Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Online Task Assignment in Crowdsourcing Markets

Chien-Ju Ho and **Jennifer Wortman Vaughan**
Department of Computer Science
University of California, Los Angeles
{cjho, jenn}@cs.ucla.edu

Abstract

We explore the problem of assigning heterogeneous tasks to workers with different, unknown skill sets in crowdsourcing markets such as Amazon Mechanical Turk. We first formalize the *online task assignment problem*, in which a requester has a fixed set of tasks and a budget that specifies how many times he would like each task completed. Workers arrive one at a time (with the same worker potentially arriving multiple times), and must be assigned to a task upon arrival. The goal is to allocate workers to tasks in a way that maximizes the total benefit that the requester obtains from the completed work. Inspired by recent research on the online adwords problem, we present a two-phase exploration-exploitation assignment algorithm and prove that it is competitive with respect to the optimal offline algorithm which has access to the unknown skill levels of each worker. We empirically evaluate this algorithm using data collected on Mechanical Turk and show that it performs better than random assignment or greedy algorithms. To our knowledge, this is the first work to extend the online primal-dual technique used in the online adwords problem to a scenario with unknown parameters, and the first to offer an empirical validation of an online primal-dual algorithm.

Introduction

Crowdsourcing markets, such as Amazon Mechanical Turk, are online labor markets in which individuals post short "microtasks" that workers can complete in exchange for a small payment. A typical Turk task might involve translating a paragraph from German into English, verifying the address of a company, or labeling the content of an image, and payments are typically on the order of ten cents. Crowdsourcing markets provide a mechanism for task requesters to inexpensively obtain distributed labor and information, and have recently become popular among researchers who use sites like Mechanical Turk to conduct user studies (Kittur, Chi, and Suh 2008), run behavioral experiments (Mason and Suri 2012), and collect data (Horton and Chilton 2010).

Despite the relatively small size of payments, crowdsourcing markets attract a diverse pool of workers from around the world, who participate both to pick up extra cash and to kill time in a fun way (Ipeirotis 2010). In principle, requesters should be able to take advantage of this diversity by assigning each task to a worker with the proper skills necessary to complete it. However, requesters typically assign tasks randomly to arriving workers, and often rely on duplicated, redundant assignments to boost the quality of the information they collect (Ipeirotis, Provost, and Wang 2010; Karger, Oh, and Shah 2011). Random assignment can be fine when tasks do not require specific skills, but can result in severely suboptimal performance for human computation tasks that require specialized training or knowledge, such as analyzing the nutritional value of food, as proposed by Noronha et al. (2011), or helping people plan their travel, as proposed by Law and Zhang (2011).

In this work, we address the challenge that a single requester faces when assigning heterogeneous tasks to workers with unknown, heterogeneous skills. We begin by formalizing this challenge, which we call the *online task assignment problem*. In our formalization, a requester has a fixed set of tasks and a budget for each task which specifies how many times he would like the task completed. Workers arrive online, and must be assigned to a task upon arrival. Each worker is assumed to have an underlying skill level for each task; on average, workers with higher skill levels produce work that is more valuable to the requester. Skill levels are initially unknown, but can be learned through observations as workers complete tasks. This leads to a natural exploration-exploitation trade-off; the requester must sample the performance of each new worker on different tasks to estimate her skills, but would prefer to assign a worker to the tasks at which she excels. The problem is further complicated by the task budgets, which make greedy assignments based on estimated skill levels suboptimal.

The online task assignment problem is closely related to several problems that have been studied in the literature. If we make the simplifying assumption that workers' skill levels are known, the offline version of our problem becomes the well-studied *assignment problem*, which can be optimally solved using the Hungarian algorithm (Kuhn 1955) or linear programming. Moving to the online setting but still assuming known skill levels, our problem can be formulated in a similar way to the *online adwords problem* and the related *display ad allocation problem* (Feldman et al. 2010; Devanur et al. 2011). For these problems, it is possible to achieve a competitive ratio of $(1 - 1/e)$ in a setting with adversarial arrivals (Buchbinder, Jain, and Naor 2007) or

$(1-\varepsilon)$ for stochastic arrivals, with ε very small when the total budget is large (Devanur and Hayes 2009).

Our proposed algorithm, the Dual Task Assigner (DTA), builds on a central idea from this literature, the *online primal-dual formulation*, but handles the case in which worker skills are unknown and must be learned through exploration. We prove that under the assumption of a stochastic worker arrival order, as long as the number of worker arrivals is sufficiently high, DTA is competitive with respect to the offline optimal algorithm that has access to the unknown skill levels of each worker.

