Accurate inference of crowdsourcing properties when using efficient allocation strategies

Abigail Hotaling[1,2] and James P. Bagrow[1,2,*]

[1]Department of Mathematics & Statistics, University of Vermont, Burlington, VT, United States
[2]Vermont Complex Systems Center, University of Vermont, Burlington, VT, United States
[*]Corresponding author. Email: james.bagrow@uvm.edu, Homepage: bagrow.com

March 7, 2019

Abstract Allocation strategies improve the efficiency of crowdsourcing by decreasing the work needed to complete individual tasks accurately. However, these algorithms introduce bias by preferentially allocating workers onto easy tasks, leading to sets of completed tasks that are no longer representative of all tasks. This bias challenges inference of problem-wide properties such as typical task difficulty or crowd properties such as worker completion times, important information that goes beyond the crowd responses themselves. Here we study inference about problem properties when using an allocation algorithm to improve crowd efficiency. We introduce Decision-Explicit Probability Sampling (DEPS), a method to perform inference of problem properties while accounting for the potential bias introduced by an allocation strategy. Experiments on real and synthetic crowdsourcing data show that DEPS outperforms baseline inference methods while still leveraging the efficiency gains of the allocation method. The ability to perform accurate inference of general properties when using non-representative data allows crowdsourcers to extract more knowledge out of a given crowdsourced dataset.

Keywords— algorithmic crowdsourcing; data efficiency; efficient data collection; data bias; Dawid-Skene; statistical inference; informative priors

1 Introduction

Crowdsourcing has become a valuable source of information for a wide variety of applications [1, 2, 3], particularly in areas where humans can perform work not possible or not well suited to computational methods. Common crowdsourcing tasks include generating labeled training data for machine learning algorithms [4, 5, 6], performing text recognition or natural language tasks [4], completing surveys [7], or generating novel questions or other creative inputs [8, 9, 10, 11, 12]. However, using human volunteers or paid crowd workers brings with it a host of new problems [3, 13]. Human workers are often more costly than computational methods, receiving responses from workers can be relatively slow, and one must worry about worker motivation [14, 15] and reliability [16, 3]. Even with reliable workers, challenging tasks often require a consensus approach: multiple workers are given a task and their set of (possibly noisy) responses are aggregated to generate the final response for the crowdsourcer [17].

Algorithmic crowdsourcing focuses on computational approaches to dealing with issues introduced when using a crowdsourcing solution [18, 19, 20, 21]. These algorithms improve the crowd's efficiency by enabling more accurate information gain, and often reduce the numbers of workers needed. Some methods learn the reliability of workers and minimize or eliminate the contributions of unreliable workers [22, 18, 16]. Allocation methods, on the other hand, focus on the tasks being crowdsourced, and attempt to allocate workers towards those tasks that can be completed most quickly and accurately [19, 21]. Quickly identifying those tasks where workers tend to give the same response can prevent redundant worker responses and allows the crowdsourcer to decide upon the aggregate response more efficiently.

The focus of crowd allocation algorithms has been on maximizing accuracy when aggregating responses from the crowd while minimizing the number of responses needed in order to meet budget constraints. However, other aspects of a crowdsourcing problem are important to understand besides the final responses to the set of tasks comprising that problem. For example, a crowdsourcer may wish to identify experts within the crowd or understand how many tasks are difficult for workers to complete compared with tasks that are easy for workers. Or a crowdsourcer concerned about latency may want to understand how much time it takes workers to perform a task [23, 12]. Worker completion times are an example of the more general behavioral traces of workers [24], and a crowdsourcer may be interested in understanding the pattern of worker dynamics as they perform work. Yet in all these cases, an allocation algorithm focused on completing tasks accurately can change the set of tasks being shown to workers, introducing a bias where easily completed tasks are more likely to receive responses from workers and leaving the crowdsourcer with a dataset that may be unrepresentative of the typical tasks or typical worker behaviour within the crowdsourcing problem.

In this work, we ask if it is possible to reason about properties of crowdsourcing problems while also employing an efficient allocation algorithm. Of course, one can provide an unbiased view of tasks and workers by assigning problems at random, but doing so sacrifices the efficiency gains an allocation strategy can provide; it may simply not be worth the added cost to learn these problem properties, especially if developing accurate responses is the main goal. We introduce *Decision-Explicit Probability Sampling* (DEPS), a strategy to perform inference on problem properties that places the decision process of an efficient allocation algorithm within the inference model. DEPS can provide inference for a variety of problem properties, although our focus here is on inferring the distribution of problem difficulty, information a crowdsourcer often desires. Experiments with real crowdsourcing data and synthetic models show that DEPS provides accurate information about the crowdsourcing problem while using an allocation algorithm, allowing a crowdsourcer to leverage the efficiency gains of an allocation algorithm while retaining more of the problem information that can be lost when using an allocation algorithm.

The rest of this paper is organized as follows. In Sec. 2 we describe a model for a crowdsourcing problem with a fixed budget, discuss the properties of crowdsourcing problems, and detail allocations algorithms that have been introduced to improve crowdsourcing efficiency. Section 3 illustrates the challenge of property inference when using efficient allocation strategies. Section 4 introduces Decision-Explicit Probability Sampling (DEPS), a strategy for integrating the results of an allocation algorithm into property inference. Then, in Secs. 5 and 6 we describe and present experiments using DEPS on real and synthetic crowdsourcing data. We conclude with a discussion in Sec. 7.

2 Background

Here we describe the model of a crowdsourcing problem along with various problem properties that are meaningful to study for such a problem. We also provide background on efficient allocation algorithms a crowdsourcer can use to maximize the information gained from the crowd. Crowdsourcing problem models can be used to generate synthetic crowdsourced data (as a generative model), but the primary use of the model is to build inference procedures to be used when processing real crowdsourced data (as a statistical model).

2.1 Crowdsourcing model

As is standard practice, we employ the Dawid-Skene model for a crowdsourcing problem [22]. A crowdsourcing problem consists N tasks, each of which is considered a binary labeling task. Binary tasks can represent image classification tasks, survey questions, and so forth, and while relatively simplistic, binary labeling forms the basis of most crowdsourcing models. The binary task model can generalize to categorical tasks as well, but such tasks can always be binarized by treating the most common individual response as a '1' and all other responses as '0'. Let $z_i \in \{0, 1\}$ be the true (unknown) label for task. A total of M workers are given one or more of these tasks and respond by providing a label for the given task. Let $y_{ij} \in \{0, 1\}$ be the response of worker $j \in [1, M]$ to task $i \in [1, N]$, a_i the total number of 1-labels for task i, b_i the total number of 0-labels, and $n_i = a_i + b_i$ the total number of responses to i. Individual worker responses are taken as iid for a given task. Not all workers necessarily respond to every task, so define J_i as the set of workers who responded to task i, and $\left|\cup_{i=1}^{N} J_i\right| = M$ (we assume every worker responds to at least one task and no worker responds to the same task more than once). Likewise, let I_j be the set of tasks that worker j responded to, with $|\cup_{j=1}^{M} I_j| = N$. The crowdsourcer can then aggregate the individual responses to reach a consensus response. The goal is to infer best estimates $\hat{z}_i \approx z_i$ given worker responses y_{ij}. Often the crowdsourcer must approach this goal under budget constraints, as costs are incurred when employing a crowd. Let T be a crowdsourcer's total budget, meaning the crowdsourcer can request at most T individual worker responses. This constraint leads to $\sum_i n_i \leq T$. In seminal work, Dawid and Skene use an EM-algorithm to infer z_i that outperforms the basic majority-vote strategy [22] and much subsequent work has studied and generalized this approach [4, 17].

2.2 Problem properties

Learning labels accurately is the goal of the crowdsourced labeling problem described above. However, there are potentially many other properties of such problems that are of interest to researchers. For example, what constitutes a typical task within the problem? Do tasks tend to be easy for workers to complete or are tasks difficult? Are there distinguishing features of tasks, such as groups or categories of tasks that are similar in some way? Likewise, a crowdsourcer may wish to learn properties of the crowd workers. Are there experts among the workers? Are workers reliable or not? How quickly can workers complete a task? Can we learn about worker behavioral traces [24], the patterns of activities the workers undertake as they respond to a task?

