Learning Halfspaces with Pairwise Comparisons: Breaking the Barriers of Query Complexity via Crowd Wisdom

Shiwei Zeng
Stevens Institute of Technology
szeng4@stevens.edu

Jie Shen
Stevens Institute of Technology
jie.shen@stevens.edu

November 3, 2020

Abstract
In this paper, we study the problem of efficient learning of halfspaces under Massart noise. This is a challenging noise model and only very recently has significant progress been made for problems with sufficient labeled samples. We consider the more practical yet less explored setting where the learner has a limited availability of labeled samples, but has extra access to pairwise comparisons. For the first time, we show that even when both labels and comparisons are corrupted by Massart noise, there is a polynomial-time algorithm that provably learns the underlying halfspace with near-optimal query complexity and noise tolerance, under the distribution-independent setting. In addition, we present a precise tradeoff between label complexity and comparison complexity, showing that the former is order of magnitude lower than the latter, making our algorithm especially suitable for label-demanding problems. Our central idea is to appeal to the wisdom of the crowd: for each instance (respectively each pair of instances), we query the noisy label (respectively noisy comparison) for a number of times to alleviate the noise. We show that with a novel design of an filtering process, together with a powerful boosting framework, the total query complexity is only a constant times more than learning from experts.

Keywords: PAC learning, Massart noise, crowdsourcing, boosting

1 Introduction
Learning of halfspaces has been one of the most important problems in machine learning dating back to the Perceptron algorithm [Rosenblatt, 1958]. In noise-free settings, where all the labels are consistent with some underlying hypothesis, it is well-known that the problem can be solved in polynomial time by linear programming [Maass and Turán, 1994; Cristianini and Shawe-Taylor, 2000]. However, in real-world scenarios, labels are often corrupted, which has driven a large body of works on designing computationally efficient and noise-tolerant algorithms. For instance, [Blum et al., 1996] proposed a polynomial-time algorithm that recovers the underlying hypothesis in presence of random classification noise, where each label is flipped with probability exactly equal to $\frac{1}{2} - \alpha$ for some $\alpha \in (0, \frac{1}{2}]$. A more realistic model, called Massart noise (also known as bounded noise), was proposed in [Sloan, 1988, 1992; Massart and Nédélec, 2006] as a generalization to the random classification noise. In this noise model, the label of each instance $x_i$ could be flipped with different probability $\frac{1}{2} - \alpha_i \leq \frac{1}{2} - \alpha$, where $\alpha_i$ is unknown but its lower bound $\alpha \in (0, \frac{1}{2}]$ is given to the learner. The quantity $\eta := \frac{1}{2} - \alpha$ is termed the Massart noise rate, and $\alpha$ is called the Massart parameter. Due to the highly asymmetric nature of this noise type, however, it is much more challenging to design computationally and query efficient algorithms.
Generally speaking, most of the existing theoretical analyses of learning halfspaces with Massart noise are based on the framework of probably approximately correct (PAC) learning, and can be categorized as either distribution-dependent or distribution-independent in terms of the instance space. In the distribution-dependent setting, there have been a considerable number of works that developed polynomial-time algorithms that can achieve any error rate $\epsilon > 0$, though with different noise tolerance and label complexity. For example, Awasthi et al. [2015] initiated the study of distribution-dependent PAC learning with Massart noise by assuming that the instance distribution is uniform over a unit sphere, and obtained a computationally efficient algorithm which can tolerate small constant noise rate $\eta \leq 1.8 \times 10^{-6}$. The same distributional condition was also studied in Zhang et al. [2017] and Yan and Zhang [2017], where they successfully handled any noise rate $\eta < \frac{1}{2}$ and obtained label complexity polynomial in $\text{poly}(d, \frac{1}{\alpha}, \epsilon)$. Awasthi et al. [2016] investigated whether there exist polynomial algorithms that work for the broad family of isotropic log-concave distributions, which covers prominent distributions such as logistic and normal distribution. They showed that using the technique of polynomial regression, it is possible to obtain label complexity bound of $\frac{d^{2}\text{poly}(1/\alpha)}{\text{poly}(\epsilon)}$, which was the first result that has an explicit dependence on $\alpha$ under this family of distributions. Recently, Zhang et al. [2020] and Diakonikolas et al. [2020] concurrently improved this result to $O(d/\alpha^c)$, for some small constant $c \geq 4$. It is worth noting that a label complexity lower bound of $\Omega(d/\alpha^2)$ was provided in Yan and Zhang [2017] for uniform distribution over the unit sphere, whereas it remains an open question of whether it is possible to obtain such bound under isotropic log-concave distributions [Zhang et al., 2020].

Compared to the rich literature of distribution-dependent PAC learning with Massart noise, less is known for the significantly more challenging setting of distribution-independent learning. In fact, this is a long-standing open problem and only very recently has a polynomial-time algorithm been developed by Diakonikolas et al. [2019], which obtains error rate of $\eta + \epsilon$ with label complexity $O(1/\gamma^3)$ where $\gamma > 0$ is the large margin parameter of the underlying hypothesis. This bound was further improved to $O(1/\gamma^2)$ with same error rate in Chen et al. [2020]. This naturally raises the first question: can we design a computationally efficient algorithm under the distribution-independent setting, such that it can achieve an error rate of $\epsilon$?

On the other spectrum, the success of all the aforementioned algorithms hinges on the condition that there are sufficient labeled samples available to the learner, which could be stringent on both practical and theoretical aspects. On the practical side, labels are extremely expensive to gather for many applications. For example, in medical diagnosis, a precise evaluation of an individual’s health is difficult even for medical specialists, but it is often easier to compare the status of two different patients. On the theoretical side, a number of recent works have discovered the fundamental limitation of label-only algorithms, and have suggested to leverage pairwise comparisons as a remedy Kane et al. [2017], Xu et al. [2017], Hopkins et al. [2019, 2020]. Kane et al. [2017] considered the noise-free setting and built up a framework on which learning of halfspaces costs exponentially fewer labels as long as a combinatorial quantity $k$ is small. Concurrently, Xu et al. [2017] proposed a noise-tolerant algorithm using $O(\log d)$ labels and $O(d)$ comparisons, by assuming the underlying distribution is isotropic log-concave and the comparison tags are corrupted by adversarial noise. Later, Hopkins et al. [2020] extended the framework of Kane et al. [2017] to handle the setting that both labels and comparison tags are corrupted by Massart noise. Under $s$-concave distributions, they obtained query complexity $O(k/\alpha^{10})$ while their algorithm runs in time $\text{poly}(d^{1/\alpha^3})$, which is computationally efficient only when the Massart parameter $\alpha$ is a constant. This raises our second question: can we design a polynomial-time algorithm that efficiently leverages both types of noisy queries, while still achieving a query complexity quadratic in the Massart parameters as in the label-only case?

In this paper, we answer the two questions in the affirmative. We show that, what is extremely
Table 1: A summary of our improvement over closely related works on learning halfspaces with Massart noise. For query complexity, since all the methods have polynomial dependence on error rate and VC dimension, we omit them and only display the dependence on noise rate, (note that Diakonikolas et al. [2019] didn’t establish its label dependence on noise rate). In this table, all algorithms work under Massart noise with $\alpha, \beta \in (0, \frac{1}{2}]$.

| Work                  | err($\hat{h}$) | Distribution          | #Labels       | #Comp       |
|-----------------------|----------------|-----------------------|---------------|-------------|
| Yan and Zhang [2017]  | $\epsilon$    | uniform               | $O(1/\alpha^2)$ | -           |
| Zhang et al. [2020]   | $\epsilon$    | isotropic log-concave | $O(1/\alpha^4)$ | -           |
| Diakonikolas et al. [2020] | $\epsilon$ | isotropic log-concave | $O(1/\alpha^{10})$ | -           |
| Hopkins et al. [2020] | $\epsilon$    | s-concave             | $O(1/\alpha^{10})$ | $O(1/\beta^{10})$ |
| Awasthi et al. [2017b]| $\epsilon$    | Any                   | $O(1/\alpha^2)$ | -           |
| Diakonikolas et al. [2019]| $\eta + \epsilon$ | Any                   | $O(1/\gamma^3)$ | -           |
| Ours                  | $\epsilon$    | Any                   | $O(1/\alpha^2)$ | $O(1/\beta^2)$ |

hard in standard PAC setting, is surprisingly easy if we appeal to the wisdom of the crowd. The main idea is to query the label multiple times for each instance (likewise for comparison tag), which significantly mitigates the effect of Massart noise. However, this may cause a serious issue that the query complexity could be orders of magnitude larger than what we obtain in the standard setting (i.e. query each instance for one time). As a matter of fact, we will present a natural approach in Section 3 which naively applies the idea to learn halfspaces with both labels and comparisons, and we show that such natural approach always needs $\log m$ times more queries than that of standard approach. Hence, to break the barrier we design a computationally efficient algorithm that interleaves learning and querying, and we provably show that this multiplicative logarithmic factor in query complexity can be removed. We remark that the idea of using the crowd wisdom to alleviate the Massart noise was explored in Awasthi et al. [2017b] for label-only applications, which does not extend to problems with both label and comparison queries. Indeed, in order to efficiently leverage the comparison oracle, we have to develop a new filtering scheme to detect the most informative pairs to compare, which is the core technical contribution in this work.

1.1 Summary of our contributions

We study the problem of efficient learning halfspaces with both labels and comparisons corrupted by Massart noise. We present the first polynomial-time algorithm that provably learns the underlying halfspace with near-optimal query complexity and noise tolerance in distribution-independent setting. We show that, by carefully designing the filtering process, together with a powerful boosting framework we keep a query complexity that is only a constant times more than that of learning from experts (noiseless case). Our algorithm consists of three crucial components, as stated below.

1) Mitigating labeling by comparison. Given that the learner only has a limited availability of labeled samples, we exploit the power of pairwise comparisons to mitigate the labeling cost. In particular, we will first sort all the instances with a manageable number of comparisons. Under mild conditions, we show that all the instances are sorted correctly with high probability, based on which it suffices to find a “threshold instance” where the labels shift. In particular, if the data set size is $m$, then the label complexity and comparison complexity are $\text{polylog}(m)$ and $O(m \log^2 m)$ respectively. See a more formal statement in Section 3.1.

