Soc. Netw. Anal. Min. manuscript No.
(will be inserted by the editor)

Protecting the Dawid-Skene Algorithm against Low-Competence Raters and Collusion Attacks with Gold-Selection Strategies

Conny Kühne · Klemens Böhm

Received: date / Accepted: date

Abstract More and more online communities classify contributions based on collaborative ratings of these contributions. A popular method for such a rating-based classification is the Dawid-Skene algorithm (DSA). However, despite its popularity, DSA has two major shortcomings: (1) It is vulnerable to raters with a low competence, i.e., a low probability of rating correctly. (2) It is defenseless against collusion attacks. In a collusion attack, raters coordinate to rate the same data objects with the same value to artificially increase their remuneration.

In this paper, to cope with these issues, we propose gold strategies based on the level of agreement between raters. Gold strategies adopt the notion of gold objects, i.e., contributions whose true value is known. We show that selecting gold objects at random, as is common in the literature, does not increase the accuracy of DSA in a low-competence setting to a satisfying degree. Instead, our gold strategies select contributions based on the level of agreement between community members, i.e., to which extent their ratings agree on the class of a given contribution. To maximize the net benefit of gold objects, i.e., their benefit minus their costs, we propose an adaptive algorithm. It determines the number of gold objects based on runtime information. We extensively evaluate the effectiveness of gold strategies in low-competence settings and against collusion attacks by means of simulation. We find that gold strategies based on a high level of agreement between raters improve the accuracy of DSA in low-competence settings considerably. Further, the gold strategies are highly effective against collusion attacks. Finally, the adaptive algorithm determines the optimal gold ratio for each strategy and each setting with high accuracy.

Keywords Classfication · Dawid-Skene Algorithm · Gold Strategies · Collusion Attacks

Conny Kühne
Institute for Program Structures and Data Organisation, Karlsruhe Institute of Technology
Am Fasanengarten 5, Building 50.34, 76131 Kalrsruhe, Germany
E-mail: conny.kuehne@kit.edu

Klemens Böhm
Institute for Program Structures and Data Organisation, Karlsruhe Institute of Technology
Am Fasanengarten 5, Building 50.34, 76131 Kalrsruhe, Germany
E-mail: klemens.boehm@kit.edu

1 Introduction

For an increasing number of online communities, the contributions created by the community are processed automatically. Such communities must classify the contributions, i.e., decide which class out of a set of predefined classes a contribution belongs to. For example, consider an online community that creates an ontology. Here, the community needs to decide, e.g., if a contribution is a class or an instance or if a name of an item in the ontology is correct or not. We investigate rating-based classification methods for such communities. The general scenario is the following: Members of the community create contributions collaboratively and subsequently rate each others' contributions according to the quality they perceive. After the ratings have been submitted, the community classifies the contributions by aggregating the ratings of the contributions. For the work at hand, we mainly focus on a binary classification setting, i.e., a given contribution can belong to two possible classes.

Our open-community setting differs from the typical paid crowdsourcing settings such as Amazon Mechanical Turk (AMT). AMT offers mechanisms to qualify workers by asking them to rate specific contributions. Based on this qualification, workers can be excluded from participating in certain tasks. In contrast to this, we assume a high degree of autonomy of the individual community members. We can neither

manipulate individual raters to rate selected contributions nor exclude them from rating.

A simple scheme for aggregating ratings is the well-known majority vote (MV). Despite its simplicity, MV can achieve a surprisingly high accuracy provided that the quality of ratings is sufficiently high (Condorcet 1785). One aim of ours is to give an intuition for the high performance of MV and to show under which conditions it is achieved. Aggregating ratings by means of the weighted majority vote (WMV) can increase the classification accuracy compared to MV, provided WMV knows the individual competence of each rater, i.e., his probability of rating correctly. Intuitively, weighted majority vote assigns a higher weight to high-competence raters than to low-competence raters. Yet, in general, rater competencies are unknown.

For this case, Dawid and Skene (1979) proposed an algorithm to estimate the competencies of the raters and to classify the contributions accordingly. We refer to this algorithm as the Dawid-Skene algorithm (DSA). DSA was originally developed to combine opinions of multiple physicians for medical diagnosis. With over 450 citations, DSA is one of the most widely-cited algorithms for classifying items based on ratings by raters with unknown competencies. In recent years, there have been a lot of proposals to use DSA – and algorithms based on or closely related to DSA – in particular for crowdsourcing settings (Whitehill et al. 2009; Wang et al. 2011; Raykar and Yu 2012; Wang et al. 2013). However, despite its popularity, DSA has two major shortcomings: (1) It is vulnerable to low-competence settings, and (2) it is defenseless against collusion attacks.

Firstly, if the mean competence of the community of raters is close to or less than random, e.g., close to or less than 0.5 in binary classification tasks, DSA performs rather poorly. Such a low mean competence can occur if the topic of the community is inherently difficult. It can also occur if the community has a large fraction of spammers, biased raters, malicious raters, or simply raters with consistent misunderstanding. A spammer assigns ratings randomly, independent of the true value of the object rated. Biased raters give consistently too high or too low ratings. Ipeirotis et al. (2010) give the following example of a biased rater: Think of a task of classifying web content into the categories appropriate and non-appropriate for children. For this task, parents of young children tend to rate more conservatively than the general population. That is, they tend to consistently rate sites that are objectively appropriate for children as inappropriate. Another example might be a community that builds a knowledge base on the advantages and disadvantages of various kinds of power plants. Here, environmental activists would likely be biased in their assessments. For example, they might assess nuclear power plants as more dangerous than they are objectively. Further, raters might give anti-correlated ratings. That is, on average, they invert ratings, either maliciously or because of a consistent misunderstanding. Settings with mean competence close to or less than random seem rare but do occur. For example, Kazai et al. (2013) report an average mean competence close to random (0.56) over their eight binary open crowdsourcing tasks. In one of their tasks, the workers reached only a mean competence of 0.35. Such low-competence settings can have a devastating effect on the classification accuracy of DSA and related approaches which assume that the majority is correct on average.

Secondly, DSA is defenseless against collusion attacks. In a collusion attack, raters coordinate to rate the same data objects with the same value to artificially increase their estimated competence. This is beneficial for the colluders if they receive a remuneration for their ratings that is based on their estimated competence. For example, many online communities remunerate users with reputation or Karma points for high-quality contributions. We propose to compute remunerations in such communities contingent on the competence estimates by DSA. Similarly, Wang et al. (2013) propose an algorithm that pays crowdsourcing workers based on the competence estimates calculated by DSA, among other metrics. However, they do not address collusion attacks. When the remuneration is based on the estimated competence, a collusion attack allows colluders to artificially increase their remuneration while saving cognitive effort for determining the truthful value of the data objects. Since DSA assigns an inflated weight to their low-competence ratings, colluders can also severely damage the accuracy of DSA.

We propose *gold strategies* based on the level of agreement to increase the accuracy of DSA in low-competence settings and to counter collusion attacks. Gold strategies adopt the notion of gold objects, i.e., contributions that DSA knows the true value of. Gold objects are a common approach in the literature to differentiate between high- and low-quality raters. The approaches known in the literature use predetermined, randomly selected gold objects. However, as we will show, simply selecting contributions as gold objects at random does not increase the accuracy to a satisfying degree in our setting. Instead, our gold strategies select contributions based on the ratings they have received. Specifically, gold strategies select contributions based on the level of agreement between community members, i.e., to what extent their ratings agree on the class of a given contribution. Subsequently, trusted experts rate the selected contributions, thereby turning them into gold objects. Of course, the accuracy benefit of gold objects is offset by their costs. Consequently, we are interested in maximizing the *net benefit* of gold objects, i.e., the benefit of a given number of gold objects minus their costs. Determining the number of gold objects that maximizes the net benefit a priori is infeasible. We propose an algorithm that adaptively determines the number of gold objects based on runtime information.

To summarize, we make the following contributions:

- *Properties of MV and WMV.* We discuss the relationship between the number of raters, the competence distribution, and the accuracy for MV and WMV under different assumptions w.r.t. the competence of raters.
- *Estimation quality of DSA.* We study the effect of the competence distribution and the number of ratings on the estimation quality of DSA in different settings.
- *Gold strategies based on the level of agreement.* A common approach in the literature to differentiate between high- and low-quality raters is to evaluate their ratings by means of a set of predetermined or randomly selected gold objects. In contrast, we propose gold strategies, i.e., selecting contributions for evaluation by expert raters, which use the level of agreement between the ratings of the community as a selection criterion. We test the effectiveness of these gold strategies in various settings by means of simulation.
- *Adaptive Gold Algorithm.* We propose an algorithm that determines the number of gold objects in order to maximize the net benefit. Instead of fixing a predetermined number of gold objects, the adaptive algorithm automatically decides when to stop adding further gold objects based on runtime information.
- *Collusion attacks.* We study the effects of collusion attacks against DSA. Further, we test the effectiveness of gold strategies to reduce the benefit gained by colluding. To the best of our knowledge, we are the first to study collusion attacks against DSA.

A main finding of ours is that gold strategies based on a high level of agreement between raters improve the accuracy of DSA in low-competence settings considerably. Moreover, the adaptive gold algorithm reaches over 90 percent of the net benefit that the respective gold strategy can maximally achieve. Finally, we find that gold strategies are highly effective in countering collusion attacks against DSA. Even though our analysis concentrates on DSA, the related methods that are based on DSA potentially suffer in low-competence settings and under collusion attacks as well. This is because these methods either use the output of DSA as input for their algorithm Ipeirotis et al. (2010), or they make assumptions similar to those of DSA (even though with some modifications), and, like DSA, use an expectation-maximization framework to estimate competencies and to classify the rated items (Whitehill et al. 2009; Raykar and Yu 2012). Thus, our findings are somewhat orthogonal to DSA and applicable in principle to these related methods as well.

This paper is structured as follows. Section 2 introduces the formal model and the notation. Section 3 discusses the accuracy of majority voting and related decision rules. Section 4 presents the DSA algorithm. Section 5 introduces two basic simulation settings we use as templates for the evaluation of DSA. Section 6 analyzes the effects of the number of data objects and the mean competence of the raters on the accuracy of DSA. Section 7 discusses the gold strategies and evaluates them. Section 8 introduces and evaluates our adaptive gold algorithm. Section 9 discusses the effects of collusion attacks against DSA and evaluates the gold strategies to counter them. Section 10 reviews related work. Section 11 discusses the main findings, and Section 12 concludes.

2 Model and Notation

We consider an online community where participants can rate the contributions of their peers. Let $K = \{1,\dots,m\}$ denote the set of contributions and let $k \in K$ denote a single contribution. We also call a contribution a data object in the following. We assume that each data object has a fixed type t from the set of types T. We focus on a binary setting, i.e., there are two different types (for example 'correct/incorrect', 'high/low' etc.). We encode the types with -1 and 1, i.e., $T = \{-1,1\}$.

We write o_k to denote the true type of data object k and $o_k = t$ to denote the event that the true type of k is t. Let $p(t)$ denote the prior probability of a randomly chosen data object to be of type t. Raters are those participants of the online community who issue ratings. We use $I = \{1,\dots,n\}$ to denote the set of all n raters of the community.

Let $r_{i,k} \in T$ denote the rating given by rater i to data object k. We use $R = \{r_{i,k}\}$ to denote the set of all ratings and $s = |R|$ to denote the number of all ratings. Each rater rates each data object at most once. We use $R_k = \{r_{i,k'} \mid k' = k\}$ to denote the set of ratings for data object k and $s_k = |R_k|$ to denote the number of ratings for k.

A classification method estimates the type $\hat{o}_k \in T$ of each data object k. We use $\hat{o}_k = o_k$ and $\hat{o}_k \neq o_k$ to denote the events that the classification method estimates the type of o_k correctly and incorrectly, respectively. We use $\hat{\theta}$ to indicate an estimator of a parameter θ. Finally, let $\mathbb{1}(\cdot)$ be the indicator function, i.e., $\mathbb{1}(\cdot)$ is equal to one if its argument holds true, and equal to zero otherwise. See Table 2 in Section 15 for a summary of the notation.

2.1 Competence Models

A rater perceives the type of a data object with some error and rates it according to his perception. Note that we assume that raters do not act strategically. I.e., we do not consider the case where raters might change their behavior depending on the behavior of other raters. (We do consider this case below in Section 9 when discussing collusion attacks.) Let $P(r_{i,k} = q \mid o_k = t)$ denote the *response probability* that rater i gives a rating of value $q \in T$ given that the true type of the rated data object is $t \in T$. We assume that $P(r_{i,k} = q \mid o_k = t)$ is the same for all data objects k of type t. In other words, for a given rater all data objects of a given type are equally difficult.

Further, we assume that, conditional on the type of a data object, ratings are independent and identically distributed.

We use the following three models to capture assumptions of increasing strictness on the response probability.

Type-Dependent Competence. We call the probability that rater i rates correctly given that the true type of the data object is t, i.e., $c_i^{(t)} = P(r_{i,k} = t \mid o_k = t)$, i's *competence* for type t. Since we consider binary types, i.e., $t \in \{-1, 1\}$, it follows that $P(r_{i,k} = t \mid o_k = t) = 1 - P(r_{i,k} = q \mid o_k = t)$ for $t \neq q$. Thus, the set of competencies $\{c_i^{(-1)}, c_i^{(1)}\}$ specifies all response probabilities of rater i.

Heterogeneous Type-Independent Competence. We assume that rater i rates correctly with competence $c_i = c_i^{(t)}$ that is the same for both types $t \in \{-1, 1\}$.

Homogeneous Competence. We assume that every rater i has the same type-independent competence $c = c_i$.

Having defined the competence, we can clarify the notion of spammers, biased, and anti-correlated raters introduced above. A biased rater's competence is low for one type only. An anti-correlated rater inverts ratings, either because he is malicious or he consistently mixes up both categories. This means that his competence for both types is less than 0.5. A spammer has competence 0.5.

3 The Accuracy of Majority Decision Rules

To gain insights into the relationship between rater competence and classification accuracy we start by investigating the accuracy of the following majority decision rules: majority vote, and maximum a posteriori probability (MAP) rule, as well as the weighted majority vote as a special case of the MAP rule.

3.1 Majority Vote

MV decides for type t if more than one half of the ratings are in favor for t [1]

$$\hat{o}_k = t \text{ if } \sum_{r_{i,k} \in R_k} \mathbb{1}(r_{i,k} = t) \geq \left\lfloor \frac{s_k}{2} \right\rfloor + 1 \qquad (1)$$

where $\lfloor s_k/2 \rfloor$ denotes the 'floor' under $s_k/2$, i.e., the largest integer smaller than $s_k/2$.

[1] The rule that decides in favor for the type t that receives most of the ratings, i.e., $\hat{o}_k = t$, if $\arg\max_{t \in T} = \sum_{r_{i,k}} \mathbb{1}(r_{i,k} = t)$, is called plurality vote. In the literature simple majority vote and plurality vote are often both called majority vote according to Kuncheva (2004). Plurality vote and majority vote are equivalent for settings where the number of types is two and the number of ratings is odd.

