Efficient Budget Allocation with Accuracy Guarantees for Crowdsourcing Classification Tasks

Long Tran-Thanh, Matteo Venanzi, Alex Rogers & Nicholas R. Jennings
University of Southampton
{ltt08r,mv1g10,acr,nrj}@ecs.soton.ac.uk

ABSTRACT

In this paper we address the problem of budget allocation for redundantly crowdsourcing a set of classification tasks where a key challenge is to find a trade–off between the total cost and the accuracy of estimation. We propose CrowdBudget, an agent–based budget allocation algorithm, that efficiently divides a given budget among different tasks in order to achieve low estimation error. In particular, we prove that CrowdBudget can achieve at most $\max\left\{0, \frac{K}{2} - O\left(\sqrt{B}\right)\right\}$ estimation error with high probability, where K is the number of tasks and B is the budget size. This result significantly outperforms the current best theoretical guarantee from Karger *et al.* In addition, we demonstrate that our algorithm outperforms existing methods by up to 40% in experiments based on real–world data from a prominent database of crowdsourced classification responses.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents

General Terms

Algorithms, Theory, Experimentation

Keywords

Crowdsourcing, Task Allocation, Budget Limit, Regret Bounds

1. INTRODUCTION

Crowdsourcing classification tasks, such as classification of complex images [5] or identification of buildings on maps [3], has recently become widely used as it presents a low-cost and flexible approach to solve complex classification tasks by combining human computation with agent intelligence [2]. In particular, by dedicating tasks to a population of hired users (i.e. workers) for a small fee, the taskmaster (i.e. task requester) can collect a large set of class labels from the workers and ultimely estimate classification answers from the multiple reports. To achieve this, the taskmaster *redundantly* allocates tasks to users (in order to reduce uncertainty, as a single response might be unreliable), and then aggregates the responses into a final estimate using a fusion method (e.g. majority voting [1] or IBCC [10]). Within many systems, such a process of task allocation and reports fusion is not trivial and is typically done by a computer agent as it might need complex computation that humans cannot provide [5, 6].[1]

Now, a key challenge within these crowdsourcing systems is to find an efficient trade–off between the cost of redundant task allocation and the accuracy of the result. In more detail, by assigning multiple users to a single task, we can achieve higher accuracy of answer estimation, but we might also suffer a higher cost due to hiring more users. To date, research work has typically focused on applications where the cost of task allocation is uniform for each task [2, 12, 5]. In this case, the search of the aforementioned trade–off is reduced to the problem of minimising the estimation error per task given the number of assigned users to a single task. However, in many real–world scenarios, completing different tasks might require *different costs* that depend on the difficulty and the time required for the user to complete such a task. For example, consider a habitat monitoring project where the goal is to accurately identify the living area of some rare species (e.g. the desert tortoise from the Mojave desert, US, or the New Forest cicada in the UK). To avoid sending expeditions of professionals with high cost, one low–budget, but efficient, way to tackle this problem is to ask people from neighbouring areas to make observations and then send reports in return for a certain amount of payment. However, as the approachability of geographical areas might vary (e.g. due to possible dangers or landscape), we might need to pay different costs to motivate people to approach different areas in order to provide reports. Furthermore, this allocation scheme has to cope with the fact that the project is limited by its funds.

Another example comes from the human computation domain. Suppose a system designer aims to use a crowdsourcing platform to execute complex workflows of jobs [13]. In so doing, he/she decomposes the workflows into sets of micro–tasks, and assigns these tasks to a population of users. Now, a typical classification task within this domain is to determine whether a micro–task is completed, and the goal is to maximise the total accuracy of such classifications. To do so, he/she incentivises users to participate in this classification phase and compensate their effort with a certain payment. Note that the classification tasks may vary in difficulty, as some are more time consuming, while many others are trivial. Thus, users might need different sizes of payment when

[1]Hence the combination between human and agent (i.e. computer) intelligence.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

they face tasks with different difficulty levels. In addition, the total payment is limited, as the system designer wants to keep the cost within a finite budget.

Within the aforementioned scenarios, and many others besides, the main challenge is to redundantly allocate a set of tasks with different costs to an open population of users, with respect to a *budget limit*, such that the estimation error is minimised. In particular, if we allocate large budgets to cheaper tasks, whose answers can typically be accurately estimated with only a few responses, we might fail to achieve high total accuracy of estimation, as a large portion of these budgets could be used for other tasks where the allocated budget is insufficiently low. In contrast, it might not be efficient either to allocate much of the budget to expensive tasks, as this will have less available for the others. Since existing methods are typically not designed to make such trade–offs, they are unlikely to be efficient in our settings. Moreover, these methods do not provide theoretical guarantees on the estimation error, which implies that they might perform well on average, but they might arbitrarily poorly in many specific cases. A notable exception is the work of Karger *et al.* [6] which provides a $e^{-O(n)}$ bound for the average estimation error, where n is the number of users assigned to a single task. However, this bound is only asymptotic (i.e. it holds when the number of tasks tends to infinity), so is not practical for real–world systems, where the number of tasks is typically finite. Against this background, we focus on the *budget allocation problem* in which an agent (on the behalf of the taskmaster) has to allocate budgets to a set of tasks and the total allocated budget cannot exceed a certain limit. The agent's goal is then to find an optimal budget allocation that *minimises the total estimation error* of the answers. Within this paper, we propose CrowdBudget as a budget allocation strategy that efficiently tackles this problem. In particular, by combining CrowdBudget with a majority voting-efficient fusion method (i.e. methods that outperform the majority voting rule, see Section 3), our agent can proveably achieve a $\max\left\{0, \frac{K}{2} - O\left(\sqrt{B}\right)\right\}$ upper bound on its total estimation error with high probability, where B is the budget size and K is the number of tasks. This bound is lower than the one provided by Karger *et al.*. Moreover, in contrast to most current approaches, our algorithm allocates the budgets in advance through the analysis of costs and expected accuracy. This type of functioning is motivated by many crowdsourcing platforms in which the task requester needs to pre-set the number of assignments per task (e.g. Amazon Mechanical Turk or CrowdFlower). Given this, we advance the state–of–the–art as follows:

- We introduce the problem of budget allocation for crowdsourcing classification tasks, in which the goal is to minimise the error of the estimated answers for a finite number of tasks, with respect to a budget limit.
- We develop CrowdBudget, an algorithm that, combining with a fusion method, proveably achieves an efficient bound on the estimation error, which significantly advances the best known results.
- By comparing the performance of CrowdBudget with existing algorithms through extensive numerical evaluations on real–world data taken from a prominent crowdsourcing system, we demonstrate that our algorithm typically outperforms the state–of–the–art by achieving up to 40% lower estimation error.

