Semi-Verified PAC Learning from the Crowd

Shiwei Zeng
Stevens Institute of Technology

Jie Shen
Stevens Institute of Technology

Abstract

We study the problem of *crowdsourced PAC learning* of threshold functions. This is a challenging problem and only recently have query-efficient algorithms been established under the assumption that a noticeable fraction of the workers are *perfect*. In this work, we investigate a more challenging case where the *majority* may behave adversarially and the rest behave as the Massart noise – a significant generalization of the perfectness assumption. We show that under the semi-verified model of Charikar et al. (2017), where we have (limited) access to a trusted oracle who always returns correct annotations, it is possible to PAC learn the underlying hypothesis class with a manageable amount of label queries. Moreover, we show that the labeling cost can be drastically mitigated via the more easily obtained comparison queries. Orthogonal to recent developments in semi-verified or list-decodable learning that crucially rely on data distributional assumptions, our PAC guarantee holds by exploring the wisdom of the crowd.

1 INTRODUCTION

Efficient and robust learning of threshold functions is arguably one of the most important problems in machine learning, and has been broadly investigated under different label noise models, such as the random classification noise (Angluin and Laird, 1987), Massart noise (Sloan, 1988; Massart and Nédélec, 2006), Tsybakov noise (Tsybakov, 2004), and agnostic noise (Haussler, 1992; Kearns et al., 1992). In recent years, under certain distributional assumptions on the unlabeled data, a variety of computationally efficient algorithms have been established (Awasthi et al., 2017a; Zhang et al., 2020; Diakonikolas et al., 2020c,b) under the *standard* probably approximately correct (PAC) learning model of Valiant (1984), where there is a *single* adversary who generates all the noisy labels.[1]

However, the standard PAC learning model of Valiant (1984) does not fully characterize the process of data collection and learning in modern machine learning applications. In fact, when constructing a large-scale data set, researchers often appeal to a crowdsourcing platform to hire *multiple* crowd workers for annotation, with the hope of obtaining a set of high-quality labels. To be more concrete, a crucial feature of *crowdsourced learning* is that for each instance x, the learner has a discretion to collect a set of k labels, denoted by $y_i(x), i = 1, \ldots, k$, from a large pool of heterogeneous crowd workers. Here, each single worker in the pool may behave as a certain type of label noise models in the standard PAC model, and some workers can even collude with others to decide which instances to corrupt among those assigned to them. A common practice to address such noisy annotations is to aggregate the labels via majority vote, which turns out to be a successful remedy provided that the majority are correct (Dekel and Shamir, 2009; Vaughan, 2017; Awasthi et al., 2017b; Zeng and Shen, 2022a).

In this paper, nevertheless, we consider learning from an extremely noisy pool of workers, where the majority might be adversarial, under which using majority voting will make the result even noisier, and more seriously, rendering PAC learning impossible. This problem, in addition to being theoretically interesting and has been broadly studied under the non-crowdsourcing setting (Charikar et al., 2017), also finds a variety of real-world applications (Steinhardt et al., 2016; Prelec et al., 2017; Meister and Valiant, 2018). For example, it is observed that when confronted with the question "is Philadelphia the capital of Pennsylvania", more than 60% of the respondents endorse the incorrect answer with high confidence (Prelec et al., 2017). This raises a pressing question:

> If the majority of the crowd might be malicious, can we still achieve the PAC guarantees? If the answer is positive, what is the price for it?

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

[1] In the standard PAC learning model, the label of an instance is always persistent.

1.1 Warmup: Reduction to Standard PAC Learning

We begin our exploration by considering a quite fundamental problem setup: the underlying hypothesis class is the one of homogeneous halfspaces among which there exists a hypothesis that correctly classifies all instances (Rosenblatt, 1958; Valiant, 1984). For each instance we assign one worker for labeling.

Formally, let $\mathcal{X} \subset \mathbb{R}^d$ be the instance space, $\mathcal{Y} = \{-1, 1\}$ be the label space, and D be the marginal distribution on $\mathcal{X}$. The class of homogeneous halfspaces is given by $\mathcal{H}_{\text{hs}} = \{h : x \mapsto \text{sign}(w \cdot x),\ w \in \mathbb{R}^d\}$, and the true label of x is given by $h^*(x)$ for some $h^* \in \mathcal{H}_{\text{hs}}$. The error rate of any hypothesis h is defined as $\text{err}_D(h) = \Pr_{x\sim D}\big(h(x) \neq h^*(x)\big)$. The learner has access to a large pool of crowd workers who provide labels, and we denote by $\mathcal{P}^L$ the uniform distribution on it. Given $x \in \mathcal{X}$, the quantity $\eta_t(x) := \Pr(h_t(x) \neq h^*(x) \mid x)$ characterizes how likely the learner gets an incorrect label when having worker t annotate the instance x, where the probability is taken over the internal random bits of the worker.

To further simplify the problem, we assume that a $(1-\alpha)$-fraction of workers may be adversarial but the remaining α-fraction are *perfect* in the sense that they always provide correct labels across the domain.

Assumption 1. With probability $1-\alpha$ over the draw of a worker t, t is adversarial, i.e. $\eta_t(x)$ can be an arbitrary function; otherwise, t is perfect, i.e. $\eta_t(x) \equiv 0, \forall x \in \mathcal{X}$.

We aim to approach the problem by reducing it to some well-studied standard PAC learning model, such that there exists a learner with non-trivial PAC guarantees. One promising solution is the *agnostic learner*, which works under the scenario where an adversary can corrupt an arbitrary fraction of the labels in an adversarial manner (Haussler, 1992; Kearns et al., 1992; Kalai et al., 2008; Diakonikolas et al., 2021). We give the following theorem using results from Kalai et al. (2008); Diakonikolas et al. (2021).

Theorem 1. *Suppose Assumption 1 is satisfied, and further assume the maginal distribution D is standard Gaussian. Given a set of $\Theta\big(d^{\text{poly}(1/\epsilon)}\big)$ independent instances from D, each of which is labeled by one worker, there exists an algorithm that runs in time $O\big((d/\epsilon)^{\text{poly}(1/\epsilon)}\big)$ and learns $\mathcal{H}_{\text{hs}}$ by returning a hypothesis with error rate $\leq (1-\alpha)+\epsilon$.*

Remark 1. There are a few limitations of such method. First, the error rate is non-vanishing (as $\alpha < 1/2$), though this is the best one can hope for without extra information. Second, such reduction scheme only works for the very special case of learning homogeneous halfspaces with respect to Gaussian marginals, and with quite high computational complexity and sample complexity. Last, the success of the reduction hinges on the strong assumption that an α fraction of the workers are perfect.

In this paper, we show that it is possible to break all of the above barriers by leveraging the wisdom of the crowd and by querying a trusted oracle with a small number of times.

1.2 Main Results

Observe that using a single worker to annotate an instance is not a common practice. Thus, from now on, we consider the more realistic scenario where the learner would assign an instance to multiple workers. An immediate caveat in our setting is that, the majority vote becomes even noisier. As a remedy, we follow a recent research line of Steinhardt et al. (2016); Awasthi et al. (2017b); Meister and Valiant (2018) and assume that there exists a *trusted oracle* $\mathcal{O}_{\mathcal{T}}$ who always provides the correct annotation for any instance that we query on, and we call a query to $\mathcal{O}_{\mathcal{T}}$ as a verified query. To make it practical, we must restrict the number of verified queries the learner can make. We now consider a more general hypothesis class $\mathcal{H}$ of threshold functions, i.e. $\mathcal{H} = \{h : x \mapsto \text{sign}\big(f(x)\big), f \in \mathcal{F}\}$ where $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$; e.g. $f(x) = w \cdot x$ recovers the class of linear functions and hence $\mathcal{H}$ is the class of halfspaces. Note that the target classifier h^* can be written as $h^*(x) = \text{sign}\big(f^*(x)\big)$ for some $f^* \in \mathcal{F}$. We also drop Assumption 1 on the existence of perfect workers, and instead assume the following:

Assumption 2. With probability $1-\alpha$ over the draw of a worker t, t is adversarial, i.e. $\eta_t(x)$ can be an arbitrary function; otherwise, t is Massart, i.e. $\eta_t(x) \leq \eta, \forall x \in \mathcal{X}$ for some noise rate $\eta \in [0, 1/2)$.

The non-adversarial workers now act as the Massart noise (Sloan, 1988; Massart and Nédélec, 2006) which is a challenging semi-random model that has attracted a surge of recent research interests under the standard PAC model; see e.g. Diakonikolas et al. (2019); Zhang et al. (2020); Diakonikolas et al. (2020c). Our model on the crowd workers is thus a mixture of adversarial and semi-random, with the adversarial consisting of the majority. The model itself is already new and challenging; to the best of our knowledge, no known results have been established under neither standard nor crowdsourced setting.

Now the principle of our algorithmic design is three-fold: 1) offering a PAC guarantee, i.e. finding a hypothesis h such that $\text{err}_D(h) \leq \epsilon$; 2) the number of verified queries, m_V, is few; and 3) the number of queries to the crowd, m_L, is moderate. In particular, we hope that the overhead of the algorithm, Λ_L, defined as the averaged number of crowd queries on a given instance, behaves as a constant.

We note that the last (implicit) condition is that the hypothesis class $\mathcal{H}$ has finite VC-dimension d, which is a minimum requirement for PAC learnability even without noise (Kearns and Vazirani, 1994; Anthony and Bartlett, 1999).

Assumption 3. There exists a computationally efficient algorithm $\mathcal{A}_{\mathcal{H}}$ satisfying the following: for any $\epsilon, \delta \in (0, 1)$, $\mathcal{A}_{\mathcal{H}}$ draws $m_{\epsilon,\delta}$ *correctly labeled instances* and learns $\mathcal{H}$ by

returning a hypothesis h such that with probability $1-\delta$, $\mathrm{err}_D(h) \le \epsilon$. We call $\mathcal{A}_{\mathcal{H}}$ a *realizable PAC learner*.

Remark 2. It is known that for many interesting hypothesis classes, such as the one of polynomial threshold functions, such a realizable PAC learner does exist. In our algorithm and analysis, we will make use of the well-known fact that it suffices to pick

$$m_{\epsilon,\delta} = K \cdot \frac{1}{\epsilon} \cdot \Big(d \log \frac{1}{\epsilon} + \log \frac{1}{\delta} \Big), \tag{1.1}$$

where $K \ge 1$ is some absolute constant and d is the VC-dimension of the class $\mathcal{H}$ (Kearns and Vazirani, 1994; Anthony and Bartlett, 1999).

Theorem 2 (Theorem 4, informal)**.** *Suppose that Assumptions 2 and 3 are satisfied, and $\eta < \frac{\alpha}{16}$. There exists a polynomial-time algorithm (Algorithm 1) that PAC learns $\mathcal{H}$. In addition, $\Lambda_L = O_\alpha(1)$ and $m_V = O_\alpha(1)$.*

Remark 3. The above result addresses all the shortcomings of the naive reduction approach (see Remark 1). Moreover, when the fraction of adversarial workers is a large constant, say 80%, our algorithm still succeeds, and more importantly, only makes a constant number of verified queries to the trusted oracle as well as a constant overhead – as to be clear later, a constant overhead means the total number of label queries to the crowd is only a constant multiple of the one needed in the noise-free PAC model of Assumption 3. Roughly speaking, we give an algorithm that almost does not suffer any performance loss from the extremely noisy crowd.

Remark 4. The most relevant prior work is Awasthi et al. (2017b), where they also studied crowdsourced PAC learning in the presence of large fraction of adversarial workers. Yet, their analysis crucially relies on the existence of an α fraction perfect workers (i.e. Assumption 1). When $\alpha > 1/2$, it is possible to relax their perfectness assumption to the Massart noise model as we have done. However, the case $\alpha \le 1/2$ is more subtle and requires additional treatments (and this is one of our technical contributions).

On top of Theorem 2, we address the problem of learning with pairwise comparisons, where a worker is asked questions of the form "given two instances, which one is more likely to be positive". The motivation is that on a crowdsourcing platform, experimenters often make a same payment per annotation, and pairwise comparison queries are relatively easier to respond than label queries in many applications, thus may ease the process of data acquisition. For example, there are restaurants that people feel neither like or dislike, but comparing the preference to two restaurants might be easy. Another example is about medical diagnosis: determining whether a patient needs intensive care requires evaluation from specialists, but comparing the health status of two patients may be carried out by medical residents (Park et al., 2015; Kane et al., 2017; Xu et al., 2017).

Formally, we consider the class $\mathcal{H}$ of threshold functions. For any $(x, x') \in \mathcal{X} \times \mathcal{X}$, the underlying comparison function is given by $Z^*(x, x') = \mathrm{sign}\big(f^*(x) - f^*(x')\big)$. Denote by $\mathcal{P}^C$ the uniform distribution over the pool of workers who will provide comparison tags, and a worker t is defined by a comparison function $Z_t : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$.

Assumption 4. When providing comparison tags, a $(1-\beta)$ fraction of the workers may behave adversarially, and the rest $\beta \le 1/2$ fraction are such that $\Pr(Z_t(x, x') \ne Z^*(x, x') \mid (x, x')) \le \eta$ for some $\eta \in [0, 1/2)$.

Let m_C be the total number of comparison queries to the crowd, Λ_C the comparison overhead, and m_V be the total number of verified label and comparison queries. Our second set of main results is a *label-efficient* crowdsourced PAC algorithm.