We empirically evaluate the performance of DTA using data collected on Amazon Mechanical Turk. We examine two scenarios. In the "expertise level" setting, tasks can be classified as easy, medium, and hard. A successful assignment algorithm must learn to assign easy tasks to workers with relatively low skill levels and save the experts for the hard tasks. In the "different skills" setting, there are three types of tasks and each worker is trained to perform only one of the three. A successful assignment algorithm must simply learn which of the three tasks each worker is able to perform. The experiments show that DTA outperforms greedy algorithms and random assignment on both tasks, but that the improvement over greedy algorithms is more significant in the expertise level setting. To our knowledge, this provides the first published empirical validation of any algorithm based on the online primal-dual formulation.

Related Work

Our algorithm and analysis build on the online primal-dual framework, which has been used to analyze a variety of online optimization problems, including online network optimization (Alon et al. 2004), online paging (Bansal, Buchbinder, and Naor 2007), and the online adwords problem (Buchbinder, Jain, and Naor 2007). Many of these analyses assume a worst-case, adversarial selection of input, which often leads to strong impossibility results. More recently, researchers have considered settings in which the input satisfies some stochastic properties.

The problem closest to ours is the stochastic online adwords problem, in which there is a set of advertisers bidding on keywords. Each advertiser has a budget. Keywords arrive online, and advertisers must be assigned to keywords as they arrive. The goal is to maximize revenue (i.e., the sum of the bids of the advertisers for the keywords on which their ads are displayed) without exceeding any advertiser's budget. The online primal-dual framework can be used to achieve near-optimal performance on this problem when the total budget is sufficiently large (Agrawal, Wang, and Ye 2009; Feldman et al. 2010; Devanur et al. 2011).

Our work extends these results by considering the situation in which some parameters (in our case, worker skill sets, which are analogous to advertiser bids) are unknown but can be learned from observations. This results in an exploration-exploitation trade-off, as arises in the well-studied multi-armed bandit problem (Auer, Cesa-Bianchi, and Fischer 2002; Auer et al. 2003). Unlike the traditional bandit problem, we must deal with the budget constraint. As our experiments show, this constraint makes it suboptimal to use greedy selection for exploitation, as most bandit algorithms do. There has been work on budgeted variants of the bandit problem (Guha and Munagala 2007; Goel, Khanna, and Null 2009; Tran-Thanh et al. 2010), but the budget is typically interpreted as a limit on the number of rounds that can be used for exploration, whereas our budgets are on the number of times each arm can be pulled.

The Online Task Assignment Problem

We now describe our problem formulation, which we refer to as the *online task assignment problem*. The problem is stated from the point of view of a single task requester who has a fixed set of n tasks he would like completed, and a budget b_i for each task i that specifies the maximum number of times he would like the task to be performed. For example, the requester might have $b_1 = 100$ articles that he would like translated from German to English, and $b_2 = 200$ articles that he would like translated from French to English.

Workers from a pool of size k arrive one at a time. Every time a worker arrives, the requester must assign her a task. Each worker j has an unknown underlying skill level $u_{i,j} \in [0,1]$ for each task i, which represents the utility that the requester obtains on expectation each time the worker performs that task — after worker j performs the task i that is assigned to her, the requester receives a benefit in $[0,1]$ drawn from a distribution with expectation $u_{i,j}$. We assume that the requester utility is additive. That is, if a requester receives u_t units of utility at each time t, his total utility at time T is $\Sigma_{t=1}^{T} u_t$. Once the requester has assigned workers to any task i at least b_i times, he no longer receives a benefit from assigning additional workers to that task.

We use m to denote the total number of worker arrivals, which is assumed to be known by the requester. Typically, we set $m = \Sigma_{i=1}^{n} b_i$, i.e., m is determined by the requester budget. The goal of the requester is to maximize his cumulative expected utility after time m. We evaluate algorithms using the notion of *competitive ratio*, which is a lower bound on the ratio between the cumulative utility of the algorithm and the cumulative utility of the optimal offline assignment (which can be computed using the unknown worker skill levels). For example, a competitive ratio of $1/2$ would imply that an algorithm always achieves a utility that is at least half as good as optimal. We say an algorithm is α-*competitive* if its competitive ratio is at least α.

We analyze our algorithm in a random permutation model (Devanur and Hayes 2009). In this model, the number of workers k, skill levels $u_{i,j}$ of each worker, and number of times that each worker arrives are all assumed to be chosen adversarially, and are not known by the requester. However, the order in which workers arrive is randomly permuted. Since the adversary chooses the precise number of times that each worker arrives (as opposed to simply choosing the probability that a particular worker arrives on any given time step), the offline optimal allocation is well-defined.

