Bayesian Inference and Reinforcement Learning For Incentive Mechanism Design In Crowdsourcing

Anonymous Authors[1]

Abstract

In crowdsourcing, incentive mechanisms are developed to theoretically ensure that workers will provide high-quality labels. Existing incentive mechanisms, which simply compares the reports of workers, cannot fully exploit the information contained in workers' reports. Meanwhile, they heavily rely on an impractical assumption that workers are all fully rational. On the other hand, various machine learning algorithms have empirically shown good performance in exploiting the information contained in workers' reports. However, due to the lack of theoretical supports, they cannot be used for incentive mechanism design.

In this paper, we propose a novel learning-based incentive mechanism to bridge these two disconnected communities. To achieve this objective, we develop a Bayesian inference algorithm to analyze workers' reports, and the incentives are determined based on the inference results. In addition, we develop a reinforcement learning algorithm to track workers' response to the incentives and adjust the incentives. Then, we prove that our learning-based incentive mechanism can also theoretically ensure that workers will provide high-quality labels. Besides, our empirical results show that the Bayesian inference algorithm can improve the robustness and stability of incentives. The reinforcement learning algorithm can adapt our mechanism to different worker models and thus boost the utilities of the data requester.

1. Introduction

The ability to quickly collect large scale and high quality labeled datasets is crucial for Machine Learning (ML), and more generally for Artificial Intelligence. Among all proposed solutions, one of the most promising ones is crowdsourcing (Slivkins and Vaughan, 2014; Difallah *et al.*, 2015; Simpson *et al.*, 2015). The idea is neat - instead of using a centralized amount of efforts, the to-be-labeled tasks are disseminated to a decentralized crowd of workers to parallelize the collection procedure, leveraging the power of human computation. Nonetheless, it has been noted that crowdsourced data often suffers from quality issues, due to its unique feature of no monitoring and no ground-truth verification of workers' contributed data.

This quality control challenge has been attempted by two relatively disconnected research communities. From the more ML side, quite a few inference techniques have been developed for inferring true labels from crowdsourced and potentially noisy labels (Dawid and Skene, 1979; Raykar *et al.*, 2010; Liu *et al.*, 2012; Chen *et al.*, 2015; Zheng *et al.*, 2017). These solutions often work as a one-shot, post-processing procedure facing a static set of workers, whose labeling accuracy is fixed and *informative*. Despite its nice theoretical contribution and empirical success, the above methods ignored the effects of *incentives* when dealing with human inputs. It has been observed both in theory and practice that xxx(Liu and Chen, 2017b), without appropriate incentive, selfish and rational workers can easily chose to contribute low quality, uninformative, or even malicious data. Existing inference algorithms are very vulnerable in these cases - either much more redundant crowdsourced labels will need to be collected (low quality inputs), or the methods will simply fail to work (the case with uninformative and malicious inputs).

From the less ML setting, the above quality control question has been studied in the context of *incentive mechanism design*. In particular, a family of mechanisms, jointly called *peer prediction*, has been proposed towards addressing above incentive challenges (Prelec, 2004; Gneiting and Raftery, 2007; Jurca *et al.*, 2009; Witkowski and Parkes, 2012; Radanovic and Faltings, 2013; Dasgupta and Ghosh, 2013). Existing peer prediction mechanisms focus on achieving incentive compatibility (IC), which is defined as reporting truthfully a private data, or reporting a high quality data, will maximize the expected payment issued to workers. They achieve IC via comparing the reports between the targeted worker, and a randomly selected reference worker, to bypass the challenge of no ground-truth verification.

Nonetheless, we note several undesirable properties of existing peer prediction mechanisms. Firstly, from the label inference studies (Zheng *et al.*, 2017), we can know that all

the collected labels are correlated and this correlation contains a wealth of information about the true labels and the quality of workers. However, existing peer prediction mechanisms purely rely on the reports of the reference worker, which only represents a limited share of the overall collected information. This way of design will inevitably lower the robustness, and meanwhile increase the variance of payment (which is unfavorable in practice.) Secondly, existing peer prediction mechanisms simplify workers' responses to the incentive mechanism by assuming that workers are all fully rational and only follow the utility-maximizing strategy. However, there have been many studies showing that human agents may be bounded-rational and even keep improving their responding strategies in practice (Simon, 1982; McKelvey and Palfrey, 1995; Chastain *et al.*, 2014).

Currently, the connection between ML and incentive mechanism design is far from being satisfactory. In this paper, we propose a *learning-based incentive mechanism*, aiming to merge and extend the techniques in the two areas to address the caveats when they are employed alone, as discussed above. The high level idea is as follows: we divide the large dataset into relatively small task packages. At each time step, we employ workers to handle one task package and use a certain inference algorithm to learn the true labels and workers' accuracy in their reports. Then, we use the learned worker accuracy o determine the payments for workers. Meanwhile, driving in the background, we develop a reinforcement learning algorithm to adjust the payments based on workers' historical responses to incentivizes. By doing so, our incentive mechanism can be adapted to different types of workers.

However, as the first work to combine the label inference and reinforcement learning algorithms with incentive mechanism design, there is a line of challenges. In light of the challenges, we summarize the core contributions of this paper as follows:

- In order to achieve good incentive property and calibrate the accuracy workers' reported information (for training our RL algorithm), there is a need of having an unbiased inference algorithm. Unfortunately this is not the case with existing inference method. We propose a novel one-shot peer prediction mechanism based on Bayesian inference by firstly deriving the explicit posterior distribution of true labels and then employing Gibbs sampling for inference. The most challenging problem of our mechanism is to prove the convergence of above inference method.
- Since workers' states and label accuracy both are unobservable, we need to estimate them via the label inference algorithm. However, these estimates, which are the means over all tasks, are approximately corrupted with Gaussian noise. Thus, we develop our reinforcement learning algorithm based on the data-driven Gaussian process regression.
- Besides, we conduct empirical evaluation, and the results show that our mechanism can improve the robustness and lower the variance of payments.
- Meanwhile, in the long term, our mechanism significantly increase the utility of the data requester under different worker models, such as fully rational, bounded rational and learning agent models.