2) Interleaving learning and annotation. In order to fully make use of the crowd feedback,
we present an adaptive query scheme that interleaves learning and feedback gathering. Namely, we will first learn a weak hypothesis $h_1$, based on which the algorithm concentrates on identifying the instances where $h_1$ may make mistakes on, and query the crowd to obtain the labels for these instances. Therefore, we can learn another hypothesis that complements $h_1$. The identification process, or the filtering process, is the most crucial component in our algorithm. Informally speaking, we first construct an interval using a small amount of labels and comparison tags where instances outside it are informative. For each of those instances, we use pairwise comparison to determine if $h_1$ will misclassify. This ensures that our query complexity is manageable. See a more formal statement in Section 4.

3) **Boosting weak learners.** As we carefully limited the label and comparison complexity, the obtained hypotheses might be suboptimal. Hence, we appeal to a standard boosting framework which integrates the weak learners into a strong hypothesis with desired error rate.

Our algorithm naturally implies a tradeoff between labels and comparisons (see Table 2): without any additional information or queries, the best-known label complexity is $\Theta \left( \frac{m}{\alpha^2} \right)$ [Awasthi et al., 2017b]; while both types of queries are allowed, we obtain an exponentially improved label complexity of $\tilde{O} \left( \frac{\log^2 m}{\alpha^2} \right)$ with a tradeoff of $O \left( \frac{m}{\beta^2} \right)$ comparisons.

**Roadmap.** In Section 2, we introduce more related works that address crowdsourcing and learning with pairwise comparison. In Section 3 we present the detailed problem setup and a natural approach. In Section 4 we present our main algorithm and theoretical analysis. We conclude this paper in Section 5, and the proof details are deferred to the appendix.

## 2 Related Works

**PAC learning of halfspaces.** Learning halfspaces is a long-standing problem in machine learning, dating back to the Perceptron algorithm [Rosenblatt, 1958]. The problem is extensively studied in the past few decades under the standard PAC learning model [Valiant, 1984, Sloan, 1992, Kearns and Valiant, 1988, Kearns et al., 1992, Blum, 1990, Blum et al., 1996, Kalai et al., 2005, Klivans et al., 2009], and a large body of recent studies showed significant progress of near-optimal performance guarantee in terms of noise tolerance, sample complexity, and label complexity [Awasthi et al., 2017a, Zhang, 2018, Diakonikolas et al., 2019, Zhang et al., 2020].

**Learning with comparisons.** Pairwise comparison has been widely applied in practical problems and a large volume of works have established performance guarantee under different setting. For example, Fürnkranz and Hüllermeier [2010], Park et al. [2015] considered the problem of estimating users’ preference to items by presenting pairs of items and collecting the preferences. In particular, under the matrix completion framework [Candès and Recht, 2009], it was shown in Park et al. [2015] that

---

1 We use $\tilde{O}(f) := O(f \cdot \log f)$. 

---

Table 2: A comparison between label-only and compare-and-label algorithms, where $m$ is the sample complexity necessary for distribution-independent PAC-learning the target halfspace.

| Work             | err($\hat{h}$) | Distribution | #Labels | #Comp         |
|------------------|----------------|--------------|---------|---------------|
| Awasthi et al. [2017b] | $\epsilon$     | Any          | $O \left( \frac{m}{\alpha^2} \right)$ | -             |
| Ours             | $\epsilon$     | Any          | $O \left( \frac{\log^2 m}{\alpha^2} \right)$ | $O \left( \frac{m}{\beta^2} \right)$ |
that with mild assumptions recovery of the true preference is possible. Wah et al. 2014 studied the problem of image retrieval and interactive classification based on similarity comparisons. There is also a large body of works investigating the ranking problem based on pairwise comparison, which is either parametric Jamieson and Nowak 2011, Heckel et al. 2016, Shah et al. 2015 or non-parametric Shah et al. 2016, Shah and Wainwright 2017, Pananjady et al. 2017, Shah et al. 2019. Apart from that, Heckel et al. 2018 proposed an active ranking algorithm for estimating an approximate ranking, whereas another line of research focuses on finding exact rankings Hunter et al. 2004, Negahban et al. 2012, Hajek et al. 2014. Interestingly, Balcan et al. 2016 studied the problem of learning close approximations to underlying combinatorial functions through pairwise comparisons, and they showed that a broad class of functions can be provably learned.

Learning from the crowd. The noise in crowd mainly comes from the imperfectness of its workers. A body of research assumes each instance is presented to multiple workers, enabling algorithmic designs for aggregating annotations to improve the quality of labels Aydin et al. 2014, Fan et al. 2015, Ho et al. 2013, Karger et al. 2011, Khetan and Oh 2016, Liu et al. 2012, Raykar et al. 2010, Shah and Lee 2018, Tian and Zhu 2015, Welinder et al. 2010. The Dawid-Skene model is a prominent choice for algorithmic design where the worker’s quality parameter and instance labels are updated alternatively to achieve a higher-quality classifier, and has been widely examined Ho et al. 2013, Fan et al. 2015, Khetan and Oh 2016, Zhang et al. 2016, Zhou et al. 2012. Another line of research has been focused on pruning low-quality workers. For instance, Dekel and Shamir 2009 pre-trained a model on the entire data set and reweighted the labels from each worker based on his performance with respect to the pre-trained hypothesis. Awasthi et al. 2017b constructed a graph that identifies all high-quality workers by discovering and making use of a community structure presented in the crowd, with the observation that high-quality workers would often form a large connected component in the graph. Other works proposed to design better incentive mechanisms, for instance, a mechanism based on peer prediction Jurca and Faltings 2009, Miller et al. 2009, Radanovic et al. 2016, answers with confidence Shah and Zhou 2016, penalty for blind agreement Dasgupta and Ghosh 2013.

3 Preliminaries

We study the problem of learning halfspaces with Massart noise, with access to both labels and comparison tags from the crowd. Let \( \mathcal{X} \subseteq \mathbb{R}^d \) be the instance space and \( \mathcal{Y} = \{-1, 1\} \) be the set of possible labels for each \( x \in \mathcal{X} \). Let \( \mathcal{D} \) be the joint distribution over \( \mathcal{X} \times \mathcal{Y} \), and denote by \( \mathcal{D}_X \) the marginal distribution on \( \mathcal{X} \). Let \( \mathcal{H} = \{ h : \mathcal{X} \rightarrow \mathcal{Y} \} \) be the hypothesis classes. The error rate of a hypothesis \( h \in \mathcal{H} \) over \( \mathcal{D} \) is defined as \( \text{err}_D(h) = \mathbb{P}_{(x,y) \sim \mathcal{D}}(h(x) \neq y) \). We consider the realizable setting, where there exists a hypothesis \( h^* \in \mathcal{H} \) such that \( \text{err}_D(h^*) = 0 \), i.e. \( y = h^*(x) \). In order to incorporate the pairwise comparisons, we consider a natural extension that has been widely used Xu et al. 2017, Hopkins et al. 2020: for each pair of instances \( (x, x') \in \mathcal{X} \times \mathcal{X} \), there is an underlying zero-error comparison function \( Z^*(x, x') \), which outputs 1 if \( x \geq x' \) and \( -1 \) otherwise. Here, we slightly abuse the notation \( x \geq x' \) to indicate that \( x \) is more likely a positive instance than \( x' \) is. In the context of linear classifiers, observe that any hypothesis \( h \in \mathcal{H} \) can be expressed as \( h_w(x) = \text{sign}(w \cdot x) \), and thus \( Z^*(x, x') = \text{sign}(w^* \cdot (x - x')) \) where \( w^* \) is the parameter of \( h^* \).

Standard PAC learning. We will always be interested in the \((\epsilon, \delta)\)-PAC learnable classes \( \mathcal{H} \) in the paper. That is, for any \( \epsilon, \delta \in (0, 1) \), there exists a learning algorithm which takes as input \( m_{\epsilon, \delta} \) correctly labeled instances drawn from \( \mathcal{D} \) and returns a hypothesis \( h \) such that, with probability at least \( 1 - \delta \), \( \text{err}_D(h) \leq \epsilon \). In view of the classic VC theory Anthony and Bartlett 2002, it suffices
to pick
\[ m_{\epsilon, \delta} = O\left(\frac{1}{\epsilon} \left( d \log \frac{1}{\epsilon} + \log \frac{1}{\delta} \right) \right), \]  
where \( d \) is the VC dimension of the hypothesis class \( \mathcal{H} \). We will reserve \( m_{\epsilon, \delta} \) for the above expression.

In our analysis, we concentrate on the dependence on \( \epsilon \). Hence, we will often use a shorthand \( m_{\epsilon} \) in place of \( m_{\epsilon, \delta} \), and thus \( m_{\epsilon} \) can roughly be thought of as \( \tilde{O}(1/\epsilon) \).

**Learning from the crowd.** Although we assumed the existence of a perfect hypothesis, the learner only has access to two noisy oracles, a label revealing oracle \( \mathcal{O}_L \) and a pairwise comparison oracle \( \mathcal{O}_C \). Let \( \mathcal{P} \) be the distribution of crowd workers. The oracle \( \mathcal{O}_L \) takes as input an instance \( x \in \mathcal{X} \) and returns a label \( y \) by querying a crowd worker \( i \) randomly drawn from a distribution \( \mathcal{P} \), while the other oracle \( \mathcal{O}_C \) takes as input a pair of instances \( (x, x') \in \mathcal{X} \times \mathcal{X} \), and returns a tag \( z \in \{-1, 1\} \) by querying a worker \( j \sim \mathcal{P} \). For a given instance \( x \) (respectively a pair \( (x, x') \)), by querying \( \mathcal{O}_L \) (respectively \( \mathcal{O}_C \)) multiple times, it is possible to collect a set of different responses due to the Massart noise. To be more concrete, we consider the following scenario: there exist a large pool of crowd workers, at least an \( a \) fraction of which are reliable and return correct answer w.r.t. \( h^* \) with probability at least \( p \); while the other \( 1 - a \) perform arbitrarily, i.e. they might provide adversarial feedback. In other words, for a given instance, each time \( \mathcal{O}_L \) will return its correct label with probability bounded from below by \( p \cdot a = \frac{1}{2} + \alpha \). Here, we assume \( \alpha \in (0, \frac{1}{2}) \) that ensures the correctness of majority. Likewise, for a given pair of instances, each time \( \mathcal{O}_C \) will return their correct comparison tag with probability at least \( \frac{1}{2} + \beta \), where \( \beta \in (0, \frac{1}{2}) \).

