Attack under Disguise: An Intelligent Data Poisoning Attack Mechanism in Crowdsourcing

Chenglin Miao
State University of New York
at Buffalo, NY, USA
cmiao@buffalo.edu

Qi Li
University of Illinois at
Urbana-Champaign, IL, USA
qili5@illinois.edu

Lu Su*
State University of New York
at Buffalo, NY, USA
lusu@buffalo.edu

Mengdi Huai
State University of New York
at Buffalo, NY, USA
mengdihu@buffalo.edu

Wenjun Jiang
State University of New York
at Buffalo, NY, USA
wenjunji@buffalo.edu

Jing Gao
State University of New York
at Buffalo, NY, USA
jing@buffalo.edu

ABSTRACT

As an effective way to solicit useful information from the crowd, crowdsourcing has emerged as a popular paradigm to solve challenging tasks. However, the data provided by the participating workers are not always trustworthy. In real world, there may exist malicious workers in crowdsourcing systems who conduct the data poisoning attacks for the purpose of sabotage or financial rewards. Although data aggregation methods such as majority voting are conducted on workers' labels in order to improve data quality, they are vulnerable to such attacks as they treat all the workers equally. In order to capture the variety in the reliability of workers, the Dawid-Skene model, a sophisticated data aggregation method, has been widely adopted in practice. By conducting maximum likelihood estimation (MLE) using the expectation maximization (EM) algorithm, the Dawid-Skene model can jointly estimate each worker's reliability and conduct weighted aggregation, and thus can tolerate the data poisoning attacks to some degree. However, the Dawid-Skene model still has weakness. In this paper, we study the data poisoning attacks against such crowdsourcing systems with the Dawid-Skene model empowered. We design an intelligent attack mechanism, based on which the attacker can not only achieve maximum attack utility but also disguise the attacking behaviors. Extensive experiments based on real-world crowdsourcing datasets are conducted to verify the desirable properties of the proposed mechanism.

CCS CONCEPTS

• **Security and privacy → Systems security**; • **Human-centered computing → Collaborative and social computing**;

KEYWORDS

Crowdsourcing, data poisoning, expectation maximization

*L. Su is the corresponding author.

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186032

ACM Reference Format:
Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. 2018. Attack under Disguise: An Intelligent Data Poisoning Attack Mechanism in Crowdsourcing. In *The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186032

1 INTRODUCTION

With the proliferation of online crowdsourcing services such as Amazon Mechanical Turk (AMT)[1] and CrowdFlower[2], crowdsourcing has emerged as a popular, fast and cheap problem-solving paradigm for various data analysis tasks, such as image annotation [52], entity resolution [47] and sentiment analysis [33]. Through the power of the crowd, the data requesters can obtain large amounts of data at extremely low cost. In a typical crowdsourcing system, the data requester posts the tasks that require human intelligence onto a crowdsourcing service platform, and then a large crowd of people (usually referred to as *workers*) participate in the tasks that they are interested in. In crowdsourcing systems, to reduce the errors made by individual workers, a common practice is to query an item (e.g., a picture or a question) to multiple workers and then aggregate their labels on the items.

Although crowdsourcing brings substantial advantages, the openness of the crowdsourcing systems and the potential value of the collected data offer both opportunities and incentives for malicious parties to launch attacks. In this paper, we investigate crowdsourcing in adversarial environments and study an important attack form, called *data poisoning*. In this attack, the attacker aims to maximize the error of the final results and render the crowdsourcing results useless through creating or recruiting a group of malicious workers and letting them provide manipulated data. This attack goal can be easily achieved if the attacker has the capability of creating or recruiting an overwhelming number of malicious workers. However, in practice, the attacker usually has limited resources and he can only control a few malicious workers. In such cases, the attack strategy plays an important role.

A naive attack strategy is to let the malicious workers always disagree with the normal workers. If some straightforward aggregation methods, such as majority voting, are used to aggregate the data, this naive attack model may be the optimal choice, since every malicious worker exerts the most influence in the aggregation.

[1] https://www.mturk.com
[2] https://www.crowdflower.com

However, the story would become much more complicated when some more sophisticated aggregation methods that can capture the reliability (i.e., data quality) of each worker are employed. A representative method in this category is the *Dawid-Skene* model [9], which has been widely adopted in crowdsourcing to aggregate the conflicting data from different workers. In the Dawid-Skene model, each worker is associated with an underlying confusion matrix, which can reflect the reliability degree of this worker. After the labels are collected from the workers, the final results and the workers' confusion matrices are jointly estimated based on the maximum likelihood principle. As a result, workers with low reliability degrees will have low impact in the aggregation. In this case, if an attacker adopts the aforementioned naive attack, in which the malicious workers always disagree with the normal workers, the malicious workers are very likely to be assigned a significantly low reliability degree by the Dawid-Skene model, and thus will not be able to make any difference in the final aggregated results.

To attack a crowdsourcing system with the Dawid-Skene model empowered, we propose an intelligent data poisoning attack mechanism that takes into account the malicious workers' reliability degrees. In this mechanism, the malicious workers behave more "intelligently", i.e., try to improve their reliability degrees by agreeing with the normal workers on some items whose values are unlikely to be overturned. Compared with the aforementioned naive strategy, the proposed intelligent attack model can not only disguise the malicious workers, but also enable them to launch more effective attacks on the items that are more vulnerable to attack.

Towards this end, we formulate a bi-level optimization problem. The objective in the optimization problem is to maximize the attacker's utility, which is the combination of the number of the successfully attacked items and the malicious workers' reliability degrees. Since the number of the successfully attacked items is discrete, it is hard to directly solve the optimization problem. To address this challenge, a continuous and differentiable sigmoid function is adopted to approximate the discrete component in the objective function. We solve the bi-level optimization problem by iteratively solving the upper-level and lower-level subproblems, which are solved by the projected gradient ascent and expectation-maximization (EM) methods, respectively.

In summary, the main contributions of this paper are:

- We identify the pitfalls and challenges in attacking a crowdsourcing system empowered with the Dawid-Skene model, which is able to incorporate the workers' reliability degrees into the aggregation procedure and thus can tolerate the naive malicious attacks.
- We design an intelligent data poisoning attack mechanism, based on which the attacker can achieve the optimal attack goal by intelligently disguising the malicious workers' behaviors.
- Extensive experiments based on real-world crowdsourcing datasets are conducted to verify the advantages of the proposed mechanism.

2 PROBLEM SETTING

In this paper, we consider a crowdsourcing scenario in which a cloud server and some participating workers are involved. The cloud server is a platform which can outsource the crowdsourcing tasks to the participating workers. The crowdsourcing task is to collect labels for a pool of items, each of which belongs to one of two possible categories (e.g., duchenne smile/non-duchenne smile; same/different; positive/negative; etc.). To ensure the quality of the final result, each item will be queried to multiple participating workers, who are the individuals that carry out the crowdsourcing tasks, and each worker will provide labels for a number of items. After collecting the labels from the participating workers, the cloud server aggregates these labels to derive the true label of each item.

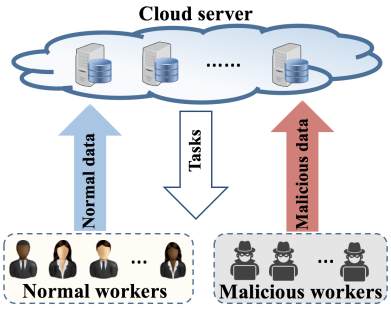


Figure 1: Crowdsourcing framework with malicious workers.

The security threats considered in this paper mainly come from an attacker who aims to attack the crowdsourcing system for malicious purposes. *The goal of the attacker is to maximize the error of the derived true labels, and meanwhile disguise his malicious behaviors so that the attack cannot be detected easily.* We assume that the attacker can recruit or create multiple participating workers (called *malicious workers*) and arbitrarily manipulate their labels, but he cannot influence the behaviors of the *normal workers* who carry out the crowdsourcing tasks without any malicious purpose. If there is no limitation on the ability of the attacker, he can achieve the attack goal easily through creating a large number of malicious workers. However, in practice, the attacker usually has limited resources and can only recruit or create a few malicious workers. In such cases, it is essential for the attacker to design a sophisticated *attack strategy* (i.e., the labels provided by the malicious workers) such that the attack goal can be maximally achieved. In order to assess the vulnerability of the crowdsourcing system in the worst case, we also assume that the attacker has full knowledge of the aggregation method and the labels from normal workers. This assumption is reasonable as it is possible for the attacker to learn the labels of normal workers through eavesdropping the communications between the cloud server and the normal workers. Figure 1 shows the crowdsourcing framework with malicious workers.

