Efficient PAC Learning from the Crowd with Pairwise Comparisons

Shiwei Zeng [1] **Jie Shen** [1]

Abstract

We study crowdsourced PAC learning of threshold function, where the labels are gathered from a pool of annotators some of whom may behave adversarially. This is yet a challenging problem and until recently has computationally and query efficient PAC learning algorithm been established by Awasthi et al. (2017). In this paper, we show that by leveraging the more easily acquired pairwise comparison queries, it is possible to exponentially reduce the label complexity while retaining the overall query complexity and runtime. Our main algorithmic contributions are a comparison-equipped labeling scheme that can faithfully recover the true labels of a small set of instances, and a label-efficient filtering process that in conjunction with the small labeled set can reliably infer the true labels of a large instance set.

1. Introduction

The recent years have witnessed an unprecedented growth of demands in annotating large-scale data sets via crowdsourcing. On the empirical side, crowdsourcing has been serving as a successful tool to harness the crowd wisdom for annotating images (Deng et al., 2009) and natural language (Callison-Burch & Dredze, 2010), evaluating and debugging machine learning models (Chang et al., 2009), performing hybrid intelligent system diagnosis (Gomes et al., 2011; Tamuz et al., 2011), aiding our understanding in the conversation between human and intelligent systems (Doshi-Velez & Kim, 2017), to name a few (Vaughan, 2017).

Compared to the practical success, on the theoretical side, less is known about how to learn a hypothesis class with desired error rate from the noisy crowd annotations. This has motivated a flurry of recent studies on improving the quality of the annotations, such as pruning low-quality workers (Dekel & Shamir, 2009), adaptive task assignment (Khetan & Oh, 2016), and self-correction (Shah & Zhou, 2016).

Orthogonal to these approaches, in this paper, we consider the generalization ability of learning algorithms when instances are annotated by the crowd. In the *standard probably approximately correct (PAC) learning model* of Valiant (1984), it is assumed that there is a perfect hypothesis that incurs zero misclassification error (also known as the realizable setting), and the learner is always given correctly labeled data. In the context of crowdsourcing, the correct labels are not available but the learner has access to a large pool of workers and can give a task to multiple workers to aggregate the results, since some workers may behave adversarially (Karger et al., 2011; Ho et al., 2013). For example, the learner is allowed to query k workers on any instance x to gather a collection of noisy labels $Y(x) := \{y_1(x), \ldots, y_k(x)\}$ and may, for example, take the majority vote as the final label for x. The goal is to find a hypothesis close to the perfect one without requesting too many crowd annotations. The *crowdsourced PAC learning model* was recently put forward by Awasthi et al. (2017b), where they showed that any hypothesis class that can be efficiently learned in the standard PAC model can also be efficiently learned from the crowd even facing a high level of label noise. More importantly, it was shown that a carefully designed crowdsourcing algorithm collects an amount of labels that is only within a constant multiplicative factor of that needed in the standard PAC setting, which is called *a constant labeling overhead*.

Generally speaking, the approach of Awasthi et al. (2017b) crucially explores the wisdom of the crowd by adaptively gathering a manageable number of labels that can be aggregated into a set of correct labels, thus reducing the problem to the standard PAC learning setting. In this work, we consider the broad setting that the learner may additionally request comparison tags of any pair of instances to reduce the labeling cost, since in practice it is relatively easier to compare than to label. For example, in a recommendation application, an annotator may be asked which one of two movies she likes more; this is an easier task than to provide an exact rating, say, from 1 to 10 (Fürnkranz & Hüllermeier, 2010; Park et al., 2015). In general, we may consider that there is an underlying real-valued function $f^*(x)$ that indicates the user's preference to an instance x;

[1]Department of Computer Science, Stevens Institute of Technology, Hoboken, New Jersey, USA. Correspondence to: Shiwei Zeng <szeng4@stevens.edu>, Jie Shen <jie.shen@stevens.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

the label query mathematically reads as "is $f^*(x) > 0$?" and the comparison query on two instances (x, x') reads as "is $f^*(x) > f^*(x')$?". Such comparison-based model has been investigated in a variety of recent works in the context of PAC learning, aiming to mitigate the labeling cost or even break the label complexity lower bound known for the label-only setting; see, e.g. Kane et al. (2017); Hopkins et al. (2020b) and the references therein for algorithms designed under the non-crowdsourcing model.

In light of the above, our goal is to develop an efficient crowdsourcing algorithm that: 1) returns a hypothesis with low error rate, namely, the PAC guarantee; 2) significantly mitigates the labeling cost through the more easily acquired pairwise comparisons, say, an $o(1)$ labeling overhead; and 3) achieves overall query complexity that is of the same order of that under the standard PAC learning model of Valiant (1984), i.e. an overall $O(1)$ overhead.

Simultaneously achieving the three properties is highly non-trivial, since some natural approaches such as annotation followed by learning would lead to overhead blowing up with the sample size. In fact, we show that if the comparison overhead were not constrained, i.e. for each pair to be compared we can query as many workers as we want, then there would be a simple algorithm that achieves the first two properties, under the assumption that the majority of the crowd workers annotate according to the perfect hypothesis.[1]

Theorem 1 (Theorem 5, informal)**.** *Let n be the sample size. Assume the majority of the crowd are perfect. Then any hypothesis class $\mathcal{H}$ can be learned from the crowd with $o(1)$ labeling overhead and $O(\log^2 n)$ comparison overhead as long as $\mathcal{H}$ is PAC-learnable in the standard setting.*

One salient feature of the above result is that the labeling overhead becomes $o(1)$, whereas in the label-only crowdsourced PAC model of Awasthi et al. (2017b) this bound is $O(1)$. However, we note that the comparison overhead grows with the sample size n. Even growing logarithmically, such phenomenon is counter-intuitive and undesirable in large-scale learning problems.

1.1. Main results: an interleaving algorithm

In this paper, we present a novel algorithm which well-controls the comparison overhead while retaining the label efficiency and PAC guarantee of Theorem 1.

Theorem 2 (Theorem 8, informal)**.** *There is an efficient crowsourcing algorithm that learns $\mathcal{H}$ with $o(1)$ labeling overhead and $O(1)$ comparison overhead provided that the majority of the crowd are perfect.*

In stark contrast to the sample-size-varying comparison overhead of Theorem 1, the constant overhead we obtain here ensures that no matter what the size of data set is, the annotation cost remains unchanged.

[1] We phrase all the results by assuming some workers are perfect; it can be easily relaxed to being reliable; see Appendix A.

1.2. Overview of techniques

Our main algorithm is inspired by that of Awasthi et al. (2017b), where the high-level idea is to utilize a celebrated boosting framework of Schapire (1990) for interleaving annotation and learning. It works as follows: first, train a good hypothesis h_1 on a small set of correctly labeled instances; second, identify and label a set of instances that h_1 may misclassify and train another good hypothesis h_2; last, feed h_1 and h_2 to the boosting framework to obtain a hypothesis with the desired error rate. Our main algorithmic contributions fall into new *label-efficient* processes to construct the data sets needed to train h_1 and h_2 by leveraging pairwise comparisons. We highlight the new techniques below and refer the readers to Section 4 for a detailed description.

1) Comparison-equipped labeling. The first ingredient of our algorithm is a randomized sorting algorithm (Algorithm 1) which can sort all the instances in a given set S_1 with an $O(|S_1| \cdot \log |S_1|)$ pairwise comparisons. It then suffices to perform binary search to find a "threshold instance" across which the labels shift, which consumes $O(\log |S_1|)$ label queries. We run this routine on a small set S_1 to train the hypothesis h_1 with desired error rate while keeping a low labeling and comparison overhead.

2) Label-efficient filtering with pairwise comparisons. The second ingredient, also the core component of our algorithm, is a label-efficient filtering process (Algorithm 4) to identify instances that h_1 may make mistakes on. We construct an interval characterized by two "support instances" x^- and x^+ with the properties that a) with high probability, x^- is negative and x^+ is positive; b) instances outside the interval $[x^-, x^+]$ are likely to be good cases to test the performance of h_1, in the sense that their true labels can be *inferred* by combining a few comparison tags from the crowd and the label of x^- or x^+. To this end, given an instance set S, we sub-sample a set U and apply the comparison-equipped labeling to find proper support instances. The size of U is carefully chosen, such that i) it is large enough so that the size of the interval $[x^-, x^+]$ is small and thus most instances in S can be tested; and ii) it is not too large to blow up the overhead. We then show that the comparison complexity for identifying whether an instance is inside the interval, or outside that h_1 misclassifies, or outside that h_1 is correct on, is low. Since there are sufficient instances being tested, we are able to gather enough data to train h_2.

2. Related Works

We discuss three research lines that are closely related to this work. In particular, we clarify the crucial connection

and difference from noise-tolerant PAC learning, describe prominent crowd models and known results, as well as recent advances in PAC learning with comparisons.

Noise-tolerant learning. In real-world applications, the labels are often corrupted randomly or adversarially. To tackle such practical challenges, a variety of label noise models have been investigated in the literature, such as the random classification noise (Angluin & Laird, 1987), the Massart noise (Sloan, 1992; Massart & Nédélec, 2006), the Tsybakov noise (Tsybakov, 2004), and the agnostic/adversarial noise (Haussler, 1992; Kearns et al., 1992). It is, however, known that without distributional or large-margin assumptions on the instances, learning even the fundamental class of linear threshold functions is computationally hard (Feldman et al., 2006; Guruswami & Raghavendra, 2006; Diakonikolas & Kane, 2020). The only noise model that allows efficient and (almost) assumption-free algorithms is the random classification noise (Blum et al., 1996), yet it is recognized to be of little practical interest. Under distributional assumptions, a fruitful set of efficient PAC algorithms have been established, typically for the class of linear threshold functions; see, e.g. Kalai et al. (2005); Awasthi et al. (2017a); Zhang et al. (2020); Diakonikolas et al. (2020; 2021); Shen & Zhang (2021); Shen (2021a;b); Zhang & Li (2021) and the references therein. Notably, all these works approach the problem of learning linear threshold functions under the non-crowdsourcing scenario: for any given x, no matter how many times the learner queries its label, the obtained label is persistent even with noise. Orthogonal to these developed algorithms, we study the problem in the crowdsourcing setting and show that annotation aggregation enables a design of powerful PAC algorithms in the sense that our performance guarantee holds for general threshold functions *without* distributional assumptions on the instances (though we do need mild conditions on the crowd).

Learning from the crowd. Unlike traditional learning settings, a crowdsourced learner always has access to a large pool of workers who can provide annotations for a given instance. The noise in crowd mainly comes from the imperfectness of the workers. There are several types of crowd models considered in the literature. For example, one fundamental model assumes that a certain fraction of the workers are perfect, i.e. always labeling according to the true hypothesis, and the remaining could behave adversarially (Karger et al., 2011; Awasthi et al., 2017b), sometimes called the hammer-spammer model. Other works, such as Welinder et al. (2010); Ho et al. (2013); Khetan & Oh (2016), studied a more general crowd model where no perfect labeler exists, and the probability that a worker makes mistake depends on the given instance. The trouble of the latter model is that the annotators are too powerful and the algorithms had to either require golden labels (Ho et al., 2013), or only showed weak guarantee that as the sample size n goes to infinity, the misclassification error is small on average (Karger et al., 2011; Khetan & Oh, 2016); this could be vacuous since it is possible that the label of $\sqrt{n}$ instances can be entirely incorrect. To the best of our knowledge, the only known algorithm that offers PAC guarantee is due to Awasthi et al. (2017b) under the fundamental crowd model (and a Massart-noise model).

Learning with comparisons. Pairwise comparisons have been widely applied in practical problems such as preference learning in recommender systems (Fürnkranz & Hüllermeier, 2010; Park et al., 2015; Xu & Davenport, 2020) and ranking (Jamieson & Nowak, 2011; Heckel et al., 2019; Pananjady et al., 2020; Shah et al., 2019). More in line with this work is learning threshold functions with pairwise comparisons (Kane et al., 2017; Xu et al., 2017; Hopkins et al., 2020a;b), though these works are based on the non-crowdsourcing setting. The main finding is that it is possible to break the label complexity lower bound of label-only learning algorithms while still achieving PAC guarantees. We show similar phenomenon for crowdsourced PAC learning but will incorporate the comparison queries in a quite different way.

3. Preliminaries

We study the problem of learning threshold functions, with access to both labels and pairwise comparison tags from the crowd. Let $\mathcal{X}$ be the instance space, and $\mathcal{Y} := \{-1, 1\}$ be the label space. Let D be the joint distribution over $\mathcal{X} \times \mathcal{Y}$, and denote by D_X the marginal distribution over $\mathcal{X}$. Let $\mathcal{F} := \{f : \mathcal{X} \to \mathbb{R}\}$ be the class of real-valued functions. The class of threshold functions is given by $\mathcal{H} := \{h : x \mapsto \operatorname{sign}(f(x)), f \in \mathcal{F}\}$. For example, when $\mathcal{F} = \{f_w : x \mapsto w \cdot x\}$, the hypothesis class $\mathcal{H}$ is the class of halfspaces (also known as linear threshold functions). We focus on the classical realizable setting of Valiant (1984) where there exists a perfect hypothesis $h^* \in \mathcal{H}$ that incurs zero error, i.e. for any (x, y) drawn from D, $y = h^*(x)$. For any hypothesis $h : \mathcal{X} \to \mathcal{Y}$, its error rate is defined as $\operatorname{err}_{D_X}(h) := \Pr_{x \sim D_X}(h(x) \neq h^*(x))$. Let $f^* \in \mathcal{F}$ be the real-valued function associated with the perfect classifier h^*. For any pair of instances $(x, x') \in \mathcal{X} \times \mathcal{X}$, the underlying comparison function is thus given by $Z^*(x, x') = \operatorname{sign}(f^*(x) - f^*(x'))$. For the purpose of presentation transparency, for a given comparison function $Z(\cdot, \cdot)$, we will often write $x >_Z x'$ and $x <_Z x'$ in place of $Z(x, x') = 1$ and $Z(x, x') = -1$ respectively. When the function Z is clear from the context, we omit the subscript and just write $x > x'$ or $x < x'$.