Theorem 3 (Theorem 11, informal)**.** *Suppose that Assumptions 2, 3, and 4 are satisfied, and $\eta < \frac{\min(\alpha,\beta)}{16}$. Given any $\epsilon, \delta \in (0, 1)$, there exists a polynomial-time algorithm (Algorithm 3) that PAC learns $\mathcal{H}$. In addition, $\Lambda_L = o_{\alpha,\beta}(1)$, $\Lambda_C = O_{\alpha,\beta}(1)$, and $m_V = O_{\alpha,\beta}(1)$.*

Remark 5. This is the first label-efficient algorithm that works in the semi-verified crowdsourced learning setting where adversarial workers can form a strong majority. Moreover, both the overheads and the number of verified data are independent of the sample size, meaning that the algorithm is query-efficient even for large-scale learning problems. The label efficiency follows from that Λ_L may tend to zero; alternatively, it also follows from the fact that $m_L = \tilde{O}(\log \frac{d}{\epsilon})$, where in label-only learning, this quantity scales as $\tilde{O}(\frac{d}{\epsilon})$ (Theorem 4). We note that a label-efficient crowdsourced PAC algorithm was obtained in Zeng and Shen (2022a), but their analysis only works for $\alpha, \beta > \frac{1}{2}$.

Remark 6. The merit of comparison-based PAC learning algorithms is to tradeoff the labeling and comparison complexity (Xu et al., 2017), rather than obtaining a lower overall query complexity. Indeed, the latter is information-theoretically impossible in general: a lower bound of $\Omega(d + 1/\epsilon)$ was shown in Kane et al. (2017).

1.3 Overview of Techniques

Win-win pruning of adversarial workers. When the adversarial workers form a majority in the crowd, a natural idea is to query the trusted oracle so as to identify and prune them. The main question here lies in how to select appropriate instances as testing cases such that a significant fraction of the adversarial workers can be pruned every time the trusted oracle is called. Our main contribution here is a *win-win argument* even when no perfect worker is present: the adversarial workers can either collude and provide incorrect annotations, for which the algorithm can identify and prune a noticeable fraction of them; or they are forced to provide correct annotations, for which we can be sure that the majority are correct. The key insight lies in a nat-

ural threshold of the majority size (the fraction of workers that agree with majority) being $(1-(1-\eta)\alpha)$, as on all accounts the adversarial workers cannot form a majority larger than that. This is also our condition for selecting the instances as testing cases, which generalizes the analysis in Awasthi et al. (2017b). See Section 3.1.

Query and label-efficient semi-verified filtering. To obtain an algorithm that is both label and query-efficient (Theorem 3), we need to resolve a more serious issue of majority voting over a constant number of workers: if the sample size increases, this majority voting can be incorrect with high probability. If we were to increase the constant number, the query complexity grows beyond what we can afford. To this end, we introduce a new testing stage to the comparison-equipped algorithm which draws a moderate set of informative samples and then queries comparison tags. This stage would give us enough information about how likely the crowd workers would form a strong majority, i.e. the majority size $\geq 1-(1-\eta)\beta$ such that the majority vote is faithful. We show that this technique suffices for us to develop a label and query-efficient algorithm. Readers may refer to Section 3.2 for details.

1.4 Additional Notations and Definitions

We will need the following quantity when designing comparison-based algorithms:

$$n_{\epsilon,\delta} := K \cdot \frac{1}{\epsilon} \cdot \left(d \log \frac{1}{\epsilon} + \left(\frac{1}{\delta}\right)^{1/1000} \right). \tag{1.2}$$

The additive term $\left(\frac{1}{\delta}\right)^{1/1000}$ is due to the fact that the failure probability of randomized quicksort only decays to zero in a rate inversely polynomial in the sample size. There is nothing magic on the exponent $\frac{1}{1000}$; it can be made to be an arbitrarily small constant in the price of a constant factor of more samples. We note that we can also use merge sort in place of the quicksort to obtain a $\log(1/\delta)$ dependence. However, since the latter is broadly deployed in practice, we choose to bring it upfront and leave the analysis with merge sort to interested readers.

Query complexity, overhead. The query complexity of a crowdsourced PAC learner is measured in three aspects: the total number of label queries from the crowd which we denote by m_L, that of comparison queries from the crowd which is denoted by m_C, and that of verified queries from the trusted expert which is denoted by m_V. Closely related to m_L and m_C are the following quantities:

$$\Lambda_L := \frac{m_L}{m_{\epsilon,\delta}}, \quad \Lambda_C := \frac{m_C}{m_{\epsilon,\delta}}, \tag{1.3}$$

termed *labeling overhead* and *comparison overhead*, respectively. These two quantities measure the query complexity of a crowdsourced PAC learner compared to the one needed in the noise-free standard PAC model. We say a crowdsourced PAC learner is *query-efficient* if $\Lambda_L + \Lambda_C + m_V = O(1)$, meaning that the overall query complexity is of the same order of the standard PAC model even facing an overwhelming fraction of adversarial workers. We say a crowdsourced PAC learner is *label-efficient* if $\Lambda_L = o(1)$. Our goal is to design a polynomial-time PAC learning algorithm that is both query-efficient and label-efficient; in addition, we aim to control the number of calls to the trusted expert as otherwise the problem is trivial.

Given an input z and a set of workers $\mathcal{W}$, we define $\mathrm{Maj}_{\mathcal{W}}(z)$ to be the majority vote over the annotations from the workers in $\mathcal{W}$, where z can either be an instance or a pair of instances. We denote by $\text{Maj-size}_{\mathcal{W}}(z)$ the fraction of the workers in $\mathcal{W}$ that agree with $\mathrm{Maj}_{\mathcal{W}}(z)$, which is referred to as the empirical majority size. Similarly, we can consider the population version $\text{Maj-size}_{\mathcal{P}}(z) = \max\{\Pr_{t\sim\mathcal{P}}(h_t(z)=1), \Pr_{t\sim\mathcal{P}}(h_t(z)=-1)\}$. Specifically, let h_1, h_2 and h_3 be three classifiers. The function $\mathrm{Maj}(h_1,h_2,h_3)$ maps any instance x to the label $y = \mathrm{Maj}_{\{h_1,h_2,h_3\}}(x)$. Given a distribution $\mathcal{P}$ over workers and an annotated set S (either a set with labels or one with comparison tags), denote by $\mathcal{P}_{|S}$ the distribution $\mathcal{P}$ conditioned on workers that agreed with the annotations in S; this will be useful to prune the adversarial workers. Given any region $R \subseteq \mathcal{X}$, denote by $D[R]$ the density of instances $x \in R$ under distribution D.

2 RELATED WORKS

Crowdsourcing has been broadly utilized as a tool for data annotation with provable algorithms (Vaughan, 2017). More in line with this work is crowdsourcing in the presence of highly noisy workers, i.e. the majority can be adversarial. Steinhardt et al. (2016); Meister and Valiant (2018) considered recovery of true values of a finite data set and Awasthi et al. (2017b) examined PAC learnability which aligns more closely with our work. Our algorithm shares the merits with Awasthi et al. (2017b) but we develop a novel analysis to address the label corruption from Massart workers, and we carefully leverage comparison queries to achieve label efficiency. We emphasize that crowdsourced PAC learning was also investigated in Zeng and Shen (2022a), but their algorithm fails immediately when the majority are incorrect.

The semi-verified learning model was recently proposed in Charikar et al. (2017). This model is also known as list-decodable learning where a learner is allowed to output a finite list of hypotheses among which at least one is guaranteed to be close to the true hypothesis. Such model has been broadly studied under the non-crowdsourcing setting for various problems such as mean estimation (Kothari and Steinhardt, 2017; Diakonikolas et al., 2020a; Zeng and Shen, 2022b), learning mixtures of Gaussian (Diakonikolas et al., 2018), linear regression (Karmalkar et al.,

2019; Raghavendra and Yau, 2020a), and subspace recovery (Bakshi and Kothari, 2020; Raghavendra and Yau, 2020b). We note, however, these works consider a setting where both instances and labels can be corrupted adversarially, while our work assumes that only the labels are noisy (since in crowdsourcing, the data curator often has control on the instances).

Learning with comparison queries has been extensively investigated in ranking (Jamieson and Nowak, 2011; Shah et al., 2016; Shah and Wainwright, 2017; Falahatgar et al., 2017; Shah et al., 2019; Agarwal et al., 2020), best-k item selection (Falahatgar et al., 2017; Ren et al., 2020), and recommendation systems (Fürnkranz and Hüllermeier, 2010; Falahatgar et al., 2018). PAC learning with pairwise comparisons is investigated recently (Kane et al., 2017; Xu et al., 2017; Hopkins et al., 2020); yet, the way that we utilize comparison queries is quite different from these works and their guarantees were established under the non-crowdsourcing model.

3 MAIN ALGORITHMS AND GUARANTEES

In this section, we present our main algorithms, one that uses label-only queries (Algorithm 1) and the other that combines label and comparison queries (Algorithm 3). In the label-only learning model, we will build upon the algorithm of Awasthi et al. (2017b) and show how a new pruning method (specifically, a new pruning threshold) can be leveraged into their algorithm to obtain Theorem 2. In the comparison-based learning model, we will extend the algorithm of Zeng and Shen (2022a) by presenting an ensemble of new algorithmic subroutines that delicately combine verified queries, label queries, and comparison queries.

First of all, we introduce the core idea of these algorithms, assuming for now that the adversarial workers only consisting a minority. We emphasize that obtaining distribution-free PAC guarantees under label noise in the standard PAC model is highly nontrivial, and among prevalent noise models, the only known computationally efficient PAC learner works for the very basic random classification noise (Blum et al., 1996), while for Massart noise or adversarial noise, hardness results have been established (Diakonikolas and Kane, 2022; Guruswami and Raghavendra, 2006). Yet, it turns out that as far as the majority is correct, by using majority vote, it is possible to PAC learn the underlying hypothesis class without distributional assumptions, where a key step is to gather sufficient labels for each instance so that with overwhelming probability, all the aggregated labels are correct, for which one can apply the realizable learner stated in Assumption 3. However, a naive label-and-train approach, as observed in Awasthi et al. (2017b), leads to a labeling overhead that grows with the sample size. Therefore, they proposed an elegant algorithm that interleaves data annotation and learning. In particular, there are three phases in the main algorithm: in phase 1, it aims to learn a hypothesis h_1 with $\mathrm{err}_D(h_1) \leq \frac{1}{2}\sqrt{\epsilon}$; in phase 2, it identifies a distribution D_2 where the performance of h_1 is almost comparable to random guess, and learns another h_2 with $\mathrm{err}_{D_2}(h_2) \leq \frac{1}{2}\sqrt{\epsilon}$; in phase 3, it concentrates on a distribution D_3 where h_1 and h_2 disagree, and learns the last hypothesis h_3 with $\mathrm{err}_{D_3}(h_3) \leq \frac{1}{2}\sqrt{\epsilon}$. The final output of the algorithm, $\hat{h}$, is a majority voting function over these three hypotheses. This is essentially the boosting algorithm of Schapire (1990) which enjoys a guarantee that $\mathrm{err}_D(\hat{h}) \leq \epsilon$.

Our algorithm in the semi-verified setting will be built upon the above, and the primary idea is to design query-efficient pruning methods to remove most adversarial workers, such that the Massart ones form the majority. To be more detailed, we will incorporate new pruning methods in all three phases to learn the desired hypotheses, while in some cases, additional technical efforts are needed to ensure the correctness of our results. We remark that the difference to obtain Theorem 3 compared to Theorem 2 is on how to use label queries in an economic way through the introduction of comparison queries.

3.1 Label-Only Algorithm: Algorithm 1

Now let us consider that the majority are adversarial. A similar semi-verified crowdsourcing model is also studied in Awasthi et al. (2017b) but their analysis, in particular, the success of their paradigm to prune adversarial workers, crucially relies on Assumption 1 to ensure that the perfect workers will always be retained during pruning, and thus eventually form the majority. This is because perfect workers make no mistake, so that none of them will be removed during the pruning step and hence the fraction of the perfect ones will increase and eventually form the majority. We here leverage a key observation that, even when some perfect workers are removed, it is still possible to increase the fraction α, so long as the fraction of the adversarial workers being removed is always strictly larger than that of the perfect workers. Hence, we can relax the assumption of perfect workers to Massart workers (Assumption 2).

The main algorithm is given in Algorithm 1, which contains three phases corresponding to our previous discussion. The most important component of the main algorithm is PRUNE-AND-LABEL (Algorithm 2). For each given instance x_t, in Step 3 of PRUNE-AND-LABEL, we check whether the current fraction of Massart workers, α, has formed the majority (note that we will update α, thus it has potential to increase), or the size of workers that agree with the majority vote is greater than a threshold. If either condition is satisfied, we take the majority vote as the label of x_t. What is subtle here is our condition on the majority size, which is essentially our pruning rule. To see why it

makes sense, observe that at the beginning, the adversarial workers form a $(1-\alpha)$ fraction, while the Massart workers form the α fraction among which, in expectation, at most $\alpha\eta$ will return incorrect labels. Therefore, the maximum number of incorrect labels will be $1-\alpha+\alpha\eta = 1-(1-\eta)\alpha$. So in expectation, if we see that the majority size is greater, it must be the case that the Massart workers who are committed to providing correct labels contribute to the voting, hence the majority vote is correct. It is easy to see that there is a constant factor difference in our algorithm, which is due to the conversion from expected value to high probability argument. When Step 3 of PRUNE-AND-LABEL is not satisfied, it is likely the case that majority vote gives incorrect label. This is a good chance to query the trusted oracle which guarantees elimination of a noticeable fraction of adversarial workers (and a few Massart workers). From these discussions, we can see that it is possible to achieve a win-win scenario: either adversarial workers give incorrect labels so that they can be eliminated using the verified queries, or they have to provide the correct label in order to stay in the pool until the end of the algorithm. Lastly, we show that in the presence of the Massart workers, we have to restart the main algorithm with $O(T_{\alpha,\eta})$ times (see Theorem 4). Note that for FILTER algorithm at Step 5 of Algorithm 1, we can reuse the one from Awasthi et al. (2017b), and we omit it due to space limit.