We have implicitly made the simplifying assumption that the requester is able to quickly and accurately evaluate the quality of the work that he receives. This assumption is realistic, for example, for retrieval tasks, such as finding the URL of local businesses in various cities or finding images

of people engaged in various activities. We leave the problem of task assignment in other settings for future work.

The Primal-Dual Formulation and Algorithm

We formalize our problem in the online primal-dual framework. For intuition, we first discuss the offline problem. We then discuss the online problem under the assumption that worker skills are known, and finally move on to the setting of interest, in which worker skills are unknown.

Primal-Dual Formulation of the Offline Problem

Consider first the *offline* version of the task assignment problem, in which the sequence of arriving workers and their skill levels are known in advance. We can formulate this problem as a linear program. To do this, we define an indicator variable $y_{i,t}$ which is 1 if task i is assigned to the worker who arrives at time t, and 0 otherwise. We use $j(t)$ to denote the identity of the worker who arrives at time t. (Recall that the same worker may arrive many times.) Using this notation, we can write the offline version of the problem as an integer program. Since the budgets b_i are integers for all i and the constraint coefficients are all integers, the offline problem is equivalent to the following linear program:

Primal		Dual	
max	$\Sigma_{t=1}^{m}\Sigma_{i=1}^{n} u_{i,j(t)} y_{i,t}$	min	$\Sigma_{i=1}^{n} b_i x_i + \Sigma_{t=1}^{m} z_t$
s.t.	$\Sigma_{i=1}^{n} y_{i,t} \leq 1, \forall t$	s.t.	$x_i + z_t \geq u_{i,j(t)}, \forall (i,t)$
	$\Sigma_{t=1}^{m} y_{i,t} \leq b_i, \forall i$		$x_i, z_t \geq 0, \forall (i,t)$
	$y_{i,t} \geq 0, \forall (i,t)$		

The Online Problem with Known Skill Levels

Consider a simplified version of our problem in which workers arrive online, but the skill levels $u_{i,j(t)}$ for all tasks i are revealed when the tth worker arrives. This can be mapped to the online adwords problem; tasks are mapped to advertisers, workers to keywords, and skills to bids. This allows us to apply the Learn-Weights (LW) algorithm and analysis of Devanur and Hayes (2009) with minor modifications.[1]

LW makes use of the dual form of the linear program above. By strong duality, if we minimize the dual objective, we maximize the primal objective. We know that in the optimal dual solution, $z_t = \max_i\{u_{i,j(t)} - x_i\}$ for all t; if this were not the case for some t then z_t could be decreased without violating any constraint, improving the objective. Furthermore, by complementary slackness, in the optimal assignment, $y_{i,t}(u_{i,j(t)} - x_t - z_t) = 0$ for all i and t. This implies that in the optimal solution, $y_{i,t} = 1$ only if $z_t = u_{i,j(t)} - x_i$, or $i = \arg\max_i\{u_{i,j(t)} - x_i\}$.[2]

The goal then becomes finding the optimal values of the variables x_i. LW has access to the values b_i. While it does not have access to the $u_{i,j(t)}$ for all t a priori, these values are revealed over time. LW begins by sampling values $u_{i,j(t)}$ while assigning workers to tasks at random for the first γm rounds, for some $\gamma \in (0,1)$. Because we are in the random permutation model, these samples can be used to get an accurate estimate of the distribution of worker skills. LW then uses these samples to estimate the optimal x_i values, i.e., find the values $\hat{x}_i$ that minimize $\Sigma_{i=1}^{n}\gamma b_i \hat{x}_i + \Sigma_{j=1}^{\gamma m}\max_i\{u_{i,j(t)} - \hat{x}_i\}$. For the remaining $(1-\gamma)m$ rounds, workers are assigned to the task that maximizes $u_{i,j(t)} - \hat{x}_i$.

To summarize, in the exploration phase, worker skills are observed and the optimal task baseline weight $\hat{x}_i$ for each task i is calculated. In the exploitation phase, each arriving worker j is assigned the task i that maximizes the marginal utility $u_{i,j} - \hat{x}_i$. Using a PAC-style analysis, one can derive the conditions on γ that imply that the assignments in these rounds will be near-optimal.