In general, learning both task properties and crowd properties can be of significant benefit to a crowdsourcer: tasks can be better designed accounting for features of the tasks and how workers interact with them, costs can be better forecast and planned, and more reliable data can be generated by the crowd. Of course, the details and importance of these properties will be heavily dependent on the type of crowdsourcing problem, but learning properties can potentially reveal significant information for the crowdsourcer, beyond that of the task labels themselves.

2.3 Efficient allocation methods

Various allocation strategies have been developed with the goal of assigning tasks to workers in order to maximize information gained about labels while minimizing costs [21, 25]. Here we provide background on how these algorithms assign tasks.

Requallo [21] is a flexible allocation framework that addresses the challenge of efficient crowdsourcing for labeling tasks. With Requallo, crowdsourcers (1) define completion requirements that tasks must satisfy to be deemed completed, (2) apply a task allocation strategy to maximize the number of completed tasks within a given budget T (number of worker responses). When a task i is completed the crowdsourcer will cease seeking new responses for i and be able to decide the final estimated label $\hat{z}_i$. There are many ways to define a completion requirement. For example, a ratio requirement for a binary labeling task i would determine that i is complete when one label is assigned by workers sufficiently more often than the other label: a ratio requirement of c determines task i to be complete if either $a_i/b_i > c$ or $b_i/a_i > c$, where a_i and b_i are the number of 1-labels and 0-labels given by workers for task i, respectively. (In practice, the Requallo authors use smoothed counts $a_i + 1$ and $b_i + 1$ for their requirement.) Using the requirement, let $C(t) \subseteq [1, \ldots, N]$ be the set of completed tasks after receiving $t \leq T$ total responses from workers. The Requallo requirement implements an early stopping rule while ensuring there is sufficient information to decide whether $z_i = 0$ or $z_i = 1$ with some degree of accuracy, with uncompleted tasks being undecided.

With a completeness requirement in place, Requallo then allocates tasks to workers using a Markov Decision Process (MDP) designed to quickly identify and distribute to workers those tasks which can be completed and to identify and avoid distributing those tasks which are unlikely to be completed. Due to computational complexity, the Requallo framework determines a policy for allocating tasks using a one-step look-ahead greedy method; see Li *et al.* [21] for full details of the Requallo MDP reward function and optimization method. Requallo's completeness requirement combined with its allocation strategy gives significant efficiency gains, with more tasks completed accurately using a fixed budget of worker responses than other allocation methods.

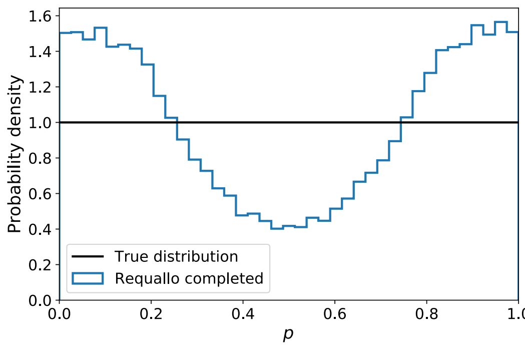
However, the completeness and allocation components of Requallo also introduce bias in the collected data. Crucially, Requallo stops allocating tasks that reach completeness and the greedy MDP optimization can lead to hard tasks (those unlikely to reach completeness) being ignored in favor of easy tasks (those likely to reach completeness). These properties together greatly contribute to the bias introduced by Requallo, which may be especially harmful when a crowdsourcer wants to study properties of tasks and/or workers beyond the resulting labels themselves.

Not all allocation methods explicitly distinguish decided and undecided tasks. Opt-KG [25], for example, works to optimize accuracy among the set of tasks by allocating to the appropriate tasks in order to optimize the accuracy across all tasks. Once the algorithm has determined a sufficiently accurate label for easier tasks, it directs the budget towards tasks that are less accurate (hard tasks). This equates to a similar bias as the Requallo framework, where less information is obtained about easy tasks, even without an explicit decision process for those tasks. Of course, any such allocation method can always be augmented with a decision process by including a completion requirement, and deciding if tasks are completed based on that requirement.

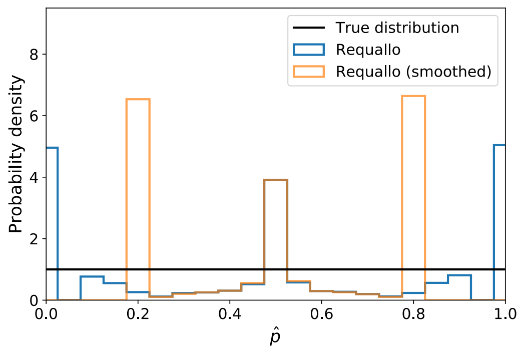
3 Allocation methods lead to non-representative sets of tasks

Before introducing our method for inferring problem-wide properties of a crowdsourcing problem, we first illustrate the challenge introduced when a crowdsourcer employs an efficient allocation algorithm. Of course, this challenge can be eliminated by forgoing use of an allocation algorithm, but doing so may be too cost prohibitive to consider in practice.

We applied the Requallo allocation algorithm to a crowdsourcing problem of $N = 1000$ tasks averaged over 100 independent simulations defined as follows. Each synthetic task i is represented by a Bernoulli RV parameterized by



(a) Distribution of p for completed tasks



(b) Distribution of $\hat{p}$ for completed and uncompleted tasks

Figure 1: Requallo Allocation leads to non-representative sets of completed tasks, and biased information about problem properties. (**a**) Easy tasks (with p far from 1/2) are over-represented in the set of completed tasks while hard tasks (with p near 1/2) are under-represented. (**b**) Estimates of p are biased, when looking at easy tasks there are pileups at $p = (0, 1)$ and $p = (0.2, 0.8)$ for the "smoothed" distribution, looking at hard tasks, there are pileups at $p = 0.5$.

p_i, the probability of a 1-label. Here, tasks closer to $p_i = 1/2$ are more difficult than tasks far from $1/2$, because a task with $p \approx 1/2$ will take many worker responses before we can accurately conclude whether $z_i = 1$ or 0. We took the prior distribution $\Pr(p_i)$ of p_i to be uniform, $p_i \sim U[0, 1]$. Knowing the true distribution of task difficulty $\Pr(p)$ in this situation allows us to compare with the observed distribution developed by employing the Requallo Algorithm with completeness ratio requirement $c = 4$ (Sec. 2.3).

Figure 1 compares the results a crowdsourcer using the Requallo allocation algorithm would receive with the underlying distribution of the problem. In Fig. 1(a) we show the difficulties of tasks deemed by Requallo to be completed versus all tasks: $\Pr(p_i \mid i \in C)$ versus $\Pr(p_i)$. There is a significant bias in these distributions with Requallo-completed tasks containing far more easy tasks and far fewer difficult tasks than the overall distribution would imply. Of course, this is an example of Requallo working by design: a net consequence of efficient allocation (and in particular the early stopping rule introduced by the completeness requirement) will always be more easy tasks than hard tasks completed. Yet it is evident that we are unable to capture the underlying uniform distribution with this method.