Massart noise rate η. Next, we demonstrate why the Massart noise rate η cannot be too large. In our model, since roughly $\eta\alpha$ fraction of the Massart workers will provide incorrect labels, they will likely be pruned away. Hence, denote by α_i the fraction of Massart workers in the pool after i restartings of the main algorithm, then a necessary condition on the success of pruning is that the fraction of Massart workers is increased, i.e.

$$\frac{(1-\eta)\alpha_i}{1-\frac{(1-\eta)\alpha_i}{8}} > \alpha_i,$$

which is equivalent to $\eta < \frac{(1-\eta)\alpha_i}{8}, \forall i \geq 0$. Assume the algorithm works properly and $\alpha_i \geq \alpha_{i-1}, \forall i$. Then, we require that $\eta < \frac{\alpha}{16}$ by telescoping, where α is the fraction of Massart workers in the crowd before running the algorithm.

Restarting times. Note that each time the main algorithm restarts, it removes at least $\frac{(1-\eta)\alpha}{8}$ fraction of the workers, with at most an $\eta\alpha$ fraction being Massart. Therefore, denote by N_i the total number of workers in the pool after i restartings of the main algorithm

$$\alpha_i \geq \frac{\alpha_0 N_0 \cdot (1-\eta)^i}{N_0 - \frac{(1-\eta)\alpha_0}{8}\cdot N_0 - \cdots - \frac{(1-\eta)\alpha_{i-1}}{8}\cdot N_{i-1}}.$$

By simple deductions on the above inquality, we can show that after $O\big(\log_{\frac{1}{1-\eta}}(\frac{\alpha-\alpha\eta}{\alpha-\eta})\big) = O(T_{\alpha,\eta})$ updates, α_i would surpass any given constant $C \in [0.7, 1]$. In other words, the main algorithm is restarted at most $O(T_{\alpha,\eta})$ times, using one verified label in each restart. Hence, the number of verified labels required from the trusted oracle $\mathcal{O}_{\mathcal{T}}$ is also bounded by $O(T_{\alpha,\eta})$.

Algorithm 1 Main Algorithm (label only)

Require: Parameters α, distributions $\mathcal{P}^L$, target error rate ϵ, confidence δ, noise rate η of Massart workers, a realizable PAC learner $\mathcal{A}_{\mathcal{H}}$.

Ensure: Hypothesis $\hat{h} : \mathcal{X} \to \mathcal{Y}$ such that with probability $1-\delta$, $\mathrm{err}_D(\hat{h}) \leq \epsilon$.

$T_{\alpha,\eta} \leftarrow \log_{\frac{1}{1-\eta}}(\frac{\alpha-\alpha\eta}{\alpha-\eta})$, $\tau \leftarrow 8T_{\alpha,\eta}$, $\delta' \leftarrow \frac{\delta}{\tau}$.

Phase 0:

$\overline{S_0} \leftarrow$ PRUNE-AND-LABEL(S_0, α, δ') for an instance set S_0 of size $\frac{1}{\epsilon}\log\frac{1}{\delta'}$ from D.

Phase 1:

$\overline{S_1} \leftarrow$ PRUNE-AND-LABEL$(S_1, \alpha, \frac{\delta'}{6})$, for an instance set S_1 of size $n_{\sqrt{\epsilon}/2,\delta'/6}$ from D.

$h_1 \leftarrow \mathcal{A}_{\mathcal{H}}(\overline{S_1}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta'}{6})$.

Phase 2:

$S_I \leftarrow$ FILTER$(S_2, h_1, \alpha, \frac{\delta'}{12})$, for an instance set S_2 of size $\Theta(n_{\epsilon,\delta'/12})$ drawn from D.

Let S_C be an instance set of size $\Theta(n_{\sqrt{\epsilon},\delta'/12})$ drawn from D.

$\overline{S_{\text{All}}} \leftarrow$ PRUNE-AND-LABEL$(S_I \cup S_C, \alpha, \delta'/12)$.

$\overline{W_I} \leftarrow \{(x,y) \in \overline{S_{\text{All}}} \mid y \neq h_1(x)\}$, $\overline{W_C} \leftarrow \overline{S_{\text{All}}} \backslash \overline{W_I}$.

Draw a sample set $\overline{W}$ of size $\Theta(n_{\sqrt{\epsilon}/2,\delta'/12})$ from the distribution $\frac{1}{2}\overline{W_I} + \frac{1}{2}\overline{W_C}$.

$h_2 \leftarrow \mathcal{A}_{\mathcal{H}}(\overline{W}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta'}{6})$.

Phase 3:

$\overline{S_3} \leftarrow$ PRUNE-AND-LABEL$(S_3, \alpha, \delta'/6)$, for an instance set S_3 of size $n_{\sqrt{\epsilon}/2,\delta'/6}$ drawn from D conditioned on $h_1(x) \neq h_2(x)$.

$h_3 \leftarrow \mathcal{A}_{\mathcal{H}}(\overline{S_3}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta'}{6})$.

return $\hat{h} \leftarrow \mathrm{Maj}(h_1, h_2, h_3)$.

Our main result for Algorithm 1 is summarized in the following; the proof is deferred to Appendix B.

Theorem 4. *Suppose that Assumptions 2 and 3 are satisfied, and $\eta < \frac{\alpha}{16}$. Denote $T_{\alpha,\eta} = \log_{\frac{1}{1-\eta}}(\frac{\alpha-\alpha\eta}{\alpha-\eta})$. The following holds with probability $1-\delta$. There exists an algorithm that runs in $O(\mathrm{poly}(d, \frac{1}{\epsilon}, T_{\alpha,\eta}, \frac{1}{\alpha}))$ time and learns $\mathcal{H}$ by returning a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ with $\mathrm{err}_D(h) \leq \epsilon$. In addition, $m_L = \tilde{O}\big(\frac{d\cdot T_{\alpha,\eta}}{\epsilon\alpha^2}\cdot\log\frac{1}{\delta}\big)$, and $m_V = O(T_{\alpha,\eta})$, where $\tilde{O}$ hides the logarithmic factors. Therefore, $\Lambda_L = O\big(\frac{T_{\alpha,\eta}}{\alpha^2}\cdot\log^2(T_{\alpha,\eta})\big)$ when $\epsilon \in \big(0, \log^{-2} d\big)$. In particular, when α is constant, $\Lambda_L = O(1)$ and $m_V = O(1)$.*

Remark 7. Recall that Awasthi et al. (2017b) studied the problem with the existence of perfect workers. Their work requires a number of $O(\frac{1}{\alpha})$ verified label from $\mathcal{O}_{\mathcal{T}}$, while in our analysis, the amount is $O(T_{\alpha,\eta}) = O(\log_{\frac{1}{1-\eta}}(\frac{\alpha-\alpha\eta}{\alpha-\eta}))$. It is not hard to show that $\lim_{\eta\to 0} T_{\alpha,\eta} = \frac{1}{\alpha}$. Thus, our re-

Algorithm 2 PRUNE-AND-LABEL

Require: An instance set S, parameters α, δ_1.
Ensure: A labeled set $\overline{S}$.
$k_0 \leftarrow \frac{1}{2(1-\eta)^2\alpha^2} \cdot \log \frac{1000m}{\delta_1}$.
Draw a set of k_0 workers $\mathcal{W}^L$ from $\mathcal{P}^L$.
if $\alpha \geq 0.7$ or $\text{Maj-size}_{\mathcal{W}^L}(x_t) \geq 1 - \frac{(1-\eta)\alpha}{4}$ **then**
  $\hat{y}_t \leftarrow \text{Maj}_{\mathcal{W}^L}(x_t)$.
else
  Get a verified label $y_t^* \leftarrow h^*(x_t)$. **Restart** Algoirthm 1 with $\mathcal{P}^L \leftarrow \mathcal{P}^L_{|(x_t, y_t^*)}$ and $\alpha \leftarrow \frac{(1-\eta)\alpha}{1-\frac{(1-\eta)\alpha}{8}}$.
end if
return $\overline{S}$.

sult is a new analysis on the structure of the Massart workers which guides an algorithmic refinement and derives a generalization of Awasthi et al. (2017b).

3.2 Comparison-Equipped Algorithm: Algorithm 3

In this section, the goal is to largely reduce the labeling cost using noisy pairwise comparisons from the crowd. We note that there exists a label-efficient algorithm in Zeng and Shen (2022a) that achieves the above tradeoff, but only works with $\alpha, \beta > \frac{1}{2}$. We will use their algorithm as the starting point and we elaborate on the technical challenges and our design details.

First, since there exists $1 - \beta \geq \frac{1}{2}$ fraction of adversarial workers in the crowd, a natural idea is to add a pruning step when collecting comparison tags from the crowd. Similar to the analysis in Section 3.1, it is easy to show that if a condition $\eta < \frac{\beta}{16}$ is satisfied, the win-win pruning scheme can also apply to $\mathcal{P}^C$ and increase the fraction of Massart workers who provide pairwise comparisons, i.e. β. Furthermore, it requires at most $O(T_{\beta,\eta}) = O\big(\log_{\frac{1}{1-\eta}}(\frac{\beta-\beta\eta}{\beta-\eta})\big)$ verified comparison tags from $\mathcal{O}_{\mathcal{T}}$ to increase β to any specified constant $C \in [0.7, 1]$. This leads to our design of algorithm PRUNE-COMPARE-AND-LABEL (see Section 3.2.1).

A more serious issue with $\beta < \frac{1}{2}$ roots in the filtering process of Zeng and Shen (2022a), which roughly speaking, will perform majority voting over a *constant* number of workers. In this regard, an empirical majority size may deviate far from the population one, thus the pruning fails. We also note that if we were to increase this constant number, we would suffer high query complexity. We thus develop a new testing stage that samples a large enough set of test cases and query the crowd for their majority size. If less than an ϵ fraction of the test cases have a small majority vote size, we can show that the total fraction of such test cases in the underlying distribution D is smaller than ϵ, hence the filtering process correctly filters all the desired instances except for an ϵ fraction; this is sufficient to establish a robust version of the performance guarantee. This

Algorithm 3 Main Algorithm (comparison-equipped)

Require: Parameters α, β, distributions $\mathcal{P}^L$ and $\mathcal{P}^C$, target error rate ϵ, confidence δ, noise rate η of Massart workers, a realizable PAC learner $\mathcal{A}_{\mathcal{H}}$.
Ensure: Hypothesis $\hat{h} : \mathcal{X} \to \mathcal{Y}$ such that with probability $1 - \delta$, $\text{err}_D(\hat{h}) \leq \epsilon$.
$T_{\alpha,\eta} \leftarrow \log_{\frac{1}{1-\eta}}(\frac{\alpha-\alpha\eta}{\alpha-\eta})$, $T_{\beta,\eta} \leftarrow \log_{\frac{1}{1-\eta}}(\frac{\beta-\beta\eta}{\beta-\eta})$, $\tau \leftarrow 8(T_{\alpha,\eta} + T_{\beta,\eta})$, $\delta' \leftarrow \frac{\delta}{\tau}$.
Phase 1:
$\overline{S_1} \leftarrow$ PRUNE-COMPARE-AND-LABEL$(S_1, \alpha, \beta, \frac{\delta'}{6})$, for an instance set S_1 of size $n_{\sqrt{\epsilon}/2,\delta'/6}$ from D.
$h_1 \leftarrow \mathcal{A}_{\mathcal{H}}(\overline{S_1}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta'}{6})$.
Phase 2:
$S_I \leftarrow$ SEMI-VERIFIED-FILTER$(S_2, h_1, \alpha, \beta, \frac{\delta'}{12})$, for an instance set S_2 of size $\Theta(n_{\epsilon,\delta'/12})$ drawn from D.
Let S_C be an instance set of size $\Theta(n_{\sqrt{\epsilon},\delta'/12})$ drawn from D.
$\overline{S_{\text{All}}} \leftarrow$ PRUNE-COMPARE-AND-LABEL$(S_I \cup S_C, \alpha, \beta, \delta'/12)$.
$\overline{W_I} \leftarrow \{(x, y) \in \overline{S_{\text{All}}} \mid y \neq h_1(x)\}$, $\overline{W_C} \leftarrow \overline{S_{\text{All}}} \backslash \overline{W_I}$.
Draw a sample set $\overline{W}$ of size $\Theta(n_{\sqrt{\epsilon}/2,\delta'/12})$ from the distribution $\frac{1}{2}\overline{W_I} + \frac{1}{2}\overline{W_C}$.
$h_2 \leftarrow \mathcal{A}_{\mathcal{H}}(\overline{W}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta'}{6})$.
Phase 3:
$\overline{S_3} \leftarrow$ PRUNE-COMPARE-AND-LABEL$(S_3, \alpha, \beta, \delta'/6)$, for an instance set S_3 of size $n_{\sqrt{\epsilon}/2,\delta'/6}$ drawn from D conditioned on $h_1(x) \neq h_2(x)$.
$h_3 \leftarrow \mathcal{A}_{\mathcal{H}}(\overline{S_3}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta'}{6})$.
return $\hat{h} \leftarrow \text{Maj}(h_1, h_2, h_3)$.

algorithmic design is presented in SEMI-VERIFIED-FILTER (see Section 3.2.2).

As a result, we obtain a label-efficient semi-verified crowdsourced learning algorithm as presented in Algorithm 3.

