Adaptive Task Assignment for Crowdsourced Classification

Chien-Ju Ho, Shahin Jabbari CJHO@CS.UCLA.EDU, SHAHIN@CS.UCLA.EDU
University of California, Los Angeles

Jennifer Wortman Vaughan JENN@MICROSOFT.COM
Microsoft Research, New York City and University of California, Los Angeles

Abstract

Crowdsourcing markets have gained popularity as a tool for inexpensively collecting data from diverse populations of workers. Classification tasks, in which workers provide labels (such as "offensive" or "not offensive") for instances (such as "websites"), are among the most common tasks posted, but due to human error and the prevalence of spam, the labels collected are often noisy. This problem is typically addressed by collecting labels for each instance from multiple workers and combining them in a clever way, but the question of how to choose which tasks to assign to each worker is often overlooked. We investigate the problem of task assignment and label inference for heterogeneous classification tasks. By applying online primal-dual techniques, we derive a provably near-optimal adaptive assignment algorithm. We show that adaptively assigning workers to tasks can lead to more accurate predictions at a lower cost when the available workers are diverse.

1. Introduction

Crowdsourcing markets provide a platform for inexpensively harnessing human computation power to solve tasks that are notoriously difficult for computers. In a typical crowdsourcing market, such as Amazon Mechanical Turk, registered users may post their own "microtasks" which are completed by workers in exchange for a small payment, usually around ten cents. A microtask may involve, for example, verifying the phone number of a business, determining whether or not an image contains a tree, or determining (subjectively) whether or not a particular website is offensive.

The availability of diverse workers willing to complete tasks inexpensively makes crowdsourcing markets appealing as tools for collecting data (Wah et al., 2011). Classification tasks, in which workers are asked to provide a binary label for an instance, are among the most common tasks posted (Ipeirotis, 2010). Unfortunately, due to a mix of human error, carelessness, and fraud — the existence of spammy workers on Mechanical Turk is widely acknowledged — the data collected is often noisy (Kittur et al., 2008; Wais et al., 2010). For classification tasks, this problem can be overcome by collecting labels for each instance from multiple workers and combining these to infer the true label. Indeed, much recent research has focused on developing algorithms for combining labels from heterogeneous labelers (Dekel & Shamir, 2009; Ipeirotis et al., 2010). However, this research has typically focused on the inference problem, sidestepping the question of how to assign workers to tasks by assuming that the learner has no control over the assignment. One exception is the work of Karger et al. (2011a;b), who focus on the situation in which all tasks are homogeneous (i.e., equally difficult and not requiring specialized skills), in which case they show that it is not possible to do better than using a random assignment.

One might expect the assignment to matter more when the tasks are heterogeneous. Classifying images of dogs versus images of cats is likely easier for the average worker than classifying images of Welsh Terriers versus images of Airedale Terriers. It might be necessary to assign more workers to tasks of the latter type to produce high confidence labels. The assignment can also be important when tasks require specialized skills. A worker who knows little about dogs may not be able to produce high quality labels for the Terrier task, but may have skills that are applicable elsewhere.

We investigate the problem of task assignment and label inference for heterogeneous classification tasks. In

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

our model, a task requester has a set of tasks, each of which consists of an instance for which he would like to infer a binary label. Workers arrive online. The learner must decide which tasks to assign to each worker, and then use the noisy labels produced by the workers to infer the true label for each task. The goal of the learner is to output a set of labels with sufficiently low error while requesting as few labels from workers as possible. Building on online primal-dual methods (Buchbinder & Naor, 2005), we propose an exploration-exploitation algorithm that is provably competitive with an optimal offline algorithm that has knowledge of the sequence of workers and their skills in advance. We then evaluate this algorithm in a variety of experiments on synthetic data and show that adaptively allocating tasks helps when the worker distribution is diverse or the tasks are heterogeneous.

2. Related Work

Our research is mostly closely related to that of Karger et al. (2011a;b) and Ho & Vaughan (2012). Karger et al. introduced a model in which a requester has a set of homogeneous labeling tasks he must assign to workers who arrive online. They proposed an assignment algorithm based on random graph generation and a message-passing inference algorithm inspired by belief propagation, and showed that their technique is order-optimal in terms of labeling budget. In particular, let p_j be the probability that worker j completes any given task correctly and $q = \mathbb{E}[(2p_j - 1)^2]$, where the expectation is over the choice of a random worker j. They proved that their algorithm requires $O((1/q)\log(1/\epsilon))$ labels per task to achieve error less than ϵ in the limit as the numbers of tasks and workers go to infinity. They also showed that adaptively assigning tasks does not help in their setting, in that $\Omega((1/q)\log(1/\epsilon))$ labels are still needed in general.

We generalize this model to allow heterogeneous tasks, so that the probability that worker j completes a task correctly may depend on the particular task. In this generalized setting, assigning tasks adaptively can provide an advantage both in theory and in practice.

Our techniques build on the online primal-dual framework, which has been used to analyze online optimization problems ranging from the adwords problem (Buchbinder et al., 2007; Devanur et al., 2011) to network optimization (Alon et al., 2004) and paging (Bansal et al., 2007). Ho & Vaughan (2012) were the first to apply this framework to crowdsourcing. In their model, a requester has a fixed set of tasks of different types, each of which must be completed exactly once. Each worker has an unknown skill level for each type of task, with workers of higher skill levels producing higher quality work on average. Workers arrive online, and the learner must assign each worker to a single task upon arrival. When the worker completes the task, the learner immediately receives a reward, and thus also a noisy signal of the worker's skill level for tasks of that type. Workers arrive repeatedly and are identifiable, so the learner can form estimates of the workers' skill levels over time. The goal is to maximize the sum of requester rewards. Ho & Vaughan provide an algorithm based on the online primal-dual framework and prove that this algorithm is competitive with respect to the optimal offline algorithm that has access to the unknown skill levels of each worker.