We define label complexity as the total number of calls made to \( \mathcal{O}_L \), and the comparison complexity as that of calls made to \( \mathcal{O}_C \). The total query complexity will be the sum of them. For a learning algorithm that \((\epsilon, \delta)-\text{PAC}\) learns a classifier, we measure its performance by its overhead, or average cost per labeled instance as introduced in [Awasthi et al. 2017b], which is defined as the ratio of total query complexity to \( m_{\epsilon, \delta} \), the number of instances needed to \((\epsilon, \delta)-\text{PAC}\) learn a hypothesis under the standard PAC setting. Likewise, we define the labeling overhead, denoted by \( \Lambda_L \), for labeling oracle and the comparison overhead, denoted by \( \Lambda_C \), for the comparison oracle.

Given a set of annotations \( A = \{a_1, \ldots, a_n\} \) (either the labels or comparison tags), we define \( \text{Maj}(A) \) as outcome by majority vote. Specifically, suppose \( h_1, h_2 \) and \( h_3 \) are three classifiers in \( \mathcal{H} \). The function \( \text{Maj}(h_1, h_2, h_3) \) maps any instance \( x \) to a label \( y \), which is the outcome of majority vote of \( h_1(x), h_2(x) \) and \( h_3(x) \).

Our theoretical analysis will depend on two mild assumptions, as stated below.

**Assumption 1.** There exists a polynomial-time algorithm \( \mathcal{O}_H \) that \((\epsilon, \delta)-\text{PAC}\) learns \( \mathcal{H} \) using \( m_{\epsilon, \delta} \) correctly labeled samples. In addition, there exists a hypothesis \( h^* : \mathcal{X} \rightarrow \mathcal{Y} \) with zero classification error over \( \mathcal{D} \), and a comparison function \( Z^* : \mathcal{X} \times \mathcal{X} \rightarrow \{-1, 1\} \) with zero comparison error over \( \mathcal{D} \times \mathcal{D} \).

**Assumption 2.** For any given instance, each time \( \mathcal{O}_L \) returns its correct label with probability at least \( \frac{1}{2} + \alpha \); and for any given pair of instances, each time \( \mathcal{O}_C \) returns their correct comparison tag with probability at least \( \frac{1}{2} + \beta \). Here \( \alpha \in (0, \frac{1}{2}) \) and \( \beta \in (0, \frac{1}{2}) \) are given to the learner.

Observe that Assumption 1 can be easily met since we presume that all the instances are correctly labeled, and there is a rich literature on efficient PAC learning of halfspaces [Rosenblatt 1958, Valiant 1984, Maass and Turán 1994]. Alternatively, one can think of Assumption 2 as that each worker \( i \sim \mathcal{P} \) gives a correct label with probability \( \frac{1}{2} + \alpha_i \), where \( \alpha_i \geq \alpha \). In that case, Assumption 2 can be reduced to standard Massart setting when the crowd only includes a single worker. However, we remark that there always exists a large pool of workers in the crowd, which
Algorithm 1 Compare-and-Label

Input: A set of unlabeled instances $S = \{x_i\}_{i=1}^{m}$, a parameter $\delta$.
Output: A sorted and labeled set $\hat{S}$.

1. Set $k_1 = O(\beta^{-2} \log \left( \frac{m}{\delta} \right))$ and $k_2 = O(\alpha^{-2} \log \left( \frac{\log m}{\delta} \right))$.
2. Apply RANDOMIZED QUICKSORT to $S$: for each pair $(x, x')$ being compared, query $k_1$ workers and take the majority. Obtain a sorted list $\hat{S} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m)$.

3. $t_{\text{min}} \leftarrow 1$, $t_{\text{max}} \leftarrow m$.
4. while $t_{\text{min}} < t_{\text{max}}$ do
5.   $t = (t_{\text{min}} + t_{\text{max}}) / 2$.
6.   Ask $k_2$ workers to give their labels of $\hat{x}_t$, and denote the majority by $\hat{y}_t$.
7.   if $\hat{y}_t \geq 0$ then
8.     $t_{\text{max}} \leftarrow t - 1$.
9.   else
10.      $t_{\text{min}} \leftarrow t + 1$.
11. end
12. end
13. For all $t' > t$, $\hat{y}_{t'} \leftarrow +1$.
14. For all $t' < t$, $\hat{y}_{t'} \leftarrow -1$.
15. return The sorted and labeled set of instances $\{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \ldots, (\hat{x}_m, \hat{y}_m)\}$.

We can provide different responses upon request. The conditions $\alpha, \beta \in (0, \frac{1}{2}]$ in Assumption 2 ensure the correctness of the majority, while some prior works showed that by collecting extra information from the crowd [Prelec et al., 2017] or by hiring a number of experts [Awasthi et al., 2017b], it is possible to handle the case $\alpha, \beta \in (-\frac{1}{2}, 0]$. We leave such extensions as our future work.

We will always implicitly assume the two conditions throughout our analysis.

3.1 A natural approach and the limitation

In order to utilize the comparison oracle, we consider a natural approach of “compare and label”, which was recently considered in [Xu et al., 2017, Hopkins et al., 2020]. The primary idea is that we can make calls to the comparison oracle $O_C$ first, and use any sorting algorithm, say Quicksort, to rank all the instances. If the comparison oracle is perfect, then it remains to find a threshold left to which the instances will be classified as negative, and otherwise be positive. Here, we choose the RANDOMIZED QUICKSORT since it often admits superior comparison complexity. A more detailed discussion can be found in appendix [F].

We present our approach for crowdsourcing in Algorithm 1. To alleviate the noise, for each instance $x_t$, we have to query $O_C$ multiple times in order to ensure that the majority vote $\hat{z}_t$ is correct with high probability (recall in Assumption 2 we only assumed a weak probability of correct comparison). Then we apply binary search to find the threshold $t$, based on which the labels of all the instances can be correctly identified with high probability.

It is well known that the randomized process in RANDOMIZED QUICKSORT guarantees an average of $O(m \log m)$ comparisons, which provides us the query complexity as follows.

**Proposition 1.** (average version) Algorithm 1 correctly labels all the instances in $S$ with probability at least $1 - \delta$. On average, it makes a total of $O(\frac{1}{\alpha} \cdot \log m \cdot \log \left( \frac{\log m}{\delta} \right))$ calls to $O_L$, and a total of $O(\frac{1}{\beta^2} \cdot m \log m \cdot \log \left( \frac{m}{\delta} \right))$ calls to $O_C$, where $m = |S|$. 


Observe that Randomized QuickSort further guarantees $O(m \log m)$ queries with probability at least $1 - \frac{1}{m}$, we present a high-probability version of Proposition 1 in the following.

**Proposition 2. (high probability guarantee)** With probability at least $1 - O(\frac{1}{m})$, the following holds. Algorithm 1 correctly labels all the instances in $S$ by making a total of $O\left(\frac{\log^2 m}{\epsilon^2}\right)$ calls to $O_L$, and a total of $O\left(\frac{m \log^2 m}{\beta^2}\right)$ calls to $O_C$, where $m = |S|$.

We prove the propositions by first showing that with high probability, the instances of $S$ are sorted correctly. Then we show that based on such sorted list of instances, we can find the correct threshold with high probability. See Appendix A for the proof.

This approach outperforms the standard algorithms with superior query complexity and noise tolerance. More importantly, our result is distribution-independent while the existing ones often impose strong assumptions on the underlying distribution. Recall that for PAC learning we will need $|S| = m_e$. Hence, in view of Proposition 2, the labeling and comparison overhead of this algorithm are respectively given by

$$\Lambda_L = O_\alpha\left(\frac{\log^2 m_e}{m_e}\right) = o_\alpha(1), \quad \Lambda_C = O_\beta\left(\log^2 m_e\right).$$

Note that the dependence on Massart parameters, i.e. $\frac{1}{\alpha^2}$ or $\frac{1}{\beta^2}$, is unavoidable even when the underlying marginal distribution is uniform over a unit sphere [Yan and Zhang 2017], so we focus on the dependence on $m_e$. Here, $\Lambda_L$ is loosely upper bounded by a constant provided that $m_e$ is large enough, which is favorable especially when labels are precious. However, we note that $\Lambda_C$ grows with the size of the training set. In other words, if we expand the training size, the average cost on each labeled instance will also increase. In the next section, we show that with a more involved algorithmic design, the comparison overhead can be reduced to a constant level as well.

## 4 Main Algorithm and Performance Guarantee

In this section, we present a more involved algorithm which interleaves learning and annotation collection. The main idea is based on the boosting framework which was used for PAC learning from label-only queries under the crowdsourcing setting [Awasthi et al. 2017b]. Suppose that our target error rate is $\epsilon$. In place of gathering all correct labels for learning (which requires $m_e$ labels), we can first learn a weak classifier $h_1$ with slightly worse error rate of $\sqrt{\epsilon}$. This step only consumes a small amount of labels and pairwise comparisons. Based on $h_1$, it is possible to identify the “hard instances”, where $h_1$ will very likely make mistakes. We will then learn another classifier $h_2$ that fits on a subset of them: we use subset in that there is a budget on labeling and comparison complexity. Finally, it is easy to find instances where $h_1$ and $h_2$ disagree on, which can be used to learn the last classifier $h_3$. Formally, we will need the following result.

**Theorem 3 (Schapire 1990).** For any $p < \frac{1}{2}$ and distribution $D$, consider three classifiers: $h_1(x), h_2(x), h_3(x)$, that are $p$-close to the target classifier over three different distribution: 1) the original distribution $D_1 = D$; 2) distribution $D_2 = \frac{1}{2}D_C + \frac{1}{2}D_I$, where $D_C$ denotes distribution $D$ conditioned on $\{x \mid h_1(x) = f^\ast(x)\}$, and $D_I$ denotes distribution $D$ conditioned on $\{x \mid h_1(x) \neq f^\ast(x)\}$; 3) distribution $D_3$ that simulated as $D$ conditioned on $\{x \mid h_1(x) \neq h_2(x)\}$. Then, taking majority vote of $h_1, h_2, h_3$ gives error bounded by $3p^2 - 2p^3$.

In particular, we will choose $p = \sqrt{\epsilon}$. By VC theory [Anthony and Bartlett 2002] it suffices to collect $\tilde{O}(1/\sqrt{\epsilon})$ correctly labeled instances to learn all three classifiers provided that we can
The following lemma summarizes the performance of the learned hypothesis $h$.

**Lemma 5.** In Phase 1, with probability $1 - O\left(\frac{1}{m^{\sqrt{\alpha}}}\right)$, $\text{err}_D(h_1) \leq \sqrt{\epsilon}$. In addition, the label complexity is $O\left(\frac{\log^2 m^{\sqrt{\alpha}}}{\alpha^2}\right)$ and the comparison complexity is $O\left(\frac{m^{\sqrt{\alpha}} \log^2 m^{\sqrt{\alpha}}}{\beta^2}\right)$.