3.2.1 PRUNE-COMPARE-AND-LABEL

The comparison-based PAC learning algorithms of Xu et al. (2017); Zeng and Shen (2022a) aid label efficiency with a "compare-and-label" approach, which first sorts all the instances in a set S and then query the label of the instances during binary search. We fortify such method by adding certain check points for both the "compare" and "label" steps to ensure that the majority vote is always correct (with high probability). We summarize the performance guarantees of Algorithms 5 and 6 below.

Lemma 5. *Consider Algorithm 5. Given any set of instances S, with probability $1 - \frac{2\delta_1}{3}$, the algorithm either increases the fraction of Massart workers from β to $\frac{(1-\eta)\beta}{1-\frac{(1-\eta)\beta}{8}}$, or correctly sorts S.*

Lemma 6. *Consider Algorithm 6. Assume $\hat{S}$ is correctly*

Algorithm 4 PRUNE-COMPARE-AND-LABEL

Require: An instance set S, parameters α, β, δ_1.
Ensure: A sorted and labeled set $\overline{S}$.
1: $\hat{S} \leftarrow$ SEMI-VERIFIED-QUICKSORT(S, β, δ_1).
2: $\overline{S} \leftarrow$ SEMI-VERIFIED-BINARYSEARCH$(\hat{S}, \alpha, \delta_1)$.
return $\overline{S}$.

Algorithm 5 SEMI-VERIFIED-QUICKSORT

Require: An instance set $S = \{x_i\}_{i=1}^n$, β, δ_1.
Ensure: A sorted list $\hat{S}$.
1: $k_1 \leftarrow \frac{1}{2(1-\eta)^2\beta^2} \cdot \log \frac{3006n \cdot \log n}{\delta_1}$.
2: Apply randomized Quicksort to S: for each pair (x, x') being compared, draw a set of k_1 workers $\mathcal{W}^C$ from $\mathcal{P}^C$ and do the following:
3: **if** $\beta \geq 0.7$ or $\text{Maj-size}_{\mathcal{W}^C}(x, x') \geq 1 - \frac{(1-\eta)\beta}{4}$ **then**
4: Use $\text{Maj}_{\mathcal{W}^C}(x, x')$ as the comparison tag.
5: **else**
6: Get a verified comparison tag $z^* \leftarrow Z^*(x, x')$. **Restart** Algorithm 1 with $\mathcal{P}^C \leftarrow \mathcal{P}^C_{|(x,x',z^*)}$ and $\beta \leftarrow \frac{(1-\eta)\beta}{1-\frac{(1-\eta)\beta}{8}}$.
7: **end if**
return $\hat{S}$.

sorted. With probability $1 - \frac{\delta_1}{3}$*, the algorithm either increases the fraction of Massart workers to* $\frac{(1-\eta)\alpha}{1-\frac{(1-\eta)\alpha}{8}}$*, or correctly labels* $\hat{S}$*, namely, for all* $(x, y) \in \overline{S}$*,* $y = h^*(x)$*.*

We further note that comparing to the non-robust algorithm in Zeng and Shen (2022a), PRUNE-COMPARE-AND-LABEL does not require extra crowd labels or comparison tags to label S, because once it decides to query the verified data, it will end up restarting the main algorithm. Thus, the label and comparison complexity of PRUNE-COMPARE-AND-LABEL matches that of Zeng and Shen (2022a).

Proposition 7. *Consider Algorithm 4. If it does not restart and* $|S| \geq (\frac{1}{\delta_1})^{1/1000}$*, then with probability* $1 - \delta_1$*, it correctly sorts and labels all the instances in* S *with* $O(\frac{\log|S|}{\alpha^2} \cdot \log\log|S|)$ *label queries and* $O(\frac{1}{\beta^2} \cdot |S| \cdot \log^2|S|)$ *comparison tags.*

In light of Theorem 23 (i.e. boosting (Schapire, 1990)), the two weak hypothese h_1 and h_3 are trained on distributions D and D_3 which are not hard to construct if h_2 is given: D is the original marginal distribution, and D_3 can be obtained by rejection sampling. By Assumption 3, drawing a set of $\Theta(n_{\sqrt{\epsilon},\delta'})$ instances on each distribution respectively and feeding them to $\mathcal{A}_{\mathcal{H}}$ gives the following corollary.

Corollary 8. *With probability* $1 - \frac{\delta'}{3}$*,* $\text{err}_D(h_1) \leq \frac{\sqrt{\epsilon}}{2}$*. With probability* $1 - \frac{\delta'}{3}$*,* $\text{err}_{D_3}(h_3) \leq \frac{\sqrt{\epsilon}}{2}$*.*

3.2.2 SEMI-VERIFIED-FILTER

Algorithm 6 SEMI-VERIFIED-BINARYSEARCH

Require: A sorted instance set $\hat{S} = \{\hat{x}_i\}_{i=1}^n$, α, δ_1.
Ensure: A labeled set $\overline{S}$.
1: $t_{\min} \leftarrow 1$, $t_{\max} \leftarrow n$, $k_2 \leftarrow \frac{32}{(1-\eta)^2\alpha^2} \log \frac{6 \log n}{\delta_1}$.
2: **while** $t_{\min} < t_{\max}$ **do**
3: $t \leftarrow (t_{\min} + t_{\max})/2$. Draw a set of k_2 workers $\mathcal{W}^L$ from $\mathcal{P}^L$.
4: **if** $\alpha \geq 0.7$ or $\text{Maj-size}_{\mathcal{W}^L}(x_t) \geq 1 - \frac{(1-\eta)\alpha}{4}$ **then**
5: $\hat{y}_t \leftarrow \text{Maj}_{\mathcal{W}^L}(x_t)$. **If** $\hat{y}_t = 1$, **then** $t_{\max} \leftarrow t - 1$; **else** $t_{\min} \leftarrow t + 1$.
6: **else**
7: Get a verified label $y_t^* \leftarrow h^*(x_t)$. **Restart** Algoirthm 1 with $\mathcal{P}^L \leftarrow \mathcal{P}^L_{|(x_t, y_t^*)}$ and $\alpha \leftarrow \frac{(1-\eta)\alpha}{1-\frac{(1-\eta)\alpha}{8}}$.
8: **end if**
9: **end while**
10: For all $t' > t$, $\hat{y}_{t'} \leftarrow +1$. For all $t' < t$, $\hat{y}_{t'} \leftarrow -1$.
return $\overline{S} = \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \ldots, (\hat{x}_m, \hat{y}_m)\}$.

A core component of Algorithm 3 is a correct filtering process. Awasthi et al. (2017b) achieved this by executing a Phase 0 that examines the crowd labels for a randomly drawn instance set. However, this approach does not apply in the comparison-equipped setting because it relies on two support instances x^-, x^+ as reference to identify the informative instances. To be more concrete, the filtering scheme incorporates a similar idea to that of "compare-and-label" in order to achieve label efficiency. Given a large enough sample set S, the goal is to identify two support instances x^-, x^+ that are close enough to the "threshold" where the labels in S shift. By spending a few label queries to infer the labels of x^-, x^+, it help to filter a large number of instances in S by querying pairwise comparisons. This is because $\forall x$, $[h^*(x^-) = -1, x < x^-]$ imply $h^*(x) = -1$ (similar holds for x^+). The pairwise comparisons of any x with respect to x^-, x^+ become exceptionally informative.

We hence build on the non-semi-verified filter of Zeng and Shen (2022a) and integrate a new component, TEST, that ensures to recover a distribution D' that is close enough to D_2 even in presence of the adversarial workers. In particular, we are concerned with two regions, R_1 and R_2

$$R_1 := \Big\{x : \text{Maj-size}_{\mathcal{P}^C}(x, x^-) \leq 1 - \frac{\beta}{2}\Big\},$$
$$R_2 := \Big\{x : \text{Maj-size}_{\mathcal{P}^C}(x, x^+) \leq 1 - \frac{\beta}{2}\Big\}, \qquad (3.1)$$

which include any instance x that would form a low-confidence pair with either support instance x^- or x^+. Observe that both x^-, x^+ are fixed. If the probability mass of $R_1 \cup R_2$ under distribution D, denoted by $D[R_1 \cup R_2]$, is smaller than ϵ, it means the filter uses x^-, x^+ to correctly identify the instances except for the ϵ fraction in $R_1 \cup R_2$. Meanwhile, TEST works on a smaller set of instances than

Algorithm 7 SEMI-VERIFIED-FILTER

Require: Set of instances S, classifier h with error $\mathrm{err}_D(h) \in \left[\frac{\sqrt{\epsilon}}{6}, \frac{\sqrt{\epsilon}}{2}\right]$, confidence δ_2, parameters α, β.
Ensure: A set S_I whose instances are misclassified by h.
$b \leftarrow \frac{4}{\sqrt{\epsilon}} \log \frac{16}{\delta_2} + (\frac{24}{\delta_2})^{1/1000}$.
Sample uniformly $U \subset S$ of b instances. $\overline{U} \leftarrow$ PRUNE-COMPARE-AND-LABEL$(U, \alpha, \beta, \delta_2/4)$.
$x^- \leftarrow$ the rightmost instance of those labeled as -1, $x^+ \leftarrow$ the leftmost instance of those labeled as $+1$.
$S_I \leftarrow \emptyset$, $S_{\text{in}} \leftarrow \emptyset$, $N \leftarrow \frac{1}{\beta^2} \log \frac{1}{\epsilon}$.
if $\beta < 0.7$ **then**
  Run TEST $(x^-, \beta, \delta_2/8)$, TEST $(x^+, \beta, \delta_2/8)$.
end if
for $x \in S \backslash U$ **do**
  ANS $\leftarrow$ YES.
  for $t = 1, \ldots, N$ **do**
    Draw a worker $t \sim \mathcal{P}^C$ to obtain the comparison tag $Z_t(x, x^-)$. **If** $Z_t(x, x^-) = \{x < x^-\}$, **then** $Z_t(x, x^+) \leftarrow \{x < x^+\}$, **else** query $Z_t(x, x^+)$.
    If t is even, **then continue** to the next iteration.
    Filtering: **If** $\left[\mathrm{Maj}(Z_{1:t}(x, x^-)) = \{x < x^-\}\right.$ and $h(x) = -1\left.\right]$ or $\left[\mathrm{Maj}(Z_{1:t}(x, x^+)) = \{x > x^+\}\right.$ and $h(x) = 1\left.\right]$, **then** ANS $\leftarrow$ NO and **break**.
  end for
  If $\mathrm{Maj}\big(Z_{1:N}(x, x^-)\big) = \{x > x^-\}$ and $\mathrm{Maj}\big(Z_{1:N}(x, x^+)\big) = \{x < x^+\}$ **then** ANS $\leftarrow$ NO and $S_{\text{in}} \leftarrow S_{\text{in}} \cup \{x\}$.
  If ANS = YES, **then** $S_I \leftarrow S_I \cup \{x\}$.
end for
$\overline{S_{\text{in}}} \leftarrow$ PRUNE-COMPARE-AND-LABEL$(S_{\text{in}}, \delta_2/4, \alpha, \beta)$.
$S_I \leftarrow S_I \cup \{x : (x, y) \in \overline{U} \cup \overline{S_{\text{in}}}$ and $y \neq h(x)\}$.
return S_I.

Algorithm 8 TEST

Require: An instance x', parameter β, confidence δ_2.
Draw a set S_r of $\frac{4}{\epsilon} \log \frac{8}{\delta_2}$ instances from D.
$N \leftarrow \frac{32}{((1-\eta)\beta)^2} \cdot \log(\frac{32|S_r|}{\delta_2})$.
for $x \in S_r$ **do**
  Draw a set of N workers $\mathcal{W}$ from $\mathcal{P}^C$ and obtain the comparison tags on (x, x').
  if $\text{Maj-size}_{\mathcal{W}}(x, x') < 1 - \frac{(1-\eta)\beta}{4}$ **then**
    Get a verified comparison tag $z^* \leftarrow Z^*(x, x')$.
    Restart Algorithm 1 with $\mathcal{P}^C \leftarrow \mathcal{P}^C_{|(x,x',z^*)}$ and $\beta \leftarrow \frac{(1-\eta)\beta}{1-\frac{(1-\eta)\beta}{8}}$.
  end if
end for

what SEMI-VERIFIED-FILTER is expected to work on. This is a key design to guarantee that the total comparison complexity does not blow up.

Lemma 9. *If Algorithm 8 terminates without restarting Algorithm 3, we have the probability mass $D[R_1 \cup R_2] \leq \frac{\epsilon}{4}$ with probability $1 - \frac{\delta_2}{4}$. In addition, the comparison complexity is $O\big(\frac{1}{\epsilon\beta^2} \log\big(\frac{1}{\delta_2}\big) \log\big(\frac{1}{\epsilon\delta_2}\big)\big)$.*

SEMI-VERIFIED-FILTER ensures that in Phase 2, the learner draws a set of $\Theta(n_{\sqrt{\epsilon},\delta'})$ instances from a distribution D' that is a good simulation of D_2, which suffices to learn a good hypothesis h_2.

Lemma 10. *Consider Phase 2 of Algorithm 3. With probability $1 - \frac{\delta'}{3}$, $\mathrm{err}_{D_2}(h_2) \leq \frac{\sqrt{\epsilon}}{2}$.*

Now we have three hypotheses h_1, h_2, h_3 that satisfy the requirement for boosting; this suffices to produce a hypothesis $\hat{h}$ with the desired PAC guarantee. We further note that each pruning step helps to remove a significant fraction of the adversarial workers but only a small fraction of the Massart workers. Hence, our main theorem, Theorem 11, follows. See Appendix C for the full proof.