The remainder of the paper is structured as follows. In Section 2 we discuss the related work in the domain of redundant task assignment in crowdsourcing applications. Then we formalise the model of budget allocation for task crowdsourcing in Section 3. Following this, we detail our proposed algorithm in Section 4. We also provide a theoretical performance analysis within this section. The results of the experimental evaluation is then discussed in Section 5. Section 6 concludes, and the appendix briefs the proofs.

2. RELATED WORK

Work on redundant allocation of classification tasks in crowdsourcing applications has typically focused on minimising the estimation error given the number of users assigned to a single task. In particular, Wellinder *et al.* proposed a multidimensional model of users in order to estimate the accuracy of a particular user's answer, and thus, to improve the estimation of the ground truth [12]. In a similar vein, a number of works used Bayesian learning techniques to predict the users' responses, such as the work of Kamar *et al.* [5] and the IBCC algorithm (for independent Bayesian classifiers combination) [10]. Apart from these, Dai *et al.* used a PoMDP–based (for partially observable Markov decision process) approach to model the estimation's quality [2]. In addition, Bachrach *et al.* relied on a machine learning based aggregator to derive an efficient estimation of the correct answer [1]. However, none of these methods will work in our setting, as they do not address the challenge of having different costs for different classification tasks. Nevertheless, they can be used as an underlying fusion method for CrowdBudget, as they provide majority voting efficient response fusion approaches (for more details, see Section 3).

More related to our work is CrowdScreen, an algorithm proposed by Parameswaran *et al.* [9], that aims to find an optimal dynamic control policy with respect to both total cost and total estimation error over a finite set of tasks. However, the cost of task allocation is considered to be uniform among different tasks in the system. In addition, as per the other aforementioned approaches, there are no guarantee on performance. One notable exception that does provide theoretical guarantees is the work of Karger *et al.* [6]. Within this work, the authors developed an algorithm based on low–rank matrix approximation to assign tasks to users and estimate correct answers. In addition, they devised an $e^{O(-n)}$ upper bound on the estimation error. However, this bound only holds when the number of tasks tends to infinity, which implies that their bound is not useful for most practical contexts, as opposed to our results (see Section 4 for more details).

3. MODEL DESCRIPTION

Let $1, \ldots, K$ denote the classification tasks whose outcomes have to be estimated. Within our model, we assume that the classifications are binary. Note that this assumption is reasonable, as it is true in many real–world systems. Let $t_k \in \{0, 1\}$ be the *unknown* ground truth (i.e. the correct answer) of each task k. To estimate t_k, we can request responses from a (large) set of users (i.e. population of the crowd) as follows. In our model, the taskmaster does not deterministically choose specific users to perform a particular task k, but only submits a set of tasks to a crowdsourcing system, as is the case in many open crowdsourcing systems with a large population of users (e.g. MechanicalTurk or

CrowdFlower) [2]. Following this, a computer agent (i.e. the system) allocates the budgets to each task k, with respect to the total budget limit B. It then redundantly requests answers for each task k from the crowd, allowing users to provide their response to the corresponding task. In return, the users receive a payment c_k from the system. Note that the agent cannot exceed the budget limit assigned to each task when requesting answers. Given the provided answers, the agent then uses a fusion method to estimate t_k.

We assume that for each user u and task k, there is an unknown Bernoulli distribution $D(u,k)$ from which u samples his answer to task k [3]. This distribution formalises the knowledge and uncertainty of user u towards task k. In our model, users from the crowd, who decide to provide an answer to a particular task, can be regarded as being chosen from the population with an unknown probability distribution X_k, which depends only on task k. In addition, we assume that the responders of each task k are chosen from the population in an i.i.d. (independent and identically distributed) manner, as they can be considered to be independent from each other.

We now introduce some terms. For each task k and user $u \sim X_k$, let $\mathbb{E}_{D(u,k)}[r_k(u)] = \mu_{u,k}$. That is, $\mu_{u,k}$ is the expected answer of user u to task k w.r.t. distribution $D(u,k)$. By denoting $\mathbb{E}_{X_k}[\mu_{u,k}] = \mu_k$, we assume that:

$$\forall k : |\mu_k - t_k| < \frac{1}{2}. \tag{1}$$

That is, we assume that for each task k, the expected value of the answers is somewhat closer to the correct answer. This assumption is common in the crowdsourcing literature [5, 10, 6] and can be justified as follows. Here, we assume that the users are not malicious (i.e. they do not provide wrong answers on purpose). As such, they only provide their best guess based on their knowledge of the task. However, this knowledge might be inaccurate with some uncertainty (hence the distributions $D(u,k)$). These together ensure that, on average, the answers will tend towards the ground truth (i.e. Equation 1 holds).

Consider a set of responses $r_k(u_1), \ldots, r_i(u_{n_k})$ of task k from the crowd, where $r_k(u_j) \sim D(u_j,k)$. Let $\hat{r}_k(n_k)$ denote the estimate of t_k, which is derived from the aforementioned n_k responses by using a fusion method. One simple, but efficient method is the majority voting rule, which can be formalised as follows:

$$\hat{r}_k^{\mathrm{MV}}(n_k) = \left\lfloor \frac{1}{n_k} \sum_{j=1}^{n_k} r_k(u_j) - \frac{1}{2} \right\rfloor + 1. \tag{2}$$

That is, $\hat{r}_k^{\mathrm{MV}}(n_k) = 1$ if $\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) \geq \frac{1}{2}$, and 0 otherwise. Note that efficient state–of–the–art fusion methods typically outperform the majority voting rule by using some user profile tracking method [1, 10]. This leads us to the following definition:

Definition 3.1 (MV–Efficiency). *A response fusion method is* MV–efficient *(or majority voting efficient) if its outcome is more accurate than the outcome of the majority voting rule on the same set of responses. More formally, let $\hat{r}_k(n_k)$ denote the estimated answer of that particular method given n_k responses. For each k we have:*

$$P\{\hat{r}_k(n_k) \neq t_k\} \leq P\{\hat{r}_k^{\mathrm{MV}}(n_k) \neq t_k\},$$

where t_k is the ground truth of task k.