2. Related Work

Our work is inspired by the following three literatures

Peer Prediction: This line of works, addressing the incentive issues for reporting high quality data without verification, starts roughly with the seminar works (Prelec, 2004; Gneiting and Raftery, 2007). A series of follow-up works have relaxed various assumptions of this family of mechanisms (Jurca *et al.*, 2009; Witkowski and Parkes, 2012; Radanovic and Faltings, 2013; Dasgupta and Ghosh, 2013).

Inference method: Recently, Inference method have been applied to crowdsourcing settings, aiming to uncover the true labels from multiple noisy reported copies. Notably success include EM method (), Belief Propagation (), and Variational Inference ().

Reinforcement Learning: XXX.

Our work differs from above literature in the connection between incentive design with ML techniques. There have been very few recenct studies that have similar taste as ours. For example, to improve the long-term utility of the data requester in crowdsourcing, Liu and Chen (2017b) develop a multi-armed bandit algorithm to adjust the state-of-the-art peer prediction mechanism, DG13 (Dasgupta and Ghosh, 2013). However, both the bandit algorithm and DG13 still need to assume that workers are fully rational and will behave as we desired. Instead of randomly choosing a reference worker, Liu and Chen (2017a) propose to use supervised learning algorithms to generate the reference reports based on the contextual information of tasks and derive the incentive compatibility conditions for the supervised-learning-based peer prediction mechanisms. In this paper, without assuming the contextual information about tasks, we develop an unsupervised-learning algorithm to learn the worker models and true labels from.

3. Problem Formulation

This paper focuses on typical crowdsourcing problems. To be more specific, at each time step t, one data requester asks $N \geq 3$ candidate workers to label M tasks with binary answer space $\{1, 2\}$. We use $L_i^t(j)$ to denote the label

worker i generates for task j at time t. For simplicity of computation, we reserve $L_i^t(j) = 0$ if j is not assigned to i. Furthermore, we use $\mathcal{L}$ and $\boldsymbol{L}$ to denote the set of ground-truth labels and the aggregate collected labels respectively.

The generated label $L_i^t(j)$ depends both on the ground-truth $\mathcal{L}(j)$ and worker i's internal state, which mainly consists of two components, effort level (high or low) and reporting strategy (truthful or deceitful). At any given time for any task, workers at their will adopt an arbitrary combination of effort level and report strategy. We thus define $\mathsf{eft}_i^t \in [0,1]$ and $\mathsf{rpt}_i^t \in [0,1]$ as worker i's probability of exerting high efforts and reporting truthfully for task j respectively. Furthermore, following Dasgupta and Ghosh (2013); Liu and Chen (2017b), we assume that tasks are homogeneous and workers share the same probability of generating the correct labels if they exert the same level of efforts. We denote the probability as $\mathbb{P}_H$ and $\mathbb{P}_L$ respectively. Note we require $\mathbb{P}_H > \mathbb{P}_L \geq 0.5$, where 0.5 is used if workers randomly label the tasks. Worker i's probability of being correct (PoBC) at time t for any given task is

$$\begin{aligned}\mathbb{P}_i^t = \ & \mathsf{rpt}_i^t\mathsf{eft}_i^t\mathbb{P}_H + (1-\mathsf{rpt}_i^t)\mathsf{eft}_i^t(1-\mathbb{P}_H) + \\ & \mathsf{rpt}_i^t(1-\mathsf{eft}_i^t)\mathbb{P}_L + (1-\mathsf{rpt}_i^t)(1-\mathsf{eft}_i^t)(1-\mathbb{P}_L)\end{aligned} \tag{1}$$

At each time t, the data requester pays worker i some money P_i^t as the incentive for providing labels. At the beginning of each time step, the two parties mutually agree to a certain rule of payment determination, which would not be changed until the next time step. The workers are self-interested and may change their internal states according to the incentive. It is no surprise that workers' different internal sates would lead to different PoBCs and finally different label qualities. After collecting the generated labels, it is a common procedure for the data requester to infer true labels $\tilde{L}^t(j)$ by running some inference algorithm. Please refer Zheng *et al.* (2017) for a good survey of the existing inference algorithms. The aggregate accuracy A^t and the data requester's utility u^t satisfy

$$\begin{aligned}A^t &= \frac{1}{M}\sum\nolimits_{j=1}^{M} 1\left[\tilde{L}^t(j) = \mathcal{L}(j)\right] \\ u^t &= F(A^t) - \eta\sum\nolimits_{i=1}^{N} P_i^t\end{aligned} \tag{2}$$

where $F(\cdot)$ is a non-decreasing monotonic function mapping accuracy to utility and η is a tunable parameter balancing label quality and costs. Intuitively, $F(\cdot)$ function is usually non-deceasing as higher accuracy is preferred.

The number of tasks in crowdsourcing is often very large, and the interaction between tasks and workers may last for hundreds of time steps. Thus, we introduce the cumulative utility $U(t)$ of the data requester from the current step t as

$$U^t = \sum\nolimits_{k=1}^{\infty} \gamma^k u^{t+k} \tag{3}$$

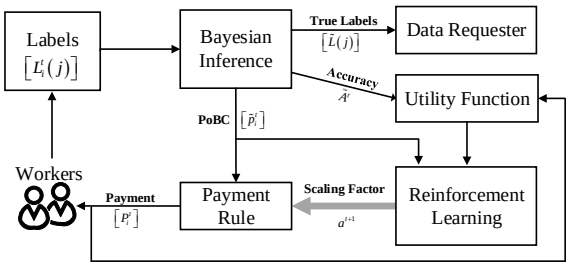


Figure 1. Layout of our incentive mechanism.

where $0 \leq \gamma < 1$ is the discount rate for future utilities. The objective of our study is to maximize U^t by optimally designing the incentive mechanism.

4. Incentive Mechanism for Crowdsourcing

We present the overall layout of our mechanism in Figure 1, where estimated values are denoted with over-the-head tildes. In this section, we formally introduce the three main components of our design: payment rule, bayesian inference and reinforcement learning.