Our model differs from that of Ho & Vaughan in several key ways. Their analysis depends heavily on the assumption that the requester can evaluate the quality of completed work immediately (i.e., learn his reward on each time step), which is unrealistic in many settings, including the labeling task we consider here; if the requester could quickly verify the accuracy of labels, he wouldn't need the workers' labels in the first place. In their model, each task may be assigned to a worker only once. In ours, repeated labeling is necessary since there would be no way to estimate worker quality without it. These differences require a different problem formulation and novel analysis techniques.

Repeated labeling has received considerable empirical attention, dating back to the EM-based algorithm of Dawid & Skene (1979). Sheng et al. (2008) considered a setting in which every worker is correct on every task with the same probability, and empirically evaluated how much repeated labeling helps. Ipeirotis et al. (2010) extended this idea to heterogeneous workers and provided an algorithm to simultaneously estimate workers' quality and true task labels. More recently, there has been work showing that label inference can be improved by first estimating parameters of the structure underlying the labeling process using techniques such as Bayesian learning (Welinder et al., 2010), minimax entropy (Zhou et al., 2012), and variational inference (Liu et al., 2012).

On the theoretical side, there have been several results on learning a binary classifier using labeled data contributed by multiple teachers, each of which labels instances according to his own fixed labeling function (Crammer et al., 2005; 2008; Dekel & Shamir, 2009). These require PAC-style assumptions and focus on filtering out low quality workers. Tran-Thanh et al. (2012) used ideas from the multi-armed bandit literature to assign tasks. Bandit ideas cannot be applied in our setting without further assumptions since the

reward corresponding to an assignment depends on whether the worker's label is correct, which cannot be inferred until the task has been assigned to others.

Ghosh et al. (2011) studied a model similar to that of Karger et al., also with homogeneous tasks, and used eigenvalue decomposition to estimate each worker's quality. Their bounds depend on a quantity essentially identical to the quantity q defined above, which they refer to as the population's *average competence*. A similar quantity plays a role in our analysis.

3. The Model

In our model, a task requester has a set of n tasks, indexed $1, \cdots, n$. Each task is a binary classification problem. The true label of task i, denoted ℓ_i, is either 1 or -1, and is unknown to the requester.

Workers arrive online. When worker j arrives, she announces the maximum number of tasks that she is willing to complete, her *capacity*, M_j. No other information is known about each worker when she arrives.

Each worker j has a *skill level*, $p_{i,j} \in [0, 1]$, for each task i. If the algorithm assigns worker j to task i, the worker will produce a label $\ell_{i,j}$ such that $\ell_{i,j} = \ell_i$ with probability $p_{i,j}$ and $\ell_{i,j} = -\ell_i$ with probability $1 - p_{i,j}$, independent of all other labels. The algorithm may assign worker j up to M_j tasks, and may observe her output on each task before deciding whether to assign her to another or move on, but once the algorithm moves on, it cannot access the worker again. This is meant to reflect that crowdsourced workers are neither identifiable nor persistent, so we cannot hope to identify and later reuse highly skilled workers.

Several of our results depend on the quantity $q_{i,j} = (2p_{i,j} - 1)^2$. Intuitively, when this quantity is close to 1, the label of worker j on task i will be informative; when it is close to 0, the label will be random noise.

To model the fact that the requester cannot wait arbitrarily long, we assume that he can only assign tasks to the first m workers who arrive, for some known m. We therefore index workers $1, \cdots, m$. Later we consider an additional γm workers who are used for exploration.

In addition to assigning tasks to workers, the learning algorithm must produce a final estimate $\hat{\ell}_i$ for the label ℓ_i of each task i based on the labels provided by the workers. The goal of the learner is to produce estimates that are correct with high probability while querying workers for as few labels as possible.

Task structure: A clever learning algorithm should infer the worker skill levels $p_{i,j}$ and assign workers to tasks at which they excel. If the skills are arbitrary, then the learner cannot infer them without assigning every worker to every task. Therefore, it is necessary to assume that the $p_{i,j}$ values exhibit some structure. Karger et al. (2011a;b) assume that all tasks are identical, i.e., $p_{i,j} = p_{i',j}$ for all j and all i and i'. We consider a more general setting in which the tasks can be divided into T *types*, and assume only that $p_{i,j} = p_{i',j}$ if i and i' are of the same type.

Gold standard tasks: As is common in the literature (Oleson et al., 2011), we assume that the learner has access to "gold standard" tasks of each task type.[1] These are instances for which the learner knows the true label a priori. They can be assigned in order to estimate the $p_{i,j}$ values. Of course the algorithm must pay for these "pure exploration" assignments.

Random permutation model: We analyze our algorithm in the random permutation model as in Devanur & Hayes (2009). The capacities M_j and skills $p_{i,j}$ of each worker j may be chosen adversarially, as long as the assumptions on task structure are satisfied. However, the arrival order is randomly permuted. Since only the order of workers is randomized, the offline optimal allocation is well-defined.

Competitive ratio: To evaluate our algorithm, we use the notion of *competitive ratio*, which is an upper bound on the ratio between the number of labels requested by the algorithm and the number requested by an *optimal offline algorithm* which has access to all worker capacities and skill levels, but must still assign enough workers to each task to obtain a high-confidence guess for the task's label. The optimal offline algorithm is discussed in Sections 4 and 5.