Problem formulation. Suppose the cloud server releases a crowdsourcing task which contains a set of items $O = \{o_1, o_2, ..., o_M\}$, and these items are queried to K normal workers which are represented as $U = \{u_1, u_2, ..., u_K\}$. The labels provided by these normal workers are denoted as $X = \{x_m^k\}_{m,k=1}^{M,K}$, in which x_m^k is the label provided by worker u_k for item o_m. For each item o_m, there is a true label x_m^* which is unknown *a priori* and needs to be estimated by the cloud server based on the labels collected from all the workers. We use $X^* = \{x_m^*\}_{m=1}^{M}$ to denote the set of true labels for all items. Assume that the attacker can create K' malicious workers represented as $\widetilde{U} = \{\widetilde{u}_1, \widetilde{u}_2, ..., \widetilde{u}_{K'}\}$. The set of labels provided by all the malicious workers is denoted as $\widetilde{X} = \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'}$,

and $\widetilde{x}_m^{k'}$ is the label provided by malicious worker $\widetilde{u}_{k'}$ for item o_m. *Our goal in this paper is to find an optimal attack strategy (i.e., an optimal $\widetilde{X}$) from the perspective of the attacker such that the attack goal can be maximally achieved.*

3 PRELIMINARY

As a low-cost problem-solving paradigm utilizing the wisdom of crowds, crowdsourcing has been widely adopted to collect labels for various tasks. The items are distributed to multiple participating workers, and then the labels are collected from them to estimate the true label of each item in the task. Due to the variety in the quality of the participating workers, it is a common practice to query each item to several workers and then aggregate their labels in order to get a more reliable result. A straightforward aggregation method is majority voting. However, this method cannot distinguish the reliability degrees of the workers. In order to take the workers' quality into account and obtain more accurate results, the Dawid-Skene model [9] has been widely adopted in crowdsourcing systems.

In the Dawid-Skene model, each participating worker is associated with an unknown confusion matrix which reflects the worker's ability (or reliability degree) when carrying out the crowdsourcing task. Each diagonal element in this matrix represents the probability that the worker provides the true label for a particular item, while each off-diagonal element represents the probability that a particular wrong label is provided. After the labels are collected from all the workers, the maximum likelihood estimation method is adopted to jointly estimate each item's true label and each worker's confusion matrix.

In this paper, we consider the binary case, i.e., we assume that each item has only two possible labels: 0 and 1. Based on the Dawid-Skene model, each worker u_k provides label for item o_m according to parameters $\alpha_k = \Pr(x_m^k = 1 | x_m^* = 1)$ and $\beta_k = \Pr(x_m^k = 0 | x_m^* = 0)$, where α_k and β_k are the diagonal elements in worker u_k's confusion matrix. They are also treated as u_k's ability parameters or reliability degrees. The larger α_k and β_k are, the higher the probability that worker u_k provides a true label. Additionally, this model assumes that the probability that an item drawn at random has true label 1 is p, which is usually unknown *a priori*. Denote $\Theta = \{p, \{\alpha_k, \beta_k\}_{k=1}^K\}$ as the set of all the model parameters. The Dawid-Skene model adopts the maximum likelihood estimation method to estimate Θ. However, due to the latent variables X^* are unknown *a priori*, it is hard to directly conduct the estimation.

To address the above challenge, the Dawid-Skene model adopts the EM algorithm [11] which contains an expectation step (E-step) and a maximization step (M-step). In the E-step, the items' true labels are derived based on the estimated model parameters Θ, and in the M-step, the parameters Θ are calculated based on the derived true labels. The details of the two steps are described as follows.

E-step: In this step, the model parameters Θ are fixed. For each item o_m, we calculate $\omega_m = \Pr\{x_m^* = 1 | X\}$ based on the Bayes theorem:

$$\omega_m = \Pr\{x_m^* = 1 | X; \Theta\} = \frac{\Pr\{X | x_m^* = 1\} \cdot p}{\Pr\{X | x_m^* = 1\} \cdot p + \Pr\{X | x_m^* = 0\} \cdot (1-p)} = \frac{\prod_{k \in U_m} \alpha_k^{x_m^k} (1-\alpha_k)^{1-x_m^k} \cdot p}{\prod_{k \in U_m} \alpha_k^{x_m^k} (1-\alpha_k)^{1-x_m^k} \cdot p + \prod_{k \in U_m} \beta_k^{1-x_m^k} (1-\beta_k)^{x_m^k} \cdot (1-p)}, \tag{1}$$

where U_m represents the set of normal workers who provide labels for item o_m.

Here ω_m is the posterior probability that the true label of the item o_m is 1. With the calculated $\Omega = \{\omega_m\}_{m=1}^M$, the expected value of the log likelihood function can be expressed as

$$\begin{aligned} Q(\Theta) =& E[\log L(\Theta; X, X^*)] = E[\log \prod_{m=1}^{M} L(\Theta; X_m, x_m^*)] \\ =& \sum_{m=1}^{M} \{\omega_m \log[\prod_{k \in U_m} \alpha_k^{x_m^k} (1-\alpha_k)^{1-x_m^k} \cdot p] \\ &+ (1-\omega_m) \log[\prod_{k \in U_m} \beta_k^{1-x_m^k} (1-\beta_k)^{x_m^k} \cdot (1-p)]\}, \end{aligned} \tag{2}$$

where X_m represents the set of labels for item o_m.

M-step: In this step, the posterior probabilities $\{\omega_m\}_{m=1}^M$ are fixed. The model parameters Θ are estimated by maximizing the expected value of the log likelihood function $Q(\Theta)$, and they are updated as follows:

$$p = \frac{\sum_{m=1}^M \omega_m}{M}, \tag{3}$$

$$\alpha_k = \frac{\sum_{m \in O_k} \omega_m \cdot x_m^k}{\sum_{m \in O_k} \omega_m}, \tag{4}$$

$$\beta_k = \frac{\sum_{m \in O_k} (1-\omega_m) \cdot (1-x_m^k)}{\sum_{m \in O_k} (1-\omega_m)}, \tag{5}$$

where O_k represents the set of items queried to u_k.

The above two steps are iteratively conducted until the convergence criterion is satisfied. Finally, if ω_m is larger than 0.5, the true label of the item o_m is assigned as 1, otherwise, it is assigned as 0.

4 THE INTELLIGENT ATTACK MECHANISM

In order to achieve the attack goal as much as possible, it is essential for the attacker to find an optimal attack strategy with the limited resources (i.e., the number of created or recruited malicious workers and the number of queried items). We first investigate the Dawid-Skene crowdsourcing model under the adversarial environment in section 4.1, and then discuss how to design an optimal attack strategy from the perspective of the attacker in section 4.2.

4.1 Dawid-Skene Crowdsourcing Model with Malicious Workers

In the adversarial environment, the malicious workers may blend into the crowdsourcing system and provide manipulated labels to the cloud server in order to distort the final aggregated results. In this section, we decompose the participating workers into normal and malicious ones, and investigate the relationship between the final aggregation results and the labels provided by malicious workers. Please note that the cloud server in the crowdsourcing system cannot differentiate the two types of participating workers when aggregating the collected labels.