The payment rule is designed to ensure that reporting truthfully and exerting high efforts is the payment-maximizing strategy for all workers at any given time. Besides, our incentive mechanism as a whole guarantees that this is also the payment-maximizing strategy for all workers in the long run. We kindly refer readers to Section 5 for the theoretical proof. This property prevents clever manipulations from workers, which bring more long-term benefits to them by sacrificing short-term ones, or the other way around. By doing so, we wish to induce workers to generate high-quality labels. The Bayesian inference algorithm is responsible for estimating the true labels, workers' PoBCs and the aggregate label accuracy from the collected labels at each time step. It utilizes soft Dirichlet priors and Gibbs sampling to prevent overestimation of accuracy when workers generate poor-quality labels. The reinforcement learning algorithm adjusts the payment rule based on the historical data of payments, workers' PoBCs and aggregate labels' accuracy, aiming to optimally balance the utility gain from high accuracy and loss from larg payments, which corresponds to $F(A^t)$ and $\sum_i P_i^t$ in Equation 2 respectively.

4.1. Payment Rule

Suppose, at time step t, worker i finishes m_i^t tasks. For any given time t, all m_i^ts as a whole must satisfy $\sum_i m_i^t = M$. Then the payment for worker i at time t is

$$P_i^t = m_i^t \cdot (a^t\mathsf{sc}_i^t + b) \ , \ \mathsf{sc}_i^t = \tilde{\mathbb{P}}_i^t - 0.5 \tag{4}$$

where sc denotes worker i's score, which is proportional to his PoBC $\tilde{\mathbb{P}}_i^t$ calculated by the Bayesian inference algorithm.

$b \geq 0$ is a constant representing the fixed base payment even if the worker purely returns random labels. We use $a^t \in \mathcal{A}$ to denote the scaling factor, determined by our reinforcement adjustment algorithm at the beginning of every time step t. We further assume $\mathcal{A}$ is a finite set.

4.2. Bayesian Inference

Before designing our own Bayesian inference algorithm, we ran several preliminary experiments using the existing inference algorithms popularly used by others. Our empirical studies reveal that those popular ones may be heavily biased towards overestimating the accuracy when the quality of labels is very low. For example, when there are 10 workers and $\mathbb{P}_i^t = 0.55$, the estimated label accuracy of the EM estimator (Dawid and Skene, 1979; Raykar *et al.*, 2010) stays at around 0.9 while the real accuracy is only around 0.5. This heavy bias will cause the data requester's utility u^t to be miscalculated and thus may potentially mislead our reinforcement learning algorithm. To reduce the inference bias, we develop our Bayesian inference algorithm by introducing the soft Dirichlet priors to both the true labels and workers' PoBCs. However, after doing so, the posterior distribution cannot be expressed as any known distributions, which motivates us to derive the explicit posterior distribution first and then employ Gibbs sampling to conduct inference.

For the simplicity of notations, we omit the superscript t in this subsection. It is not had to figure out the joint distribution of the collected labels $\boldsymbol{L}$ and the true labels $\mathcal{L}$

$$\mathbb{P}(\boldsymbol{L}, \mathcal{L}|\boldsymbol{\theta}, \boldsymbol{\tau}) = \prod_{j=1}^{M}\prod_{k=1}^{2}\left\{\tau_k \prod_{i=1}^{N}\mathbb{P}_i^{\delta_{ijk}}(1-\mathbb{P}_i)^{\delta_{ij(3-k)}}\right\}^{\xi_{jk}}$$

where $\boldsymbol{\theta} = [\mathbb{P}_1, \ldots, \mathbb{P}_N]$ and $\boldsymbol{\tau} = [\tau_1, \tau_2]$. τ_1 and τ_2 denote the distribution of true label 1 and 2, respectively. Besides, $\delta_{ijk} = \mathbb{1}(L_i(j) = k)$ and $\xi_{jk} = \mathbb{1}(\mathcal{L}(j) = k)$. Here, we assume Dirichlet priors $\mathrm{Dir}(\cdot)$ for $\mathbb{P}_i$ and $\boldsymbol{\tau}$ as

$$[\mathbb{P}_i, 1-\mathbb{P}_i] \sim \mathrm{Dir}(\alpha_1, \alpha_2)\ ,\ \boldsymbol{\tau} \sim \mathrm{Dir}(\beta_1, \beta_2).$$

Then, the joint distribution of $\boldsymbol{L}$ and $\mathcal{L}$ satisfies

$$\mathbb{P}(\boldsymbol{L}, \mathcal{L}) = \int_{\boldsymbol{\theta},\boldsymbol{\tau}} \mathbb{P}(\mathcal{L}, \boldsymbol{L}|\boldsymbol{\theta}, \boldsymbol{\tau}) \cdot \mathbb{P}(\boldsymbol{\theta}, \boldsymbol{\tau}) \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\tau}.$$

Following Bayes' theorem, we can derive that

$$\mathbb{P}(\mathcal{L}|\boldsymbol{L}) = \mathbb{P}(\boldsymbol{L}, \mathcal{L})/\mathbb{P}(\boldsymbol{L}) \propto B(\hat{\boldsymbol{\beta}})\prod_{i=1}^{N} B(\hat{\boldsymbol{\alpha}}_i). \quad (5)$$

where $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1, \hat{\alpha}_2]$, $\hat{\boldsymbol{\beta}} = [\hat{\beta}_1, \hat{\beta}_2]$ and

$$\hat{\alpha}_{i1} = \sum_{j=1}^{M}\sum_{k=1}^{K}\delta_{ijk}\xi_{jk} + 2\alpha_1 - 1$$
$$\hat{\alpha}_{i2} = \sum_{j=1}^{M}\sum_{k=1}^{K}\delta_{ij(3-k)}\xi_{jk} + 2\alpha_2 - 1$$
$$\hat{\beta}_k = \sum_{j=1}^{M}\xi_{jk} + 2\beta_k - 1.$$

Algorithm 1 Gibbs sampling for crowdsourcing

1: **Input:** the collected labels $\boldsymbol{L}$, the number of samples W
2: **Output:** the sample sequence $\mathcal{S}$
3: $\mathcal{S} \leftarrow \emptyset$, Initialize $\tilde{\mathcal{L}}$ with the uniform distribution
4: **for** $s = 1$ **to** W **do**
5: **for** $j = 1$ **to** M **do**
6: Compute $\mathbb{P}[\mathcal{L}(j) = k]$ by letting $\mathcal{L}(-j) = \tilde{\mathcal{L}}(-j)$.
7: $\tilde{\mathcal{L}}(j) \leftarrow$ Sample $\{1, 2\}$ with $\mathbb{P}[\mathcal{L}(j) = k]$
8: **end for**
9: Append $\tilde{\mathcal{L}}$ to the sample sequence $\mathcal{S}$
10: **end for**

Besides, $B(x, y) = (x-1)!(y-1)!/(x+y-1)!$ denotes the beta function. The convergence of our inference algorithm requires $\alpha_1 > \alpha_2$. To simplify the theoretical analysis, we set $\alpha_1 = 1.5$ and $\alpha_2 = 1$ in this paper.