4. An Offline Problem

To gain intuition, we first consider a simplified offline version of our problem in which the learner is provided with a full description of the sequence of m workers who will arrive, including the skill levels $p_{i,j}$ and capacities M_j for all i and j. The learner must decide which tasks to assign to each worker and then infer the task labels. We discuss the inference problem first.

4.1. Aggregating Workers' Labels

Suppose that the learner has already assigned tasks to workers and observed the workers' labels for these tasks. How should the learner aggregate this information to infer the true label for each task?

[1] If gold standard tasks are not available, they can be created by assigning a small set of tasks to many workers.

We consider aggregation methods that take a weighted vote of the workers' labels. Fix a task i. Let J_i denote the set of workers assigned to this task. We consider methods that set $\hat{\ell}_i = \text{sign}(\sum_{j \in J_i} w_{i,j} \ell_{i,j})$ for some set of weights $\{w_{i,j}\}$. The following lemma shows that this technique with weights $w_{i,j} = 2p_{i,j} - 1$ is guaranteed to achieve a low error if enough high quality workers are queried. Recall that $q_{i,j} = (2p_{i,j} - 1)^2$.

Lemma 1. *Let $\hat{\ell}_i = \text{sign}(\sum_{j \in J_i} w_{i,j} \ell_{i,j})$ for some set of weights $\{w_{i,j}\}$. Then $\hat{\ell}_i \neq \ell_i$ with probability at most $e^{-\frac{1}{2}(\sum_{j \in J_i} w_{i,j}(2p_{i,j}-1))^2 / \sum_{j \in J_i} w_{i,j}^2}$. This bound is minimized when $w_{i,j} \propto (2p_{i,j} - 1)$, in which case the probability that $\hat{\ell}_i \neq \ell_i$ is at most $e^{-\frac{1}{2}\sum_{j \in J_i} q_{i,j}}$.*

The proof, which uses a simple application of Hoeffding's inequality, is in the appendix.[2] This tells us that to guarantee that we make an error with probability less than ϵ on a task i, it is sufficient to select a set of labelers J_i such that $\Sigma_{j \in J_i} q_{i,j} \geq 2\ln(1/\epsilon)$ and aggregate labels by setting $\hat{\ell}_i = \text{sign}(\sum_{j \in J_i} (2p_{i,j} - 1)\ell_{i,j})$.

One might ask if it is possible to guarantee an error of ϵ with fewer labels. In some cases, it is; if there exists an i and j such that $p_{i,j} = q_{i,j} = 1$, then one can achieve zero error with only a single label. However, in some cases this method is optimal. For this reason, we restrict our attention to algorithms that query subsets J_i such that $\Sigma_{j \in J_i} q_{i,j} \geq 2\ln(1/\epsilon)$. We use the shorthand $C_\epsilon = 2\ln(1/\epsilon)$.

4.2. Integer Programming Formulation

There is a significant benefit that comes from restricting attention to algorithms of the form described above. Let $y_{i,j}$ be a variable that is 1 if task i is assigned to worker j and 0 otherwise. The requirement that $\sum_{j \in J_i} q_{i,j} \geq C_\epsilon$ can be expressed as a linear constraint of these variables. This would not be possible using unweighted majority voting to aggregate labels; weighting by $2p_{i,j} - 1$ is key. This allows us to express the optimal offline assignment strategy as an integer linear program (IP), with variables $y_{i,j}$ for each (i, j):

$$\begin{aligned} \min \ & \Sigma_{i=1}^n \Sigma_{j=1}^m y_{i,j} \\ \text{s.t.} \ & \Sigma_{i=1}^n y_{i,j} \leq M_j \ \ \forall j & (1) \\ & \Sigma_{j=1}^m q_{i,j} y_{i,j} \geq C_\epsilon \ \ \forall i & (2) \\ & y_{i,j} \in \{0, 1\} \ \ \forall (i, j). & (3) \end{aligned}$$

Constraint (1) guarantees that worker j does not exceed her capacity. Constraint (2) guarantees that aggregation will produce the correct label of each task with high probability. Constraint (3) implies that a task is either assigned to a worker or not.

Note that there may not exist a feasible solution to this IP, in which case it would not be possible to guarantee a probability of error less than ϵ for all tasks using weighted majority voting. For most of this paper, we assume a feasible solution exists; the case in which one does not is discussed in Section 5.1.

For computational reasons, instead of working directly with this IP, we will work with a linear programming relaxation obtained by replacing the last constraint with $0 \leq y_{i,j} \leq 1 \ \forall (i, j)$; we will see below that this does not impact the solution too much.

4.3. Working with the Dual

Solving the linear program described above requires knowing the values $q_{i,j}$ for the full sequence of workers j up front. When we move to the online setting, it will be more convenient to work with the dual of the relaxed linear program, which can be written as follows, with dual variables x_i, z_j, and $t_{i,j}$ for all (i, j):

$$\begin{aligned} \max \ & C_\epsilon \Sigma_{i=1}^n x_i - \Sigma_{j=1}^m M_j z_j - \Sigma_{i=1}^n \Sigma_{j=1}^m t_{i,j} \\ \text{s.t.} \ & 1 - q_{i,j} x_i + z_j + t_{i,j} \geq 0 \ \ \forall (i, j) \\ & x_i, z_j, t_{i,j} \geq 0 \ \ \forall (i, j). \end{aligned}$$

We refer to x_i as the *task weight* for i, and define the *task value* of worker j on task i as $v_{i,j} = q_{i,j} x_i - 1$.

