Sequential crowdsourced labeling as an epsilon-greedy exploration in a Markov Decision Process

Vikas C. Raykar
IBM Research, Bangalore, India

Priyanka Agrawal
IBM Research, Bangalore, India

Abstract

Crowdsourcing marketplaces are widely used for curating large annotated datasets by collecting labels from multiple annotators. In such scenarios one has to balance the tradeoff between the accuracy of the collected labels, the cost of acquiring these labels, and the time taken to finish the labeling task. With the goal of reducing the labeling cost, we introduce the notion of sequential crowdsourced labeling, where instead of asking for all the labels in one shot we acquire labels from annotators sequentially one at a time. We model it as an epsilon-greedy exploration in a Markov Decision Process with a Bayesian decision theoretic utility function that incorporates accuracy, cost and time. Experimental results confirm that the proposed sequential labeling procedure can achieve similar accuracy at roughly half the labeling cost and at any stage in the labeling process the algorithm achieves a higher accuracy compared to randomly asking for the next label.

1 Introduction

Annotating an unlabeled dataset is one of the major bottlenecks in using supervised learning methods to build good predictive models. Typically annotators who are task experts are hired to annotate the dataset. Getting the dataset labeled by expert annotators can be expensive and time consuming. With the advent of crowdsourcing marketplaces (Amazon Mechanical Turk (AMT) [1] being a prime example) it has become quite easy to acquire reasonably accurate labels from a large number of annotators in a short amount of time (see [21], [29] and [5] for some natural language processing and computer vision case studies). When exploring the option of crowdsourcing any labeling task one has to balance the tradeoff between three factors: the *accuracy* of the collected labels, the *cost* of acquiring these labels, and the *time* taken to finish the labeling task.

Accuracy: One drawback of most crowdsourcing marketplaces is that we do not have control over the quality of the annotators. The annotators can come from a diverse pool including genuine experts, novices, biased annotators, malicious annotators, and spammers. Hence in order to get good quality labels requesters typically get each instance labeled by multiple annotators and these multiple annotations are then consolidated either using a simple majority voting or more sophisticated methods that model and correct for annotator biases [8, 19] and/or task complexity [27]. Much of the recent work in the machine learning community has been in this area where the goal is to get an accurate estimate of the true labels based on the collected noisy labels from multiple annotators [15, 18, 16, 27]. In this paper we are more interested in the cost of acquiring the labels.

Cost: Typically all the labels are acquired in one shot. For example in AMT the *requesters* are able to pose tasks known as Human Intelligence Tasks (HITs). Annotators (called *workers*) can then browse among existing tasks and complete them for a small monetary payment set by the requester. In order to get multiple labels the task requester specifies that each HIT has to be completed by k workers. The workers then browse through the existing tasks and pick the one they are interested in. If we have n instances to be labeled then the total cost is proportional to nk labels. There are no standard guidelines on how to choose the right k. Large k will result in large cost while small k results in the loss of accuracy. One solution is to perform small pilot studies with different values of k and choose the smallest k that results in a desired consensus or accuracy. In practice requesters typically try values of k in the range from 3 to 10, depending on the task and

Appearing in Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33. Copyright 2014 by the authors.

	worker				
task	1	2	3	4	5
1	×	×	×	×	×
2	×	×	×	×	×
3	×	×	×	×	×
4	×	×	×	×	×
5	×	×	×	×	×

(a) complete labeling

	worker				
task	1	2	3	4	5
1	×	×	×		
2		×	×	×	
3			×	×	×
4	×		×		×
5	×	×			×

(b) AMT labeling

	worker				
task	1	2	3	4	5
1	×	×			
2		×	×	×	
3		×	×		
4		×	×		
5	×	×	×	×	

(c) sequential labeling

Figure 1: *Illustration of label acquisition from multiple workers.* (a) Each task is labeled by all the workers. (b) Each task gets labeled by exactly $k(=3)$ workers (AMT scenario). (c) The sequential labeling strategy where for each task labels are acquired one at a time. Each task will be labeled by a different number of workers.

the labeling budget. Some instances need more labels to reach a consensus while a lot of instances need very few labels to reach a good consensus. Hence asking for a fixed set of k labels in one shot for all the instances is not optimal in terms of the cost. This motivates the *sequential crowdsourced labeling* problem we tackle in this paper, instead of asking for k labels in one shot we acquire labels from annotators sequentially. Acquiring labels from multiple annotators can be thought of as filling the entries of a matrix where the row corresponds to an instance and the column corresponds to a worker (see illustration in Figure 1).

Time: The main disadvantage of using sequential methods over one shot labeling is that we increase the time required to finish the labeling task, since we need to wait for the labels to be provided before asking for the next label. However our experimental results show that the proposed method results in around 50% cost savings which can possibly justify the increase in the time required for completing the tasks.

Contributions: Our proposed sequential labeling algorithm has three core components. We first model the annotator labeling process and use a Bayesian approach to compute the posterior distribution of the true labels given the labels collected so far (§ 2). The sequential crowdsourced labeling problem is then modeled as an exploration/exploitation problem in an appropriately defined Markov Decision process (MDP) (§ 4). The reward function for the MDP (§ 3) is based on a decision theoretic utility function and specifically address the following issues.