Similarly, in Fig. 1(b) we examine the distributions of estimates of problem difficulty $\hat{p}$ for both completed and uncompleted tasks. Examining $\Pr(\hat{p})$ alongside $\Pr(p)$ is important as a crowdsourcer in practice will not have access to the true parameters p_i and must instead infer them from the worker responses. Further, a crowdsourcer must often deal with small-data issues as n_i may be small for a given task i. An example of this is when $a_i = 0$ or $a_i = n_i$, leading to $\hat{p} = 0$ or $\hat{p} = 1$; many inference procedures can fail when no counts for a given category are observed. A common solution to this problem is to use a ***smoothed*** estimate of $\hat{p}$, where a_i, b_i, and n_i are replaced with $a_i + \epsilon$, $b_i + \epsilon$, $n_i + 2\epsilon$, respectively (we use $\epsilon = 1$ as did the Requallo authors [21]). Examining the distributions in Fig. 1(b), we see a significant bias in the smoothed $\Pr(\hat{p})$, with a trimodal sample despite the true underlying distribution being uniform. There is a "pileup" of values at $\hat{p} = 1/2$ alongside pileups at $\hat{p} = 1/2 \pm \frac{c-1}{2(c+1)}$, where $\frac{c-1}{2(c+1)}$ parameterize the Requallo requirements. The central pileup corresponds to tasks underexplored by the crowd because Requallo deemed them too difficult and instead allocated workers to other tasks. Those other tasks, meanwhile, tend to pile up at $1/2 \pm \frac{c-1}{2(c+1)}$ because those ratios correspond to the point where Requallo decides the task is complete and ceasing allocating workers to the task, preventing us from determining if the parameter p is either more or less extreme than the estimate $\hat{p}$. These phenomena show how Requallo is actually working well in terms of avoiding redundant worker responses, but also emphasizes the bias introduced by Requallo: hard tasks are underexplored, but easy tasks are also underexplored.

4 Decision-Explicit Probability Sampling (DEPS)

Suppose we wish to learn about the properties x of a crowdsourcing problem, such as the distribution of problem difficulty p. Accurately estimating the distribution of p may be helpful for understanding the types of labeling tasks we ask workers to perform, for categorizing tasks into different groups, and for forecasting budgetary expenses such as total cost to perform typical tasks or total time we expect to wait for workers to complete their assignments. Given sufficient data, the distribution of p can be estimated relatively quickly. Let n_i be the total number of responses to binary labeling task i and a_i be the number of 1-labels given by workers. Then $\hat{p}_i = a_i/n_i$ is the MLE (point) estimate for p_i. Aggregating many estimates $\hat{p}$ can then be used to infer the distribution $P\left(p \mid \{y_{ij}\}\right) \approx P(\hat{p})$.

Unfortunately, this estimation strategy will be data-intensive in practice, leading to costly crowdsourcing simply to infer the overall distribution of p. Efficient algorithms are commonly introduced to "speed up" crowdsourcing by allowing more accurate labels to be determined with fewer workers. However, as we saw in Sec. 3 and Fig. 1, these methods work at the expense of information about p: by focusing crowd resources on the easiest-to-complete tasks, the crowdsourcer is left with incomplete and unrepresentative data about the overall distribution of, in this case, p. We ask in this work if it is possible to overcome the bias introduced by an efficient allocation algorithm, allowing one to both generate labels efficiently and still perform accurate inference of problem-wide features.

We propose Decision-Explicit Probability Sampling (DEPS), a method to perform inference of the problem distribution for a property x when using crowdsourced data gathered by an allocation method. The allocation method will introduce some form of bias into the estimates $\hat{x}$, so our main focus is introducing a means to reason about the unbiased distribution of x given the observed, biased distribution of $\hat{x}$. DEPS works by explicitly incorporating the decision process of the efficient algorithm into the inferential model.

DEPS proceeds in two phases. The first is developing an inferential model for x that incorporates the allocation method's decisions and the resulting bias, leading to a corrected variable $\tilde{x}$. Next, a probability model is fit to $\tilde{x}$ to infer the distribution $P(x)$ of property x. To begin, notice that a completed task i may be decided to have label $\hat{z}_i = 0$ or label $\hat{z}_i = 1$, depending on the $\{y_{ij} \mid j \in J_i\}$, otherwise a task is undecided. Let the decision indicator $d_i(t)$ be the crowdsourcer's decision for task i after receiving a total of $t \leq T$ responses (across all tasks):

$$d_i(t) = \begin{cases} -1 & \text{if } i \text{ decided } z_i = 0, \\ 0 & \text{if } i \text{ undecided}, \\ 1 & \text{if } i \text{ decided } z_i = 1. \end{cases} \tag{1}$$

with the set of completed tasks $C(t) = \{i \in [1, N] \mid d_i(t) \neq 0\}$. DEPS incorporates the decision variable into our inference model as follows. For each x_i, we determine a posterior distribution using

$$P(x_i \mid \{y_{ij}\}, d_i) \propto \prod_j P(y_{ij} \mid x_i) P(x_i \mid d_i), \tag{2}$$

We incorporate the decision d_i into the prior $P(x_i \mid d_i)$ for x_i, choosing a different prior distribution for each decision status. In other words, we use the prior distributions $x_i \mid d_i \sim D_{d_i}(\Phi_{d_i})$, where D_d is the prior distribution associated with decision d and parameterized by Φ_{d_i}. Depending on the decision of task i and the features of property x, this prior can be used to reflect the mechanism of the efficient algorithm. Specifically, once $d_i \neq 0$, the algorithm no longer allocates budget to that task which creates truncation in the distribution of tasks and information about x is lacking. If the efficient budget allocation algorithm is performing well, we can reason that the tasks that are considered complete could have a different distribution for x than what $\hat{x}$, estimated from the truncated crowd data, tells us. The priors D_d can reflect our understanding of how x may be affected by the allocation algorithm. Next, to develop a single probability distribution for x, DEPS continues by generating a sample $\tilde{x}$ to serve as a debiased x (as opposed to, for example, working with a mixture of N per-task distributions): for each task $i \in [1, \ldots, N]$, sample $\tilde{x}_i$ from the distribution $P(x_i \mid y_{ij}, d_i)$. Inference on these aggregated $\{\tilde{x}\}$ is then performed using standard techniques such as maximum likelihood estimation (MLE) or the method of moments. For MLE, for example, parameters θ for $P(\tilde{x})$ are determined by maximizing the joint log-likelihood $\ln \mathcal{L}(\theta \mid \{\tilde{x}\})$ of the parameters given the data.

As a concrete application of DEPS, we now focus on the property of problem difficulty discussed above (where now $x = p$); we discuss other properties in Sec. 4.2. An allocation algorithm will introduce bias into the data, making the point estimates $\hat{p}_i = a_i / n_i$ unreliable (Fig. 1(b)). For problem difficulty p, the beta distribution is the natural choice of prior: $p_i \mid d_i \sim \text{Beta}(\alpha_{d_i}, \beta_{d_i})$ with parameters $\Phi_{d_i} = (\alpha_{d_i}, \beta_{d_i})$. Worker responses are modeled $y_{ij} \sim \text{Bernoulli}(p_i)$, then the posterior for p_i (Eq. 2) will also follow a beta distribution with parameters $a_i + \alpha_{d_i}$ and $b_i + \beta_{d_i}$, where a_i

and b_i are the number of 1 and 0 labels, respectively, for task i. The choice of hyperparameters α_d and β_d gives the researcher flexibility in how they incorporate the efficient algorithm's decision into the inference model. For example, a beta prior that is skewed towards $p = 0$ could be used for tasks that have $d_i = -1$ with another beta skewed towards $p = 1$ used for $d_i = 1$. The amount of skewness of the $d \neq 0$ priors can then reflect how confident the researcher is in the allocation algorithm's decision. And tasks that are undecided will have priors that provide more weight near $p = 1/2$. We illustrate two choices of decision priors in Fig. 2 and we discuss a data-driven approach to calibrating these priors in Sec. 4.1.