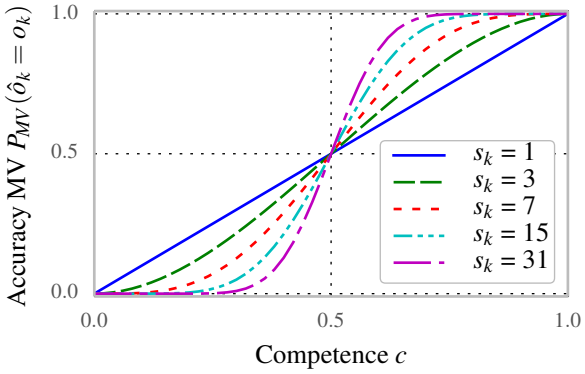


Fig. 1: Accuracy of majority vote for homogeneous competence and odd numbers of ratings s_k.

3.2 Accuracy of Majority Vote for Homogeneous Competencies

First, we discuss the case where all raters have the same competence c, i.e., $c = c_i$ for all i. (Later, we will drop this assumption.)

MV classifies a contribution correctly if more than half of the ratings are correct (cf. Equation 1). For example, suppose that we want to classify a contribution based on three ratings. In this case, MV decides correctly if exactly two ratings are correct, for which there are three possible ways, or if exactly three ratings are correct. Thus, the probability of correct classification is the sum of two terms: $3c^2(1-c) + c^3$. The general formula for the accuracy of MV under homogeneous competence can be derived by summing up the probabilities that l_k out of s_k ratings are correct for all $l_k \geq \lfloor s_k/2 \rfloor + 1$, i.e.,

$$P_{MV}(\hat{o}_k = o_k) = \sum_{l_k = \lfloor s_k/2 \rfloor + 1}^{s_k} \binom{s_k}{l_k} (c)^{l_k} (1-c)^{s_k - l_k}. \qquad (2)$$

Figure 1 illustrates the relationship between the accuracy of MV and the competence for different odd numbers of raters.

The relationship between the competence, the number of raters, and the accuracy of MV has first been formulated by Condorcet (1785) and is known as Condorcet's jury theorem (CJT): For odd s_k and homogeneous rater competence c

- if $c > 0.5$, then $P_{MV}(\hat{o}_k = o_k)$ increases monotonically in s_k,
  and $\lim_{s_k \to \infty} P_{MV}(\hat{o}_k = o_k) = 1$,
- if $c < 0.5$, then $P_{MV}(\hat{o}_k = o_k)$ decreases monotonically in s_k,
  and $\lim_{s_k \to \infty} P_{MV}(\hat{o}_k = o_k) = 0$,
- if $c = 0.5$, then $P_{MV}(\hat{o}_k = o_k) = 0.5$ for all s_k.

Grofman et al. (1983) show that the CJT is also valid for heterogeneous type-independent competencies c_i if the distribution of c_i is symmetric around the mean $\bar{c}$. In that case, $\bar{c}$ substitutes c in the CJT.

3.2.1 Optimality of Majority Vote for Homogeneous Competencies greater than 0.5

Even though it is a simple method, MV achieves a high accuracy, provided that the competence is greater than 0.5. In fact, if we add the assumption that the prior is not too heavily skewed in favor of one type, we can show that MV is the optimal classification scheme for two-type settings.

Proposition 1 (Optimality of Majority Vote under Homogeneous Competence) *Under the assumption of the homogeneous competence model in Section 2.1 and given that s_k is odd, $c > 0.5$, and $1-c < p(t) < c$ for both types $t \in \{-1,1\}$, majority vote is an optimal decision rule, i.e., there does not exist any decision rule that has a higher accuracy.*

The proof is in Section 13.

3.3 MAP Rule and Weighted Majority Vote for Known Competencies and Type Priors

The MAP rule and WMV are restatements of Bayes' theorem. Thus, both methods yield optimal estimates of the data object types $\hat{o}_k$, assuming that they know the true competencies and the true type prior. This is rarely satisfied in the real world. However, discussing these methods reveals insights into the behavior of DSA.

3.3.1 Maximum a Posteriori Probability Rule

The MAP rule considers the type dependent competence model with known competencies. It decides in favor of the type t with the maximum posterior probability, given the ratings, i.e.,

$$\hat{o}_k = \arg\max_{t \in T} P(o_k = t \mid R_k). \tag{3}$$

Ties are handled arbitrarily. Since $T = \{-1,1\}$, this is equivalent to deciding in favor of the type with the higher posterior log-odds

$$\hat{o}_k = \operatorname{sign}\left(\log \frac{P(o_k = 1 \mid R_k)}{P(o_k = -1 \mid R_k)}\right).$$

By Bayes' theorem, the posterior probability that data object k is of type t given the ratings R_k is

$$P(o_k = t \mid R_k) = \frac{P(R_k \mid o_k = t) \cdot p(t)}{P(R_k)}.$$

Since we assume conditional independence of the ratings given the type of the data object, the likelihood of the ratings can be expressed as the product of the individual likelihoods $P(R_k \mid o_k = t) = \prod_{r_{i,k} \in R_k} P(r_{i,k} \mid o_k = t)$. Thus, the posterior log-odds for data object k are

$$\log \frac{P(o_k = 1 \mid R_k)}{P(o_k = -1 \mid R_k)} = \log \frac{p(1)}{p(-1)} + \sum_{r_{i,k} \in R_k} \log \frac{P(r_{i,k} \mid o_k = 1)}{P(r_{i,k} \mid o_k = -1)}. \tag{4}$$

In other words, in our two-type model, the MAP rule is equivalent to a majority vote that weighs each rating by the log ratio of the rating likelihoods of its rater:

$$\hat{o}_k = \operatorname{sign}\left(\log \frac{p(1)}{p(-1)} + \sum_{r_{i,k} \in R_k} \log \frac{P(r_{i,k} \mid o_k = 1)}{P(r_{i,k} \mid o_k = -1)}\right). \tag{5}$$

That is, a rating $r_{i,k} = q$ has weight $\log(P(r_{i,k} = q \mid o_k = 1)/P(r_{i,k} = q \mid o_k = -1))$. Later, we will use this insight to derive a weighted measure of the level of agreement between raters.

3.3.2 Weighted Majority Vote with Optimal Weights

WMV with optimal weights is a special case of the MAP rule. It assumes known type-independent heterogeneous competence $c_i = c_i^{(t)} = P(r_{i,k} = t \mid o_k = t)$ for all types t and all data objects k. Thus, we can reformulate the individual likelihoods

$$\frac{P(r_{i,k} \mid o_k = 1)}{P(r_{i,k} \mid o_k = -1)} = \begin{cases} c_i/(1-c_i) & \text{if } r_{i,k} = 1 \\ (1-c_i)/c_i & \text{if } r_{i,k} = -1. \end{cases}$$

Since $\log c_i/(1-c_i) = -\log(1-c_i)/c_i$ we can restate Eq. 5 as the WMV rule

$$\hat{o}_k = \operatorname{sign}\left(\log \frac{p(1)}{p(-1)} + \sum_{r_{i,k} \in R_k} r_{i,k} v_i(c_i)\right)$$

with

$$v_i(c_i) = \log \frac{c_i}{(1-c_i)}. \tag{6}$$

being the optimal weight of rater i.

Equation 6 reveals a relationship between the (type-independent) competence of a given rater i and i's usefulness for the estimation of the data object type. Raters with competence $c_i > 0.5$ have positive weight. Raters with competence $c_i < 0.5$ give on average the opposite of the true rating, either maliciously or because of consistent misunderstanding. The function v_i reverses the "direction" of their ratings by assigning them a negative weight. Moreover, $v_i(c_i) = -v_i(1-c_i)$. In other words, a low competence rater i' with $c_{i'} < 0.5$ is equally beneficial for the accuracy of WMV as a high competence rater i'' with $c_{i''} = 1 - c_{i'}$. Spammers, i.e., raters with competence near 0.5, on the other hand, are worst for the accuracy of WMV. This is because they generate ratings that are completely random. They have zero weight, i.e., $v_i(0.5) = 0$.

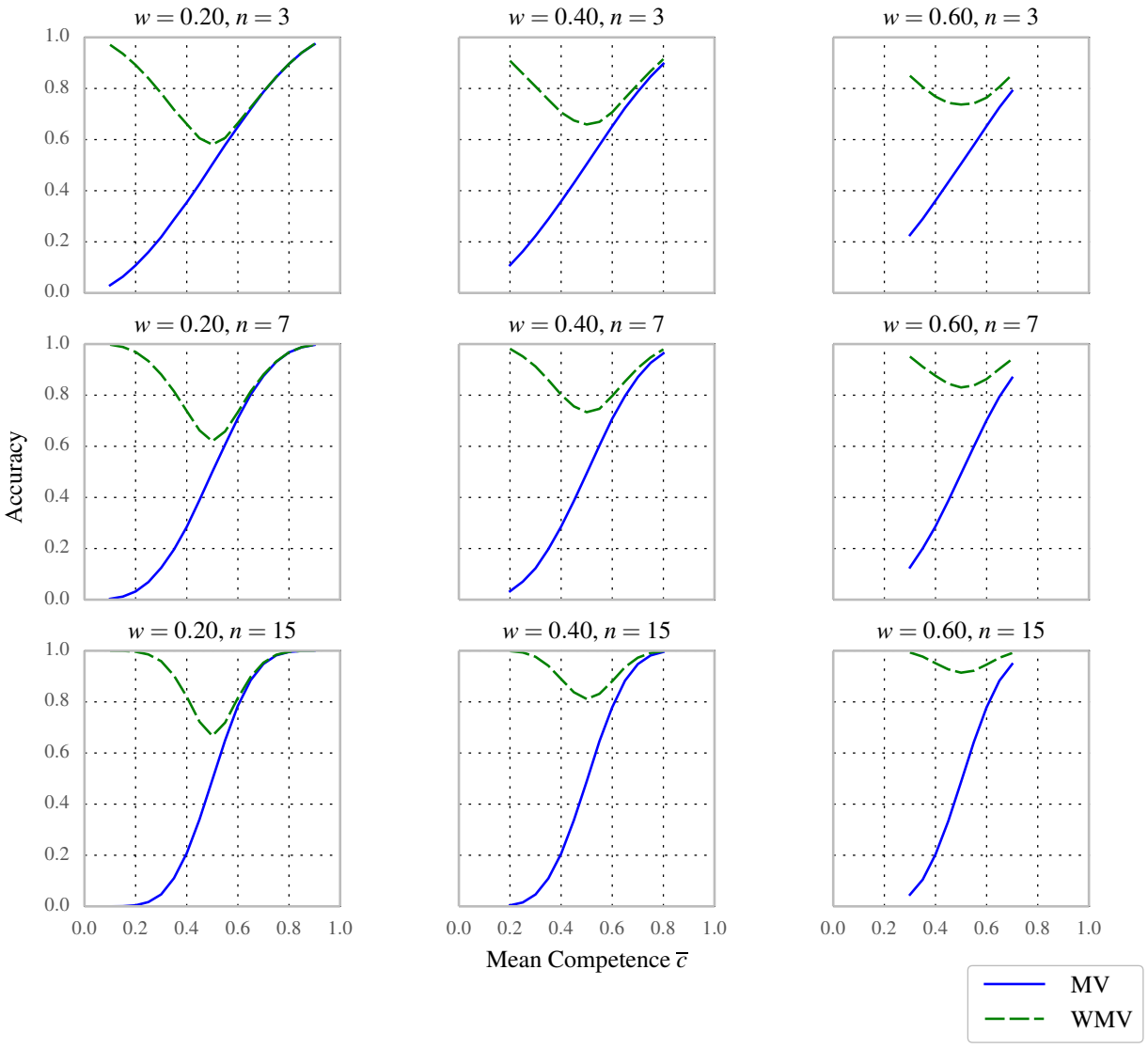


Fig. 2: Accuracy of MV and WMV for known competencies c_i for different competence interval widths w and numbers of raters n.

3.4 Accuracy of Majority Vote vs. Accuracy of Weighted Majority Vote with Known Rater Competencies

We explore the influence of the number of raters and of the competence distribution on the accuracy of WMV. To this end, we run a simulation under the type-independent heterogeneous competence model. We draw the competence c_i of each rater i uniformly at random from the interval $[\bar{c} - w/2, \bar{c} + w/2]$, with mean competence $\bar{c}$ and interval width w. For example, for $w = 0.4$, and $\bar{c} = 0.5$ we draw competencies uniformly from the interval $[0.3, 0.7]$. We vary the mean competence $\bar{c}$ from $0 + w/2$ to $1 - w/2$ in 0.05 steps. We use a uniform prior, i.e., $p(-1) = p(1) = 0.5$ and average the results over 100 simulation runs. The simulation breaks ties by fair coin toss. In the simulation we use the competence c_i to compute i's optimal weight. Since the prior $p(t)$ is uniform, it has no influence on the type estimates. Figure 2 shows the accuracy of MV and WMV as a function of the mean competence $\bar{c}$ for different numbers of raters n and different interval widths w. For $\bar{c} > 0.5$ the accuracy of WMV is higher than or as high as the accuracy of MV. Like MV, WMV benefits from higher n.

The accuracy curves are minimal at $\bar{c} = 0.5$ and symmetric w.r.t. the line $\bar{c} = 0.5$. In other words, the further $\bar{c}$ is away from 0.5, the higher the accuracy. This is because high-competence and low-competence raters are equally beneficial for the accuracy of WMV, as described above. Competencies near 0.5, on the other hand, are worst for the accuracy of WMV because they generate random ratings. This also explains why WMV benefits from larger competency ranges

w. Namely, the probability of having raters with very high or with very low competence increases with w.

As already mentioned, WMV with known competencies is optimal (under the model in Section 2) because WMV is a restatement of Bayes' theorem. Therefore, it represents an upper bound for the mean accuracy of rating aggregation methods discussed, in particular for DSA. However, WMV assumes that the competencies c_i of raters and the $p(t)$ are known quantities. In open settings, this assumption does not hold. The remainder of this paper deals with the problem of what to do if competencies are unknown.

4 Estimation of Rater Competencies and Data Object Types with DSA

DSA relies on the type dependent competence model.

Using only the ratings as inputs, DSA estimates (1) the competencies $\hat{c}_i^{(t)}$, (2) the type priors $\hat{p}(t)$, and (3) the type probabilities $\hat{P}(o_k = t)$. DSA iterates between computing the competencies (1) and the type priors (2) using the type probabilities (3) as input, and computing the type probabilities (3) using the competencies (1) and the type priors (2) as input.

Algorithm 1: Our implementation of DSA.