Theorem 11 (Comparison-equipped learning)**.** *Suppose that Assumptions 2, 3, and 4 are satisfied. Given any $\alpha, \beta, \epsilon, \delta \in (0, 1)$, assume $\eta < \frac{\min(\alpha,\beta)}{16}$. Denote $T_{\alpha,\eta} = \log_{\frac{1}{1-\eta}}(\frac{\alpha-\alpha\eta}{\alpha-\eta})$ and $T_{\beta,\eta} = \log_{\frac{1}{1-\eta}}(\frac{\beta-\beta\eta}{\beta-\eta})$. The following holds with probability $1 - \delta$. There exists an algorithm (Algorithm 3) that runs in $O(\mathrm{poly}(d, \frac{1}{\epsilon}, T_{\alpha,\eta}, T_{\beta,\eta}, \frac{1}{\alpha}, \frac{1}{\beta}))$ time and learns $\mathcal{H}$ by returning a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ with $\mathrm{err}_D(h) \leq \epsilon$. In addition, $m_L = T^2_{\alpha,\eta}\big(T_{\alpha,\eta} + T_{\beta,\eta}\big) \cdot \tilde{O}\big(\log \frac{d+(T_{\alpha,\eta}+T_{\beta,\eta})\frac{1}{\delta}}{\epsilon}\big)$, $m_C = T^2_{\beta,\eta} \cdot \tilde{O}_\delta\big(\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)^{\frac{1001}{1000}} \cdot \frac{d}{\epsilon}\big)$, and $m_V = T_{\alpha,\eta} + T_{\beta,\eta}$. Therefore, $\Lambda_L = \tilde{o}\big(T^2_{\alpha,\eta}(T_{\alpha,\eta} + T_{\beta,\eta})\big)$ and $\Lambda_C = \tilde{O}_\delta\big(T^2_{\beta,\eta}\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)^{\frac{1001}{1000}}\big)$ when $\epsilon \in \big(0, (\log d)^{-4}\big)$. In particular, when α and β are constants, we have $\Lambda_L = o(1)$, $\Lambda_C = O(1)$, and $m_V = O(1)$.*

4 CONCLUSION

In this paper, we studied the problem of semi-verified learning threshold functions from the crowd. We showed that when the majority of the crowd workers are adversarial and the rest behave as Massart noise, it is still possible to achieve PAC guarantees without distributional assumptions. In addition, our algorithms enjoy both query and label-efficiency, and run in polynomial time.

Acknowledgements

We thank the anonymous reviewers and meta-reviewer for valuable feedback. This work is supported by NSF-IIS-1948133 and by the startup funding from Stevens Institute of Technology.

References

Arpit Agarwal, Shivani Agarwal, Sanjeev Khanna, and Prathamesh Patil. Rank aggregation from pairwise comparisons in the presence of adversarial corruptions. In

Proceedings of the 37th International Conference on Machine Learning, pages 85–95, 2020.

Dana Angluin and Philip D. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1987.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

Pranjal Awasthi, Maria-Florina Balcan, and Philip M. Long. The power of localization for efficiently learning linear separators with noise. *Journal of the ACM*, 63 (6):50:1–50:27, 2017a.

Pranjal Awasthi, Avrim Blum, Nika Haghtalab, and Yishay Mansour. Efficient PAC learning from the crowd. In *Proceedings of the 30th Annual Conference on Learning Theory*, pages 127–150, 2017b.

Ainesh Bakshi and Pravesh Kothari. List-decodable subspace recovery via sum-of-squares. *CoRR*, abs/2002.05139, 2020.

Avrim Blum, Alan M. Frieze, Ravi Kannan, and Santosh S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 330–338, 1996.

Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 47–60, 2017.

Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *Proceedings of the 22nd Conference on Learning Theory*, 2009.

Ilias Diakonikolas and Daniel Kane. Near-optimal statistical query hardness of learning halfspaces with massart noise. In *Conference on Learning Theory*, pages 4258–4282, 2022.

Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1047–1060. ACM, 2018.

Ilias Diakonikolas, Themis Gouleakis, and Christos Tzamos. Distribution-independent PAC learning of halfspaces with Massart noise. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, pages 4751–4762, 2019.

Ilias Diakonikolas, Daniel Kane, and Daniel Kongsgaard. List-decodable mean estimation via iterative multi-filtering. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, 2020a.

Ilias Diakonikolas, Daniel M. Kane, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. A polynomial time algorithm for learning halfspaces with Tsybakov noise. *CoRR*, abs/2010.01705, 2020b.

Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. Learning halfspaces with Massart noise under structured distributions. In *Proceedings of the 33rd Annual Conference on Learning Theory*, pages 1486–1513, 2020c.

Ilias Diakonikolas, Daniel M. Kane, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. Agnostic proper learning of halfspaces under gaussian marginals. In *Proceedings of the 34th Annual Conference on Learning Theory*, pages 1522–1551, 2021.

Moein Falahatgar, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. Maxing and ranking with few assumptions. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pages 7060–7070, 2017.

Moein Falahatgar, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. The limits of maxing, ranking, and preference learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1426–1435, 2018.

Johannes Fürnkranz and Eyke Hüllermeier. Preference learning and ranking by pairwise comparison. In *Preference Learning*, pages 65–82. Springer, 2010.

Venkatesan Guruswami and Prasad Raghavendra. Hardness of learning halfspaces with noise. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 543–552, 2006.

David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Noise-tolerant, reliable active classification with comparison queries. In *Proceedings of the 33rd Annual Conference on Learning Theory*, pages 1957–2006, 2020.

Kevin G. Jamieson and Robert D. Nowak. Active ranking using pairwise comparisons. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, pages 2240–2248, 2011.

Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008.

Daniel M. Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science*, pages 355–366, 2017.

Sushrut Karmalkar, Adam R. Klivans, and Pravesh Kothari. List-decodable linear regression. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, pages 7423–7432, 2019.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

Michael J. Kearns, Robert E. Schapire, and Linda Sellie. Toward efficient agnostic learning. In *Proceedings of the 5th Annual Conference on Computational Learning Theory*, pages 341–352, 1992.

Pravesh K. Kothari and Jacob Steinhardt. Better agnostic clustering via relaxed tensor norms. *CoRR*, abs/1711.07465, 2017.

Pascal Massart and Élodie Nédélec. Risk bounds for statistical learning. *The Annals of Statistics*, pages 2326–2366, 2006.

Michela Meister and Gregory Valiant. A data prism: Semi-verified learning in the small-alpha regime. In *Proceedings of the 31st Conference On Learning Theory*, pages 1530–1546, 2018.

Dohyung Park, Joe Neeman, Jin Zhang, Sujay Sanghavi, and Inderjit S. Dhillon. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1907–1916, 2015.

Dražen Prelec, H Sebastian Seung, and John McCoy. A solution to the single-question crowd wisdom problem. *Nature*, 541(7638):532–535, 2017.

Prasad Raghavendra and Morris Yau. List decodable learning via sum of squares. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, pages 161–180, 2020a.

Prasad Raghavendra and Morris Yau. List decodable subspace recovery. In *Proceedings of the 33rd Annual Conference on Learning Theory*, pages 3206–3226, 2020b.

Wenbo Ren, Jia Liu, and Ness B. Shroff. The sample complexity of best-k items selection from pairwise comparisons. *CoRR*, abs/2007.03133, 2020.

Frank Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

Nihar B. Shah and Martin J. Wainwright. Simple, robust and optimal ranking from pairwise comparisons. *Journal of Machine Learning Research*, 18:199:1–199:38, 2017.

Nihar B. Shah, Sivaraman Balakrishnan, Aditya Guntuboyina, and Martin J. Wainwright. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 11–20, 2016.

Nihar B. Shah, Sivaraman Balakrishnan, and Martin J. Wainwright. Feeling the bern: Adaptive estimators for bernoulli probabilities of pairwise comparisons. *IEEE Transactions on Information Theory*, 65(8):4854–4874, 2019.

Robert H. Sloan. Types of noise in data for concept learning. In *Proceedings of the First Annual Workshop on Computational Learning Theory*, pages 91–96, 1988.

Jacob Steinhardt, Gregory Valiant, and Moses Charikar. Avoiding imposters and delinquents: Adversarial crowdsourcing and peer prediction. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems*, pages 4439–4447, 2016.

Alexander B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1): 135–166, 2004.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Jennifer Wortman Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18:193:1–193:46, 2017.

Yichong Xu, Hongyang Zhang, Aarti Singh, Artur Dubrawski, and Kyle Miller. Noise-tolerant interactive learning using pairwise comparisons. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pages 2431–2440, 2017.

Shiwei Zeng and Jie Shen. Efficient PAC learning from the crowd with pairwise comparisons. In *Proceedings of the 39th International Conference on Machine Learning*, pages 25973–25993, 2022a.

Shiwei Zeng and Jie Shen. List-decodable sparse mean estimation. *CoRR*, abs/2205.14337, 2022b.

Chicheng Zhang, Jie Shen, and Pranjal Awasthi. Efficient active learning of sparse halfspaces with arbitrary bounded noise. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, pages 7184–7197, 2020.

A OMITTED PROOF OF THEOREM 1

Proof. Recall that in the agnostic model, the adversary can flip an arbitrary $(1-\alpha)$-fraction of the labels in an adversarial manner while retaining the marginal distribution on $\mathcal{X}$. It is shown in Kalai et al. (2008); Diakonikolas et al. (2021) that if $\mathcal{H}_{\text{hs}}$ is the class of homogeneous halfspaces and D is the standard Gaussian, then there exists a learning algorithm which takes as input $\theta(d^{\text{poly}(1/\epsilon)})$ samples generated in such a way, runs in time $O((d/\epsilon)^{\text{poly}(1/\epsilon)})$, and returns a hypothesis with error rate less than $(1-\alpha)+\epsilon$ with overwhelming probability.

Now we note that in our crowdsourcing model, each instance is assigned one worker which is randomly chosen from the pool and may be adversarial with probability $1-\alpha$, while otherwise he is perfect. Therefore, the labels gathered in such a way satisfy the condition of agnostic noise model, and we apply the above results directly which gives Theorem 1. □

B OMITTED PROOF FOR THEOREM 4

The key idea of the proof is the following: we show that with the carefully designed pruning approach, the adversarial workers either corrupt the labels, under which a noticeable fraction of them will be pruned away and Algorithm 1 will be restarted with the cleaner pool of workers, or they provide labels in such a way that the majority vote is correct and Algorithm 1 is exactly mirroring the easier case where the majority is correct. Our goal here is to show that in the former case, the algorithm must make significant progress such that it obtains a pool of workers with most of them, say 70%, are Massart workers, for which we show that with high probability, a constant labeling overhead suffices to guarantee PAC learnability. Therefore, at the technical level, our algorithm and analysis are different from Awasthi et al. (2017b) in two aspects: first, we draw new analysis to handle Massart workers when bounding the restarting time, and second, we show that a majority of Massart workers also suffices under the condition that the Massart noise rate η is not large.

Lemma 12. *Consider the* PRUNE-AND-LABEL *subroutine in Algorithm 1. With probability $1-\delta_1$, the algorithm either increases the fraction of Massart workers from α to $\frac{(1-\eta)\alpha}{1-\frac{(1-\eta)\alpha}{8}}$, or correctly labels $\hat{S}$, namely, for all $(x,y)\in\overline{S}$, $y=h^*(x)$.*

Proof. Recall that Algorithm 2 queries a set $\mathcal{W}^L$ of $k_0=\frac{1}{2(1-\eta)^2\alpha^2}\cdot\log\frac{1000m}{\delta_1}$ workers for labeling each instance. Since $1-\eta\in(\frac{1}{2},1]$, by applying the Chernoff bound, we have

$$\Pr\left(\left|\text{Maj-size}_{\mathcal{P}^L}(x)-\text{Maj-size}_{\mathcal{W}^L}(x)\right|\geq\frac{(1-\eta)\alpha}{8}\right)\leq 2\cdot e^{-2k_0\cdot\left(\frac{\alpha}{16}\right)^2}\leq\frac{\delta_1}{1000m}.$$

Taking the union bound over the labeling of $\log n$ instances, we have that with probability at least $1-\delta_1$, for all $x\in S$, $\left|\text{Maj-size}_{\mathcal{P}^L}(x)-\text{Maj-size}_{\mathcal{W}^L}(x)\right|\leq\frac{(1-\eta)\alpha}{8}$. Thus, if $\text{Maj-size}_{\mathcal{W}^L}(x)\geq 1-\frac{(1-\eta)\alpha}{4}$, we have the population majority size $\text{Maj-size}_{\mathcal{P}^L}(x)\geq 1-\frac{3(1-\eta)\alpha}{8}>1-(1-\eta)\alpha$ with high probability, indicating that the majority voting is correct, i.e. $\text{Maj}_{\mathcal{W}^L}(x)=h^*(x)$. As a result, if the algorithm finishes without restarting, we are sure that $\forall(x,y)\in\overline{S}, y=h^*(x)$. On the flip side, if the algorithm prunes due to $\text{Maj-size}_{\mathcal{W}^L}(x)<1-\frac{(1-\eta)\alpha}{4}$, we have $\text{Maj-size}_{\mathcal{P}^L}(x)<1-\frac{(1-\eta)\alpha}{8}$ with high probability, indicating that if we query a verified label for x, it is guaranteed that more than a $\frac{(1-\eta)\alpha}{8}$-fraction of the workers can be pruned (among which only $\eta\alpha$ could be Massart workers), and $\alpha\leftarrow\frac{(1-\eta)\alpha}{1-\frac{(1-\eta)\alpha}{8}}$. □

Proposition 13. *Consider Algorithm 2. If it does not restart, then with probability $1-\delta_1$, it correctly labels all the instances in S with $O(\frac{m\log m}{\alpha^2})$ label queries.*

Proof. Due to Lemma 12, for any input set S of size m, the total number of crowd labels is $k_0\cdot O(|S|)=O\left(\frac{1}{\alpha^2}\log\left(\frac{m}{\delta_1}\right)\cdot m\right)=O\left(\frac{m\log m}{\alpha^2}\right)$. □

Lemma 14. *Consider Algorithm 1. $\forall\alpha\in(0,1]$, if $\eta<\frac{\alpha}{16}$, then with probability $1-\delta$, Algorithm 1 will restart $O(T_{\alpha,\eta})$ times. In addition, the required number of verified labels from $\mathcal{O}_{\mathcal{T}}$ is $m_V=O(T_{\alpha,\eta})$.*

Proof. By Lemma 12, each time we get a verified label y^* from $\mathcal{O}_{\mathcal{T}}$, we prune a $\frac{(1-\eta)\alpha}{8}$ fraction of the workers with high probability. Since it is not guaranteed that all Massart workers would give the correct labels, some of them would be pruned. Given that the pool is large, with probability $1-\delta/\tau$, at most an η fraction of the Massart workers will be pruned.