Having generated a sample $\{\tilde{p}\}$, inference is performed using standard maximum likelihood estimates. The joint log-likelihood for the parameters of (in this case) a beta distribution given the data are

$$\begin{aligned}\ln \mathcal{L}(\alpha, \beta \mid \{\tilde{p}\}) &= \sum_{i=1}^{N} \ln \mathcal{L}_i(\alpha, \beta \mid \tilde{p}_i) \\ &= (\alpha - 1) \sum_{i=1}^{N} \ln \tilde{p}_i + (\beta - 1) \sum_{i=1}^{N} \ln(1 - \tilde{p}_i) - N \ln \mathrm{B}(\alpha, \beta).\end{aligned} \tag{3}$$

MLE values for α and β are then found numerically by solving:

$$\frac{\partial \ln \mathcal{L}(\alpha, \beta \mid \{\tilde{p}\})}{\partial \alpha} = \sum_{i=1}^{N} \ln \tilde{p}_i - N\left(\psi(\alpha) - \psi(\alpha + \beta)\right) = 0 \tag{4}$$

$$\frac{\partial \ln \mathcal{L}(\alpha, \beta \mid \{\tilde{p}\})}{\partial \beta} = \sum_{i=1}^{N} \ln(1 - \tilde{p}_i) - N\left(\psi(\beta) - \psi(\alpha + \beta)\right) = 0 \tag{5}$$

where ψ is the digamma function. An alternative approach is to use the method of moments estimators:

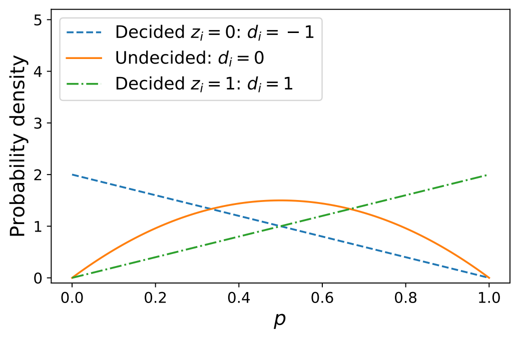
$$\hat{\alpha} = \overline{p}\left(\frac{\overline{p}\,(1 - \overline{p})}{S_{\hat{p}}^2} - 1\right), \quad \hat{\beta} = (1 - \overline{p})\left(\frac{\overline{p}\,(1 - \overline{p})}{S_{\hat{p}}^2} - 1\right), \tag{6}$$

where $\overline{p}$ and $S_{\hat{p}}^2$ are the sample mean and variance, respectively, of $\hat{p}$.

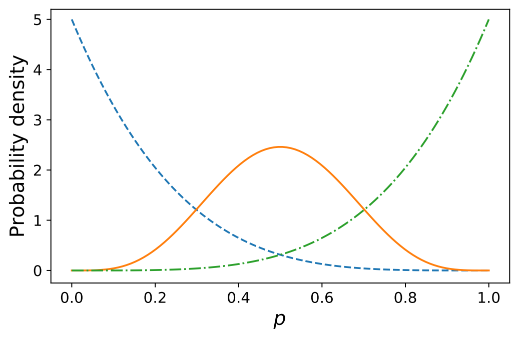
4.1 Calibrating decision priors

Our approach to inferring the difficulty distribution introduces decision priors D_d to integrate the crowdsourcing allocation algorithm into the inference. The exact choice of these priors is up to the needs of the research; this freedom provides the researcher a framework for expressing their confidence in the algorithm. For example, a researcher studying problem difficulty p who is very confident in the algorithm's decision $d_i = 1$ ($d_i = -1$) would select priors that are sharply peaked at $p_i = 1$ ($p_0 = 0$). Likewise, a researcher with less confidence in the decisions would choose flatter priors over $0 \leq p_i \leq 1$.

Depending on their problem, researchers may have *a priori* reasons for choosing their priors. But, sometimes,



(a) Less confidence is algorithm's decisions



(b) More confidence in decisions

Figure 2: Example decision priors for when a researcher is (**a**) less confident in the allocation method's decisions, (**b**) more confident in the decisions.

they may wish to calibrate their decision priors from data. Here we describe one calibration method. Suppose the researcher has access to a small number of gold standard (GS) crowdsourcing tasks, where the true labels z_i for these items are known. We propose to apply the crowdsourcing algorithm to these gold standard items in order to estimate its accuracy, then calibrate the decision priors accordingly.

Suppose there are n_0 GS tasks with true label $z = 0$ and n_1 GS tasks with true label $z = 1$. Let m_{00} be the number of n_0 tasks decided by the algorithm to be label zero ($d = -1$) and let m_{01} be the number of n_0 tasks decided to be label one ($d = 1$) (with $m_{00} + m_{01} = n_0$). Likewise, define m_{11} and m_{10} for the n_1 tasks (with $m_{11} + m_{10} = n_1$). Now, consider the $d = -1$ prior. We argue that the prior probability for $p < 1/2$ should be m_{00}/n_0. (In practice, these counts m will likely need to be smoothed to avoid zero counts; see below.) Likewise, for $d = 1$, the prior probability for $p > 1/2$ should be m_{11}/n_1. For a beta $d = -1$ decision prior, we can thus choose the Beta parameters to satisfy

$$\frac{m_{00}}{n_0} = \int_0^{1/2} D_{-1}(p \mid a_{-1}, b_{-1})\, dp, \tag{7}$$

where D_{-1} is the beta prior for $d = -1$ with parameters a_{-1} and b_{-1}. This integral is the regularized incomplete Beta function $I_{1/2}(a_{-1}, b_{-1})$. Choosing $a_{-1} = 1$ reduces this to $I_{1/2}(1, b_{-1}) = 1 - (1/2)^{b_{-1}}$, giving a calibrated b_{-1} of

$$b_{-1} = \log_2\left(\frac{n_0}{m_{01}}\right). \tag{8}$$

Likewise, to calibrate the $d = 1$ decision prior $D_1 = \mathrm{Beta}(a_1, b_1)$, choose parameters a_1 and b_1 that satisfy

$$\frac{m_{11}}{n_1} = 1 - \int_0^{1/2} D_1(p \mid a_1, b_1)\, dp. \tag{9}$$

Again, choosing $b_1 = 1$ for simplicity gives a calibrated $a_1 = \log_2(n_1/m_{10})$.

These choices of prior parameters $(1, b_0)$ and $(a_1, 1)$ now serve to calibrate the decision priors given the performance on the GS tasks. Of course, not all problems have corresponding GS tasks (indeed, we do not use GS tasks in this work), but for those problems where such tasks are available, this method, which is applicable beyond the case of $x = p$ that we focus on, provides a simple means to inform the prior before continuing on to the main set of tasks.

4.2 Generalizations and other applications of DEPS

DEPS is a general-purpose approach to inference about problems when using efficient allocation strategies. Although we focus our study on inferring the distribution of task difficulties, here we briefly discuss other applications of DEPS.

One application of DEPS is to infer worker completion times. If workers are consistently given easy tasks only, then we may have a biased representation of how quickly they can complete work. We can use DEPS to infer completion times as follows. Let $\{S_{ij}\}$ be the completion times for tasks $i = 1, \ldots, N$ and workers $j = 1, \ldots, M$, e.g., the number of seconds they took to complete the tasks. Human interevent times are often modeled with a log-normal, $\ln(S_{ij}) \sim \mathcal{N}(\mu_j, \tau_j)$ [26]. We establish conjugate priors, $\mu_j \mid \tau_j, d \sim \mathcal{N}(\mu_d, \tau_j)$ and $\tau_j \mid d \sim \mathrm{Gamma}(\alpha_d, \beta_d)$, with $\alpha_d, \beta_d > 0$. The posterior for μ_j and τ_j is

$$P(\mu_j, \tau_j \mid d_i, \{S_{ij}\}) \propto \prod_{i \in I_j} P(S_{ij} \mid \mu_j, \tau_j, d_i) P(\mu_j \mid \tau_j, d_i) P(\tau_j \mid d_i). \tag{10}$$

This model yields an estimated distribution for completion time for each worker. To understand the distribution for all workers, the researcher can sample from each distribution and aggregate the samples, $\tilde{\mu}_j$ and $\tilde{\tau}_j$. The distribution $P(\tilde{\mu}, \tilde{\tau}) \approx P(\mu, \tau)$, the true, overall distribution of worker completion times. Assuming that completion times and task difficulty are related, we can reflect the decision d_i of the algorithm by varying the parameters of the priors for μ_j and τ_j given d_i For example, a normal prior with a larger mean may be appropriate for a task that has been undecided ($d_i = 0$), while decided tasks ($d_i \neq 0$), should have a prior with a smaller mean.

