]$.
$T_{avl} = T$; $q = 0$; $B = B - (B_0/3)$; $B_c = B_0/3$; $r = (B_0/3)/n$
`//Initial Estimation:`
Allocate r workers to each task in T_{avl} and get their answers
Estimate $d_j^q, \alpha_i^q, \beta_i^q, \hat{c}_j^q, 1 \leq j \leq n, 1 \leq i \leq B_0/3$ using EM aggregation
Compute the stopping threshold Th^q.
for $j = 1, \ldots, n$ **do**
  if $\hat{c}_j^q > Th^q$ **then** $T_{avl} = T \setminus \{j\}$;
end
`//Convergence:`
while $(B > 0)$ && $(T_{avl} \neq \emptyset)$ **do**
  $q = q + 1$; $l = |T_{avl}|$
  Allocate $\mathbf{a}_1^q, \ldots, \mathbf{a}_l^q$ workers to tasks $t_1, \ldots t_l$ based on difficulty.
  Get the corresponding answers by all the newly allocated workers.
  Estimate $d_j^q, \alpha_i^q, \beta_i^q, \hat{c}_j^q$ using aggregation model.
  $B = B - \sum_{i \in 1..|T_{avl}|} \mathbf{a}_i^q$
  Compute the stopping threshold Th^q
  for $j = 1, \ldots, n$ **do**
    if $\hat{c}_j^q > Th^q$ **then** $T_{avl} = T_{avl} \setminus \{j\}$;
  end
end

The complete process of Crowdinc is given in Algorithm 2. It starts with the *Estimation* phase and allocates workers for an initial evaluation round ($q = 0$). After collection of answers, and then at each round $q > 0$, we first apply EM based aggregation to estimate the difficulty d_j^q of each of task $t_j \in T_{avl}$, the confidence $\hat{c}_j^q$ in final aggregated answer y_j^q, and the expertise α_i^q, β_i^q of the workers. Then, we use the stopping threshold to decide whether we need more answers for a task. If $\hat{c}_j^q$ is greater than Th^q, the task t_j is removed from T_{avl}. This stopping criterion hence takes a decision based on the confidence in the final answers for a task and on the remaining budget. Once solved tasks have been removed, we allocate $\mathbf{a}_j^q$ workers to each remaining task t_j in T_{avl} following our difficulty aware policy. The algorithm stops when either all budget is exhausted or there is no additional task left. It returns the aggregated answers for all tasks.

5 Experiments

We evaluate the algorithm on three public available dataset, namely the product identification [16], duck identification [17] and Sentiment Analysis [11] benchmarks. We briefly detail each dataset and the corresponding tagging tasks. All tags appearing in the benchmarks were collected via Amazon Mechanical Turk.

In the **Product Identification** use case, workers were asked to decide whether a *product-name* and a *description* refer to the same product. The answer returned is *True* or *False*. There are 8315 samples and each of them was evaluated by 3 workers. The total number of unique workers is 176 and the total number of answers available is 24945. In the **Duck Identification** use case, workers had to decide if sample images contain a duck. The total number of tasks is 108 and each of task was allocated to 39 workers. The total number of unique worker is 39 and the total number of answers is 4212. In the **Sentiment Popularity** use case, workers had to annotate movie reviews as Positive or Negative opinions. The total number of tasks was 500. Each task was given to 20 unique workers and a total number of 143 workers were involved, resulting in a total number of 10000 answers. All these information are synthesized in table 1.

Dataset	Number of Tasks	Number of tasks with ground truth	Total Number of answers provided by crowd	Average number of answers for each task	Number of unique crowd workers
Product Identification	8315	8315	24945	3	176
Duck Identification	108	108	4212	39	39
Sentiment Popularity	500	500	10000	20	143

Table 1: Datasets description.

Methods	Recall	Specificity	Balanced Accuracy
MV	0.56	0.91	0.73
D&S [2]	0.81	0.93	0.87
GLAD [18]	0.47	0.98	0.73
PMCRH [9]	0.58	0.95	0.76
LFC [13]	0.87	0.91	0.89
ZenCrowd [3]	0.39	0.98	0.68
EM + recall, specificity & difficulty	**0.89**	**0.91**	**0.90**

(a) Duck Identification

Methods	Recall	Specificity	Balanced Accuracy
MV	0.61	0.93	0.77
D&S [2]	0.65	0.97	0.81
GLAD [18]	0.48	0.98	0.73
PMCRH [9]	0.61	0.93	0.77
LFC [13]	0.64	0.97	0.81
ZenCrowd [3]	0.51	0.98	0.75
EM + recall, specificity & difficulty	**0.77**	**0.90**	**0.83**