Suppose that we were given access to the task weights x_i for each task i and the values $q_{i,j}$. (We will discuss how to approximate these values later.) Then we could use the following algorithm to approximate the optimal primal solution. Note that to run this algorithm, it is not necessary to have access to all $q_{i,j}$ values at once; we only need information about worker j when it comes time to assign tasks to this worker. This is the advantage of working with the dual.

Algorithm 1 Primal Approximation Algorithm

Input: Values x_i and $q_{i,j}$ for all (i, j)
For every worker $j \in \{1, \ldots, m\}$, compute the task values, $v_{i,j} = q_{i,j} x_i - 1$, for all tasks i. Let n_j be the number of tasks i such that $v_{i,j} \geq 0$. If $n_j \leq M_j$, then set $y_{ij} \leftarrow 1$ for all n_j tasks with non-negative task value. Otherwise, set $y_{i,j} \leftarrow 1$ for the M_j tasks with highest task value. Set $y_{i,j} \leftarrow 0$ for all other tasks.

The following theorem shows that this algorithm produces a near-optimal primal solution to our original IP when given as input the optimal dual solution for the relaxed LP. The condition that $q_{i,j} x_i^* \neq q_{i',j} x_{i'}^*$ for all $i \neq i'$ is needed for technical reasons, but can be

[2] An appendix containing all omitted proofs and additional details can be found in the long version of this paper available on the authors' websites.

relaxed by adding small random perturbations to $q_{i,j}$ values as in Devanur et al. (2011).[3] For the rest of the paper, we assume that perturbations have been added and that the condition above holds. Call this the "perturbation assumption." In our final algorithm, we will perturb our estimates of the $q_{i,j}$ values for this reason.

Theorem 1. *Let $\boldsymbol{y}^*$ be the primal optimal of the IP and $\boldsymbol{x}^*$ be the dual optimal of the relaxed formulation. Let $\boldsymbol{y}$ be the output of the Primal Approximation Algorithm given input $\boldsymbol{x}^*$ and the true values $\boldsymbol{q}$. Then $\boldsymbol{y}$ is feasible in the IP, and under the perturbation assumption, $\Sigma_{i=1}^{n}\Sigma_{j=1}^{m} y_{i,j} - \Sigma_{i=1}^{n}\Sigma_{j=1}^{m} y^*_{i,j} \leq \min(m,n)$.*

The proof shows that the $y_{i,j}$ values assigned by the Primal Approximation Algorithm differ from the optimal solution of the relaxed LP for at most $\min(m,n)$ pairs (i,j), and that this implies the result.

5. Moving to the Online Setting

We have shown that, given access to $\mathbf{q}$ and the optimal task weights $\mathbf{x}^*$, the Primal Approximation Algorithm generates an assignment which is close to the optimal solution of the IP in Section 4.2. However, in the online problem that we initially set out to solve, these values are unknown. In this section, we provide methods for estimating these quantities and incorporate these into an algorithm for the online problem.

Our online algorithm combines two varieties of exploration. First, we use exploration to estimate the optimal task weights $\mathbf{x}^*$. To do this, we hire an additional γm workers on top of the m workers we originally planned to hire, for some $\gamma > 0$, and "observe" their $q_{i,j}$ values. (We will actually only estimate these values; see below.) Then, by treating these γm workers as a random sample of the population (which they effectively are under the random permutation model), we can apply online primal-dual methods and obtain estimates of the optimal task weights. These estimates can then be fed to the Primal Approximation Algorithm in order to determine assignments for the remaining m workers, as described in Section 5.1.

The second variety is used to estimate workers' skill levels. Each time a new worker arrives (including the γm extras), we require her to complete a set of gold standard tasks of each task type. Based on the labels she provides, we estimate her skill levels $p_{i,j}$ and use these to estimate the $q_{i,j}$ values. The impact of these estimates on performance is discussed in Section 5.2.

[3] Adding noise will introduce an error, but this error can be made arbitrarily small when C_ϵ is large. To simplify presentation, we do no include the error in our discussion.

If we require each worker to complete s gold standard tasks, and we hire an extra γm workers, we need to pay for an extra $(1+\gamma)ms$ assignments beyond those made by the Primal Approximation Algorithm. We precisely quantify how the number of assignments compares with the offline optimal in Section 5.3.

5.1. Estimating the Task Weights

In this section, we focus on the estimation of task weights in a simplified setting in which we can observe the quality of each worker as she arrives. To estimate the task weights, we borrow an idea from the literature on the online primal-dual framework. We use an initial sampling phase in which we hire γm workers in addition to the primary m workers, for some γ. We observe their skill levels and treat the distribution over skills of the sampled workers as an estimate of the distribution of skills of the m primary workers. Given the $q_{i,j}$ values from the sampled γm workers, we can solve an alternative linear programming problem, which is the same as our relaxed offline linear programming problem, except that m is replaced by γm and C_ϵ is replaced by γC_ϵ. Let $\hat{\mathbf{x}}^*$ be the optimal task weights in this "sampled LP" problem. We show that if ϵ is small enough, running the Primal Approximation Algorithm using $\hat{\mathbf{x}}^*$ and $\mathbf{q}$ yields a near-optimal solution, with a number of assignments close to optimal, and a prediction error close to ϵ after aggregation.