Such MV–efficient methods other than the majority voting rule include, but are not limited to: weighted majority voting [8], graphical model based methods [1], and IBCC [10].

We now formalise our problem, the budget allocation for redundant task assignment, as follows. Let $r_{k,1}, \ldots, r_{k,n_k}$ be the (*a priori* unknown) responses of the crowd to task k, where n_k is the number of users who provided an answer to the task. Let $\hat{r}_k(n_k)$ denote the estimate of the correct answer t_k based on these responses, using a MV–efficient fusion method (see Definition 3.1). Recall that by requesting a response for task k, we have to pay a cost c_k. Let B denote our total budget that we can use for requesting answers. Let $\Delta(n_k)$ denote the estimation error, that is, the difference between the estimate $\hat{r}_k$, derived from answers $r_{k,1}, \ldots, r_{k,n_k}$, and the correct answer t_k. More formally, we have:

$$\Delta(n_k) = |\hat{r}_k(n_k) - t_k|. \tag{3}$$

The total expected estimation error with respect to the budget limit B can be then defined as

$$\mathbb{E}[\Delta(B)] = \sum_{k=1}^{K} \mathbb{E}[\Delta(n_k)] = \sum_{k=1}^{K} P\{\hat{r}_k(n_k) \neq t_k\} \tag{4}$$

Our task is to determine a set of $n_1, \ldots, n_K$ that minimises the total error, and thus, maximise the accuracy of estimation. More formally, we aim to solve the following optimisation problem:

$$\min_{\{n_k\}} \mathbb{E}[\Delta(B)] = \min_{\{n_k\}} \sum_{k=1}^{K} P\{\hat{r}_k(n_k) \neq t_k\}, \text{s.t.} \sum_{k=1}^{K} c_k n_k \leq B \tag{5}$$

In what follows, we will propose an algorithm that efficiently tackles the aforementioned optimisation problem.

4. BUDGET–LIMITED TASK ALLOCATION

Given the model formalisation, we now develop CrowdBudget, a budget allocation algorithm for redundant task assignment that can efficiently crowdsource tasks in order to minimise the total estimation error with respect to the budget limit. To do so, we first describe the algorithm in Section 4.1, before providing theoretical guarantees on its performance in Section 4.2.

4.1 The CrowdBudget Algorithm

As mentioned in Section 1, it is a requirement in many real–world applications to have the number of users assigned to each task *a priori* set up. Thus, we design CrowdBudget within this spirit as follows (for the pseudo code, see Algorithm 1).

Recall that n_k denotes the number of users the agent aims to assign to task k. To determine this value, it first pre–sets $n_k = \left\lfloor \frac{B}{c_k^2 \sum_{j=1}^{K} \frac{1}{c_j}} \right\rfloor$ (lines 4−8). The agent also maintains B^{r} that denotes the residual budget, which is initially set to be B. After each pre–set of n_k, B^{r} is decreased by $n_k c_k$. Next, if $B^{\mathrm{r}} > 0$, the agent sequentially increases the number of

[2]There are crowdsourcing systems where we can deterministically assign tasks to specific users. The main advantage of these systems is that we might be able to use user profiles in order to achieve good estimations even with a few answers. The analysis of such systems, however, remains as future work.

[3]Note that the tasks are binary.

Algorithm 1 CrowdBudget

1: **Inputs**: number of tasks K, budget B, allocation costs $c_1, \ldots, c_K$;
2: **Outputs**: number of allocations $n_1, \ldots, n_K$;
3: Parameter setting: $\forall k : n_k = 0$, $B^{\mathrm{r}} = B$;
4: Initial phase:
5: **for** $k = 1 \rightarrow K$ **do**
6: $\quad n_k = \left\lfloor \frac{B}{c_k^2 \sum_{j=1}^{K} \frac{1}{c_j}} \right\rfloor$;
7: $\quad B^{\mathrm{r}} = B^{\mathrm{r}} - n_k c_k$;
8: **end for**
9: Residual phase:
10: $k = 1$;
11: **while** $B^{\mathrm{r}} \geq 0$ and $k \leq K$ **do**
12: $\quad n_k = n_k + 1$, $B^{\mathrm{r}} = B^{\mathrm{r}} - c_k$, $k = k + 1$;
13: **end while**
14: Fusion phase:
15: Allocate n_k users to each task k;
16: Use MV–efficient method to estimate answers;

allocated users for each task k by 1, until the original budget is exceeded (see lines $9-13$). This phase guarantees that the budget is fully used. Note that if the residual budget is positive after the initial phase, the number of users we allocate to task k is

$$n_k = \left\lfloor \frac{B}{c_k^2 \sum_{j=1}^{K} \frac{1}{c_j}} \right\rfloor + 1. \tag{6}$$

Following this, the agent redundantly submits the tasks to the system, and once it receives the responses from the users, it uses an MV–efficient fusion method to estimate the answers to each of the tasks (lines $14-16$). We now show that with the aforementioned setting of n_k, CrowdBudget does not exceed the total budget B (i.e. $\sum n_k c_k \leq B$). In particular, after the initial phase $B^{\mathrm{r}} = B - \sum_{j=1}^{K} c_j$. Thus, we have:

$$\begin{aligned} \sum_{k=1}^{K} n_k c_k &\leq \sum_{k=1}^{K} \left(\frac{B^{\mathrm{r}}}{c_k^2 \sum_{j=1}^{K} \frac{1}{c_j}} + 1 \right) c_k \\ &= \left(\sum_{k=1}^{K} \frac{1}{c_k} \right) \left(\frac{B^{\mathrm{r}}}{\sum_{j=1}^{K} \frac{1}{c_j}} \right) + \sum_{j=1}^{K} c_j = B. \end{aligned}$$

Now, the intuition behind the specific budget allocation of CrowdBudget is as follows. By using the Hoeffding–Azuma concentration bounds [4] on the probability of estimation error for each single task defined in Equation 3, we can reduce our objective described in Equation 5, which is stochastic constraint optimisation problem, into a non–stochastic model (see the appendix for more details). By solving the fractional version of this non–stochastic optimisation problem, we get that the optimal (fractional) budget allocation is $n_k^* = \frac{B}{c_k^2 \sum_{j=1}^{K} \frac{1}{c_j}}$. Given this, by setting n_k as described in Equation 6 (which is the rounded value of the optimal solution for the fractional problem), we achieve efficient near–optimal estimation error.