Theorem 1. *(Devanur and Hayes 2009) Consider the simplified online task assignment problem with revealed skill levels. Let V_{opt} be the optimal objective value, $u_{max} = \max_{i,j}\{u_{i,j}\}$, and $\lambda = \max_{i,i',j}\{u_{i'j}/u_{i,j} : u_{i,j} \neq 0\}$. LW($\gamma$) is $(1-\gamma)$-competitive for any γ satisfying*

$$\frac{V_{opt}}{u_{max}} \geq \Omega\left(\frac{n^2 \log(\lambda/\gamma)}{\gamma^3}\right). \quad (1)$$

Unknown Skills and the Dual Task Assigner

We now turn our attention back to the online task assignment problem and show how to modify LW to handle unknown skill levels. We introduce the Dual Task Assigner (DTA), which estimates unknown worker skill levels and assigns tasks to online arriving workers based on the estimation. Note that LW is already split into two phases: an exploration phase which is used to select the values of the variables $\hat{x}_i$ based on the observed distribution over worker skills, and an exploitation phase that aims to optimize the dual objective. DTA simply uses observations from the exploration phase to estimate the values $u_{i,j}$ (in addition to learning the $\hat{x}_i$), and runs the exploitation phase using the estimated values. As we will show in Theorem 2, DTA is guaranteed to have near-optimal performance when m is large compared with nk, the number of parameters that must be estimated. This implies that we do not lose much by not knowing the skill levels of workers a priori if workers return sufficiently often. The algorithm is formally stated in Algorithm 1; r_i keeps track of the remaining budget for task i.

We remark that although this algorithm was designed for the online task assignment problem, it can also be applied in the online adwords setting if the goal is to maximize expected clicks (determined by an unknown click-through rate for each ad-keyword pair) instead of maximizing payments.

Competitive Analysis

In this section, we provide an overview of the derivation of our main theoretical result, the competitive ratio of DTA, which is stated in Theorem 2. We first examine how the performance of an online assignment algorithm that has access

[1] They require $\Sigma_{t=1}^{m} u_{i,j(t)} y_{i,t} \leq b_i$ instead of $\Sigma_{t=1}^{m} y_{i,t} \leq b_i$.

[2] This argument relies on an assumption that there is a unique task i maximizing $u_{i,j(t)} - x_i$. To break ties and ensure a unique maximizing task, we can add small amounts of random noise as in Devanur and Hayes (2009).

Algorithm 1: Dual Task Assigner (γ)

Set $s_{i,j} \leftarrow 0 \; \forall (i,j)$
Set $r_i \leftarrow b_i \; \forall i$
for $t \leftarrow 1$ *to* γm **do**
  Set $i \leftarrow \arg\min_{i:r(i)>0} s_{i,j(t)}$
  Assign worker $j(t)$ to task i
  Set $s_{i,j(t)} \leftarrow s_{i,j(t)} + 1$
  Set $r_i \leftarrow r_i - 1$
Set $\hat{u}_{i,j} \leftarrow$ average observed benefit of j in task $i \; \forall (i,j)$
Set $\{\hat{x}_i\} \leftarrow \arg\min_{\{x_i\}} \Sigma_{i=1}^{n} \gamma x_i b_i + \Sigma_{j=1}^{\gamma m} \max_i \{\hat{u}_{i,j} - x_i\}$
for $t \leftarrow \gamma m + 1$ *to* m **do**
  Set $i \leftarrow \arg\max_{i:r(i)>0} \hat{u}_{i,j(t)} - \hat{x}_i$
  Assign worker $j(t)$ to task i
  Set $r_i \leftarrow r_i - 1$

to the worker skill levels degrades when estimates of the skill levels are used in place of the true values. We then derive a bound on the error of the skill level estimates obtained using random sampling as in the exploration phase of DTA. These results are combined to derive the competitive ratio.

We first define the error measure. Given a value u and an estimate $\hat{u}$ of u, we say that the *multiplicative error* of $\hat{u}$ is bounded by ε if $1-\varepsilon \leq \hat{u}/u \leq 1/(1-\varepsilon)$. Similarly, given estimates $\hat{u}_{i,j}$ of worker skill levels $u_{i,j}$ and an assignment of workers to tasks in which each worker j completes task i $n_{i,j}$ times, we say that the *average multiplicative error* for the assignment is bounded by ε if

$$1-\varepsilon \leq \frac{\Sigma_{i,j} n_{i,j} \hat{u}_{i,j}}{\Sigma_{i,j} n_{i,j} u_{i,j}} \leq \frac{1}{(1-\varepsilon)}.$$

Lemma 1 shows how the performance of any competitive online assignment algorithm that has access to worker skill levels degrades when approximations are used.[3]

Lemma 1. *Let A be an α-competitive algorithm for the online task assignment problem with known worker skills. If A is run using approximated worker skill levels such that the average multiplicative error for any possible assignment is bounded by ε, then the ratio of the performance of the algorithm to the offline optimal is bounded by $\alpha(1-2\varepsilon)$.*

Note that the lemma is true only when the average multiplicative error is bounded by ε for *any possible assignment*. If we do not have any restrictions about the assignment, ε would be bounded by the worst estimation of $u_{i,j}$ for all (i,j). However, since we are considering the random permutation model, we can get a more reasonable bound for our algorithm. We next derive a bound on the average multiplicative error with estimates obtained during the first phase of DTA. As a first step, we state a bound on the multiplicative error for each worker quality $u_{i,j}$ which follows from a simple application of the multiplicative Chernoff bounds.