Standard PAC learning. The terminology "standard PAC learning" is reserved for the setting of Valiant (1984), that is, there exists a perfect hypothesis h^* with zero error rate, and the learner is always given correctly labeled data $(x, h^*(x))$ where $x \sim D_X$. We are interested in the classes $\mathcal{H}$ that are

efficiently learnable in the standard PAC learning model.

Assumption 3. There exists an efficient algorithm $\mathcal{A}_{\mathcal{H}}$ which takes as input a target error rate $\epsilon \in (0,1)$, confidence $\delta \in (0,1)$, a set of $m_{\epsilon,\delta}$ correctly labeled instances S randomly drawn from D, and returns a hypothesis h such that, with probability at least $1-\delta$, $\mathrm{err}_{D_X}(h) \le \epsilon$. We call $\mathcal{A}_{\mathcal{H}}$ a standard PAC learner.

For example, when $\mathcal{H}$ is the class of halfspaces, a linear program that fits the training data is also a standard PAC learner (Maass & Turán, 1994). Note that in Assumption 3, we did not impose distributional assumptions on D_X.

The quantity $m_{\epsilon,\delta}$ in Assumption 3 characterizes the query complexity of $\mathcal{A}_{\mathcal{H}}$. In view of the classic result (Kearns & Vazirani, 1994), it suffices to pick

$$m_{\epsilon,\delta} = K \cdot \frac{1}{\epsilon}\Big(d \log \frac{1}{\epsilon} + \log \frac{1}{\delta}\Big), \tag{3.1}$$

where d is the Vapnik–Chervonenkis dimension of the hypothesis class $\mathcal{H}$ and $K > 1$ is some absolute constant[2]. We reserve $m_{\epsilon,\delta}$ for the above expression. In our analysis, we will also need the following quantity

$$n_{\epsilon,\delta} := K \cdot \frac{1}{\epsilon}\Big(d \log \frac{1}{\epsilon} + \Big(\frac{3}{\delta}\Big)^{\frac{1}{1000}}\Big), \tag{3.2}$$

which will characterize the query complexity of a comparison-equipped algorithm. The additive term $(\frac{3}{\delta})^{\frac{1}{1000}}$ is due to the fact that the failure probability of the sorting algorithm, specifically the randomized Quicksort algorithm we will use, only decays to zero in a rate inversely polynomial in the sample size. If there were sorting algorithms with failure probability decaying exponentially fast with the sample size, we would be able to choose $n_{\epsilon,\delta} = m_{\epsilon,\delta}$. Note also that there is nothing special on the exponent $\frac{1}{1000}$ of $\frac{1}{\delta}$: it can be made to be a constant arbitrarily close to 0 with the cost of increasing the comparison complexity with a constant multiplicative factor (see, e.g. Lemma 32).

Crowdsourced PAC learning. Let $\mathcal{P}_L$ be the uniform distribution over the pool of crowd workers who provide labels, and $\mathcal{P}_C$ be the uniform distribution of those providing comparisons. In the context of crowdsourcing, the learner does not have access to the correct labels but has instances randomly drawn from D_X, and it is allowed to query a worker $t \sim \mathcal{P}_L$ on the label of an instance, or worker $t' \sim \mathcal{P}_C$ on the comparison tag of two instances. For a given input (either a single instance or a pair), by querying multiple workers, it is possible to collect a set of different (noisy) annotations to aggregate (say, via majority voting). We say a crowdsourcing algorithm PAC learns $\mathcal{H}$ if for any given $\epsilon, \delta \in (0,1)$, by drawing m_S instances and collecting m_L labels and m_C comparison tags, it outputs a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ such that with probability at least $1-\delta$, $\mathrm{err}_{D_X}(h) \le \epsilon$. We call m_L the *label complexity*, and m_C the *comparison complexity*. The query complexity refers to the sum of m_L and m_C. The *labeling overhead* and *comparison overhead* are respectively defined as

$$\Lambda_L := \frac{m_L}{m_{\epsilon,\delta}} \quad \text{and} \quad \Lambda_C := \frac{m_C}{m_{\epsilon,\delta}}, \tag{3.3}$$

which characterize how the query complexity of a crowdsourced PAC learner compares to that of a standard PAC learner $\mathcal{A}_{\mathcal{H}}$. We say a crowdsourced PAC learner is query efficient if $\Lambda_L + \Lambda_C = O(1)$; if additionally $\Lambda_L = o(1)$, we say the learner is label efficient.

Crowd model. We consider the following fundamental model for the crowd: there exists a large pool of crowd workers, at least $\frac{1}{2}+\alpha$ fraction of whom are perfect that always return correct labels (i.e. they label according to h^*), while the other $\frac{1}{2}-\alpha$ fraction may perform adversarially. In other words, for a given instance, each time the learner queries a randomly chosen worker on its label, the obtained label is correct with probability at least $\frac{1}{2}+\alpha$. Likewise, we assume that $\frac{1}{2}+\beta$ fraction of the workers are perfect when providing the comparison tag. Throughout the paper, we assume that $\alpha \in (0, \frac{1}{2}]$ and $\beta \in (0, \frac{1}{2}]$, which ensures the correctness of the majority. It is worth mentioning that all of our analysis essentially holds for a much stronger noise model where the perfect workers can be replaced by *reliable* workers; we defer such extension to Appendix A.

Given a set of annotations $A = \{a_1, \ldots, a_n\}$ (either the labels or comparison tags), we define $\mathrm{Maj}(A)$ as the outcome of majority voting. Specifically, suppose h_1, h_2 and h_3 are three classifiers in $\mathcal{H}$. The function $\mathrm{Maj}(h_1, h_2, h_3)$ maps any instance x to a label y, which is the outcome of the majority vote of $h_1(x)$, $h_2(x)$ and $h_3(x)$. Let $Z_1, \ldots, Z_t$ be t comparison functions. We denote the set of the comparison tags given by them on a pair (x, x') by $Z_{1:t}(x, x')$, namely, $Z_{1:t}(x, x') = \{Z_1(x, x'), \ldots, Z_t(x, x')\}$.

We will use $\tilde{O}(f)$ to denote $O(f \cdot \mathrm{polylog}(f))$, use $\tilde{\Omega}(f)$ to denote $\Omega(f/\mathrm{polylog}(f))$, and use $\tilde{\Theta}(f)$ to denote a quantity that is between them. The notation $O_\delta(f)$ means we treat the parameter δ as a constant, which will be useful to simplify our discussion on the overhead.

3.1. A natural approach and the limitation

In order to utilize the comparison queries, we consider a natural primitive of "compare and label" (Ailon & Mohri, 2008; Xu et al., 2017). The idea is to use pairwise comparisons along with the well-known RANDOMIZED QUICKSORT to sort all the instances. It then remains to perform binary search of a threshold instance across which labels shift from

[2] For proper learning of Boolean-valued functions, there is a lower bound on the sample complexity that matches Eq. (3.1); see the discussion in Hanneke (2019).

Algorithm 1 COMPARE-AND-LABEL

Require: A set of instances $S = \{x_i\}_{i=1}^n$, confidence δ.
Ensure: A labeled set $\overline{S}$.

$k_1 \leftarrow \frac{1}{2\beta^2} \cdot \log \frac{3006|S|\log|S|}{\delta}$ and $k_2 \leftarrow \frac{1}{2\alpha^2} \cdot \log \frac{3\log|S|}{\delta}$.
Apply RANDOMIZED QUICKSORT on S: for each pair (x, x') being compared, query k_1 workers and take the majority. Let $\hat{S} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ be the sorted list.
$t_{\min} \leftarrow 1$, $t_{\max} \leftarrow n$.
while $t_{\min} < t_{\max}$ **do**
  $t \leftarrow (t_{\min} + t_{\max})/2$.
  Query k_2 workers to obtain their labels on $\hat{x}_t$, and let $\hat{y}_t$ be the majority vote.
  If $\hat{y}_t = 1$ **then** $t_{\max} \leftarrow t - 1$; **else** $t_{\min} \leftarrow t + 1$.
end while
For all $t' \geq t, \hat{y}_{t'} \leftarrow 1$; for all $t' < t, \hat{y}_{t'} \leftarrow -1$.
return $\overline{S} \leftarrow \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \ldots, (\hat{x}_n, \hat{y}_n)\}$.

-1 to 1; this step is label-efficient.

The success of the above approach hinges on the correctness of the comparison tags and labels. To this end, for each input, the crowdsourcing algorithm may aggregate the annotations from multiple workers in order to ensure that the majority vote is correct with high probability. The details of COMPARE-AND-LABEL are presented in Algorithm 1.

Proposition 4. *Consider the* COMPARE-AND-LABEL *algorithm. 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* $O\big(\frac{1}{\alpha^2} \cdot \log|S| \cdot \log\log|S|\big)$*, and the comparison complexity is given by* $O\big(\frac{1}{\beta^2} \cdot |S| \cdot \log^2|S|\big)$.

Equipped with the labeled set, it is straightforward to learn a classifier using the standard PAC learner $\mathcal{A}_{\mathcal{H}}$ as follows, where we recall that $n_{\epsilon,\delta}$ was defined in (3.2).

> **Natural Approach**: Draw a set S of size $n_{\epsilon,\delta}$ from D_X. Let $\overline{S} \leftarrow$ COMPARE-AND-LABEL(S). Return $\mathcal{A}_{\mathcal{H}}(\overline{S}, \epsilon, \delta)$.

We have the following performance guarantee.

Theorem 5 (Natural approach)**.** *With probability at least* $1 - \delta$*, the natural approach runs in time* $\mathrm{poly}(d, \frac{1}{\epsilon})$ *and returns a classifier* h *with error rate* $\mathrm{err}_{D_X}(h) \leq \epsilon$*. The label complexity is* $O\big(\frac{1}{\alpha^2} \cdot \log n_{\epsilon,\delta} \cdot \log\log n_{\epsilon,\delta}\big)$ *and the comparison complexity is* $O\big(\frac{1}{\beta^2} \cdot n_{\epsilon,\delta} \cdot \log^2 n_{\epsilon,\delta}\big)$*. Moreover, the labeling overhead* $\Lambda_L = \frac{1}{\alpha^2} \cdot \tilde{O}\big(\frac{\log(d/\epsilon)}{d/\epsilon}\big)$ *and the comparison overhead* $\Lambda_C = O_\delta\big(\frac{1}{\beta^2} \cdot \log^2 n_{\epsilon,\delta}\big)$.

Remark 6 (Label complexity and labeling overhead)**.** Awasthi et al. (2017b) considered label-only crowdsourced PAC learning and obtained a label complexity bound of $O(m_{\sqrt{\epsilon},\delta} \log m_{\sqrt{\epsilon},\delta} + m_{\epsilon,\delta})$ and a labeling overhead $\Lambda'_L = O(\sqrt{\epsilon} \log \frac{m_{\epsilon,\delta}}{\delta} + 1) = \tilde{O}_\delta\big(\sqrt{\epsilon}\log(d/\epsilon) + 1\big)$ under the condition that $\alpha \geq \Omega(1)$. First, observe that our label complexity scales as $\log(d/\epsilon)$, while theirs is proportional to d/ϵ which is exponentially worse. Second, in terms of labeling overhead, our results show that it tends to zero as ϵ/d goes to zero, while their labeling overhead stays as a non-zero constant: this is precisely what we mean by $o(1)$ labeling overhead in Theorem 1.

However, we observe that the comparison overhead Λ_C grows with the sample size. This is undesirable because when we demand a small error rate ϵ, the sample size will increase and hence the comparison cost per pair.

4. Main Algorithm and Performance Guarantee

The natural approach separates learning and annotation by first labeling all the data and then training a classifier. In this section, we present a more involved algorithm that interleaves learning and annotation to achieve constant comparison overhead. Inspired by the work of Awasthi et al. (2017b), we will use the celebrated boosting algorithm of Schapire (1990) as the starting point.

Theorem 7 (Boosting)**.** *For any* $p < \frac{1}{2}$ *and distribution* D_X*, consider three classifiers* $h_1(x)$*,* $h_2(x)$*,* $h_3(x)$ *satisfying the following. 1)* $\mathrm{err}_{D_X}(h_1) \leq p$*; 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_X *conditioned on* $\{x : h_1(x) = h^*(x)\}$*, and* D_I *denotes* D_X *conditioned on* $\{x : h_1(x) \neq h^*(x)\}$*; 3)* $\mathrm{err}_{D_3}(h_3) \leq p$ *where* D_3 *is* D_X *conditioned on* $\{x : h_1(x) \neq h_2(x)\}$*. Then* $\mathrm{err}_{D_X}(\mathrm{Maj}(h_1, h_2, h_3)) \leq 3p^2 - 2p^3$.

There are two salient features that we will exploit from the theorem. First, the theorem implies that in order to learn a hypothesis with a target error rate $\epsilon \in (0, 1)$, it suffices to learn three weak hypotheses each of which has error rate less than $p = \frac{\sqrt{\epsilon}}{2}$. Second, the framework is fairly flexible in the sense that one can apply any algorithm to obtain the three nontrivial hypotheses, as long as it enjoys PAC guarantee in the distribution-independent regime. In view of Assumption 3, it suffices to gather $m_{\sqrt{\epsilon}/2,\delta/3}$ *correctly labeled* instances from D_X, D_2 and D_3 respectively and invoke the standard PAC learner $\mathcal{A}_{\mathcal{H}}$. Our main algorithm, Algorithm 2, is designed in such a way that such an amount of high-quality labels are gathered without suffering an overwhelming query complexity (or, overhead).

Our main theoretical results are as follows.