Based on the joint posterior distribution $\mathbb{P}(\mathcal{L}|\boldsymbol{L})$, we cannot derive an explicit formulation for the true label distribution of task j. Hence, we resort to Gibbs sampling for the inference. More specifically, according to Bayes' theorem, we can know the conditional distribution of the true label of task j satisfies $\mathbb{P}[\mathcal{L}(j)|\boldsymbol{L}, \mathcal{L}(-j)] \propto \mathbb{P}(\mathcal{L}|\boldsymbol{L})$, where $-j$ denotes all the tasks except for task j. In this case, we can generate the samples of the true label vector $\mathcal{L}$ by using Algorithm 1. At each step of sampling (line 6-7), Algorithm 1 calculates $\mathbb{P}[\mathcal{L}(j)|\boldsymbol{L}, \mathcal{L}(-j)]$ at first and then generates a new sample of $\mathcal{L}(j)$ to replace the old one in $\tilde{\mathcal{L}}$. Through traversing all tasks, Algorithm 1 generates a new sample of the true label vector $\mathcal{L}$. Repeating this process for W times, we can get the required samples of $\mathcal{L}$, which is recorded in $\mathcal{S}$. Here, we write the s-th sample as $\tilde{\mathcal{L}}^{(s)}$. Since Gibbs sampling requires a burn-in process, we need to discard the first W_0 samples in $\mathcal{S}$. Thus, we estimate worker i's PoBC $\mathbb{P}_i$ as

$$\tilde{\mathbb{P}}_i = \frac{\sum_{s=W_0+1}^{W}\left[2\alpha_1 - 1 + \sum_{j=1}^{M}\mathbb{1}(\tilde{\mathcal{L}}^{(s)}(j) = L_i(j))\right]}{(W - W_0)\cdot(2\alpha_1 + 2\alpha_2 - 2 + M)} \quad (6)$$

and the distribution of true labels $\boldsymbol{\tau}$ as

$$\tilde{\tau}_k = \frac{\sum_{s=W_0+1}^{W}\left[2\beta_1 - 1 + \sum_{j=1}^{M}\mathbb{1}(\tilde{\mathcal{L}}^{(s)}(j) = k)\right]}{(W - W_0)\cdot(2\beta_1 + 2\beta_2 - 2 + M)}. \quad (7)$$

Furthermore, we define the log-ratio of task j as

$$\tilde{\sigma}_j = \log\frac{\mathbb{P}[\mathcal{L}(j) = 1]}{\mathbb{P}[\mathcal{L}(j) = 2]} = \log\left(\frac{\tilde{\tau}_1}{\tilde{\tau}_2}\prod_{i=1}^{N}\tilde{\lambda}_i^{\delta_{ij1} - \delta_{ij2}}\right) \quad (8)$$

where $\tilde{\lambda}_i = \tilde{\mathbb{P}}_i/(1 - \tilde{\mathbb{P}}_i)$. Then, we decide the true label estimate $\tilde{\mathcal{L}}(j)$ as 1 if $\tilde{\sigma}_j > 0$ and as 2 if $\tilde{\sigma}_j < 0$. Correspondingly, the label accuracy A can be estimated as

$$\tilde{A} = \mathbb{E}(A) = \frac{1}{M}\sum_{j=1}^{M} e^{|\tilde{\sigma}_j|}\left(1 + e^{|\tilde{\sigma}_j|}\right)^{-1}. \quad (9)$$

Note that, both W and W_0 should be large values, and in this paper, we set $W = 1000$ and $W_0 = 100$.

4.3. Reinforcement Incentive Adjustment

In this subsection, we formally introduce our reinforcement learning (RL) algorithm, which adjusts the incentive scaling level at each time step t. Stepping back and viewing it under the large picture, the reinforcement learning serves as the glue to connect each other component in our framework.

In an RL problem, an agent interacts with an unknown environment and attempts to maximize its utility (Sutton and Barto, 1998; Szepesvári, 2010). The environment is commonly formalized as a Markov Decision Process (MDP) defined as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$. At time t the agent is in state $s_t \in \mathcal{S}$ where it takes an action $a_t \in \mathcal{A}$ leading to the next state $s_{t+1} \in \mathcal{S}$ according to the transition probability kernel $\mathcal{P}$, which encodes $Pr(s_{t+1} \mid s_t, a_t)$. In most RL problems, P is unknown to the agent. The agent's goal is to learn the optimal policy, a conditional distribution $\pi(a \mid s)$ that maximizes the sate value function

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k} \mid s_t = s\right].$$

The crowdsourcing problem we aim to tackle in this paper can perfectly fit into this RL formalization. To be more specific, the data requester is the agent and it interacts with workers (i.e. the environment); incentive scaling levels are actions; the implicit utility after paying workers (see Formula 2) is reward; how workers respond to different incentives and potentially change their effort levels and reporting strategies thereafter forms the transition kernel, which is unknown to the data requester; which incentive scaling level to be picked at each time t given workers' labelling constructs the policy, which needs to be learned by the data requester.

Besides presented as the overall layout of our mechanism, Figure 1 also visualizes how our RL algorithm interacts with the environment and the rest of the framework; it takes as input workers' PoBC, reward signal, and internally its action history, and outputs the current incentive scaling level. The latest determined incentive scaling level gets plugged back into the payment rule, and by following formula (4), the exact payment to each worker is decided.