---

Algorithm 2 PAC LEARNING FROM THE CROWD WITH PAIRWISE COMPARISONS

**Input:** An oracle $EX$ that draws unlabeled instances from $D_X$, hypotheses class $H$, parameters $\epsilon$ and $\delta$.

**Output:** Hypothesis $h$.

**Phase 1:**
16 Let $\overline{S}_1 \leftarrow \text{COMPARE-AND-LABEL}(S_1, \frac{\delta}{6})$, for a set of sample $S_1$ of size $m^{\sqrt{\epsilon}} \frac{\delta}{6}$ from $D_X$.
17 $h_1 \leftarrow \mathcal{O}_H(\overline{S}_1)$.

**Phase 2:**
20 Let $S_I \leftarrow \text{FILTER}(S_2, h_1)$, for a set of samples $S_2$ of size $\Theta(m, \delta)$ drawn from $D_X$.
21 Let $S_C$ be a sample set of size $\Theta(m^{\sqrt{\epsilon}}, \delta)$ drawn from $D_X$.
22 Let $\overline{S}_\text{All} \leftarrow \text{COMPARE-AND-LABEL}(S_I \cup S_C, \frac{\delta}{6})$.
23 Let $W_I \leftarrow \{(x, y) \in \overline{S}_\text{All} | y \neq h_1(x)\}$, and let $W_C \leftarrow \overline{S}_\text{All} \setminus W_I$.
24 Draw a sample set $W$ of size $\Theta(m^{\sqrt{\epsilon}}, \delta)$ from a distribution that equally weights $W_I$ and $W_C$.
25 Let $h_2 \leftarrow \mathcal{O}_H(W)$.

**Phase 3:**
27 Let $\overline{S}_3 \leftarrow \text{COMPARE-AND-LABEL}(S_3, \frac{\delta}{6})$, for a sample set $S_3$ of size $m^{\sqrt{\epsilon}} \frac{\delta}{6}$ drawn from $D_X$ conditioned on $h_1(x) \neq h_2(x)$.
28 Let $h_3 \leftarrow \mathcal{O}_H(\overline{S}_3)$.
29 \textbf{return} $\text{Maj}(h_1, h_2, h_3)$.

---

We construct the distributions $D_1$, $D_2$, and $D_3$. The main algorithm is described in Algorithm 2 where in each phase we carefully control the label and comparison complexity to simulate these distributions. We state the main result of this paper below.

**Theorem 4 (Main result).** With probability $1 - O\left(\frac{1}{m^{\sqrt{\epsilon}}}\right)$, Algorithm 2 returns a classifier $h \in H$ with error rate $\text{err}_D(h) \leq \epsilon$ by making $O\left(\frac{1}{\alpha^2} \cdot \log^2 m \cdot \log \log m\right)$ total calls to $\mathcal{O}_L$ and $O\left(\frac{m}{\beta^2}\right)$ total calls to $\mathcal{O}_C$, given that $\log^2(\frac{d}{\sqrt{\epsilon}}) \leq \frac{1}{\sqrt{\epsilon}}$. Furthermore, Algorithm 2 runs in polynomial time.

In the following, we elaborate on each phase, and provide our performance guarantee. The proof of Theorem 4 is sketched at the end of this section.

**4.1 Analysis of Phase 1**

The goal of Phase 1 is to learn a weak classifier with error rate $\sqrt{\epsilon}$ on $D$. This can easily be fulfilled if we sample a set of $m^{\sqrt{\epsilon}}$ unlabeled instances, and invoke Algorithm 1 to obtain the desired hypothesis $h_1$. Note that this is different from the natural approach as we presented in 3.1: the natural approach gathers all necessary instances (i.e. $m_\epsilon$ unlabeled data) and feed them to Algorithm 1 which leads to an overwhelming comparison overhead.

Since the set $\overline{S}_1$ is guaranteed to be labeled correctly with high probability (Proposition 2), we can learn $h_1$ with any standard polynomial-time algorithm $\mathcal{O}_H$ (see Assumption 1 for the definition). The following lemma summarizes the performance of the learned hypothesis $h_1$.

**Lemma 5.** In Phase 1, with probability $1 - O\left(\frac{1}{m^{\sqrt{\epsilon}}}\right)$, $\text{err}_D(h_1) \leq \sqrt{\epsilon}$. In addition, the label complexity is $O\left(\frac{\log^2 m^{\sqrt{\epsilon}}}{\alpha^2}\right)$ and the comparison complexity is $O\left(\frac{m^{\sqrt{\epsilon}} \log^2 m^{\sqrt{\epsilon}}}{\beta^2}\right)$. 

---

9
Algorithm 3 Filter

Input: Set of instances $S$, classifier $h$, parameter $b = 100 \log |S|$  
Output: The set $S_I$ whose instances are misclassified by $h$.

Initialization: $S_I \leftarrow \emptyset$, $N \leftarrow O(\log(1/\epsilon))$.

while $S \neq \emptyset$ do
  if $|S| \leq b$ then
    Call COMPARE-AND-LABEL($S, \delta'$) to obtain the labels of all instances in $S$, and put those $h$
    makes mistakes on into $S_I$.
  else
    Sample uniformly a subset $S_p \subset S$ of $b$ points.
    $S_p' \leftarrow$ COMPARE-AND-LABEL($S_p, \delta'$). Denote by $x^-$ the rightmost instance of those being
    labeled as $-1$, and denote by $x^+$ the leftmost instance of those being labeled as $+1$.
    for $x \in S \setminus S_p$ do
      $\rho \leftarrow 1$.
      for $t = 1, \ldots, N$ do
        Draw a random worker $t$ from $\mathcal{P}$ to compare $x$ with $x^-$ and $x^+$, obtain $Z_t(x, x^-)$ and
        $Z_t(x, x^+)$.
        If $t$ is even, then continue to the next iteration.
        If $\text{Maj}(Z_{1:t}(x, x^-)) = 1$ and $\text{Maj}(Z_{1:t}(x, x^+)) = -1$, then $\rho \leftarrow 0$ and break.
        Filtering: If $[\text{Maj}(Z_{1:t}(x, x^-)) = -1$ and $h(x) = -1]$ or $[\text{Maj}(Z_{1:t}(x, x^+)) = +1$ and
        $h(x) = +1]$, then $S' \leftarrow S' \cup \{x\}$, $\rho \leftarrow 0$, and break.
      end
      If $\rho = 1$, then $S_I \leftarrow S_I \cup \{x\}$.
    end
    $S \leftarrow S \setminus (S_I \cup S' \cup S_p)$.
  end
return $S_I$

4.2 Analysis of Phase 2

Phase 2 aims to simulate the distribution $\mathcal{D}_2$ using the noisy feedback from the crowd, while
maintaining a reasonable overhead. In particular, we need to find $O(m_\sqrt{\epsilon})$ instances that $h_1$
will misclassify, and $O(m_\sqrt{\epsilon})$ instances that $h_1$ will label correctly. Since the error rate of $h_1$ is $O(\sqrt{\epsilon})$
(which is very small), identifying a batch of instances $S_I$ that $h_1$ will make mistakes on is the
primary challenge.

We tackle this challenge by designing a novel algorithm termed Filter. Note that $err_\mathcal{P}(h_1) \leq
\sqrt{\epsilon}$ indicates that if we hope to obtain $S_I$ with size $m_\sqrt{\epsilon}$, we have to collect $m_\sqrt{\epsilon}/\sqrt{\epsilon} = m_\epsilon$ instances
in view of Chernoff bound. That gives the size of the set $S_2$.

In Algorithm 3 we present the Filter algorithm. Note that since the size of $S$ (i.e. $S_2$) is
$O(m_\epsilon)$, we cannot invoke Algorithm 1 directly: this leads to an overwhelming overhead as the
natural approach. In contrast, we randomly sample a subset $S_p$ of $S$ with $O(\log |S|)$ members, for
which we can apply Algorithm 1 to obtain a sorted list of labeled instances $S_p'$. Based on $S_p'$, we
identify two crucial instances $x^-$ and $x^+$ termed support instances. Roughly speaking, $x^-$ is the
negative instance that is “closest” to positive, and $x^+$ is the positive instance that is “closest” to
negative (similar to the support vectors in SVM). See Figure 1 for an illustration. Now for each
$x \in S \setminus S_p$, we can make use of the comparison to identify whether it lies between $x^-$ and $x^+$,
4.3 Analysis of Phase 3

Given the hypotheses $h_1$ and $h_2$, we sample a set of $m_{\sqrt{\epsilon}}$ unlabeled instances that fall into the region $R := \{x : h_1(x) \neq h_2(x)\}$. Such instances can easily be gathered by rejection sampling: we randomly draw an instance $x \sim D$ and put it in $S_3$ if $x \in R$; otherwise we reject this instance and continue drawing another instance from $D$. Note that in this phase, we do not need to call $O_L$ nor $O_C$. Then we apply Algorithm 1 to the unlabeled set $S_3$. Since the size of $S_3$ is $m_{\sqrt{\epsilon}}$, we are guaranteed an error rate of $\sqrt{\epsilon}$ on $\mathcal{D}_3$ in view of Proposition 2.
Lemma 8. In Phase 3, with probability \(1 - O\left(\frac{1}{m\sqrt{\epsilon}}\right)\), \(\text{err}_{D_3}(h_3) \leq \sqrt{\epsilon}\). In addition, the label complexity is \(O\left(\frac{\log^2 m}{\alpha^2} \sqrt{\epsilon}\right)\) and the comparison complexity is \(O\left(\frac{m^2\log^2 m}{\beta^2} \sqrt{\epsilon}\right)\).

4.4 Proof Sketch of Theorem 4

Combining Lemma 5, Lemma 7 and Lemma 8, and using the union bound in probability, with probability at least \(1 - O\left(\frac{1}{m\sqrt{\epsilon}}\right)\), the following holds: \(\text{err}_{D_1}(h_1) \leq \sqrt{\epsilon}, \text{err}_{D_2}(h_2) \leq \sqrt{\epsilon}, \text{err}_{D_3}(h_3) \leq \sqrt{\epsilon}\), which contributes to a classifier \(h\) with \(\text{err}_D(h) \leq \epsilon\) by taking majority vote of \(h_1, h_2, h_3\) (Theorem 3). In addition, the total label complexity is \(O\left(\frac{\log^2 m \cdot \log \log m}{\alpha^2}\right)\), and the total comparison complexity is \(O\left(\frac{m}{\beta^2}\right)\), as long as \(\log^2\left(\frac{d}{\sqrt{\epsilon}}\right) \leq \frac{1}{\sqrt{\epsilon}}\). Hence, the labeling and comparison overhead of Algorithm 2 are respectively given by

\[
\Lambda_L = O_\alpha \left(\frac{\log^2 m \cdot \log \log m}{m^2}\right) = o_\alpha(1), \quad \Lambda_C = O_\beta \left(\frac{m^2}{m^2}\right) = O_\beta (1).
\]

This completes the proof of Theorem 4.