Lemma 2. *Suppose we observe worker j completing task i $s_{i,j}$ times and set $\hat{u}_{i,j}$ to the average requester utility. For any $\delta > 0$, with probability $1-\delta$, Let $\varepsilon_{i,j}$ be the multiplicative error of $\hat{u}_{i,j}$ with respect to $u_{i,j}$. We have*

$$\varepsilon_{i,j} \leq \sqrt{\frac{3\ln(2/\delta)}{s_{i,j} u_{i,j}}}.$$

This bound is small for workers who appear frequently during the sampling phase, and large for workers who do not. However, in the random permutation model, we would expect that any worker who rarely arrives in the sampling phase should also rarely arrive in the exploitation phase, so the average multiplicative error in the exploitation phase should be small as long as m is large compared with k.

Let s_j be the number of times worker j arrives in the sampling phase. Define $u_{avg} = \sum_j s_j \min_i u_{i,j} / \sum_j s_j$, the average skill level of the workers on their worst tasks, weighted by how often each appears in the sampling phase (which is roughly proportional to how often he appears overall).

Lemma 3. *Under the random permutation model, given the first γm rounds are used for uniform sampling, for any $\delta > 0$, with probability $1-\delta$, if $d := \min_j s_j / \sqrt{\gamma m \ln(k/\delta)} > 1$, then for any possible assignment in the exploitation phase, the average multiplicative error ε in the exploitation phase is bounded as*

$$\varepsilon \leq \frac{d+1}{d-1} \sqrt{3\ln(4nk/\delta)} \sqrt{\frac{nk}{\gamma m}} \frac{1}{u_{avg}}. \qquad (2)$$

Let's take a moment to examine this bound. The first term goes to one as m gets reasonably large. The second term is a standard logarithmic penalty that we must pay in order to obtain that the result holds with high probability. The third term has a familiar form common in many PAC-style bounds (see, for example, Kearns and Vazirani (1994)). The quantity nk can be viewed as the complexity of the model that we must learn, captured by the number of unknown parameters (i.e., $u_{i,j}$ values), while the quantity γm is the number of examples observed in the first phase of the algorithm, giving us the familiar square root of complexity divided by sample size. The final term appears because competitive ratio is defined in terms of a ratio, requiring the use of multiplicative error in the analysis; by Lemma 2, more examples are required to obtain a low error for small values of $u_{i,j}$.

Combining these, we derive the competitive ratio of DTA.

Theorem 2. *Under the random permutation model, for any sampling ratio $\gamma \leq 1/2$ that satisfies the condition in Equation 1, for any $\delta > 0$, with probability $1-\delta$, if $d := \min_j s_j / \sqrt{\gamma m \ln(k/\delta)} > 1$, then the ratio between the utility obtained by DTA and the offline optimal V_{opt} is at least $(1-\gamma)(1-2\varepsilon)$, for some ε satisfying Equation 2.*

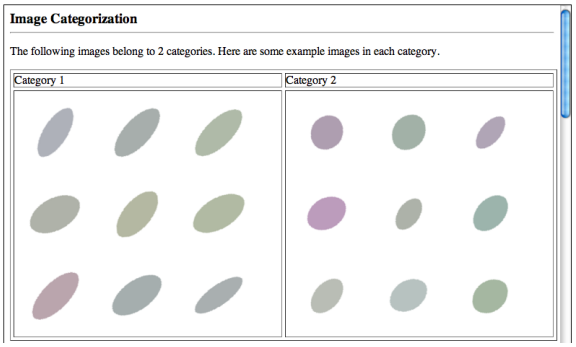
Strictly speaking, the result in Theorem 2 is not a competitive ratio since it only holds with high probability. However, one can easily transform the result to a competitive ratio by considering the performance of the algorithm on expectation. This adds a multiplicative factor of $(1-\delta)$ to the bound. From Equation 2, we can show that δ can be chosen to be $o(\varepsilon)$ if m is sufficiently large. We can then get a competitive ratio of $(1-\gamma)(1-O(\varepsilon))$ (using a similar technique

[3] All proofs appear in an appendix which can be found in the version of this paper posted on the authors' websites.