As a critical step towards improving a given, it is a standard practice for RL algorithms to learn a state-action value function (i.e. Q-function), denoted:

$$Q^{\pi}(s, a) = \mathbb{E}\left[\mathcal{R}(s_t, a_t, s_{t+1}) + \gamma V^{\pi}(s_{t+1}) \mid s_t = s, a_t = a\right]$$

In real-world problems, in order to achieve a better generalization, instead of learning a value for each state-action pair, it is more common to learn an approximate value function: $Q^{\pi}(s, a; \theta) \approx Q^{\pi}(s, a)$. A standard approach is to learn a feature-based state representation $\phi(s)$ of the state s. Due to the popularity of Deep Reinforcement learning, it has been a trend to deploy neural networks to automatically extract high-level features (Silver *et al.*, 2017; Mnih *et al.*, 2015). However, most deep RL methods' success is only demonstrated in the domains where the environment is very high-dimensional(). Unfortunately, this prerequisite does not hold in most crowdsourcing problems, where the number of workers are limited to be fewer than thousands. Due to this fact, we turn our attention to designing a high-quality human-engineered feature representation, which embodies our knowledge of this domain. Several studies also reveal that a carefully designed static feature representation can achieve performance as good as the most sophisticated state-of-the-art deep RL models, even in those most challenging problems (Liang *et al.*, 2016).

Recall that the data requester's implicit utility at each time t only depends on the aggregate probability of being correct averaged across the whole worker body. Such observation already points out to a representation design which guarantees generalization. To be more specific, we design our state representations as

$$\phi(s) = \frac{1}{N} \cdot \sum_{k=1}^{2} \mathbb{P}(L = k) \cdot \sum_{i=1}^{N} c_{ikk}.$$

Further recall, when deciding the scaling level a_t the data requester does not observe the latest labelling and thus cannot estimate the current $\phi(s_t)$ via Bayesian inference. Due to this one-step delay, we have to build our state representation using the previous observation. Since most workers would only change their effort levels and reporting strategies after receiving a new incentive, there exists some imperfect mapping function $\phi(s_t) \approx f(\phi(s_{t-1}), a_{t-1})$. Putting into another perspective, the combination of $\phi(s_{t-1})$ and a_{t-1} also reflects our best knowledge of the current state. Utilizing this implicit function, we formally introduce an augmented state representation for our RL algorithm

$$\hat{s_t} = \langle \phi(s_{t-1}), a_{t-1} \rangle.$$

Since the data requester never possesses the ground truth for each task, the utility u_t is not accurately observed. Also $\hat{s_t}$ is not accurate, as the mapping function can not be perfect. Combining both together, it would not be a surprise that some noise that cannot be directly learned exists in our state-action value function. As section 3 mentions, according to the central limit theorem, these noise can be modeled using Gaussian process. To be more specific, we calculate our temporal difference (TD) as

$$r_t \approx Q^{\pi}(\hat{s_t}, a_t) - \gamma V^{\pi}(\hat{s}_{t+1}) + \epsilon_t$$

where the noise ϵ follows a Gaussian process $\mathcal{N}(\hat{s_t}, \hat{s_{t+1}})$. Note we gain two benefits doing so. First, it greatly sim-

plifies our derivation of the update for the Q-function. Secondly, our empirical results later show that this Gaussian approximation has achieved robust performance under different worker models.

Under the Gaussian process approximation, we can put all the observed rewards and the corresponding Q-function up to the current step t together and obtain

$$\boldsymbol{r} = \boldsymbol{H}\boldsymbol{Q} + \boldsymbol{N} \tag{10}$$

where $\boldsymbol{r}$, $\boldsymbol{Q}$ and $\boldsymbol{N}$ denote the collection of rewards, Q values, and residual values up to step t, respectively. Due to the Gaussian process assumption of the residual, $\boldsymbol{N} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}^2)$, where $\boldsymbol{\sigma}^2 = \text{diag}(\sigma^2, \ldots, \sigma^2)$. The hat matrix $\boldsymbol{H}$ satisfies that $\boldsymbol{H}(k,k) = 1$ and $\boldsymbol{H}(k,k+1) = -\gamma$ for $k = 1, \ldots, t$. Then, the Q-function can be learned effectively using the online Gaussian process regression algorithm (Engel *et al.*, 2005). Furthermore, we use the classic ϵ-greedy method to construct policy from the learned Q-function.

5. Game-Theoretic Analysis

In this section, we present the game-theoretic analysis on our incentive mechanism. Our main results are as follows:

Theorem 1 (One-Step IC)**.** *In any time step t, when $M \gg 1$ and $(2\mathbb{P}_H)^{2(N-1)} \geq M$, reporting truthfully and exerting high efforts is the payment-maximizing strategy for any worker i if the other workers all follow this strategy.*

Theorem 2 (Long-Term IC)**.** *Suppose the conditions in Proposition 1 are satisfied and the learned Q-function approaches the real $Q(s,a)$. When*

$$\eta M(N-1)\mathbb{P}_H \min_{a,b\in\mathcal{A}} |a-b| > \frac{F(1)-F(1-\psi)}{1-\gamma} \tag{11}$$

$$\psi = (\tau_1\tau_2^{-1} + \tau_1^{-1}\tau_2)[4\mathbb{P}_H(1-\mathbb{P}_H)]^{\frac{N-1}{2}} \tag{12}$$

always reporting truthfully and exerting high efforts is the payment-maximizing strategy for any worker i in the long term if the other workers all follow this strategy.

We put the detailed proof of these two theorems in the following two subsections. Here, we provide an overview about our main idea at first. To prove Theorem 1, three steps are needed, and we explain them in the reversed order:

- **Step 3**: In time step t, worker i's payment-maximizing strategy must maximize $\tilde{\mathbb{P}}_i^t$ because the other parts of our payment rule are given at the beginning. Thus, we can conclude Theorem 1 by proving $\tilde{\mathbb{P}}_i^t \approx \mathbb{P}_i^t$.
- **Step 2**: In Equation 6, if the samples are all correct, namely $\tilde{\mathcal{L}}^{(s)}(j) \equiv \mathcal{L}(j)$, $\tilde{P}_i^t \approx \mathbb{P}_i^t$ must hold. This motivates to bound $|\tilde{\mathbb{P}}_i^t - \mathbb{P}_i^t|$ by calculating the upper bound of the ratio of wrong labels in the samples.
- **Step 1**: The samples are generated base on

$$P(\mathcal{L}|\boldsymbol{L}) = B(\hat{\boldsymbol{\beta}})\prod_{i=1}^{N} B(\hat{\boldsymbol{\alpha}}_i)/[C_p \cdot P(\boldsymbol{L})] \tag{13}$$

where C_p is the nomalization constant. Both the numerator and denominator in Equation 13 are changing with $\boldsymbol{L}$, making the distribution difficult to analyze . Thus, before Step 2, we derive a proper approximation for the denominator of Equation 13 at first.