As described in the problem setting, we assume that the attacker creates K' malicious workers to conduct the data poisoning attacks against the crowdsourcing system. The attacker cannot influence the behaviors of the normal workers, but he can arbitrarily manipulate the labels of malicious workers. We denote the ability

parameters of malicious worker $\widetilde{u}_{k'}$ in the Dawid-Skene model as $\widetilde{\alpha}_{k'} = \Pr(\widetilde{x}_m^{k'} = 1 | x_m^* = 1)$ and $\widetilde{\beta}_{k'} = \Pr(\widetilde{x}_m^{k'} = 0 | x_m^* = 0)$. We use $\widetilde{\Theta} = \{p, \{\alpha_k, \beta_k\}_{k=1}^K, \{\widetilde{\alpha}_{k'}, \widetilde{\beta}_{k'}\}_{k'=1}^{K'}\}$ to denote the set of the model parameters and $\{\alpha_k, \beta_k\}_{k=1}^K$ are the ability parameters of the normal workers. Suppose $\widehat{X}$ is the set of the labels provided by all the participating workers, including the normal and malicious ones. The E-step and M-step in the Dawid-Skene model after data poisoning attacks can be described as follows:

E-step: For each item o_m, we calculate $\widetilde{\omega}_m = \Pr\{x_m^* = 1 | \widehat{X}\}$ based on the Bayes theorem:

$$\begin{aligned}\widetilde{\omega}_m &= \Pr\{x_m^* = 1 | \widehat{X}; \widetilde{\Theta}\} \\ &= \frac{\Pr\{\widehat{X} | x_m^* = 1\} \cdot p}{\Pr\{\widehat{X} | x_m^* = 1\} \cdot p + \Pr\{\widehat{X} | x_m^* = 0\} \cdot (1-p)} \\ &= \frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1} + \widetilde{A}_{m0}},\end{aligned} \tag{6}$$

where

$$\widetilde{A}_{m1} = \prod_{k \in U_m} \alpha_k^{x_m^k} (1-\alpha_k)^{1-x_m^k} \cdot \prod_{k' \in \widetilde{U}_m} \widetilde{\alpha}_{k'}^{\widetilde{x}_m^{k'}} (1-\widetilde{\alpha}_{k'})^{1-\widetilde{x}_m^{k'}} \cdot p \tag{7}$$

$$\widetilde{A}_{m0} = \prod_{k \in U_m} \beta_k^{1-x_m^k} (1-\beta_k)^{x_m^k} \cdot \prod_{k' \in \widetilde{U}_m} \widetilde{\beta}_{k'}^{1-\widetilde{x}_m^{k'}} (1-\widetilde{\beta}_{k'})^{\widetilde{x}_m^{k'}} \cdot (1-p). \tag{8}$$

Here we use $\widetilde{U}_m$ to denote the set of malicious workers who provide labels for item o_m. $\widetilde{\omega}_m$ represents the posterior probability that the true label of item o_m is 1 after the data poisoning attacks.

M-step: In this step, we fix the the posterior probabilities $\{\widetilde{\omega}_m\}_{m=1}^M$ and update the model parameters $\widetilde{\Theta} = \{p, \{\alpha_k, \beta_k\}_{k=1}^K, \{\widetilde{\alpha}_{k'}, \widetilde{\beta}_{k'}\}_{k'=1}^{K'}\}$ as follows:

$$p = \frac{\sum_{m=1}^M \widetilde{\omega}_m}{M} \tag{9}$$

$$\alpha_k = \frac{\sum_{m \in O_k} \widetilde{\omega}_m \cdot x_m^k}{\sum_{m \in O_k} \widetilde{\omega}_m}, \quad \beta_k = \frac{\sum_{m \in O_k} (1-\widetilde{\omega}_m) \cdot (1-x_m^k)}{\sum_{m \in O_k} (1-\widetilde{\omega}_m)}, \tag{10}$$

$$\widetilde{\alpha}_{k'} = \frac{\sum_{m \in \widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m \in \widetilde{O}_{k'}} \widetilde{\omega}_m}, \quad \widetilde{\beta}_{k'} = \frac{\sum_{m \in \widetilde{O}_{k'}} (1-\widetilde{\omega}_m) \cdot (1-\widetilde{x}_m^{k'})}{\sum_{m \in \widetilde{O}_{k'}} (1-\widetilde{\omega}_m)}, \tag{11}$$

where $\widetilde{O}_{k'}$ represents the set of items queried to $\widetilde{u}_{k'}$.

The above equations show that once the labels of normal workers (i.e., X) are given, the final estimated true labels of the items and the ability parameters (i.e., $\{\alpha_k, \beta_k\}_{k=1}^K, \{\widetilde{\alpha}_{k'}, \widetilde{\beta}_{k'}\}_{k'=1}^{K'}$) of the participating workers are only dependent on the malicious workers' data. Different values of the malicious workers' labels can lead to different estimated results. Based on this fact, the attacker can conduct data poisoning attacks through carefully designing the malicious workers' labels such that the goal of the attacker can be optimally achieved.

4.2 Optimal Attack Strategy

In this paper, the attacker conducts the data poisoning attacks for the purpose of maximizing the error of the final aggregated results, and at the same time tries to disguise his attack behaviors as much as possible. We can understand the goal of the attacker in two aspects. On one hand, the attacker aims to maximize the deviation between the outputs of the Dawid-Skene crowdsourcing model before and after the data poisoning attacks. In other words, the attacker wants to maximize the number of the successfully attacked items, where we say an item is attacked successfully if the estimated true label is changed from one label to the other after taking the malicious workers' labels into account. On the other hand, the attacker wants to disguise the malicious workers as normal workers in the crowdsourcing system such that the attack behaviors cannot be detected easily. One way to achieve the disguise is to get high values on the malicious workers' ability parameters (or reliability degrees), i.e., $\{\widetilde{\alpha}_{k'}\}_{k'=1}^{K'}$ and $\{\widetilde{\beta}_{k'}\}_{k'=1}^{K'}$. Since the workers with large ability parameters will be treated as high-quality workers in the Dawid-Skene model, the crowdsourcing system then cannot distinguish the malicious workers from the normal workers. In this section, we stand on the attacker's position and discuss how to find an optimal attack strategy so that the goal of the attacker can be achieved as much as possible.

Suppose the attacker is able to create or recruit K' malicious workers, and for each malicious worker, the queried items are given. When conducting the data poisoning attacks to break the crowdsourcing system, the attacker needs to find the optimal assignments for the malicious workers' labels. An intuitive strategy is let the malicious workers provide the label which is not likely to be true for each queried item. This strategy may work well when the aggregation method is majority voting. But for the Dawid-Skene model, it is not the optimal choice, especially when only a few malicious workers are created or recruited. Due to the fact that malicious workers always disagree with the majority, the Dawid-Skene model will assign low ability values to these malicious workers, and consequently, their impact will also be decreased. In such way, the malicious workers can be detected easily and the attack may fail on all the items. Thus the ability parameters of malicious workers (i.e., $\{\widetilde{\alpha}_{k'}\}_{k'=1}^{K'}$ and $\{\widetilde{\beta}_{k'}\}_{k'=1}^{K'}$) should be taken into account when finding the optimal attack strategy.

In order to address the above challenge, we formulate the goal of the attacker as the following optimization problem:

$$\begin{aligned}\max_{\widetilde{X}} \quad & \sum_{m=1}^M \mathbb{1}(x_m^{*a} \neq x_m^{*b}) + \lambda \sum_{k'=1}^{K'} (\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'}) \\ \text{s.t.} \quad & \{X^{*a}, \widetilde{\Theta}\} = \arg\max_{X^{*a}, \widetilde{\Theta}} \log L(\widetilde{\Theta}; \widehat{X}, X^{*a}) \\ & \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in \{0,1\}\end{aligned} \tag{12}$$

where $X^{*a} = \{x_m^{*a}\}_{m=1}^M$ denotes the set of the estimated true labels after the data poisoning attacks and x_m^{*b} denotes the estimated true label for item o_m before the attacks (i.e., calculated based on the labels of normal workers). Once the normal workers' labels are given, x_m^{*b} is a constant. The objective function contains two components. The first component, i.e., $\sum_{m=1}^M \mathbb{1}(x_m^{*a} \neq x_m^{*b})$, where $\mathbb{1}(\cdot)$ is the indicator function, represents the number of the successfully attacked items. In the second component, $\sum_{k'=1}^{K'} (\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'})$ is the summation of the malicious workers' ability parameters, and λ is a parameter used to trade off the two components. The summation of the two components can also be treated as the utility of the attacker. The intuition of the objective function is to maximize the number of the successfully attacked items and the malicious workers' ability values simultaneously, where the first component is the

goal of the attack and the latter ensures that the malicious workers cannot be detected easily. In this optimization problem, the Dawid-Skene model becomes a constraint. This is a bi-level optimization problem [2]. The optimization over the labels of the malicious workers (i.e., $\widetilde{X}$) is the upper-level problem, and the optimization over $\{X^{*a}, \widetilde{\Theta}\}$ is the lower-level problem.