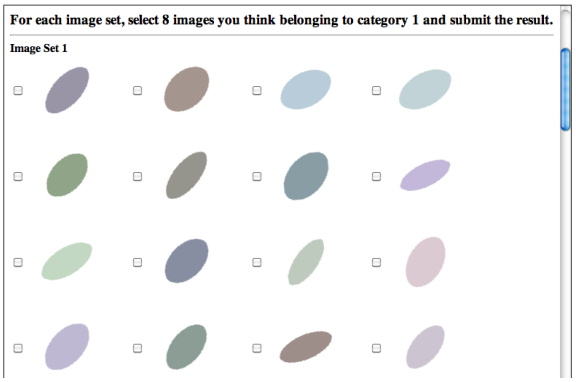
to Devanur and Hayes (2009)). However, since this result would sacrifice some preciseness, we keep our theorem in the current form.

We would also like to remark that the value of ε is $O(1/\sqrt{\gamma})$. Therefore, increasing the sample ratio γ would decrease the error ε. This results in an exploration-exploitation trade-off in choosing γ.

This result shows that as the number of arrivals m grows large compared with nk, the performance of DTA approaches the performance of LW in the simplified but unrealistic setting in which worker skills are known. One might ask if it is possible to tighten this bound, or to relax the condition that no s_j can be small. We offer the following intuitive argument that one should not expect to do much better. Suppose that m is small compared with nk. In the extreme, if a new worker appeared on every time step ($m = k$), then any assignment algorithm would effectively be random since the algorithm would have no information about each worker. If workers appeared repeatedly but only a small number of times, then there would not be sufficient data to learn about the workers' skills, and assignments would still be essentially random. Even if *most* workers appeared frequently, we could still be in trouble if the few who appeared rarely had much higher skill levels than the rest, since assigning these workers to tasks suboptimally could have a large (unbounded) impact on competitive ratio. Therefore, in order to achieve a meaningful competitive ratio style bound, it is necessary that either all workers appear relatively often, or most workers appear often, and the ones who appear rarely don't have unusually high skill levels.



1(a) Instructions given to workers in the first group. The sample images appear to be classified by length.



1(b) A task. This example is easy to classify using color, but hard using length and the rotation angle. The ellipses in one of the categories are darker while the others are lighter.

Empirical Evaluation

Theorem 2 shows that DTA is competitive with the offline optimal when the worker arrival sequence is random and the number of arrivals is large. These assumptions might not always hold in the real world. In this section, we empirically evaluate the performance of DTA on data collected on Amazon Mechanical Turk.

Tasks and Skill Sets

We create a set of ellipse classification tasks. When a new worker arrives for the first time, she is randomly assigned to one of three groups, which determines which set of instructions she will receive. The text of the instructions for all three groups is identical, telling them that they will classify images of ellipses into two groups. However, the sample images that they see are different. The first group sees sample images that appear to be classified by the length of their major axis, as in Figure 1(a). The second group sees images that appear to be classified by color. The third sees images that appear to be classified by angle of rotation. These sample images prime the workers to look for different characteristics in the ellipses they will later classify, effectively creating sets of workers with different "skills."

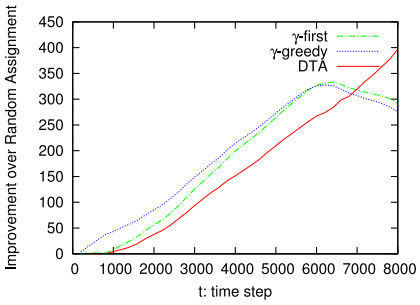
There are eight different ellipse classification tasks. In each, the worker is presented with sixteen images of ellipses and asked to classify them into two categories. The difference between the tasks is the way in which the images are generated. In particular, the generation process has three parameters: 1) whether or not the two underlying groups are easy to be classified using the length of the major axis, 2) whether or not the two groups are easy to be classified using color, and 3) whether or not the two groups are easy to be classified using rotation angles. Each of the three parameters has two settings, leading to eight different parameter values for the eight tasks. Figure 1(b) shows an example of a task.

The utility of the requester is taken to be the fraction of images that a worker correctly classifies.

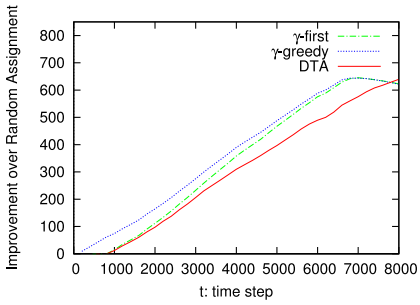
Data Collection for Offline Evaluation

It is notoriously challenging to design empirical evaluation techniques to compare algorithms with exploration-exploitation components (Langford, Strehl, and Wortman 2008; Li et al. 2011); if we collect data by running a single task assignment algorithm, we cannot know what would have happened if we had instead used a different assignment, since we only observe workers completing the tasks they are assigned. We could attempt to compare algorithms (and parameter settings for each algorithm) by publishing separate sets of tasks for each, but this would be both time consuming and costly. Furthermore, because different workers would complete the tasks, the comparison may be unfair.