In addition to worker completion times, DEPS can also consider task completion times. Since the tasks are not randomly shown to the workers the data collected on task completion time will also be biased. For this application of DEPS, we again model the time to complete tasks, S_{ij}, as a log-normal, $\ln(S_{ij}) \sim \mathcal{N}(\mu_i, \tau_i)$ and establish priors $\mu_i \mid \tau_i, d_i \sim \mathcal{N}(\mu_{d_i}, \tau_i)$ and $\tau_i \mid d_i \sim \mathrm{Gamma}(\alpha_{d_i}, \beta_{d_i})$, with $\alpha_{d_i}, \beta_{d_i} > 0$. The posterior for μ_i and τ_i of task i is

$$P(\mu_i, \tau_i \mid d_i, \{S_{ij}\}) \propto \prod_{j \in J_i} P(S_{ij} \mid \mu_i, \tau_i, d_i) P(\mu_i \mid \tau_i, d_i) P(\tau_i \mid d_i). \tag{11}$$

The priors are then adjusted to reflect the decisions d_i in a manner similar to the worker completion time model given above, as is the inference of the overall distribution of task completion times.

Even further generalizations of DEPS are possible. Briefly, suppose a crowdsourcer is interested in understanding the behavioural traces of workers as they complete tasks. One approach is to use observed worker dynamics to determine a vector space for embedding each worker. Distances in this space can then be used to predict worker features or activities [27]. Let $\mathbf{u}_{ij} = \mathbf{x}_i + \mathbf{e}_{ij}$ be a k-dimensional data vector representing the behavioural trace data for worker j responding to task i, where $\mathbf{x}_i$ is the population data for that task and $\mathbf{e}_{ij}$ is the worker-specific deviations from the population. Then, let $\mathbf{w}_j$ be a k-dimensional unit vector for encoding worker j. This encoding vector and its estimator $\hat{\mathbf{w}}_j$, can capture the various contributions of the behavioural trace data to the worker's labeling decision: the worker responds with $y_{ij} = 1$ if the projection $\langle \mathbf{u}_{ij}, \mathbf{w}_j \rangle \geq \tau_j$, otherwise $y_{ij} = 0$. Inference of this vector can be performed given priors for the parameters (such as τ_j) and $\mathbf{u}_{ij}$, parameterized by a multivariate Gaussian distribution, for example. The allocation strategy's decisions can then be incorporated by defining suitable decision priors for $\mathbf{u}_{ij}$, $\mathbf{w}_j$, and so forth, conditioned on d_i.

While only brief outlines, the above applications of DEPS indicate that suitable conditioning of statistical models, even complex ones, using the decision variables, has the potential to improve the information attainable about the crowdsourcing problem at hand.

5 Materials and Methods

In this section, we describe the real-world crowdsourcing datasets we analyze (Sec. 5.1), the synthetic crowdsourcing we simulate (Sec. 5.4), and how DEPS is applied including the details of the efficient allocation framework we use (Sec. 5.2). We focus on using DEPS to infer the task difficulty distribution $\Pr(p)$, where p is the probability a worker response $y_{ij} = 1$; tasks with $p \approx 1/2$ are difficult in that it takes many worker responses to accurately distinguish if $z = 1$ or $z = 0$. See Sec. 4 for a general specification of DEPS along with examples for inferring properties other than p. We also describe a traditional "baseline" method to compare DEPS (Sec. 5.3) to, and we describe our quantitative measures of evaluation for these methods (Sec. 5.5). The results of our experiments are presented in Sec. 6

5.1 Datasets

We study three crowdsourcing datasets. These data were not generated using an efficient allocation algorithm, and so it has become standard practice to evaluate such algorithms with these data [21, 10]— since labels were collected independently, one can use an allocation algorithm to choose what order to reveal labels from the full set of labels, essentially "rerunning" the crowdsourcing after the fact. Following Li, *et al.* [21], we only utilize at most 50% of the total responses available so that the allocation algorithm does not "run out" of requested labels as it polls the data.

RTE Recognizing Textual Entailment (RTE) dataset [4]. Pairs of written statements were taken from the PASCAL RTE-1 data challenge [28] and shown to Amazon Mechanical Turk workers who responded whether or not one statement entailed the other. These data consist of $N = 800$ binary tasks each of which received 10 labels from workers, giving a total of $8,000$ responses.

Bluebirds Identifying Bluebirds dataset [27]. Each task is a photo of a bluebird, either an Indigo Bunting or a Blue Grosbeak, and the worker is asked if the photo contains an Indigo Bunting. There are $N = 108$ binary tasks, and 39 responses for each tasks, therefore, a total of $4,212$ responses.

Relevance Identifying Page Relevance dataset [29]. Each task displays a webpage and a given topic, the worker is asked to determine if the webpage is relevant to the given topic. This dataset contains $N = 2,275$ tasks, with a range of 1 to 10 responses per task. There is on average 6.04 responses per task. In total there are $13,749$ responses.

When applying DEPS to infer the distributions of task difficulty p for these datasets, we used beta decision priors with parameters $\Phi_{-1} = (1,2)$, $\Phi_0 = (2,2)$, and $\Phi_1 = (2,1)$.

5.2 Allocation method

We used the Requallo allocation algorithm to choose which task labels are received, either from the real datasets (Sec. 5.1) or the simulated crowdsourcing (Sec. 5.4). We chose for completeness a ratio requirement with $c = 4$ (Sec. 2.3). All other details were as those given by Li *et al.* [21].

5.3 Baseline method—Wald estimation

To understand the performance of DEPS, we compare to the following baseline method, known as Wald estimation, which is a conventional approach to this inference problem. Wald estimation uses the MLE estimator for p: for each task i, $\hat{p}_i = a_i/n_i$, where a_i is the number of positive responses to i from workers and n_i is the total number of responses from workers. The $\hat{p}$ are then used directly to approximate the distribution $\Pr(p) \approx \Pr(\hat{p})$ and estimate the beta parameters using Eqs. (3)–(5) or Eq. (6).

Unlike DEPS (Sec. 4), this baseline estimation using $\hat{p}$ suffers from small-data problems. Specifically, if either $a_i = 0$ or $a_i = n_i$, then the likelihood used in MLE is undefined. Yet, either situation is likely when only a few labels are collected for task, which can often occur when using an efficient allocation strategy (see Fig. 1(b)). A common solution to this problem is to use a *smoothed* estimate of $\hat{p}$, where a_i, b_i, and n_i are replaced with $a_i + \epsilon$, $b_i + \epsilon$, and $n_i + 2\epsilon$, respectively (we use $\epsilon = 1$). Unfortunately, while smoothing to this degree using "pseudo-labels" is common, such a level of smoothing may overly bias our Wald estimates of $\hat{p}$ towards $\hat{p} = 1/2$, so we also explore an alternative solution: the values of $\hat{p}$ are *transformed* using $(\hat{p}(N-1) + 1/2)/N$ such that now $\hat{p} \in (0,1)$ instead of $\hat{p} \in [0,1]$ [30]. We explore both solutions in our results (Sec. 6), but focus on the transformed $\hat{p}$, which we found generally outperformed the smoothed $\hat{p}$. In practice, when fitting to $\{\hat{p}\}$, we found better parameter estimates when using MLE for smoothed Wald and Method of Moments for transformed Wald. DEPS, in contrast, is not affected by this small-data problem.

5.4 Crowdsourcing simulations

We wish to supplement our results using real crowdsourcing with controlled simulations of crowdsourcing problems. To generate synthetic datasets according to the crowdsourcing model defined in Sec. 2, we assume each task i has an intrinsic parameter p_i with each worker response to task i as a Bernoulli variable with parameter p_i. This probability governs how difficult a task is in terms of how many responses are necessary to determine its label: the closer p is to $1/2$ the more labels are necessary to accurately distinguish $z = 0$ from $z = 1$. We further endow the model with a prior probability distribution on p_i, specifically $p \sim \mathrm{Beta}(\alpha, \beta)$, a beta distribution with hyperparameters α and β. This prior distribution lets us generate N tasks and control the difficulty of each by determining how many tasks are near $p \approx 1/2$ and how many tasks are near $p \approx 0$ or $p \approx 1$. Conversely, statistical inference can be performed to determine the posterior distribution of p given the data $\{y_{ij}\}$. Using this simulation model we can implement efficient budget allocation techniques such as Requallo in order to study the effect of efficient allocation on the distribution $\Pr(p)$. Unlike with the real data, the true distribution of p is known, and we can test inference methods by comparing their estimates to the true distribution.