In the Dawid-Skene model, the final estimated true labels of the items are dependent on the posterior probabilities $\Omega = \{\omega_m\}_{m=1}^{M}$ or $\widetilde{\Omega} = \{\widetilde{\omega}_m\}_{m=1}^{M}$: if ω_m (or $\widetilde{\omega}_m$) is larger than 0.5, x_m^{*b} (or x_m^{*a}) is assigned as 1, otherwise, it is assigned as 0. Thus we can reformulate optimization problem (12) as follows:

$$\begin{aligned} \max_{\widetilde{X}} \quad & \sum_{m=1}^{M} \frac{1}{2}\{1 - \mathrm{sgn}[(\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5)]\} + \lambda \sum_{k'=1}^{K'} (\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'}) \\ \text{s.t.} \quad & \{\widetilde{\Omega}, \widetilde{\Theta}\} = \arg\max_{\widetilde{\Omega},\widetilde{\Theta}} \ \log L(\widetilde{\Theta}; \widehat{X}, \widetilde{\Omega}) \\ & \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in \{0, 1\}, \end{aligned} \tag{13}$$

where

$$\mathrm{sgn}[(\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5)] = \begin{cases} 1 & if\ (\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5) > 0 \\ 0 & if\ (\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5) = 0 \\ -1 & if\ (\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5) < 0. \end{cases} \tag{14}$$

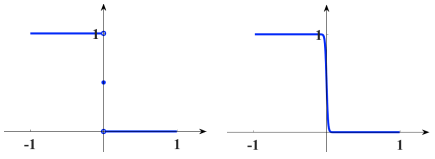
Once the labels of normal workers are given, the posterior probability ω_m for item o_m is a constant. $\widetilde{\omega}_m$, $\widetilde{\alpha}_{k'}$ and $\widetilde{\beta}_{k'}$ are dependent on the labels of the malicious workers (i.e., the attack strategy $\widetilde{X}$) and they can be different when the malicious workers vary their labels. In this way, $\widetilde{\omega}_m$, $\widetilde{\alpha}_{k'}$ and $\widetilde{\beta}_{k'}$ can be expressed as the functions of $\widetilde{X}$ according to Eqn. (6) and Eqn. (11). Then problem (13) becomes:

$$\begin{aligned} \max_{\widetilde{X}} \quad & \sum_{m=1}^{M} \frac{1}{2}\{1 - \mathrm{sgn}[(\omega_m - 0.5)\cdot(\frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1} + \widetilde{A}_{m0}} - 0.5)]\} \\ & + \lambda \sum_{k'=1}^{K'} (\frac{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m} + \frac{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)\cdot(1-\widetilde{x}_m^{k'})}{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)}) \\ \text{s.t.} \quad & \{\widetilde{\Omega}, \widetilde{\Theta}\} = \arg\max_{\widetilde{\Omega},\widetilde{\Theta}} \ \log L(\widetilde{\Theta}; \widehat{X}, \widetilde{\Omega}) \\ & \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in \{0, 1\}. \end{aligned} \tag{15}$$

Since the objective function in problem (15) is not continuous, it is hard to directly solve this optimization problem. In order to address this challenge, we approximate the objective function in problem (15) by the following one:

$$\begin{aligned} \max_{\widetilde{X}} \quad & \sum_{m=1}^{M} \{1 - \frac{1}{1 + \exp[-\theta(\omega_m - 0.5)\cdot(\frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1} + \widetilde{A}_{m0}} - 0.5)]} \\ & + \lambda \sum_{k'=1}^{K'} (\frac{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m} + \frac{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)\cdot(1-\widetilde{x}_m^{k'})}{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)}). \end{aligned} \tag{16}$$

The basic idea behind the approximation is that function $h_1(x) = \frac{1}{2}(1 - \mathrm{sgn}\, x)$ can be approximated by function $h_2(x) = 1 - \frac{1}{1+\exp(-\theta x)}$ when $x \in (-1, 1)$. The parameter θ in $h_2(x)$ represents the steepness of the curve. The curves of the two functions when $\theta = 100$ are shown in Figure 2. We can see $h_2(x)$ is a good approximation of $h_1(x)$. Additionally, the continuous property of $h_2(x)$ allows us to solve the optimization problem based on the objective function (16).



(a) $h_1(x)$ **(b)** $h_2(x)$ **with** $\theta = 100$

Figure 2: Curves of $h_1(x)$ and $h_2(x)$.

Another challenge when solving the above optimization problem is that each element in $\widetilde{X}$ has a categorical value (0 or 1). This introduces difficulties when solving the upper-level problem. In this paper, we relax the values of the elements in $\widetilde{X}$ to the range $[0, 1]$ such that the optimization problem can be solved according to the gradient-based methods. In other words, we treat $\widetilde{x}_m^{k'}$ as the probability that malicious worker $\widetilde{u}_{k'}$ provides label 1 for item o_m. Finally, the value of $\widetilde{x}_m^{k'}$ will be transformed to categorical data: if the probability is larger than 0.5, $\widetilde{x}_m^{k'}$ is assigned as 1, otherwise, it is assigned as 0. Then the attacker needs to solve the following optimization problem in order to get the optimal attack strategy:

$$\begin{aligned} \max_{\widetilde{X}} \quad & f(\widetilde{X}) = \sum_{m=1}^{M} \{1 - \frac{1}{1 + \exp[-\theta(\omega_m - 0.5)\cdot(\frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1} + \widetilde{A}_{m0}} - 0.5)]} \} \\ & + \lambda \sum_{k'=1}^{K'} (\frac{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m} + \frac{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)\cdot(1-\widetilde{x}_m^{k'})}{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)}) \\ \text{s.t.} \quad & \{\widetilde{\Omega}, \widetilde{\Theta}\} = \arg\max_{\widetilde{\Omega},\widetilde{\Theta}} \ \log L(\widetilde{\Theta}; \widehat{X}, \widetilde{\Omega}) \\ & \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in [0, 1]. \end{aligned} \tag{17}$$

Next, we discuss how to solve the above optimization problem. The solution we adopted here is a two-step iterative procedure.

Step 1: In this step, we first fix the labels of malicious workers, i.e., $\widetilde{X}$, which are estimated in the previous iteration. If it is the first iteration, the elements in $\widetilde{X}$ can be initialized randomly or be set as some particular values. Then we solve the lower-level problem through conducting the E-step and M-step described in section 4.1 to get the optimal parameters $\{\widetilde{\Omega}, \widetilde{\Theta}\}$. Please note that all the elements in $\widetilde{X}$ need to be transformed to the categorical values (i.e., 1 or 0) before solving the lower-level problem.

Step 2: In this step, we fix the parameters $\{\widetilde{\Omega}, \widetilde{\Theta}\}$ calculated in Step 1, and then adopt the projected gradient ascent method to solve the upper-level problem. More specifically, in iteration t, we update $\widetilde{x}_m^{k'}$ as follows:

$$\widetilde{x}_m^{k'(t+1)} \leftarrow \mathrm{Proj}_{[0,1]}(\widetilde{x}_m^{k'(t)} + s_t \cdot \nabla_{\widetilde{x}_m^{k'}} f(\widetilde{X})) \tag{18}$$

where s_t is the step size in iteration t and $\mathrm{Proj}_{[0,1]}(\cdot)$ is the projection operator onto the range $[0, 1]$. The gradient $\nabla_{\widetilde{x}_m^{k'}} f(\widetilde{X})$ is calculated as follows:

$$\begin{aligned} \nabla_{\widetilde{x}_m^{k'}} f(\widetilde{X}) = & -\frac{\exp(\theta d_1 d_2)}{[1 + \exp(\theta d_1 d_2)]^2} \cdot \theta d_1 \cdot \frac{\partial d_2}{\partial \widetilde{x}_m^{k'}} \\ & + \lambda(\frac{\widetilde{\omega}_m}{\sum_{\tilde{m}\in\widetilde{O}_{k'}} \widetilde{\omega}_{\tilde{m}}} + \frac{\widetilde{\omega}_m - 1}{\sum_{\tilde{m}\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_{\tilde{m}})}) \end{aligned} \tag{19}$$

where $d_1 = \omega_m - 0.5$, $d_2 = \frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1} + \widetilde{A}_{m0}} - 0.5$. Through combining with Eqn. (7) and Eqn. (8), we can calculate $\frac{\partial d_2}{\partial \widetilde{x}_m^{k'}}$ as