Input: set of ratings $\{r_{i,k}\}$ given by rater i to data object k, additive smoothing parameters a and d
Output: set of estimated types $\{\hat{o}_k\}_{k\in K}$, set of estimated response probabilities $\{\hat{P}(r_{i,k} = q \mid o_k = t)\}_{i\in I, q\in T, t\in T}$ (equivalent to set of estimated competencies $\{\hat{c}_i^{(t)}\}_{i\in I, t\in T}$)

1 **foreach** *contribution* $k \in K$ **do**
2 initialize $\hat{P}(o_k = t)$ with majority vote.
3 **repeat**
4 **foreach** *type* $t \in T$ **do**
5 $\hat{p}(t) \leftarrow \dfrac{a + \sum_{k=1}^{m} \hat{P}(o_k = t)}{ad + m}$
6 **foreach** $i \in I$, $q \in T$, $t \in T$ **do**
7 $\hat{P}(r_{i,k} = q \mid o_k = t) \leftarrow \dfrac{a + \sum_{k\in K} r_{i,k}^{(q)} \cdot \hat{P}(o_k = t)}{ad + \sum_{q\in T}\sum_{k\in K} r_{i,k}^{(q)} \cdot \hat{P}(o_k = t)}$
8 **foreach** *contribution* $k \in K$ *and each type* $t \in T$ **do**
9 $\hat{P}(o_k = t) \leftarrow \dfrac{\hat{p}(t)\prod_{i\in I}\prod_{q\in T} \hat{P}(r_{i,k} = q \mid o_k = t)^{\mathbb{1}(r_{i,k}=q)}}{\sum_{t\in T} \hat{p}(t)\prod_{i\in I}\prod_{q\in T} \hat{P}(r_{i,k} = q \mid o_k = t)^{\mathbb{1}(r_{i,k}=q)}}$
10 **until** $\{\hat{P}(r_{i,k} = q \mid o_k = t)\}_{i\in I, q\in T, t\in T}$ *converges*;
11 **foreach** $k \in K$ **do**
12 $\hat{o}_k \leftarrow \arg\max_{t\in T} \hat{P}(o_k = t)$

For brevity and clarity we use the estimates of the response probabilities $\hat{P}(r_{i,k} = q \mid o_k = t)$ instead of the competence estimates in the following description. Since we use binary types this is equivalent to using the type dependent competencies (cf. Section 2.1). Having said this, DSA proceeds as follows (cf. Algorithm 1). It initializes the type estimates for each data object. Then it repeats the following three steps until convergence.

1. It estimates the prior of type t by summing up the estimated probabilities of each data object being of type t and dividing the sum by the number of data objects m (line 5 in Algorithm 1).
2. To infer the response probability $\hat{P}(r_{i,k} = q \mid o_k = t)$, DSA sums up the ratings i has given in favor for type q and weighs each rating by the estimated probability that the rated data object is of type t. It normalizes the obtained sum by the weighted sum of all ratings of i (line 7).
3. DSA computes the posterior probability that data object k has type t given the ratings it has received (line 9). Since it does not know the true response probabilities and the true type prior neccessary for the computation, DSA uses the estimates of these quantities obtained in the two previous steps.

Finally, DSA classifies each data object by assigning the type t that has the maximum estimated posterior probability.

We have added two implementation details that the authors of DSA did not specify. First, the authors leave the initialization of $\hat{P}(o_k = q)$ unspecified. We use MV to this end. Further, we use additive smoothing (line 5 and line 7 of Algorithm 1) to avoid 0 probabilities that would cancel out all other factors of the product in line 9 of Algorithm 1. For our two-type setting, we set the smoothing parameters $a = 0.1$ and $d = 2$.

Obtaining the competence estimates $\hat{c}_i$ of the type-independent competence model is straightforward. We simply sum up DSA's estimates of the type dependent competencies and weigh them with the estimates of the type priors

$$\hat{c}_i = \hat{c}_i^{(-1)}\hat{p}(-1) + \hat{c}_i^{(1)}\hat{p}(1). \tag{7}$$

In the following we investigate the behavior of DSA.

5 Settings of a Simulation to Analyze DSA

To gain insights into the behavior of DSA we analyze its performance by means of simulation. This simulation is rather unrelated to the previous one in Section 3.4. We use the type-independent heterogeneous competence model. For each rater i we generate random ratings according to his competence c_i. We draw the competence c_i of each rater i uniformly at random from the interval $[\bar{c} - w/2, \bar{c} + w/2]$, with mean competence $\bar{c}$ and interval width w. To describe the number of ratings per rater we introduce a simulation parameter *rating rate*. The rating rate of rater i, $rr_i \in [0,1]$, is the probability that i assigns a rating to a given data object.

We use two basic simulation settings – UNIFORM and SKEWED – to cover two common scenarios. They differ

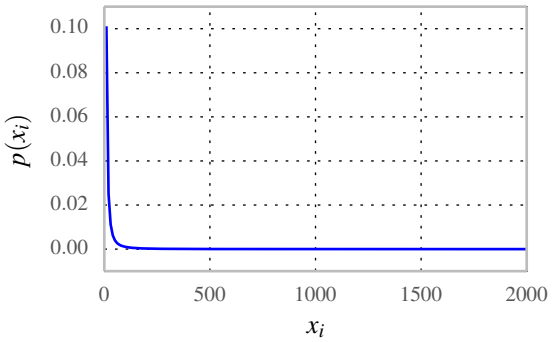


Fig. 3: The probability density function of the truncated Pareto distribution of setting SKEWED.

with respect to the distribution of user ratings, the number of raters, the number of data objects, and the type prior. We have determined the default number of data objects m for each setting by simulating each setting with successively increasing m while keeping the other parameters fixed (see Section 6.1). The resulting default m for UNIFORM and SKEWED are the points where the accuracy of DSA starts to converge to a steady state.

5.1 Simulation Setting UNIFORM

This setting represents a small, homogeneous community for example in a company or a lecture community. The default number of raters for this setting is $n = 50$. We also use smaller n in experiments where we want to analyze the effect of n on the accuracy of DSA. The rating rate in this setting is the same for all raters, i.e., $rr = rr_i$ for all raters $i \in I$. We set $rr = 0.4$, that is, on average 2 out of every 5 raters issue a rating for a given data object. The prior for a data object to be of a given type is uniform, i.e., $p(0) = p(1) = 0.5$. We set the default number of data objects to $m = 400$. We average results of this settings over 100 simulation runs with different random seeds.

5.2 Simulation Setting SKEWED

This setting represents an open online community with a highly skewed rating rate and a skewed prior. To this end we draw the number of ratings per rater from a Power-law distribution. The main characteristic of such a distribution is that most ratings come from a small fraction of the raters while most raters issue only very few ratings each. Power-law distributions are frequently observed in open online communities (Mamykina et al. 2011; Meka et al. 2009). Since the number of ratings is bounded by the number of data objects, we draw the number of ratings x_i for each rater i from a truncated Pareto distribution (Aban et al. 2006) defined by the density function

$$p(x_i) = \frac{\alpha x_{min}^{\alpha} x_i^{-\alpha-1}}{1-(x_{min}/x_{max})^{\alpha}}$$

for $0 < x_{min} \leq x_i \leq x_{max} < \infty$, where $x_{min} < x_{max}$. The upper bound is $x_{max} = m$ since each rater can issue at most one rating per data object. We set m to 2000 for this setting. Further, we set the lower bound for the number of ratings per rater to $x_{min} = 10$. Estimating the competencies of raters who have issued less than 10 ratings becomes unnecessarily inaccurate (we argue). We set the shape parameter of the distribution to a typical value of $\alpha = 1$. To obtain discrete values for the number of ratings we round to the nearest integer (Clauset et al. 2009). This results in a highly right-skewed distribution of x_i with a skewness of approx. 5.29 and a mean rating rate of approx. 0.026 (see Fig. 3). We set the prior for this setting to $p(1) = 0.7$ because we assume that a typical online community has an uneven ratio of good vs. bad data objects. Finally, we set the number of raters to $n = 100$.[2] As in UNIFORM we average the results of 100 simulation runs with different random seeds.

6 Analyzing the Estimation Quality of DSA

We conduct a series of simulation experiments to find out how accurately DSA estimates

- the true type of the data objects, and
- the competence of the raters.

In the following, we call the ratio of data objects that DSA classifies correctly *classification accuracy* or simply *accuracy*. Alternatively, we measure the error rate, which equals one minus the accuracy. Further, we measure the error rate of the competence estimates. To this end, we define the mean absolute difference between the competencies of the raters and their estimated competencies

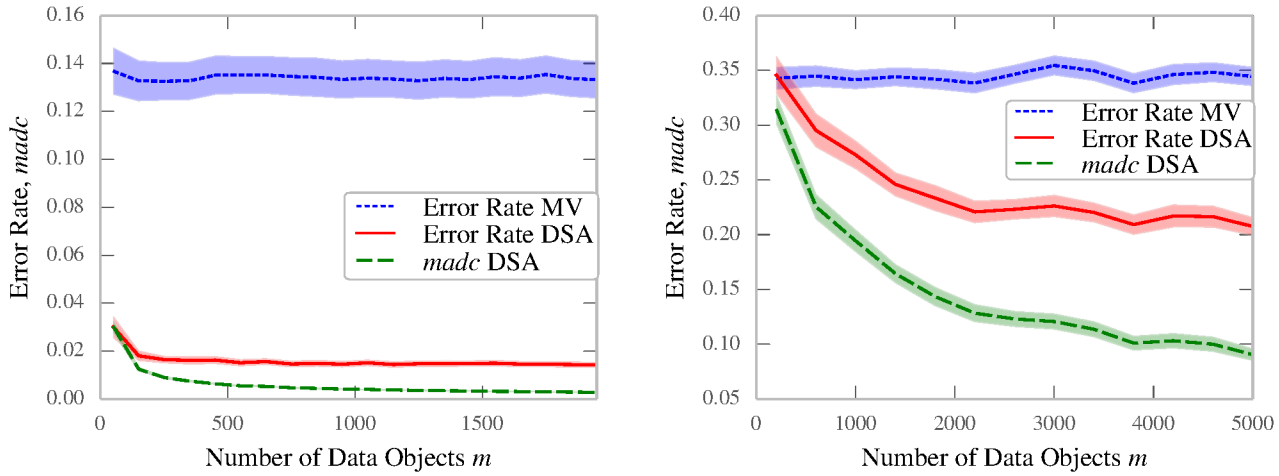
$$madc = \sum_{i=1}^{n} |c_i - \hat{c}_i|/n.$$

where $\hat{c}_i$ denotes the estimate of the type-independent competence of rater i obtained by means of Eq. 7.

6.1 Effect of the Number of Data Objects on the Estimation Quality of DSA

How does the number of data objects m influence the estimation quality of DSA? Or, put differently: If the community size stays constant but the number of contributions created

[2] We chose $n = 100$, since we deem smaller, more unstable communities the more interesting case. Further, our results (not shown) indicate, that larger n do not change the simulation results to a significant degree.



(a) Setting UNIFORM

(b) Setting SKEWED

Fig. 4: Error rate and mean absolute difference between estimated and real competence *madc*. Bands show 95% percent confidence intervals.

and rated by the community grows over time, then how many contributions does it take for DSA's accuracy and *madc* to stabilize? To answer this question, we conduct one simulation experiment for each basic setting. For both settings we draw the c_i uniformly at random from the interval $[0.3, 0.95]$.

Figure 4 shows the effect of the number of ratings on the error rate and *madc*. For setting UNIFORM (see Fig. 4a) the error rate as well as *madc* of DSA converge for $m > 400$. For SKEWED we set the lower bound for the number of ratings per rater proportional to m, $x_{min} = m/200$, to keep the rating rate approximately equal for different values of m. The DSA error rate in SKEWED shows less convergent behavior (see Fig. 4b). Nevertheless, there is a strong reduction of the error rate and the *madc* of DSA for increasing numbers of data objects up to about 2000. Consequently, in the following simulations we set $m = 400$ for UNIFORM and $m = 2000$ for SKEWED.

In simple terms, this means that in more homogeneous settings, where raters have roughly the same rating rate and the rating rate is high, DSA stabilizes relatively early. In a more open setting with a low rating rate and a skewed rating distribution that is typical for open internet communities, this is different. Here, DSA requires a much higher number of data objects to stabilize.

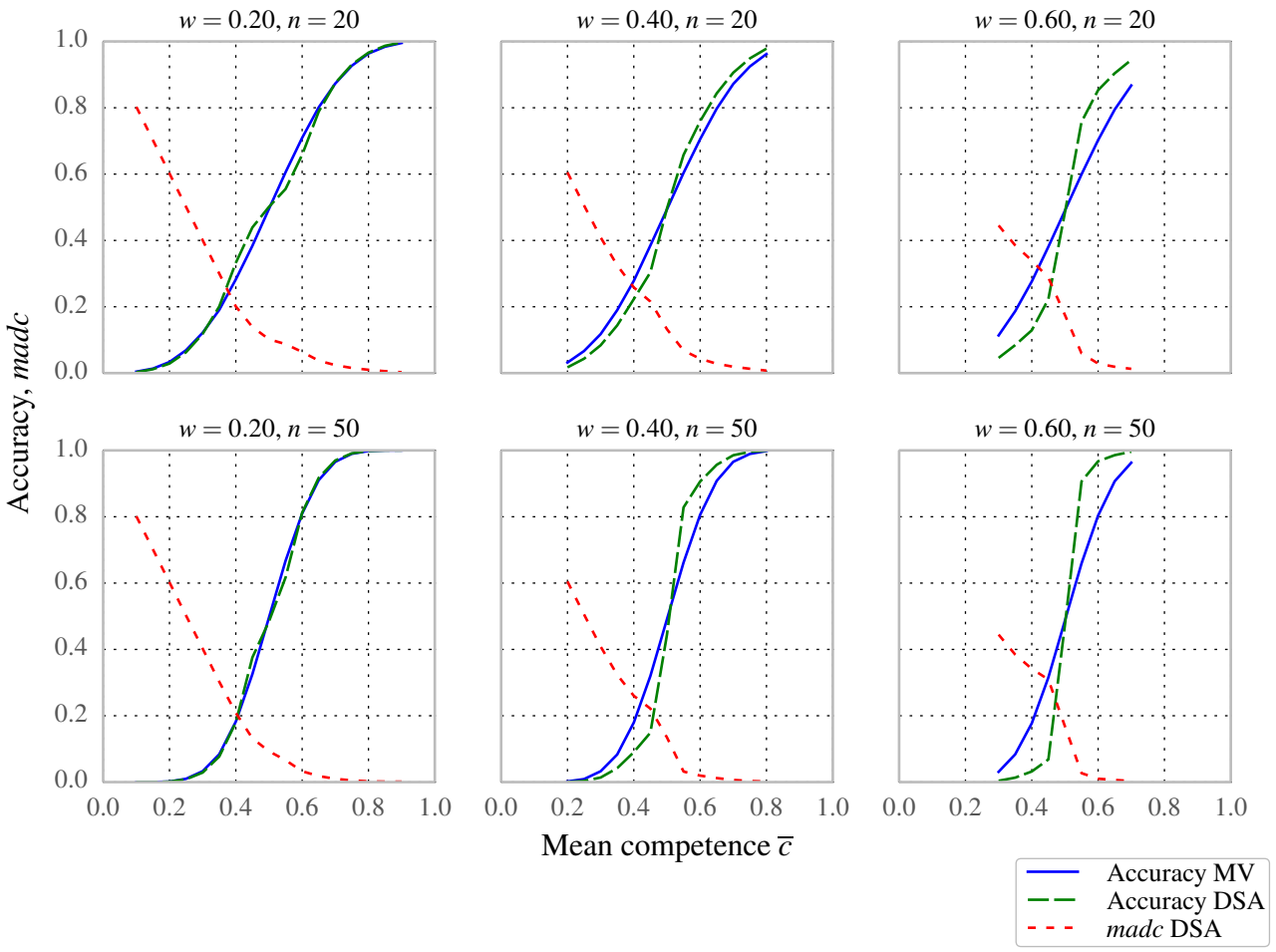
6.2 Effects of the Number of Raters and of the Competence Distribution on the Estimation Quality of DSA

As we have seen, the accuracies of MV and WMV depend on the number of raters and on their competencies (Section 3.4). To find out how these two parameters affect the estimation accuracy of DSA, we conduct a simulation experiment using the same procedure as in Section 3.4. That is, we draw the competencies of the n raters uniformly at random from the interval $[\bar{c} - w/2, \bar{c} + w/2]$, with mean $\bar{c}$ and interval width w. We vary the mean competence $\bar{c}$ from $0 + w/2$ to $1 - w/2$ in 0.05 steps. The accuracy of MV serves as a baseline.