Theorem 2. *For any $\epsilon, \delta \in (0, 1/2)$, for any $\gamma = \ell/m$ with $\ell \in \{1, 2, \cdots, m\}$ and $\gamma \in [1/C_\epsilon, 1]$, let $\hat{\boldsymbol{y}}^{s,*}$ and $\hat{\boldsymbol{x}}^*$ be the primal and dual optimal solutions of the sampled LP with parameters ϵ and γ. Let $\hat{\boldsymbol{y}}^*$ be the output of the Primal Approximation Algorithm with inputs $\hat{\boldsymbol{x}}^*$ and $\boldsymbol{q}$, and let $\bar{\boldsymbol{y}}^*$ be the optimal assignment of the relaxed offline formulation with parameter ϵ. Let $q_{min} = \min_{(i,j):\hat{y}^{s,*}_{i,j}>0} q_{i,j}$. Then under the perturbation assumption, with probability at least $1-\delta$,*

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \hat{y}^*_{i,j} \leq \left(1 + \frac{\min(m,n)}{q_{min} n C_\epsilon} + \frac{35\ln(2/\delta)}{q_{min}\sqrt{\gamma C_\epsilon}}\right) \sum_{i=1}^{n}\sum_{j=1}^{m} \bar{y}^*_{i,j}.$$

If the labels collected from the resulting assignment are used to estimate the task labels via weighted majority voting, the probability that any given task label is predicted incorrectly is no more than $\epsilon^{1-6\ln(2/\delta)/\sqrt{\gamma C_\epsilon}}$.

The requirement that $\gamma \geq 1/C_\epsilon$ stems from the fact that if C_ϵ is small, the total number of assignments will also be small, and the quality of the assignment is more sensitive to estimation errors. If C_ϵ is large, small estimation errors effect the assignment less and we can set the sampling ratio to a smaller value.

In the proof, we show that the gap between the ob-

jectives of the primal solution generated by the Primal Approximation Algorithm using $\hat{\mathbf{x}}^*$ and $\mathbf{q}$ and the corresponding dual solution is exactly the summation of $\hat{x}_i^*(C_\epsilon - \Sigma_{j=1}^m q_{i,j}\hat{y}_{i,j}^*)$ over all tasks i, which is small if enough workers are sampled. By weak duality, the optimal number of assignments is between the primal and the dual objectives, so the primal solution output by the algorithm must be near-optimal.

A note on feasibility: We have implicitly assumed that the sampled LP is feasible. In practice, it may not be, or even if it is, there may exist tasks i such that $\min_{j:\hat{y}_{i,j}^{s,*}>0} q_{i,j}$ is very small, leading to a small value of q_{min}. If either of these things happen, the task requester may want to discard some of the tasks or lower his desired error, solve the sampled LP with these modified constraints, and continue from there, as there is no way to guarantee low error on all tasks.

5.2. Using Estimates of Skill Levels

We now discuss the effect of estimating worker skills. Given observations of the gold standard tasks of type τ that worker j completed, we can estimate $p_{i,j}$ for any task i of type τ as the fraction of these tasks she labeled correctly. The following lemma, follows from a straightforward application of the Hoeffding bound; we state it here as it will be useful, but omit the proof.

Lemma 2. *For any worker j, for any task type τ, and for any $t, \delta \in (0,1)$, suppose that worker j labels $\ln(2/\delta)/(2t^2)$ gold standard tasks of type τ. Then with probability at least $1-\delta$, for all tasks i of type τ, if we set $\hat{p}_{i,j}$ to the fraction of gold standard tasks of type τ answered correctly then $|p_{i,j} - \hat{p}_{i,j}| \leq t$.*

This estimate of $p_{i,j}$ can then be used to derive an estimate for $q_{i,j}$, with error bounded as follows.

Lemma 3. *For any worker j and task i, if $\hat{p}_{i,j}$ is an estimate of $p_{i,j}$ such that $|p_{i,j} - \hat{p}_{i,j}| \leq t$, and $\hat{q}_{i,j}$ is set to $(2\hat{p}_{i,j}-1)^2$, then $|q_{i,j} - \hat{q}_{i,j}| \leq 4t$.*

Of course the use of estimated values impacts performance. Consider the offline problem discussed in the previous section. One might hope that if we applied the Primal Approximation Algorithm using $\hat{\mathbf{q}}$, the number of assignments would be close to the number made using $\mathbf{q}$. Unfortunately, this is not true. Consider this toy example. Let $q_{i,1} = q_{i,2} = q_{i,3} = 1$ for all i, $q_{i,j} = 10^{-4}$ for all i and $j > 3$, and $M_j = n$ for all j. Set $\epsilon = 0.224$ so that $C_\epsilon \approx 3$. In the optimal solution, each task i should be assigned only to workers $1, 2$, and 3. If we underestimate the $q_{i,j}$ values, we could end up assigning each task to many more workers. This can be made arbitrarily bad.

To address this, instead of solving the relaxed offline formulation directly, we consider an alternative LP which is identical to the relaxed offline formulation, except that $\mathbf{q}$ is replaced with $\hat{\mathbf{q}}$ and C_ϵ is replaced with a smaller value $C_{\epsilon'}$ (corresponding to a higher allowable error ϵ'). We call this *the approximated LP*. We show that, if ϵ' is chosen properly, we can guarantee the optimal solution in the relaxed offline formulation is feasible in the approximated LP, so the optimal solution of the approximated LP will yield an assignment with fewer tasks assigned to workers than the optimal solution of the relaxed offline formulation, even though it is based on estimations.