1. ***When should the requester stop asking for more labels for a given instance?*** Once the labels from annotators are collected we can compute the posterior distribution of the true label (see § 2). Based on the posterior we have to decide whether to collect one more label for this instance or to stop collecting more labels. We take a decision theoretic approach (§ 3) to this problem by first specifying a (logarithmic) *utility function* and stopping when the maximum expected utility (also known as the *value function*) is $\geq \delta$, for a user defined value δ. For the logarithmic utility function the value function turns out to be the negative of the Shannon entropy of the label posterior.

2. ***Which annotator should the requester ask for a label from?*** Having specified the stopping criterion we can implement a sequential strategy by acquiring one label at a time from a randomly chosen annotator. This could be a reasonable strategy when all our annotators are experts. However this is not optimal in market places where the workers come from a diverse pool. One would like to invest only in those annotators who can provide a maximum increase in the value function, *prior to observing the label from the annotator*. In § 3 we quantify the expected change in the value function due to collecting one more label from an annotator. For the logarithmic utility function this turns out to be the *Lindley information* [14], which is the expected value of the Kullback-Leibler divergence between two distributions.

3. ***Incorporating labeling costs*** By casting the problem to a decision theoretic framework we can incorporate the actual labeling costs into the utility function (§ 3). In the AMT platform the cost is set by the requester, however in general the annotators can also specify the cost at which they are willing to provide the labels. Incorporating the labeling costs into the utility function allows us to balance the accuracy of the annotator with the cost of getting the label.

4. ***Task allocation in pull marketplaces*** We distinguish between two kinds of crowdsourcing marketplaces, the *push* and the *pull* marketplace. In the push marketplace (for example annotators hired to perform specific annotation tasks) the requesters push the task to the workers. Once a task is allocated the workers are guaranteed to finish the task. In contrast, in a pull market place (AMT being a prime example) the workers pull the tasks from the requesters. The requester posts tasks on the marketplace for a fixed price, the worker then goes through the list of tasks and takes up any task which he is interested in. From the sequential labeling perspective, this implies that even if we assign a task to a particular worker, we are not guaranteed that the worker will provide the label. In § 3 we show how to modify the utility function to handle worker selection in pull marketplaces.

2 Variational Bayes for approximating the posterior of the true label

The core inference primitive needed is the evaluation of the posterior distribution of the true labels given the labels collected so far. For ease of exposition we describe the method for binary labeling tasks and characterize each annotator by one parameter, the accuracy [1]. We use a fully Bayesian model in which the annotator accuracies are given prior distributions and are absorbed into the set of latent variables. This is relevant for our sequential algorithms since during the early stages we have few labels and we are essentially in the exploration phase when selecting the annotators for labeling. The consequence of this is that the estimated annotator model parameters for such annotators will have large standard errors. The Bayesian approach allows us to absorb this uncertainty into the estimation of the consensus label.

We have n instances to be labeled and a pool of m annotators. Let $y_i^j \in \{0,1\}$ be the binary label assigned to the i^{th} instance by the j^{th} annotator. Let $z_i \in \{0,1\}$ be the (unknown) true label. In a crowdsourcing setup an annotator labels only a few instances. Let $\mathcal{N}_j$ be the set of instances labeled by the j^{th} annotator (with $n_j = |\mathcal{N}_j|$) and $\mathcal{M}_i$ be the set of annotators who have labeled the i^{th} instance (with $m_i = |\mathcal{M}_i|$). Let $\theta^j := \Pr[y_i^j = z_i]$ be the *accuracy* of the j^{th} annotator. We will use the notation $\boldsymbol{\theta} = [\theta^1, \ldots, \theta^m]$ to refer to all the parameters of the model, $\boldsymbol{z} = [z_1, \ldots, z_n]$ the true labels (hidden variables), and $\boldsymbol{y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]$ the observed data where $\boldsymbol{y}_i = \{y_i^j\}_{j\in\mathcal{M}_i}$ is all the labels collected for the i^{th} instance.

Given the observed labels $\boldsymbol{y}$ the task is to estimate the posterior distribution $p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{y})$ of the true labels $\boldsymbol{z}$ and the annotator parameters $\boldsymbol{\theta}$. We assume a beta prior $p(\theta^j|a^j, b^j) = \text{Beta}(\theta^j|a^j, b^j) \propto (\theta^j)^{a^j-1}(1-\theta^j)^{b^j-1}$ for the annotator accuracies [2]. We use Bernoulli prior $p(z_i|p_i) = \text{Bernoulli}(z_i|p_i) = p_i^{z_i}(1-p_i)^{1-z_i}$ for z_i.

We use Variational Bayes (VB) methods [3] to approximate the required posterior distribution. The basic idea in the VB framework is to approximate the posterior over both the hidden variables and the parameters with a simpler distribution, usually one which assumes that the hidden variables and parameters are independent given the data. We constrain the posterior to be a simpler, factorised approximation to $p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{y}) \approx q_{\boldsymbol{z}}(\boldsymbol{z})q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$. VB iteratively minimizes the Kullback-Leibler divergence $\text{KL}\left[q_{\boldsymbol{z}}(\boldsymbol{z})q_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \parallel p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{y})\right]$ (which is equivalent to maximizing a lower bound on the marginal $p(\boldsymbol{y})$) with respect to the free distributions $q_{\boldsymbol{z}}(\boldsymbol{z})$ and $q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ by performing the following iterative updates: VBE-step: $q_{\boldsymbol{z}}^{(t+1)}(\boldsymbol{z}) \propto \exp\left[\int d\boldsymbol{\theta}\, q_{\boldsymbol{\theta}}^{(t)}(\boldsymbol{\theta}) \ln p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta})\right]$ and VBM-step: $q_{\boldsymbol{\theta}}^{(t+1)}(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}) \exp\left[\int d\boldsymbol{z}\, q_{\boldsymbol{z}}^{(t+1)}(\boldsymbol{z}) \ln p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta})\right]$.