$$\frac{\partial d_2}{\partial \widetilde{x}_m^{k'}} = \frac{\frac{\partial \widetilde{A}_{m1}}{\partial \widetilde{x}_m^{k'}} \cdot \widetilde{A}_{m0} - \frac{\partial \widetilde{A}_{m0}}{\partial \widetilde{x}_m^{k'}} \cdot \widetilde{A}_{m1}}{(\widetilde{A}_{m1} + \widetilde{A}_{m0})^2} \tag{20}$$

where

$$\frac{\partial \widetilde{A}_{m1}}{\partial \widetilde{x}_m^{k'}} = p \prod_{k \in U_m} \alpha_k^{x_m^k} (1-\alpha_k)^{1-x_m^k} \prod_{\bar{k}' \in \widetilde{U}_m \setminus \{k'\}} \widetilde{\alpha}_{\bar{k}'}^{\widetilde{x}_m^{\bar{k}'}} (1-\widetilde{\alpha}_{\bar{k}'})^{1-\widetilde{x}_m^{\bar{k}'}} \cdot [\widetilde{\alpha}_{k'}^{\widetilde{x}_m^{k'}} (1-\widetilde{\alpha}_{k'})^{1-\widetilde{x}_m^{k'}} \log(\widetilde{\alpha}_{k'}) - \widetilde{\alpha}_{k'}^{\widetilde{x}_m^{k'}} (1-\widetilde{\alpha}_{k'})^{1-\widetilde{x}_m^{k'}} \log(1-\widetilde{\alpha}_{k'})], \tag{21}$$

$$\frac{\partial \widetilde{A}_{m0}}{\partial \widetilde{x}_m^{k'}} = (1-p) \prod_{k \in U_m} \beta_k^{1-x_m^k} (1-\beta_k)^{x_m^k} \prod_{\bar{k}' \in \widetilde{U}_m \setminus \{k'\}} \widetilde{\beta}_{\bar{k}'}^{1-\widetilde{x}_m^{\bar{k}'}} (1-\widetilde{\beta}_{\bar{k}'})^{\widetilde{x}_m^{\bar{k}'}} \cdot [-\widetilde{\beta}_{k'}^{1-\widetilde{x}_m^{k'}} (1-\widetilde{\beta}_{k'})^{\widetilde{x}_m^{k'}} \log(\widetilde{\beta}_{k'}) + \widetilde{\beta}_{k'}^{1-\widetilde{x}_m^{k'}} (1-\widetilde{\beta}_{k'})^{\widetilde{x}_m^{k'}} \log(1-\widetilde{\beta}_{k'})]. \tag{22}$$

The above two steps will be iteratively conducted until the convergence criterion is satisfied. In this paper, we define the convergence criterion as $\sqrt{\sum_{k'=1}^{K'} \sum_{m=1}^{M} (\widetilde{x}_m^{k'(t+1)} - \widetilde{x}_m^{k'(t)})^2} < \delta$, which represents the change of $\widetilde{X}$ in two consecutive iterations being less than a threshold δ. After the attacker get the final $\widetilde{X}$, the elements in $\widetilde{X}$ will be transformed to 0 or 1 and then provided to the cloud server as the labels of the malicious workers. The submitted labels $\widetilde{X}$ will be treated as the optimal attack strategy of the attacker. The optimization procedure is summarized as Algorithm 1.

Algorithm 1: Optimal attack against the Dawid-Skene crowdsourcing model

Input: The number of items: M; the number of normal workers: K; the normal workers' labels: X; the number of malicious workers: K'; the items queried to the malicious workers: $\{\widetilde{O}_{k'}\}_{k'=1}^{K'}$

Output: The optimal attack strategy: $\widetilde{X}$

```
Initialize the optimal attack strategy X̃;
repeat
    Estimate the optimal parameters {Ω̃, Θ̃} through conducting
      the EM algorithm described in section 4.1;
    for each x̃_m^{k'} ∈ X̃ do
        Update x̃_m^{k'} according to Eqn. (18);
    end
until The convergence criterion is satisfied;
Transform the elements in X̃ to 0 or 1;
return The optimal attack strategy X̃;
```

5 ATTACK WITH LIMITED KNOWLEDGE

In order to assess the vulnerability of the crowdsourcing system in the worst case, we consider the full knowledge scenario in the above mechanism and assume that the attacker has complete knowledge of the labels from the normal workers (i.e., normal labels) for all items. In fact, the proposed mechanism can also be employed to implement an effective attack even when the attacker only has limited knowledge of the items' normal labels.

Suppose the attacker only knows the normal labels for M' ($M' < M$) items represented as $O' = \{o'_1, o'_2, ..., o'_{M'}\}$. We denote the set of the normal labels for the M' items as $X' = \{x'^k_m\}_{m,k=1}^{M',K}$, which is a subset of X. Since the attacker has no knowledge of the items except those in O', a good choice for him in such a scenario is to let the malicious workers only provide manipulated labels for the items in O' and try to maximize the error of the final results for the M' items. In order to achieve the goal, the attacker could treat X' as the surrogate data of X and employ the above proposed mechanism to derive the attack strategy. In other words, the attack strategy in such a scenario can be derived by solving the following optimization problem:

$$\begin{aligned} \max_{\widetilde{X'}} \quad & \sum_{m=1}^{M'} \mathbb{1}(x'^{*a}_m \neq x'^{*b}_m) + \lambda \sum_{k'=1}^{K'} (\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'}) \\ \text{s.t.} \quad & \{X'^{*a}, \widetilde{\Theta}\} = \arg\max_{X'^{*a}, \widetilde{\Theta}} \log L(\widetilde{\Theta}; \widehat{X'}, X'^{*a}) \\ & \{\widetilde{x'}^{k'}_m\}_{m,k'=1}^{M',K'} \in \{0,1\}, \end{aligned} \tag{23}$$

where $\widetilde{X'} = \{\widetilde{x'}^{k'}_m\}_{m,k'=1}^{M',K'}$ is the attack strategy, i.e., the labels provided by the malicious workers for the items in O'. $X'^{*a} = \{x'^{*a}_m\}_{m=1}^{M'}$ and $X'^{*b} = \{x'^{*b}_m\}_{m=1}^{M'}$ represent the estimated true labels for the M' items based on $\widehat{X'} = X' \cup \widetilde{X'}$ and X' respectively. Although the attack strategy $\widetilde{X'}$ derived based on Eqn. (23) may not be as good as $\widetilde{X}$ based on the full knowledge X, it is the optimal choice for the attacker in the limited knowledge scenario. The performance of the proposed mechanism with limited knowledge is evaluated in Section 6.5.

6 EXPERIMENTS

We conduct experiments based on real-world crowdsourcing datasets to verify the performance of the proposed intelligent attack mechanism.

6.1 Experiment Setup

In this section, we introduce the adopted real-world crowdsourcing datasets, the baseline methods which are compared with the proposed mechanism, and the performance measure.

6.1.1 Datasets. To verify the advantages of the proposed intelligent attack mechanism, we adopt the following real-world crowdsourcing datasets.

Duchenne Smile Dataset [53]. In this dataset, the task is to judge whether the simile in a face image (an item) is Duchenne (enjoyment smile) or Non-Duchenne. The authors in [53] create tasks on the Amazon Mechanical Turk platform, and collect the labels from the participating workers. The number of the items in this dataset is 2,134. Totally, there are 64 normal workers and they provide 17,729 labels.

Product Dataset [51, 58]. Each item in this dataset contains two products (with descriptions), the task is to judge whether the two products are the same or not. The participating workers need to identify whether the two descriptions describe the same product or not, and then provide their labels. In this dataset, there are 8,315 items which are observed by 176 normal workers. Totally, these participating workers provide 24,945 labels.

Sentiment Dataset [58]. Each item in the dataset is a tweet related to a company. The participating workers need to identify whether the tweet has positive sentiment or not to the company. The authors in [58] create 1,000 items and collect labels from 85

normal workers through the AMT platform. Totally, there are 20,000 labels in this dataset.

6.1.2 Baseline Methods. We compare the proposed attack mechanism with two baseline methods: *Baseline_rand* and *Baseline_inversion*.

In the *Baseline_rand* method, the attacker does not consider any strategy, and he just randomly sets the labels of each malicious worker on a given item. This method introduce less overhead to the attacker, as he does not need to take effort to obtain and analyze the crowdsouring data collected from the normal workers.