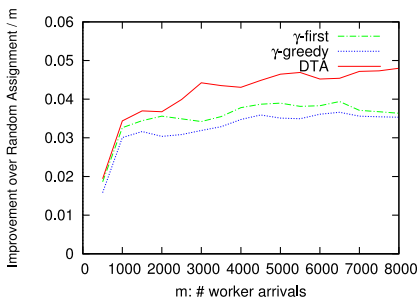
We instead use an offline evaluation methodology. During



2(a) Hard, medium, and easy tasks.



2(b) Tasks requiring different type of skills.



2(c) Dependence on the number of arrivals.

our data collection phase, rather than running a task assignment algorithm to assign tasks to workers online, we have each worker complete *all eight tasks* every time she accepts a job. This allows us to collect data on the performance of the worker on all of the tasks simultaneously. We can then use this data to evaluate task assignment algorithms offline. When a task assignment algorithm assigns a worker to a particular task, we allow the algorithm to observe the performance of the worker on that task only, and disregard the information collected about the other seven tasks. In this way, we can use a single data set to evaluate and compare several algorithms with a variety of parameter settings without needing to collect new data for each algorithm.

In total, we published 8,000 task sets, each of which consists of an instance of all 8 tasks. We offered $.05 for each task set. To encourage workers to complete as many sets as possible, we provided $.40 and $1 bonuses to workers completing at least 25 and 50 task sets respectively. As a result, 612 workers participated in the experiments. Among these workers, 268 of them only completed one task set. However, many workers completed more, with 53 workers completing more than 50 task sets each. Over half (4,122) of the 8,000 published task sets were completed by these 53 workers. [4]

The Algorithms

In addition to DTA, we implemented a random assignment algorithm, which assigns workers to tasks uniformly at random, and two greedy algorithms, γ-first and γ-greedy.[5] The γ-first algorithm makes random assignments for the first γm rounds, and uses a greedy assignment based on estimates of worker skill levels on the remaining rounds, no longer assigning a particular task once the budget has been used. The γ-greedy algorithm randomly explores on each round with probability γ, and otherwise assigns a task greedily.

Both of the greedy algorithms and DTA use estimates of worker skill levels in order to make decisions. In practice, worker arrivals are not randomly permuted and some workers may arrive late. To handle this, in our implementation of these algorithms, a worker is assigned to each task once the first n times she arrives, even if the sampling phase is over, and the observations are used to estimate her skill levels.

Experiments and Results

To simulate real-world scenarios using the collected data, we modified the budgets of the 8 tasks in two ways, setting some of the task budgets to zero to effectively produce a smaller n. In these experiments, the default sampling ratio γ is 0.1 if not mentioned. To reduce noise, the plots are generated by averaging results from 10 runs of each algorithm.

Expertise level. The first experiment was designed to simulate a scenario in which the requester possesses tasks that require different expertise levels, i.e., hard tasks, medium tasks, and easy tasks. Hard tasks can only be solved by a portion of workers, while easy tasks can be solved by anyone. To simulate this, we set the budget to be non-zero for three of the tasks: 1) classifying ellipses that are easy to classify using any of the three criteria, 2) classifying ellipses that are easy to classify by color or angle, and 3) classifying ellipses that are easy to classify using only angle. These three tasks were each given a budget of $m/3$.

Figure 2(a) shows the results of this experiment. The x-axis is time and the y-axis is the difference between the cumulative utility of each algorithm and the cumulative utility of the random assignment algorithm. Although the greedy algorithms perform better in the beginning, their performance degrades quickly once budgets are used up; both run out of budget for easy tasks, and are forced to assign hard tasks to workers who are not capable of solving them. Because DTA is explicitly designed to take budget into consideration, it outperforms both greedy algorithms in the end.

Different skills. The second experiment was designed to simulate a scenario in which the requester possesses tasks that require different types of skills, such as translating paragraphs from French to English versus translating paragraphs from Mandarin to English. To simulate this scenario, we set the budget to be non-zero for a different set of three tasks: classifying ellipses that are easy only using length, classifying ellipses that are easy only using color, and classifying ellipses that are easy only using angle of rotation. Each of these three tasks was given a budget of $m/3$.

Because the budgets of the three tasks are equal, one would expect that a greedy algorithm should be near-optimal in this setting; to perform well, the assignment algorithm

[4] Our success obtaining a high level of worker return suggests that others might benefit from the use of bonuses to encourage workers to return many times.

[5] See, e.g., Sutton and Barto (1998). We use γ instead of the more traditional ε to avoid confusion with the use of ε for error.

must simply determine what the specialty of each worker is and assign each worker accordingly. As we see in Figure 2(b), the greedy algorithms do perform well in this setting. However, DTA still achieves a small advantage by explicitly considering budgets.