(b) Product Identification

Methods	Recall	Specificity	Balanced Accuracy
MV	0.93	0.94	0.4
D&S [2]	0.94	0.94	0.94
GLAD [18]	0.94	0.94	0.94
PMCRH [9]	0.93	0.95	0.94
LFC [13]	0.94	0.94	0.94
ZenCrowd [3]	0.94	0.94	0.94
EM + recall, specificity & difficulty	**0.94**	**0.95**	**0.94**

(c) Sentiment Popularity

Table 2: Comparison of EM + aggregation (with Recall, specificity & task difficulty) w.r.t MV, D&S, GLAD, PMCRH, LFC, ZenCrowd.

Evaluation of aggregation: We first compared our aggregation technique to several methods available in the literature: MV, D&S [2], GLAD [18], PMCRH [9], LFC [13], and ZenCrowd [3]. We ran the experiment 30 times with different initial values and found that the aggregation is insensitive to initial prior values. The standard deviation over all the iteration was less than 0.05%. We compare *Recall*, *Specificity* and *Balanced Accuracy* of all methods. Balanced Accuracy is the average of recall and specificity. We can observe in table 2 that our method outperforms other techniques in Duck Identification, Product Identification, and is comparable for Sentiment Popularity.

Evaluation of Crowdinc: The goal of the next experiment was to verify that the cost model proposed in Crowdinc achieves at least the same accuracy but with a smaller budget. We have used Duck identification and Sentiment popularity for this test. We did not consider the Product Identification benchmark: indeed, as shown in table 1, the Product Identification associates only 3 answers

to each task. This does not allow for a significant experiment with Crowdinc. We compared the performance of Crowdinc to other approaches in terms of cost and accuracy. The results are given in Figure 2. Static(MV) denotes the traditional crowdsourcing platforms with majority voting as aggregation technique and Static(EM) shows more advanced aggregation technique with EM based aggregation technique. Both algorithms allocate all the workers (and hence use all their budget) at the beginning of crowdsourcing process.

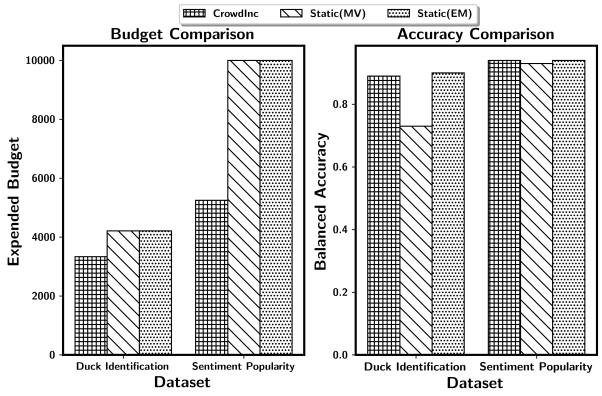

Fig. 2: Comparison of cost vs. Accuracy.

The following observation can be made from figure 2. First, *CrowdInc* achieves better accuracy than a static(MV) approach. This is not a real surprise, as MV already showed bad accuracy in table 2. Then, *CrowdInc* achieves almost the same accuracy as a Static(EM) based approach in Duck identification, and the same accuracy in Sentiment Popularity. Last, *CrowdInc* uses a smaller budget than static approaches in all cases.

Table 3 shows the time (in seconds) needed by each algorithm to aggregate answers. Static(MV) is the fastest solution: it is not surprising, as the complexity is linear in the number of answers. We recall however that MV has the worst accuracy of all tested aggregation techniques. We have tested aggregation with EM when the number of workers is fixed a priori and is the same for all tasks (Static(EM)). *CrowdInc* uses EM, but on a dynamic sets of workers and tasks, stopping easiest tasks first. This results in a longer calculus, as EM is used several times on sets of answers of growing sizes. The accuracy of *static(EM)* and *CrowdInc* are almost the same. Aggregation with *CrowdInc* takes approximately 11% longer than *static(EM)* but for a smaller budget, as shown in the Figure 2. To summarize the *CrowdInc* aggregation needs more time and a smaller budget to aggregate answers with a comparable accuracy. In general, clients using crowdsourcing services can wait several days to see their task completed. Hence, when time is not a major concern *CrowdInc* can reduce the cost of crowdsourcing.