As a consequence of the i.i.d instances, for the particular annotator model and the choice of priors these updates simplify to (see supplemental material for details): $q_{\theta^j}^{(t+1)}(\theta^j) = \text{Beta}\left(\theta^j|a^j + \sum_{i\in\mathcal{N}_j} q_{z_i}^{(t+1)}(y_i^j), b^j + \sum_{i\in\mathcal{N}_j} 1 - q_{z_i}^{(t+1)}(y_i^j)\right)$ for $j = 1, \ldots, m$ and $q_{z_i}^{(t+1)}(z_i) \propto p_i^{z_i}(1-p_i)^{1-z_i} \prod_{j\in\mathcal{M}_i} (A^j)^{\delta(y_i^j, z_i)} (B^j)^{1-\delta(y_i^j, z_i)}$ for $i = 1, \ldots, n$, where $A^j := \exp\left(\mathbb{E}_{q_{\theta^j}^{(t)}(\theta^j)}\left[\ln \theta^j\right]\right)$ and $B^j := \exp\left(\mathbb{E}_{q_{\theta^j}^{(t)}(\theta^j)}\left[\ln(1-\theta^j)\right]\right)$. If $x \sim \text{Beta}(a, b)$ then $\mathbb{E}[\ln x] = \psi(a) - \psi(a+b)$ and $\mathbb{E}[\ln(1-x)] = \psi(b) - \psi(a+b)$, where ψ is the digamma function, defined as the logarithmic derivative of the gamma function. At convergence $p(z_i|\boldsymbol{y}) \approx q_{z_i}(z_i)$ and $p(\theta^j|\boldsymbol{y}) \approx q_{\theta^j}(\theta^j)$. These updates are similar to the mean field updates derived in [15], except that we have additionally introduced a Bernoulli prior for each instance.

3 Value of collecting a new label

We are now interested in the situation where the labels are acquired sequentially one at a time. Consider an instance i for which the true label is $z_i \in \{0,1\}$. We will assume that we have a large pool of m workers/annotators denoted as $\mathcal{M}$. The requester can ask for labels $y_i^j \in \{0,1\}, j \in \mathcal{M}$ from multiple annotators. We will also assume that we have a push marketplace for crowdsourcing, that is we can ask for a label from a particular annotator and we are guaranteed to get the label. Acquiring a label y_i^j for instance i from annotator j incurs a finite *cost* $c_i^j > 0$ and has a probability θ^j of providing the correct label, that is, $\theta^j := \Pr[y_i^j = z_i]$ is the *accuracy* of the annotator. We take a decision theoretic approach to this problem by first specifying an *utility function* and then quantifying the change in utility due to collecting one more label.

[1]The methods proposed in this paper can be extended to more sophisticated annotator models proposed in the literature [19, 27, 28]. Two important extensions are the two-coin model and the task specific model. The two-coin model [19] uses two parameters per annotator, the sensitivity $\alpha^j := \Pr[y_i^j = 1|y_i = 1]$ and specificity $\beta^j := \Pr[y_i^j = 0|y_i = 0]$. The task-specific model [27] tries to model the difficulty of labeling each instance.

[2]In the absence of any strong prior we use uninformative priors. Commonly used uninformative priors are $\text{Beta}(\theta^j|1,1)$ (Bayes uniform prior), $\text{Beta}(\theta^j|1/2,1/2)$ (Jeffrey prior) and $\text{Beta}(\theta^j|0,0)$ (Haldane prior).

Prior and Posterior Let us say we have collected labels from k annotators until now for the i^{th} instance. Let $\pi_i^{(0)}(z_i) := p(z_i)$ denote the prior probability of z_i and let $\pi_i^{(k)}(z_i) := p(z_i|\boldsymbol{y}_i^{(k)})$ be the posterior probability of z_i after having collected k labels $\boldsymbol{y}_i^{(k)}$. The computation of the posterior depends on the specific annotator model we use. In § 2 we discussed how to approximate this posterior via a mean field approximation using the variational Bayes framework.

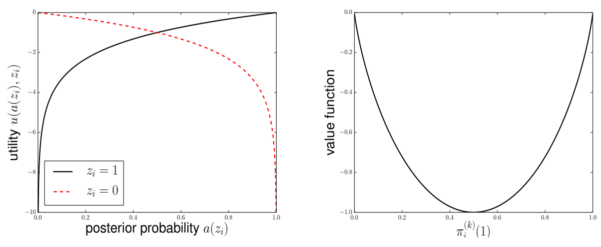
Utility specification The utility function $u(a(z_i), z_i)$ specifies the payoff you would receive when the true label is z_i and you predict the posterior as $a(z_i)$. We choose a logarithmic utility function defined as $u(a(z_i), z_i) = \log_2(a(z_i))$. The utility function is specified such that if the true label is z_i then we get a higher payoff or utility if the estimated posterior probability $a(z_i)$ is close to one and decreasing payoff as the posterior approaches zero (see Figure 2(a)).