To apply DEPS to simulated data, we used decision prior parameters $\Phi_{-1} = (1, 5)$, $\Phi_0 = (5, 5)$, and $\Phi_1 = (5, 1)$. These are more confident priors than the ones used with real data (Sec. 5.1); see Sec. 7 for further discussion.

5.5 Evaluating performance

We use an information-theoretic measure to quantitatively compare the distribution $\Pr(p)$ of task difficulty p found under various conditions. For synthetic datasets we have imposed the ground truth distribution of p, so we can compare our inferred estimates to this known ground truth. The ground truth distributions are not available to us when examining the real datasets, so instead we compare the estimated $\Pr(p)$ found with a biased portion of the data revealed using Requallo with the estimated $\Pr(p)$ found using all the data, which were collected in an unbiased manner.

The Kullback-Leibler (KL) divergence (or relative entropy) between two random variables X and Y (measured in 'nats') is

$$D_{\mathrm{KL}}(X,Y) = \int_{-\infty}^{\infty} f_X(x;\theta) \ln\left(\frac{f_X(x;\theta)}{f_Y(x;\theta')}\right) dx \tag{12}$$

$$= \ln\left(\frac{\mathrm{B}(\alpha',\beta')}{\mathrm{B}(\alpha,\beta)}\right) + (\alpha-\alpha')\,\psi(\alpha) + (\beta-\beta')\,\psi(\beta) + (\alpha'-\alpha+\beta'-\beta)\,\psi(\alpha+\beta), \tag{13}$$

where f_X (f_Y) is the density for X (Y), and the second line holds for the case where X and Y are both beta-distributed: $X \sim \mathrm{Beta}(\alpha,\beta)$ and $Y \sim \mathrm{Beta}(\alpha',\beta')$, B is the Beta function, and ψ is the digamma function. We utilize Eq. (13) to measure how well our inference method estimates the underlying prior beta distribution of p. If the distributions of p are not Beta, or one is interested in a different property than p, one can still use the general Eq. (12), perhaps with an appropriate sample estimator for the KL-divergence [31]. We take X to be the ground truth distribution for p (synthetic data) or the distribution estimated using the full, unbiased data (real data), and Y to be the distribution inferred using DEPS on Requallo-collected data.

6 Experiments

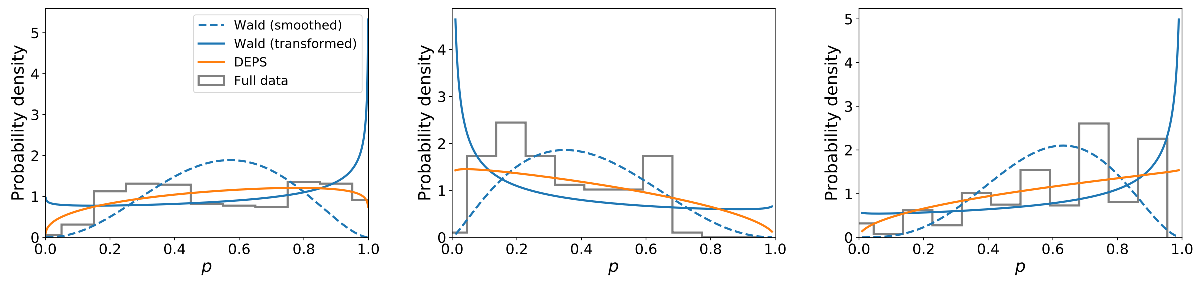
We divide our experiments into those analyzing real-world data (Sec. 6.1, Figs. 3–5, and Table 1) and those analyzing synthetic crowdsourcing simulations (Sec. 6.2 and Figs. 6, 7).

6.1 Real-world data

Figure 3 considers estimation of the difficulty distribution Pr(p) when limited to 25% of the available budget. For the Wald baseline method, Fig. 3 show how efficient allocation biases the estimated $\hat{p}$ towards $1/2$ in the "smoothed" case and toward 0 or 1 in the "transformed" case. In contrast to Wald, which requires either smoothed or transformed point estimates, DEPS shows good qualitative agreement with the full data, and fits the unmodified data, making it more robust than Wald to the bias from efficient allocation.

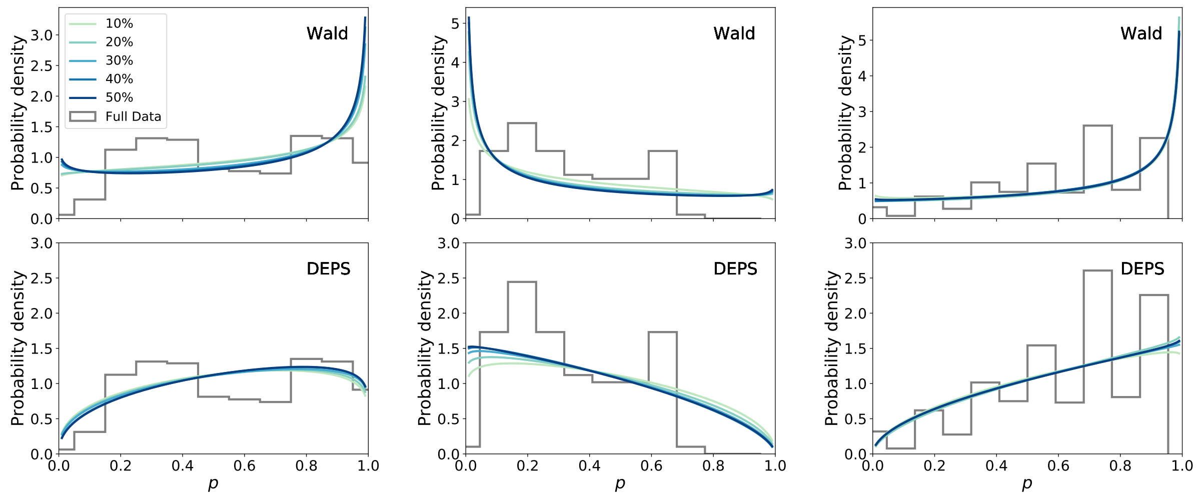
Expanding upon Fig. 3 we now investigate in Fig. 4 how the estimates of Pr(p) change as more budget is made available to the crowdsourcer, up to 50% of the total number of labels provided by the data. We observe in both Wald and DEPS that the estimated distributions converge quickly, often with as little as 10% of available data.

Expanding on the distribution convergence, Fig. 5 shows how much information about the distribution of p given by the full (unbiased) data is provided by DEPS and Wald as more biased data are made using Requallo. With the exception of the Relevance dataset, where performance is relatively comparable, DEPS provides more information as evidenced by the lower KL-divergence (Sec. 5.5). Interestingly, Wald shows performance that *degrades* slightly with more data in Fig. 5(a) and 5(b), likely due to the fact that, while more data are available, there is more biased data



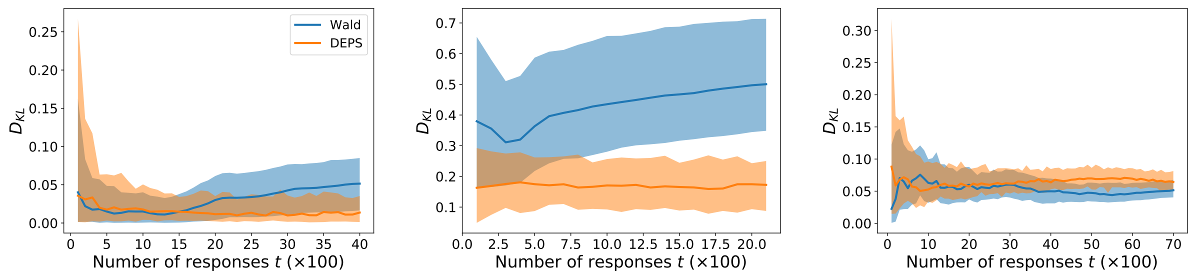
(a) RTE (b) Bluebirds (c) Relevance

Figure 3: Inference of problem difficulty distributions using 25% of labels. Tasks are allocated using Requallo, then Wald estimation or DEPS estimation of problem difficulty is performed. These results are compared to the distribution of $\hat{p}$ estimated using the full dataset (100% of labels), show as histograms. The "smoothed" and "transformed" variants of Wald estimation tend to add too much probability near either the center ($p = 1/2$) or extremes ($p = 0, 1$). DEPS achieves good agreement with the full data distribution.