4.2 Performance Analysis

We now turn to the analysis of CrowdBudget's performance. Specifically, we aim to provide theoretical guarantees on the estimation accuracy of the algorithm. To do so, we introduce the following terms. Let $d_{\min} = \min_j \left| \mu_j - \frac{1}{2} \right|$. In addition, let $c_{\min} = \min_j c_j$ and $c_{\max} = \max_j c_j$ denote the smallest and largest costs, respectively. For the sake of simplicity, we hereafter assume that Equation 1 holds for each task k if not stated otherwise. Given this, we state the following:

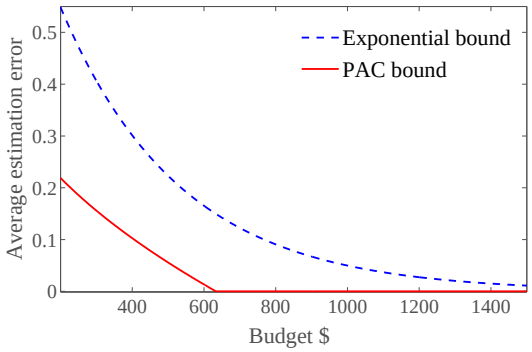

Figure 1: Example values of the upper bounds.

THEOREM 1 (MAIN RESULT 1). *For any $B \geq \sum_{j=1}^{K} c_j$, the expected estimation error of* CrowdBudget *is bounded as follows:*

$$\mathbb{E}\left[\Delta\left(B\right)\right] \leq K e^{\frac{-2Bd_{\min}^2}{c_{\max}^2 \sum_{j=1}^{K} \frac{1}{c_j}}}.$$

That is, the expected total estimation error exponentially decreases as we increase the budget. Now, we improve the error bound above by investigating the performance of CrowdBudget from a PAC (probably approximately correct) perspective [11]. That is, we aim to provide a more efficient bounds which only holds with a high probability. In particular, we state the following:

THEOREM 2 (MAIN RESULT 2). *For any $B \geq \sum_{j=1}^{K} c_j$ and $0 < \beta < 1$, the following bound holds for the expected estimation error of* CrowdBudget *with at least $(1-\beta)^K$ probability:*

$$\mathbb{E}\left[\Delta\left(B\right)\right] \leq \max\left\{0, \frac{K}{2} - d_{\min}\sqrt{\frac{2B}{-\ln\frac{\beta}{2}} \sum_{j=1}^{K} \frac{1}{c_j}}\right\}$$

It can easily be shown that this PAC bound outperforms the exponential bound. In addition, to demonstrate that these bounds are indeed efficient (i.e. they guarantee low expected estimation error), we depict the typical values of these bounds in Figure 1. Here, we set the parameters as follows: $K = 100$, $c_k = \$0.6$ for each k, $\beta = 0.001$, and $d_{\min} = 0.3$. These values are the average values of the parameter settings from our experiments (see Section '5 for more details). From this figure, we can see that from $B \geq \$600$ (i.e. at least 10 users are assigned to each task on average), the average estimation error per task of CrowdBudget is below 0.2, according to the exponential bound. On the other hand, with the PAC bound, we can guarantee with at least 90% probability (since $(1-\beta)^K \approx 0.9047$) that the average estimation error is 0 (i.e. we can accurately estimate all the tasks). This implies that CrowdBudget can guarantee low estimation error with low budget (i.e. low average number of assigned users per task). Note that in real–world systems such as Amazon Mechanical Turk or GalaxyZoo, the average number of users that existing methods typically assign to a task is significantly larger than 30 in order to efficiently estimate the answers [5, 10].

In addition, we show that the results in Theorems 1 and 2 advance the state–of–the–art result of Karger *et al.* [6]. Note

that existing work (and thus Karger *et al.* as well) typically assume that the cost is uniform. Hence, we need to consider the bounds in the case of tasks with uniform costs (i.e. each task has the same cost). Within this case, it is easy to see that the number of users CrowdBudget assigns per task is uniform. Given this, let n and c denote the number of assigned users and cost per task. Note that $n = \frac{B}{Kc}$. Thus, $\Delta(n)$ can be regarded as the *average estimation error* (i.e. per each task) of CrowdBudget (since all the task have the same number of assigned users). Given this, theorems 1 and 2 can be reduced to the following corollary:

COROLLARY 3. *For the case of uniform cost, the expected error estimation per task of* CrowdBudget *for any $n > 0$ and $K > 0$ is at most*

$$\mathbb{E}[\Delta(n)] \leq e^{-2nd_{\min}^2}.$$

Furthermore, for any $0 < \beta < 1$, we can guarantee that with at least $(1-\beta)^K$ probability, the following holds:

$$\mathbb{E}[\Delta(n)] \leq \max\left\{0, \frac{1}{2} - d_{\min}\sqrt{\frac{2n}{-\ln\frac{\beta}{2}}}\right\}$$

These results show that our algorithm can also achieve low estimation error in the case of uniform cost. Note that the exponential bound is similar to that of Karger *et al.*, but it holds for any finite K as well. In contrast, Karger *et al.*'s result is correct only in an asymptotic manner (i.e. K has to converge to infinity). In addition, the PAC bound is tighter than its exponential counterpart (see Figure 1). Thus, both bounds indeed advance the state–of–the–art results.