Due to the existence of Theorem 1, when proving Theorem 2, we know that, to get higher long-term payments, worker i must let our RL algorithm at least increase the scaling factor from a to any $b > a$ at a certain state s. Actually, our RL algorithm will only increase the scaling factor when the state-action value function satisfies $Q^\pi(s,a) \leq Q^\pi(s,b)$. Equation 2 tells us that our value is consist of the utility obtained from the collected labels ($F(\tilde{A}^t)$) and the utility lost in the payment ($\eta\sum_{i=1}^{N} P_i^t$). If increasing the scaling factor, we at least need to increase the payments for the other $N-1$ workers by $M(N-1)\mathbb{P}_H \min_{a,b\in\mathcal{A}} |a-b|$, which corresponds to the left-hand side of Equation 11. If worker i cannot make up our value loss increment in the payments, our RL algorithm will reject to increase the scaling factor, and the only payment-maximizing strategy for worker i is to report truthfully and exert high efforts in all time steps. Thus, our proof will focus on the right-hand side of Equation 11 which denotes the upper bound of the value increment that worker i can bring. Note that, in our proof, if we omit the superscript t in an equation, we mean that this equation holds for all time steps. Besides, we employ the convention that $\bar{\mathbb{P}} = 1-p$, $\hat{\mathbb{P}} = \max\{\mathbb{P}, \bar{\mathbb{P}}\}$ and $\mathbb{P}_0 = \tau_1$.

5.1. Proof for One-Step IC

(Step 1) Denote the labels generated by N workers for task j as vector $\boldsymbol{L}(j)$. The distribution of $\boldsymbol{L}(j)$ satisfies

$$\mathbb{P}_{\hat{\boldsymbol{\theta}}}[\boldsymbol{L}(j)] = \sum_{k=1}^{2}\tau_k \prod_{i=1}^{N} p_i^{\delta_{ijk}}(1-p_i)^{\delta_{ij(3-k)}} \tag{14}$$

where $\hat{\boldsymbol{\theta}} = [\tau_1, p_1, \ldots, p_N]$ denotes all the parameters and $\delta_{ijk} = \mathbb{1}(L_i(j) = k)$. Then, we can have

Theorem 3. *When $M \to \infty$,*

$$\mathbb{P}(\boldsymbol{L}) \to C_L(M) \cdot \prod_{\boldsymbol{L}(j)} \left\{\mathbb{P}_{\hat{\boldsymbol{\theta}}}[\boldsymbol{L}(j)]\right\}^{M\cdot\mathbb{P}_{\hat{\boldsymbol{\theta}}}[\boldsymbol{L}(j)]}$$

where $C_L(M)$ denotes a constant that depends on M.

We put the proof in the supplementary file. Our main idea is to connect $P(\mathcal{L})$ with $\prod_{j=1}^{M}\mathbb{P}_{\hat{\boldsymbol{\theta}}}[\boldsymbol{L}(j)]$.

(Step 2) We introduce n and m to denote the number of tasks of which the true label sample in Equation 6 is correct ($\mathcal{L}^{(s)}(j) = \mathcal{L}(j)$) and wrong ($\mathcal{L}^{(s)}(j) \neq \mathcal{L}(j)$), respectively. Then, we can derive the upper bound of wrong labels as

Theorem 4. *When $M \gg 1$,*

$$\mathbb{E}[m/M] \lesssim (1+e^{\delta})^{-1}(\varepsilon+e^{\delta})(1+\varepsilon)^{M-1} \quad (15)$$

$$\mathbb{E}[m/M]^2 \lesssim (1+e^{\delta})^{-1}(\varepsilon^2+e^{\delta})(1+\varepsilon)^{M-2} \quad (16)$$

where $\varepsilon^{-1} = \prod_{i=0}^{N}(2\hat{p}_i)^2$, $\delta = O[\Delta \cdot \log(M)]$ *and*

$$\Delta = \sum\nolimits_{i=1}^{N}[1(p_i < 0.5) - 1(p_i > 0.5)].$$

We also put the detailed proof in the supplementary file. Our main idea is to introduce a set of variables to describe the collected labels at first. Among the n tasks of which the posterior true label is correct, x_i and y_i denote the number of tasks of which worker i's label is correct and wrong, respectively. Among the remaining m tasks, w_i and z_i denote the number of tasks of which worker i's label is correct and wrong, respectively. Then, we calculate the approximation of $P(m)$ based on the conditional probabilities $P(x_i, y_i, w_i, z_i|m)$ and $P(\mathcal{L}|\boldsymbol{L})$. The upper bounds of $\mathbb{E}[m/M]$ and $\mathbb{E}[m/M]^2$ can be obtained by calculating the upper bounds of $\sum_m mP(m)$ and $\sum_m m^2P(m)$.

(Step 3) When $M \gg 1$, in Equation 6, $\tilde{\mathbb{P}}_i \approx \mathbb{E}_{\mathcal{L}}(x_i+z_i)/M$, where $\mathbb{E}_{\mathcal{L}}$ denotes the expectation based on the distribution $P(\mathcal{L}|\boldsymbol{L})$. Meanwhile, according to the law of large numbers, $\mathbb{P}_i \approx (x_i+w_i)/M$. Thus, we can have

$$|\tilde{\mathbb{P}}_i - \mathbb{P}_i| \approx \mathbb{E}_{\mathcal{L}}|w_i - z_i|/M \leq \mathbb{E}_{\mathcal{L}}[m/M]. \quad (17)$$

If workers except for worker i all report truthfully and exert high efforts, then $\Delta \leq -1$ in Theorem 4 because we require $N \geq 3$ in Section 3. Thus, $e^{\delta} \approx 0$. Considering $2\tilde{\mathbb{P}}_i \geq 1$,we can have $\varepsilon^{-1} \geq (2\mathbb{P}_H)^{2(N-1)}$. Hence, $\varepsilon \leq M^{-1}$ when $(2\mathbb{P}_H)^{2(N-1)} \geq M$. Taking the above analysis into consideration, Equations 15 and 16 can be calculated as

$$\mathbb{E}\left[\frac{m}{M}\right] \lesssim \frac{C_1}{M \cdot C_2} \ , \ \mathbb{E}\left[\frac{m}{M}\right]^2 \lesssim \frac{C_1}{M^2 \cdot C_2^2} \quad (18)$$

where $C_1 = (1+M^{-1})^M \approx e$ and $C_2 = 1 + M^{-1} \approx 1$. Then, $m/M \approx 0$ because $\mathbb{E}[m/M] \approx 0$ and $\mathrm{Var}[m/M] = \mathbb{E}[m/M]^2 - (\mathbb{E}[m/M])^2 \approx 0$. In this case, $\tilde{\mathbb{P}}_i \approx \mathbb{P}_i$ and Thereby, worker i can only get the maximal payment when $\mathbb{P}_i = \mathbb{P}_H$, which concludes Proposition 1.