Expected utility and Bayes decision From a decision theoretic perspective we choose the action (in our case the posterior $a(z_i)$) that maximizes the *expected utility*. The expected utility for action $a(z_i)$ is given by $\mathcal{U}_i(a(z_i)) = \mathbf{E}_{z_i}[u(a(z_i), z_i)] = \sum_{z_i \in \{0,1\}} \pi_i^{(k)}(z_i) \log_2(a(z_i))$, where the expectation is taken with respect to $\pi_i^{(k)}(z_i)$. Hence the Bayes action which maximizes the expected utility is $a^*(z_i) = \arg\max_{a(z_i)} \mathcal{U}_i(a(z_i)) = \pi_i^{(k)}(z_i)$.

Value function (maximum expected utility) The decision rule essentially tells us that $\pi_i^{(k)}(z_i)$ is the optimal posterior (under the specified utility function) if we stop after collecting k labels $\mathbf{y}_i^k$ and the expected utility of this decision is given by the *value function*

$$V_i(\pi_i^{(k)}) = \sup_{a(z_i)} \mathcal{U}_i(a(z_i)) = \sum_{z_i \in \{0,1\}} \pi_i^{(k)}(z_i) \log_2\left(\pi_i^{(k)}(z_i)\right). \tag{1}$$

This represents the *value* to the task requester of collecting k labels for the i^{th} instance. The value function is minimum when the posterior probability is 0.5 and increases in both directions to reach a maximum value of 0 (see Figure 2(b)). The negative of this value function is the Shannon entropy of the posterior $\pi_i^{(k)}(z_i)$. At the beginning of the data collection process when no labels have been collected so far the Bayes decision which maximizes the expected utility is essentially the prior $\pi_i^{(0)}(z_i)$. In the absence of any strong prior information on the true labels we use a Bernoulli prior $\pi_i^{(0)}(z_i) = p_i^{z_i}(1-p_i)^{1-z_i}$ with $p_i = 0.5$, which corresponds to the most uninformative prior and has an expected utility of -1. *A reasonable strategy to stop collecting labels for the i^{th} instance is when the value $V_i(\pi_i^{(k)}) \geq \delta$, for a user defined value δ close to zero.*



(a) utility function (b) value function

Figure 2: (a) The utility function $u(a(z_i), z_i)$. (b) The value function $V_i^{(k)}(\pi_i^{(k)})$ which is the maximum expected utility, as a function of the posterior probability $\pi_i^{(k)}(1)$

Expected value function If we now collect one more label we can recompute the posterior $\pi_i^{(k+1)}(z_i)$ and also the value function $V_i(\pi_i^{(k+1)})$. However we would like to know the value of collecting one more label, *prior to observing that label.* Let us say we ask for a label y_i^j from the j^{th} annotator. The annotator can give a label either 0 or 1 based on his accuracy. Hence we compute the expectation of $V_i(\pi_i^{(k+1)})$ with respect to the marginal [3] of the label y_i^j and define the expected change in the value function as

$$\mathcal{V}_i^j = \left[\sum_{y_i^j \in \{0,1\}} V_i(\pi_i^{(k+1)}|y_i^j) p(y_i^j)\right] - V_i(\pi_i^{(k)}). \tag{2}$$

This quantity is known as the *expected value of sample information.* Using (1), $\mathcal{V}_i^j$ can be written as

$$\begin{aligned}\mathcal{V}_i^j &= \sum_{y_i^j} p(y_i^j) \sum_{z_i} \pi_i^{(k+1)}(z_i) \log_2\left(\frac{\pi_i^{(k+1)}(z_i)}{\pi_i^{(k)}(z_i)}\right) \\ &= E_{y_i^j}\left[E_{z_i|y_i^j}\left[\log_2\left(\frac{\pi_i^{(k+1)}(z_i)}{\pi_i^{(k)}(z_i)}\right)\right]\right].\end{aligned}$$

This quantity is also known as the *Lindley information* [14], which is essentially the expected value with respect to y_i^j of the Kullback-Leibler (KL) divergence between $\pi_i^{(k+1)}(z_i)$ and $\pi_i^{(k)}(z_i)$. Since V is convex and $\mathcal{V}_i^j$ can be written as $E_{y_i^j}[V_i(\pi_i^{(k+1)})] - V_i(E_{y_i^j}[\pi_i^{(k+1)}])$, from Jensen's inequality we have $\mathcal{V}_i^j \geq 0$. This means that the expected utility monotonically increases from

[3]For the annotator model defined in § 2 the marginal $p(y_i^j)$ can be computed as follows $p(y_i^j) = \sum_{z_i \in \{0,1\}} \pi_i^{(k)}(z_i) \int_{\theta^j} (\theta^j)^{\delta(y_i^j, z_i)} (1-\theta^j)^{1-\delta(y_i^j, z_i)} p(\theta^j|\mathbf{y}) d\theta^j$. Since $\theta^j \sim \text{Beta}(\theta^j|\alpha, \beta)$ (with α and β as defined in the VB updates) the marginal can be computed as $p(y_i^j) = \sum_{z_i \in \{0,1\}} \pi_i^{(k)}(z_i) \frac{B(\alpha + \delta(y_i^j, z_i), \beta + 1 - \delta(y_i^j, z_i))}{B(\alpha, \beta)}$, where $B(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1} dt$ is the beta function.

-1 to 0 as we collect more labels. The term $\mathcal{V}_i^j$ essentially quantifies the expected increase in utility by asking for a label from annotator j.