Theorem 8 (Main results)**.** *Consider Algorithm 2. Assume* $\beta \geq c_0$ *for some absolute constant* $c_0 \in (0, 1/2]$*. For any* $\epsilon, \delta \in (0, 1)$*, with probability at least* $1 - \delta$*, it runs in time* $\mathrm{poly}(d, \frac{1}{\epsilon})$ *and returns a classifier* $\hat{h}$ *such that* $\mathrm{err}_{D_X}(\hat{h}) \leq \epsilon$*. The label complexity is* $\frac{1}{\alpha^2} \cdot \tilde{O}\big(\log \frac{d + \frac{1}{\delta}}{\epsilon}\big)$,

and the comparison complexity is $\tilde{O}\big(\frac{d+(1/\delta)^{\frac{1}{1000}}}{\epsilon}\big)$. *Moreover, the labeling overhead is* $\frac{1}{\alpha^2}\cdot\frac{\epsilon}{d}\cdot\tilde{O}\big(\log\frac{d}{\epsilon}\big)$, *and the comparison overhead is* $O_\delta\big(\sqrt{\epsilon}\cdot\log^2\frac{d}{\epsilon}+1\big)$.

Remark 9. The label complexity and labeling overhead are similar to what we obtained in Theorem 5, hence inheriting all the improvements over Awasthi et al. (2017b) as we described in Remark 6.

Remark 10. A crucial message of the main theorem is that the comparison overhead can now be upper bounded by a constant; this occurs when ϵ is sufficiently small relative to the VC dimension d, i.e. $\epsilon \le (\log d)^{-4}$. This is a mild condition since given a learning problem, the hypothesis class will be fixed and hence is the VC dimension d. Therefore, in the most interesting regime that ϵ is close to zero, i.e. $\epsilon \in (0, (\log d)^{-4})$, our algorithm PAC learns $\mathcal{H}$ with $o(\frac{1}{\alpha^2})$ labeling overhead and $O_\delta(1)$ comparison overhead. Here, we use $o(\frac{1}{\alpha^2})$ to emphasize that the labeling overhead may decay to zero as ϵ tends to zero. It is easy to see that if α is also lower bounded by some absolute constant, then the labeling overhead reads as $o(1)$.

Remark 11. Our results demonstrate a useful tradeoff between the complexity of the two query types: though the overall query complexity is comparable to that of Awasthi et al. (2017b), our algorithm needs drastically fewer labels. This is especially useful for label-demanding applications such as what we introduced in Section 1. One interesting question is the possibility to simultaneously reduce the label and overall query complexity. Unfortunately, there is evidence showing that this is in general impossible, though under the non-crowdsourcing setting; see e.g. Theorem 4.11 of Kane et al. (2017) and Theorem 11 of Xu et al. (2017).

Remark 12. Observe that in our definition of the overheads (see Eq. (3.3)), we compare the query complexity to that of a standard PAC learner that uses only labels. Hence, the notion of overheads highlights the savings of labels with pairwise comparisons. Another natural way to define the overheads is to compare the query complexity of a crowdsourcing algorithm to that of a non-crowdsourcing one which uses both labels and pairwise comparisons. This, however, appears subtle since it is unclear how the query complexity scales under the distribution-free setting. For example, combining Theorem 4.11 and Corollary 4.12 of Kane et al. (2017) results in a query complexity lower bound of $\Omega\big(d+\frac{1}{\epsilon}\big)$, yet its sharpness is open. We remark that if the query complexity were $\Theta\big(d+\frac{1}{\epsilon}\big)$, our main results would still follow for fixed VC dimension d. See Remark 31 in Appendix F for more details.

In the following, we elaborate on the design of the main algorithm, along with the performance guarantees. To reduce clutter, we omit the confidence parameter δ in the discussion and write $n_\epsilon := n_{\epsilon,\delta}$; it will be more convenient for the readers to further think of this quantity as being roughly $\Theta(\frac{1}{\epsilon})$.

Algorithm 2 Main Algorithm

Require: Hypothesis class $\mathcal{H}$, target error rate ϵ, confidence δ, a standard PAC learner $\mathcal{A}_\mathcal{H}$.
Ensure: A hypothesis $\hat{h}: \mathcal{X}\to\mathcal{Y}$.
Phase 1:
$\overline{S_1} \leftarrow$ COMPARE-AND-LABEL$(S_1, \delta/6)$, for an instance set S_1 of size $n_{\sqrt{\epsilon}/2,\delta/6}$ from D_X.
$h_1' \leftarrow \mathcal{A}_\mathcal{H}(\overline{S_1}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta}{6})$.
$h_1 \leftarrow$ ANTI-ANTI-CONCENTRATE$(h_1', \frac{\sqrt{\epsilon}}{2}, \frac{1}{2})$.
Phase 2:
$S_I \leftarrow$ FILTER$(S_2, h_1, \delta/12)$, for an instance set S_2 of size $\Theta(n_{\epsilon,\delta/12})$ drawn from D_X.
$S_C \leftarrow$ draw $\Theta(n_{\sqrt{\epsilon},\delta/12})$ instances from D_X.
$\overline{S_{\text{All}}} \leftarrow$ COMPARE-AND-LABEL$(S_I \cup S_C, \delta/12)$.
$\overline{W_I} \leftarrow \{(x,y)\in\overline{S_{\text{All}}} : y \ne h_1(x)\}$, $\overline{W_C} \leftarrow \overline{S_{\text{All}}}\backslash\overline{W_I}$.
Draw a sample set $\overline{W}$ of size $n_{\sqrt{\epsilon}/2,\delta/12}$ from a distribution that equally weights $\overline{W_I}$ and $\overline{W_C}$.
$h_2 \leftarrow \mathcal{A}_\mathcal{H}(\overline{W}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta}{12})$.
Phase 3:
$\overline{S_3} \leftarrow$ COMPARE-AND-LABEL$(S_3, \delta/6)$, for an instance set S_3 of size $n_{\sqrt{\epsilon}/2,\delta/6}$ from D_X conditioned on $h_1(x)\ne h_2(x)$.
$h_3 \leftarrow \mathcal{A}_\mathcal{H}(\overline{S_3}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta}{6})$.
return $\hat{h} \leftarrow \mathrm{Maj}(h_1, h_2, h_3)$.

The precise form of $n_{\epsilon,\delta}$ (defined in Eq. (3.2)) and the concrete choices of the confidence parameter will appear in the statements of all theorems and algorithms. It will also be useful to keep in mind that feeding COMPARE-AND-LABEL with a set of size n_ϵ would lead to comparison overhead growing with the sample size, but a set of size $n_{\sqrt{\epsilon}}$ would not since in this case the comparison complexity is $\tilde{O}(\frac{1}{\sqrt{\epsilon}})$ (see Theorem 5) because $m_{\epsilon,\delta} = \tilde{\Theta}(\frac{1}{\epsilon})$.

4.1. Analysis of Phase 1

In Phase 1, we aim to learn a nontrivial classifier h_1 with error rate at most $\frac{\sqrt{\epsilon}}{2}$ on D_X. This can easily be fulfilled since D_X is available to the learner. In fact, this phase is very similar to the natural approach, except for the number of instances being used: here we draw $n_{\sqrt{\epsilon}/2} = \tilde{\Theta}(\frac{1}{\sqrt{\epsilon}})$ samples from D_X and feed them to COMPARE-AND-LABEL, resulting in a nontrivial hypothesis h_1' with $\mathrm{err}_{D_X}(h_1') \le \frac{\sqrt{\epsilon}}{2}$ and an $O(1)$ comparison overhead.

We then invoke ANTI-ANTI-CONCENTRATE (Algorithm 3) to prevent the performance of h_1' from being too good. In particular, the new classifier h_1 is constructed in such a way that most of the time, it predicts as h_1' does but with a small chance, it predicts as $-h_1'$. This would make sure that $\mathrm{err}_{D_X}(h_1) = \Theta(\sqrt{\epsilon})$. While this step seems counterintuitive, it is inherently important for our algorithm, es-

Algorithm 3 ANTI-ANTI-CONCENTRATE

Require: A classifier h', a quantity $\eta \leq 1$ such that $\mathrm{err}_{D_X}(h') \leq \eta$, an absolute constant $c \leq 1$ with $\eta \leq c$.
Ensure: A classifier h with $\mathrm{err}_{D_X}(h) \in \big[c_1\eta, c_2\eta\big]$ where $c_1 = \min\big\{\frac{1}{2}, \frac{1-c}{2-c}\big\}$, $c_2 = \max\big\{1, c + \frac{1}{2}\big\}$.
Pick bias $\lambda \in [0,1]$ such that $(1-\lambda)(1-\frac{1}{2}\eta) = \frac{1}{2}\eta$.
Let h be as follows: for any given x, independently toss a coin; **if** outcome is HEADS (which happens with probability λ), **then** $h(x) \leftarrow h'(x)$, **else** $h(x) \leftarrow -h'(x)$.
return h.

pecially for Phase 2. Indeed, the construction of D_2 in Theorem 7 relies on the conditional distribution D_I, consisting of instances from D_X that h_1 misclassifies. Since the probability density of D_I is exactly the error rate of h_1 on D_X, in order to draw, say, n instances from D_I, we would have to sample at least $n/\mathrm{err}_{D_X}(h_1)$ data from D_X in view of the Chernoff bound. Therefore, if the error rate of h_1 were not bounded from below, the number of samples to be drawn from D_X would be unbounded as well. One may question that if h_1' is very good, say in reality we already have $\mathrm{err}_{D_X}(h_1') \leq \epsilon$, then why not terminating Algorithm 2 and just return h_1'. The trouble here is that we cannot test its error rate, because the correct labels are not available. In fact, by the Chernoff bound, it is not hard to see that we would have to obtain $\tilde{\Theta}(1/\mathrm{err}_{D_X}(h_1'))$ correctly labeled instances from D_X to guarantee that the empirical error rate of h_1' equals $\Theta(\mathrm{err}_{D_X}(h_1'))$, but a) the quantity $\mathrm{err}_{D_X}(h_1')$ is unknown, and b) acquiring such amount of correct labels would lead to overwhelming overhead (once $\mathrm{err}_{D_X}(h_1') = \epsilon$ this is exactly the amount needed by the natural approach).

We summarize the performance guarantee of Phase 1 below.

Proposition 13. *In Phase 1, with probability* $1 - \frac{\delta}{3}$, $\mathrm{err}_{D_X}(h_1) \in \big[\frac{\sqrt{\epsilon}}{6}, \frac{\sqrt{\epsilon}}{2}\big]$. *The label complexity is* $O\big(\frac{1}{\alpha^2} \cdot \log n_{\sqrt{\epsilon},\delta} \cdot \log\log n_{\sqrt{\epsilon},\delta}\big)$ *and the comparison complexity is* $O\big(\frac{1}{\beta^2} \cdot n_{\sqrt{\epsilon},\delta} \cdot \log^2 n_{\sqrt{\epsilon},\delta}\big)$.

4.2. Analysis of Phase 2

Phase 2 aims to obtain instances obeying the distribution D_2 as defined in Theorem 7. To this end, we need to find a set S_I of $\Theta(n_{\sqrt{\epsilon}})$ instances that h_1 will misclassify, and $\Theta(n_{\sqrt{\epsilon}})$ instances that h_1 will predict correctly. It then suffices to call the standard PAC learner $\mathcal{A}_{\mathcal{H}}$ to learn h_2. As we discussed in the preceding section, in order to guarantee the existence of S_I with such size, we have to draw a set S_2 of $\Theta(n_\epsilon)$ instances from D_X. In order to identify them, we have to design a computationally efficient algorithm. We remind that, with $\Theta(n_\epsilon)$ instances in S_2, we could not directly apply COMPARE-AND-LABEL on S_2 to identify S_I since it would lead to an undesirable overhead similar to that of the natural approach. We tackle this challenge by designing a novel algorithm termed FILTER (Algorithm 4). On the other hand, obtaining S_C is relatively easy: since the error rate of h_1 is upper bounded by $\frac{\sqrt{\epsilon}}{2}$, it makes correct prediction on most of the instances. Therefore, it suffices to draw $\Theta(n_{\sqrt{\epsilon}})$ samples from D_X and call COMPARE-AND-LABEL to obtain the correct label of all these data, which can be used to test the performance of h_1. By the Chernoff bound, it is guaranteed that we can find at least $(1-\frac{\sqrt{\epsilon}}{2})n_{\sqrt{\epsilon}} \geq \frac{1}{2}n_{\sqrt{\epsilon}}$ such instances to form S_C. In the following, we elaborate on the design of the FILTER algorithm, which is the key technical contribution of the paper.

Algorithm 4 FILTER

Require: An instance set S, classifier h with $\mathrm{err}_{D_X}(h) \in \big[\frac{\sqrt{\epsilon}}{6}, \frac{\sqrt{\epsilon}}{2}\big]$, confidence δ.
Ensure: A set S_I whose instances are misclassified by h.
$b \leftarrow \frac{4}{\sqrt{\epsilon}} \log\frac{16}{\delta} + (\frac{24}{\delta})^{1/1000}$.
Sample uniformly a subset $U \subset S$ of b instances.
$\overline{U} \leftarrow$ COMPARE-AND-LABEL$(U, \delta/3)$. Let x^- be the rightmost instance of those labeled as -1, and x^+ be the leftmost instance of those labeled as $+1$.
$S_I \leftarrow \emptyset$, $S_{\text{in}} \leftarrow \emptyset$, $N \leftarrow \frac{1}{\beta^2}\log\frac{1}{\epsilon}$.
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** $\big[\mathrm{Maj}(Z_{1:t}(x, x^-)) = \{x < x^-\}$ and $h(x) = -1\big]$ or $\big[\mathrm{Maj}(Z_{1:t}(x, x^+)) = \{x > x^+\}$ and $h(x) = 1\big]$, **then** ANS $\leftarrow$ NO and **exit** loop.
  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
$\overline{S_{\text{in}}} \leftarrow$ COMPARE-AND-LABEL$(S_{\text{in}}, \delta/3)$.
$S_I \leftarrow S_I \cup \{x : (x, y) \in \overline{U} \cup \overline{S_{\text{in}}}$ and $y \neq h(x)\}$.
return S_I.