5.2. Proof for Long-Term IC

Following our analysis in the beginning of this section, to prove Theorem 2, we need to derive the upper bound of the value increment that worker i can bring. Worker i can only increase our value by increasing the estimated label accuracy $\tilde{A}$. From Equation 9, we can know that, when $M \gg 1$, the estimated accuracy $\tilde{A}$ satisfies

$$\tilde{A} \approx 1 - \mathbb{E}g(\tilde{\sigma}_j) \ , \ g(\tilde{\sigma}_j) = 1/(1+e^{|\tilde{\sigma}_j|}). \quad (19)$$

From the proof of Proposition 1, we can know that $\tilde{p}_i^t \approx p_i^t$. In this case, according to Equation 8, we can have

$$\tilde{\sigma}_j(\mathbb{P}_i) \approx \log\left(\frac{\tau_1}{\tau_2}\lambda_i^{\delta_{ij1}-\delta_{ij2}}\prod\nolimits_{k\neq i}\lambda_H^{\delta_{kj1}-\delta_{kj2}}\right). \quad (20)$$

where $\lambda_i = \mathbb{P}_i/(1-\mathbb{P}_i)$ and $\lambda_H = \mathbb{P}_H/(1-\mathbb{P}_H)$.

We know that $\tilde{A} \leq 1.0$ holds no matter what strategy worker i takes. Thus, to bound the value increment caused by worker i's strategy changes, we consider two cases where worker i intentionally provide low-quality labels:

- If $\mathbb{P}_i = 0.5$, we can eliminate λ_i from Equation 20 because $\lambda_i = 1$. Furthermore, according to Lemma 11 in the supplementary file, we can know that $g(\tilde{\sigma}_j) < e^{\tilde{\sigma}_j}$ and $g(\tilde{\sigma}_j) < e^{-\tilde{\sigma}_j}$ both hold.Thus, we build a more tight upper bound of $g(\tilde{\sigma}_j)$ by dividing all the combinations of δ_{kj1} and δ_{kj2} in Equation 20 into two sets and using the smaller one of $e^{\tilde{\sigma}_j}$ and $e^{-\tilde{\sigma}_j}$ in each set. By using this method, if the true label is 1, we can have $\mathbb{E}_{[L(j)=1]}g(\tilde{\sigma}_j) < q_1 + q_2$, where

  $$q_1 = \frac{\tau_2}{\tau_1}\sum\nolimits_{n=K+1}^{N-1} C_{N-1}^{n}(\frac{1}{\lambda_H})^{n-m}p_H^n(1-p_H)^m$$
  $$q_2 = \frac{\tau_1}{\tau_2}\sum\nolimits_{n=0}^{K} C_{N-1}^{n}\lambda_H{}^{n-m}p_H^n(1-p_H)^m$$
  $$n = \sum\nolimits_{k\neq i}\delta_{kj1} \ , \ m = \sum\nolimits_{k\neq i}\delta_{kj2}$$

  and $K = \lfloor (N-1)/2 \rfloor$. By using Lemma 12 in the supplementary file, we can thus get

  $$\mathbb{E}_{[L(j)=1]}g(\tilde{\sigma}_j) < c_\tau[4p_H(1-p_H)]^{\frac{N-1}{2}}.$$

  where $c_\tau = \tau_1\tau_2^{-1} + \tau_1^{-1}\tau_2$. Similarly,

  $$\mathbb{E}_{[L(j)=2]}g(\tilde{\sigma}_j) < c_\tau[4p_H(1-p_H)]^{\frac{N-1}{2}}.$$

  Thereby, $\tilde{A} > 1 - c_\tau[4p_H(1-p_H)]^{\frac{N-1}{2}} = 1-\psi$.

- If $\mathbb{P} = 1 - \mathbb{P}_H$, we can rewrite Equation 20 as

  $$\tilde{\sigma}_j(1-\mathbb{P}_H) \approx \log\left(\frac{\tau_1}{\tau_2}\lambda_H^{x-y}\prod\nolimits_{k\neq i}\lambda_H^{\delta_{kj1}-\delta_{kj2}}\right)$$

  where $x = \delta_{ij2}$ and $y = \delta_{ij1}$. Since $\mathbb{P}_i = 1 - \mathbb{P}_H$, x and y actually has the same distribution as δ_{kj1} and δ_{kj2}. Thus, the distribution of $\tilde{\sigma}_j(1-p_H)$ is actually the same as $\tilde{\sigma}_j(\mathbb{P}_H)$. In other words, since Proposition 1 ensures p_i to be accurately estimated, our Bayesian inference algorithm uses the information provided by worker i via flipping the label when $\mathbb{P}_i < 0.5$.

Thus, even if worker i intentionally lower the label quality, $\tilde{A} \geq 1-\psi$ still holds. Considering $F(\cdot)$ is a non-decreasing monotonic function, we can know that worker i can at most

increase our value by $F(1) - F(1-\psi)$ in each step and $(1-\gamma)^{-1}[F(1) - F(1-\psi)]$ in the long term. Taking our analysis about the payment increment in the beginning of this section, we find that worker i cannot make up our value loss increment in the payments if Equation 11 is satisfied. In this case, our RL algorithm will never response to the strategy changes of worker i, which concludes Theorem 2.

6. Empirical Experiments

In this section, we firstly test the one-step performance of our incentive mechanism by comparing it with the state-of-the-art incentive mechanism. Then, we show the advantages of including the reinforcement algorithm via conducting experiments on three representative worker models, including fully rational, bounded rational and self-learning agents.