Incorporating labeling costs We can now incorporate the labeling costs into the utility. Acquiring a label y_i^j for instance i from annotator j incurs a finite *cost* $c_i^j > 0$. Let $c_i(k) := \sum_{j=1}^k c_i^j$ denote the total cost of acquiring k labels for the i^{th} instance. The value function (1) incorporating the labeling costs can now be written as $V_i(\pi_i^{(k)}) = -c_i(k) + \sum_{z_i \in \{0,1\}} \pi_i^{(k)}(z_i) \log_2 \left(\pi_i^{(k)}(z_i)\right)$. Similarly the expected value of collecting one more label can be written as $\mathcal{V}_i^j = -c_i^j + \left[\sum_{y_i^j \in \{0,1\}} V_i(\pi_i^{(k+1)}|y_i^j) p(y_i^j)\right] - V_i(\pi_i^{(k)})$. Using this expression now allows us to balance the accuracy with the cost. An accurate annotator may not necessarily contribute to the largest change in utility if the cost of labeling is very high.

Extension to the pull marketplace In the previous sections we assumed that we had a push model. This guaranteed us that if we asked for a label from an annotator we were sure to get it at the cost specified by the annotator. The push model is suitable when annotators are internally hired to work on our annotation tasks. However most commercial crowdsourcing market places like AMT work based on a pull model where workers can browse among existing tasks and complete them for a monetary payment set by the requester. So while we may like to get a label from a particular worker we are not guaranteed that the worker will be interested in completing the task. Workers are free to ignore any HITs that are not to their liking or that don't seem to pay enough to be worth doing. In order to account for this we keep track of the number of tasks accepted by a particular worker and compute the quantity p^j as the probability that the worker j would accept a task. A less accurate worker who always accepts the tasks may result in a higher utility than a highly accurate worker who seldom accepts the tasks. We can now incorporate this probability into the utility function thus balancing the accuracy of the worker, the cost for acquiring the label, and the probability that the worker accepts the task (this would indirectly influence the time taken to complete the task). While there are many ways to penalize smaller p^j the following utility function works well in practice.

$$\mathcal{V}_i^j = \frac{1}{p^j}\left[-c_i^j + \left[\sum_{y_i^j \in \{0,1\}} V_i(\pi_i^{(k+1)}|y_i^j) p(y_i^j)\right] - V_i(\pi_i^{(k)})\right] \tag{3}$$

So when p^j decreases the expected value of collecting one more label $\mathcal{V}_i^j$ increases. The probability p^j can be initially set to 1 and updated as the labeling proceeds. In practice we compute the confidence interval (using the Jeffreys method [6]) for p^j and use the upper limit of the confidence interval in the utility function.

4 Markov Decision Process

Using the utility function defined earlier we now model the sequential crowdsourced labeling problem as an exploration/exploitation problem in an appropriately defined Markov Decision process (MDP).

A MDP is a natural framework to model sequential decision making problems under uncertainty. A MDP [17] is a four tuple: $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$, where $\mathcal{S}$ is a finite set of *states*, $\mathcal{A}(s)$ is a finite set of *actions* available in state s, $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ denotes the *transition probabilities* between the states, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto [0, R_{max}]$ is positive bounded reward function. If the agent is in state s and performs action a, then $\mathcal{P}(.|s,a)$ is the distribution over next possible states and $\mathcal{R}(s,a)$ is the expected immediate reward received. The state transitions possess the *Markov property*, given s and a, the next state is conditionally independent of all previous states and actions.

In our setup at any time t the state s_t corresponds to the set of (instance,annotator) pairs which have been labeled so far. An action a_t corresponds to querying for the label of the i^{th} instance from the j^{th} annotator, and the acquired label is represented as y_i^j. Since we do not allow repeated labeling, at any given state s_t the set of actions available corresponds to the all the (instance,annotator) pairs which have not yet been queried for labels so far. The action a_t changes the state to s_{t+1}, the transition probabilities $\mathcal{P}(s_{t+1}|s_t, a_t)$ are given by the marginals $p(y_i^j)$. The expected immediate reward is given by utility function (3).

The behavior of an agent in an MDP is modeled using the notion of a *policy*. A deterministic stationary policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ determines what action to take depending only on the current state. If the dynamics of the MDP are known, a policy can be found mapping states to actions that maximizes the expected discounted reward by solving the recursively defined *Bellman optimality equations* [4], $Q(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q(s',a')$, and selecting action $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q(s,a)$. The three standard methods [20] used are value iteration, policy iteration, and linear programming.

Since our defined MDP has a very large state space, direct computation with the above methods is infeasible. So we take a greedy strategy and ask for a label which maximizes the reward function. This essentially corresponds to the first step of the value iteration algorithm and can be viewed as a local multi armed bandit approximation [9] to the MDP.

ϵ-greedy exploration in MDP If the model (transition probabilities) are known then we can get a near-optimal policy. In our case the parameters of the model are also re-estimated after each action. Hence one needs to also introduce an *exploration phase* to explore the state space. We use ϵ-greedy exploration [25], where the agent chooses actions greedily with probability $1-\epsilon$ and chooses actions randomly with probability ϵ. The ϵ-greedy exploration ensures convergence to an optimal policy in the limit and it is often suggested [23, 22] that this algorithm is hard to beat compared to more sophisticated exploration strategies.