Choice of sampling ratio. One might ask how sensitive these results are to the choice of the sampling ratio γ used in DTA, γ-first, and γ-greedy. To test this, we ran each of the algorithms in the expertise level setting with a wide range of sampling ratios to see how their performance would change. Encouragingly, we found that none of the algorithms were especially sensitive to the choice of the sampling ratio γ. DTA consistently performed better than the greedy algorithms except when DTA was given a very small (less than .005) sampling ratio.

Dependence on the number of arrivals. Given our theoretical results, one might expect DTA to perform poorly when the number of arrivals m is small. To test this, we modified the size of data by using only the first m arrivals for different values of m in the expertise level setting. (Note that this requires running DTA from scratch for each value of m since m impacts the budgets.) The results appear in Figure 2(c). The x-axis shows the value m, and the y-axis shows the difference between the utility of each algorithm and the utility of the random assignment normalized by m, i.e., the average improvement per time step of each algorithm over random assignment. The performance of DTA starts to converge after about 2,000 to 3,000 points. This experiment shows that DTA can be applied successfully even if m is not very large compared with kn.

Conclusion

We introduced the *online task assignment problem* in which heterogeneous tasks must be assigned to workers with different, unknown skill sets. We showed that a simplified version of this problem in which skill sets are known is mathematically similar to the well-studied online adwords problem. Exploiting this similarity, we designed the Dual Task Assigner algorithm by modifying the algorithm of Devanur and Hayes (2009) to handle and learn unknown parameters, and showed that DTA is competitive with respect to the offline optimal when the number of worker arrivals is large. We compared the performance of DTA to random assignment and greedy approaches using data collected on Amazon Mechanical Turk, and showed that DTA outperforms other algorithms empirically, especially in the scenario in which "expert" workers must be saved for the most difficult tasks.

Acknowledgements

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

References

Agrawal, S.; Wang, Z.; and Ye, Y. 2009. A dynamic near-optimal algorithm for online linear programming. Working Paper.

Alon, N.; Awerbuch, B.; Azar, Y.; Buchbinder, N.; and Naor, J. S. 2004. A general approach to online network optimization problems. In *SODA*.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 2003. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing* 32:48–77.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47:235–256.

Bansal, N.; Buchbinder, N.; and Naor, J. S. 2007. A primal-dual randomized algorithm for weighted paging. In *FOCS*.

Buchbinder, N.; Jain, K.; and Naor, J. S. 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*.

Devanur, N. R., and Hayes, T. P. 2009. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM EC*.

Devanur, N. R.; Jain, K.; Sivan, B.; and Wilkens, C. A. 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *ACM EC*.

Feldman, J.; Henzinger, M.; Korula, N.; Mirrokni, V. S.; and Stein, C. 2010. Online stochastic packing applied to display ad allocation. In *ESA*.

Goel, A.; Khanna, S.; and Null, B. 2009. The ratio index for budgeted learning, with applications. In *SODA*.

Guha, S., and Munagala, K. 2007. Approximation algorithms for budgeted learning problems. In *STOC*.

Horton, J. J., and Chilton, L. B. 2010. The labor economics of paid crowdsourcing. In *ACM EC*.

Ipeirotis, P. G.; Provost, F.; and Wang, J. 2010. Quality management on Amazon Mechanical Turk. In *HCOMP*.

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

Karger, D. R.; Oh, S.; and Shah, D. 2011. Iterative learning for reliable crowdsourcing systems. In *NIPS*.

Kearns, M., and Vazirani, U. V. 1994. *An Introduction to Computational Learning Theory*. Cambridge, MA, USA: MIT Press.

Kittur, A.; Chi, E. H.; and Suh, B. 2008. Crowdsourcing user studies with Mechanical Turk. In *CHI*.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2(1-2):83–97.

Langford, J.; Strehl, A.; and Wortman, J. 2008. Exploration scavenging. In *ICML*.

Law, E., and Zhang, H. 2011. Towards large-scale collaborative planning: Answering high-level search queries using human computation. In *AAAI*.

Li, L.; Chu, W.; Langford, J.; and Huang, X. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*.

Mason, W., and Suri, S. 2012. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior Research Methods*. To appear.

Noronha, J.; Hysen, E.; Zhang, H.; and Gajos, K. Z. 2011. Platemate: Crowdsourcing nutrition analysis from food photographs. In *UIST*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press.

Tran-Thanh, L.; Chapman, A.; Munoz De Cote Flores Luna, J. E.; Rogers, A.; and Jennings, N. R. 2010. Epsilon-first policies for budget-limited multi-armed bandits. In *AAAI*.