6.1. One-Step Peroformance Analysis

In Figures 2a-c, we compare the average payments per task for worker 1 in our incentive mechanism with DG13, the state-of-the-art peer prediction mechanism for binary labels (Dasgupta and Ghosh, 2013; Liu and Chen, 2017b). In all these experiments, we fix the scaling factor $a^t = 1$ and set $M = 100$, $N = 10$, $p_H = 0.8$ and $b = 0$. The labels are generated by simulating workers' observation process. We firstly generate the true label for task j based on the true label distriubtion (τ_1, τ_2). Then, we generate worker i's label for task j based on worker i's PoBC p_i and the true label $L(j)$. For each point in these figures, we run the experiments for 1000 rounds and present the means.

In Figure 2a and b, we show the variation of the payment for worker 1 with the distribution of true labels and the strategies of other workers, respectively. More specifically, in Figure 2a, we let all the other workers report truthfully and exert high efforts ($p_{i\neq 1} = p_H$), and meanwhile increase τ_1 from 0.05 to 0.95. In Figure 2b, we let $\tau_1 = 0.5$, and increase $p_{i\neq 1}$ from 0.6 to 0.95. From these two figures, we can find that the payment for worker 1 in our mechanism almost only depend on worker 1's own strategy. By contrast, the payments in DG13 is severely affected by the distribution of true labels and the strategies of other workers. Furthermore, in Figure 2c, we present the standard variance of the payment for worker 1. We let $\tau_1 = 0.5$, $p_{i\neq 1} = p_H$ and meanwhile increase p_1 from 0.6 to 0.95. Form the figure, we can find that the payment vairance of our mechanism is much smaller than that of DG13. All in all, our mechanism is much fairer and more stable than DG13. This is because we can fully exploit the information provided by all workers while traditional peer prediciton mechanisms only compare the labels of two workers.

In Figure 2d, we compare our Bayesian inference algorithm with two popular inference algorithms in the studies of crowdsourcing, that is, the Dawid-Skene estimator (Dawid and Skene, 1979; Raykar *et al.*, 2010) and the variational inference estimator (Liu *et al.*, 2012; Chen *et al.*, 2015). Here, we set workers' PoBC p_i to be equal and increase the value of p_i from 0.5 to 0.9, which means the quality of labels is gradually improved. The other settings are the same as Figure 2b. From the figure, we can find that, when the quality of labels is very low, the inference bias of the Dawid-Skene and variational inference estimators on the label accuracy can be larger than 0.3 while the range of the label accuracy is only $[0.5, 1.0]$. This observation shows that these two estimators become over-optimistic for low-quality labels, which will be disastrous for our reinforcement algorithm. Thus, we develop a novel Bayesian inference algorithm which reduces the inference bias for low-quality labels by considering the connection between tasks.

References

Erick Chastain, Adi Livnat, Christos Papadimitriou, and Umesh Vazirani. Algorithms, games, and evolution. *PNAS*, 111(29):10620–10623, 2014.

Xi Chen, Qihang Lin, and Dengyong Zhou. Statistical decision making for optimal budget allocation in crowd labeling. *Journal of Machine Learning Research*, 16:1–46, 2015.

Anirban Dasgupta and Arpita Ghosh. Crowdsourced judgement elicitation with endogenous proficiency. In *Proc. of WWW*, 2013.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

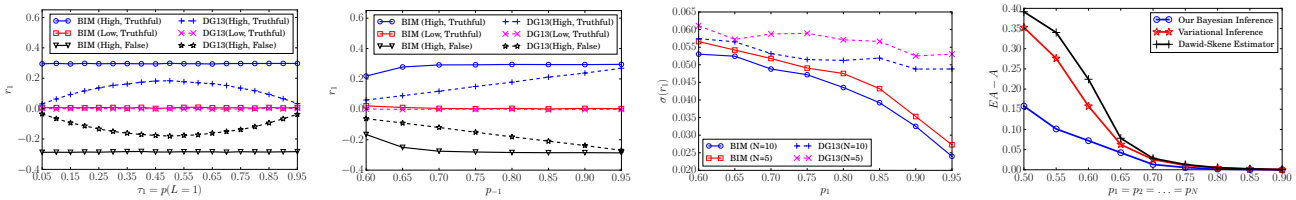
Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G Ipeirotis, and Philippe Cudré-Mauroux. The dynamics of micro-task crowdsourcing: The case of amazon mturk. In *Proc. of WWW*, 2015.

Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. In *Proc. of ICML*, 2005.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Radu Jurca, Boi Faltings, et al. Mechanisms for making crowds truthful. *Journal of Artificial Intelligence Research*, 34(1):209, 2009.

Yitao Liang, Marlos C. Machado, Erik Talvitie, and Michael Bowling. State of the art control of atari games using shallow reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS' 16, pages 485–493, 2016.



(a) (b) (c) (d)

Figure 2. One-step performance of our reinforcement incentive mechanism (RIM) (a) payment variation with the distribution of true labels (c) payment variation as the PoBC of other workers (d) the standard variance of the payment (d) the inference bias on label accuracy

Yang Liu and Yiling Chen. Machine-learning aided peer prediction. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 63–80. ACM, 2017.

Yang Liu and Yiling Chen. Sequential peer prediction: Learning to elicit effort using posted prices. In *AAAI*, pages 607–613, 2017.

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

Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level Control through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 02 2015.

Dražen Prelec. A bayesian truth serum for subjective data. *science*, 306(5695):462–466, 2004.

Goran Radanovic and Boi Faltings. A robust bayesian truth serum for non-binary signals. In *Proc. of AAAI*, 2013.

Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354 EP –, 10 2017.

Herbert Alexander Simon. *Models of bounded rationality: Empirically grounded economic reason*, volume 3. MIT press, 1982.

Edwin D Simpson, Matteo Venanzi, Steven Reece, Pushmeet Kohli, John Guiver, Stephen J Roberts, and Nicholas R Jennings. Language understanding in the wild: Combining crowdsourcing and machine learning. In *Proc. of WWW*, 2015.

Aleksandrs Slivkins and Jennifer Wortman Vaughan. Online decision making in crowdsourcing markets: Theoretical challenges. *ACM SIGecom Exchanges*, 12(2):4–23, 2014.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2010.

Jens Witkowski and David C Parkes. Peer prediction without a common prior. In *Proc. of ACM EC*, 2012.

Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: is the problem solved? *Proc. of the VLDB Endowment*, 10(5):541–552, 2017.