In what follows, we prove that if the budget is sufficiently large, we can assure that accurate estimation can be guaranteed with high probability. In particular, we have:

THEOREM 4 (MAIN RESULT 3). *Let $0 < \beta < 1$. Suppose that*

$$B \geq \frac{-\ln\beta}{2d_{\min}^2}\sum_{j=1}^{K}\frac{c_{\max}^2}{c_j}.$$

Given this, by using CrowdBudget*, $\mathbb{E}[\Delta(B)] = 0$ with at least $(1-\beta)^K$ probability.*

That is, we can accurately estimate the ground truth with high probability if the budget is sufficiently large. In particular, if we want to increase the probability of accurate estimation (i.e. the estimation error is 0), then we need to decrease the value of β. This implies an increase in the value of budget B. Note that we sketch the proofs of all the theorems and corollaries in Section 6.

5. EXPERIMENTAL EVALUATION

While we have so far developed theoretical upper bounds for the expected total estimation error of CrowdBudget, we now turn to practical aspects and examine its performance in a realistic setting. Doing so allows us to investigate whether the algorithm achieves a low estimation error when applied to practical crowdsourcing systems. To this end, we run the algorithm on a range of classification tasks, using data from a real–world application and simulation, and compare its results with a number of benchmarks. Specifically, with real–world data, we demonstrate that our algorithm outperforms the state–of–the–art in practice. On the other hand, the simulation data allows us to better understand the behaviour of CrowdBudget, as we can finely vary the parameter settings in such datasets. In what follows, we first describe our benchmark algorithms. We then describe the experimental setting and detail the results.

		Votes		*Debiased*		*Class*	
Galaxy ID	N_{votes}	E	CS	E_{deb}	CS_{deb}	E_{c}	CS_{c}
2704605332	52	.88	.08	.88	.08	1	0
1093533953	55	.81	.07	.81	.07	1	0
1093599373	61	.33	.60	.06	.86	0	1
1080819805	55	.27	.71	.08	.89	0	1
1075183799	54	.18	.76	.04	.89	0	1
556728347	32	.81	.09	.72	.18	1	0
2704212289	34	.26	.70	.04	.92	0	1
109359942	36	.42	.56	.11	.85	0	1
556728347	57	.39	.58	.11	.84	0	1
2704212289	33	.27	.72	.06	.94	0	1
1093599453	65	.33	.61	.07	.87	0	1
1630863601	75	.24	.70	.05	.93	0	1
1093599326	38	.89	.08	.86	.11	1	0
162922504	36	.41	.55	.21	.76	0	1
1093599538	32	.81	.19	.81	.19	1	0

Table 1: Classification of SDSS data release 7 galaxies in Galaxy Zoo.

5.1 Benchmarks

To show that our algorithm outperforms other budget allocation strategies, we compare it against the following methods:

- *Uniform*: This algorithm allocates the *same number* of users to each task. If some residual budget remains, it uniformly allocates the residual budget across the set of tasks. That is, for each k, we have

$$n_k = \left\lfloor \frac{B}{\sum_{j=1}^{K} c_j} \right\rfloor \quad \text{or} \quad n_k = \left\lfloor \frac{B}{\sum_{j=1}^{K} c_j} \right\rfloor + 1.$$

  Since the existing task allocation algorithms also uniformly assign users to each task (see Section 2 for more details), this algorithm can be regarded as a general representation of state–of–the–art approaches in this domain.

- *Random*: This algorithm splits the budget among the tasks in a random way. In particular, it randomly samples K (i.e. the number of tasks) times from the uniform distribution $U[1, 10]$. Let $w_1, w_2, \ldots, wK$ denote these values. Following this, it assigns budget $B_k = \frac{w_k B}{\sum_{j=1}^{K} w_j}$ to each task k. It can be regarded as a representation of algorithms that do not intelligently divide the budget among the tasks.

To have a fair comparison, we use the same underlying fusion method for our algorithm and the benchmarks as well. For the sake of simplicity, we run the majority voting rule in our experiments [4].

5.2 Experiments on Real-World Data

To test our algorithm on realistic settings, we use test data of crowdsourced votes for celestial objects provided by Galaxy Zoo [7]. In more detail, Galaxy Zoo is a citizen science project that classifies images of galaxies by collecting multiple votes from non–experts. Specifically, high resolution images are taken from a repository of more than 900,000

[4]The results with other methods show similar broad view.

galaxies collected by the Sloan Digital Sky Survey (SDSS) and users classify the images as either elliptical (E) or a spiral (CS) (or unknown) galaxy based on the shape and the size of the object (after a preliminary visualisation of some training samples). We used the Galaxy Zoo dataset for the SDSS image release 7. From this dataset, we randomly selected a subset of 700 galaxies with more than 30 votes that were classified (i.e. at least 80% of its votes have the same response) after applying Bamford *et al.*'s technique of debiasing the votes according to image features referring to the morphology and the colour of the galaxy extracted with automatic image processing (see [7] for more details). The reason for choosing these galaxies is twofold: (i) the result of agreement can be regarded as the ground truth for a particular galaxy; and (ii) the number of votes per galaxy is large enough to build up a detailed empirical distribution of responses. Table 1 reports a sample of some galaxies from our test set described by the SDSS galaxy ID, number of votes, the percentages of votes for E and CS (the rest of the votes are for unknown answers), the debiased percentages for the votes E_{deb} and CS_{deb}, and the answers [5].

Now, since users participate Galaxy Zoo on a voluntary basis, they do not receive payments for completing tasks. However, due to its large and detailed dataset of user responses, we still use it to test the efficiency of our approach. To do so, we introduce the cost of allocating tasks to users as follows. To build a realistic cost function for getting tasks completed (i.e. getting a response for an image), we again use Bamford *et al.*'s debiasing technique to divide the galaxies into four groups based on their difficulty (i.e. how difficult they are for a typical user to classify). Note that this division indeed reflects the realistic values of the galaxies' difficulty level [7]. We then assign a price chosen from array [\$0.2, \$0.5, \$0.7, \$1] such that easier tasks are less costly. The intuition here is that harder tasks need higher compensation. In addition, note that for each classified galaxy, more than 80% of the responses are the same (due to the 80% threshold of agreement). That is, any response sampled from the given distribution of the votes would give the correct answer with at least 80% success rate. This is not realistic as it is unlikely that a single user can be this accurate in voting the galaxies. Given this, we added a noise function to the answers sampled from the empirical distributions of Galaxy Zoo in order to make the settings more realistic. In particular, this noise function adds a decreasing noise, guaranteeing a progressive improvement of accuracy as the level of redundancy of votes increases.