(a) RTE (b) Bluebirds (c) Relevance

Figure 4: Convergence of estimated distributions as more crowd data are used. Visually, we see that DEPS and Wald (transformed) both converge relatively quickly, often after using only 10% of the available data.



(a) RTE (b) Bluebirds (c) Relevance

Figure 5: DEPS generally provides more information about the distribution estimated using the full, unbiased dataset than Wald, whose performance degrades as more (biased) data are received, with the exception of Relevance. Shaded areas denote 95% CI computed over Requallo realizations.

Table 1: In all cases, DEPS parameter estimates are closer than Wald estimates to the parameters inferred using the full, unbiased dataset.

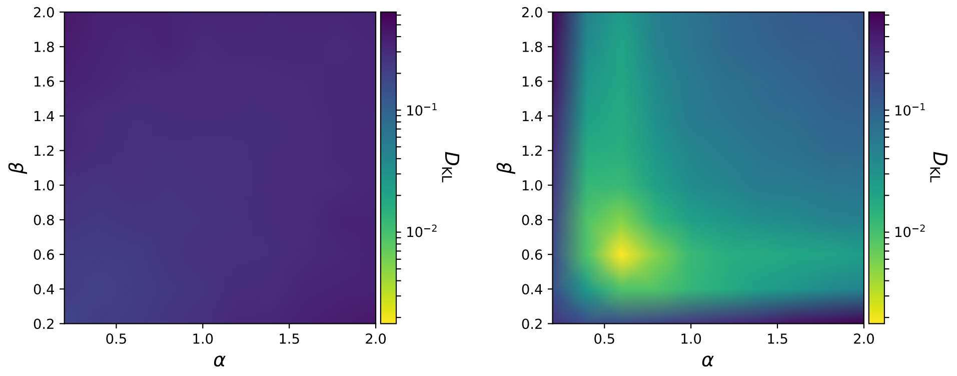
		Estimates from biased 50% data		Full data	
Dataset	Method	α [95% CI]	β [95% CI]	α	β
RTE	Wald	0.88 [0.775, 1.05]	0.62 [0.532, 0.728]	1.24	0.92
	DEPS	1.43 [1.265, 1.625]	1.12 [0.974, 1267]	1.24	0.92
Bluebirds	Wald	0.46 [0.31, 0.61]	0.88 [0.628, 1.186]	2.25	3.66
	DEPS	1.00 [0.780, 1.249]	1.56 [1.28, 2.071]	2.25	3.66
Relevance	Wald	0.98 [0.926, 1.03]	0.49 [0.463, 0.523]	1.51	0.72
	DEPS	1.53 [1.529, 1.608]	1.00 [0.945, 1.056]	1.51	0.72

available, as these responses are gathered in a non-uniform manner due to Requallo. DEPS, however, does not exhibit this degraded performance on these data. We discuss this further in Sec. 7.

Lastly, to further compare DEPS and Wald estimates on the real-world datasets, Table 1 shows the estimated parameters $\theta = (\alpha, \beta)$ for $\Pr(p)$. We applied both methods to 50% of the data gathered using Requallo (and the bias entailed by Requallo). For comparison, we also report θ as estimated using the full, unbiased data. Although there remains room for improvement, Table 1 shows that DEPS achieves estimates of θ closer to estimates from the unbiased data than Wald does in all cases.

6.2 Synthetic data

Supplementing our results using real datasets, we also explored the performance of DEPS and the Wald baseline method for crowdsourcing problems where the true distribution $\Pr(p)$ is known. Figure 6 shows the KL-divergence D_{KL} between estimated $\Pr(p)$ and the true distribution across a range of parameter values (α, β). Across most parameters, except for some cases with extremely small values of α or β, DEPS provides more information (lower D_{KL}) than Wald does about the true distribution. (Note the logarithmic color scale for D_{KL} used in Fig. 6.)



(a) Wald (b) DEPS

Figure 6: The KL-divergence (or relative entropy) D_{KL} between the inferred and true distributions of p over a range of parameters for the true distribution. Here we see that DEPS outperforms Wald for most parameter values, achieving a lower divergence in its estimates of the true distribution.

Next, we investigated how DEPS and Wald performed in terms of budget efficiency. Figure 7(a) shows the evolution of D_{KL} as more responses are received. Overall, DEPS is consistently more informative about the true distribution of p than Wald. As with the real-world data (Fig. 5), we do not see a monotonic decrease in D_{KL} as more responses are received. DEPS generally fares better in this regard, although some degradation is also observed (note the logarithmic vertical scale, however). Likewise, Fig. 7(b) shows the average number of responses needed to reach $D_{\mathrm{KL}} < 0.3$ nats. In these cases, DEPS reaches this information-theoretic threshold with fewer responses than the traditional Wald method.

Taken together, DEPS is budget-efficient, able to generate more information about the underlying distribution $\Pr(p)$ than baseline methods even when baseline methods use the same efficient allocation algorithm.

7 Discussion

DEPS is a flexible approach to estimating a distribution when given biased data. By explicitly incorporating the efficient algorithm's decisions into the prior distributions, DEPS can adjust for the bias induced by task allocation algorithms. Using Requallo as an allocation strategy, DEPS estimates a more accurate distribution for the true, unknown property distribution than if estimation were performed using traditional methods. This allows researchers to collect data from the crowd efficiently, while being able to extract information about more than just the correct label for a task.

Our experiments (Sec. 6) provide evidence supporting the performance of DEPS, both its accuracy and its efficiency, but more work is warranted. We focused our validation procedure around a single inference task, estimating the difficulty distribution for a crowdsourcing problem. Future work should consider other problem properties, such as

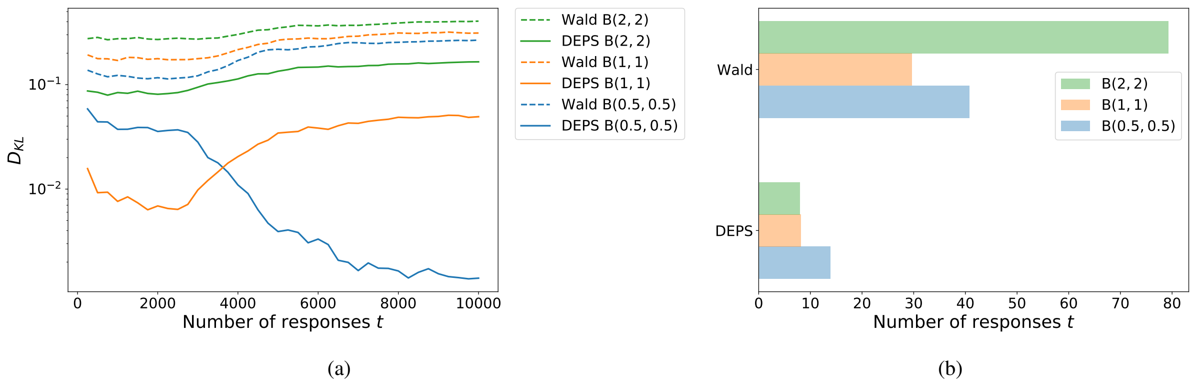


Figure 7: Panel (**a**) shows DEPS and Wald estimates of true distribution over budget (total number of responses) for a subset of underlying distributions. DEPS consistently provides more information than the Wald baseline about the true distribution, as measured using the KL divergence. Panel (**b**) denotes the average number of responses needed to reach $D_{\mathrm{KL}} \leq 0.3$. For these cases DEPS achieves this accuracy level in fewer responses than the Wald baseline.

better understanding completion times or behavioural traces of works, as we briefly sketched out in Sec. 4.2. DEPS works using an informative "decision" prior, and this introduces subjectivity into the property inference. We provided some guidance for calibrating decision priors, but there remains considerable researcher flexibility. Indeed, we found variations in DEPS performance across the parameter space in Fig. 6 that imply care may be needed when matching decision priors to specific problems. A systematic study of choosing and tuning decision priors efficiently is warranted.