From now on, we switch to the notation in Algorithm 4. The principle of FILTER is to utilize a modest amount of comparisons and a small amount of label queries to infer the correct label of a vast fraction of the instances of S (i.e. the set S_2 in Algorithm 2). To aid intuition, imagine that all the instances in S are ordered by the true comparison function $Z^*(\cdot,\cdot)$ and then mapped into the interval $[0,1]$ on the real line. There are two major stages in FILTER: a sub-sampling stage to obtain U and a filtering stage to obtain S_I.

In the sub-sampling stage, we uniformly sample a set $U \subset S$

and invoke the COMPARE-AND-LABEL algorithm to obtain a correctly sorted and labeled set $\overline{U}$, enabling us to find two support instances x^- and x^+. Roughly speaking, x^- is the negative instance that is "closest" to positive instances in U, and x^+ is the positive instance that is "closest" to negative. One important consequence of this step is that the label of the instances outside the interval $[x^-, x^+]$ can be inferred provided that we have truthful comparison tags. On the other hand, we are uncertain about the label of those inside the interval. Since our objective is to collect a sufficient number of instances to form S_I, we need to constrain the size of the points residing in the interval. While it seems that a constant length (i.e. it contains a constant fraction of points in S) suffices, we will show that it needs to be as small as $\frac{\sqrt{\epsilon}}{4}$ – this will be clarified when we explain the comparison complexity of the filtering step. Observe that such narrow interval can be guaranteed when the size of U is as large as $\frac{4}{\sqrt{\epsilon}} \log \frac{16}{\delta}$; see Lemma 24 in Appendix D.

The filtering stage is devoted to identifying good cases to test the performance of h. In particular, we hope to find instances whose true label can be inferred from a few noisy comparison tags, which can then be compared to the label predicted by h. In view of the availability of the support instances x^- and x^+, it suffices to gather comparison tags for (x, x^-) and (x, x^+) for all $x \in S \backslash U$. When x resides in the interval, we store it in S_{in} and invoke COMPARE-AND-LABEL on S_{in} after all instances in $S \backslash U$ have been visited. This step is query-efficient since U is constructed in such a way that $|S_{\text{in}}| \leq O(\sqrt{\epsilon}\,|S|) = O(n_{\sqrt{\epsilon}})$.

For those outside the interval, we think of them as good cases to test h. This is because Proposition 4 guarantees that U will be annotated correctly with high probability and hence, instances left to x^- are likely to be negative (likewise instances right to x^+ are likely to be positive). In other words, the labels of x^- and x^+ together with the comparison model we considered (see Section 3) would help infer the label of the instances outside the interval; these *inferred labels* will then be contrasted to the predictions made by h. Indeed, Step 10 of FILTER is exactly testing the matching between the inferred label and the predicted label from h. Such procedure would truthfully identify S_I provided that the comparison to the support instances returned by the crowd has good quality. In this regard, each instance x will be assigned to multiple workers, and the underlying question is how many times are sufficient to justify the correctness of $h(x)$. A straightforward approach is to acquiring $N = O(\frac{1}{\beta^2} \log \frac{|S|}{\delta})$ comparison tags per instance x to guarantee that the majority votes $\text{Maj}(Z_{1:N}(x, x^-))$ and $\text{Maj}(Z_{1:N}(x, x^+))$ are correct for all $x \in S$ with high probability. This, however, leads to a comparison complexity bound of $O(|S| \log |S|) = O(n_\epsilon \log n_\epsilon)$, which is comparable to the natural approach. In our algorithm, we explore the structure of the crowd and the error rate of the learned classifier to reduce it to $O(|S|) = O(n_\epsilon)$. In particular, since $\text{err}_{D_X}(h) = \Theta(\sqrt{\epsilon})$, the classifier essentially performs well on most of the data in S. Therefore, if there is one time stamp t such that the inferred label matches the predicted label from h, then it implies that h is correct on such instance x with high probability. On the other hand, if after N steps, the algorithm never found any group of workers whose majority vote combined with the label of x^- and x^+ agrees with the predicted label from h, then this is strong evidence that h misclassifies the current instance. Now we are in the position to spell out how to obtain the $\tilde{O}(n_{\sqrt{\epsilon}})$ comparison complexity. For those x that h classifies correctly, it follows that the algorithm will run only a few steps to find a group of workers whose comparison tags in allusion to the label of the support instances would agree with h. In fact, we show that for any such x, the number of comparison queries stays as a constant provided that $\beta \geq \Omega(1)$, say, 70% of the workers are perfect when providing comparison tags. Since all of such x consist of $1 - \frac{\sqrt{\epsilon}}{2}$ fraction of S (recall that $\text{err}_{D_X}(h) \leq \frac{\sqrt{\epsilon}}{2}$), the comparison complexity is $O(|S|)$. For those x that h misclassifies, the algorithm will need N comparison queries; since they consist of $\frac{\sqrt{\epsilon}}{2}$ fraction, we need a total of $O(\sqrt{\epsilon}\,|S|\,N)$ comparison queries. Lastly, for those inside the interval, the algorithm also runs N steps; yet, as the sub-sampling stage guarantees that the fraction is $O(\sqrt{\epsilon})$, it ends up with $O(\sqrt{\epsilon}\,|S|\,N)$ comparisons as well.

We summarize the performance guarantee of FILTER below.

Lemma 14. *Consider the* FILTER *algorithm. Assume that U is correctly labeled and $\beta \geq c_0$ for some absolute constant $c_0 \in (0, 1/2]$. Consider any given instance $x \in S$ outside the interval $[x^-, x^+]$. 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 $\frac{4c_0}{1+2c_0}$. For any instance $x \in S$ that falls into the interval $[x^-, x^+]$, it will be added to S_I with probability at most $\frac{1}{4}\sqrt{\epsilon}$.*

Proposition 15. *Consider the* FILTER *algorithm with $|S| = \Theta(n_{\epsilon,\delta})$. Assume $\beta \geq c_0$ for some absolute constant $c_0 \in (0, 1/2]$. Then, with probability at least $1 - \delta$, the algorithm returns an instance set S_I with size $\Theta(n_{\sqrt{\epsilon},\delta})$. The label complexity is $O\big(\frac{1}{\alpha^2} \cdot \log n_{\sqrt{\epsilon},\delta} \cdot \log\log n_{\sqrt{\epsilon},\delta}\big)$, and the comparison complexity is $O\big(n_{\sqrt{\epsilon},\delta} \cdot \log^2 n_{\sqrt{\epsilon},\delta} + n_{\epsilon,\delta}\big)$.*

These observations together lead to the following performance guarantee of Phase 2 in Algorithm 2. Readers may refer to Appendix D for a detailed proof.

Proposition 16. *Assume $\beta \geq c_0$ for some absolute constant $c_0 \in (0, 1/2]$. In Phase 2, with probability $1 - \frac{\delta}{3}$, $\text{err}_{D_2}(h_2) \leq \frac{\sqrt{\epsilon}}{2}$. The label complexity is $O\big(\frac{1}{\alpha^2} \cdot \log n_{\sqrt{\epsilon},\delta} \cdot \log\log n_{\sqrt{\epsilon},\delta}\big)$, and the comparison complexity is $O\big(n_{\sqrt{\epsilon},\delta} \cdot \log^2 n_{\sqrt{\epsilon},\delta} + n_{\epsilon,\delta}\big)$.*

4.3. Analysis of Phase 3

Given the hypotheses h_1 and h_2, we can draw instances from the disagreement region via rejection sampling.

Proposition 17. *In Phase 3, with probability* $1 - \frac{\delta}{3}$, $\mathrm{err}_{D_3}(h_3) \leq \frac{\sqrt{\epsilon}}{2}$. *In addition, the label complexity is* $O\big(\frac{1}{\alpha^2} \cdot \log n_{\sqrt{\epsilon},\delta} \cdot \log\log n_{\sqrt{\epsilon},\delta}\big)$ *and the comparison complexity is* $O\big(\frac{1}{\beta^2} \cdot n_{\sqrt{\epsilon},\delta} \cdot \log^2 n_{\sqrt{\epsilon},\delta}\big)$.

Observe that now Theorem 8 follows from Propositions 13, 16, 17, and Theorem 7; see Appendix F for the full proof.

5. Conclusion

We have shown that for any class of threshold functions that is efficiently PAC learnable under the standard setting, it is possible to efficiently learn it from the noisy crowd annotations, with an $o(1)$ labeling overhead and $O(1)$ comparison overhead. To this end, we have developed a set of new techniques, including a comparison-equipped labeling primitive and a label-efficient filtering process. It would be interesting to study agnostic crowdsourced PAC learning, or to study other query types (Balcan & Hanneke, 2012).

Acknowledgements

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

References

Ailon, N. and Mohri, M. An efficient reduction of ranking to classification. In *Proceedings of the 21st Conference on Learning Theory*, pp. 87–98, 2008.

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

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

Awasthi, P., Blum, A., Haghtalab, N., and Mansour, Y. Efficient PAC learning from the crowd. In *Proceedings of the 30th Annual Conference on Learning Theory*, pp. 127–150, 2017b.

Balcan, M. and Hanneke, S. Robust interactive learning. In *Proceedings of the 25th Conference on Learning Theory*, pp. 20.1–20.34, 2012.

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

Callison-Burch, C. and Dredze, M. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pp. 1–12, 2010.

Chang, J., Boyd-Graber, J. L., Gerrish, S., Wang, C., and Blei, D. M. Reading tea leaves: How humans interpret topic models. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, pp. 288–296, 2009.

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

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Diakonikolas, I. and Kane, D. M. Near-optimal statistical query hardness of learning halfspaces with Massart noise. *arXiv:2012.09720*, 2020.

Diakonikolas, I., Kontonis, V., Tzamos, C., and Zarifis, N. Learning halfspaces with Massart noise under structured distributions. In *Proceedings of the 33rd Annual Conference on Learning Theory*, pp. 1486–1513, 2020.

Diakonikolas, I., Kane, D. M., Kontonis, V., Tzamos, C., and Zarifis, N. Efficiently learning halfspaces with tsybakov noise. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 88–101, 2021.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *CoRR*, abs/1702.08608, 2017.

Feldman, V., Gopalan, P., Khot, S., and Ponnuswami, A. K. New results for learning noisy parities and halfspaces. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 563–574, 2006.

Feller, W. *An introduction to probability theory and its applications, volume 2*. John Wiley & Sons, 2008.

Fürnkranz, J. and Hüllermeier, E. Preference learning and ranking by pairwise comparison. In *Preference Learning*, pp. 65–82. Springer, 2010.

Gomes, R., Welinder, P., Krause, A., and Perona, P. Crowdclustering. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, pp. 558–566, 2011.

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

Hanneke, S. Proper PAC learning VC dimension bounds. Theoretical Computer Science Stack Exchange, 2019. URL https://cstheory.stackexchange.com/q/44252. (version: 2019-07-11).

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

Heckel, R., Shah, N. B., Ramchandran, K., and Wainwright, M. J. Active ranking from pairwise comparisons and when parametric assumptions don't help. *The Annals of Statistics*, 47(6):3099–3126, 2019.

Ho, C., Jabbari, S., and Vaughan, J. W. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 534–542, 2013.

Hopkins, M., Kane, D., and Lovett, S. The power of comparisons for actively learning linear classifiers. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, pp. 6342–6353, 2020a.

Hopkins, M., Kane, D., Lovett, S., and Mahajan, G. Noise-tolerant, reliable active classification with comparison queries. In *Proceedings of the 33rd Annual Conference on Learning Theory*, pp. 1957–2006, 2020b.

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

Kalai, A. T., Klivans, A. R., Mansour, Y., and Servedio, R. A. Agnostically learning halfspaces. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pp. 11–20, 2005.

Kane, D. M., Lovett, S., Moran, S., and Zhang, J. Active classification with comparison queries. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science*, pp. 355–366, 2017.

Karger, D. R., Oh, S., and Shah, D. Iterative learning for reliable crowdsourcing systems. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, pp. 1953–1961, 2011.

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

Kearns, M. J., Schapire, R. E., and Sellie, L. Toward efficient agnostic learning. In *Proceedings of the 5th Annual Conference on Computational Learning Theory*, pp. 341–352, 1992.

Khetan, A. and Oh, S. Achieving budget-optimality with adaptive schemes in crowdsourcing. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems*, pp. 4844–4852, 2016.

Maass, W. and Turán, G. How fast can a threshold gate learn? In *Proceedings of a workshop on computational learning theory and natural learning systems (vol. 1): constraints and prospects*, pp. 381–414, 1994.

Massart, P. and Nédélec, É. Risk bounds for statistical learning. *The Annals of Statistics*, pp. 2326–2366, 2006.

Pananjady, A., Mao, C., Muthukumar, V., Wainwright, M. J., and Courtade, T. A. Worst-case vs average-case design for estimation from fixed pairwise comparisons. *The Annals of Statistics*, 48(2):1072–1097, 2020.

Park, D., Neeman, J., Zhang, J., Sanghavi, S., and Dhillon, I. S. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1907–1916, 2015.

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

Shah, N. B. and Zhou, D. No oops, you won't do it again: Mechanisms for self-correction in crowdsourcing. In *Proceedings of the 33nd International Conference on Machine Learning*, pp. 1–10, 2016.

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

Shen, J. On the power of localized Perceptron for label-optimal learning of halfspaces with adversarial noise. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 9503–9514, 2021a.

Shen, J. Sample-optimal PAC learning of halfspaces with malicious noise. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 9515–9524, 2021b.

Shen, J. and Zhang, C. Attribute-efficient learning of halfspaces with malicious noise: Near-optimal label complexity and noise tolerance. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, pp. 1072–1113, 2021.