Denote by α_i the fraction of Massart workers and N_i the total number of workers in the pool after i prunings. Clearly, we have $\alpha_0 = \alpha$. Note that

$$\alpha_i N_i \geq \alpha_{i-1} N_{i-1} \cdot (1-\eta) \geq \cdots \geq \alpha_0 N_0 \cdot (1-\eta)^i.$$

Then, we have

$$\begin{aligned}
\alpha_K &\geq \frac{\alpha_0 N_0 \cdot (1-\eta)^K}{N_0 - \frac{(1-\eta)\alpha_0}{8} \cdot N_0 - \frac{(1-\eta)\alpha_1}{8} \cdot N_1 - \cdots - \frac{(1-\eta)\alpha_{K-1}}{8} \cdot N_{K-1}} \\
&= \frac{\alpha_0 N_0 \cdot (1-\eta)^K}{N_0 - \frac{(1-\eta)\alpha_0 N_0}{8} \cdot \left(1 + (1-\eta) + \cdots + (1-\eta)^K\right)} \\
&= \frac{\alpha_0 N_0 \cdot (1-\eta)^K}{N_0 - \frac{(1-\eta)\alpha_0 N_0}{8} \cdot \frac{1-(1-\eta)^{K+1}}{\eta}} \\
&= \frac{8\eta \cdot \alpha_0 \cdot (1-\eta)^K}{8\eta - \alpha_0(1-\eta) + \alpha_0(1-\eta)^{K+2}}
\end{aligned} \tag{B.1}$$

Consider parameters α_i and η. Obviously, it must be satisfied that

$$\eta < \frac{(1-\eta)\alpha_i}{8} \ \forall i \geq 0,$$

as otherwise, more fraction of Massart workers would be removed than that of the adversarial workers, i.e. the fraction of Massart workers in the pool decreases. Since $\alpha_i \geq \alpha_0$ for any $i \geq 0$, and $1-\eta \in (\frac{1}{2}, 1]$, it only requires $\eta < \frac{\alpha_0}{16}$.

From Eq. (B.1), to pick a sufficient number of verified labels K that increases α_K to any given constant $C \in [0.7, 1]$, it suffices to choose

$$K \geq \log_{(1-\eta)} \frac{C\alpha_0(1-\eta) - 8C\eta}{C\alpha_0(1-\eta)^2 - 8\eta\alpha_0}.$$

Due to $C \in [0.7, 1]$, $1-\eta \in (\frac{1}{2}, 1]$, $\alpha_0 < 1$, it requires at most

$$O\left(\log_{\frac{1}{1-\eta}} \frac{\alpha_0 - \eta\alpha_0}{\alpha_0 - \eta}\right) = O\left(T(\alpha_0, \eta)\right)$$

verified label from the trusted oracle $\mathcal{O}_{\mathcal{T}}$ for α_K to surpass 0.7. By union bound, with probability at least $1-\delta$, it requires at most $O(T_{\alpha,\eta})$ verified labels to increase the fraction of Massart workers such that they form a strong majority, where the algorithm no longer prunes and must return the desired hypothesis. Therefore, the main algorithm only restarts for $O(T_{\alpha,\eta})$ times. □

Proof of Theorem 4. Given that in all three phases, Algorithm 1 gathers a sample of size $O(m_{\sqrt{\epsilon},\delta'})$ and is correctly labeled with high probability. In addition, the algorithm is restarted at most $O(T_{\alpha,\eta})$ times (Lemma 14). Then, by union bound, Assumption 3, Lemma 24 and Theorem 23, we conclude that with probability at least $1-\delta$, Algorithm 1 returns h such that $\mathrm{err}_D(h) \leq \epsilon$.

The number of required verified labels from $\mathcal{O}_{\mathcal{T}}$ is $m_V = O(T_{\alpha,\eta})$ by Lemma 14. It remains to show the label complexity and label overhead. From Lemma 4.9 of Awasthi et al. (2017b)and Proposition 13, it requires

$$m_L = O\left(T_{\alpha,\eta} \cdot m_{\epsilon,\delta'} + T_{\alpha,\eta} \cdot \frac{1}{\epsilon} \cdot \log\frac{1}{\delta'} \cdot \log\frac{1}{\epsilon\delta'} + T_{\alpha,\eta} \cdot \frac{m_{\sqrt{\epsilon},\delta'} \log m_{\sqrt{\epsilon},\delta'}}{\alpha^2}\right) \tag{B.2}$$

label queries from the crowd, where we recall that $\delta' = \frac{\delta}{8T_{\alpha,\eta}}$ as defined in Algorithm 1. This combined with the definition of $m_{\epsilon,\delta}$ (see Eq. (1.1)) gives the announced label complexity of $\tilde{O}(\frac{dT_{\alpha,\eta}}{\epsilon\alpha^2} \cdot \log\frac{1}{\delta})$.

Finally, the calculation of labeling overhead follows from the definition, i.e.

$$\begin{aligned}
\Lambda_L = \frac{m_L}{m_{\epsilon,\delta}} &= O\left(T_{\alpha,\eta}\log T_{\alpha,\eta} + T_{\alpha,\eta}\log^2 T_{\alpha,\eta} + T_{\alpha,\eta} \cdot \frac{m_{\sqrt{\epsilon},\delta'}\log m_{\sqrt{\epsilon},\delta'}}{\alpha^2 \cdot m_{\epsilon,\delta}}\right) \\
&= O\left(T_{\alpha,\eta}\log^2 T_{\alpha,\eta} + \frac{T_{\alpha,\eta}}{\alpha^2}\sqrt{\epsilon}\log d \cdot \log^2 T_{\alpha,\eta}\right).
\end{aligned}$$

Thus, when $\epsilon < \log^{-2} d$, the labeling overhead reads as $O(\frac{T_{\alpha,\eta}}{\alpha^2}\log^2 T_{\alpha,\eta})$, which is upper bounded by a constant as far as α is a small constant. The proof is complete. □

C OMITTED PROOF FOR THEOREM 11

Lemma 15 (Restatement of Lemma 5)**.** *Consider Algorithm 5. Given any set of instances S, with probability $1-\frac{2\delta_1}{3}$, the algorithm either increases the fraction of Massart workers from β to $\frac{(1-\eta)\beta}{1-\frac{(1-\eta)\beta}{8}}$, or correctly sorts S.*

Proof. Given that $k_1 = \frac{1}{2\beta^2}\cdot\log\frac{3006n\cdot\log n}{\delta_1}$ in Algorithm 5. By the guarantee of celebrated algorithm RANDOMIZED QUICKSORT, with probability $1-\frac{1}{n^c}$, the total number of pairs that needs to be compared is $(c+2)n\log n$. By the sample size of input set S and setting $c=1000$, with probability $1-\frac{\delta_1}{3}$, $(c+2)n\log n = 1002n\log n$. In addition, similar to the proof of 6, we have $\text{Maj-size}_{\mathcal{P}^C}(x,x') = \mathbb{E}[\text{Maj-size}_{\mathcal{W}^C}(x,x')]$ for any (x,x'), and

$$\Pr\left[\bigcup_{l=1}^{1002n\log n}\left[\left|\text{Maj-size}_{\mathcal{P}^C}(x,x')_l - \text{Maj-size}_{\mathcal{W}^C}(x,x')_l\right| \le \frac{(1-\eta)\beta}{8}\right]\right] \le \frac{\delta_1}{3}.$$

In other words, with probability $1-\frac{\delta_1}{3}$, the following is guaranteed. If $\text{Maj-size}_{\mathcal{W}^C}(x,x') \ge 1-\frac{(1-\eta)\beta}{4}$, we have the population majority size $\text{Maj-size}_{\mathcal{P}^C}(x,x') \ge 1-\frac{3(1-\eta)\beta}{8} > 1-(1-\eta)\beta$, indicating that the majority voting is correct, i.e. $\text{Maj}_{\mathcal{W}^C}(x,x') = Z^*(x,x')$. As a result, if the algorithm never restarts, $\hat{S}$ is correctly sorted according to Z^*. On the other hand, if the algorithm prunes because $\text{Maj-size}_{\mathcal{W}^C}(x,x') < 1-\frac{(1-\eta)\beta}{4}$, we have $\text{Maj-size}_{\mathcal{P}^C}(x,x') \ge 1-\frac{(1-\eta)\beta}{8}$ with high probability, meaning that by querying a verified comparison for (x,x') we can remove at least a $\frac{(1-\eta)\beta}{8}$-fraction of the workers (with at most $\eta\beta$ being Massart) and update $\beta \leftarrow \frac{(1-\eta)\beta}{1-\frac{(1-\eta)\beta}{8}}$. □

Lemma 16 (Restatement of Lemma 6)**.** *Consider Algorithm 6. Assume $\hat{S}$ is correctly sorted. With probability $1-\frac{\delta_1}{3}$, the algorithm either increases the fraction of Massart workers to $\frac{(1-\eta)\alpha}{1-\frac{(1-\eta)\alpha}{8}}$, or correctly labels $\hat{S}$, namely, for all $(x,y)\in\overline{S}$, $y = h^*(x)$.*

Proof. The proof follows the same pipeline as that of Lemma 12, with parameter k_2 for SEMI-VERIFIED-BINARYSEARCH algorithm and the fact that binary search only queries labels on at most $\log n$ instances. □

Proposition 17 (Restatement of Proposition 7)**.** *Consider Algorithm 4. If it does not restart and $|S| \ge (\frac{1}{\delta_1})^{1/1000}$, then with probability $1-\delta_1$, it correctly sorts and labels all the instances in S with $O(\frac{\log n}{\alpha^2}\cdot\log\log n)$ label queries and $O(\frac{1}{\beta^2}\cdot n\log^2 n)$ comparison tags.*

Proof. Due to Lemma 15 and 16, for any input set S of size n, the total number of crowd labels is

$$k_2\cdot O(\log|S|) = O\left(\frac{1}{\alpha^2}\log\left(\frac{\log n}{\delta_1}\right)\cdot\log n\right) = O\left(\frac{\log n}{\alpha^2}\cdot\log\log n\right),$$

and the total number of crowd comparison tags is

$$k_1\cdot O\left(|S|\log|S|\right) = O\left(\frac{1}{\beta^2}\cdot\log\frac{n\cdot\log n}{\delta_1}\cdot n\log n\right) = O\left(\frac{1}{\beta^2}\cdot n\log^2 n\right).$$

□

Corollary 18 (Restatement of Corollary 8)**.** *With probability $1-\frac{\delta'}{3}$, $\text{err}_D(h_1) \le \frac{\sqrt{\epsilon}}{2}$. With probability $1-\frac{\delta'}{3}$, $\text{err}_{D_3}(h_3) \le \frac{\sqrt{\epsilon}}{2}$.*

Proof. By drawing a sample S_1 of $n_{\sqrt{\epsilon}/2,\delta'/6}$ from D and labeling it by PRUNE-COMPARE-AND-LABEL, with probability at least $1-\frac{\delta'}{6}$, $\overline{S_1}$ is labeled correctly according to h^* (Proposition 17). Furthermore, we note that $n_{\sqrt{\epsilon}/2,\delta'/6} \ge m_{\sqrt{\epsilon}/2,\delta'/6}$. By Assumption 3, $\text{err}_D(h_1) \le \frac{\sqrt{\epsilon}}{2}$ with probability $1-\frac{\delta'}{3}$.