5 Conclusion and Future Works

In this paper, we have presented a computationally and query efficient algorithm that learns an unknown hypothesis from the labels and pairwise comparison tags corrupted with Massart noise in distribution-independent setting. We provably showed that our algorithm enjoys near-optimal query complexity with only \(o_\alpha(1)\) labeling overhead and \(O_\beta(1)\) comparison overhead. To this end, we have developed a set of new algorithmic results with theoretical guarantee. We believe that it would be interesting to explore other types of crowd feedback, such as multiclass labels. It is also interesting to study agnostic PAC learning \(\text{Kalai et al., 2005}\) under the crowdsourcing setting.

References

Martin Anthony and Peter L. Bartlett. Neural Network Learning: Theoretical Foundations. Cambridge University Press, 2002.

Pranjal Awasthi, Maria-Florina Balcan, Nika Haghtalab, and Ruth Urner. Efficient learning of linear separators under bounded noise. In Proceedings of the 28th Conference on Learning Theory, pages 167–190, 2015.

Pranjal Awasthi, Maria-Florina Balcan, Nika Haghtalab, and Hongyang Zhang. Learning and 1-bit compressed sensing under asymmetric noise. In Proceedings of the 29th Conference on Learning Theory, pages 152–192, 2016.

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

Pranjal Awasthi, Avrim Blum, Nika Haghtalab, and Yishay Mansour. Efficient PAC learning from the crowd. In Satyen Kale and Ohad Shamir, editors, Proceedings of the 30th Conference on Learning Theory, pages 127–150, 2017b.
Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. Crowdsourcing for multiple-choice question answering. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, pages 2946–2953. AAAI Press, 2014.

Maria-Florina Balcan, Ellen Vitercik, and Colin White. Learning combinatorial functions from pairwise comparisons. In Proceedings of the 29th Conference on Learning Theory, pages 310–335, 2016.

Avrim Blum. Learning boolean functions in an infinite attribute space. In Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, pages 64–72, 1990.

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

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. Foundations of Computational Mathematics, 9(6):717–772, 2009.

Sitan Chen, Frederic Koehler, Ankur Moitra, and Morris Yau. Classification under misspecification: Halfspaces, generalized linear models, and connections to evolvability. CoRR, abs/2006.04787, 2020.

Nello Cristianini and John Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2000.

Anirban Dasgupta and Arpita Ghosh. Crowdsourced judgement elicitation with endogenous proficiency. In Proceedings of the 22nd International World Wide Web Conference, pages 319–330, 2013.

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

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

Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, and Nikos Zaniris. Learning halfspaces with massart noise under structured distributions. In Proceedings of 33rd the Conference on Learning Theory, volume 125, pages 1486–1513, 2020.

Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-Lee Tan, and Jianhua Feng. icrowd: An adaptive crowdsourcing framework. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pages 1015–1030, 2015.

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

Bruce Hajek, Sewoong Oh, and Jiaming Xu. Minimax-optimal inference from partial rankings. In Advances in Neural Information Processing Systems, pages 1475–1483, 2014.

Reinhard Heckel, Nihar B. Shah, Kannan Ramchandran, and Martin J. Wainwright. Active ranking from pairwise comparisons and the futility of parametric assumptions. CoRR, abs/1606.08842, 2016.
Reinhard Heckel, Max Simchowitz, Kannan Ramchandran, and Martin J. Wainwright. Approximate ranking from pairwise comparisons. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, volume 84, pages 1057–1066, 2018.

Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 534–542, 2013.

Max Hopkins, Daniel M. Kane, and Shachar Lovett. The power of comparisons for actively learning linear classifiers. *CoRR*, 2019. URL http://arxiv.org/abs/1907.03816.

Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Noise-tolerant, reliable active classification with comparison queries. *CoRR*, abs/2001.05497, 2020.

David R Hunter et al. Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406, 2004.

Kevin G. Jamieson and Robert D. Nowak. Active ranking using pairwise comparisons. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, pages 2240–2248, 2011.

Radu Jurca and Boi Faltings. Mechanisms for making crowds truthful. *Journal Artificial Intelligence Research*, 34:209–253, 2009.

Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 11–20, 2005.

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

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

M. Kearns and L. G. Valiant. Learning boolean formulae or finite automata is as hard as factoring. Technical Report TR 14-88, Harvard University Aiken Computation Laboratory, 1988.

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

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

Adam R. Klivans, Philip M. Long, and Rocco A. Servedio. Learning halfspaces with malicious noise. *Journal of Machine Learning Research*, 10:2715–2740, 2009.

Qiang Liu, Jian Peng, and Alexander T. Ihler. Variational inference for crowdsourcing. In *Proceedings of the 26th Annual Conference on Neural Information Processing System*, pages 701–709, 2012.
Wolfgang Maass and György Turán. 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: constraints and prospects, pages 381–414, 1994.

Pascal Massart and Élodie Nédélec. Risk bounds for statistical learning. The Annals of Statistics, 34:2326–2366, 2006.

Nolan Miller, Paul Resnick, and Richard Zeckhauser. Eliciting informative feedback: The peer-prediction method. In Jennifer Golbeck, editor, Computing with Social Trust, Human-Computer Interaction Series, pages 185–212. Springer, 2009.

Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pair-wise comparisons. In Advances in neural information processing systems, pages 2474–2482, 2012.

Ashwin Pananjady, Cheng Mao, Vidya Muthukumar, Martin J. Wainwright, and Thomas A. Courtade. Worst-case vs average-case design for estimation from fixed pairwise comparisons. CoRR, abs/1707.06217, 2017. URL http://arxiv.org/abs/1707.06217

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

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

Goran Radanovic, Boi Faltings, and Radu Jurca. Incentives for effort in crowdsourcing using the peer truth serum. ACM Trans. Intell. Syst. Technol., 7(4):48:1–48:28, 2016.

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

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, pages 65–386, 1958.

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

Devavrat Shah and Christina E. Lee. Reducing crowdsourcing to graphon estimation, statistically. In Proceedings of the 21st International Conference on Artificial Intelligence and Statistics, volume 84, pages 1741–1750, 2018.

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

Nihar B. Shah and Dengyong Zhou. Double or nothing: Multiplicative incentive mechanisms for crowdsourcing. Journal of Machine Learning Research, 17:165:1–165:52, 2016.

Nihar B. Shah, Sivaraman Balakrishnan, Joseph K. Bradley, Abhay Parekh, Kannan Ramchandran, and Martin J. Wainwright. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, volume 38, 2015.
Nihar B. Shah, Sivaraman Balakrishnan, Aditya Guntuboyina, and Martin J. Wainwright. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. In Proceedings of the 33nd International Conference on Machine Learning, pages 11–20, 2016.

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

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

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

Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. In Proceedings of the 29th Annual Conference on Neural Information Processing Systems, pages 1621–1629, 2015.

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

Catherine Wah, Grant Van Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge J. Belongie. Similarity comparisons for interactive fine-grained categorization. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 859–866, 2014.

Peter Welinder, Steve Branson, Serge J. Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In Proceedings of the 24th Annual Conference on Neural Information Processing Systems, pages 2424–2432, 2010.

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

Songbai Yan and Chicheng Zhang. Revisiting perceptron: Efficient and label-optimal learning of halfspaces. In Proceedings of the 31st Annual Conference on Neural Information Processing Systems, pages 1056–1066, 2017.

Chicheng Zhang. Efficient active learning of sparse halfspaces. In Proceedings of the 31st Annual Conference On Learning Theory, pages 1856–1880, 2018.

Chicheng Zhang, Jie Shen, and Pranjal Awasthi. Efficient active learning of sparse halfspaces with arbitrary bounded noise. CoRR, abs/2002.04840, 2020.

Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. Journal of Machine Learning Research, 17:102:1–102:44, 2016.

Yuchen Zhang, Percy Liang, and Moses Charikar. A hitting time analysis of stochastic gradient langevin dynamics. In Proceedings of the 30th Conference on Learning Theory, volume 65, pages 1980–2022, 2017.

Dengyong Zhou, John C. Platt, Sumit Basu, and Yi Mao. Learning from the wisdom of crowds by minimax entropy. In Proceedings of the 26th Annual Conference on Neural Information Processing Systems, pages 2204–2212, 2012.
\section*{A Proof of Propositions in 3.1}

\textbf{Proposition 9} (Restatement of Proposition 1). Algorithm 1 correctly labels all the instances in \( S \) with probability at least \( 1 - \delta \). On average, it makes a total of \( O\left(\frac{1}{\beta^2} \cdot m \cdot \log \left( \frac{m}{\delta} \right) \right) \) calls to \( O_L \), and a total of \( O\left(\frac{1}{\beta^2} \cdot m \log m \cdot \log \left( \frac{m}{\delta} \right) \right) \) calls to \( O_C \), where \( m = |S| \).

\textit{Proof.} By assumption 2, given an instance pair \((x, x') \in \mathcal{X} \times \mathcal{X} \), if we randomly sample a worker \( j \sim \mathcal{P} \), with probability at least \( \frac{1}{2} + \beta \), the worker \( j \) gives the correct comparison result. Let \( k_1 \) be the number of workers we query for each pair in Algorithm 1, then the probability that majority vote makes mistake on pair \((x, x') \) is given by \( \Pr(z^* (x, x') \cdot \sum_{j=1}^{k_1} z_j (x, x') \leq 0) \). Without loss of generality, we assume the ground truth to be \(-1\). Then, \( \mathbb{E}_{j \sim \mathcal{P}_C} [z_j (x, x')] = -2\beta < 0 \).