To set ϵ', we assume the requester has a rough idea of how hard the tasks are and how inaccurate his estimates of worker skills are. The latter can be achieved by applying Lemma 2 and the union bound to find a value of t such that $|p_{i,j} - \hat{p}_{i,j}| \leq t$ for all (i,j) pairs with high probability, and setting each $\hat{q}_{i,j} = (2\hat{p}_{i,j}-1)^2$ as in Lemma 3. For the former, let $\bar{\mathbf{y}}^*$ be the optimal solution of the relaxed offline formulation. Define $\bar{q}_i^* = \Sigma_{j=1}^m q_{i,j}\bar{y}_{i,j}^* / \Sigma_{j=1}^m \bar{y}_{i,j}^*$. We assume that the requester can produce a value $\bar{q}_{min}^*$ such that $\bar{q}_{min}^* \leq \bar{q}_i^*$ for all i and then set $C_{\epsilon'} = 2\ln(1/\epsilon')$ where $\epsilon' = \epsilon^{1-4t/\bar{q}_{min}^*}$. If the requester doesn't have much information, he can conservatively set $\bar{q}_{min}^*$ much smaller than $\min_i\{\bar{q}_i^*\}$, but will both need more accurate estimates of $p_{i,j}$ and sacrifice some prediction accuracy.

Theorem 3. *Assume that we have access to a value $\bar{q}_{min}^*$ such that $\bar{q}_{min}^* \leq \bar{q}_i^*$ for all i and values $\hat{p}_{i,j}$ such that $|p_{i,j} - \hat{p}_{i,j}| \leq t$ for all (i,j) pairs for any known value $t < \bar{q}_{min}^*/4$. Then for any $\epsilon > 0$, the optimal solution of the approximated LP with parameter $\epsilon' = \epsilon^{1-4t/\bar{q}_{min}^*}$ and skill levels $\hat{q}_{i,j} = (2\hat{p}_{i,j}-1)^2$ is no bigger than the optimal solution of the relaxed offline formulation with parameter ϵ and skill levels $q_{i,j}$.*

Of course this guarantee is not free. We pay the price of decreased prediction accuracy since we are using ϵ' in place of ϵ. We also pay when it comes time to aggregate the workers' labels, since we must now use $\hat{\mathbf{q}}$ in place of $\mathbf{q}$ when applying the weighted majority voting method described in Section 4.1. This is quantified in the following theorem. Note that this theorem applies to *any* feasible integer solution of the approximated LP and therefore also the best integer solution.

Theorem 4. *Assume again that we have access to a value $\bar{q}_{min}^*$ such that $\bar{q}_{min}^* \leq \bar{q}_i^*$ for all i and values $\hat{p}_{i,j}$ such that $|p_{i,j} - \hat{p}_{i,j}| \leq t$ for all (i,j) pairs for any known value $t < \bar{q}_{min}^*/4$. For any $\epsilon > 0$, let $\boldsymbol{y}$ be any feasible integer assignment of the approximated LP with parameter $\epsilon' = \epsilon^{1-4t/\bar{q}_{min}^*}$ and skill levels $\hat{q}_{i,j} = (2\hat{p}_{i,j}-1)^2$. Let $J_i = \{j : y_{i,j} = 1\}$ denote the*

set of workers that are assigned to task i according to $\boldsymbol{y}$, and define $\hat{q}_i = \Sigma_{j \in J_i} q_{i,j} / |J_i|$. If the tasks are assigned according to $\boldsymbol{y}$ and the results aggregated using weighted majority voting with weights $w_{i,j} = 2\hat{p}_{i,j} - 1$, the error probability of the predicted task label for task i is bounded by $\epsilon^{(1-4t/\bar{q}^*_{min})(1-4t/\hat{q}_i)}$.

Theorem 4 tells us that if our estimates of the worker skills are accurate (i.e., if t is small), then our prediction error will be close to the error we would have achieved if we had known the true worker skills. How good the estimates need to be depends on the quality of the workers, as measured by $\bar{q}^*_{min}$ and $\hat{q}_i$. Intuitively, if $\bar{q}^*_{min}$ is small, there may exist some task i at which workers perform poorly in the optimal solution. In this case, the assignment will be very sensitive to the value of ϵ' chosen, and it will be necessary to set ϵ' larger to guarantee that the true optimal solution is feasible in the approximated LP. If $\hat{q}_i$ is small, then a small amount of error in estimated worker quality would dramatically change the weights used in the weighted majority voting aggregation scheme.

5.3. Putting it All Together

We have separately considered relaxations of the task assignment and label inference problem in which the optimal task weights or worker skill levels are already known. We now put all these pieces together, give a combined algorithm, and state our main theorem.

Algorithm 2 Main Algorithm

Input: Values (ϵ, γ, s, and $\bar{q}^*_{min}$)
Hire γm preliminary workers.
for each preliminary worker **do**
  Assign s gold standard tasks of each task type.
  Calculate $\hat{q}_{i,j}$ values as in Section 5.2 and perturb with a negligible amount of noise.
end for
Calculate $C_{\epsilon'}$ and solve the sampled LP with $\hat{\mathbf{q}}$ to obtain primal $\mathbf{y}^s$ and dual $\hat{\mathbf{x}}^*$ as in Section 5.1.
for each worker $j \in \{1, ..., m\}$ **do**
  Assign s gold standard tasks of each task type.
  Calculate $\hat{q}_{i,j}$ values as in Section 5.2 and perturb with a negligible amount of noise.
  Run the Primal Approximation Algorithm with inputs $\hat{\mathbf{x}}^*$ and (perturbed) $\hat{\mathbf{q}}$ to $\mathbf{y}$.
  Assign worker j to all tasks i with $y_{i,j} = 1$.
end for
Aggregate the workers' labels using weighted majority voting as in Section 4.1.

The complete algorithm is stated in Algorithm 2, and its performance guarantee is given below. Recall that T is the number of task types. Again, we assume that the optimization problems are feasible.

