COMS 6998-4 Fall 2017 Presenter: Yuemei Zhang
November 27, 2017 Scribe: Che Shen

Learning from the Crowd

1 Introduction

In recent years crowdsourcing has become the method of choice for gathering labeled training data for learning algorithms. However, in most cases, there are no known computationally efficient learning algorithms that are robust to the high level of noise that exists in crowdsourced data, and efforts to eliminate noise through voting often require a large number of queries per example. In this note we will introduce a computationally efficient algorithm with much less overhead in the labeling cost. In particular, we mainly consider the case when a noticeable fraction of labelers are perfect, and the rest behave arbitrary. we show that any hypothesis space $\mathcal{F}$ that can be efficiently learned in the traditional realizable PAC model can be learned in a computationally efficient manner by querying the crowd, despite high amounts of noise in the responses.

2 The Setting

We consider the realizable PAC learning setting. Suppose we have an instance space $\mathcal{X}$ and labels $\mathcal{Y} = \{+1, -1\}$. Also, there is a distribution D over $\mathcal{X} \times \mathcal{Y}$ and a hypothesis class $\mathcal{F}$. We are considering the realizable setting. There exists $f^* \in \mathcal{F}$ such that $\text{err}_D(f^*) = 0$. Recall that in the traditional PAC learning setting, to learn $\mathcal{F}$ with ϵ accuracy and δ confidence, the label complexity is

$$m_{\epsilon,\delta} = O\left(\frac{d}{\epsilon}\left(\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\right)\right),$$

where d is the VC-dimension of $\mathcal{F}$.Furthermore, we assume that efficient algorithms for the realizable setting exist. That is, we consider an oracle $\mathcal{O}_{\mathcal{F}}$ that for a set of labeled instances S, returns a function $f \in \mathcal{F}$ that is consistent with the labels in S.

In crowdsourced labeling, we have a set of labelers L, in which each labeler i is a classification function $g_i : \mathcal{X} \to \mathcal{Y}$. A labeler i is called perfect if $\text{err}_D(g_i) = 0$. Also, we assume a uniform distribution P over all labelers, and let $\alpha = Pr_{i\ P}[\text{err}_D(g_i) = 0]$ be the fraction of perfect labelers. (So, when randomly draw a labeler, the probability of it being a perfect labeler is α) In this note, we mainly focus on the case when $\alpha > \frac{1}{2}$. In the following sections, unless explicit stated, we always assume $\alpha = \frac{1}{2} + \Theta(1)$.

3 A Baseline Algorithm

A quite natural algorithm is to get enough samples and label them with majority vote by labelers. If L is a set of labeler and x is an instance, we denote by $\text{Maj}_L(x)$ the majority vote of L on x.

Baseline Algorithm: Draw a sample of size $m = m_{\epsilon,\delta}$ from D_X and label each example x by $\text{Maj}_L(x)$, where L P_K for $k = O(\ln \frac{m}{\delta})$ is a set of randomly drawn labelers. Let S be the resulting labeled set. Return classifier $\mathcal{O}_{\mathcal{F}}(S)$.

It can be seen that this algorithm needs $O(m_{\epsilon,\delta} \ln \frac{m}{\delta})$ queries to the labelers, which is $\ln \frac{m}{\delta}$ times more than traditional PAC learning. We will try to improve this complexity.

4 An Interleaving Algorithm

We introduce a new algorithm for crowdsourced labeling that would significantly reduce the query complexity. We call it an **interleaving** algorithm because, different from the baseline algorithm that uses same amount of queries on every instance, this algorithm may use very few queries for some instances, while more queries on others.

This note will focus on the intuition of the theorems and lemmas, but the proof may be omitted. The detailed proof can be found in [1].

Before introducing the algorithm we need to show some basic techniques used in this algorithm.

4.1 Boosting

Boosting algorithms[2] provide a mechanism for producing a classifier of error ϵ using learning algorithms that are only capable of producing classifiers with considerably larger error rates. More specifically, the theorem in [2] states as follows

Theorem 1. *Boosting (Schapire 1990): For any $p < 1/2$ and distribution D, consider three classifiers:*

1. *h_1: $\text{err}_D(h_1) \leq p$;*
2. *h_2: $\text{err}_{D_2}(h_2) \leq p$, where $D_2 = \frac{1}{2}D_C + \frac{1}{2}D_I$, D_C is D conditioned on $\{x|h_1(x) = f^*(x)\}$, and D_I is D conditioned on $\{x|h_1(x) \neq f^*(x)\}$;*
3. *h_3: $\text{err}_{D_3}(h_3) \leq p$. D_3 is D conditioned on $\{x|h_1(x) \neq h_2(x)\}$.*

Then, the majority vote of h_1, h_2 and h_3 has error $\leq 3p^2 - 2p^3$ under distribution D.