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

Sloan, R. H. Corrigendum to types of noise in data for concept learning. In *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory*, pp. 450, 1992.

Tamuz, O., Liu, C., Belongie, S. J., Shamir, O., and Kalai, A. Adaptively learning the crowd kernel. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 673–680, 2011.

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

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

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

Welinder, P., Branson, S., Belongie, S. J., and Perona, P. The multidimensional wisdom of crowds. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems*, pp. 2424–2432, 2010.

Xu, A. and Davenport, M. A. Simultaneous preference and metric learning from paired comparisons. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, pp. 454–465, 2020.

Xu, Y., Zhang, H., Singh, A., Dubrawski, A., and Miller, K. Noise-tolerant interactive learning using pairwise comparisons. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pp. 2431–2440, 2017.

Zhang, C. and Li, Y. Improved algorithms for efficient active learning halfspaces with Massart and Tsybakov noise. In *Proceedings of the 34th Annual Conference on Learning Theory*, pp. 4526–4527, 2021.

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

A. From Perfect Workers to Reliable Workers

Crowd model with perfect workers. All of the theoretical results in this paper are established based on the existence of perfect workers: $(\frac{1}{2}+\alpha)$ fraction of the pool of workers are perfect when providing labels, in the sense that they always label according to the perfect hypothesis h^*, while the remaining $(\frac{1}{2}-\alpha)$ may behave adversarially. Likewise, $(\frac{1}{2}+\beta)$ fraction always provide comparison tags according to Z^* and the remaining $(\frac{1}{2}-\beta)$ may behave adversarially.

In this section, we show that it is easy to relax the requirement of perfectness to reliability.

Definition 18 (Reliable worker)**.** A reliable worker t who provides labels is defined by a function $g_t : \mathcal{X} \to \mathcal{Y}$ satisfying the following condition: given any instance $x \sim D_X$,

$$\Pr\big(g_t(x) = h^*(x)\big) \geq 1 - \eta_t(x), \tag{A.1}$$

where the function $\eta_t(x) \in [0, \eta_L]$ for some given parameter $\eta_L < 1/2$. Likewise, a reliable worker t' who provides comparison tags is defined by a function $g'_t : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$ satisfying the following condition: given any pair of instances (x, x'),

$$\Pr\big(g'_t(x, x) = Z^*(x, x')\big) \geq 1 - \eta'_t(x, x'), \tag{A.2}$$

where the function $\eta'_t(x, x') \in [0, \eta_C]$ for some given parameter $\eta_C < 1/2$.

Note that the function $\eta_t(x)$ (likewise for $\eta'_t(x, x')$), namely the probability that a worker t will flip a given instance, may vary across workers and instances, and the only restriction is that the flipping probabilities are upper bounded by a parameter $\eta < 1/2$; thus this is a very flexible and strong noise model. In fact, such noise model is exactly the Massart noise (Sloan, 1988; Massart & Nédélec, 2006) which has been broadly studied in the literature under the non-crowdsourcing setting and only until recently have polynomial-time algorithms been established under distributional assumptions (Zhang et al., 2020; Diakonikolas et al., 2020); in addition, under the non-crowdsourcing setting, there is evidence that efficient learning with Massart noise without distributional assumptions is computationally hard (Diakonikolas & Kane, 2020).

Crowd model with reliable workers. We consider the following crowd model: $\frac{1}{2}+\alpha'$ fraction of the workers are reliable while the remaining $\frac{1}{2}-\alpha'$ are adversarial for label queries. Likewise, $\frac{1}{2}+\beta'$ are reliable while the remaining $\frac{1}{2}-\beta'$ are adversarial for comparison queries.

We show that *all of our analysis based on the perfect version can be easily extended to the reliable version*.[3] To simplify the discussion, we show how to do so for workers that provide labels; the case of comparison queries follows exactly the same reasoning.

Indeed, all of our analysis uses the following fact from the perfect version of crowd: for any given instance x, by querying a randomly chosen worker, we are able to obtain the correct label with probability at least $\frac{1}{2}+\alpha$, namely

$$\Pr_{t\sim\mathcal{P}_L}\big(h_t(x) = h^*(x)\big) \geq \frac{1}{2} + \alpha, \tag{A.3}$$

where we recall that $\mathcal{P}_L$ is the uniform distribution over the pool of workers that provide labels.

Now we can consider that the perfect workers are replaced with reliable workers, and it suffices to establish a condition similar to (A.3).

Given any instance $x \sim D_X$, we have

$$\begin{aligned}
\Pr_{t\sim\mathcal{P}_L}\big(g_t(x) = h^*(x)\big) &\geq \Pr_{t\sim\mathcal{P}}\big(g_t(x) = h^*(x),\ t \text{ is reliable}\big) \\
&= \Pr_{t\sim\mathcal{P}}\big(g_t(x) = h^*(x) \mid t \text{ is reliable}\big) \cdot \Pr_{t\sim\mathcal{P}}\big(t \text{ is reliable}\big) \\
&\geq \big(1 - \eta_t(x)\big) \cdot \Big(\frac{1}{2} + \alpha'\Big) \\
&\geq (1 - \eta) \cdot \Big(\frac{1}{2} + \alpha'\Big) \\
&= \frac{1}{2} + \Big[(1 - \eta) \cdot \Big(\frac{1}{2} + \alpha'\Big) - \frac{1}{2}\Big].
\end{aligned}$$

[3] The analysis of Awasthi et al. (2017b) for $\alpha > 0$ holds for the reliable version of crowd model as well.

Therefore, under the reliable version of the crowd model, all of our analysis holds but with a new parameter $\alpha = (1-\eta)\cdot\left(\frac{1}{2}+\alpha'\right)-\frac{1}{2}$, where the parameters $\eta \in [0,1/2)$ and $\alpha' \in (0,1/2]$ are such that $(1-\eta)\cdot\left(\frac{1}{2}+\alpha'\right)-\frac{1}{2} \in (0,1/2]$.

Similarly, the assumption that $\frac{1}{2}+\beta$ fraction of workers are perfect when providing comparison tags can be relaxed to that $\frac{1}{2}+\beta'$ fraction are reliable, and all of our analysis holds but with a new parameter $\beta = (1-\eta)\cdot\left(\frac{1}{2}+\beta'\right)-\frac{1}{2}$, where the parameters $\eta \in [0,1/2)$ and $\beta' \in (0,1/2]$ are such that $(1-\eta)\cdot\left(\frac{1}{2}+\beta'\right)-\frac{1}{2} \in (0,1/2]$.

B. Omitted Proof from Section 3.1

Proposition 19 (Restatement of Proposition 4)**.** *Consider the* COMPARE-AND-LABEL *algorithm, i.e. Algorithm 1. 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* $O\big(\frac{1}{\alpha^2}\cdot\log|S|\cdot\log\log|S|\big)$*, and the comparison complexity is given by* $O\big(\frac{1}{\beta^2}\cdot|S|\cdot\log^2|S|\big)$*.*

Proof. Recall that given any pair of instances $(x,x') \in \mathcal{X}\times\mathcal{X}$, if we randomly draw a worker $t \sim \mathcal{P}_C$, with probability at least $\frac{1}{2}+\beta$, the comparison tag is correct.

Let k_1 be the number of workers we query for each pair of instances in Algorithm 1. The probability that the majority of the k_1 tags is incorrect on a given pair (x,x') is

$$\Pr\left(Z^*(x,x')\cdot\sum_{j=1}^{k_1} Z_t(x,x') \leq 0\right),$$

where $Z_t(x,x')$ is the annotation from the j-th worker.

Without loss of generality, we assume the ground truth $Z^*(x,x') = -1$. It then follows that $\mathbb{E}_{t\sim\mathcal{P}_C}\left[Z_t(x,x')\right] = -2\beta < 0$. By Hoeffding's inequality, we have

$$\Pr\left(\sum_{t=1}^{k_1} Z_t(x,x') \geq k_1\cdot\mathbb{E}_t\left[Z_t(x,x')\right]+t\right) \leq e^{-\frac{2t^2}{k_1\cdot(b-a)^2}}, \tag{B.1}$$

where $a=-1$ and $b=1$ are the corresponding lower and upper bounds of each tag $Z_t(x,x')$. Let t be such that $k_1\cdot\mathbb{E}_t\left[Z_t(x,x')\right]+t=0$. Then

$$\Pr\left(\sum_{t=1}^{k_1} Z_t(x,x') \geq 0\right) \leq e^{-2k_1\beta^2}. \tag{B.2}$$

Let q_S be the total number of comparisons made by algorithm RANDOMIZED QUICKSORT, and denote by (x_l,x_l') the pair being compared in the l-th iteration. We apply the union bound and obtain that

$$\Pr\left(\bigcup_{l=1}^{q_S}\left[Z^*(x_l,x_l')\sum_{t=1}^{k_1} Z_t(x_l,x_l') \leq 0\right]\right) \leq \sum_{l=1}^{q_S} e^{-2k_1\beta^2} \leq q_S\cdot e^{-2k_1\beta^2}. \tag{B.3}$$

By setting

$$k_1 = \frac{1}{2\beta^2}\cdot\log\frac{3q_S}{\delta},$$

we have that with probability at least $\frac{\delta}{3}$, the set S is correctly sorted.

Now we upper bound the quantity q_S. Recall that by Lemma 32, with probability at least $1-\frac{1}{|S|^c}$, we have $q_S \leq (c+2)\,|S|\log_{8/7}|S|$. By our requirement on the size of S, $c=1000$. Thus, with probability at least $1-\frac{\delta}{3}$, $q_S \leq 1002\,|S|\log_{8/7}|S|$. Thus, it suffices to set

$$k_1 = \frac{1}{2\beta^2}\cdot\log\frac{3006\,|S|\log_{8/7}|S|}{\delta} \tag{B.4}$$

to ensure that with probability at least $1-\frac{2}{3}\delta$, the set S is correctly sorted. It is not hard to see that the number of comparisons is upper bounded by

$$k_1 \cdot q_S \leq \frac{1}{2\beta^2} \cdot \log \frac{3006\,|S| \log_{8/7}|S|}{\delta} \cdot 1002\,|S| \cdot \log_{8/7}|S| \leq O\Big(\frac{1}{\beta^2}\,|S| \cdot \log^2|S|\Big). \tag{B.5}$$

Likewise, for any given instance $x \in \mathcal{X}$, let $g_i(x)$ be the label given by a randomly selected worker $i \sim \mathcal{P}_L$. Recall that $g_i(x) = h^*(x)$ with probability at least $\frac{1}{2}+\alpha$. Similar to the previous analysis, by Hoeffding's inequality and the union bound (over the labeling of the $\log|S|$ instances), we have that

$$\Pr\left(\bigcup_{l=1}^{\log|S|}\left[h^*(x_l)\sum_{i=1}^{k_2} g_i(x_l) \leq 0\right]\right) \leq \sum_{l=1}^{\log|S|}\Pr\left(h^*(x_l)\sum_{i=1}^{k_2} g_i(x_l) \leq 0\right) \leq \log|S| \cdot e^{-2k_2\alpha^2} = \frac{\delta}{3},$$

provided that we set

$$k_2 = \frac{1}{2\alpha^2} \cdot \log\frac{3\log|S|}{\delta}. \tag{B.6}$$

The total number of calls to the labeling oracle equals

$$k_2 \cdot \log|S| = \frac{1}{2\alpha^2} \cdot \log\frac{3\log|S|}{\delta} \cdot \log|S| = O\Big(\frac{1}{\alpha^2} \cdot \log|S| \cdot \log\log|S|\Big). \tag{B.7}$$

By union bound, we have that with probability at least $1-\delta$, S is correctly sorted and labeled, which proves the desired result. □

Theorem 20 (Restatement of Theorem 5)**.** *With probability at least $1-\delta$, the natural approach runs in time $\mathrm{poly}(d, \frac{1}{\epsilon})$ and returns a classifier h with error rate $\mathrm{err}_{D_X}(h) \leq \epsilon$. The label complexity is $O\big(\frac{1}{\alpha^2}\cdot\log n_{\epsilon,\delta}\cdot\log\log n_{\epsilon,\delta}\big)$ and the comparison complexity is $O\big(\frac{1}{\beta^2}n_{\epsilon,\delta}\cdot\log^2 n_{\epsilon,\delta}\big)$. Therefore, the labeling overhead $\Lambda_L = \frac{1}{\alpha^2}\cdot\tilde{O}\big(\frac{\log(d/\epsilon)}{d/\epsilon}\big)$ and the comparison overhead $\Lambda_C = O_\delta\big(\frac{1}{\beta^2}\cdot\log^2 n_{\epsilon,\delta}\big)$.*

Proof. We note that by (3.2), $|S| = n_{\epsilon,\delta} \geq (\frac{3}{\delta})^{1/1000}$. Thus, we can apply Proposition 19 to show that with probability $1-\delta$, the set S is correctly labeled by Algorithm 1. This in allusion to Assumption 3 implies the PAC guarantee of the natural approach.

Now we give the labeling and comparison overhead. To this end, we first note that the label and comparison complexity inherits from Proposition 19. Thus, we have the label complexity and comparison complexity as follows:

$$m_L = O\Big(\frac{1}{\alpha^2}\cdot\log n_{\epsilon,\delta}\cdot\log\log n_{\epsilon,\delta}\Big) = \frac{1}{\alpha^2}\cdot\tilde{O}\Big(\log\frac{d+\frac{1}{\delta}}{\epsilon}\Big), \tag{B.8}$$

$$m_C = O\Big(\frac{1}{\beta^2}n_{\epsilon,\delta}\cdot\log^2 n_{\epsilon,\delta}\Big). \tag{B.9}$$

In the above expression, the $\tilde{O}(\cdot)$ notation hides poly-logarithmic factors of the argument.