We found in our results that performance of DEPS (for synthetic data) would degrade as more data were collected. As more labels are received overall, more difficult tasks are more likely to reach the completeness requirement, pushing the "horizon" of completeness towards $p = 1/2$. In this case, we may want to flatten our priors accordingly. In other words, we motivated the choice of steep decision priors in Sec. 4 as representing the confidence that we have in the algorithm's decisions, but, in fact, that steepness is also related to the budget available to the crowdsourcer. Incorporating a budget dependency into the design of the decision priors is therefore an important avenue for improvement, particularly for a large-scale deployment of DEPS. This would be especially interesting for crowdsourcing problems designed to distribute the crowd non-uniformly, deploying more workers in some areas of the problem space than others and adapting (perhaps dynamically) the DEPS decision priors accordingly.

In summary, DEPS provides a method to account for the bias introduced by efficient allocation algorithms in order to better understand properties of interest for a crowdsourcing problem. The researcher can apply DEPS to problems of interest by adapting the decision priors to reflect how the allocation method introduces bias. The flexibility of DEPS allows for researchers to implement DEPS for a variety of research questions, such as understanding problem difficulty, worker completion times, and behavioral traces. Researchers can implement efficient crowdsourcing while performing accurate inference about the distribution of interest, showing that DEPS can help address one of the key challenges of crowdsourcing: maximizing the information gained from finite, and often costly to gather, data.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1447634.

References

[1] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006. 1

[2] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, vol. 14, no. 1, pp. 75–90, 2008. 1

[3] A. Kittur, J. V. Nickerson, M. Bernstein, E. Gerber, A. Shaw, J. Zimmerman, M. Lease, and J. Horton, "The future of crowd work," in *Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 1301–1318, ACM, 2013. 1

[4] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks," in *Proceedings of the conference on empirical methods in natural language processing*, pp. 254–263, Association for Computational Linguistics, 2008. 1, 4, 12

[5] Y. Yan, R. Rosales, G. Fung, and J. G. Dy, "Active learning from crowds.," in *ICML*, vol. 11, pp. 1161–1168, 2011. 1

[6] E. Kamar, S. Hacker, and E. 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*, pp. 467–474, International Foundation for Autonomous Agents and Multiagent Systems, 2012. 1

[7] T. S. Behrend, D. J. Sharek, A. W. Meade, and E. N. Wiebe, "The viability of crowdsourcing for survey research," *Behavior research methods*, vol. 43, no. 3, p. 800, 2011. 1

[8] J. C. Bongard, P. D. Hines, D. Conger, P. Hurd, and Z. Lu, "Crowdsourcing predictors of behavioral outcomes," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 1, pp. 176–185, 2013. 1

[9] M. J. Salganik and K. E. Levy, "Wiki surveys: Open and quantifiable social data collection," *PloS one*, vol. 10, no. 5, p. e0123483, 2015. 1

[10] T. C. McAndrew, E. A. Guseva, and J. P. Bagrow, "Reply & supply: Efficient crowdsourcing when workers do more than answer questions," *PloS one*, vol. 12, no. 8, p. e0182662, 2017. 1, 12

[11] M. D. Wagy, J. C. Bongard, J. P. Bagrow, and P. D. Hines, "Crowdsourcing predictors of residential electric energy usage," *IEEE Systems Journal*, no. 99, pp. 1–10, 2017. 1

[12] D. Berenberg and J. P. Bagrow, "Efficient crowd exploration of large networks: The case of causal attribution," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, p. 24, 2018. 1, 2

[13] H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, and V. Verroios, "Challenges in data crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 4, pp. 901–911, 2016. 1

[14] H. Zheng, D. Li, and W. Hou, "Task design, motivation, and participation in crowdsourcing contests," *International Journal of Electronic Commerce*, vol. 15, no. 4, pp. 57–88, 2011. 1

[15] N. Kaufmann, T. Schulze, and D. Veit, "More than fun and money. worker motivation in crowdsourcing-a study on mechanical turk.," in *AMCIS*, vol. 11, pp. 1–11, Detroit, Michigan, USA, 2011. 1

[16] C. Eickhoff and A. P. de Vries, “Increasing cheat robustness of crowdsourcing tasks,” *Information retrieval*, vol. 16, no. 2, pp. 121–137, 2013. 1, 2

[17] V. S. Sheng, F. Provost, and P. G. Ipeirotis, “Get another label? Improving data quality and data mining using multiple, noisy labelers,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 614–622, ACM, 2008. 2, 4

[18] D. R. Karger, S. Oh, and D. Shah, “Iterative learning for reliable crowdsourcing systems,” in *Advances in neural information processing systems*, pp. 1953–1961, 2011. 2

[19] D. R. Karger, S. Oh, and D. Shah, “Budget-optimal task allocation for reliable crowdsourcing systems,” *Operations Research*, vol. 62, no. 1, pp. 1–24, 2014. 2

[20] S. Oyama, Y. Baba, Y. Sakurai, and H. Kashima, “Accurate integration of crowdsourced labels using workers’ self-reported confidence scores,” in *IJCAI*, pp. 2554–2560, 2013. 2

[21] Q. Li, F. Ma, J. Gao, L. Su, and C. J. Quinn, “Crowdsourcing high quality labels with a tight budget,” in *Proceedings of the ninth acm international conference on web search and data mining*, pp. 237–246, ACM, 2016. 2, 4, 5, 6, 12, 13

[22] A. P. Dawid and A. M. Skene, “Maximum likelihood estimation of observer error-rates using the EM algorithm,” *Applied statistics*, pp. 20–28, 1979. 2, 3, 4

[23] S. Dow, A. Kulkarni, S. Klemmer, and B. Hartmann, “Shepherding the crowd yields better work,” in *Proceedings of the ACM 2012 conference on computer supported cooperative work*, pp. 1013–1022, ACM, 2012. 2

[24] J. M. Rzeszotarski and A. Kittur, “Instrumenting the crowd: using implicit behavioral measures to predict task performance,” in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 13–22, ACM, 2011. 2, 4

[25] X. Chen, Q. Lin, and D. Zhou, “Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing,” in *International Conference on Machine Learning*, pp. 64–72, 2013. 4, 5

[26] J. L. Iribarren and E. Moro, “Impact of human activity patterns on the dynamics of information diffusion,” *Physical review letters*, vol. 103, no. 3, p. 038702, 2009. 11

[27] P. Welinder, S. Branson, P. Perona, and S. J. Belongie, “The multidimensional wisdom of crowds,” in *Advances in neural information processing systems*, pp. 2424–2432, 2010. 12, 13

[28] I. Dagan, O. Glickman, and B. Magnini, “The PASCAL recognising textual entailment challenge,” in *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pp. 177–190, Springer, 2006. 12

[29] M. D. Smucker, G. Kazai, and M. Lease, “Overview of the TREC 2012 crowdsourcing track,” in *Proceedings of the 21st NIST text retrieval conference (TREC)*, 2012. 13

[30] M. Smithson and J. Verkuilen, “A better lemon squeezer? maximum-likelihood regression with beta-distributed dependent variables.,” *Psychological methods*, vol. 11, no. 1, p. 54, 2006. 13

[31] F. Pérez-Cruz, “Kullback-Leibler divergence estimation of continuous distributions,” in *2008 IEEE international symposium on information theory*, pp. 1666–1670, IEEE, 2008. 15