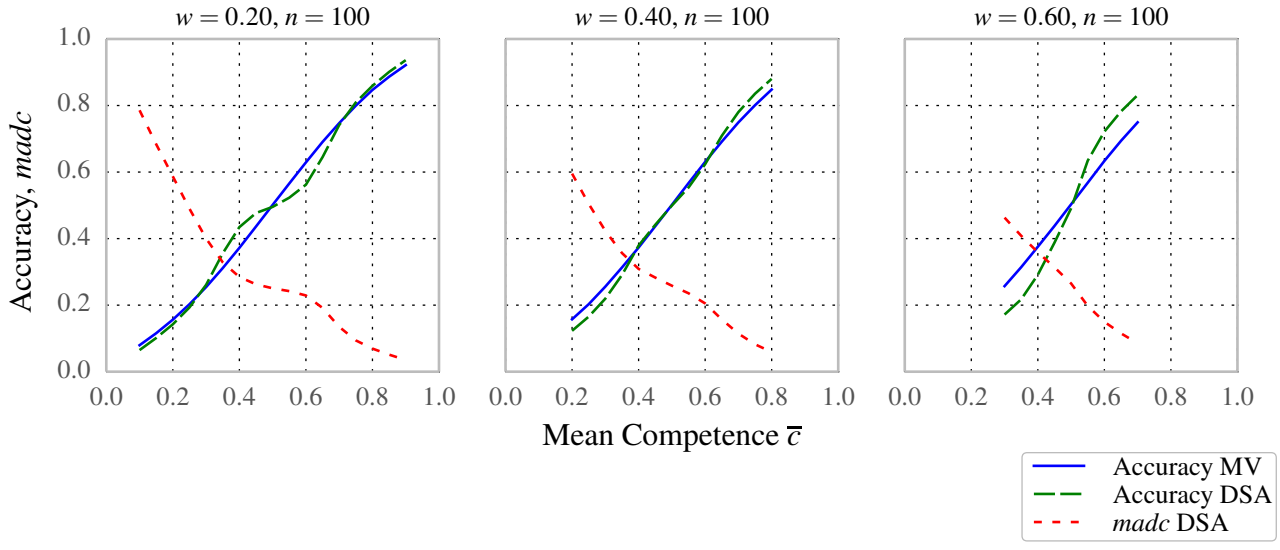
Unsurprisingly, DSA's accuracy increases with increasing mean competencies (Fig. 5). For the narrow competence range $w = 0.2$, the accuracies of MV and DSA are similar.

For higher w, DSA's accuracy differs markedly from MV's accuracy. Here, depending on the mean competence $\bar{c}$, DSA's accuracy is either (1) worse than MV's accuracy, for $\bar{c} < 0.5$, or (2) better than MV's accuracy, for $\bar{c} > 0.5$. The reason for this effect is that DSA uses the type estimates $\hat{P}(o_k = t)$ to compute the competence estimates $c_i^{(t)}$, and vice versa. For $\bar{c} < 0.5$, DSA performs worse than MV because in this case the type estimates are highly inaccurate. This in turn causes DSA to invert the competency estimates to some degree: It assigns a low competence to high-competence raters and vice versa. This leads to even more inaccurate type estimates, and so on. The opposite is the case for $\bar{c} > 0.5$. Here, DSA estimates the types more accurately. This in turn yields more accurate competence estimates. This is reflected in the error rate of the competence estimates *madc*. For UNIFORM, *madc* drops almost to 0 for $\bar{c} > 0.6$. In that case, DSA's accuracy approaches that of WMV for known rater competencies (Fig. 2). The higher w and n, the more pronounced this effect becomes.

The curves for the accuracy of both DSA and MV are flatter in the SKEWED setting (Fig. 5b) than in the UNIFORM setting (Fig. 5a). This means that, for the same $\bar{c}$, the accuracy in SKEWED is higher than in UNIFORM for $\bar{c} < 0.5$ and



(a) Setting UNIFORM



(b) Setting SKEWED

Fig. 5: Accuracy and *madc* as a function of the mean competence $\bar{c}$ for different numbers of raters n and different competence interval widths w.

lower for $\bar{c} > 0.5$. Consequently, the *madc* curves are flatter in SKEWED than in UNIFORM as well.

7 Using Gold Strategies to Increase the Accuracy of DSA in Low-Competence Settings

As we have seen, for mean competencies $\bar{c} < 0.5$ DSA's accuracy is low. *Gold objects*, i.e., data objects which we know the true type of, can increase the accuracy of DSA in such low-competence settings. The idea behind using gold objects for DSA is the following. An accuracy of DSA of less than 1.0 means that DSA misclassifies some data objects. Knowing the true type of these data objects with certainty allows DSA to estimate more accurately the competence of the raters who have given ratings to these data objects. This in turn leads to a higher accuracy for the type estimates of non-gold objects these raters have rated. This increases the accuracy of competence estimates even further and so on.

We obtain gold objects by selecting some data objects and letting trusted experts rate these data objects.

Algorithm 2: Modified DSA with gold objects.

```
/* dots (...) indicate unchanged parts from
   Algorithm 1.                                  */
```

Input: ...; set of known types for gold objects $\{o_{k'}\}_{k' \in K_{gold}}$

```
...
repeat
    ...
    foreach contribution k' ∈ K_gold and each type t ∈ T do
        if o_k' = t then
            P̂(o_k' = t) ← 1
        else
            P̂(o_k' = t) ← 0
until {P̂(r_i,k = q | o_k = t)}_i∈I,q∈T,t∈T converges;
...
```

We use $K_{gold} \subseteq K$ to denote the set of gold objects. To integrate gold objects into DSA we use the same straightforward procedure as Wang et al. (2011) and simply set the type estimates of the gold objects to their known values at the end of the `repeat until` loop in Algorithm 1. Algorithm 2 shows the resulting modified DSA. For simplicity it shows mostly the modified parts. Dots (...) indicate unchanged parts from Algorithm 1.

Since gold objects are costly we want to use them effectively. For crowdsourcing services such as Amazon Mechanical Turk, Wang et al. (2011) propose to achieve this by actively forcing crowdsourcing workers to rate predefined gold objects. However, forcing raters to rate particular contributions is not possible in an open community scenario like ours.

Instead, we propose *gold strategies* to determine which existing contributions of the community to choose as gold objects. We use the following procedure to incorporate gold strategies into an open community scenario. (1) Select contributions as gold objects based on the selection criterion of the respective gold strategy. (2) Determine the true type of these gold objects by means of trusted experts.[3] (3) Run the modified DSA (Algorithm 2) with the gold objects to estimate the types of all contributions.

The most straightforward of the gold strategies, the UNI strategy, selects gold objects uniformly at random.

7.1 Gold Strategies Based on the Level of Agreement

Additionally to UNI, we propose gold strategies that take the *level of agreement* between raters into account, i.e., to what extent the raters agree on the type of a given data object. The strategies select data objects as gold objects that either have a high (HI) or a low (LO) level of agreement. The rationale for using a high level of agreement is the following. In low-competence communities, i.e., communities with $\bar{c} < 0.5$, a high level of agreement on a type t of data object k indicates that t is likely not the correct type of k. This is because raters with competence less than 0.5 have a higher chance of being incorrect than of being correct. Thus, DSA will likely compute the estimate $\hat{P}(o_k = t)$ inaccurately and, as a consequence, inaccurately estimate the competencies of the raters who have rated k. So the benefit of selecting k as a gold object is potentially high. Conversely, if $\bar{c} > 0.5$, the probability of estimating $\hat{P}(o_k = t)$ inaccurately is highest for data objects with a low level of agreement. Later, we quantify the exact error probability given a certain level of agreement for a simplified setting using MV and homogeneous competence.

In this section, we introduce two methods to measure the level of agreement: the absolute rating sum (ARS) and the estimated absolute posterior log-odds (ALO). Based on the two measures, we define four gold strategies: HI-ARS, LO-ARS, HI-ALO, LO-ALO (prefix HI-/LO- for the level of agreement).

In the following, we use g to denote the gold ratio. The gold ratio is the ratio of gold objects among all data objects, i.e., $g = m_{gold}/m$, where $m_{gold} = |K_{gold}|$ is the number of gold objects.

7.1.1 Gold Strategies Based on the Absolute Rating Sum

We define the absolute rating sum of data object k as $ars_k = \left|\sum_{r_{i,k} \in R_k} r_{i,k}\right|$, where $r_{i,k} \in \{-1, 1\}$. Low and high values of ars_k indicate low and high levels of agreement, respectively.

[3] For domains where we do not trust experts to be completely accurate we could combine the ratings of several experts, for example by means of DSA, to achieve a higher accuracy.

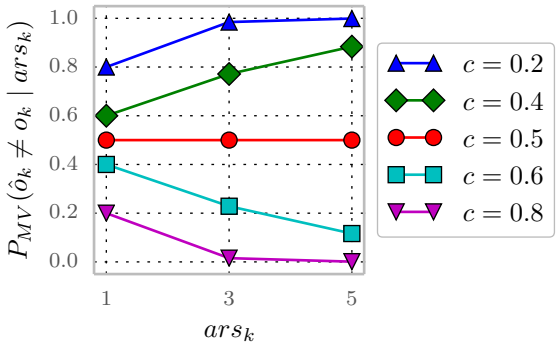


Fig. 6: Probability of incorrect classification by MV given the absolute rating sum (ars_k) for five raters with homogeneous competence c.

Consequently LO-ARS selects the first m_{gold} data objects ordered by ars_k ascending, while HI-ARS selects the first m_{gold} data objects ordered by ars_k descending.

As an example, consider a setting with two data objects 1 and 2 and their ratings $R_1 = \{1,-1,1\}$ and $R_2 = \{1,1,-1,-1\}$. Assume a gold ratio of $g = 0.5$, i.e., half of the data objects are selected as gold objects. The absolute rating sums are $ars_1 = 1$ and $ars_2 = 0$. Thus, HI-ARS selects data object 1 as gold object while LO-ARS selects object 2.

Error probability given ars_k using MV and homogeneous competence. Earlier, we have given an intuition why the level of agreement is a useful criterion for selecting data objects as gold objects: it can identify data objects whose type DSA will likely estimate inaccurately. To further strengthen this intuition, we now quantify – even though only for a simplified setting – the probability that MV will classify a data object incorrectly given the absolute rating sum. We use MV because (1) it allows for an analytical quantification, and (2) it has roughly a similar accuracy as DSA for a given mean competence (see Section 6).

Proposition 2 *Let an odd number of ratings s_k for data object k, and homogeneous competence c be given. Then, the probability that MV estimates the type of k incorrectly given the absolute rating sum ars_k is*

$$P_{MV}(\hat{o}_k \neq o_k \mid ars_k) = \frac{c^{l_k} \cdot (1-c)^{s_k - l_k}}{c^{l_k} \cdot (1-c)^{s_k-l_k} + (1-c)^{l_k} \cdot c^{s_k-l_k}}$$

where $l_k = (s_k - ars_k)/2$ is the number of correct ratings for k.

See Section 14 for the proof of Proposition 2.

Figure 6 illustrates this relationship for $s_k = 5$. Depending on the competence c, the probability of incorrect classification either increases or decreases with an increasing absolute rating sum: It increases for $c < 0.5$, and it decreases for $c > 0.5$. This is what the intuition suggests. If the (mean) competence is below 0.5, a high level of agreement indicates a high error probability. Conversely, if the competence is above 0.5, a low level of agreement indicates a high error probability.

7.1.2 Gold Strategies Based on the Estimated Absolute Posterior Log-Odds

The absolute posterior log-odds for data object k are the absolute value of the posterior log-odds (Eq. 4)

$$\begin{aligned} alo_k &= \left| \log \frac{P(o_k = 1 \mid R_k)}{P(o_k = -1 \mid R_k)} \right| \\ &= \left| \log \frac{p(1)}{p(-1)} + \sum_{r_{i,k} \in R_k} \log \frac{P(r_{i,k} \mid o_k = 1)}{P(r_{i,k} \mid o_k = -1)} \right|. \end{aligned}$$

Like the posterior log-odds, the absolute posterior log-odds are a weighted measure of the agreement level. In addition to summing up the ratings like ars_k, alo_k weighs each rating by the log ratio of the response likelihoods of its rater. See Section 3.3.1 for a discussion of the weights.

We cannot calculate alo_k directly because the true competencies and the true type priors are unknown parameters. Instead we run DSA to obtain estimates of the posterior probability $\hat{P}(o_k = t \mid R_k) = \hat{P}(o_k = t)$ for data object k being of type t (see line 9 of Algorithm 1) and use these estimates to calculate the estimate of alo_k

$$\widehat{alo_k} = \left| \log \frac{\hat{P}(o_k = 1 \mid R_k)}{\hat{P}(o_k = -1 \mid R_k)} \right|.$$

High values of $\widehat{alo_k}$ indicate a high level of agreement for data object k, while low values indicate a low level of agreement. Thus, LO-ALO selects the first m_{gold} data objects ordered by $\widehat{alo_k}$ ascending, and HI-ALO selects the first m_{gold} data objects ordered by $\widehat{alo_k}$ descending.

7.2 Evaluation of Gold Strategies

How much does the accuracy of DSA benefit from the different gold strategies? Ideally, the use of gold objects increases the number of non-gold objects that DSA classifies correctly. To study to which extent this indeed occurs, we define the net accuracy as the ratio of correctly classified non-gold data objects

$$netacc = \frac{\sum_{k \in K \setminus K_{gold}} \mathbb{1}(\hat{o}_k = o_k)}{\left| K \setminus K_{gold} \right|}. \tag{8}$$

To quantify the accuracy gains of DSA with gold objects compared to the vanilla DSA without gold objects, we do the following. For each $\bar{c}$ and each gold strategy, we calculate the net accuracy that DSA achieves using gold objects $netacc^{gold}$.

For the same input data, we then run DSA without gold objects ($g = 0$) and measure its net accuracy $netacc^{nogold}$ (which is equal to its accuracy). The net accuracy gain is the difference between the net accuracy of DSA with gold and the net accuracy of DSA without gold

$$netaccgain = netacc^{gold} - netacc^{nogold}. \quad (9)$$

To find out which mean competencies benefit most from the use of gold strategies, we simulate the net accuracy gains as a function of the mean competence. As in the previous sections, we vary the mean competence $\bar{c}$ from $0 + w/2$ to $1 - w/2$ in 0.05 steps and set $w = 0.5$. For setting UNIFORM, we simulate the gold ratios $g \in \{0.05, 0.1, 0.15\}$. Our results indicate that SKEWED requires fewer gold objects than UNIFORM for similar gains. Consequently, for this setting, we simulate gold ratios $g \in \{0.02, 0.04, 0.06\}$.