Theorem 5. *For any $\epsilon, \delta \in (0, 1/2)$, for any $\gamma = \ell/m$ for an $\ell \in \{1, 2, ..., m\}$ such that $\gamma \in [1/C_\epsilon, 1]$, assume we have access to a value $\bar{q}^*_{min}$ satisfying the condition in Theorem 3, let s be any integer satisfying $s \geq 8\ln(4T(1+\gamma)m/\delta)/\bar{q}^{*2}_{min}$, and let $\epsilon' = \epsilon^{1-4\sqrt{\ln(4T(1+\gamma)m/\delta)/(2s)}/\bar{q}^*_{min}}$. Then under the perturbation assumption, with probability at least $1-\delta$, when the Main Algorithm is executed with input $(\epsilon, \gamma, s, \bar{q}^*_{min})$, the following two things hold:*

1) The number of assignments of to non-gold standard tasks is no more than

$$\left(1 + \frac{\min(m,n)}{\hat{q}_{min} n C_{\epsilon'}} + \frac{35\ln(4/\delta)}{\hat{q}_{min}\sqrt{\gamma C_{\epsilon'}}}\right)$$

times the optimal objective of the IP, where $\hat{q}_{min} = \min_{(i,j):y^s_{i,j}=1} \hat{q}^s_{i,j}$.

2) The probability that the aggregated label for each task i is incorrect is bounded by $\epsilon^{(1-l_1)(1-l_2)(1-l_{3,i})}$, where $l_1 = 4t/\bar{q}^*_{min}, l_2 = 6\ln(4/\delta)/\sqrt{\gamma C_{\epsilon'}}$, $l_{3,i} = 4t/\hat{q}_i$, and $t = \sqrt{\ln(4T(1+\gamma)m/\delta)/(2s)}$.

When ϵ is small, $C_{\epsilon'}$ is large, and l_2 approaches 0. The competitive ratio may shrink, but if ϵ is too small, $\hat{q}_{min}$ will shrink as well, and at some point the problem may become infeasible. When s is large, t is small, and so l_1 and $l_{3,i}$ approach 0, leading to error almost as low as if we knew the true $\mathbf{q}$ values, as we would expect.

6. Synthetic Experiments

In this section, we evaluate the performance of our algorithm through simulations on synthetically generated data. As a comparison, we also run the message-passing inference algorithm of Karger et al. (2011a;b) on the same data sets. As described in Section 2, Karger et al. use a non-adaptive, random assignment strategy in conjunction with this inference algorithm. We show that adaptively allocating tasks to workers using our algorithm can outperform random task assignment in settings in which *(i)* the worker distribution is diverse, or *(ii)* the set of tasks is heterogeneous.

We create $n = 1,000$ tasks and $m = 300$ workers with capacity $M_j = 200$ for all j, and vary the distribution over skill levels $p_{i,j}$. We would like to compare the error rates of the algorithms when given access to the same total number of labels. In the message-passing algorithm, we can directly set the number of labels by altering the number of assignments. In our algorithm, we change the parameter ϵ and observe the number of labels (including exploration) and the prediction error.

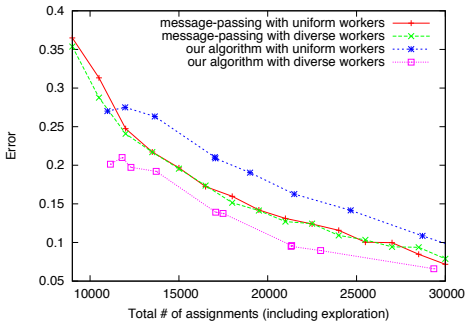

Figure 1. Uniform tasks with one or two worker types.

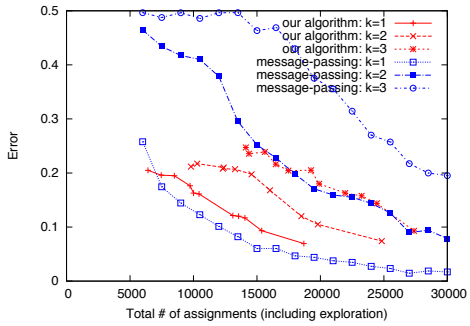

Figure 2. Heterogeneous tasks.

6.1. Worker Diversity

In their analysis, Karger et al. assume there is only one task type (that is, $p_{i,j} = p_{i',j}$ for all i, i', and j), and claim that in this setting adaptively assigning tasks does not yield much of an advantage. Our first experiment simulates this setting. We would like to see if our algorithm can perform better if the worker distribution is diverse, even though it requires some "pure exploration" — we need to pay each worker to complete the gold standard tasks, and we need to hire an extra γm workers to estimate the task weights.

For our algorithm, we set $\gamma = 0.3$ and sample 90 extra workers from the same distribution to learn task weights. Each worker is required to complete $s = 20$ gold standard tasks of each type when she arrives. These values were not optimized, and performance could likely be improved by tuning these parameters.

We examine two settings. In the first, every worker gives us a correct label with probability 0.6414 for all tasks. In the second, the population is 50% spammers and 50% hammers. The spammers give random answers, while the hammers answer correctly with probability 0.7. Note that these values are chosen such that $E[q_{i,j}] = E[(2p_{i,j} - 1)^2]$ is the same in both settings.

The results are shown in Figure 1. The performance of the message-passing algorithm is almost identical in the two settings. Our algorithm performs relatively poorly in the setting with uniform workers since we can't benefit from adaptive assignments but still pay the exploration costs. However, our algorithm outperforms message passing in the setting with two types of workers, quickly learning not to assign any tasks to the spammers beyond those used for exploration.