Now, we analyse the behaviour of each algorithm in different scenarios by varying the budget B, using the aforementioned parameter settings. The results are depicted in Figure 2. As we can see from the results, CrowdBudget provides the best performance. In more detail, it outperforms the benchmarks by up to 40% when the budget is small or moderately large (i.e. $B \leq$ \$1150). This implies that CrowdBudget can allocate the tasks in a more efficient way in the case of restricted budget. However, as the budget is increased, the performance of Uniform and Random converges to that of CrowdBudget. This is due to the fact that with large budgets, both benchmark algorithms can also allocate sufficiently large number of users to each of the tasks, and thus, can achieve low total estimation error.

[5]The columns are the percentages of votes for elliptical (E) and spiral (CS) categories, the same percentages after the debasing (E_{deb}, CS_{deb}), and the 80% classification flags (E_{c}, CS_{c}) based on the debased votes.

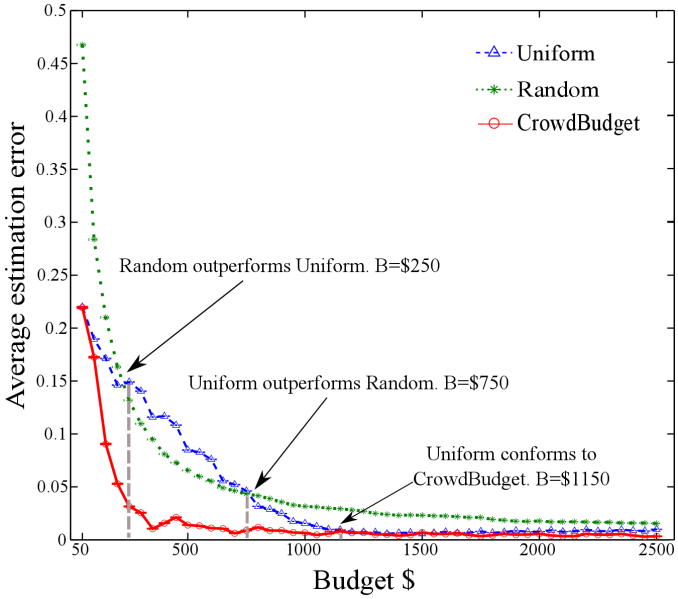

Figure 2: Average estimation error per task of the three budget allocation algorithms on the Galaxy Zoo dataset with 700 tasks and a total budget varying between \$50 and \$2500.

In addition, we can observe that for a subset of moderately large budget size (i.e. \$250 $\leq B \leq$ \$750), Random outperforms the Uniform budget allocation. In particular, it is easy to show that on average, Random allocates the same size of budget to all the tasks (as its random weights are sampled from the same uniform distribution). Since the costs for completing a classification task vary, Random allocates more users to cheaper tasks, which are typically easier (i.e. need fewer responses to achieve correct estimation). In the case of $B \leq$ \$750, the budget is not too large such that Uniform could sufficiently allocate enough users to each task, and thus, all of them might suffer from high estimation error. In contrast, Random can achieve a lower estimation error by efficiently allocating higher number of users to easier tasks. This, however, does not hold for $B \leq$ \$250, as Random does not have enough budget to allocate a sufficient number of users to the tasks.

5.3 Experiments on Synthetic Data

To better understand the behaviour of CrowdBudget, we also run it on a set of synthetic data as this allows us to freely set the parameters in order to thoroughly study its performance. In particular, recall that more difficult tasks might need more responses from the users in order to get the correct estimation. Given this, we aim to investigate how different levels of task difficulty might affect the performance of our algorithm. To this end, we choose three scenarios: (i) small budget case with $B =$ \$100; (ii) moderately large budget with $B =$ \$200, and (iii) large budget with $B =$ \$400. The intuition behind these choices is that they present three typical cases we observed in the previous experiments, namely: (i) CrowdBudget is the best, but Uniform outperforms Random; (ii) CrowdBudget still outperforms the others, but Random comes second; and (iii) all the algorithms have the same performance, due to sufficiently large budget size. For all of the three scenarios, we

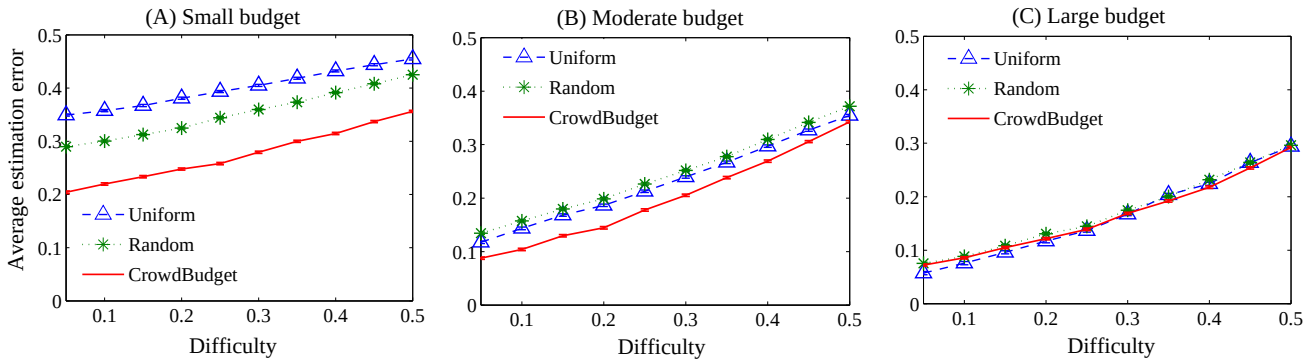

Figure 3: Average estimation error of the three task allocation algorithms on a synthetic dataset with (A) small, (B) moderately large, and (C) large budgets. The average difficulty varies between 0.05 and 0.5.

set the number of tasks to $K = 100$. The cost of each task is set to be the same as in the previous section.