Similarly, in Phase 3 of Algorithm 3, by rejection sampling we can successfully sample a set S_3 of $n_{\sqrt{\epsilon}/2,\delta'/6}$ from D_3. Again by Proposition 7 and Assumption 3, $\mathrm{err}_{D_3}(h_3) \le \frac{\sqrt{\epsilon}}{2}$ with probability $1 - \frac{\delta'}{3}$. □

Lemma 19. *$\forall \alpha, \beta \in (0,1]$, with probability $1-\delta$, Algorithm 3 will restart $O\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)$ times. In addition, the required number of verified labels from $\mathcal{O}_{\mathcal{T}}$ is $O(T_{\alpha,\eta})$ and that of the verified comparison tags is $O(T_{\beta,\eta})$.*

Proof. The deduction for removing adversarial workers from the pool of workers who provide comparison tags is similar to the one who provide labels. By union bound, with probability at least $1-\delta$, it requires at most $O(T_{\alpha,\eta})$ verified labels and $O(T_{\beta,\eta})$ verified comparison tags to increase the fraction of Massart workers such that they form a strong majority, where the algorithm no longer prunes and must return the desired hypothesis. Therefore, the main algorithm only restarts for $O(T_{\alpha,\eta} + T_{\beta,\eta})$ times. □

C.1 Performance Guarantee of Phase 2

Lemma 20 (Restatement of Lemma 9)**.** *If Algorithm 8 terminates without restarting Algorithm 3, we have the probability mass $D[R_1 \cup R_2] \le \frac{\epsilon}{4}$ with probability $1 - \frac{\delta_2}{4}$. In addition, the comparison complexity is $O\big(\frac{1}{\epsilon\beta^2}\log\big(\frac{1}{\delta_2}\big)\log\big(\frac{1}{\epsilon\delta_2}\big)\big)$.*

Proof. Recall that by definition, $R_1 := \{x : \text{Maj-size}_{\mathcal{P}^C}(x, x^-) \le 1 - \frac{\beta}{2}\}$ and $R_2 := \{x : \text{Maj-size}_{\mathcal{P}^C}(x, x^+) \le 1 - \frac{\beta}{2}\}$. Without loss of generality, we prove the lemma for x^-. We remark that the following guarantee holds for x^+ as well.

For any instance $x \in S_r$, let

$$\ell_i = \begin{cases} 1, & \text{if worker } i \text{ agrees with } \mathrm{Maj}_{\mathcal{P}^C}(x, x^-), \\ 0, & \text{otherwise.} \end{cases} \tag{C.1}$$

Since we query a set $\mathcal{W}$ of $N = \frac{32}{\beta^2} \cdot \log\big(\frac{32|S_r|}{\delta_2}\big)$ workers from the crowd to compare (x, x^-). Note that $\text{Maj-size}_{\mathcal{P}^C}(x, x^-) = \mathbb{E}[\text{Maj-size}_{\mathcal{W}}(x, x^-)] = \mu$, which is also the probability that $\ell_i = 1$. Therefore, by Hoeffding's inequality we have

$$\begin{aligned}
\Pr\left[\text{Maj-size}_{\mathcal{W}}(x, x^-) \ge 1 - \frac{(1-\eta)\beta}{4}\right] &\le \Pr\left[\left|\text{Maj-size}_{\mathcal{W}}(x, x^-) - \text{Maj-size}_{\mathcal{P}^C}(x, x^-)\right| \ge \frac{(1-\eta)\beta}{8}\right] \\
&= \Pr\left[\left|\frac{1}{N}\sum_{i=1}^{N}\ell_i - \mu\right| \ge \frac{(1-\eta)\beta}{8}\right] \\
&\le 2 \cdot e^{-\frac{2N\cdot\left(\frac{\beta}{16}\right)^2}{(1-0)^2}} \\
&\le 2 \cdot e^{-2\cdot\frac{32}{\beta^2}\cdot\log\left(\frac{32|S_r|}{\delta_2}\right)\cdot\left(\frac{\beta}{16}\right)^2} \\
&\le \frac{\delta_2}{16|S_r|}.
\end{aligned}$$

By union bound, with probability $1 - \frac{\delta_2}{16}$, if there exists some $x \in R_1$ in set S_r, the algorithm detects and removes it. In other words, if TEST terminates without restarting Algorithm 3, the probability mass $D[R_1] \le \frac{\epsilon}{8}$ with probability $1 - \frac{\delta_2}{8}$. Therefore, if Algorithm 7 reaches its Step 7, we have $D[R_1 \cup R_2] \le \frac{\epsilon}{4}$ with probability at least $1 - \frac{\delta_2}{4}$.

In addition, the total number of comparison tags in TEST is

$$\frac{8}{\epsilon}\log\frac{16}{\delta_2} \cdot \frac{32}{\beta^2} \cdot \log\Big(\frac{8|S_r|}{\delta_2}\Big) = O\Big(\frac{1}{\epsilon\beta^2}\log\frac{1}{\delta_2}\log\Big(\frac{1}{\epsilon\delta_2}\Big)\Big). \tag{C.2}$$

□

Lemma 21 (Restatement of Lemma 10)**.** *Consider Phase 2 of Algorithm 3, with probability $1 - \frac{\delta'}{3}$, $\mathrm{err}_{D_2}(h_2) \le \frac{\sqrt{\epsilon}}{2}$.*

Proof. We acknowledge that some of the deductions in this proof follow directly from that of Awasthi et al. (2017b) and are included for completeness.

Let $R := R_1 \cup R_2$. By Lemma 9 and $\delta_2 = \frac{\delta'}{12}$, with probability at least $1 - \frac{\delta'}{12}$, $D[R] \leq \frac{\epsilon}{4}$. We first argue that for any $x \notin R$, SEMI-VERIFIED-FILTER does a good job to simulate D_I. Consider distribution D' that has equal probability on the distributions induced by $\overline{W}_I$ and $\overline{W}_C$ and let $d'(x)$ denote the density of point x in this distribution. Likewise let $d_2(x)$ be the density of points in D_2. We want to show that for any $x \notin R$, $d'(x) = \Theta(d_2(x))$.

Recall that $\mathrm{err}_D(h_1) = \Theta(\frac{\sqrt{\epsilon}}{2})$. Let $d(x)$, $d_C(x)$, and $d_I(x)$ be the density of instance x in distributions D, D_C, and D_I, respectively. Note that, for any x such that $h_1(x) = h^*(x)$, we have $d(x) = d_C(x)(1 - \frac{1}{2}\sqrt{\epsilon})$. Similarly, for any x such that $h_1(x) \neq h^*(x)$, we have $d(x) = d_I(x)\frac{1}{2}\sqrt{\epsilon}$. $N_C(x)$, $N_I(x)$, $M_C(x)$ and $M_I(x)$ be the number of occurrences of x in the sets S_C, S_I, $\overline{W_C}$ and $\overline{W_I}$, respectively. For any x, there are two cases:

If $h_1(x) = h^*(x)$: Then, there exist absolute constants c_1 and c_2 according to Lemma 26, such that

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_C(x)}{|\overline{W_C}|}\right] \geq \frac{\mathbb{E}[M_C(x)]}{c_1 \cdot m_{\sqrt{\epsilon}}} \geq \frac{\mathbb{E}[N_C(x)]}{c_1 \cdot m_{\sqrt{\epsilon}}} = \frac{|S_C| \cdot d(x)}{c_1 \cdot m_{\sqrt{\epsilon}}}$$
$$= \frac{|S_C| \cdot d_C(x) \cdot (1 - \frac{1}{2}\sqrt{\epsilon})}{c_1 \cdot m_{\sqrt{\epsilon}}} \geq c_2 d_C(x) = \frac{c_2 d_2(x)}{2},$$

where the second and sixth transitions are by the sizes of $\overline{W_C}$ and $|S_C|$ and the third transition is by the fact that if $h(x) = h^*(x)$, $M_C(x) > N_C(x)$.

If $h_1(x) \neq h^*(x)$: Then, there exist absolute constants c_1' and c_2' according to Lemma 26, such that

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_I(x)}{|\overline{W_I}|}\right] \geq \frac{\mathbb{E}[M_I(x)]}{c_1' \cdot m_{\sqrt{\epsilon}}} \geq \frac{\mathbb{E}[N_I(x)]}{c_1' \cdot m_{\sqrt{\epsilon}}} \geq \frac{\frac{1}{2}\,d(x)|S_2|}{c_1' \cdot m_{\sqrt{\epsilon}}}$$
$$= \frac{\frac{1}{2}\,d_I(x)\frac{1}{2}\sqrt{\epsilon} \cdot |S_2|}{c_1' \cdot m_{\sqrt{\epsilon}}} \geq c_2' d_I(x) = \frac{c_2' d_2(x)}{2},$$

where the second and sixth transitions are by the sizes of $\overline{W_I}$ and $|S_2|$, the third transition is by the fact that if $h(x) \neq h^*(x)$, $M_I(x) > N_I(x)$, and the fourth transition holds by Lemma 25.

For $x \in R$, we have $D[R] \leq \frac{\epsilon}{4}$. Therefore, $D_2[R] \leq \frac{\sqrt{\epsilon}}{4}$ because $D = \frac{\sqrt{\epsilon}}{2}D_I + (1 - \frac{\sqrt{\epsilon}}{2})D_C$ and $D_2 = \frac{1}{2}D_I + \frac{1}{2}D_C$. As a result, except for a $\frac{\sqrt{\epsilon}}{4}$ fraction under D_2, $\forall x\ d'(x) \geq \Theta(d_2(x))$, meaning that D' is a good simulation of D_2. By Lemma 26, since $\left|\overline{W}\right| = \Theta(n_{\sqrt{\epsilon},\frac{\delta'}{12}})$, applying the super-sampling lemma (Lemma 24) we know that $\mathrm{err}_{D_2}(h_2) \leq O(\frac{\sqrt{\epsilon}}{2})$ with probability $1 - \frac{\delta'}{3}$. □

Theorem 22 (Restatement of Theorem 11)**.** *Given any $\alpha, \beta, \epsilon, \delta \in (0,1)$, assume $\eta < \frac{\min(\alpha,\beta)}{16}$. Denote $T_{\alpha,\eta} = \log_{\frac{1}{1-\eta}}(\frac{\alpha-\alpha\eta}{\alpha-\eta})$ and $T_{\beta,\eta} = \log_{\frac{1}{1-\eta}}(\frac{\beta-\beta\eta}{\beta-\eta})$. The following holds with probability $1-\delta$. There exists an algorithm (Algorithm 3) that runs in $O(\mathrm{poly}(d, T_{\alpha,\eta}, T_{\beta,\eta}, \frac{1}{\epsilon}))$ time and returns a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ with $\mathrm{err}_D(h) \leq \epsilon$. In addition, $m_L = \frac{T_{\alpha,\eta}+T_{\beta,\eta}}{\alpha^2} \cdot \tilde{O}\big(\log \frac{d+(T_{\alpha,\eta}+T_{\beta,\eta})\frac{1}{\delta}}{\epsilon}\big)$, $m_C = \frac{1}{\beta^2} \cdot \tilde{O}\big(\big(T_{\alpha,\eta}+T_{\beta,\eta}\big)^{\frac{1001}{1000}} \cdot n_{\epsilon,\delta}\big)$, and $m_V = O(T_{\alpha,\eta}+T_{\beta,\eta})$. Therefore, $\Lambda_L = \tilde{o}\big(\frac{T_{\alpha,\eta}+T_{\beta,\eta}}{\alpha^2}\big)$ and $\Lambda_C = \tilde{O}_\delta\big(\frac{1}{\beta^2}\big(T_{\alpha,\eta}+T_{\beta,\eta}\big)^{\frac{1001}{1000}}\big)$ when $\epsilon \in \big(0, (\log d)^{-4}\big)$. In particular, when α and β are constants, $\Lambda_L = o(1)$, $\Lambda_C = O(1)$, and $m_V = O(1)$.*

Remark 8. The above theorem presents tighter bounds for $m_L, m_C, \Lambda_L, \Lambda_C$ than that in Theorem 11. Notice that $\frac{1}{\alpha} = O(T_{\alpha,\eta})$ and $\frac{1}{\beta} = O(T_{\beta,\eta})$. Hence, Theorem 11 in our main paper is more general. In addition, the quantities $T_{\alpha,\eta}, T_{\beta,\eta}, \frac{1}{\alpha}, \frac{1}{\beta}, \eta$ are indepedent of the dimension d, the desired error rate ϵ, and the confidence parameter δ. Hence, when consider m_V and the overheads, our algorithm is query-efficient.

Proof. Following Corollary 18, Lemma 19 and 21, if the main algorithm outputs a hypothesis h without restarting, we have the following guarantees with probability $1-\delta$: (i) $\mathrm{err}_D(h_1) \leq \frac{\sqrt{\epsilon}}{2}$; $\mathrm{err}_{D_2}(h_2) \leq \frac{\sqrt{\epsilon}}{2}$; $\mathrm{err}_{D_3}(h_3) \leq \frac{\sqrt{\epsilon}}{2}$ hold simultaneously; (ii) the algorithm queries at most $O(T_{\alpha,\eta})$ verified labels and $O(T_{\beta,\eta})$ verified comparison tags from $\mathcal{O}_{\mathcal{T}}$. Applying Theorem 23, the hypothesis $\hat{h}$ returned by Algorithm 1 is such that $\mathrm{err}_D(\hat{h}) \leq \epsilon$ with probability $1-\delta$.

In Lemma 19, we show that the success of our pruning scheme crucially relies on a condition that $\eta < \frac{\alpha}{c}$ and $\eta < \frac{\beta}{c}$ for some large enough constant $c > 0$. Set $c = 16$. We require $\eta < \frac{\min(\alpha,\beta)}{16}$ for the guarantees in Theorem 11 to hold. Moreover, $m_V = O(T_{\alpha,\eta} + T_{\beta,\eta})$.