Therefore, by $m_{\epsilon,\delta} \geq K \cdot \frac{d+\log(1/\delta)}{\epsilon}$, we have

$$\Lambda_L = \frac{m_L}{m_{\epsilon,\delta}} \leq \frac{1}{\alpha^2}\cdot\tilde{O}\Big(\frac{\epsilon}{d+\log(1/\delta)}\cdot\log\frac{d+1/\delta}{\epsilon}\Big) = \frac{1}{\alpha^2}\cdot\tilde{O}\Big(\frac{\log(d/\epsilon)}{d/\epsilon}\Big), \tag{B.10}$$

which goes to zero as ϵ goes to 0.

For the comparison overhead, we have

$$\Lambda_C = O\Big(\frac{1}{\beta^2}\cdot\frac{n_{\epsilon,\delta}\log^2 n_{\epsilon,\delta}}{m_{\epsilon,\delta}}\Big) = O_\delta\Big(\frac{1}{\beta^2}\cdot\log^2 n_{\epsilon,\delta}\Big). \tag{B.11}$$

To see that the algorithm runs in polynomial time, first, we recall that Algorithm 1 runs in polynomial time. In addition, the standard PAC learner $\mathcal{A}_\mathcal{H}$ is assumed to run in polynomial time. The proof is complete. □

C. Analysis of Phase 1

Proposition 21 (Restatement of Prop. 13)**.** *In Phase 1, with probability* $1-\frac{\delta}{3}$*, the classifier* h_1 *is such that* $\mathrm{err}_{D_X}(h_1) \in \left[\frac{1}{6}\sqrt{\epsilon}, \frac{1}{2}\sqrt{\epsilon}\right]$*. In addition, the label complexity is* $O\big(\frac{1}{\alpha^2}\cdot \log n_{\sqrt{\epsilon},\delta}\cdot \log\log n_{\sqrt{\epsilon},\delta}\big)$ *and the comparison complexity is* $O\big(\frac{1}{\beta^2}\cdot n_{\sqrt{\epsilon},\delta}\cdot \log^2 n_{\sqrt{\epsilon},\delta}\big)$.

Proof. Recall that $|S_1| = n_{\sqrt{\epsilon}/2,\delta/6}$. By Proposition 19, all the instances in $\overline{S_1}$ are correctly labeled with probability $1-\frac{\delta}{6}$. Thus, conditioned on this event, with probability at least $1-\frac{\delta}{6}$, $\mathcal{A}_{\mathcal{H}}(\overline{S_1}, \frac{\sqrt{\epsilon}}{2}, \frac{\delta}{6})$ returns a classifier h_1' with $\mathrm{err}_{D_X}(h_1') \le \frac{1}{2}\sqrt{\epsilon}$. By union bound (over the labeling of S_1 and the learner $\mathcal{A}_{\mathcal{H}}$), we know that with probability at least $1-\frac{\delta}{3}$, $\mathrm{err}_{D_X}(h_1') \le \frac{1}{2}\sqrt{\epsilon}$. Observe that the classifier h_1' corresponds to the parameters $\eta = \frac{1}{2}\sqrt{\epsilon}$ and $c = \frac{1}{2}$ in Algorithm 3. Hence, by Lemma 22, the obtained classifier h_1 is such that $\mathrm{err}_{D_X}(h_1) \in \left[\frac{1}{6}\sqrt{\epsilon}, \frac{1}{2}\sqrt{\epsilon}\right]$.

Note that we only query workers when training h_1'. Hence, by Proposition 19, the label complexity is $O\big(\frac{1}{\alpha^2}\cdot \log n_{\sqrt{\epsilon},\delta}\cdot \log \frac{\log n_{\sqrt{\epsilon},\delta}}{\delta}\big)$ and the comparison complexity is $O\big(\frac{1}{\beta^2}\cdot n_{\sqrt{\epsilon},\delta}\cdot \log n_{\sqrt{\epsilon},\delta}\cdot \log \frac{n_{\sqrt{\epsilon},\delta}}{\delta}\big)$. □

Lemma 22. *Consider Algorithm 3. The returned classifier* $h : \mathcal{X} \to \mathcal{Y}$ *satisfies* $\mathrm{err}_{D_X}(h) \in [c_1\eta, c_2\eta]$*, where* $c_1 = \min\left\{\frac{1}{2}, \frac{1-c}{2-c}\right\}$ *and* $c_2 = \max\left\{1, c+\frac{1}{2}\right\}$.

Proof. Denote $\xi := \mathrm{err}_{D_X}(h')$. Observe that

$$\begin{aligned}
&\Pr_{x\sim D_X}(h(x)\neq h^*(x))\\
&= \Pr_{x\sim D_X}(h(x)\neq h^*(x) \mid \text{HEADS})\cdot \Pr(\text{HEADS}) + \Pr_{x\sim D_X}(h(x)\neq h^*(x) \mid \text{TAILS})\cdot \Pr(\text{TAILS})\\
&= \Pr_{x\sim D_X}(h'(x)\neq h^*(x) \mid \text{HEADS})\cdot \Pr(\text{HEADS}) + \Pr_{x\sim D_X}(-h'(x)\neq h^*(x) \mid \text{TAILS})\cdot \Pr(\text{TAILS})\\
&= \xi\cdot\lambda + (1-\xi)(1-\lambda).
\end{aligned}$$

First, we consider that $\xi < \frac{1}{2}\eta$. Note that since $\xi\cdot\lambda \ge 0$ and $\xi < \frac{1}{2}\eta$, we have

$$\xi\cdot\lambda + (1-\xi)(1-\lambda) \ge 0 + (1-\frac{1}{2}\eta)(1-\lambda) = \frac{1}{2}\eta, \tag{C.1}$$

where the last step follows from our choice of λ. On the other hand, by using the fact that $0 \le \xi < \frac{1}{2}\eta$, we have

$$\xi\cdot\lambda + (1-\xi)(1-\lambda) \le \frac{1}{2}\eta\cdot\lambda + (1-\lambda) = \frac{1}{2}\eta\big(1-\frac{\frac{1}{2}\eta}{1-\frac{1}{2}\eta}\big) + \frac{\frac{1}{2}\eta}{1-\frac{1}{2}\eta} = \eta. \tag{C.2}$$

Therefore, we prove that the error rate of h falls into the range of $\left[\frac{1}{2}\eta, \eta\right]$ when $\xi < \frac{1}{2}\eta$.

Now, we consider that in reality, $\xi \in [\frac{1}{2}\eta, \eta]$. In this case, we have

$$\xi\cdot\lambda + (1-\xi)(1-\lambda) \ge 0 + (1-\eta)(1-\lambda) = \frac{1-\eta}{2-\eta}\cdot\eta \ge \frac{1-c}{2-c}\cdot\eta. \tag{C.3}$$

On the other hand, we also have

$$\xi\cdot\lambda + (1-\xi)(1-\lambda) \le \eta\lambda + (1-\frac{1}{2}\eta)(1-\lambda) = \eta\lambda + \frac{1}{2}\eta \le (c+\frac{1}{2})\eta, \tag{C.4}$$

where in the equality we use the setting of λ. Therefore, we prove that when $\frac{1}{2}\eta \le \xi \le \eta$, we still have $\mathrm{err}_D(h) \in \left[\frac{1-c}{2-c}\eta, (c+\frac{1}{2})\eta\right]$.

The result follows by taking the union of the intervals obtained for the two cases of ξ. □

D. Analysis of Phase 2

We will first analyze the performance of the FILTER algorithm, i.e. Algorithm 4. Then, we give a full analysis for Phase 2 of Algorithm 2.

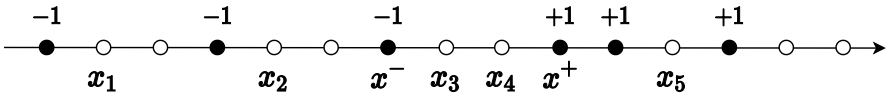

Figure 1. **Illustration of FILTER.** Each circle represents an instance in S, where we use solid circles to indicate the instances in $U \subset S$. We call the oracles to sort and label all the instances in U correctly. We use x^- to denote the rightmost instance in U that is labeled as -1, and use x^+ to denote the leftmost instance in U that is labeled as $+1$. We will think of $[x^-, x^+]$ as an interval, and instances outside the interval (e.g. x_1, x_5) are potentially informative, since we can combine the label of x^- or x^+ and the comparison tag to infer their labels (e.g. x_5 is likely positive). Thus, these points will be used to test the performance of h.

D.1. Analysis of FILTER

We will frequently discuss instances outside the interval $[x^-, x^+]$, which can be formally defined as follows: for the input set S in FILTER, the perfect comparison function $Z^*(\cdot,\cdot)$ gives a full ranking of all the instances. We then map these points to the interval $[0,1]$ of the real line by their order. An instance x is called outside the interval $[x^-, x^+]$ if its coordinate in the real line is less than that of x^- or greater than that of x^+. Note that this is for illustration purpose; in our algorithm we have no access to $Z^*(\cdot,\cdot)$.

In the subsequent analysis, we use the notation (S, h, δ) to denote the inputs to FILTER, which corresponds to the arguments $(S_2, h_1, \delta/12)$ of FILTER when invoked in Algorithm 2. We recall that $|S| = \Theta(n_{\epsilon,\delta})$ and $\mathrm{err}_{D_X}(h) = \Theta(\sqrt{\epsilon})$.

D.1.1. ANALYSIS OF SUB-SAMPLING

We first show that with high probability, U is correctly sorted and labeled. Then we will condition on this event and analyze the remaining steps of FILTER.

Lemma 23. *Consider* FILTER *(Algorithm 4) with input (S, h, δ). With probability $1 - \delta/8$, Algorithm 4 correctly sorts and labels the subset U. The label complexity of the sub-sampling step is $O\big(\frac{1}{\alpha^2} \cdot \log|U| \cdot \log\log|U|\big)$, and the comparison complexity of the sub-sampling step is $O\big(\frac{1}{\beta^2} \cdot |U| \cdot \log^2|U|\big)$.*

Proof. Note that $|U| = \frac{4}{\sqrt{\epsilon}} \log\frac{16}{\delta} + (\frac{24}{\delta})^{1/1000} \geq (\frac{24}{\delta})^{1/1000}$. Hence, the results follow from Proposition 19. □

Lemma 24. *Consider Algorithm 4. Suppose that the subset U is correctly labeled. With probability at least $1 - \delta/8$, the interval $[x^-, x^+]$ contains less than η fraction of the instances in S provided that $|U| \geq \frac{1}{\eta}\log\frac{16}{\delta}$.*

Proof. The algorithm uniformly samples a subset $U \subseteq S$ with b instances. Without loss of generality, let $x_1 < x_2 < \cdots < x_b$ be the instances in U.

Denote by E_i the event that interval $[x_i, x_{i+1}]$ contains more than a η fraction of the instances in S. Let ρ be the distance from x_i to the leftmost instance x_0 (note that all the points are evenly mapped to the interval $[0,1]$ and hence ρ is exactly the coordinate of x_i after mapping). Let L be the set of instances that are left to x_i, and R be those right to x_{i+1}. Then, E_i happens only if the distance of all cross-group points (i.e. for all $x \in L$ and $x' \in R$) is at least η. In the following, we only consider the case that $\rho \in [0, 1-\eta]$, because otherwise the fraction of the data contained in $[x_i, x_{i+1}]$ must be less than η. Therefore, for any $1 \leq i \leq b-1$, we have

$$\Pr\left(E_i\right) \leq \int_0^{1-\eta} \rho^{i-1} \cdot \left(1-\eta-\rho\right)^{b-i} d\rho \leq \frac{2(1-\eta)}{b+1} \cdot \left(1-\eta\right)^b \leq \frac{2}{b+1} e^{-b\eta},$$

where the last step follows from the inequality $1 - t \leq e^{-t}$ for all $t \geq 0$ and the fact that $1 - \eta \leq 1$.

By union bound over the intervals, we have

$$\Pr\Big(\bigcup_{i=1}^{b-1} E_i\Big) \leq \frac{2(b-1)}{b+1} \cdot e^{-b\eta} \leq 2 \cdot e^{-b\eta}.$$

Recall that we set the parameter $b \geq \frac{1}{\eta}\log\frac{16}{\delta}$, which implies that the above probability is upper bounded by $\delta/8$. Namely, the probability that the interval $[x^-, x^+]$ contains less than η fraction of the instances of S is at least $1 - \delta/8$. □

D.1.2. Analysis of the structure of S_I

Lemma 25 (Restatement of Lemma 14). *Consider Algorithm 4. Assume that the subset U is correctly labeled and $\beta \geq c_0$ for some absolute constant $c_0 \in (0, 1/2]$. Consider any given instance $x \in S$ outside the interval $[x^-, x^+]$. 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 $\frac{4c_0}{1+2c_0}$. For any instance $x \in S$ that falls into the interval $[x^-, x^+]$, it will be added to S_I with probability at most $\frac{1}{4}\sqrt{\epsilon}$.*

Proof. Consider the following two events that were involved in the Filtering step of the algorithm FLITER: (A_t) $\mathrm{Maj}(Z_{1:t}(x, x^-)) = \{x < x^-\}$ and $h(x) = -1$; (B_t) $\mathrm{Maj}(Z_{1:t}(x, x^+)) = \{x > x^+\}$ and $h(x) = 1$.

Case 1. The instance x is outside the interval and $h(x) = h^*(x)$. By assumption, both x^- and x^+ are labeled correctly. On the other hand, the probability that the comparison tag from a random worker $t \sim \mathcal{P}_C$ is correct is at least $\frac{1}{2} + \beta$. Thus, by the Chernoff bound, the majority vote of $N = \frac{1}{\beta^2} \log \frac{16}{\epsilon}$ comparison tags is correct with probability at least $1 - \frac{1}{4}\sqrt{\epsilon}$, i.e. $\mathrm{Maj}(Z_{1:N}(x, x^-)) = Z^*(x, x^-)$ and $\mathrm{Maj}(Z_{1:N}(x, x^+)) = Z^*(x, x^+)$. Consider that in reality, $Z^*(x, x^-) = \{x < x^-\}$. This immediately implies $\mathrm{Maj}(Z_{1:N}(x, x^-)) = \{x < x^-\}$. On the other hand, this in allusion to the comparison model in Section 3 also implies $h^*(x) = -1$. Since we are considering x such that $h(x) = h^*(x)$, we have $h(x) = -1$. Therefore, the event A_N occurs. Similarly, we can show that if in reality, $Z^*(x, x^+) = \{x > x^+\}$, then the event B_N occurs. Together, we have that either A_t or B_t occurs when $t = N$ if not earlier, and hence we will set ANS to NO and such instance will not be added to S_I.