In the *Baseline_inversion* method, the attacker first conducts the Dawid-Skene model on the labels provided by the normal workers and get the estimated true label for each item. Then he sets each malicious worker's label on a given item as the candidate answer which is different from the estimated true label. This method is an intuitive attack strategy, in which the attacker tries to maximize the number of bad labels injected into the crowdsourcing data.

6.1.3 Performance Measure. In order to evaluate the performance of the proposed attack mechanism, we compare the aggregation results before and after the data poisoning attacks, and adopt the *change rate* as the measure metric. The *change rate* is defined as $\frac{||X^{*a}-X^{*b}||}{M}$, where $X^{*a} = \{x_m^{*a}\}_{m=1}^M$ and $X^{*b} = \{x_m^{*b}\}_{m=1}^M$ are the estimations for the items' true labels after and before the data poisoning attacks. Since the goal of the attacker is to maximize the error of the aggregation results and meanwhile maximally raise the reliability degrees of the malicious workers, thus, the larger the *change rate*, the better the method.

6.2 The Effect of the Percentage of the Malicious Workers

When conducting the data poisoning attack, we assume that the attacker cannot manipulate the labels of normal workers, but he can create or recruit multiple malicious workers. Thus, the number of the malicious workers created or recruited by the attacker plays an important role in the attack. If the attacker is able to create or recruit overwhelming number of malicious workers, the goal of the attacker can be easily achieved with the intuitive attack strategy, i.e., the *Baseline_inversion* method. However, in practice, the attacker can only create or recruit a limited number of malicious workers due to the limitation of his ability. In this experiment, we consider the scenarios where the percentage of malicious workers is low, and evaluate the performance of the proposed mechanism when the percentage is varying.

Suppose N is the number of labels provided by the normal workers for all items. Here we assume that each malicious worker can observe N/K items, which is the average number of the items observed by each normal worker. For each malicious worker, the N/K observed items are randomly selected. In this paper, we set the parameters θ and λ as 100 and 1, respectively. Then we vary the percentage of the malicious workers from 0.03 to 0.27. All the experiments are conducted 50 times and we report the average results. The *change rate* for the three real-world crowdsourcing datasets is shown in Figure 3, in which we represent the proposed mechanism as *The intelligent attack*. From this figure, we can see the proposed attack mechanism performs better than the baseline methods in all cases. When the percentage of the malicious workers is very low (e.g., 3%), since the malicious workers are too few to change the final aggregation results much, the advantage of the proposed mechanism is small. However, when the percentage of the malicious workers increases, the advantage of the proposed attack scheme becomes bigger. For example, when the percentage of malicious workers is 27%, the proposed mechanism successfully attacks nearly 50% of the items in the Duchenne Smile datasets while the baseline methods only obtain marginal utility.

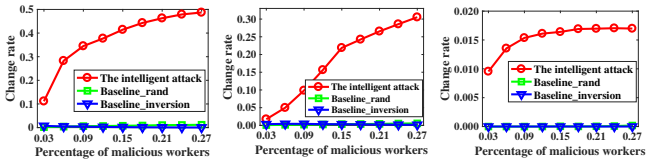


Figure 3: Change rate w.r.t. the percentage of the malicious workers. (a): Duchenne Smile Dataset. (b): Product Dataset. (c): Sentiment Dataset.

6.3 The Effect of the Number of the Queried Items

When the percentage of the malicious workers is given, the number of the items queried to each malicious worker is another important factor in the attack. In this experiment, we study the performance of the proposed mechanism when the number of the items queried to each malicious worker varies.

Here we consider a scenario where the percentage of the malicious workers is very low and we set the value as 3%, i.e., the attacker creates or recruits 2, 6 and 3 malicious workers to the three datasets, respectively. For the Duchenne Smile dataset and the Product dataset, we vary the number of items queried to each malicious worker from 50 to 500, and for the Sentiment dataset, the number of the queried items varies from 100 to 700. The *change rate* for the three datasets is shown in Figure 4. The results in this figure clearly verify that the proposed attack mechanism outperforms the baseline methods in all cases. When the number of the items queried to each malicious worker increases, the advantage of the proposed attack mechanism also increases. The reason is that with the increment of the number of the queried items, the malicious workers can exert more impact on the final aggregation results based on the proposed mechanism. Additionally, this figure also shows that the proposed mechanism can achieve good utility even with very few malicious workers. Take the Duchenne Smile dataset as an example, when each malicious worker provides 250 labels (less than the average number of that from normal workers), the proposed mechanism can successfully attack more than 10% of the items with only 2 malicious workers.

6.4 Comparison on the Ability Parameters of the Malicious Workers

Besides maximizing the error of the aggregation results, the attacker also tries to maximize the malicious workers' reliability degrees (or ability) such that they can be treated as high-quality workers and thus be disguised well. In fact, the proposed mechanism outperforms the baseline methods mainly because we take the effect of the

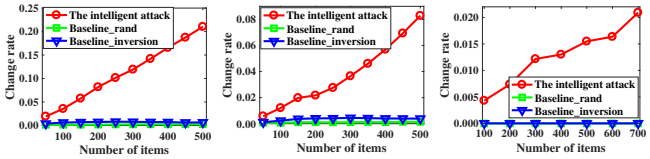


Figure 4: Change rate w.r.t. the number of the items queried to each malicious worker. (a): Duchenne Smile Dataset. (b): Product Dataset. (c): Sentiment Dataset.

malicious workers' reliability degrees into account. The malicious workers can disguise themselves as good workers on some items to enhance their reliability degrees. For the baseline methods, since the malicious workers always disagree with the normal ones or randomly provide their labels, the attack behaviors may be detected by the Dawid-Skene model and the malicious workers will be assigned with low reliability degrees.

In this experiment, we investigate the distribution of the participating workers' ability parameters, i.e., $\alpha = \{\alpha_k, \widetilde{\alpha}_{k'}\}_{k,k'=1}^{K,K'}$ and $\beta = \{\beta_k, \widetilde{\beta}_{k'}\}_{k,k'=1}^{K,K'}$, which can be treated as the reliability degrees of these workers based on the Dawid-Skene model. For each dataset, the percentage of the malicious workers is fixed as 5%. We report the results of the parameters α and β for the three datasets after the data poisoning attacks in Figure 5, Figure 6 and Figure 7, respectively. The results show that the malicious workers from the proposed mechanism have high reliability degrees (both α and β) comparing with the normal workers. This means that the malicious workers blend into the normal workers successfully and they will be treated as high-quality workers according to the Dawid-Skene model. This also verifies that the proposed mechanism can well disguise the malicious behaviors of the attacker while maximizing the error of the aggregated results. In contrast, in the *Baseline_inversion* method, since the malicious workers always disagree with the normal workers, they will be assigned significantly low reliability degrees, which not only limit the performance of the malicious workers, but also make them easy to be detected. As for the *Baseline_rand* method, since the malicious workers randomly select their labels, the values of the ability parameters will be around 0.5. Although the malicious workers from the *Baseline_rand* method can disguise themselves to some extent, their reliability degrees are not large enough to impact the aggregated results.

6.5 The Effect of the Attacker's Knowledge

As described in Section 5, the proposed mechanism can also be employed to implement an effective attack when the attacker only has limited knowledge of the items' normal labels. In this experiment, we evaluate the performance of the proposed mechanism with respect to the value of M'/M, i.e., the percentage of the items whose labels from the normal workers can be known by the attacker. Here we still consider a scenario where the percentage of the malicious workers is very low (3%). We also assume that each malicious worker can observe N/K items, which are randomly selected from O'. Then we vary the value of M'/M from 0.3 to 1 and calculate the *change rate* for the three real-world datasets. We conduct the experiment for 50 times and report the average

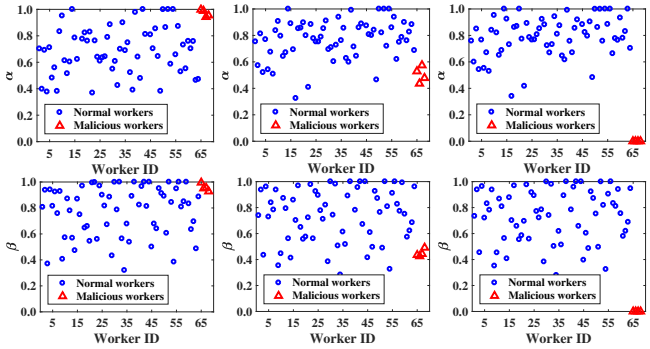


Figure 5: The ability parameters of the normal and malicious workers for the Duchenne Smile dataset.