By Hoeffding’s inequality,
\[
\Pr\left( \sum_{j=1}^{k_1} z_j (x, x') \geq k_1 \cdot \mathbb{E}_j [z_j (x, x')] + t \right) \leq e^{-\frac{2t^2}{k_1 (\beta - a)^2}}, \tag{A.1}
\]
where \( a = -1 \), \( b = +1 \) are the corresponding lower and upper bounds of \( z_j (x, x') \). Let \( k_1 \cdot \mathbb{E}_j [z_j (x, x')] + t = 0 \), the RHS of the inequality becomes \( e^{-2k_1\beta^2} \). Let \( q_m \) be the total number of comparisons made by algorithm \textsc{RandomizedQuickSort}. We apply the union bound to conclude that
\[
\Pr\left( \bigcup_{l=1}^{q_m} \left[ z^* (x, x') \sum_{j=1}^{k_1} z_j (x, x') \leq 0 \right] \right) \leq \sum_{l=1}^{q_m} e^{-2k_1\beta^2} \leq q_m \cdot e^{-2k_1\beta^2}. \tag{A.2}
\]
Let the RHS be the confidence parameter \( \delta \). Then we have
\[
k_1 = O\left(\frac{1}{\beta^2} \cdot \log \left( \frac{q_m}{\delta} \right) \right). \tag{A.3}
\]

Given that the average number of comparisons made by \textsc{RandomizedQuickSort} is \( O(m \log m) \), while the worst case is \( O(m^2) \). In either case,
\[
k_1 = O\left(\frac{1}{\beta^2} \cdot \log \left( \frac{m}{\delta} \right) \right) \tag{A.4}
\]
on each pair suffices to guarantee that with probability at least \( 1 - \delta \), all the instances in the sample set will be sorted correctly. While on average, the total number of calls made to the comparison oracle is
\[
m \log m \cdot k_1 = O\left(\frac{1}{\beta^2} \cdot m \log m \cdot \log \left( \frac{m}{\delta} \right) \right). \tag{A.5}
\]

As for the label complexity. By Assumption 2 for any given instance \( x \in \mathcal{X} \), if we randomly draw a worker \( i \sim \mathcal{P} \), the probability that worker \( i \) gives a correct label is at least \( \frac{1}{2} + \alpha \). Similar to the previous analysis, let \( y_i = h^*(x_i) \) be the true label for instance \( x_i \), by Hoeffding’s inequality, we conclude that
\[
\Pr\left( \bigcup_{l=1}^{\log m} \left[ y_i \sum_{i=1}^{k_2} g_i (x_i) \leq 0 \right] \right) \leq \sum_{l=1}^{\log m} \Pr\left( y_i \sum_{i=1}^{k_2} g_i (x_i) \leq 0 \right) \leq \log m \cdot e^{-2k_2\alpha^2} = \delta, \tag{A.6}
\]
provided that we use
\[
k_2 = O\left(\frac{1}{\alpha^2} \cdot \log \left( \frac{\log m}{\delta} \right) \right).
\]
workers to label each instance. Due to binary search, the total number of calls we made to the labeling oracle is

\[ k_2 \cdot \log m = O\left(\frac{1}{\alpha^2} \cdot \log m \cdot \log \left(\frac{\log(m)}{\delta}\right)\right). \]  

Combining the guarantees for comparison and labeling, the given instance set \( S \) is sorted correctly, and the correct threshold is identified with high probability. On average, this algorithm makes a total of \( O\left(\frac{1}{\alpha^2} \cdot \log m \cdot \log \left(\frac{\log(m)}{\delta}\right)\right) \) calls to \( \mathcal{O}_L \), and a total of \( O\left(\frac{1}{\beta^2} \cdot m \log m \cdot \log \left(\frac{\delta}{\beta^2}\right)\right) \) calls to \( \mathcal{O}_C \).

**Proposition 10 (Restatement of Proposition 2).** With probability at least \( 1 - O\left(\frac{1}{m}\right) \), the following holds. Algorithm 1 correctly labels all the instances in \( S \) by making a total of \( O\left(\frac{\log^2 m}{\alpha^2}\right) \) calls to \( \mathcal{O}_L \), and a total of \( O\left(\frac{m \log^2 m}{\beta^2}\right) \) calls to \( \mathcal{O}_C \), where \( m = |S| \).

**Proof.** Based on what we have in Proposition 9, provided that RANDOMIZED QUICKSORT further guarantees that with probability at least \( 1 - \frac{1}{m} \), it makes \( O(m \log m) \) comparisons (Lemma 21). Let \( \delta = \frac{1}{m} \), then the numbers of calls to \( \mathcal{O}_L \) and \( \mathcal{O}_C \) are \( O\left(\frac{\log^2 m}{\alpha^2}\right) \) and \( O\left(\frac{m \log^2 m}{\beta^2}\right) \) respectively.

\[ \square \]

## B  Analysis of Phase 1

**Lemma 11 (Restatement of Lemma 5).** In Phase 1, with probability \( 1 - O\left(\frac{1}{m^{\sqrt{\epsilon}}}\right) \), \( \text{err}_D(h_1) \leq \sqrt{\epsilon} \). In addition, the label complexity is \( O\left(\frac{\log^2 \log m}{\alpha^2 \epsilon}\right) \) and the comparison complexity is \( O\left(\frac{m \cdot \sqrt{\epsilon} \cdot \log^2 m}{\beta^2}\right) \).

**Proof.** In Phase 1, by Proposition 10 all the instances in \( S_1 \) are correctly labeled with probability \( 1 - \frac{\delta}{6} \). In addition, given the sample size \( |S_1| = m^{\sqrt{\epsilon}, \frac{\epsilon}{6}} \), with probability \( 1 - \frac{\delta}{6} \), the oracle \( \mathcal{O}_H \) returns a classifier with error rate \( \leq \sqrt{\epsilon} \) in view of the VC Theory (Theorem 22). Applying the union bound, the returned classifier \( h_1 \) has error rate \( \text{err}_D(h_1) \leq \sqrt{\epsilon} \) with probability \( 1 - \frac{\delta}{3} \). Let \( \frac{\delta}{3} = \frac{1}{m^{\sqrt{\epsilon}}} \), the total calls to \( \mathcal{O}_L \) is \( O\left(\frac{\log^2 \log m}{\alpha^2 \epsilon}\right) \) and the comparison complexity is \( O\left(\frac{m \cdot \sqrt{\epsilon} \cdot \log^2 m}{\beta^2}\right) \). 

\[ \square \]

## C  Analysis of Phase 2

**C.1  Iteration of While-loop in Filter**

**Lemma 12.** In each iteration of the while-loop of Algorithm 3, after identifying the support instances \( x^- \), \( x^+ \), with probability at least \( 1 - \frac{1}{m^{\beta^2 \alpha^2 \epsilon}} \), the interval \([x^-, x^+]\) contains less than a \( 1/2 \) fraction of unfiltered instances, where \( m \) is the initial size of the input set \( S \).

**Proof.** In each while-loop, the algorithm randomly samples a subset \( S_p \subseteq S \) with \( b \) instances.

Naturally, the support instances \( x^- \) and \( x^+ \) could be any \( x_i \) and \( x_{i+1} \) in \( S_p \) respectively. We consider that extreme cases like all the sub-samples are subject to identical labels are also possible. Thus, we hope the interval formed by any \( x_i \) and \( x_{i+1} \), \( \forall i \in \{0, 1, 2, \ldots, b\} \), contains less than a \( 1/2 \) fraction of unfiltered instances.

Denote by \( E_i \) the event that interval \([x_i, x_{i+1}]\) contains more than a \( 1/2 \) fraction of the unfiltered instances. As illustrated by Figure 2 let \( a \) be the distance from \( x_i \) to the leftmost instance \( x_0 \). Then, event \( E_i \) is given by that \( \max(i-1, 0) \) of the sub-samples are on the left-hand-side of \( x_i \) and
The support instances $x^-$ and $x^+$ are defined as in Figure 1. Here, we use $x_i$ to indicate the $i$-th sample in $S_p$, where $i \in \{0, 1, \ldots, b\}$. In addition, we use $x_0$ to denote the leftmost instance in $S$ and use $x_{b+1}$ to denote the rightmost instance in $S$. We consider $x_0$ and $x_{b+1}$ are not observed yet.

In Algorithm 3, the parameter $(a, b) = (\frac{1}{2}, 100 \log(m))$, which means the probability that the interval $[x^-, x^+]$ contains less than a $1/2$ fraction of the unfiltered instances is no less than $1 - \frac{1}{m^{1/5}}$.

Lemma 13. With probability at least $1 - \frac{1}{m^{1/5}}$, the iteration number of the while-loop in Algorithm 3 is $O(\log m)$, where $m$ is the initial size of the input set $S$.

Proof. The iteration number of the while-loop in Algorithm 3 is determined by the amount of instances being removed from $S$ in each iteration. We will first show that a significant fraction of the unfiltered instances are potentially informative after the sub-sampling. Then, for each informative instances, we show that it will go to $S'$ or $S_f$ with probability $> 1/2$. At last, using the Chernoff bound, we show that in each iteration, at least a $1/8$ fraction of instances will be removed from $S$ with high probability, which contributes to logarithmic iteration number as desired.

Lemma 12 guarantees that, in one while-loop, with probability at least $1 - \frac{1}{m^{1/5}}$, the interval $[x^-, x^+]$ contains no more than a $1/2$ fraction of the instances, which means more than $1/2$ fraction are outside the interval and potentially informative, i.e. $Z^*(x, x^-) = -1$ or $Z^*(x, x^+) = +1$.

Suppose Assumption 2 is satisfied, and further consider that $\frac{1}{2} + \beta > 0.85$ here for ease of presentation, as it’s always easy to boost any $\frac{1}{2} + \beta > \frac{1}{2}$. Then, for any instance $x$, the probability
that worker $t \sim \mathcal{P}$ gives both correct comparison tags with respect to $x^-$ and $x^+$ is

$$\Pr \left[ (z_t(x, x^-) = Z^*(x, x^-)) \cup (z_t(x, x^+) = Z^*(x, x^+)) \right] = \left( \frac{1}{2} + \beta \right)^2 > 0.7. \quad (C.1)$$

For an $x$ lying outside the interval $[x^-, x^+]$, we are interested in the probability that even though $N$ workers are queried, the following never occurs: $\text{Maj}(z_{1:t}(x, x^-)) = 1$ and $\text{Maj}(z_{1:t}(x, x^+)) = -1$, $\forall t = \{1, 2, \ldots, N\}$, as it will keep $x$ in $S$ as an unfiltered instance. Denote by $E_t$ the event that these two equations hold when worker $t$ provides its tags. If any worker $t$ makes mistake, it would potentially trigger an event $E'_t$ in the future, where $t' \geq t$. Therefore, we consider the following: if any one of the vote makes mistake, we require the next 2 workers to give both correct comparison tags such that the majorities keep correct. The problem boils down to a biased random walk. Recall that in each round, worker $t$ gives both correct tags with probability $> 0.7$. Consider a walk starting from the origin. With probability $> 0.7$, it move to the right, and with the remaining probability, it move to the left. The probability that this random walk ever crosses the origin is given by the ruin of gambling probability (Theorem 23). Therefore,

$$\Pr \left( \bigcup_{t=1}^N E_t \right) \leq \left( 1 - \left( \frac{0.7}{1 - 0.7} \right)^N \right) / \left( 1 - \left( \frac{0.7}{1 - 0.7} \right)^{N+1} \right) < \frac{3}{7}, \quad (C.2)$$

In other words, with probability larger than $\frac{4}{7} > \frac{1}{2}$, an $x$ lying outside the interval $[x^-, x^+]$ will added either in set $S'$ or set $S_I$. In either case, it will be removed from $S$.