Figure 7 presents the results of the simulation experiments. In particular, Fig. 7a, and Fig. 7b show the net accuracy gains of gold strategies for setting UNIFORM with 20 raters and 50 raters, respectively. Figure 7c shows the net accuracy gains of gold strategies for setting SKEWED.

As we have expected, the net accuracy gains, both for UNIFORM and SKEWED, are higher for higher gold ratios. For all strategies, the gains for $\bar{c}$ greater than approximately 0.55 are close to zero. The reason is that the accuracy of DSA without gold objects is already high for $\bar{c} > 0.55$ (cf. Fig. 5) so there is not much room for improvement. For $\bar{c} < 0.55$, HI-ALO has gains greater than or equal to all other gold strategies. For some $\bar{c} < 0.55$, HI-ALO outperforms the other strategies by a wide margin. In the setting with a high number of ratings per data object (UNIFORM with 50 raters), the highest gains concentrate near $\bar{c} = 0.5$ for low gold ratios. In settings where ratings are sparse – either because the number of raters is low (UNIFORM with 20 raters) or because the rating rate is low (SKEWED) – and which have a high gold ratio, HI-ALO achieves high gains also for very low $\bar{c}$. In these settings, HI-ARS has a performance as good as or slightly worse than HI-ALO. We discuss the implications of these findings in Section 11.

8 Optimizing the Net Benefit of Gold Objects with an Adaptive Gold Algorithm

What is the optimal number of gold objects for DSA? This number not only depends on the benefits but also on the costs. As we have seen above, DSA benefits from gold objects by an increase in correctly classified non-gold data objects. This benefit is offset by the costs to obtain the gold objects. The exact costs and benefits depend on the specific scenario. For simplicity, we assume a benefit of 1 per correctly classified data object and costs of 1 per gold object. We define the net benefit as the difference of correctly classified non-gold data objects and the number of gold objects used by DSA

$$netbenefit = \sum_{k \in K \setminus K_{gold}} \mathbb{1}(\hat{o}_k = o_k) - \left|K_{gold}\right|.$$

In the following, we discuss how to maximize the net benefit.

8.1 Adaptive Gold Algorithm

As before, we select gold objects by means of a gold strategy. But instead of fixing a gold ratio a priori, we let an algorithm decide how many gold objects to use in order to achieve the highest net benefit. The algorithm is adaptive, i.e., it decides when to stop based on runtime information. It works iteratively: Starting with zero gold objects, it adds one gold object per iteration. The goal of the algorithm is to stop adding further gold objects when the net benefit is highest. Of course, in the real world, the net benefit is unknown. Therefore, we cannot use it as a stop condition for the algorithm. Instead, we can only observe how the output of DSA changes in order to decide when to stop adding further gold objects. The adaptive gold algorithm is outlined in Algorithm 3. In the following, we derive the stop condition.

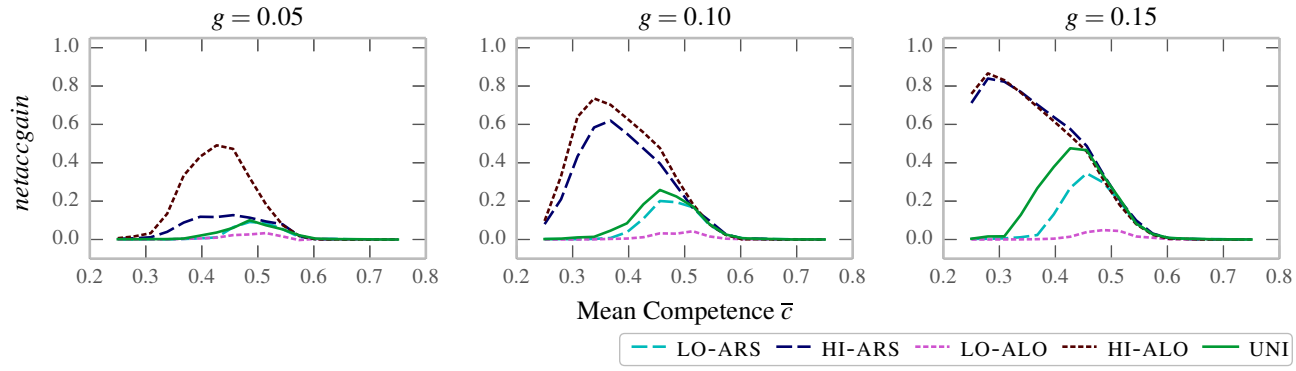
Algorithm 3: Adaptive Gold Algorithm for DSA.

Input: set of ratings $\{r_{i,k}\}$, gold strategy gs
Output: set of estimated types $\{\hat{o}_k\}_{k \in K}$, set of estimated response probabilities $\{\hat{P}(r_{i,k} = q \mid o_k = t)\}_{i \in I, q \in T, t \in T}$

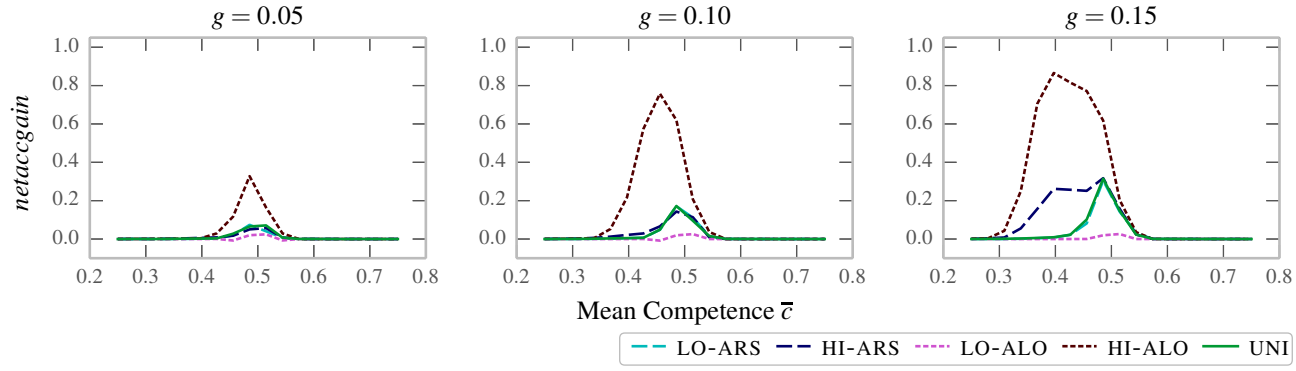
$\{d_1, d_2, \ldots, d_{|K|}\} \leftarrow$ set of data objects ordered according to gs
$K_{gold} \leftarrow \emptyset$
$itr \leftarrow 0$
$\{\hat{o}_k^{itr}\}_{k \in K} \leftarrow$ run Algorithm 2 with $\{r_{i,k}\}$ and K_{gold} as input
repeat
  $itr \leftarrow itr + 1$
  select data object d_{itr} as gold object
  obtain expert ratings for d_{itr}
  $K_{gold} \leftarrow K_{gold} \cup d_{itr}$ (add d_{itr} to set of gold objects)
  $\{\hat{o}_k^{itr}\}_{k \in K} \leftarrow$ run Algorithm 2 with $\{r_{i,k}\}$ and K_{gold} as input
until *Eq. stop condition is satisfied* $\vee$ $itr \geq |K|$;

8.1.1 Stop Condition Based on the Output of DSA

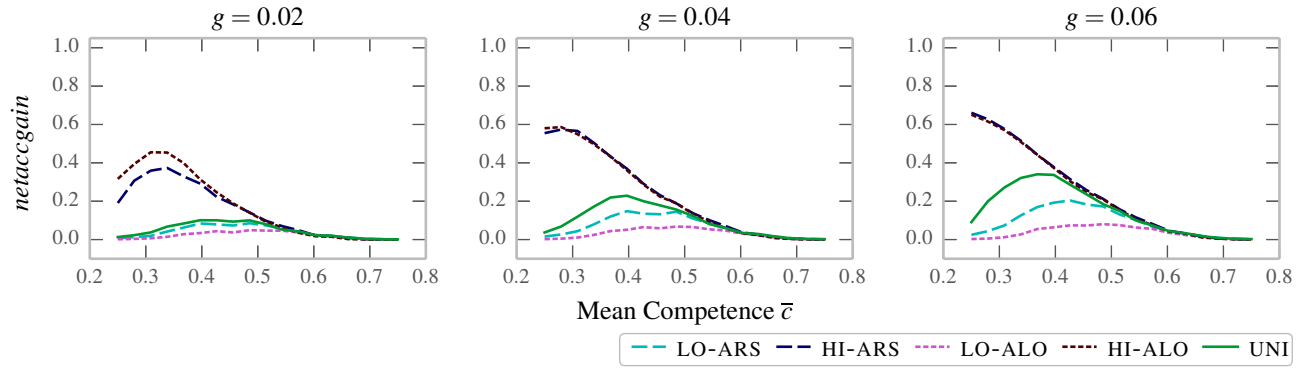
A stop condition that yields good results based on the DSA outputs is not obvious. This is because the changes of the output do not decrease monotonically over the iterations, as one might expect. Instead, they can vary strongly in either direction from one iteration to the next (cf. rightmost plot in Fig. 8). Further, they behave very differently in different simulations. Consider the *docchange* (data object classification change), i.e., the number of data objects whose classification



(a) Setting UNIFORM with 20 raters.



(b) Setting UNIFORM with 50 raters.



(c) Setting SKEWED.

Fig. 7: Net accuracy gains of DSA with different gold strategies using gold ratio g compared to DSA without gold objects ($g = 0$).

has changed in iteration *itr* compared to iteration $itr - 1$ of the adaptive algorithm

$$docchange(itr) = \sum_k \mathbb{1}(\hat{o}_k^{itr-1} \neq \hat{o}_k^{itr}),$$

where $\hat{o}_k^{itr}$ denotes the estimated type of data object k in iteration *itr*, $itr \geq 1$. In the simulation run displayed in the leftmost plot in Fig. 8, the *docchange* does not change much until iteration 63. Adding the 63rd gold object, however, gives DSA enough "knowledge" about the competence of the raters to reverse the classification of the data objects almost completely. This results in a large increase of the *netbenefit* and a large jump of the *docchange*. In the middle plot, the *docchange* is flat in the beginning as well, but there is no reversal of the classification later. In the rightmost plot, *docchange* changes with almost every iteration. The other

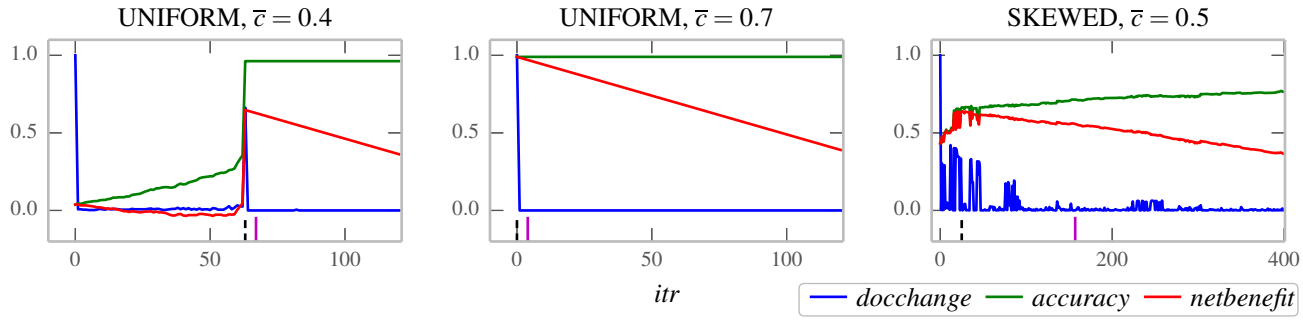


Fig. 8: Behavior of *docchange* (normalized), accuracy, and *netbenefit* over single runs of the adaptive algorithm. The vertical lines show the *itr* of the maximum *netbenefit* (black, dashed), and the *itr* where the stop condition with parameters $st_{width} = 4$, $st_{maxsum} = 2$ stops (magenta).

				mean *ntmnr*	
Setting	Gold Strategy	st_{width}	st_{maxsum}	training	test
UNIFORM	HI-ARS	3	0	0.96	0.95
UNIFORM	HI-ALO	1	0	0.98	0.98
SKEWED	HI-ARS	2	1	0.94	0.96
SKEWED	HI-ALO	2	1	0.89	0.94

Table 1: Combinations of st_{width} and st_{maxsum} that maximize the mean *ntmnr* for different settings and gold strategies of the training dataset.

outputs of DSA, e.g., the change of the competence estimates, behave similarly.

To gain robust results, our stop condition computes the sum of changes of the last st_{width} ("stop width") iterations. Let itr' be the current iteration. The stop condition tests if the sum of the *docchange* values of the previous st_{width} iterations is below the threshold st_{maxsum}

$$\sum_{itr=itr'-st_{width}}^{itr'} docchange(itr) \leq st_{maxsum}. \qquad \text{(stop condition)}$$

8.1.2 Finding Parameters for the Stop Condition

We have created a training dataset that contains the results of more than 1000 simulation runs of the adaptive algorithm to obtain values for st_{width} and st_{maxsum} that maximize the net benefit. For this training dataset, we have varied the rating distribution (uniform, Pareto with different parameters), the mean competence, the number of data objects, the number of raters, and the random seeds. Table 1 shows for each setting the combinations of st_{width} and st_{maxsum} that maximize the mean *netbenefit* over all simulation runs. We consider the HI-strategies only. This is because the LO-strategies and the UNI strategy perform strictly worse with the adaptive algorithm than the HI-strategies. Besides the training dataset, we have applied the adaptive algorithm to the test data we used in the previous evaluations of the gold strategies (cf. Section 7.2). We have computed the "net benefit to maximal net benefit ratio" *ntmnr*, i.e., the ratio of the *netbenefit* achieved by the st_{width}, st_{maxsum} combination and the maximum achievable *netbenefit* for each run of the adaptive algorithm. Table 1 shows the mean *ntmnr* that the st_{width}, st_{maxsum} combination reached in the training dataset and when applied to the test dataset.

Further, the combination $st_{width} = 2$, $st_{maxsum} = 0$ yields *ntmnr* values almost as high as the values identified in Table 1. It is among the top-five combinations in each setting and each strategy tested. Using it might avoid overfitting to some degree, thus giving way to more robust results compared to the maximizing combinations when applied to other datasets.