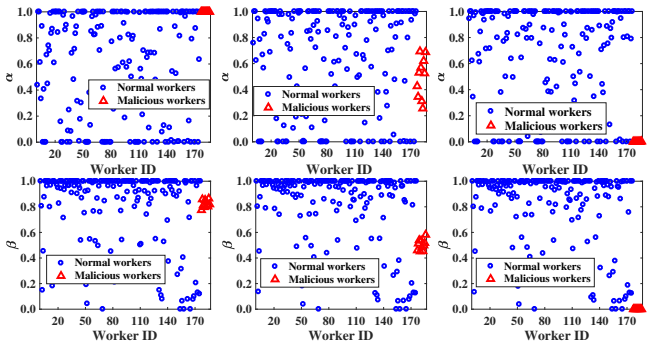


Figure 6: The ability parameters of the normal and malicious workers for the Product dataset.

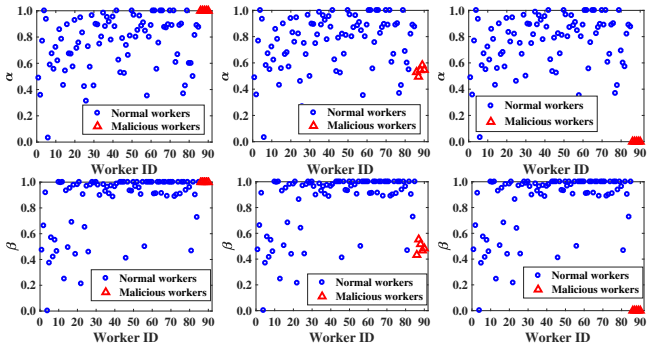
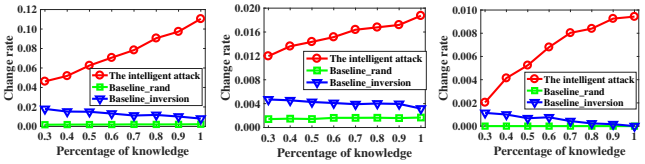


Figure 7: The ability parameters of the normal and malicious workers for the Sentiment dataset.

results in Figure 8, from which we can see the proposed mechanism outperforms the baseline methods in all cases, and the advantage of the proposed mechanism becomes bigger when the attacker's knowledge increases. These results verify that the proposed mechanism can still achieve good utility when the attacker only has limited knowledge of the items' normal labels.

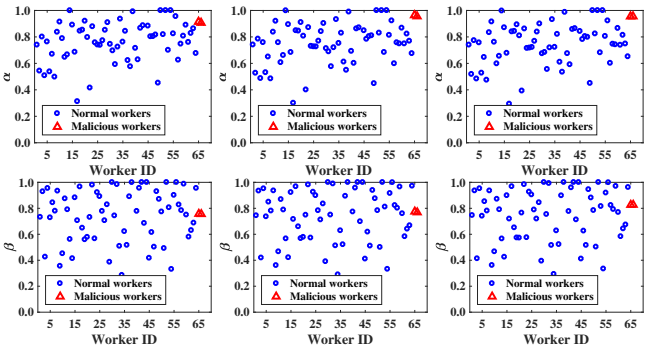
To further evaluate the performance of the proposed mechanism in the limited knowledge scenarios, we investigate the distribution of the workers' ability parameters when the attacker only has partial



(a) (b) (c)

Figure 8: Change rate w.r.t. the percentage of the knowledge known by the attacker (i.e., M'/M). (a): Duchenne Smile Dataset. (b): Product Dataset. (c): Sentiment Dataset.

knowledge of the normal labels. Here we consider three cases in which the percentage of the known items (i.e., M'/M) is set as 0.3, 0.5 and 0.7, respectively. In Figure 9 we report the results of the parameters α and β derived from the proposed mechanism on the the Duchenne Smile Dataset. The results show that the malicious workers keep the high reliability degrees, which means that the proposed mechanism can well disguise the attack behaviors in the limited knowledge scenarios. As for the baseline methods, the results of them on the Duchenne Smile dataset are similar to those in Figure 5.



(a) $M'/M = 0.3$ (b) $M'/M = 0.5$ (c) $M'/M = 0.7$

Figure 9: The workers' ability parameters calculated by the intelligent attack mechanism in the limited knowledge scenarios for the Duchenne Smile Dataset.

7 RELATED WORK

As an effective and low-cost way to solve challenging problems, crowdsourcing, which utilizes the wisdom of crowds, has become more and more popular [5, 10, 13, 15, 16, 27, 29, 31, 32, 36, 38–40]. One important issue for crowdsourcing is that the participating workers are non-experts, so they are likely to provide noisy labels. To address this problem, the researchers have proposed many aggregation methods to estimate the true labels. Among these methods, the Dawid-Skene model [9] has been widely adopted in practice. Compared with the naive aggregation methods such as majority voting, the Dawid-Skene model takes the reliability degrees of workers into account, and it can jointly estimate the items' true labels and each worker's reliability degree. Although different variants have been developed and the theoretical analysis has been conducted for the Dawid-Skene model [7, 8, 30, 34, 41, 44, 46, 57, 59], these works do not take into consideration the sophisticated data poisoning attacks against this model in crowdsourcing. In this paper, we propose an effective data poisoning attack mechanism which can maximize the error of the final results estimated based on the Dawid-Skene model in crowdsourcing. With the spirit of disguising the malicious workers, the above methods cannot defend against this attack effectively.

There are also some crowdsourcing methods which are proposed to eliminate the spammers when collecting labels or conducting aggregation [12, 14, 18, 20, 22, 25, 43, 48]. However, the spammers discussed in these papers are usually the workers who uniformly and/or randomly provide labels in crowdsourcing, which is similar to the *Baseline_rand* attack mechanism. As shown in various experiments, the proposed mechanism can launch significantly more effective attacks than the *Baseline_rand*. Since the malicious workers can disguise themselves well by providing labels intelligently, they will not be detected as spammers.

The importance of the data poisoning attacks has recently been recognized in many crowdsourcing and crowdsensing scenarios [6, 17, 26, 42, 49, 50, 55]. Additionally, there also has been existing work that investigates the data poisoning attacks and related defense schemes in the applications of Internet of Things [21, 45, 56], electric power grids [35] and machine learning algorithms [1, 3, 4, 19, 28, 37, 54]. However, the attacked algorithms discussed in these papers are different from ours. In our designed mechanism, we study the optimal data poisoning attacks against the crowdsourcing systems empowered with the Dawid-Skene model. Compared with the naive aggregation methods such as majority voting, the Dawid-Skene model can defend against the naive malicious workers to some degree, which makes the attack more difficult. The most relevant papers to this work are [23, 24], in which the proposed schemes can identify the malicious workers who conduct the sophisticated data poisoning attacks. However, based on these schemes, the workers who agree with the majority will be classified as normal ones. Since the malicious workers in our proposed mechanism can disguise themselves by agreeing with the majority on some items, these methods will fail to detect them.

8 CONCLUSIONS

In this paper, we investigate crowdsourcing in adversarial environments and study the data poisoning attacks against the crowdsourcing systems with the Dawid-Skene model empowered. In order to find an effective attack strategy for the attacker who aims to maximize the error of the aggregated results, we design an intelligent attack mechanism, based on which an optimal attack strategy can be derived by solving a bi-level optimization problem. With the derived optimal attack strategy, the attacker can not only achieve maximum attack utility but also intelligently disguise the introduced malicious workers as normal ones or even good ones. The experimental results based on real-world datasets demonstrate that the proposed attack mechanism can achieve higher attack utility with very few malicious workers and at the same time, is harder to be detected by the defense mechanisms.

ACKNOWLEDGMENTS

We thank Dr. Niao He from the University of Illinois at Urbana-Champaign for her valuable suggestions. This work was supported in part by the US National Science Foundation under grants CNS-1566374, CNS-1652503, IIS-1553411 and CNS-1742845.

REFERENCES

[1] Scott Alfeld, Xiaojin Zhu, and Paul Barford. 2016. Data Poisoning Attacks against Autoregressive Models. In *Proc. of AAAI*. 1452–1458.