Applying the Hoeffding’s inequality, in an iteration, with probability $1 - \exp(- |S|/32)$, we have at least $1/8$ of the instances in $S \setminus S_p$ being removed from $S$. Although the size of $S$ decreases, it is guaranteed that $|S| \geq 100 \log m$. As a result, by the union bound, with probability at least

$$1 - \left( \frac{1}{m^{50}} + e^{-\Omega(100 \log m)/32} \right) \cdot \log_{8/7} m > 1 - \frac{1}{m^2}, \quad (C.3)$$

$O(\log_{8/7} |S|)$ iterations successfully terminate Algorithm 3.

\[\square\]

### C.2 Query Complexity in Sub-sampling

**Lemma 14.** With probability $1 - \frac{1}{m}$, Algorithm 3 correctly label the subsets $S_p$’s in all iterations by making $O\left( \frac{1}{\alpha^2} \cdot \log^2 m \cdot \log \log m \right)$ total calls to the $O_L$ and $O\left( \frac{1}{\beta^2} \cdot \log^3 m \cdot \log \log m \right)$ total calls to the $O_C$, where $m$ is the initial size of the input set $S$.

**Proof.** By Lemma 13 the iteration number of the while-loop in Algorithm 3 is $\log m$. In each iteration, suppose we invoke Algorithm 1 to correctly label $S_p$ that contains $b = 100 \log m$ instances with probability $1 - \beta'$. In order to guarantee that $S_p$’s are labeled correctly in all iterations, we let $\beta' \cdot \log m = \frac{1}{m}$. Then, in each iteration, the number of calls required to $O_L$ is

$$O\left( \frac{1}{\alpha^2} \cdot \log b \cdot \log \left( \frac{\log b}{\beta'} \right) \right) = O\left( \frac{1}{\alpha^2} \cdot \log m \cdot \log \log m \right).$$

Similarly, the number of calls required to $O_C$ is

$$O\left( \frac{1}{\beta^2} \cdot b \log b \cdot \log \left( \frac{b}{\beta'} \right) \right) = O\left( \frac{1}{\beta^2} \cdot \log^2 m \cdot \log \log m \right).$$

20
In conclusion, the total number of calls to \( O_L \) in all iterations is given by

\[
O \left( \frac{1}{\alpha^2} \cdot \log^2 m \cdot \log \log m \right),
\]

respectively, the total number of calls to \( O_C \) is given by

\[
O \left( \frac{1}{\beta^2} \cdot \log^3 m \cdot \log \log m \right).
\]

\[\square\]

### C.3 Analysis of \( S_I \)

**Lemma 15** ([Awasthi et al. 2017b], Lemma 4.6). For each instances in FILTER, if \( h^*(x) = h(x) \), it goes to \( S_I \) w.p. < \( \sqrt{\epsilon} \); if \( h^*(x) \neq h(x) \), it goes to \( S_I \) w.p. > 1/2.

**Proof.** Eventually, an instance \( x \) that goes to either \( S_I \) or \( S' \) will be removed from \( S \). As we showed in Lemma 13, the probability that labeler \( t \sim \mathcal{P} \) gives both correct comparison tags with respect to \( x^-, x^+ \) with probability > 0.7. Thus, if \( h^*(x) = h(x) \), consider \( t = N \), the majority of \( N = O(\log(1/\sqrt{\epsilon})) \) labels are correct with probability at least \( 1 - \sqrt{\epsilon} \), such an instance goes to \( S_I \) with probability at most \( \sqrt{\epsilon} \). If \( h^*(x) \neq h(x) \), again we consider a biased random walk that with probability > 0.7, the vote is correct and with remaining probability, it is wrong. We give the probability that the majority ever makes mistake, or the majority ever agrees with \( h_1 \), as Eq. [C.2] which is < \( \frac{3}{7} \). Therefore, for each \( x \) where \( h^*(x) \neq h(x) \), it goes to \( S_I \) w.p. at least \( \frac{3}{7} > \frac{1}{2} \).

**Lemma 16** ([Awasthi et al. 2017b], Lemma 4.7). With probability \( 1 - \exp(-\Omega(m \sqrt{\tau, \delta})) \), \( W_I, W_C \) and \( S_I \) all have size \( \Theta(m \sqrt{\tau, \delta}) \).

**Proof.** For ease of exposition, we assume that the error rate of \( h_1 \) is exactly \( \sqrt{\epsilon} \) as per Lemma 11. According to Lemma 15, we have

\[
O(m \sqrt{\tau, \delta}) \geq \frac{1}{2} \sqrt{\epsilon} \big| S_2 \big| + \sqrt{\epsilon} \big| S_2 \big| \geq \mathbb{E}[|S_I|] \geq \frac{1}{2} \left( \frac{1}{2} \sqrt{\epsilon} \right) \big| S_2 \big| \geq \Omega(m \sqrt{\tau, \delta}).
\]

Then, following the sampling process in Algorithm 2, we have

\[
O(m \sqrt{\tau, \delta}) \geq \mathbb{E}[|S_I|] + |S_C| \geq \mathbb{E}[|W_I|] \geq \frac{1}{2} \left( \frac{1}{2} \sqrt{\epsilon} \right) \big| S_2 \big| \geq \Omega(m \sqrt{\tau, \delta}),
\]

\[
O(m \sqrt{\tau, \delta}) \geq \mathbb{E}[|S_I|] + |S_C| \geq \mathbb{E}[|W_C|] \geq \left( 1 - \frac{1}{2} \sqrt{\epsilon} \right) \big| S_C \big| \geq \Omega(m \sqrt{\tau, \delta})
\]

The claim follows by the Chernoff bound.

\[\square\]

### C.4 Query Complexity in Filtering

**Lemma 17** (Restatement of Lemma 6). With probability at least \( 1 - O \left( \frac{1}{m^2} \right) \), the following holds. Algorithm 3 returns an instance set \( S_I \) with size \( m \sqrt{\epsilon} \) by making \( O \left( \frac{m \sqrt{\epsilon}}{\beta^2} \right) \) calls to the \( O_C \).

**Proof.** By Lemma 12, with probability at least \( 1 - \frac{1}{m^2} \), the interval \( [x^-, x^+] \) contains less than a 1/2 fraction of the unfiltered instances. By Lemma 13, with probability \( 1 - \frac{1}{m^2} \), \( O(\log m) \) iterations will terminate Algorithm 3.
When the interval \([x^-, x^+]\) contains less than a 1/2 fraction of the instances, the remaining 1/2 fraction are informative for testing hypothesis \(h_1\). For each of these instances, the expected number of iterations that it participated in is \(L < 2\), because in each iteration it gets removed with probability larger than 1/2.

By Chernoff bound, w.p. at least \(1 - \exp(-\Omega(|S|/\sqrt{\epsilon}))\), the number of instances that \(h_1\) will misclassify is \(\Theta(|S|/\sqrt{\epsilon})\), and w.p. at least \(1 - \exp(-\Omega(|S|))\), the number of instances that \(h_1\) correctly labels is \(\Theta(|S| (1 - \sqrt{\epsilon})) = \Theta(|S|)\).

If \(h_1(x) = h^*(x)\), let \(N_i\) be the expected number of queries until we have \(i\) more correct tag pairs than the incorrect ones. Then, \(N_1 \leq 0.7(1) + 0.3(N_2 + 1)\) and \(N_2 = 2N_1\), which gives \(N_1 \leq 2.5\). In other words, the inner For-loop breaks at the Filtering step by querying less than 2.5 workers. If the inner For-loop breaks at the Filtering step, \(x\) goes to \(S'\) and gets removed from \(S\). On the other hand, if \(x\) is mistakenly discard (reserved for the next iteration) and the inner loop breaks at one step before Filtering, the expected number of queries must be no more than \(N_1\) as well, because as an interruption to the filtering process, it always costs less queries in view of the law of total probability. Thus, for each \(x\) where \(h(x) = h^*(x)\), the expected number of queries we need is less than \(LN_1\), which is upper bounded by a constant. Apply the Bernstein’s inequality, for all \(\Theta(|S| (1 - \sqrt{\epsilon}))\) such instances, w.p. \(1 - \exp(-\Omega(|S|))\), the number of queries is \(O(|S|)\).

If \(h_1(x) \neq h^*(x)\), the number of queries needed for each \(x\) is no more than \(LN = O(\log(1/\epsilon))\) in Algorithm 3. Therefore, w.p. \(1 - \exp(-\Omega(|S|/\sqrt{\epsilon}))\), the total number of queries spent on these instances is at most \(O(|S|/\sqrt{\epsilon} \log(1/\epsilon)) \leq O(|S|)\).

On the other hand, for these instances that lie in the interval \([x^-, x^+]\), we will think of an \(x\) in different While-loop iterations as different test cases. Because of Lemma 12 and 13 we have the total size of such uninformative test cases no more than

\[
\frac{1}{2} \left( m + \frac{7}{8} m + \left( \frac{7}{8} \right)^2 m + \left( \frac{7}{8} \right)^3 m + \cdots + 1 \right) = O(m).
\]

In addition, for each of these test cases, let \(N'_1\) be the expected number of queries until we have one more correct tag pairs than the incorrect ones. Then, similar to the informative cases where \(h_1(x) = h^*(x)\), \(N'_1 \leq 2.5\). Again, we apply the Bernstein’s inequality and the total number of queries for these instances is no more than \(O(|S|)\) w.p. at least \(1 - \exp(-|S|)\).