8.2 Net Benefit Gains of the Adaptive Algorithm

We evaluate the adaptive gold algorithm by comparing its net benefit to the net benefit of the DSA without gold objects. As above, we consider the HI-strategies only. Similarly to Eq. 9, we define the net benefit gain as the normalized difference between the net benefit of the adaptive algorithm $netbenefit^{adaptive}$ and the net benefit of the vanilla DSA without gold $netbenefit^{nogold}$:

$$netbenefitgain = \frac{netbenefit^{adaptive} - netbenefit^{nogold}}{|K|}.$$

We evaluate the adaptive algorithm with the same simulation experiments detailed in Section 7.2. We use the robust parameters values $st_{width} = 2$ and $st_{maxsum} = 0$ for the stop condition of the adaptive algorithm. Figure 9 shows the net benefit gains and the gold ratio used. In general, the adaptive algorithm achieves very high gains for $\bar{c} \leq 5$. The HI-ALO strategy outperforms HI-ARS in the UNIFORM settings, while HI-ARS performs slightly better than HI-ALO in SKEWED. For each strategy in the UNIFORM settings, the adaptive algorithm uses a relatively high gold ratio for the lower competence range $\bar{c} < 0.5$. In this range, it also

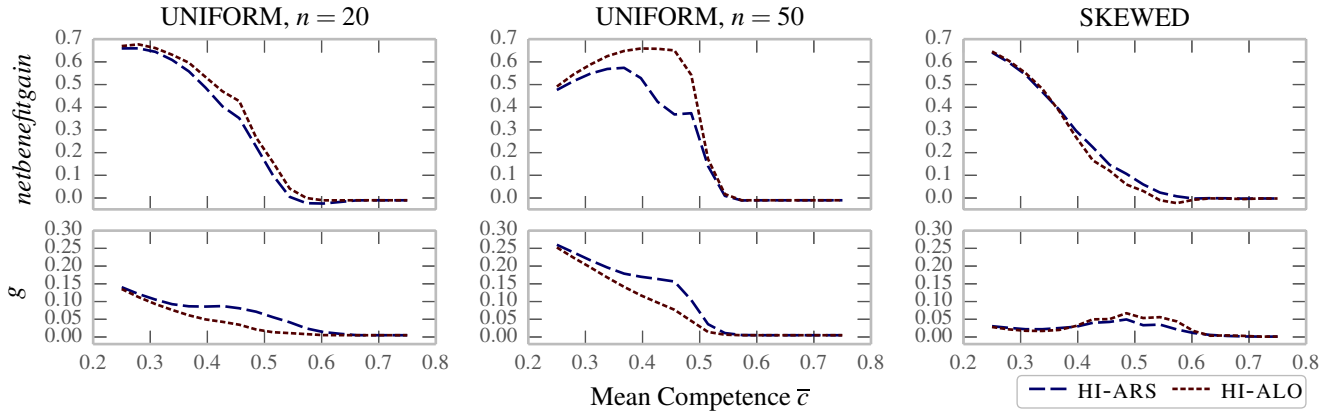


Fig. 9: Net benefit gains and gold ratio g used by adaptive gold algorithm compared to DSA without gold objects. Stop condition with parameters $st_{width} = 2$ and $st_{maxsum} = 0$.

achieves the highest gains. In SKEWED, the adaptive algorithm uses a much lower gold ratio than in UNIFORM. Finally, for $\bar{c} > 0.6$ the gold ratio used by the adaptive algorithm is close to zero in all settings and for both gold strategies. The net gain in this range is zero or slightly negative.

9 Using Gold Strategies to Counter Collusion Attacks against DSA

The previous discussion has focused on improving the accuracy and the net benefit of DSA in the presence of low-competence raters in particular. A potential problem that has a negative impact on the accuracy of DSA as well are collusion attacks. In a collusion attack, raters coordinate to rate the same data objects with the same value to artificially increase their estimated competence. This is beneficial for the colluders if they receive a remuneration for their ratings that is based on their estimated competence. For example, many online communities remunerate users with reputation or Karma points for high-quality contributions. We propose to compute remunerations in such communities contingent on the competence estimates by DSA. Similarly, Wang et al. (2013) propose an algorithm that pays crowdsourcing workers based on the competence estimates calculated by DSA, next to other metrics. (However, as mentioned, they do not address collusion attacks.) In such settings a collusion attack allows colluders to artificially increase their remuneration while saving cognitive effort for determining the truthful value of the data objects. Since DSA assigns an inflated weight to their low-competence ratings colluders can also severely damage the accuracy of DSA.

Gold objects can counter a collusion attack. They allow for more precise competence estimates. Thus, they correct the overly high competence estimates of colluders. This reduces the benefit gained by colluders, thereby making collusions less desirable. It also reduces the damage of collusions on the accuracy of DSA.

9.1 Model of a Collusion Attack

We extend the model of a peer-rating online community introduced in Section 2.

Our model of a collusion attack partitions the set of raters into a set of colluders $I_{col} \subseteq I$ and a set of *honest raters*, i.e., raters that do not collude, $I_{hon} = I \setminus I_{col}$. Colluders coordinate – for example by using the internet as a communication channel – to give the same ratings for each data object in a subset of the data objects. We call the subset of data objects that colluders use for the collusion attack *collusion data objects* and refer to it as K_{col}. For simplicity, we assume that every colluder rates all data objects from the set K_{col} but no other data objects. The set of non-collusion data objects $K_{hon} = K \setminus K_{col}$ is the set of data objects that colluders do not rate. (Honest raters rate objects from both K_{hon} and K_{col}.) Without loss of generality, we assume that colluders always assign ratings with value 1 independent of the true type of the data object in question. I.e., $r_{i,k} = 1$ for all $i \in I_{col}$ and for all $k \in K_{col}$. Other collusion strategies – like coordinating on a rating value per data object or, if the prior is highly skewed, choosing the a priori most likely value – add little to the discussion at hand (we argue).

We use $n_{col} = |I_{col}|$ and $m_{col} = |K_{col}|$ to denote the number of colluders and the number of collusion data objecs, respectively. We define the *ratio of collusion objects* as the ratio of the number of collusion data objects to that of all data objects, i.e., m_{col}/m. Further, we define the *ratio of colluders* as the proportion of colluders among all raters, i.e., n_{col}/n.

To simplify the discussion, we assume type-independent competencies c_i of raters and colluders. We use the *collusion rent* as a metric for the benefit raters gain from colluding. The collusion rent of a colluder i is the difference between his estimated competence and his real competence $\hat{c}_i - c_i$.

Since we assume that all ratings of colluders are 1, their competence equals the prior probability of type 1, that is, $c_i = p(1)$ for all colluders $i \in I_{col}$. In other words, collusion ratings are correct with probability $p(1)$. The definition of the colluders rent implies that the maximum collusion rent is $1 - p(1)$.

9.2 Influence of Colluders and Honest Raters on the Outcome of a Collusion Attack

Honest ratings, i.e., the ratings of honest raters, are a countermeasure against collusions. A high number of honest ratings for a given data object makes it less likely for colluders to influence the classification of the data object.

We conduct a simulation experiment to gain intuition on how the number of honest ratings and the number of collusion ratings influence the outcome of a collusion attack. In particular, we focus on the number of both honest and collusion ratings per collusion data object. In the simulation, we vary three parameters: (1) the number of colluders n_{col}, (2) the number of honest raters n_{hon}, and (3) the rating rate of honest raters rr_{hon}. All three parameters determine the ratio of honest ratings to collusion ratings per collusion data object. The parameter rr_{hon} is not an exogenous parameter of SKEWED, i.e., we can only indirectly manipulate it by changing the parameters of the Pareto rating distribution. This is why we use the setting UNIFORM for this simulation only. Our intention behind the simulation at hand is solely to build intuition. Therefore, we do not deem the omission of SKEWED limiting. We simulate both settings below where we evaluate how gold strategies can reduce collusion attacks. In the simulation, we vary rr_{hon} from 0 to 1 in 0.05 steps and assign each honest rater $i \in I_{hon}$ the same rating rate $rr_i = rr_{hon}$ (see Section 5 for the definition of rr_i). We fix the number of raters to $n = 50$ and vary the ratio of colluders n_{col}/n from 0 to 1 in 0.04 steps. Note that this varies both the number of colluders n_{col} and the number of honest raters n_{hon}. We set the ratio of collusion objects to $m_{col}/m = 0.3$, i.e., colluders coordinate on 30 percent of the data objects. The simulation draws the competence of the honest raters from a uniform distribution on the interval $[0.35, 0.95]$.

Figure 10 shows the average collusion rent (left-hand side) and the error rate of DSA (right-hand side) resulting from a collusion attack. Both collusion rent and error rate are shown as a function of the rating rate of the honest raters rr_{hon} and also of the ratio of colluders among all raters n_{col}/n. As expected, a collusion attack requires more colluders to maximize the collusion rent the higher the rating rate of honest raters.

Once the ratio of collusion ratings to honest ratings (determined by the combination of rr_{hon} and n_{col}/n) reaches a certain threshold, the collusion rent rises sharply. For example, in the left-hand plot, this threshold runs approximately from (0, 0.2) to (1, 0.4) through the rr_{hon}-n_{col}/n plane. Intuitively, if there are enough collusion ratings for a data object $k \in K_{col}$, DSA shifts the type estimate of k in favor of the collusion ratings. Because of this, DSA assigns higher competence estimates and thus higher weights to the colluders. The higher weights increase the influence of the collusion ratings on the other collusion data objects, and so on.

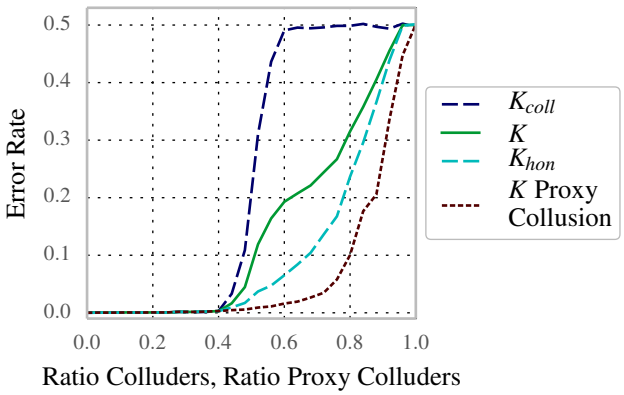


Fig. 11: Error rate of different subsets of data objects for colluders and proxy colluders.

9.2.1 Effect of Collusions on the Error Rate of Non-Collusion Data Objects

The effect of collusions on the error rate is two-fold: they directly affect the error rate of collusion data objects and indirectly affect the error rate of non-collusion data objects. The indirect effect is as follows. Colluders decrease DSA's accuracy of the type estimates of collusion data objects. This in turn lowers DSA's accuracy of the competence estimates of those honest raters who have also rated the collusion data objects. The lowered accuracy of the competence estimates of honest raters decreases the accuracy of non-collusion data objects.

Figure 11 represents the two-dimensional slice of Fig. 10 where the rating rate of honest raters is 1.0. It differentiates between the error rates of three different subsets of data objects – of all data objects (K), of collusion data objects (K_{col}), and of non-collusion data objects (K_{hon}). The error rates are shown as a function of the ratio of colluders.

To visualize the indirect influence of colluders on the error rate of K_{hon} we simulate a *proxy collusion*. For this purpose, we run the simulation experiment of a collusion attack described above again and replace the colluders with *proxy colluders*. Proxy colluders are regular raters, i.e., they do not coordinate. To make them comparable to colluders, they rate the same number of data objects as colluders – but not necessarily from the set K_{hon}. Further, they have the same observed error rate as colluders, i.e., they rate as many

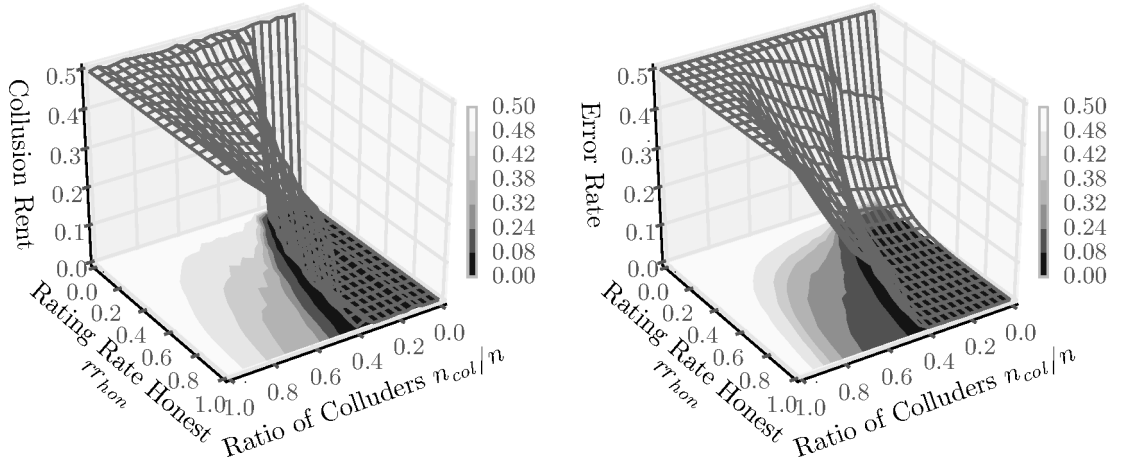


Fig. 10: Average collusion rent and error rate of DSA as functions of the rating rate of the honest raters and the ratio of colluders.

data objects incorrectly as colluders. The other simulation parameters are the same as before.

Figure 11 shows the error rate for all data objects K in a proxy collusion as a function of the ratio of proxy colluders. Note that there are no equivalents to K_{hon} and K_{col} in a proxy collusion. This is because proxy colluders do not coordinate on a subset of K. For the collusion attack, the error rates of K, K_{col}, and K_{hon} are all higher than the error rate of K in a proxy collusion. Just by coordinating, colluders cause higher error rates – even for the data objects K_{hon} they did not rate – than the otherwise identical proxy colluders.

9.3 Reducing the Collusion Rent with Gold Objects

We show that gold objects can reduce the collusion rent and the error rate which result from a collusion attack. As an example, we simulate the setting discussed in the previous section but using five percent randomly chosen data objects as gold objects. This (see results in Fig. 12) reduces the collusion rent to almost zero. It also reduces the error rate significantly compared to the collusion attack without gold objects (cf. Fig. 10).

In the following, we analyze the influence of gold strategies as a means to reduce the collusion rent. The analysis focuses on reducing the collusion rent for the following reason. We assume that raters consider the collusion rent an incentive for colluding. Therefore, a reduced collusion rent makes colluding less desirable and thus should lower the occurrence of collusions.

To investigate the effects of gold strategies on the collusion rent, we conduct several simulation experiments for UNIFORM and SKEWED. In these experiments we vary the gold ratio g and the ratio of colluders n_{col}/n. We fix the other simulation parameters of UNIFORM and SKEWED to their default values defined in Section 5. As in the previous section, we set the ratio of collusion objects m_{col}/m for the setting UNIFORM to 0.3. In setting SKEWED, honest raters rate approximately 2.6 percent of the data objects. This is a much lower rating rate than in setting UNIFORM where honest raters rate 40 percent of the data objects. Accordingly, we adjust the ratio of collusion data objects for SKEWED to a lower value as well and set it to $m_{col}/m = 0.1$, i.e., colluders coordinate on 10 percent of the data objects.

As we have seen in the previous section, the maximum achievable collusion rent depends on the type prior $p(t)$. The type priors of UNIFORM and SKEWED differ. To make the results comparable between the two settings, we calculate the *normalized collusion rent*. The normalized collusion rent for a simulation is the collusion rent divided by the maximum collusion rent observed in that simulation.