Case 2. The instance x is outside the interval and $h(x) \neq h^*(x)$. Let $E_t = A_t \cup B_t$. We aim to show that with constant probability, E_t^c, the complement of E_t, occurs for all $t \leq N$, hence x will be added to S_I. To this end, it suffices to show that there exists at least a time stamp $t \leq N$, such that E_t happens with probability at most $O(1)$, because $\Pr(\cap E_t^c) \geq 1 - \Pr(\cup E_t)$. Observe we can rewrite A_t and B_t in conjunction with the condition $h(x) \neq h^*(x)$ as follows: (A_t) $\mathrm{Maj}(Z_{1:t}(x, x^-)) = \{x < x^-\}$ and $h^*(x) = 1$; (B_t) $\mathrm{Maj}(Z_{1:t}(x, x^+)) = \{x > x^+\}$ and $h^*(x) = -1$. Recall that when either A_t or B_t occurs, it must be the case that the majority vote of the t workers is incorrect, where we defined the incorrectness of a comparison tag in Section 3. It hence remains to upper bound the probability of such event. This can essentially be formed as a biased random walk, also known as the probability of ruin in gambling (Feller, 2008): we are given a random walk that takes a step to the right with probability $\frac{1}{2} + \beta$ (corresponding to a draw of a perfect worker) and takes a step to the left with probability $\frac{1}{2} - \beta$, and the question is how likely the walk will ever cross the origin to the left while taking N steps. By Lemma 33, we know that such probability is given by

$$\frac{1 - \left(\frac{1/2+\beta}{1/2-\beta}\right)^N}{1 - \left(\frac{1/2+\beta}{1/2-\beta}\right)^{N+1}} \leq \frac{\frac{1}{2} - \beta}{\frac{1}{2} + \beta} \leq \frac{1 - 2c_0}{1 + 2c_0}, \tag{D.1}$$

provided that $\beta \geq c_0$ for some absolute constant $c_0 > 0$. Therefore, with probability at least $\frac{4c_0}{1+2c_0}$, x will be added to S_I.

Case 3. The instance x falls into the interval. Using same argument as in Case 1, we know that the majority vote of N workers agrees with Z^* with probability at least $1 - \frac{1}{4}\sqrt{\epsilon}$. Therefore, such instance must be detected after the N-th iteration. □

We now have the following corollary regarding the size of S_I.

Corollary 26. *Consider Algorithm 4 with $|S| = \Theta(n_{\epsilon,\delta})$. Assume that the subset U is correctly labeled and $\beta \geq c_0$ for some absolute constant $c_0 \in (0, 1/2]$. With probability at least $1 - \delta/4$, $|S_I| = \Theta(\sqrt{\epsilon}\,|S|)$.*

Proof. For any $x \in S$, let r_x be a random variable that takes value 0 if x was not added to S_I and takes value 1 otherwise. Consider any instance $x \in S$ that is outside the interval $[x^-, x^+]$. By Lemma 25, we have

$$\begin{aligned}\Pr(r_x = 1) = {} & \Pr(r_x = 1 \mid h(x) = h^*(x)) \cdot \Pr(h(x) = h^*(x)) \\ & + \Pr(r_x = 1 \mid h(x) \neq h^*(x)) \cdot \Pr(h(x) \neq h^*(x)) \\ \leq {} & \frac{1}{4}\sqrt{\epsilon} \cdot 1 + 1 \cdot \frac{1}{2}\sqrt{\epsilon} \leq \sqrt{\epsilon}.\end{aligned}$$

On the other hand, it is not hard to see that

$$\Pr(r_x = 1) \geq 0 + \frac{4c_0}{1+2c_0} \cdot \frac{1}{6}\sqrt{\epsilon} = \Theta(\sqrt{\epsilon}).$$

Therefore, by the Chernoff bound, with probability $1 - e^{\Theta(-\sqrt{\epsilon}|S|)}$, we have $|S_I| = \Theta(\sqrt{\epsilon}(|S| - |S_{\text{in}}|) + M)$, where M denotes the number of instances we added from $U \cup S_{\text{in}}$. Now by Lemma 24 and our setting with $|U| \geq \frac{4}{\sqrt{\epsilon}} \log \frac{16}{\delta}$, we have $0 \leq |S_{\text{in}}| \leq \frac{1}{4}\sqrt{\epsilon}\,|S| \leq \frac{1}{4}\,|S|$ with probability at least $1 - \frac{\delta}{8}$. In addition, $0 \leq M \leq |U| + |S_{\text{in}}| = \Theta(\sqrt{\epsilon}\,|S|)$. These together with $|S| \geq \Omega(\frac{1}{\epsilon} \log \frac{8}{\delta})$ implies that with probability $1 - \delta/4$, $|S_I| = \Theta(\sqrt{\epsilon}\,|S|)$. □

D.1.3. Performance guarantee of Filter

A naive worst-case analysis would give query complexity bound of $O(|S|\,N)$. In the following, we show an improved result. The proof follows closely from Awasthi et al. (2017b), where the new ingredient is that we need to examine the instances within and outside the interval respectively. This is where Lemma 24 plays a role in controlling the overall query complexity.

Proposition 27 (Restatement of Prop. 15)**.** *Consider the* Filter *algorithm with $|S| = \Theta(n_{\epsilon,\delta})$. Assume that $\beta \geq c_0$ for some absolute constant $c_0 \in (0, 1/2]$. Then, with probability at least $1 - \delta$, the algorithm returns an instance set S_I with size $\Theta(n_{\sqrt{\epsilon},\delta})$. The label complexity is $O\big(\frac{1}{\alpha^2} \cdot \log n_{\sqrt{\epsilon},\delta} \cdot \log\log n_{\sqrt{\epsilon},\delta}\big)$, and the comparison complexity is $O\big(n_{\sqrt{\epsilon},\delta} \cdot \log^2 n_{\sqrt{\epsilon},\delta} + n_{\epsilon,\delta}\big)$.*

Proof. First, by Lemma 23 and Lemma 24 (and the setting of $|U|$), with probability $1 - \delta/3$, the sub-sampling step identifies x^- and x^+ such that: 1) x^- is negative and x^+ is positive; and 2) the interval $[x^-, x^+]$ contains less than $\frac{1}{4}\sqrt{\epsilon}\,|S|$ instances. In addition, since $|U| \leq O(n_{\sqrt{\epsilon},\delta})$, the label complexity and comparison complexity of sub-sampling are $O\big(\frac{1}{\alpha^2} \cdot \log n_{\sqrt{\epsilon},\delta} \cdot \log\log n_{\sqrt{\epsilon},\delta}\big)$ and $O\big(\frac{1}{\beta^2} n_{\sqrt{\epsilon},\delta} \cdot \log^2 n_{\sqrt{\epsilon},\delta}\big)$ respectively.

We condition on these happening and consider comparison complexity for three (overlapping) cases for $x \in S$: it falls into the interval $[x^-, x^+]$, it is such that $h(x) \neq h^*(x)$, it is such that $h(x) = h^*(x)$.

Case 1. For the instances inside the interval, the total number of comparison queries on these points is upper bounded by $\frac{1}{4}\sqrt{\epsilon}\,|S|\,N$.

Case 2. Since the error rate of h is $\Theta(\sqrt{\epsilon})$, we know that with probability at least $1 - e^{-\sqrt{\epsilon}|S|} \geq 1 - \delta/8$, the number of instances in S on which h disagrees with h^* is $\Theta(\sqrt{\epsilon}\,|S|)$. Hence, the total number of comparison queries is $\Theta(\sqrt{\epsilon}\,|S|\,N)$.

Case 3. Lastly, we consider $x \in S$ such that $h(x) = h^*(x)$. Let N_i be the expected number of queries we need until having i more correct comparison tags than the incorrect ones. Then,

$$N_1 \leq \big(\frac{1}{2} + \beta\big) \cdot 1 + \big(\frac{1}{2} - \beta\big)(N_2 + 1). \tag{D.2}$$

This is because with probability at least $\frac{1}{2} + \beta$, we obtain a correct comparison tag and stop, and with probability at most $\frac{1}{2} - \beta$, we get an incorrect tag and in the future, we must query the workers until we see 2 more correct tags than incorrect ones. On the other hand, it is not hard to see that $N_2 = 2N_1$, since in order to reach the status of N_2, we just repeat the process of getting N_1 twice. This combined with (D.2) implies that

$$N_1 \leq \frac{1}{2\beta} \leq \frac{1}{2c_0}$$

as long as $\beta \geq c_0$. Now using the Bernstein's inequality (see e.g. Appendix D of Awasthi et al. (2017b)), we know that with probability at least $1 - e^{-|S|} \geq 1 - \delta/8$, the total number of comparison queries on all x with $h(x) = h^*(x)$ is $O(|S|)$.

Combining all three cases and the fact that $|S| = \Theta(n_{\epsilon,\delta})$ and $N = \frac{1}{\beta^2} \log \frac{1}{\epsilon}$, we have that with probability $1 - \delta/4$, the comparison complexity of the "for $x \in S \backslash U$" loop is

$$\frac{1}{4}\sqrt{\epsilon}\,|S|\,N + \Theta(\sqrt{\epsilon}\,|S|\,N) + O(|S|) \leq O(|S|) = O(n_{\epsilon,\delta}). \tag{D.3}$$

To see why the inequality holds, note that $\sqrt{\epsilon}\log\frac{1}{\epsilon} \le O(1)$ and $\beta \ge \Omega(1)$, hence $\sqrt{\epsilon}N = \sqrt{\epsilon}\cdot\frac{1}{\beta^2}\log\frac{1}{\epsilon} \le O(1)$. Next, the algorithm invokes COMPARE-AND-LABEL on S_{in} whose size is upper bounded by $\sqrt{\epsilon}\,|S| = O(n_{\sqrt{\epsilon},\delta})$ in view of Lemma 24. Therefore, by Proposition 19, the label complexity and comparison of this step is $O\big(\frac{1}{\alpha^2}\cdot\log n_{\sqrt{\epsilon},\delta}\cdot\log\log n_{\sqrt{\epsilon},\delta}\big)$ and $O\big(\frac{1}{\beta^2}\cdot n_{\sqrt{\epsilon},\delta}\cdot\log^2 n_{\sqrt{\epsilon},\delta}\big)$ respectively. These, in conjunction with the label and comparison complexity of the sub-sampling step gives the announced results. □

D.2. Performance guarantee of Phase 2

We now switch to the notation in Algorithm 2.

Lemma 28 (Lemma 4.7 in Awasthi et al. (2017b)). *With probability $1-\delta/12$, $\overline{W}_I$ and $\overline{W}_C$ both have size $\Theta(n_{\sqrt{\epsilon},\delta})$.*

Proposition 29 (Restatement of Prop. 16). *Assume that $\beta \ge c_0$ for some absolute constant $c_0 \in (0, 1/2]$. In Phase 2, with probability $1-\frac{\delta}{3}$, $\mathrm{err}_{D_2}(h_2) \le \frac{1}{2}\sqrt{\epsilon}$. The label complexity is $O\big(\frac{1}{\alpha^2}\cdot\log n_{\sqrt{\epsilon},\delta}\cdot\log\log n_{\sqrt{\epsilon},\delta}\big)$, and the comparison complexity is $O\big(n_{\sqrt{\epsilon},\delta}\cdot\log^2 n_{\sqrt{\epsilon},\delta} + n_{\epsilon,\delta}\big)$.*

Proof. To see the query complexity in Phase 2, observe that we query the workers when invoking FILTER to obtain S_I, and when invoking COMPARE-AND-LABEL to obtain $\overline{S_{\text{All}}}$ where $\big|\overline{S_{\text{All}}}\big| = \Theta(n_{\sqrt{\epsilon},\delta})$. Therefore, with probability $1-\delta/6$, we obtain the announced query complexity in view of Proposition 27 and Proposition 19.

It remains to show that h_2 achieves error rate $\frac{1}{2}\sqrt{\epsilon}$ on the target distribution D_2. Let $d(x)$, $d_C(x)$ and $d_I(x)$ be the density functions of D, D_C and D_I respectively. Since the error rate of h_1 is $\Theta(\sqrt{\epsilon})$ (see Proposition 21), we have that for any x with $h_1(x) = h^*(x)$, $d(x) = d_C(x)\cdot(1-\Theta(\sqrt{\epsilon}))$; for any x with $h_1(x) \ne h^*(x)$, $d(x) = d_I(x)\cdot\Theta(\sqrt{\epsilon})$.

Let $N_C(x)$, $N_I(x)$, $M_C(x)$ 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. Let $d'(x)$ be the density function of the distribution D' underlying the empirical distribution of $\frac{1}{2}\overline{W_I} + \frac{1}{2}\overline{W_C}$.

We condition on Lemma 28, which occurs with probability $1-\delta/12$.