5 Related work

Much of the recent work in the machine learning community has focused on getting an accurate estimate of the true labels based on the collected noisy labels from multiple annotators [15, 18, 16, 27]. In this paper we are interested both in the accuracy and also the cost of acquiring the labels. There are some methods proposed to detect spammers [18, 11] with the goal of eliminating them from further labeling in order to reduce the costs. Some of the recent works [13, 12] have introduced approaches that are budget-optimal and selectively allocates the tasks to the annotators. Most of these works consider only *one-shot setting*, where all the tasks are given at once to the annotators and then all the labels are obtained simultaneously. Recently [24] proposed an active learning approach which asks for image labels that are the most informative, but assumes the expertise of annotators to be fixed and known. algorithm to assigning tasks to workers.

The closest related work is the algorithm in [26] where they propose an online algorithm for estimation of annotators expertise and the ground truth. Our proposed algorithm differs from this work in the following aspects: (1) They use an EM algorithm to compute the MAP estimate for the annotator parameters, whereas our proposed algorithm is full Bayesian. (2) They use a threshold directly on the posterior to decide when to stop collecting labels. Our proposed approach is more principled and is based on a decision theoretic approach. (3) At each iteration they maintain a list of spammers and eliminate them from further labeling. Our approach inherently does not let the requester select any spammer till it has exhausted all the good annotators. (4) Our proposed approach can easily incorporate costs and can be directly extended to the pull marketplaces. In spirit this is quite similar to the sequential labeling strategy where the workers are chosen randomly, except that they maintain a list of experts and spammers at each round and eliminate the spammers from further labeling. In the experimental section we compare our algorithm with this variant.

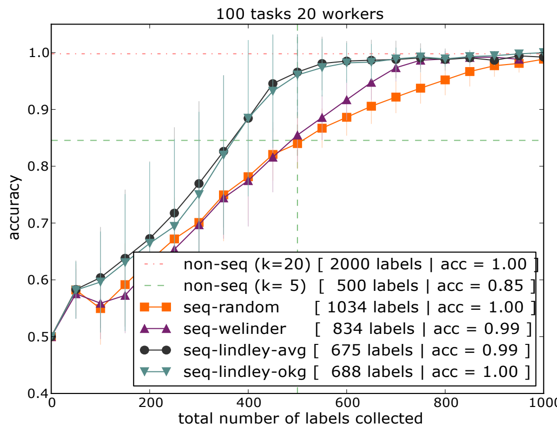
Recently [7] also cast the sequential labeling problem as an MDP and proposed to use the optimistic knowledge gradient as an approximate allocation policy. We differ from this work in the following aspects: (1) They consider only the accuracy in the reward function. Our proposed utility function is quite different and incorporates the notion of both cost and accuracy and is specifically designed for pull marketplaces. (2) Unlike our approach the algorithm just greedily exploits and does not have an exploration phase. (3) They used the maximum instead of the average in defining the reward function (which they termed it as the optimistic knowledge gradient based on the earlier work of [10]). In our setting this corresponds to

$$\mathcal{V}_i^j = \left[\max_{y_i^j \in \{0,1\}} V_i(\pi_i^{(k+1)}|y_i^j)\right] - V_i(\pi_i^{(k)}). \quad (4)$$

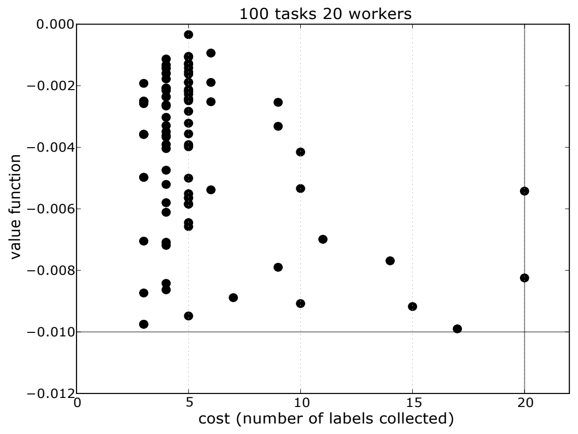
We experimentally also compare our algorithm with a corresponding variant where instead of the average we use the maximum in the value function.

6 Experimental validation

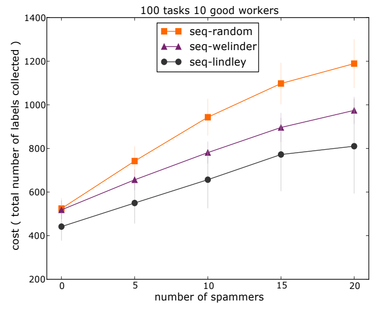
Simulated data We first experimentally validate the proposed sequential labeling algorithm using simulated data. We first sample 100 instances with equal prevalence for positives and negatives. We simulate labels from a pool of 20 annotators with randomly chosen accuracies. Figure 3(a) and (b) plots the accuracy and the average value per task as a function of the total number of labels collected. The results are averages over 100 repetitions. The plot compares the following five methods: (1) ***non-seq (k=20)*** This corresponds to the non-sequential approach where we collect labels from all the 20 annotators. This essentially has the maximum accuracy (1.00) that can be achieved and costs a total of 2000 labels. (2) ***non-seq (k=5)*** This is also a non-sequential approach where we collect 5 labels per task from randomly chosen annotators. This costs us 500 labels and achieves an accuracy of 0.85. This is the approach typically used on the AMT marketplace, where the workers randomly choose the tasks to work on and the labeling stops when we have k labels per task. The main drawback is that unless k is large we are not guaranteed accurate labels if the market place has a lot of spammers. The next three methods are the sequential labeling approaches. For all the three approaches we stop collecting more labels when the value function for each task is above $\delta = -10^{-2}$. In order to see the trend at each round we ask for labels from 50 tasks. (3) ***seq-random*** This is the sequential labeling strategy where the next an-