Figure 13 shows the normalized collusion rent (0/black: best, 1/white: worst outcome) as a function of the colluders ratio and the gold ratio, for the different gold strategies. For SKEWED (Fig. 13b), HI-ARS pushes the collusion rent to 0 for gold ratios larger than 0.1 independent of the ratio of colluders. HI-ALO reduces the collusion rent to almost 0 for most of the area of the contourplot as well. The LO strategies, on the other hand, can only reduce the collusion rent to close to 0 for very high gold ratios (> 0.95) or very low colluder ratios (< 0.02 and < 0.08 for LO-ARS and LO-ALO, respectively). The strategy UNI performs better than the LO strategies but worse than the HI strategies in this setting.

For UNIFORM the differences between HI and LO strategies are lower but still pronounced (Fig. 13a). Here, HI strategies perform better than LO strategies as well. Colluders in this setting gain rent > 0.05 only if their ratio is larger than 30 percent, independent of the gold strategy. The UNI strategy performs best in this setting. It reduces the collusion rent to 0

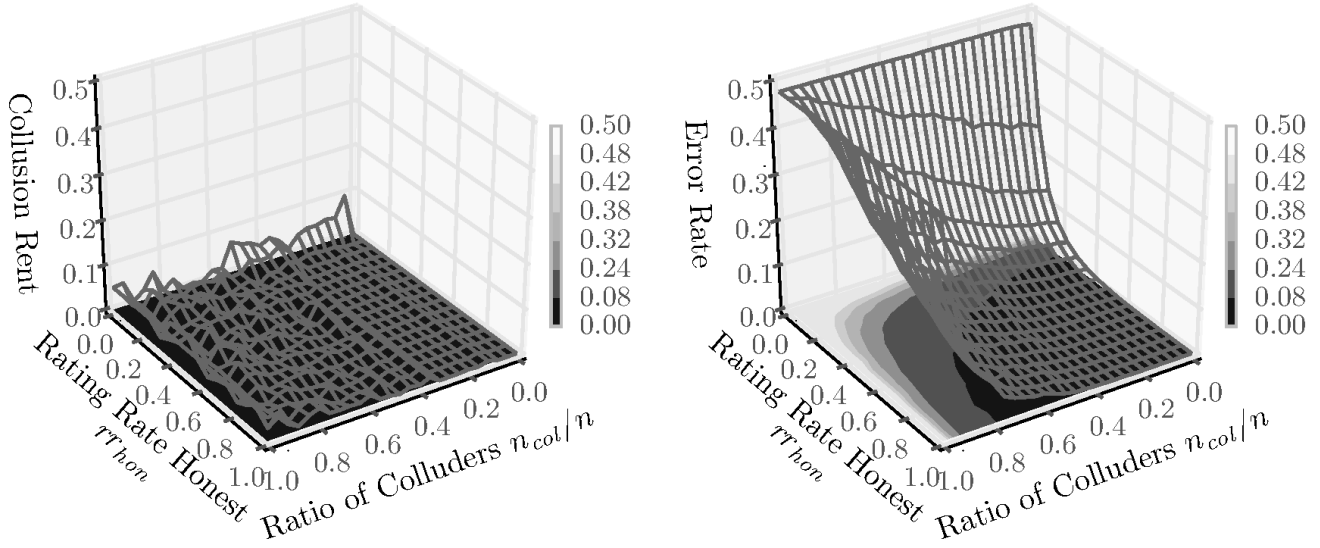


Fig. 12: Colluders rent and the error rate of DSA with five percent gold objects. Otherwise same setting as in Fig. 10.

for gold ratios > 0.05. HI-ARS performs only slightly worse than UNI.

The reason why the HI strategies are so effective in reducing the effects of a collusion attack is that they select data objects as gold objects that have a high level of agreement. A high level of agreement is a property of collusion data objects since all colluders agree on those data objects. I.e., HI strategies are a collusion detection mechanism.

10 Related Work

There exists a large body of literature on the accuracy of MV. The earliest work on the accuracy of MV, the Condorcet jury theorem, dates back to 1785 (Condorcet 1785). Grofman et al. (1983) review results on the accuracy of voting processes as a function of the competences of the individual voters, the decision procedure, and the number of voters. The optimal weights for raters with type-independent competence in binary choice situations (Eq. 6) have been identified several times independently (Minsky and Papert 1969; Duda and Hart 1973; Nitzan and Paroush 1982). Lam and Suen (1997) generalize the Condorcet jury theorem to cases where even numbers of voters are allowed, where the number of choices is greater than two, and thus ties are possible. Kuncheva et al. (2003) derive upper and lower limits on the accuracy of MV for both dependent and independent classifiers.

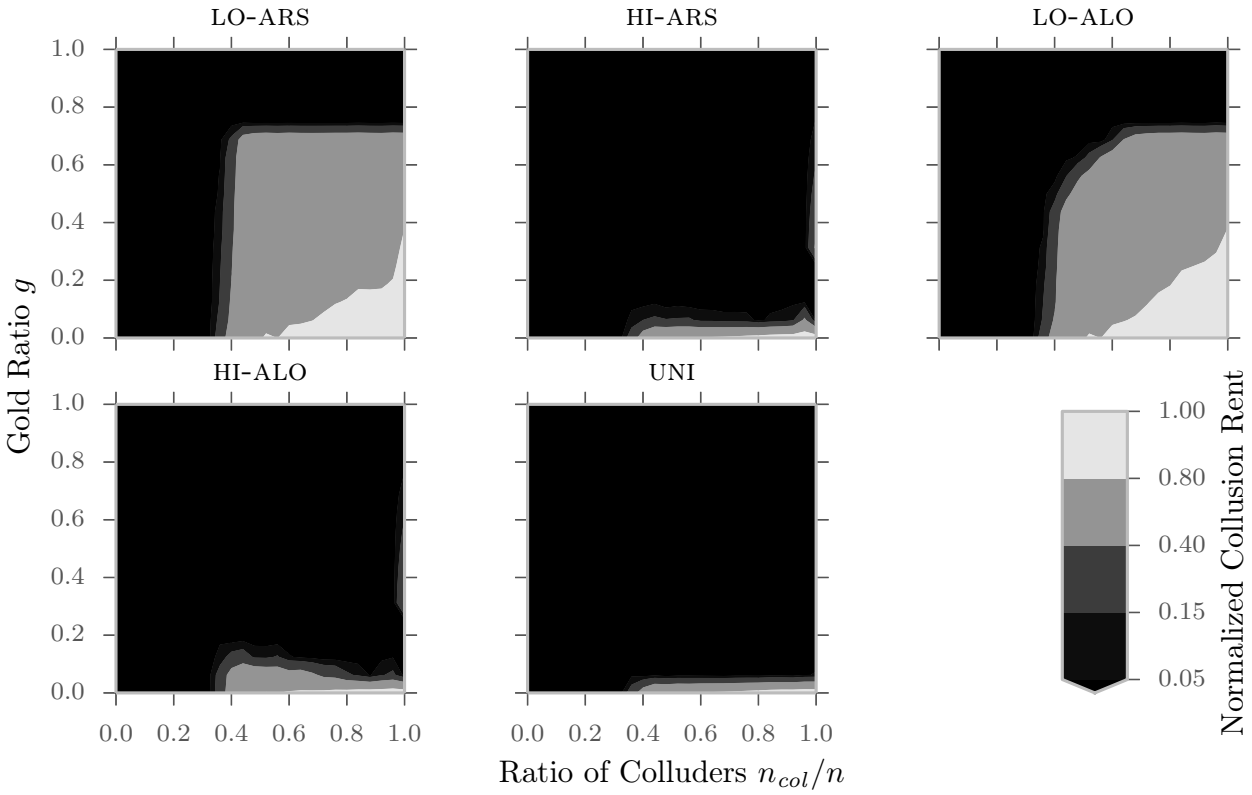
Li et al. (2013) derive theoretical error bounds (approximate and expected) on linear threshold rules, such as MV and WMV, for binary labeling tasks. This includes bounds on the expected error rate of the MAP rule with known competencies.

Dawid and Skene (1979) have developed DSA for the estimation of error rates made by clinicians in the assessment of patients. The algorithm is based on the general algorithm for expectation maximization (EM) developed by Dempster et al. (1977). Whitehill et al. (2009) introduce an EM algorithm similar to DSA that takes varying difficulties between the data objects of the same type into account. They simulate a setting with a high variance of data-object difficulty and find that their algorithm performs better (4.5% error rate) than MV and DSA (11.2%, and 8.4% error rate respectively).

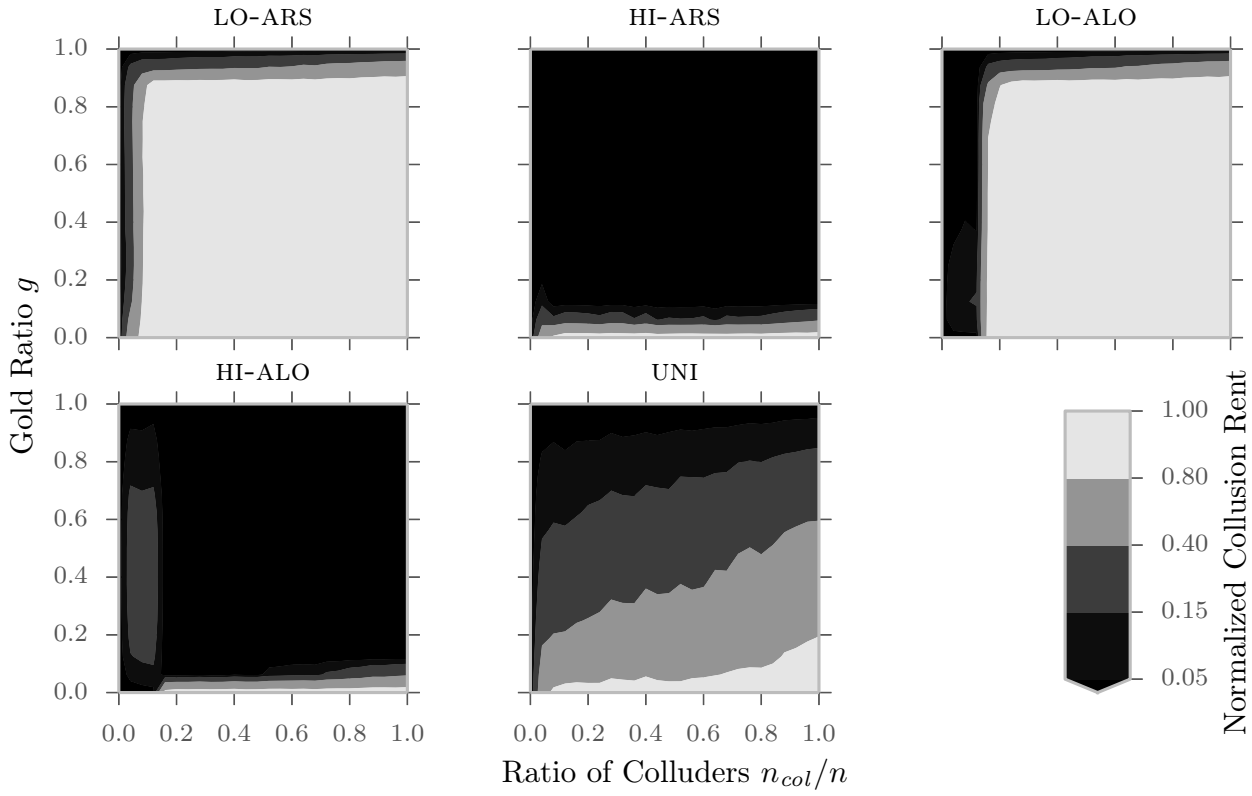
Snow et al. (2008) discuss an experiment for the estimation accuracies of AMT workers for word annotation tasks. Similarly to DSA, their method estimates the worker competencies based on comparisons with gold standard examples, and uses a MAP estimate to infer the annotation quality.

Ipeirotis et al. (2010) develop an algorithm based on so-called soft labels and expected costs of each soft label to differentiate between biased raters ($c < 0.5$) and spammers ($c = 0.5$). The algorithm uses DSA to estimate rater competencies. In an experiment, the algorithm performs better at detecting those spammers that always rate the class with the highest type prior. This leads to a higher accuracy (0.998) compared to DSA (0.95). The algorithm assumes that DSA returns "some reasonably accurate estimates" of the competencies. However, this assumption does not hold true for $\bar{c} < 0.5$, as our results show.

Raykar and Yu (2012) propose an algorithm to eliminate spammers and malicious raters. They compare their algorithm to MV and DSA and find that it has a better area under the ROC curve than MV and DSA, in particular if the fraction of spammers is high. Further, they find that their algorithm is better at detecting spammers than MV and DSA. However,



(a) Setting UNIFORM with ratio of collusion objects 0.3.



(b) Setting SKEWED with ratio of collusion objects 0.1.

Fig. 13: Collusion rent for different gold strategies as a function of the ratio of colluders and the gold ratio.

their algorithm (implicitly) assumes that the mean competence $\bar{c}$ is higher than 0.5. For example they state that "the methods proposed in Dawid and Skene [. . .] can automatically flip the labels for the malicious annotators", which is clearly not true if $\bar{c} < 0.5$, as our results show. Further, it is clear from the description of their simulation experiments that the settings they study to show the effects of their method have a mean competence greater than 0.5.

Wang et al. (2011) propose an integration of gold objects into DSA. Further, they develop an algorithm that integrates gold objects in a setting with Amazon Mechanical Turk workers, as opposed to an open community setting like ours. Their method tells AMT workers to label a priori created gold objects based on the expected utility of additional ratings by these workers. However, forcing members of the community to rate particular contributions is not possible in our setting.

11 Discussion

To discuss the implications of our findings, we distinguish between four situations where a community of raters classifies data objects.

The first situation is a typical crowdsourcing setting with payments such as Amazon Mechanical Turk that lets the employer decide which data objects a crowd worker has to rate. There, the method proposed by Wang et al. (2011) might allow using gold objects efficiently based on the expected utility of additional ratings issued by the crowd worker.

The second and the third situation occur in an open community setting where it is not possible to force raters to rate specific data objects. Here, we have to differentiate between communities with a high probability that the mean competence is greater than 0.5 and those with the risk of having a mean competence less than 0.5. If there is a high probability that the mean competence is greater than 0.5, DSA will classify data objects with high accuracy. Our results show that gold objects cannot improve this situation much.

If, on the other hand, the open community faces the risk that its mean competence is near 0.5 or lower, DSA or related methods will not suffice. Such a low mean competence occurs whenever the community is afflicted by a large fraction of low-competence raters, such as spammers, biased raters, malicious raters, and raters with a consistent misunderstanding. In that case, the gold strategies we have proposed can increase the classification accuracy considerably, compared to the vanilla DSA without gold objects. Here, the preferred gold strategies are HI-ALO and, to a lesser degree, HI-ARS. The accuracy gains achieved by a given gold ratio depend on the characteristics of the community. Roughly, the more homogeneous the community, the higher the rating rate, and the lower the competence, the higher the gold ratio required to achieve high accuracy gains compared to the vanilla DSA. In case of a typical open online community with sparse ratings and a skewed rating distribution, a gold ratio as low as two percent can be enough to offset the impact even of a large number of spammers and biased raters. Further, if we also take the costs of gold objects into account, the adaptive gold algorithm can determine the optimal gold ratio with high accuracy.