Dataset/Methods	**CrowdInc**	**Static(EM)**	**Static(MV)**
Duck Identification	843.26	106.81	0.073
Sentiment Popularity	1323.35	137.79	0.102

Table 3: Running time(in seconds) of Crowdinc, MV and Static EM.

6 Conclusion and discussions

In this paper, we have introduced a new aggregation technique for crowdsourcing platforms. Aggregation is based on expectation maximization and jointly estimates the answers, the difficulty of tasks, and the expertise of workers. Using difficulty and expertise as latent variables improves the accuracy of aggregation in terms of recall and specificity. We also proposed *CrowdInc* an incremental labeling technique that optimizes the cost of answers collection. The algorithm implements a worker allocation policy that takes decisions from a dynamic threshold computed at each round, which helps achieving a trade off between cost and accuracy. We showed in experiments that our aggregation technique outperforms the existing state-of-the-art techniques. We also showed that our incremental crowdsourcing approach achieves the same accuracy as traditional solutions such as majority voting at lower costs.

The ideas proposed in this paper can lead to several improvements that will be considered in future work. In the paper, we addressed binary tasks for simplicity, but the approach can be easily extended to tasks with a finite number m of answers. The difficulty of each task remains a parameter d_j. Expertise is the ability to classify a task as m when its ground truth is m. An EM algorithm just has to consider probabilities of the form $Pr(L_{ij} = m|y_j = m)$ to derive hidden parameters and final labels for each task. Another easy improvement is to consider incentives that depend on workers characteristics. This can be done with a slight adaptation of costs in the CrowdInc algorithm. Another possible improvement is to try to hire experts when the synthesized difficulty of a task is high, to avoid hiring numerous workers or increase the number of rounds.

Last, we think that the complexity of CrowdInc can be improved. The complexity of each E-step of the aggregation is linear in the number of answers. The M-step maximizes the log likelihood with an iterative process (truncated Newton algorithm). However, the E and M steps have to be repeated many times. The cost of this iteration can be seen in table 3, where one clearly see the difference between a linear approach such as Majority Voting (third column), a single round of EM (second column), and Crowdinc. Using Crowdinc to reduce costs results in an increased duration to compute final answers. Indeed, the calculus performed at round i to compute hidden variables for a task t is lost at step $i+1$ if t is not stopped. An interesting idea is to consider how a part of computations can be reused from a round to the next one to speed up convergence.

References

1. F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys*, 51(1):7, 2018.
2. A.Ph. Dawid and A.M. Skene. Maximum likelihood estimation of observer errorrates using the em algorithm. *J. of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

3. G. Demartini, D.E. Difallah, and Ph. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proc. of WWW 2012*, pages 469–478. ACM, 2012.
4. A.P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
5. P.A. Flach. *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.
6. H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, and V. Verroios. Challenges in data crowdsourcing. *Trans. on Knowledge and Data Engineering*, 28(4):901–911, 2016.
7. D.R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *Proc. of NIPS'11*, pages 1953–1961, 2011.
8. J. Le, A. Edmonds, V. Hester, and L. Biewald. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR 2010 workshop on crowdsourcing for search evaluation*, volume 2126, pages 22–32, 2010.
9. Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proc. of SIGMOD'14*, pages 1187–1198. ACM, 2014.
10. S. G. Nash. Newton-type minimization via the lanczos method. *SIAM J. on Numerical Analysis*, 21(4):770–788, 1984.
11. B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.
12. V. Raykar and P. Agrawal. Sequential crowdsourced labeling as an epsilon-greedy exploration in a markov decision process. In *Artificial intelligence and statistics*, pages 832–840, 2014.
13. V. C. Raykar, S. Yu, L.H. Zhao, G.H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. of Machine Learning Research*, 11(Apr):1297–1322, 2010.
14. L. Tran-Thanh, M. Venanzi, A. Rogers, and N.R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proc. of AAMAS'13*, pages 901–908. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
15. M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *Proc. of WWW'14*, pages 155–164. ACM, 2014.
16. J. Wang, T. Kraska, M.J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *Proc. of the VLDB Endowment*, 5(11):1483–1494, 2012.
17. P. Welinder, S. Branson, P. Perona, and S.J. Belongie. The multidimensional wisdom of crowds. In *Proc. Of NIPS'10*, pages 2424–2432, 2010.
18. J. Whitehill, T. Wu, J. Bergsma, J.R. Movellan, and P.L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. of NIPS'09*, pages 2035–2043, 2009.
19. Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proc. of the VLDB Endowment*, 10(5):541–552, 2017.