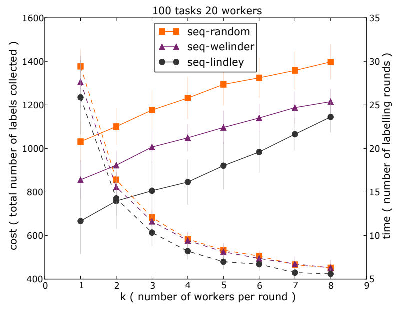
(a) accuracy



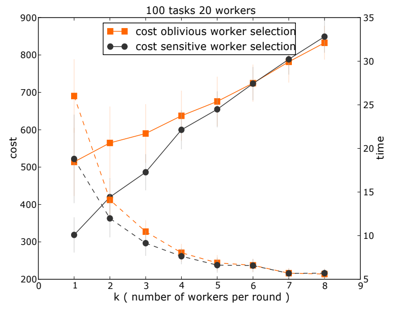
(b) cost vs value



(c) spammers



(d) number of workers



(e) costs

Figure 3: *Experiments on simulated data (see § 6)* We simulate labels from a pool of 20 annotators with randomly chosen accuracies. (a) Accuracy as a function of the total number of labels collected for different sequential labeling strategies. (b) The number of labels collected for each task on the x-axis and the corresponding value on the y-axis for our proposed method (seq-lindley-avg). (c) The total number of labels collected as a function of the number of spammers. (d) The total number of labels collected (solid line) and the number of rounds (dotted line) as a function of the number of workers per round. (e) The total cost for the seq-lindley-avg method with and without including the labeling cost.

notator is randomly chosen from the pool of annotators. (4) ***seq-welinder*** This is the sequential labeling strategy proposed in [26]. This is essentially same as seq-random except that at each round we eliminate spammers from the labeling process. (5) ***seq-lindley-avg*** This is our proposed sequential labeling strategy which does an $\epsilon(=0.1)$-greedy exploration in an MDP with the reward function based on the average value function (see Eq (2)). (6) ***seq-lindley-okg*** This is same as the earlier method except that instead of the average we use the maximum value of the value function (see Eq (4)) as the reward as proposed in [7].

From Figure 3(a) it can be seen that at any stage of the labeling process the proposed seq-lindley-avg algorithm achieves the highest accuracy. At termination all three strategies achieve the same accuracy. However seq-lindley uses the least amount of labels (675 labels, a 35% reduction over seq-random and a 66% reduction over using all the annotators). Even if we had a limited budget of say 500 labels, we achieve a much higher accuracy using the sequential approach rather than collecting all 500 labels in one shot. The seq-welinder algorithm is very similar to seq-random during the early stages, because we do not tag an annotator as a spammer until we have enough data. There is also no significant difference between seq-lindley-avg and seq-lindley-okg and for further experiments we show the results only for seq-lindley-avg.

The sequential algorithm collects different number of labels for different tasks. This is illustrated in Figure 3(b) which plots the number of labels collected for each task on the x-axis and the corresponding value on the y-axis.

Effect of spammers Figure 3(c) shows the difference in cost among the three methods as a function of the number of spammers. We start with an initial pool of 10 good workers (accuracies > 0.7) and keep adding spammers (with accuracy 0.5). The proposed algorithm has the lowest cost.

	cost (# of labels)				**% reduction in cost**			**accuracy**			
		——–*sequential labeling*——–			——–*sequential labeling*——–				——–*sequential labeling*——–		
dataset	*orig*	*random*	*welinder*	*lindley*	*random*	*welinder*	*lindely*	*orig*	*random*	*welinder*	*lindely*
anger	1000	462	439	385	53.8%	56.1%	61.5%	0.96	0.97	0.97	0.96
disgust	1000	463	444	409	53.7%	55.6%	59.1%	1.00	0.99	1.00	1.00
fear	1000	427	396	385	57.3%	60.4%	61.5%	0.91	0.90	0.88	0.91
joy	1000	419	414	349	58.1%	58.6%	65.1%	0.89	0.89	0.89	0.89
sadness	1000	478	493	451	52.2%	50.7%	54.9%	0.94	0.93	0.94	0.95
surprise	1000	386	388	343	61.4%	61.2%	65.7%	0.91	0.91	0.91	0.91

Table 1: *Cost savings for the affective text analysis data [21] collected via Amazon Mechanical Turk*(see § 6) Each dataset contains 100 tasks and 38 distinct workers. Each task is labeled by 10 workers thus the total original cost being 1000 labels. *seq-r*, *seq-w* and *seq-l* refer to the three sequential crowdsourcing methods.

Getting labels from k workers Our proposed algorithm asks for labels from one worker at a time. However practically it makes sense to ask for labels from k annotators due to two reasons (1) In pull market places since there is a chance that the worker may not accept the task we can increase our chance of acquiring a label by asking for labels from more workers. (2) Asking for k labels at once reduces the annotation time. However the downside is that we may end up acquiring more labels than needed. This is illustrated in Figure 3(d) where we compare the cost (solid line) and the number of rounds (dotted line) as a function of k.