As the fourth situation, communities face the risk of collusion attacks, in particular if raters are remunerated based on their competence inferred by DSA. Here, gold strategies can successfully counter collusion attacks. HI-ARS is particularly effective against collusion attacks, both in homogeneous settings and in settings with a skewed rating rate that is typical for open communities.

In summary, HI-ARS and HI-ALO can safeguard against the risk of both the impact of (i) a large fraction of low-competence raters, and (ii) of collusion attacks. In such circumstances, a low ratio of gold objects together with the HI-ALO and HI-ARS strategies can be much more effective than simply selecting gold objects randomly. Further, the optimal gold ratio does not need to be guessed but can be determined with high accuracy by our adaptive gold algorithm.

12 Conclusion

In this paper we have analyzed the problem of classifying contributions in an open peer-rating online community. We have discussed the accuracy of majority voting schemes under homogeneous competencies and known heterogeneous competencies. We have analyzed the estimation quality of DSA in various settings. We find that in a homogeneous setting, where raters have roughly the same, relatively high, rating rate, DSA stabilizes after it has classified a relatively low number of data objects. In a more open setting, on the other hand, with a low rating rate and a skewed rating distribution that is typical for open internet communities, DSA requires a much higher number of data objects to stabilize. Further, we find that for a mean competence higher/lower than 0.5, DSA performs better/worse than majority vote. This effect becomes more pronounced, the more widespread the competence distribution is, and the higher the number of raters is.

We have proposed and tested gold strategies based on the level of agreement to increase the accuracy of DSA in low-competence settings. Further, we have proposed and evaluated an adaptive algorithm to maximize the net benefit of gold objects. Finally, we have discussed the damage done by collusion attacks against DSA and have tested how gold strategies can reduce this damage. A main finding of ours is that the HI-ALO gold strategy is very effective in increasing the accuracy of DSA in low-competence settings. Further, the adaptive algorithm determines the optimal gold ratio for each strategy and each setting with high accuracy. Finally, we find

that the HI-ARS and the UNI strategy are effective in reducing the benefit gained by colluders. Thus, they render collusions less attractive for raters. We have discussed the implication of these findings and have found that HI-ALO and HI-ARS can effectively safeguard typical open online communities against the risk of both the impact of (i) a large fraction of low-competence raters, and (ii) of collusion attacks. In such circumstances, a low ratio of gold objects together with the HI-ALO or HI-ARS strategy can be much more effective than simply selecting gold objects at random.

We have focused on a binary setting. However, the methods we have discussed are also applicable to multi-type settings, i.e., $|T| > 2$. In particular the gold strategies can be readily modified to work in multi-type settings. In this case, ars_k would have to be computed for each type t of a given data object individually. For this computation each rating needs to be encoded as 1 if it is in favor of t, as -1 otherwise. The modification of strategies based on alo_k is straightforward as well: instead of using alo_k the modified strategies select data objects that have the highest/lowest MAP estimate for any of their types.

13 Proof of Proposition 1

Proof For simplicity and brevity, we omit the subscript k in the following. Thus, s_k becomes s, R_k becomes R, o_k becomes o, $r_{i,k}$ becomes r_i, and so on. Further, let i refer to the s raters of the data object in question.

The posterior probability that the object is of type t given the ratings is

$$P(o = t \mid R) = \frac{P(R \mid o = t)P(o = t)}{P(R)}.$$

Since $P(R)$ is the same for both types, the ratio of the posterior probabilities (or posterior probability ratio, *ppr*) of the data object being of type -1 and being of type 1 given its ratings is

$$ppr = \frac{P(o = -1 \mid R)}{P(o = 1 \mid R)} = \frac{P(o = -1)}{P(o = 1)} \frac{P(R \mid o = -1)}{P(R \mid o = 1)}.$$

Since we assume conditional independence of the ratings given a type, their joint probability can be written as a product

$$ppr = \frac{P(o = -1)}{P(o = 1)} \prod_{i=1}^{s} \frac{P(r_i \mid o = -1)}{P(r_i \mid o = 1)}. \tag{10}$$

Per definition of the homogeneous competence we have $P(r_i = q \mid o = t) = c$, if $q = t$, and $1 - c$ otherwise. Thus, we can express each *likelihood ratio* as

$$\frac{P(r_i = q \mid o = -1)}{P(r_i = q \mid o = 1)} = \begin{cases} c/(1-c) & \text{if } q = -1 \\ (1-c)/c & \text{if } q = 1. \end{cases}$$

Let $lrr = \prod_{i=1}^{s} P(r_i \mid o = -1)/P(r_i \mid o = 1)$ denote the product of the likelihood ratios of all s ratings. Since s is odd and $c > 0.5$ and therefore $c/(1-c) > 1$ the likehood ratio *llr* cannot be equal to 1. Instead either

- $lrr \geq (c)/(1-c) > 1$, if more ratings are in favor for type -1, or
- $lrr \leq (1-c)/c < 1$, if more ratings are in favor for type 1.

The same is true for the *ppr*. This is because we assume that $1 - c < P(o = t) < c$ for both types $t \in \{-1, 1\}$. Thus, the ratio of the priors is $(1-c)/c < P(o = -1)/P(o = 1) < c/(1-c)$ and therefore cannot "reverse the direction" of the *ppr*: If $lrr > 1$, then $ppr = lrr \cdot P(o = -1)/P(o = 1) > 1$. Likewise, if $lrr < 1$, then $ppr = lrr \cdot P(o = -1)/P(o = 1) < 1$.

Thus, the posterior probability is greater for the type that receives the majority of ratings in its favor. □

14 Proof of Proposition 2

Proof For simplicity and brevity, we omit the subscript k in the following. Thus, s_k becomes s, l_k becomes l, ars_k becomes *ars*, and so on.

Let $corr(\hat{o} = o)$ denote the number of correct ratings for a correct classification by MV. Correspondingly, let $corr(\hat{o} \neq o)$ denote the number of correct ratings for an incorrect classification by MV. We first proof the following lemma.

Lemma 1 *For a given ars and a given odd s, if the number of correct ratings for an incorrect classification by MV is $corr(\hat{o} \neq o) = l$, then the number of correct ratings for a correct classification by MV is $corr(\hat{o} = o) = s - l$. Further, $l = (s - ars)/2$.*

Proof Let *rs* denote the rating sum for a given *ars*, i.e.,

$$ars = |rs| = \begin{cases} -rs & \text{if } rs < 0 \\ rs & \text{if } rs > 0 \end{cases}. \tag{11}$$

Note that since the number of ratings is odd, *rs* cannot be 0. Let rs^+ denote positive rating sum, i.e., the rating sum $rs > 0$ in Eq. 11. Further, let s^- be the number of negative ratings of rs^+, that is, the number of ratings that equal -1 if $rs > 0$. Thus, the number of negative ratings for rs^+ is $(s - s^-)$. The positive rating sum is the sum of the negative ratings and the positive ratings

$$rs^+ = -s^- + (s - s^-) \tag{12}$$

with the number of negative ratings of rs^+ being the smaller of the two summands $0 \leq s^- < s - s^-$.

Note, that there is at most one s^- for a given rs^+ and a given s. To see why this is true, suppose, for contradiction, there is a second $s^- := s^- + k$, for some integer $k \neq 0$. Then,

the number of positive ratings for rs^+ is $(s-(s^-+k))$. But the sum of the number of negative ratings and the number of positive ratings must be $s = s^- + k + (s-(s^-+k))$ which can only be true if $k = 0$.

The negative rating sum rs^-, i.e., the $rs < 0$, is

$$\begin{aligned} rs^- &= -rs^+ \\ &= -(s-s^-) + s^-. \end{aligned} \tag{13}$$

Hence, s^- is the number of positive ratings of rs^- and $(s-s^-)$ is the number of negative ratings of rs^-.

For a given *ars* and a given s, let $corr(\hat{o} \neq o) = l$ be the number of correct ratings for an incorrect classification. For incorrect classification by MV, the number of correct ratings must be less than the number of incorrect ratings. Thus, l must be the smaller one of the two terms s^- and $(s-s^-)$, i.e., $l = s^-$. For the correct classification by MV, the number of correct ratings must be the larger of the two terms, i.e., $corr(\hat{o} = o) = s - s^- = s - l$.

To prove the second part of Lemma 1, we use the relationship between *ars* and rs^- and rs^+ defined in Eq. 11. We substitute $s^+ = s - l$ and $s - s^+ = l$ in Eq. 12

$$\begin{aligned} ars = rs^+ &= -l + (s-l) \\ &= s - 2l \end{aligned}$$

and in Eq. 13

$$\begin{aligned} ars = -rs^- &= -(-(s-l)+l) \\ &= s - 2l. \end{aligned}$$

Rearranging, either one of the above equations proves the second part of the lemma: $l = (s - ars)/2$.

The probability of an incorrect classification by MV given *ars* is

$$P(\hat{o} \neq o \mid ars) = \frac{P(\hat{o} \neq o, ars)}{P(ars)} = \frac{P(\hat{o} \neq o, ars)}{P(\hat{o} \neq o, ars) + P(\hat{o} = o, ars)}. \tag{14}$$

As Lemma 1 shows, to obtain a given *ars* out of s ratings, we need exactly $l = (s - ars)/2$ correct ratings. The probability that exactly l out of s ratings are correct for a given competence c, is $\binom{s}{l} c^l (1-c)^{(s-l)}$ (see Eq. 2). Thus, the joint probability of MV classifying the data object incorrectly and having rating sum *ars* is

$$P(\hat{o} \neq o, ars) = \binom{s}{l} c^l (1-c)^{s-l}.$$

where $l = (s - ars)/2$. By Lemma 1, the joint probability of MV classifying the data object correctly and having rating sum *ars* is

$$P(\hat{o} = o, ars) = \binom{s}{s-l} c^{s-l} (1-c)^l.$$

Substituting the two equations above in Eq. 14, we obtain

$$P(\hat{o} \neq o \mid ars) = \frac{\binom{s}{l} c^l (1-c)^{s-l}}{\binom{s}{l} c^l (1-c)^{s-l} + \binom{s}{s-l} c^{s-l} (1-c)^l}.$$

$$P(\hat{o} \neq o \mid ars) = \frac{c^l (1-c)^{s-l}}{c^l (1-c)^{s-l} + c^{s-l} (1-c)^l},$$

where $l = (s - ars)/2$.

15 Summary of Notation

Table 2 summarizes the most frequently used symbols of this paper and their meanings.

References

Inmaculada B Aban, Mark M Meerschaert, and Anna K Panorska. Parameter estimation for the truncated Pareto distribution. *Journal of the American Statistical Association*, 101(473):270–277, 2006. doi: 10.1198/016214505000000411. URL `http://amstat.tandfonline.com/doi/abs/10.1198/016214505000000411`.

Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51 (4):661–703, November 2009. ISSN 0036-1445. doi: 10.1137/070710111. URL `http://dx.doi.org/10.1137/070710111`.

Jean-Antoine-Nicolas de Caritat Marquis de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie royale, Paris, 1785. URL `http://gallica.bnf.fr/ark:/12148/bpt6k417181`.

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1): 1–38, 1977.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc, 1 edition, February 1973. ISBN 0471223611.

Bernard Grofman, Guillermo Owen, and Scott L. Feld. Thirteen theorems in search of the truth. *Theory and Decision*, 15(3):261–278, 1983. ISSN 0040-5833. doi: 10.1007/BF00125672. URL `http://dx.doi.org/10.1007/BF00125672`.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, New York, NY,

USA, 2010. ACM. ISBN 978-1-4503-0222-7. doi: 10.1145/1837885.1837906. URL http://doi.acm.org/10.1145/1837885.1837906.

Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval*, 16 (2):138–178, 2013.

Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004. ISBN 0471210781.

Ludmila I. Kuncheva, Christopher J. Whitaker, and Catherine A. Shipp. Limits on the majority vote accuracy in classifier fusion. *Pattern Anal. Appl.*, 6(1):22–31, 2003.

Louisa Lam and C.Y. Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 27(5):553–568, 1997. ISSN 1083-4427. doi: 10.1109/3468.618255.

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

Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2857–2866, New York, NY, USA, 2011. ACM.

Raghu Meka, Prateek Jain, and Inderjit S Dhillon. Matrix completion from power-law distributed samples. In *Advances in Neural Information Processing Systems*, pages 1258–1266, 2009.

Marvin L. Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

Shmuel Nitzan and Jacob Paroush. Optimal decision rules in uncertain dichotomous choice situations. *International Economic Review*, 23:289–297, 1982. URL http://www.jstor.org/stable/2526438.

Vikas C Raykar and Shipeng Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13(2), 2012.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1613715.1613751.

Jing Wang, Panagiotis G Ipeirotis, and Foster Provost. Managing crowdsourcing workers. In *The 2011 Winter Conference on Business Intelligence*, pages 10–12, 2011.

Jing Wang, Panagiotis G. Ipeirotis, and Foster Provost. Quality-based pricing for crowdsourced workers, 2013. URL http://hdl.handle.net/2451/31833. NYU-CBA Working Paper CBA-13-06.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.

Symbol	Meaning
$T = \{-1, 1\}$	binary set of types;
$q, t \in T$	type q and type t;
$p(t)$	prior probability of type t;
$K = \{1, \ldots, m\}$	set of data objects;
$k \in K$	data object;
o_k	true type of k;
$m = \|K\|$	number of data objects;
I	set of raters $I = \{1, \ldots, n\}$;
i	rater i;
$n = \|I\|$	number raters;
$r_{i,k}$	rating that rater i gives to data object k;
$R = \{r_{i,k}\}$	set of all ratings;
$R_k = \{r_{i,k'} \mid k' = k\}$	the set of ratings of data object k;
$s_k = \|R_k\|$	number of ratings given to data object k;
l_k	number of correct ratings for k;
$c_i^{(t)}$	type dependent competence, i.e., probability $P(r_{i,k} = t \mid o_k = t)$ that rater i rates objects of type t correctly;
c_i	type-independent competence, i.e., probability that rater i rates correctly;
c	homogeneous competence, i.e., competence $c = c_i$ that is the same for all raters i;
g	gold ratio, i.e., ratio of gold objects;
K_{gold}	set of gold objects;
$m_{gold} = \|Kgold\|$	number of gold objects;
$I_{col} \subseteq I$	the subset of colluders among all raters;
$K_{col} \subseteq K$	the subset of data objects that the colluders use for the collusion attack;
$K_{hon} = K \setminus K_{col}$	the set of data objects rated by honest raters only;
$n_{col} = \|I_{col}\|$	number of colluders;
$m_{col} = \|K_{col}\|$	number of collusion objects;
m_{col}/m	ratio of collusion objects;
$\hat{\theta}$	an estimator of a given parameter θ;
$\mathbb{1}(\cdot)$	indicator function, i.e., $\mathbb{1}(\cdot)$ is equal to one if its argument holds true, and equal to zero otherwise;

Table 2: Symbols and meanings.