To estimate the difficulty of task k, we take the difference of μ_k (i.e. the expected value of the response) from t_k (i.e. the ground truth). The larger this distance the harder a task can be regarded (i.e. the greater the number of responses needed to achieve the correct answer). Within our settings, we vary the average difficulty value from 0.05 (very easy tasks) to 0.5 (very hard tasks, where the responses can be regarded as total random choices).

The results are depicted in Figure 3. In more detail, we can observe that if the budget is fixed and the average difficulty level is increased, the performance of all the algorithms decreases. However, as the budget is increased, CrowdBudget (and the others as well) can allocate sufficient number of users to each task, and thus, it can achieve low estimation error on average, even for very hard tasks (see Figure 3c). We can also see that if the task is very hard (i.e. the difficulty distance is larger than 0.4), the performance of our algorithm does not significantly differ from that of the benchmarks (Figures 3a – 3c). In sum, our algorithm significantly outperforms the others if the budget is moderately large and the average difficulty is not very hard, which is a typical scenario in many real–world systems [5, 10].

6. CONCLUSIONS AND FUTURE WORK

We introduced the problem of budget allocation for crowdsourcing classification tasks with different classification costs. In particular, we aim to minimise the total estimation error of the answers with respect to a budget limit. To solve this problem, we proposed a budget allocation algorithm, CrowdBudget, that efficiently divides a given budget among the tasks such that the total estimation error is proveably bounded with $e^{O(-B)}$. We then improved this upper bound to $\max\left\{0, \frac{K}{2} - O\left(\sqrt{B}\right)\right\}$ by relaxing to a PAC bound. We also showed that our results advance the best known theoretical results, as we provided tighter bounds, that also hold for finite number of tasks. Finally, we demonstrated that our algorithm outperforms state–of–the–art methods by up to 40% in simulations using real–world data. As a result, our work could potentially form an efficient basis to real–world systems, such as Amazon Mechanical Turk, oDesk or CrowdFlower, where taskmasters have to efficiently crowdsource a set of classification tasks without exceeding a certain budget. This is now an ongoing discussion.

Note that currently we have not addressed non–binary classifications, nor systems with explicit task allocation (i.e. where the taskmaster can explicitly choose the users to whom he/she wants to assign the tasks). In particular, extending our results to such cases is not trivial, as the techniques we used here do not support such models. Given this, we intend to provide further analysis to these settings in future work.

APPENDIX: PROOFS

In this section, we provide the proofs for the theorems and corollary stated in Section 4. In so doing, we use the definitions and terms introduced in Sections 3 and 4. From the definition of MV–efficiency, we can easily derive that for any MV–efficient fusion method, $\mathbb{E}\left[\Delta\left(B\right)\right] \leq \mathbb{E}\left[\Delta^{\mathrm{MV}}\left(B\right)\right]$ where $\mathbb{E}\left[\Delta^{\mathrm{MV}}\left(B\right)\right]$ is the expected estimation error of the majority voting. Thus, hereafter we only focus on the majority voting method during the proofs, and for the sake of simplicity, we omit the superscript MV.

Proof sketch of Theorem 1. Consider the estimation $\hat{r}\left(n_k\right)$ of the majority voting rule for task k. Without loss of generality, we assume that $t_k = 0$ (the proof works the same for the case $t_k = 1$). In this case, $\mathbb{E}\left[\Delta\left(n_k\right)\right] = P\left\{\hat{r}\left(n_k\right) = 1\right\}$. By definition, $\hat{r}\left(n_k\right) = 1$ if and only if $\frac{1}{n_k}\sum_{j=1}^{n_k} r_k\left(u_j\right) \geq \frac{1}{2}$ where $r_k\left(u_1\right), \ldots, r_i\left(u_{n_k}\right)$ are the corresponding responses from the crowd. From the assumption defined in Equation 1, we have $\mu_k < \frac{1}{2}$. Let $d_k = \frac{1}{2} - \mu_k$. From the Hoeffding–Azuma concentration bounds [4] we have:

$$P\left\{\frac{1}{n_k}\sum_{j=1}^{n_k} r_k\left(u_j\right) \geq \frac{1}{2}\right\} = P\left\{\frac{1}{n_k}\sum_{j=1}^{n_k} r_k\left(u_j\right) - \mu_k \geq d_k\right\} \leq e^{-2n_k d_k^2}. \quad (7)$$

From Equation 6, we get

$$n_k = 1 + \left\lfloor \frac{B}{c_k^2 \sum_{j=1}^{K} \frac{1}{c_j}} \right\rfloor \geq \frac{B}{c_k^2 \sum_{j=1}^{K} \frac{1}{c_j}}. \quad (8)$$

Recall that $d_{min} = \min_k \left|\mu_k - \frac{1}{2}\right|$. Thus, we have $d_k \geq d_{min}$. In addition, $c_k \leq c_{\max}$. Substituting these inequalities and Equation 8 into Equation 7 we achieve:

$$P\left\{\frac{1}{n_k}\sum_{j=1}^{n_k} r_k\left(u_j\right) \geq \frac{1}{2}\right\} \leq e^{\frac{-2Bd_{\min}^2}{c_{\max}^2 \sum_{j=1}^{K} \frac{1}{c_j}}}.$$

By summing this up over K tasks we conclude the proof. □

Proof sketch of Theorem 2. Again, we can assume without loss of generality that $t_k = 0$ for all k. For any

$\varepsilon_k > 0$, the Hoeffding–Azuma concentration bounds imply:

$$P\left\{\left|\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) - \mu_k \geq \varepsilon_k\right|\right\} \leq 2e^{-2n_k\varepsilon_k^2}. \quad (9)$$

Now, let $\varepsilon_k = \sqrt{\frac{-\ln\frac{\beta}{2}}{2n_k}}$ for a chosen $0 < \beta < 1$. Equation 9 implies that with at least $(1-\beta)^K$ probability, the following holds for all k:

$$\left|\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) - \mu_k\right| \leq \sqrt{\frac{-\ln\frac{\beta}{2}}{2n_k}} = \varepsilon_k. \quad (10)$$