Case 1. x is such that $h_1(x) = h^*(x)$. We have

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_C(x)}{|\overline{W_C}|}\right] \overset{\zeta_1}{\ge} \frac{\mathbb{E}[M_C(x)]}{\Theta(n_{\sqrt{\epsilon},\delta})} \overset{\zeta_2}{\ge} \frac{\mathbb{E}[N_C(x)]}{\Theta(n_{\sqrt{\epsilon},\delta})} = \frac{|S_C|\cdot d(x)}{\Theta(n_{\sqrt{\epsilon},\delta})}$$
$$= \frac{\Theta(n_{\sqrt{\epsilon},\delta})\cdot d_C(x)\cdot(1-\Theta(\sqrt{\epsilon}))}{\Theta(n_{\sqrt{\epsilon},\delta})} \ge \Theta(d_C(x)) = \Theta(d_2(x)).$$

In the above expression, ζ_1 follows from Lemma 28, ζ_2 holds since $\overline{W_C}$ was obtained in such a way that the majority vote has high probability to be correct while S_C was just drawn from D_X, and the last step simply follows from the fact that D_2 is an equally weighted mixture of D_C and D_I.

Case 2. x is such that $h_1(x) \ne h^*(x)$. Similar to the first case, we can show that

$$d'(x) = \frac{1}{2}\,\mathbb{E}\left[\frac{M_I(x)}{|\overline{W_I}|}\right] = \frac{\mathbb{E}[M_I(x)]}{\Theta(n_{\sqrt{\epsilon},\delta})} \ge \frac{\mathbb{E}[N_I(x)]}{\Theta(n_{\sqrt{\epsilon},\delta})} \ge \frac{\frac{4c_0}{1+2c_0}\,|S_2|\,d(x)}{\Theta(n_{\sqrt{\epsilon},\delta})}$$
$$= \frac{\frac{4c_0}{1+2c_0}\,|S_2|\,d_I(x)\Theta(\sqrt{\epsilon})}{\Theta(n_{\sqrt{\epsilon},\delta})} = \frac{\Theta(n_{\sqrt{\epsilon},\delta})\cdot d_I(x)}{\Theta(n_{\sqrt{\epsilon},\delta})} = \Theta(d_I(x)) = \Theta(d_2(x)).$$

Now by the super-sampling lemma (Lemma 4.2 in Awasthi et al. (2017b)), we know that the obtained h_2 is such that with probability at least $1-\frac{\delta}{12}$, $\mathrm{err}_{D_2}(h_2) \le \frac{1}{2}\sqrt{\epsilon}$. The desired success probability of $1-\delta/3$ follows by considering the union bound of all the events. □

E. Analysis of Phase 3

Proposition 30 (Restatement of Prop. 17). *In Phase 3, with probability $1-\frac{\delta}{3}$, $\mathrm{err}_{D_3}(h_3) \le \frac{1}{2}\sqrt{\epsilon}$. In addition, the label complexity is $O\big(\frac{1}{\alpha^2}\cdot\log n_{\sqrt{\epsilon},\delta}\cdot\log\log n_{\sqrt{\epsilon},\delta}\big)$ and the comparison complexity is $O\big(\frac{1}{\beta^2}\cdot n_{\sqrt{\epsilon},\delta}\cdot\log^2 n_{\sqrt{\epsilon},\delta}\big)$.*

Proof. Similar to the proof of Proposition 21, in Phase 3, all instances in S_3 are correctly labeled with probability at least $1-\frac{\delta}{6}$. The label complexity is $O\big(\frac{1}{\alpha^2}\cdot\log n_{\sqrt{\epsilon},\delta}\cdot\log\log n_{\sqrt{\epsilon},\delta}\big)$ and the comparison complexity is $O\big(\frac{1}{\beta^2}\cdot n_{\sqrt{\epsilon},\delta}\cdot\log^2 n_{\sqrt{\epsilon},\delta}\big)$. We condition on this happening. Then with probability $1-\frac{\delta}{6}$, the base PAC learner $\mathcal{A}_{\mathcal{H}}$ returns a classifier with error rate $\leq\frac{1}{2}\sqrt{\epsilon}$ in view of Assumption 3. By union bound, with probability at least $1-\frac{\delta}{3}$, the error rate of h_3 is $\mathrm{err}_{D_3}(h_3)\leq\frac{1}{2}\sqrt{\epsilon}$. □

F. Proof of Theorem 8

Proof. By Proposition 21, Proposition 29, and Proposition 30, we have that with probability at least $1-\delta$, the error guarantees $\mathrm{err}_{D_X}(h_1)\leq\frac{1}{2}\sqrt{\epsilon}$, $\mathrm{err}_{D_2}(h_2)\leq\frac{1}{2}\sqrt{\epsilon}$ and $\mathrm{err}_{D_3}(h_3)\leq\frac{1}{2}\sqrt{\epsilon}$ hold simultaneously. Therefore, by Theorem 7, it follows that the classifier $\hat{h}:=\mathrm{Maj}(h_1,h_2,h_3)$ is such that $\mathrm{err}_D(\hat{h})\leq\frac{3}{4}\epsilon\leq\epsilon$ with probability at least $1-\delta$.

Note that each phase of Algorithm 2 runs in polynomial time. In particular, the COMPARE-AND-LABEL algorithm runs in polynomial time because its core component is RANDOMIZED QUICKSORT which runs in polynomial time. The FILTER algorithm runs in polynomial time because the sub-sampling step is efficient and the number of iterations for filtering is $N=\frac{1}{\beta^2}\log\frac{1}{\epsilon}$. Last, the base learner $\mathcal{A}_{\mathcal{H}}$ is assumed to be efficient.

Finally, we consider the query complexity of the main algorithm. By Propositions 21, 29 and 30, the overall label complexity is given by

$$m_L=O\Big(\frac{1}{\alpha^2}\cdot\log n_{\sqrt{\epsilon},\delta}\cdot\log\log n_{\sqrt{\epsilon},\delta}\Big)=\frac{1}{\alpha^2}\cdot\tilde{O}\Big(\log\frac{d+\frac{1}{\delta}}{\epsilon}\Big),\tag{F.1}$$

and the overall comparison complexity, under the assumption $\beta\geq c_0$, is given by

$$m_C=O\Big(n_{\sqrt{\epsilon},\delta}\cdot\log^2 n_{\sqrt{\epsilon},\delta}+n_{\epsilon,\delta}\Big)=\tilde{O}\Big(\frac{d+(1/\delta)^{\frac{1}{1000}}}{\sqrt{\epsilon}}\Big)+O\Big(\frac{1}{\epsilon}\Big(d\log\frac{1}{\epsilon}+(\frac{1}{\delta})^{\frac{1}{1000}}\Big)\Big)=\tilde{O}\Big(\frac{d+(1/\delta)^{\frac{1}{1000}}}{\epsilon}\Big).\tag{F.2}$$

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

$$\Lambda_L=O\Big(\frac{1}{\alpha^2}\cdot\frac{\log n_{\sqrt{\epsilon},\delta}}{m_{\epsilon,\delta}}\cdot\log\log n_{\sqrt{\epsilon},\delta}\Big)\leq\frac{1}{\alpha^2}\cdot\frac{\epsilon}{d+\log(1/\delta)}\cdot\tilde{O}\Big(\log\frac{d+\frac{1}{\delta}}{\epsilon}\Big)=\frac{1}{\alpha^2}\cdot\frac{\epsilon}{d}\cdot\tilde{O}\Big(\log\frac{d}{\epsilon}\Big),\tag{F.3}$$

and

$$\Lambda_C=O\Big(\frac{n_{\sqrt{\epsilon},\delta}}{m_{\epsilon,\delta}}\cdot\log^2 n_{\sqrt{\epsilon},\delta}+\frac{n_{\epsilon,\delta}}{m_{\epsilon,\delta}}\Big)\leq O\Big(\sqrt{\epsilon}\cdot\frac{d+(1/\delta)^{\frac{1}{1000}}}{d+\log(1/\delta)}\cdot\log^2\big(\frac{d+1/\delta}{\epsilon}\big)+\frac{d+(1/\delta)^{\frac{1}{1000}}}{d+\log(1/\delta)}\Big).\tag{F.4}$$

Recall that by the definition of $n_{\epsilon,\delta}$ and $m_{\epsilon,\delta}$ in Section 3, we have $m_{\epsilon,\delta}=\Theta(n_{\epsilon,\delta})=\Theta(\frac{d}{\epsilon}\log\frac{1}{\epsilon})$ when δ is a constant (note that assuming δ as a constant only simplifies our discussion). In this case, we can see that

$$\Lambda_C\leq O_\delta\Big(\sqrt{\epsilon}\cdot\log^2\frac{d}{\epsilon}+1\Big).$$

The theorem is proved. □

Remark 31. Observe that the denominator we use in the analysis, i.e. $\Omega\big(\frac{d}{\epsilon}\big)$, is the query complexity lower bound of a PAC learner that uses only labels, thus highlighting the saving of labels with access to the comparison queries. On the other side, if we were to compare our query complexity to the lower bound of a comparison-equipped algorithm in the non-crowdsourcing setting, we can combine Theorem 4.11 and Corollary 4.12 of (Kane et al., 2017) to get a (probably loose) lower bound $\Omega(d+\frac{1}{\epsilon})$. With this bound, it is easy to see that now Eq. (F.3) and Eq. (F.4) become

$$\text{(F.3')}\quad\Lambda_L\leq\frac{1}{\alpha^2}\cdot\frac{1}{d+1/\epsilon}\cdot\tilde{O}_\delta(\log\frac{d}{\epsilon}),$$

$$\text{(F.4')}\quad\Lambda_C\leq O\Big(\frac{\frac{1}{\sqrt{\epsilon}}(d+(1/\delta)^{\frac{1}{1000}})}{d+1/\epsilon}\cdot\log^2\frac{d+1/\delta}{\epsilon}+\frac{\frac{1}{\epsilon}(d+(1/\delta)^{\frac{1}{1000}})}{d+1/\epsilon}\Big)=O_\delta\Big(\frac{d/\sqrt{\epsilon}}{d+1/\epsilon}\cdot\log^2\frac{d}{\epsilon}+\frac{d/\epsilon}{d+1/\epsilon}\Big).$$

When $\epsilon\to 0$, we have $\Lambda_L=o_\delta(1)$ and for fixed d, $\Lambda_C=O_\delta(1)$. Hence, the performance guarantees we highlighted in Remark 10 still hold for fixed d, as we mentioned in Remark 12.

G. Useful Lemmas

We record some standard results in this section.

Lemma 32 (High-probability bound for RANDOMIZED QUICKSORT)**.** *With probability at least $1 - 1/m^c$ for any constant $c > 1$, the following holds. Given an instance set S with m elements, the comparison complexity of sorting all the elements by* RANDOMIZED QUICKSORT *is $(c+2)m \log_{8/7} m$.*

Proof. QUICKSORT is a recursive algorithm: in each round, it picks a pivot, splits the problem into two subsets, and recursively calls itself on each subset. The program keeps doing this until all the recursive calls contain at most one element. We consider RANDOMIZED QUICKSORT in our algorithm. Note that RANDOMIZED QUICKSORT differs from QUICKSORT only in the way it picks the pivots: in each round, it picks a random element in set S.

Consider a special element $t \in S$. Let L_i be the size of input in the ith level of recursion that contains t. Obviously $L_0 = m$, and we have

$$\mathbb{E}[L_i | L_{i-1}] < \frac{1}{2} \cdot \frac{3}{4} L_{i-1} + \frac{1}{2} L_{i-1} \leq \frac{7}{8} X_{i-1},$$

because with probability $\frac{1}{2}$, the pivot ranks between $\frac{1}{4}L_{i-1}$ and $\frac{3}{4}L_{i-1}$; and with probability $\frac{1}{2}$, the size of the subset does not shrink significantly. By the tower rule $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]]$, it follows that

$$\mathbb{E}[L_i] = \mathbb{E}_y[L_i | L_{i-1} = y] \leq \mathbb{E}_{L_{i-1}=y}\left[\frac{7}{8} y\right] = \frac{7}{8}\,\mathbb{E}\left[L_{i-1}\right] \leq \left(\frac{7}{8}\right)^i \mathbb{E}[L_0] = \left(\frac{7}{8}\right)^i m.$$

Let $M = (c+2) \cdot \log_{8/7} m$ for some constant $c > 0$. Then, the above inequality gives

$$\mathbb{E}[L_M] \leq \left(\frac{7}{8}\right)^M \cdot m \leq \frac{1}{m^{c+2}} \cdot m = \frac{1}{m^{c+1}}.$$

Applying Markov's inequality, we have

$$\Pr[L_M \geq 1] \leq \frac{\mathbb{E}[L_M]}{1} \leq \frac{1}{m^{c+1}},$$

which denotes the probability that element t participates in more than M recursive calls. Taking a union bound over all m elements, the probability that any element participates in more than M recursive calls is at most $\left(1/m^{c+1}\right) \cdot m = 1/m^c$. In other words, with probability at least $1 - 1/m^c$, the total number of pairwise comparisons performed is upper bounded by $m \cdot M = (c+2)m \log_{8/7} m$. □

Lemma 33 (Probability of Ruin (Feller, 2008))**.** *Consider a player who starts with i dollars against an adversary that has N dollars. The player bets one dollar in each gamble, which he wins with probability p. The probability that the player ends up with no money at any point in the game is*

$$\frac{1 - \left(\frac{p}{1-p}\right)^N}{1 - \left(\frac{p}{1-p}\right)^{N+i}}.$$

Lemma 34 (Chernoff bound)**.** *Let $Z_1, Z_2, \ldots, Z_n$ be n independent random variables that take value in $\{0, 1\}$. Let $Z = \sum_{i=1}^n Z_i$. For each Z_i, suppose that $\Pr(Z_i = 1) \leq \eta$. Then for any $\alpha \in [0, 1]$*

$$\Pr\left(Z \geq (1+\alpha)\eta n\right) \leq e^{-\frac{\alpha^2 \eta n}{3}}.$$

When $\Pr(Z_i = 1) \geq \eta$, for any $\alpha \in [0, 1]$

$$\Pr\left(Z \leq (1-\alpha)\eta n\right) \leq e^{-\frac{\alpha^2 \eta n}{2}}.$$

The above two probability inequalities hold when η equals exactly $\Pr(Z_i = 1)$.