Effect of labeling costs In this experiment we incorporate labeling costs. We assume each worker has a labeling cost proportional to his accuracy. Figure 3(e) illustrates that incorporating costs into the utility function leads to a lower total cost of labeling.

Experiments on data collected via Amazon Mechanical Turk In this section we perform experiments using this publicly available dataset [2] collected by [21]. We specifically use the six affective analysis datasets (see Table 1), wherein each annotator is presented with a list of short headlines, and is asked to give numeric judgments in the interval [0,100] rating the headline for six emotions: anger, disgust, fear, joy, sadness, and surprise. The dataset contains 100 tasks and 38 distinct annotators. Each task is labeled by a random set of 10 annotators. Each annotator on the average has labeled 26 tasks. For our experiments we threshold the numeric ordinal ratings to binary ratings. For all the datasets the original gold standard labels are also available. Since each task is labeled by 10 annotators we have a total of 1000 labels. Using this dataset we can consolidate the labels using our proposed variational Bayes approach and evaluate the accuracy of the resulting consensus ground truth using the gold standard labels.

The goal of this experiment is to analyze if using the proposed sequential crowdsourcing approach, the same accuracy could have been achieved at a reduced cost (that is, using fewer labels). Table 1 summarizes the results where we compare the following three methods, (a) *orig.* (using the entire original dataset), (b) *random* (sequential crowdsourcing where workers are sampled randomly), (c)*welinder* (the methods proposed by [26]) and (d) *lindely* (proposed algorithm). For the sequential methods for each round we query labels for all the 100 tasks tasks one worker at a time. Note that unlike the earlier simulation setup, we are now in a pull mode, where if we ask for a label from a worker it is not guaranteed that he will provide the label. We set the stopping criterion to $\delta = -0.01$ and the maximum number of workers per task to 10. We compare these three approaches in terms of cost and accuracy. The following observations can be made from Table 1: (1) All the sequential strategies achieve similar accuracies as compared to using all the labels from 10 workers. (2) The sequential strategies can achieve the same accuracies as the original dataset at roughly half the cost (number of labels), resulting in a $50\% - 65\%$ reduction of cost. The sequential method based on Lindley information results in a higher reduction in cost compared to the sequential strategy where workers are sampled randomly and also the sequential strategy proposed in [26]. In summary the proposed sequential labeling procedure can achieve similar accuracy at roughly half the labeling cost.

7 Conclusions

In this paper we proposed a Bayesian decision theoretic algorithm for sequential crowdsourced labeling that balances the accuracy of the collected labels and the cost of collecting them by casting it as an exploration/exploitation tradeoff in a Markov Decision Process. While we assumed that the annotator accuracy are fixed the proposed algorithm can easily handle the situation where the annotator accuracies change over time. The only change needed is to use the current posterior as the prior and use only the most recent labels to update the posterior. The notion of utility function gives the task requester flexibility to specify his design constraints, for example, one could also incorporate the time taken by a worker to complete the task into the utility function.

References

[1] URL `https://www.mturk.com`.

[2] URL `https://sites.google.com/site/nlpannotations/`.

[3] M. J. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London, 2003.

[4] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[5] T. Berg, A. Sorokin, G. Wang, D. Forsyth, D. Hoiem, I. Endres, and A. Farhadi. It's All About the Data. *Proceedings of the IEEE*, 98(8):1434–1452, 2010.

[6] L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16:101–133, 2001.

[7] X. Chen, Q. L., and D. Z. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 64–72, 2013.

[8] A. P. Dawid and A. M. Skene. Maximum likeihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

[9] M. O. Duff and A. G. Barto. Local bandit approximation for optimal learning problems. In *Advances in Neural Information Processing Systems*. 1997.

[10] P. I. Frazier, W. B. Powell, and S. Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

[11] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, 2010.

[12] D. Karger, S. Oh, and D. Shah. Budget-optimal crowdsourcing using low-rank matrix approximations. In *49th Annual Allerton Conference on Communication, Control, and Computing*, pages 284–291. 2011.

[13] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in Neural Information Processing Systems 24*, pages 1953–1961. 2011.

[14] D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

[15] Q. Liu, J. Peng, and A. Ihler. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems 25*, pages 701–709. 2012.

[16] W. P., S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23*, pages 2424–2432. 2010.

[17] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, NY, 1994.

[18] V. C. Raykar and S. Yu. Ranking annotators for crowdsourced labeling tasks. In *Advances in Neural Information Processing Systems 24*, pages 1809–1817. 2011.

[19] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11: 1297–1322, April 2010.

[20] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*. Englewood Cliffs: Prentice hall, 1995.

[21] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and Fast—but is it good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, 2008.

[22] A. L. Strehl and M. L. Littman. An empirical evaluation of interval estimation for Markov Decision Processes. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 128–135, 2004.

[23] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the European Conference on Machine Learning (ECML 2005)*, pages 437–448. 2005.

[24] S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2262–2269, 2009.

[25] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[26] P. Welinder and P. Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *Workshop on Advancing Computer Vision with Humans in the Loop at CVPR 2010*, 2010.

[27] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22*, pages 2035–2043. 2009.

[28] Y. Yan, R. Rosales, G. Fung, M. Schmidt, G. Hermosillo, L. Bogoni, L. Moy, and J. Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS) 2010*, pages 932–939, 2010.

[29] O. F. Zaidan and C. Callison-Burch. Crowdsourcing Translation: Professional Quality from Non-Professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229, 2011.