Note that even though we only presented the dependence on \(m\) for the clarity of proof, we didn’t remove the \(\frac{1}{\sqrt{\epsilon}}\) factor because of the majority vote process. Thus, the total number of calls to \(\mathcal{O}_C\) is \(O\left(\frac{m}{\sqrt{\epsilon}}\right)\) with probability at least \(1 - O\left(\frac{1}{m^\epsilon}\right)\).

\[\square\]

C.5 Performance of \(h_2\)

**Lemma 18** (Restatement of Lemma 7 [Awasthi et al. 2017b, Lemma 4.8]). In Phase 2, with probability \(1 - O\left(\frac{1}{m^\epsilon}\right)\), \(\text{err}_\mathcal{D}_2(h_2) \leq \sqrt{\epsilon}\). In addition, when \(\log^2\left(\frac{d}{\sqrt{\epsilon}}\right) \leq \frac{1}{\sqrt{\epsilon}}\), the label complexity is \(O\left(\frac{1}{\alpha^2} \cdot \log^2 m_e \cdot \log \log m_e\right)\) and the comparison complexity is \(O\left(\frac{m}{\alpha^2}\right)\).

**Proof.** As guaranteed by Lemma 16 the size of \(\mathcal{W}\) is \(\Theta(m^\epsilon)\) almost surely. The error rate of classifier \(h_2\) on the distribution \(\mathcal{D}'\), which has equal probability on the distributions induced by \(\mathcal{W}_1\) and \(\mathcal{W}_C\), is \(\sqrt{\epsilon}\).

It remains to show that \(h_2\) achieves the matching error rate \(\sqrt{\epsilon}\) on the target distribution \(\mathcal{D}_2\) in phase 2. We apply the super-sampling technique proposed in [Awasthi et al. 2017b]. Let \(d'(x)\)
denote the density of point \( x \) in distribution \( D' \), likewise \( d_2(x) \) for points in \( D_2 \). It is sufficient to show that for any \( x \), \( d'(x) = \Theta (d_2(x)) \).

For ease of presentation, we again assume \( \text{err}_D(h_1) \) is exactly \( \sqrt{\epsilon} \). Let \( d(x), d_C(x), \) and \( \text{err}_D(x) \) be the density of instance \( x \) in distributions \( D, D_C, \) and \( D_I \), respectively. Note that, for any \( x \) such that \( h_1(x) = h^*(x) \), we have \( d(x) = d_C(x)(1 - \frac{1}{2}\sqrt{\epsilon}) \). Similarly, for any \( x \) such that \( h_1(x) \neq h^*(x) \), we have \( d(x) = \text{err}_D(x) \frac{1}{2}\sqrt{\epsilon} \).

Let \( N_C(x), N_I(x), M_C(x) \) and \( M_I(x) \) be the number of occurrences of \( x \) in the sets \( S_C, S_I, W_C \) and \( W_I \), respectively. For any \( x \), there are two cases:

If \( h_1(x) = h^*(x) \): Then, there exist absolute constants \( c_1 \) and \( c_2 \) according to Lemma 16 such that

\[
d'(x) = \frac{1}{2} \mathbb{E} \left[ \frac{M_C(x)}{|W_C|} \right] \geq \frac{\mathbb{E}[M_C(x)]}{c_1 \cdot m \sqrt{\epsilon}, \delta} \geq \frac{\mathbb{E}[N_C(x)]}{c_1 \cdot m \sqrt{\epsilon}, \delta} = \frac{|S_C| \cdot d(x)}{c_1 \cdot m \sqrt{\epsilon}, \delta},
\]

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

If \( h_1(x) \neq h^*(x) \): Then, there exist absolute constants \( c'_1 \) and \( c'_2 \) according to Lemma 16 such that

\[
d'(x) = \frac{1}{2} \mathbb{E} \left[ \frac{M_I(x)}{|W_I|} \right] \geq \frac{\mathbb{E}[M_I(x)]}{c'_1 \cdot m \sqrt{\epsilon}, \delta} \geq \frac{\mathbb{E}[N_I(x)]}{c'_1 \cdot m \sqrt{\epsilon}, \delta} \geq \frac{\frac{4}{7} d(x) |S_2|}{c'_1 \cdot m \sqrt{\epsilon}, \delta} \geq c'_2 d_I(x) = \frac{c'_2 d_2(x)}{2},
\]

where the second and sixth transitions are by the sizes of \( W_I \) and \( |S_2| \), the third transition is by the fact that \( h(x) \neq h^*(x), \) \( M_I(x) > N_I(x) \), and the fourth transition holds by part 2 of Lemma 15.

Given the above, with probability \( 1 - O \left( \frac{1}{m \sqrt{\epsilon}} \right) \), \( \text{err}_{D_2}(h_2) \leq \sqrt{\epsilon} \).

According to Lemmas 14 and 17, when \( \log^2 \left( \frac{d}{\sqrt{\epsilon}} \right) \leq \frac{1}{\sqrt{\epsilon}} \), the query complexity follows.

D Analysis of Phase 3

**Lemma 19** (Restatement of Lemma 8). In Phase 3, with probability \( 1 - O \left( \frac{1}{m \sqrt{\epsilon}} \right) \), \( \text{err}_{D_3}(h_3) \leq \sqrt{\epsilon} \). In addition, the label complexity is \( O \left( \frac{\log^2 m \sqrt{\epsilon}}{\alpha^2} \right) \) and the comparison complexity is \( O \left( \frac{m \sqrt{\epsilon} \log^2 m \sqrt{\epsilon}}{\beta^2} \right) \).

**Proof.** Similar to Proposition 11 in Phase 3, with probability at least \( 1 - \frac{\delta}{6} \), all instances in \( S_3 \) are correctly labeled. With probability at least \( 1 - \frac{\delta}{6} \), by VC theory (Theorem 22), oracle \( O_H \) return a classifier with error rate \( \leq \sqrt{\epsilon} \). By union bound, with probability at least \( 1 - \frac{\delta}{3} \), the error rate of \( h_3 \) is \( \text{err}_D(h_3) \leq \sqrt{\epsilon} \). Let \( \frac{\delta}{3} = \frac{1}{m} \), the complexities follow.

E Proof of Theorem 4

**Theorem 20** (Restatement of Theorem 4). With probability \( 1 - O \left( \frac{1}{m \sqrt{\epsilon}} \right) \), Algorithm 2 returns a classifier \( h \in H \) with error rate \( \text{err}_D(h) \leq \epsilon \) by making \( O \left( \frac{1}{\alpha^2} \cdot \log^2 m \epsilon \cdot \log \log m \epsilon \right) \) total calls to \( O_L \) and \( O \left( \frac{m \epsilon \sqrt{\epsilon}}{\alpha^2} \right) \) total calls to \( O_C \), given that \( \log^2 \left( \frac{d}{\sqrt{\epsilon}} \right) \leq \frac{1}{\sqrt{\epsilon}} \). Furthermore, Algorithm 2 runs in polynomial time.

23
Applying union bound, with probability at least \(1 - 1/m\), the total number of calls to \(O_L\) is \(O\left(\frac{1}{m} \cdot \log^2 m \cdot \log \log m \right)\), and the total number of calls to \(O_C\) is \(O\left(\frac{m}{\epsilon^2}\right)\), as long as \(\log^2\left(\frac{d}{\epsilon}\right) \leq \frac{1}{\sqrt{\delta}}\). In addition, in each phase, with high probability, the algorithm returns a classifier with error rate no more than \(\sqrt{\epsilon}\) on its corresponding distribution \(D_1, D_2\) or \(D_3\). Applying union bound, with probability at least \(1 - O\left(\frac{1}{m^2}\right)\), according Theorem 3 we conclude that the classifier \(h\) returned by Algorithm 3 has error < \(\epsilon\) on \(D\).

\[ \square \]

## F Useful Lemmas

**Lemma 21** (High-probability bound for Randomized QuickSort). With probability at least \(1 - 1/m\), the following holds. Given a set of instance \(S\) with \(m\) elements, the computation complexity of sorting all the elements by Randomized QuickSort is \(O(m \log m)\).

**Proof.** QuickSort is a recursive algorithm: in each round, it picks a pivot, splits the problem into two subsets, and continue playing QuickSort on each subset. The program keeps doing this until the inputs for all recursive calls contain only 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 \(i\)th level of recursion that contains \(t\). Obviously \(L_0 = m\), and

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

as with probability \(\frac{1}{2}\), the pivot is in the middle that ranks between \(\frac{1}{4}L_{i-1}\) and \(\frac{3}{4}L_{i-1}\); and with probability \(\frac{1}{4}\) the size of the subset does not shrink significantly.

Then, by the Tower rule, \(\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]\), we have

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

Let \(M = 3 \log_8/7 m\), we have

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

Applying Markov’s inequality, we have

\[ \Pr[L_M \geq 1] \leq \frac{\mathbb{E}[L_M]}{1} \leq \frac{1}{m^2}, \]

which is the probability that element \(t\) participates in more than \(M\) recursive calls. Summing up this probability on all \(m\) elements, the probability that any element participates in more than \(M = O(\log m)\) recursive calls is at most \((1/m^2) \cdot m = 1/m\). The claim in this lemma follows. \(\square\)

**Theorem 22** (Anthony and Bartlett [2002], VC Theory). Under the realizable PAC setting, for a hypothesis class \(\mathcal{H}\) with VC-dimension \(d\), it suffices to learn a classifier \(h \in \mathcal{H}\) with error rate \(\text{err}_D(h) \leq \epsilon\), with \(m_{\epsilon,d} = O\left(\epsilon^{-1} (d \log \frac{1}{\epsilon}) + \log \left( \frac{1}{\delta} \right) \right)\) instances drawn from underlying distribution \(D\) and correctly labeled according to the ground truth \(h^*\).
**Theorem 23** (Probability of Ruin). 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

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

**Theorem 24** (Bernstein Inequality). Let $X_1, \ldots, X_n$ be independent random variables with expectation $\mu$. Suppose that for some positive real number $L$ and every $k > 1$,

$$\mathbb{E}[(X_i - \mu)^k] \leq \frac{1}{2} \mathbb{E}[(X_i - \mu)^2] L^{k-2} k!.$$  \hfill (F.1)

Then,

$$\Pr \left[ \sum_{i=1}^{n} X_i - n\mu \geq 2t \sqrt{\sum_{i=1}^{n} \mathbb{E}[(X_i - \mu)^2]} \right] < \exp(-t^2), \text{ for } 0 < t \leq \frac{1}{2L \sqrt{\mathbb{E}[(X_i - \mu)^2]}}.$$ \hfill (F.2)