This indicates that with at least $(1-\beta)^K$ probability, the average of the responses for task k is within the radius of ε_k around μ_k. From now on we only focus on this case. Recall that $\mathbb{E}[\Delta(n_k)] = P\{\hat{r}(n_k) = 1\}$ and $\hat{r}(n_k) = 1$ if and only if $\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) \geq \frac{1}{2}$. Since Equation 10 holds for all k and $\mu_k < \frac{1}{2}$ (see Section 3 for more details), we have:

$$\mathbb{E}[\Delta(n_k)] = P\left\{\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) \geq \frac{1}{2}\right\} = \max\left\{0, \frac{\varepsilon_k - d_k}{2\varepsilon_k}\right\}. \quad (11)$$

In particular, since $\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) \in [\mu_k - \varepsilon_k, \mu_k + \varepsilon_k]$, we have the following scenarios. If $\mu_k + \varepsilon_k < \frac{1}{2}$, we have $\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) < \frac{1}{2}$, and thus, the estimation is always correct (note that $t_k = 0$). On the other hand, if $\mu_k + \varepsilon_k \geq \frac{1}{2}$, then there is a chance to get the incorrect estimation. Specifically, the probability of getting the wrong answer is $\frac{\varepsilon_k - d_k}{2\varepsilon_k}$. Now, we give an upper bound to this probability. Note that

$$\frac{\varepsilon_k - d_k}{2\varepsilon_k} = \frac{1}{2} - d_k\sqrt{\frac{2n_k}{-\ln\frac{\beta}{2}}} \leq \frac{1}{2} - d_{\min}\sqrt{\frac{2n_k}{-\ln\frac{\beta}{2}}}. \quad (12)$$

Substituting this into Equation 11 and summing up over K tasks, we get:

$$\mathbb{E}[\Delta(B)] \leq \max\left\{0, \frac{K}{2} - \sum_{k=1}^{K} d_{\min}\sqrt{\frac{2n_k}{-\ln\frac{\beta}{2}}}\right\}. \quad (13)$$

In what follows, we aim to minimise the second term on the RHS of Equation 13. In so doing, we consider the following constraint optimisation problem:

$$\max \sum_{k=1}^{K} d_{\min}\sqrt{\frac{2n_k}{-\ln\frac{\beta}{2}}} \text{ s. t. } \sum_{k=1}^{K} n_k c_k \leq B. \quad (14)$$

By relaxing the problem by allowing n_k to be fractional, we can easily show (e.g. by using the Lagrangian relaxation method) that the optimal solution of this fractional problem is $n_k^* = \frac{B}{c_k^2\sum_{j=1}^{K}\frac{1}{c_j}}$. However, this cannot be the solution for our problem as n_k cannot be fractional. Nevertheless, it can also easily shown that by assigning values to n_k as described in Equation 6, we can achieve near–optimal solution. Substituting Equation 6 into Equation 13 concludes the proof. □

Proof sketch of Corollary 3. By setting $c_k = c$ for all k, we have that $n_k = n$ (i.e. the number of assigned users per task is the same for all k). Furthermore, we have $n = \frac{B}{Kc}$. Substituting this into the bounds of Theorems 1 and 2 concludes the proof. □

Proof sketch of Theorem 4. Suppose that $0 < \beta < 1$. Recall that with at least $(1-\beta)^K$ probability, we have:

$$\left|\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) - \mu_k\right| \leq \varepsilon_k, \quad (15)$$

where $\varepsilon_k = \sqrt{\frac{-\ln\frac{\beta}{2}}{2n_k}}$. Now, for $B \geq \frac{-\ln\beta}{2d_{\min}^2}\sum_{j=1}^{K}\frac{c_{\max}^2}{c_j}$, we have $\varepsilon_k < d_k$ for all k. Thus, with at least $(1-\beta)^K$ probability, we have $\mathbb{E}[\Delta(n_k)] = P\left\{\frac{1}{n_k}\sum_{j=1}^{n_k} r_k(u_j) \geq \frac{1}{2}\right\} = 0$. By summing up over K, we conclude the proof. □

ACKNOWLEDGEMENTS

This work was funded by the ORCHID project (`http://www.orchid.ac.uk/`).

7. REFERENCES

[1] Y. Bachrach, T. Graepel, G. Kasneci, M. Kosinski, and J. V. Gael. Crowd iq - aggregating opinions to boost performance. *Autonomous Agent and Multi Agent Systems (AAMAS)*, 2012.

[2] P. Dai, Mausam, and D. S. Weld. Artificial intelligence for artificial artificial intelligence. *AAAI*, 2011.

[3] M. Ebden, T. D. Huynh, L. Moreau, S. Ramchurn, and S. Roberts. Network analysis on provenance graphs from a crowdsourcing application. *International Provenance and Annotation Workshop*, 2012.

[4] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 58(301):13–30, 1963.

[5] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large–scale crowdsourcing. *Autonomous Agent and Multi Agent Systems (AAMAS)*, 2012.

[6] D. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. *Neural Information Processing Systems (NIPS)*, 2011.

[7] C. Lintott, K. Schawinski, S. Bamford, and A. Slosar. Galaxy zoo 1: Data release of morphological classification for nearly 900,000 galaxies. *arXiv: 1007.3265*, [astro-ph.GA], 2010.

[8] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[9] A. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: Algorithms for ltering data with humans. *Very Large Data Bases (ACM VLDB)*, 2010.

[10] E. Simpson, S. Roberts, I. Psorakis, A. Smith, and C. Lintott. Bayesian combination of multiple imperfect classifiers. *Neural Information Processing Systems (NIPS)*, 2011.

[11] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11), 1984.

[12] P. Welinder, B. S., S. Belongie, and P. Perona. The multidimensional wisdom of crowds. *Neural Information Processing Systems (NIPS)*, 2010.

[13] H. Zhang, E. Law, K. Gajos, R. Miller, D. Parkes, and E. Horvitz. Human computation tasks with global constraints. *ACM Conference on Human Factors in Computing Systems (CHI)*, 2012.