It remains to show the label and comparison compleixty, m_L, m_C, and corresponding overheads Λ_L, Λ_C. The label complexity follows from Proposition 17 by setting $n = n_{\sqrt{\epsilon},\delta'}$ and the fact that main algorithm is only restarted by a total of $O(T_{\alpha,\eta} + T_{\beta,\eta})$ times. Note that $O(\frac{1}{\delta_1}) = O(\frac{1}{\delta_2}) = O(\frac{1}{\delta'}) = O((T_{\alpha,\eta} + T_{\beta,\eta}) \cdot \frac{1}{\delta})$. We have

$$\begin{aligned} m_L &= (T_{\alpha,\eta} + T_{\beta,\eta}) \cdot O\bigg(\frac{\log n_{\sqrt{\epsilon},\delta'}}{\alpha^2} \cdot \log\log n_{\sqrt{\epsilon},\delta'} \bigg) \\ &= \frac{T_{\alpha,\eta} + T_{\beta,\eta}}{\alpha^2} \cdot \tilde{O}\bigg(\log \frac{d + (T_{\alpha,\eta} + T_{\beta,\eta})\frac{1}{\delta}}{\epsilon} \bigg). \end{aligned}$$

On the other hand, the comparison complexity is a summation of the results from Lemma 20 and that of Zeng and Shen (2022a),

$$\begin{aligned} m_C &= (T_{\alpha,\eta} + T_{\beta,\eta}) \cdot O\bigg(\frac{n_{\sqrt{\epsilon},\delta'}}{\beta^2} \cdot \log^2 n_{\sqrt{\epsilon},\delta'} + n_{\epsilon,\delta'} + \frac{1}{\epsilon\beta^2} \log\Big(\frac{1}{\delta'}\Big) \log\Big(\frac{1}{\epsilon\delta'}\Big) \bigg) \\ &= \frac{T_{\alpha,\eta} + T_{\beta,\eta}}{\beta^2} \cdot O\bigg(\frac{d\log\frac{1}{\epsilon} + \big((T_{\alpha,\eta} + T_{\beta,\eta})\frac{1}{\delta}\big)^{\frac{1}{1000}} + \log\big((T_{\alpha,\eta} + T_{\beta,\eta})\frac{1}{\delta}\big) \cdot \big(\log\frac{1}{\epsilon} + \log(T_{\alpha,\eta} + T_{\beta,\eta})\frac{1}{\delta}\big)}{\epsilon} \bigg) \\ &= \frac{1}{\beta^2} \cdot \tilde{O}\bigg((T_{\alpha,\eta} + T_{\beta,\eta})^{\frac{1001}{1000}} \cdot \frac{d + \big(\frac{1}{\delta}\big)^{\frac{1}{1000}}}{\epsilon} \bigg) = \frac{1}{\beta^2} \cdot \tilde{O}\big(\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)^{\frac{1001}{1000}} \cdot n_{\epsilon,\delta} \big) \end{aligned}$$

These in allusion to $m_{\epsilon,\delta} = K \cdot \big(\frac{1}{\epsilon}(d\log(1/\epsilon) + \log(1/\delta)\big)$ immediately give the overheads as follows:

$$\begin{aligned} \Lambda_L &= O\bigg(\frac{T_{\alpha,\eta} + T_{\beta,\eta}}{\alpha^2} \cdot \frac{\log n_{\sqrt{\epsilon},\delta'}}{m_{\epsilon,\delta}} \cdot \log\log n_{\sqrt{\epsilon},\delta'} \bigg) \\ &\le \frac{T_{\alpha,\eta} + T_{\beta,\eta}}{\alpha^2} \cdot \frac{\epsilon}{d + \log(1/\delta)} \cdot \tilde{O}\bigg(\log \frac{d + \big(T_{\alpha,\eta} + T_{\beta,\eta}\big)\frac{1}{\delta}}{\epsilon} \bigg) \\ &= \frac{T_{\alpha,\eta} + T_{\beta,\eta}}{\alpha^2} \log\Big(T_{\alpha,\eta} + T_{\beta,\eta}\Big) \cdot \frac{\epsilon}{d} \cdot \tilde{O}\Big(\log\frac{d}{\epsilon} \Big), \end{aligned}$$

and

$$\begin{aligned} \Lambda_C &= \frac{T_{\alpha,\eta} + T_{\beta,\eta}}{\beta^2} \cdot O\bigg(\frac{n_{\sqrt{\epsilon},\delta'}}{m_{\epsilon,\delta}} \cdot \log^2 n_{\sqrt{\epsilon},\delta'} + \frac{n_{\epsilon,\delta'}}{m_{\epsilon,\delta}} + \frac{\frac{1}{\epsilon}\log\big(\frac{1}{\delta'}\big)\log\big(\frac{1}{\epsilon\delta'}\big)}{m_{\epsilon,\delta}} \bigg) \\ &\le \frac{T_{\alpha,\eta} + T_{\beta,\eta}}{\beta^2} \cdot O\bigg(\sqrt{\epsilon} \cdot \frac{d\log\frac{1}{\epsilon} + \big(\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)\frac{1}{\delta}\big)^{\frac{1}{1000}}}{d\log\frac{1}{\epsilon} + \log\frac{1}{\delta}} \cdot \log^2\bigg(\frac{d + \big(T_{\alpha,\eta} + T_{\beta,\eta}\big)\frac{1}{\delta}}{\epsilon} \bigg) \\ &\quad + \frac{d\log\frac{1}{\epsilon} + \big(\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)\frac{1}{\delta}\big)^{\frac{1}{1000}}}{d\log\frac{1}{\epsilon} + \log(\frac{1}{\delta})} + \frac{\log\big(\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)\frac{1}{\delta}\big)\log\big(\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)\frac{1}{\epsilon\delta}\big)}{d\log\frac{1}{\epsilon} + \log\frac{1}{\delta}} \bigg) \\ &\le \frac{1}{\beta^2}\Big(T_{\alpha,\eta} + T_{\beta,\eta}\Big)^{\frac{1001}{1000}} \log^2\Big(T_{\alpha,\eta} + T_{\beta,\eta}\Big) \cdot O\bigg(\sqrt{\epsilon} \cdot \frac{d\log\frac{1}{\epsilon} + \big(\frac{1}{\delta}\big)^{\frac{1}{1000}}}{d\log\frac{1}{\epsilon} + \log\frac{1}{\delta}} \cdot \log^2\bigg(\frac{d + \frac{1}{\delta}}{\epsilon} \bigg) \\ &\quad + \frac{d\log\frac{1}{\epsilon} + \big(\frac{1}{\delta}\big)^{\frac{1}{1000}}}{d\log\frac{1}{\epsilon} + \log(\frac{1}{\delta})} + \frac{\log\big(\frac{1}{\delta}\big)\big(\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\big)}{d\log\frac{1}{\epsilon} + \log\frac{1}{\delta}} \bigg). \end{aligned}$$

Recall that by the definition of $n_{\epsilon,\delta}$ and $m_{\epsilon,\delta}$ in Section 1.4, we have $m_{\epsilon,\delta} = \Theta(n_{\epsilon,\delta}) = \Theta(\frac{d}{\epsilon}\log\frac{1}{\epsilon})$ when δ is a constant. In this case, we can see that

$$\Lambda_C \le \frac{1}{\beta^2}\Big(T_{\alpha,\eta} + T_{\beta,\eta}\Big)^{\frac{1001}{1000}} \log^2\Big(T_{\alpha,\eta} + T_{\beta,\eta}\Big) \cdot O_\delta\Big(\sqrt{\epsilon} \cdot \log^2\frac{d}{\epsilon} + 2 \Big).$$

When $\epsilon \in \big(0, (\log d)^{-4}\big)$, $\Lambda_C = \tilde{O}_\delta\Big(\frac{1}{\beta^2}\big(T_{\alpha,\eta} + T_{\beta,\eta}\big)^{\frac{1001}{1000}}\Big)$. In addition, $\Lambda_L = o\Big(\frac{T_{\alpha,\eta}+T_{\beta,\eta}}{\alpha^2} \cdot \log(T_{\alpha,\eta} + T_{\beta,\eta})\Big)$, because Λ_L goes to 0 when ϵ goes to 0. □

D USEFUL LEMMAS

Theorem 23 (Boosting, Schapire (1990)). *For any $p < \frac{1}{2}$ and distribution D, consider three classifiers $h_1(x)$, $h_2(x)$, $h_3(x)$ satisfying the following. 1) $\mathrm{err}_D(h_1) \leq p$; 2) $\mathrm{err}_{D_2}(h_2) \leq p$ where $D_2 := \frac{1}{2}D_C + \frac{1}{2}D_I$, D_C denotes the distribution D conditioned on $\{x : h_1(x) = h^*(x)\}$, and D_I denotes D conditioned on $\{x : h_1(x) \neq h^*(x)\}$; 3) $\mathrm{err}_{D_3}(h_3) \leq p$ where D_3 is D conditioned on $\{x : h_1(x) \neq h_2(x)\}$. Then $\mathrm{err}_D(\mathrm{Maj}(h_1, h_2, h_3)) \leq 3p^2 - 2p^3$.*

Lemma 24 (Robust Super-Sampling Lemma, Lemma 4.12 in Awasthi et al. (2017b)). *Given a hypothesis class $\mathcal{H}$ consider any two discrete distributions D and D' over $\mathcal{X}$ such that except for an ϵ fraction of the mass under D, we have that for all x, $d'(x) \geq c \cdot d(x)$ for an absolute constant $c > 0$ and both distributions are labeled according to $h^* \in \mathcal{H}$. There exists a constant $c' > 1$ such that for any ϵ, δ, with probability $1 - \delta$ over a labeled sample set S of size $c' m_{\epsilon,\delta}$ drawn from D', $\mathcal{A}_{\mathcal{H}}(S)$ has error of at most 2ϵ with respect to D.*

Lemma 25 (Lemma 14 in Zeng and Shen (2022a)). *Consider Algorithm 7. Assume that the subset U is correctly labeled. Consider any given instance $x \in S_2$ except for an $\frac{\epsilon}{2}$-fraction, we have the following guarantee. If $h(x) = h^*(x)$, it will be added to S_I with probability at most $\frac{1}{4}\sqrt{\epsilon}$; if $h(x) \neq h^*(x)$, it goes to S_I with probability at least $\frac{4}{7}$.*

Proof. First, for any instance $x \in S_{\text{in}} \cup U$, the guarantee follows from that of PRUNE-COMPARE-AND-LABEL.

Now consider any instance $x \in S \backslash \{S_{\text{in}} \cup U\}$. Recall that Lemma 9 guarantees the probability mass $D[R_1 \cup R_2] \leq \frac{\epsilon}{4}$ with probability $1 - \frac{\delta_2}{4}$. Given that the input size of S is $\Theta(n_{\epsilon,\delta_2})$, with probabiltiy at least $1 - \frac{\delta_2}{12}$, the fraction of the instances $x \in S_2$ that falls in region $R_1 \cup R_2$ is less than $\frac{\epsilon}{2}$. Therefore, at least $1 - \frac{\epsilon}{2}$ fraction of the instances in S falls outside $R_1 \cup R_2$ such that $\text{Maj-size}_{\mathcal{P}^C}(x, x^-) \geq 1 - \frac{\beta}{2}$ and $\text{Maj-size}_{\mathcal{P}^C}(x, x^+) \geq 1 - \frac{\beta}{2}$ (Eq. (3.1)). For these instances, we can consider that the majority is correct. Now if x is outside the interval $[x^-, x^+]$, when $\beta < 0.7$, $\text{Maj-size}_{\mathcal{P}^C}(x, x') \geq \frac{1}{2} + \Theta(1)$ which suffices for the filtering scheme from Awasthi et al. (2017b) to work; when $\beta \geq 0.7$, lemma follows from that of Zeng and Shen (2022a). If x falls into the interval $[x^-, x^+]$, the probability that it will be added to S_I but $h(x) = h^*(x)$, or it will not be added to S_I but $h(x) \neq h^*(x)$ are both less than $\frac{1}{4}\sqrt{\epsilon}$. The proof is complete.

□

Lemma 26 (Lemma 4.7 in Awasthi et al. (2017b)). *Consider Algorithm 1. With probability at least $1 - \exp(-\Omega(m_{\sqrt{\epsilon},\delta'}))$, $\overline{W}_I$, $\overline{W}_C$ and S_I all have size $\Theta(m_{\sqrt{\epsilon},\delta'})$. Consider Algorithm 3. With probability at least $1 - \exp(-\Omega(n_{\sqrt{\epsilon},\delta'}))$, $\overline{W}_I$, $\overline{W}_C$ and S_I all have size $\Theta(n_{\sqrt{\epsilon},\delta'})$.*

Proof. Notice that in both Algorithm 1 and 3, we have the input sample size $m_{\epsilon,\delta'}, n_{\epsilon,\delta'}$ lower bounded by $\Omega(m_{\epsilon,\delta'})$, the lemma follows from Lemma 4.7 of Awasthi et al. (2017b). □

Proposition 27 (Proposition 4 in Zeng and Shen (2022a)). *Suppose $\alpha \geq 0.7, \beta \geq 0.7$. Consider the* PRUNE-COMPARE-AND-LABEL *algorithm, i.e. Algorithm 4. If $|S| \geq (\frac{3}{\delta})^{1/1000}$, then with probability at least $1 - \delta$, it correctly sorts and labels all the instances in S. The label complexity is $O\big(\log|S| \cdot \log\log|S|\big)$, and the comparison complexity is given by $O\big(|S| \cdot \log^2|S|\big)$.*

Theorem 28 (Theorem 8 in Zeng and Shen (2022a)). *Suppose $\alpha \geq 0.7, \beta \geq 0.7$. With probability $1 - \delta$, Algorithm 3 runs in time $\mathrm{poly}(d, \frac{1}{\epsilon})$ and returns a classifier $h \in \mathcal{H}$ with error rate $\mathrm{err}_D(h) \leq \epsilon$. In addition, the label complexity is $O\Big(\log n_{\sqrt{\epsilon},\delta} \cdot \log\log n_{\sqrt{\epsilon},\delta}\Big)$, and the comparison complexity is $O\left(n_{\sqrt{\epsilon},\delta} \log^2 n_{\sqrt{\epsilon},\delta} + n_{\epsilon,\delta}\right)$.*