This theorem is an important method that use weak classifiers (which requires less queries to find) to construct strong classifiers. However, it requires a distribution D_2 that pose equal weight on the area that h_1 is correct and that h_1 is incorrect. To construct this distribution, we need the following techniques.

4.2 Probabilistic Filtering

A naive way to simulate D_2 in the boosting algorithm is to get $O(\frac{1}{p}m_{p,\delta})$ samples and use majority vote to get their label, then choose equal number of samples from the area that h_1 is correct or incorrect. However in our case $p = O(\sqrt{\epsilon})$, so this naive method needs $O(m_{\epsilon,\delta}\ln(\frac{m_\epsilon}{\delta}))$ label queries, which is as large as that of baseline. In this section we introduce probabilistic filtering approach, called FILTER, which is shown as follows.

Algorithm 1 FILTER(S, h)

Let $S_I = \emptyset$ and $N = \log(1/\epsilon)$
for $x \in S$ **do**
    for $t = 1, 2, \ldots, N$ **do**
        Draw a random labeler i P and let $y_t = g_i(x)$
        if t is odd and $\text{Maj}(y_{1:t}) = h(x)$ **then**
            goto the next x
        end if
    end for
    Let $S_I = S_I \cup x$
end for
return S_I

Intuitively, this algorithm tries to maintain instances that are mislabeled by h_1 but discard instances that h_1 labels correctly. A more concise description of this property is proved in the following lemma

Lemma 1. *FILTER(S, h) has the following property:*
If $h_1(x) = f^*(x)$, then $x \in$ FILTER$(S,\ h_1)$ with probability $< \sqrt{\epsilon}$.
If $h_1(x) \neq f^*(x)$, then $x \in$ FILTER$(S,\ h_1)$ with probability $\geq 1/2$.

However, with only probabilistic filtering we still cannot precisely simulate D_2. To ensure that $\text{err}_{D_2}(h_2) \leq p$, we need another technique.

4.3 Super Sampling

Intuitively, super sampling states that, given a classifier h and two distributions D_1, D_2 such that D_1 is upper bounded by D_2 times a constant, then $\text{err}_{D_1}(h)$ is also bounded by $\text{err}_{D_2}(h)$ within a constant. We state it formally as follows:

Lemma 2. *Given a hypothesis class $\mathcal{F}$. consider any two discrete distributions D and D_0, with density function ρ and ρ' respectively, such that for all x, $\rho'(x) \geq c \cdot \rho(x)$ for an absolute constant $c > 0$, and both distributions are labeled according to $f^* \in \mathcal{F}$. There exists a constant $c' > 1$ such that for any ϵ and δ, with probability $1 - \delta$ over a labeled sample set S of size $c' m_{\epsilon,\delta}$ drawn from D', $\mathcal{O}_{\mathcal{F}}(S)$ has error of at most ϵ with respect to distribution D.*

Although we cannot precisely simulate D_2 required in the boosting algorithm, we can construct a distribution that has the density function within some constant to it. With these tools in hand, we are ready to introduce our algorithm.

4.4 The Interleaving Algorithm

Our algorithm is as follows.

Algorithm 2 Interleaving: boosting with probabilistic filtering

---

Input Given a distribution $D_{|\mathcal{X}}$, a class of hypotheses $\mathcal{F}$, parameters ϵ and δ. And a subroutine CORRECT-LABEL(S, δ) that label each instance in S with the majority vote of k labelers, where k = $O(\log \frac{|S|}{\delta})$

Phase 1:

1: Draw S_1 of size $2m_{\sqrt{\epsilon},\delta/6}$ from D
2: $\overline{S_1}$ = CORRECT-LABEL(S_1, $\delta/6$)
3: $h_1 = \mathcal{O}_{\mathcal{F}}(\overline{S_1})$

Phase 2:

4: Draw S_2 of size $\Theta(m_{\epsilon,\delta})$, S_C of size $\Theta(m_{\sqrt{\epsilon},\delta})$ from D
5: S_I = FILTER(S_2, h_1)
6: CORRECT-LABEL($S_I \cup S_C$, $\delta/6$)
7: Divide the labeled set into $\overline{W_I}$ and $\overline{W_C}$ according to whether the label agrees with h_1
8: Draw $\overline{W}$ of size $\Theta(m_{\sqrt{\epsilon},\delta})$ from a distribution that equally weights $\overline{W_I}$ and $\overline{W_C}$
9: $h_2 = \mathcal{O}_{\mathcal{F}}(\overline{W})$

Phase 3:

10: Draw S_3 of size $2m_{\sqrt{\epsilon},\delta/6}$ from D_3
11: $\overline{S_3}$ = CORRECT-LABEL(S_3, $\delta/6$)
12: $h_3 = \mathcal{O}_{\mathcal{F}}(\overline{S_3})$

It involves a subroutine CORRECT-LABEL(S, δ), which returns the correct labels of each instances in S with probability at least $1 - \delta$. The algorithm is divided into 3 phases. In phase 1, the algorithm creates a classifier h_1 with $\text{err}_D(h_1) < \sqrt{\epsilon}$. In phase 2, the algorithm uses FILTER to generate a distribution that has close weight on the instances that h_1 labels correct and incorrect, and use this distribution to construct a classifier h_2. In phase 3, the algorithm constructs a distribution on instances that is labeled incorrectly by h_1 by rejective sampling. It can be shown that h_1, h_2, h_3 satisfy the properties required in boosting algorithm and can be used to construct a classifier with lower error. This leads to our main theorem

Theorem 2. *Algorithm 2 returns $f \in \mathcal{F}$ with $\text{err}_D(f) \leq \epsilon$ with probability $1 - \delta$, using $O\left(m_{\sqrt{\epsilon},\delta} \log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right) + m_{\epsilon,\delta}\right)$ labels.*

It can be seen that, when $\frac{1}{\sqrt{\epsilon}} \geq \log\left(\frac{m_{\sqrt{\epsilon}}}{\delta}\right)$, the query complexity is equal to $O(m_{\epsilon,\delta})$, which is equal to the query complexity of traditional PAC learning.

We will prove this theorem through the following lemmas.

Lemma 3. *With probability at least* $1 - \exp(-\Omega(|S|\sqrt{\epsilon}))$, $\overline{W_I}, \overline{W_C}$ *and* S_I *all have size* $\Theta(m_{\sqrt{\epsilon},\delta})$.

This is easy to see by combining lemma 1 with Chernoff bound.

Lemma 4. *Let* D_C *and* D_I *denote distribution* D *when it is conditioned on* $\{x|h_1(x) = f^*(x)\}$ *and* $\{x|h_1(x) \neq f^*(x)\}$, *respectively, and let* $D_2 = \frac{1}{2}D_I + \frac{1}{2}D_C$, *then with probability* $1 - 2\delta/3$, $\text{err}_{D_2}(h_2) \leq \sqrt{\epsilon}/2$.

This is a direct result from super sampling. All we need to prove is that the density function of D_2 is within some constant factor of the density of $\overline{W}$. This is true because we use probabilistic filtering to intentionally choose $\overline{W}$ to have similar weight on D_I and D_C.

Lemma 5. *Let* S *be a sample set drawn from distribution* D, *with probability at least* $1 - \exp(-\Omega(|S|\sqrt{\epsilon}))$, *FILTER(*$S$, h_1*) makes* $O(|S|)$ *label queries.*

This is because the filter algorithm would soon discard an instance in D_C without querying too much labels. So the amortized label complexity is $O(1)$.
Now we are ready to prove theorem 2.
Proof of theorem 2: From the above lemmas, we can see that Phase 1 and Phase 3 use $O\left(m_{\sqrt{\epsilon},\delta} \log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right)\right)$ labels. In Phase 2, FILTER uses $O(m_{\epsilon,\delta})$ labels, and CORRECT-LABEL uses $O\left(m_{\sqrt{\epsilon},\delta} \log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right)\right)$ labels. So the total label complexity is $O\left(m_{\sqrt{\epsilon},\delta} \log\left(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}\right) + m_{\epsilon,\delta}\right)$.

References

[1] Pranjal Awasthi, Avrim Blum, Nika Haghtalab, and Yishay Mansour. Efficient pac learning from the crowd. *arXiv preprint arXiv:1703.07432*, 2017.

[2] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.