[2] Jonathan F Bard. 1998. *Practical bilevel optimization: algorithms and applications*. Kluwer Academic Publishers.

[3] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. 2006. Can machine learning be secure?. In *Proc. of ASIACCS*. 16–25.

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *Proc. of ICML*.

[5] Marco Brambilla, Stefano Ceri, Andrea Mauri, and Riccardo Volonterio. 2014. Community-based crowdsourcing. In *Proc. of WWW*. 891–896.

[6] Shih-Hao Chang and Zhi-Rong Chen. 2016. Protecting Mobile Crowd Sensing against Sybil Attacks Using Cloud Based Trust Management System. *Mobile Information Systems* 2016 (2016).

[7] Xi Chen, Qihang Lin, and Dengyong Zhou. 2013. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proc. of ICML*. 64–72.

[8] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. 2013. Aggregating crowdsourced binary ratings. In *Proc. of WWW*. 285–294.

[9] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics* (1979).

[10] Luca de Alfaro, Vassilis Polychronopoulos, and Michael Shavlovsky. 2015. Reliable aggregation of boolean crowdsourced tasks. In *Proc. of HCOMP*.

[11] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society* (1977), 1–38.

[12] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2012. Mechanical Cheat: Spamming Schemes and Adversarial Techniques on Crowdsourcing Platforms.. In *CrowdSearch*. 26–30.

[13] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2016. Scheduling human intelligence tasks in multi-tenant crowd-powered systems. In *Proc. of WWW*. 855–865.

[14] Carsten Eickhoff and Arjen P de Vries. 2013. Increasing cheat robustness of crowdsourcing tasks. *Information retrieval* 16, 2 (2013), 121–137.

[15] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-lee Tan, and Jianhua Feng. 2015. icrowd: An adaptive crowdsourcing framework. In *Proc. of SIGMOD*. 1015–1030.

[16] Ujwal Gadiraju, Gianluca Demartini, Ricardo Kawase, and Stefan Dietze. 2015. Human beyond the machine: Challenges and opportunities of microtask crowdsourcing. *IEEE Intelligent Systems* 30, 4 (2015), 81–85.

[17] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. 2015. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proc. of CHI*. 1631–1640.

[18] Matthias Hirth, Tobias Hoßfeld, and Phuoc Tran-Gia. 2010. Cheat-detection mechanisms for crowdsourcing. *University of Würzburg, Tech. Rep* 4 (2010).

[19] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. 2011. Adversarial machine learning. In *Proc. of AISec*. 43–58.

[20] Nguyen Quoc Viet Hung, Duong Chi Thang, Matthias Weidlich, and Karl Aberer. 2015. Minimizing efforts in validating crowd answers. In *Proc. of SIGMOD*. 999–1014.

[21] Vittorio P Illiano and Emil C Lupu. 2015. Detecting malicious data injections in wireless sensor networks: A survey. *ACM Computing Surveys (CSUR)* (2015).

[22] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proc. of the ACM SIGKDD workshop on human computation*. 64–67.

[23] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. 2014. Reputation-based worker filtering in crowdsourcing. In *Proc. of NIPS*. 2492–2500.

[24] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. 2016. Identifying Unreliable and Adversarial Workers in Crowdsourced Labeling Tasks. (2016).

[25] David R Karger, Sewoong Oh, and Devavrat Shah. 2014. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research* 62, 1 (2014), 1–24.

[26] Walter S Lasecki, Jaime Teevan, and Ece Kamar. 2014. Information extraction and manipulation threats in crowd-powered systems. In *Proc. of CSCW*. 248–256.

[27] Edith Law, Ming Yin, Joslin Goh, Kevin Chen, Michael A Terry, and Krzysztof Z Gajos. 2016. Curiosity killed the cat, but makes crowdwork better. In *Proc. of CHI*. 4098–4110.

[28] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *Proc. of NIPS*. 1885–1893.

[29] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J Franklin. 2016. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2296–2319.

[30] Hongwei Li, Bin Yu, and Dengyong Zhou. 2013. Error rate analysis of labeling by crowdsourcing. In *ICML Workshop: Machine Learning Meets Crowdsourcing*.

[31] Qi Li, Fenglong Ma, Jing Gao, Lu Su, and Christopher J Quinn. 2016. Crowdsourcing high quality labels with a tight budget. In *Proc. of WSDM*. 237–246.

[32] Yaliang Li, Jing Gao, Patrick PC Lee, Lu Su, Caifeng He, Cheng He, Fan Yang, and Wei Fan. 2017. A weighted crowdsourcing approach for network quality measurement in cellular data networks. *IEEE Transactions on Mobile Computing* 16, 2 (2017), 300–313.

[33] Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5, 1 (2012), 1–167.

[34] Qiang Liu, Jian Peng, and Alexander T Ihler. 2012. Variational inference for crowdsourcing. In *Proc. of NIPS*. 692–700.

[35] Yao Liu, Peng Ning, and Michael K Reiter. 2011. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security* 14, 1 (2011), 13.

[36] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. 2015. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proc. of KDD*. 745–754.

[37] Shike Mei and Xiaojin Zhu. 2015. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In *Proc. of AAAI*. 2871–2877.

[38] Chuishi Meng, Wenjun Jiang, Yaliang Li, Jing Gao, Lu Su, Hu Ding, and Yun Cheng. 2015. Truth discovery on crowd sensing of correlated entities. In *Proc. of SenSys*. 169–182.

[39] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. 2015. Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In *Proc. of SenSys*. 183–196.

[40] Quoc Viet Hung Nguyen, Tam Nguyen Thanh, Ngoc Tran Lam, Son Thanh Do, and Karl Aberer. 2013. A Benchmark for Aggregation Techniques in Crowdsourcing. In *Proc. of SIGIR*.

[41] Jungseul Ok, Sewoong Oh, Jinwoo Shin, and Yung Yi. 2016. Optimality of belief propagation for crowdsourced classification. In *Proc. of ICML*. 535–544.

[42] Zhengrui Qin, Qun Li, and George Hsieh. 2013. Defending against cooperative attacks in cooperative spectrum sensing. *IEEE Transactions on Wireless Communications* 12, 6 (2013), 2680–2687.

[43] Vikas C Raykar and Shipeng Yu. 2012. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research* 13, Feb (2012), 491–518.

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

[45] Mohsen Rezvani, Aleksandar Ignjatovic, Elisa Bertino, and Sanjay Jha. 2015. Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks. *IEEE Transactions on Dependable and Secure Computing* 12, 1 (2015), 98–110.

[46] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proc. of the EMNLP*. 254–263.

[47] Norases Vesdapunt, Kedar Bellare, and Nilesh Dalvi. 2014. Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment* 7, 12 (2014), 1071–1082.

[48] Jeroen Vuurens, Arjen P de Vries, and Carsten Eickhoff. 2011. How much spam can you take? an analysis of crowdsourcing results to increase accuracy. In *Proc. of CIR*. 21–26.

[49] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. 2016. Defending against sybil devices in crowdsourced mapping services. In *Proc. of MobiSys*. 179–191.

[50] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. 2014. Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers.. In *USENIX Security Symposium*. 239–254.

[51] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. Crowder: Crowdsourcing entity resolution. *Proc. of the VLDB Endowment* 5, 11 (2012), 1483–1494.

[52] Peter Welinder and Pietro Perona. 2010. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *Proc. of CVPRW*. 25–32.

[53] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. of NIPS*. 2035–2043.

[54] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning?. In *Proc. of ICML*. 1689–1698.

[55] Dong Yuan, Guoliang Li, Qi Li, and Yudian Zheng. 2017. Sybil Defense in Crowdsourcing Platforms. In *Proc. of CIKM*. 1529–1538.

[56] Kuan Zhang, Xiaohui Liang, Rongxing Lu, and Xuemin Shen. 2014. Sybil attacks and their defenses in the internet of things. *IEEE Internet of Things Journal* 1, 5 (2014), 372–383.

[57] Yuchen Zhang, Xi Chen, Denny Zhou, and Michael I Jordan. 2014. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. In *Proc. of NIPS*. 1260–1268.

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

[59] Denny Zhou, Sumit Basu, Yi Mao, and John C Platt. 2012. Learning from the wisdom of crowds by minimax entropy. In *Proc. of NIPS*. 2195–2203.