6.2. Heterogeneous Tasks

We next examine a setting in which there are multiple types of tasks, and every worker is skilled at exactly one type. We generate k task types and k corresponding worker types, for $k = 1$, 2, and 3. Type α workers complete type α tasks correctly with probability 0.7, but other tasks correctly with probability 0.5.

For our algorithm, we set $\gamma = 0.3$. Each worker completes $s = 10$ gold standard tasks of each type.

The results are shown in Figure 2. Not surprisingly, since the message-passing algorithm does not attempt to match tasks to suitable workers, its performance degrades quickly when k grows. Since our algorithm attempts to find the best match between workers and tasks, the performance degrades much more slowly when k grows, even with the extra exploration costs.

7. Conclusion

We conclude by mentioning several extensions of our model. We have assumed that the requester pays the same price for any label. Our results can be extended to handle the case in which different workers charge different prices. Let $c_{i,j}$ denote the cost of obtaining a label for task i from worker j. The objective in the integer program would become $\Sigma_{i=1}^{n}\Sigma_{j=1}^{m} c_{i,j} y_{i,j}$. This is linear and the same techniques would apply.

The framework can also be extended to handle more intricate assumptions about the structure of tasks. We have assumed that there are T task types, with $p_{i,j} = p_{i',j}$ whenever i and i' are of the same type. However, this assumption is used only in the exploration phase in which workers' skills are estimated. While the amount of exploration required by the algorithm depends on the particular task structure assumed, the derivation of our algorithm and the general analysis are independent of the task structure.

Acknowledgements

This research was partially supported by the National Science Foundation under grant IIS-1054911.

References

Alon, Noga, Awerbuch, Baruch, Azar, Yossi, Buchbinder, Niv, and Naor, Joseph. A general approach to online network optimization problems. In *SODA*, 2004.

Bansal, Nikhil, Buchbinder, Niv, and Naor, Joseph. A primal-dual randomized algorithm for weighted paging. In *FOCS*, 2007.

Buchbinder, Niv and Naor, Joseph. Online primal-dual algorithms for covering and packing problems. In *ESA*, 2005.

Buchbinder, Niv, Jain, Kamal, and Naor, Joseph. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, 2007.

Crammer, Koby, Kearns, Michael, and Wortman, Jennifer. Learning from data of variable quality. In *NIPS*, 2005.

Crammer, Koby, Kearns, Michael, and Wortman, Jennifer. Learning from multiple sources. *Journal of Machine Learning Research*, 9:1757–1774, 2008.

Dawid, Philip and Skene, Allan. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 28(1):20–28, 1979.

Dekel, Ofer and Shamir, Ohad. Vox populi: Collecting high-quality labels from a crowd. In *COLT*, 2009.

Devanur, Nikhil and Hayes, Thomas. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM EC*, 2009.

Devanur, Nikhil, Jain, Kamal, Sivan, Balasubramanian, and Wilkens, Christopher. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *ACM EC*, 2011.

Ghosh, Arpita, Kale, Satyen, and McAfee, Preston. Who moderates the moderators? Crowdsourcing abuse detection in user-generated content. In *ACM EC*, 2011.

Ho, Chien-Ju and Vaughan, Jennifer Wortman. Online task assignment in crowdsourcing markets. In *AAAI*, 2012.

Ipeirotis, Panagiotis. Analyzing the Amazon Mechanical Turk marketplace. *ACM XRDS*, 17(2):16–21, 2010.

Ipeirotis, Panagiotis G., Provost, Foster, and Wang, Jing. Quality management on Amazon Mechanical Turk. In *HCOMP*, 2010.

Karger, David, Oh, Sewoong, and Shah, Devavrat. Iterative learning for reliable crowdsourcing systems. In *NIPS*, 2011a.

Karger, David, Oh, Sewoong, and Shah, Devavrat. Budget-optimal task allocation for reliable crowdsourcing systems. CoRR, abs/1110.3564, 2011b.

Kittur, Aniket, Chi, Ed, and Suh, Bongwon. Crowdsourcing user studies with Mechanical Turk. In *CHI*, 2008.

Liu, Qiang, Peng, Jian, and Ihler, Alexander. Variational inference for crowdsourcing. In *NIPS*, 2012.

Oleson, Dave, Hester, Vaughn, Sorokin, Alex, Laughlin, Greg, Le, John, and Biewald, Lukas. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *HCOMP*, 2011.

Sheng, Victor, Provost, Foster, and Ipeirotis, Panagiotis. Get another label? Improving data quality using multiple, noisy labelers. In *KDD*, 2008.

Tran-Thanh, Long, Stein, Sebastian, Rogers, Alex, and Jennings, Nicholas. Efficient crowdsourcing of unknown experts using multi-armed bandits. In *ICAI*, 2012.

Wah, Catherine, Branson, Steve, Welinder, Peter, Perona, Pietro, and Belongie, Serge. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Wais, Paul, Lingamneni, Shivaram, Cook, Duncan, Fennell, Jason, Goldenberg, Benjamin, Lubarov, Daniel, and Martin, David. Towards building a high-quality workforce with mechanical turk. Presented at the *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, 2010.

Welinder, Peter, Branson, Steve, Belongie, Serge, and Pietro, Perona. The Multidimensional Wisdom of Crowds. In *NIPS*, 2010.

Zhou, Dengyong, Basu, Sumit, Mao, Yi, and Platt, John. Learning from the wisdom of crowds by minimax entropy. In *NIPS*, 2012.