Efficient PAC Learning from the crowd

Ian Gomez

December 29, 2021

Contents

notes were made after I read the Intro and notation however the core idea is that we want to create an algorithm to efficiently learn from a crowd of generated data where we cannot assure how correct a label is from a given labeler, and we want to do this with low label cost, and high accuracy. The main idea is to use boosting methods and interleave the data collection process with the boosting algorithm since most papers focused on data collection and learning from noisy data as seperate tasks. this paper novelly interleaves both to make an algorithm which has better performance.

1 3. A baseline algorithm

- The baseline algorithm is simple in that we query k workers and take the majority vote from those k workers for each sample in the dataset.
- The main problem is that the amount of workers needed to statistically guarentee all labels are correct scales in $O(log(m/\delta))$
- Second when the number of perfect labelers $< 1/2$ there can be errors in the dataset which means the final classifier will have bad performance
- To solve this we mix probablistic filtering and boosting algorithms to solve the problems mentioned prior

- This also provides a good way of finding perfect labelers in the sense that we can query labelers on a test set of images and iteratively discard labelers who get a label wrong.

2 4. An Interleaving Algorithm.

- For this we utilize research done by Schapire on boosting algorithms. Specifically we use the fact that with 3 classifiers you can achieve O(p^2) error with classifiers that get error p = 0.5 - y where y is small.
- Schapire proved that with 3 classifers h1, h2, h3 where $err_D(h_1) \leq p$, $err_{D2}(h2) \leq p$, $err_{D3}(h3) \leq p$ that $err_D(Maj(h_1, h_2, h_3)) \leq O(p^2)$
- As opposed to the originally learning p = 0.5 - y we extend this to any p where we sample $m_{p,\delta}$. Where as p increases the label cost decreases.
- We particularly use this to learn classifier h_1 of $O(\sqrt{\epsilon})$ using $O(m_{\sqrt{\epsilon},\delta})$ that are labeled with $O(log(m_{\sqrt{\epsilon},\delta}))$
- Given classifier h_1 which is trained on dataset D. we construct D_2 such that $D_2 = \frac{1}{2}D_I + \frac{1}{2}D_C$
- The reasoning is that we want to get informative samples which h_1 will perform poorly on.
- However, when constructing D_2 it becomes difficult to construct a dataset such that half of the samples incorrect since it will only be incorrect with probability P.
- Therefore we must construct a faster filtering algorithm.
- Filter algorithm
- The idea is that we gather $N = log(\frac{1}{\epsilon})$ workers and for each sample we compute the majority of $L_1...L_k$ such that k is odd. if $Maj(L_1...L_k) = h_1(x)$ then stop for that sample.
- If we manage to iterate through all N then we add the sample to D_I
- This algorithm makes it so that D_I is found with constant overhead to the original label complexity

- due to the fact that eventually the majority will equal the classifier prediction given some classification accuracy > 0.5 for all correctly labeled samples.
- Lemma 4.9 proves that this happens with a constant time in most cases.
- For the case where we find incorrectly labeled samples it is very unlikely that the labeler gets it wrong since the probability that the majority is incorrect is small
- Because the probability of being wrong is 1-p therefore the probability that the majority over all iterations is wrong is very small.
- This is proved in Lemma 4.6
- In essence we are super sampling with the filter algorithm. The idea is that when in the realizable setting we can super sample from D_2 such that we can construct a new distribution which is a constant density higher than that of the original D_2. It is proven that this super sampled dataset performs with a constant performance.
- We then arrive at this algorithm.

Algorithm 2 INTERLEAVING: BOOSTING BY PROBABILISTIC FILTERING FOR $\alpha = \frac{1}{2} + \Theta(1)$

Input: Given a distribution $D_{|\mathcal{X}}$, a class of hypotheses $\mathcal{F}$, parameters ϵ and δ.

Phase 1:
- Let $\overline{S_1} = \text{CORRECT-LABEL}(S_1, \delta/6)$, for a set of sample S_1 of size $2m_{\sqrt{\epsilon},\delta/6}$ from $D_{|\mathcal{X}}$.
- Let $h_1 = \mathcal{O}_{\mathcal{F}}(\overline{S_1})$.

Phase 2:
- Let $S_I = \text{FILTER}(S_2, h_1)$, for a set of samples S_2 of size $\Theta(m_{\epsilon,\delta})$ drawn from $D_{|\mathcal{X}}$.
- Let S_C be a sample set of size $\Theta(m_{\sqrt{\epsilon},\delta})$ drawn from $D_{|\mathcal{X}}$.
- Let $\overline{S_{All}} = \text{CORRECT-LABEL}(S_I \cup S_C, \delta/6)$.
- Let $\overline{W_I} = \{(x, y) \in \overline{S_{All}} \mid y \neq h_1(x)\}$ and Let $\overline{W_C} = \overline{S_{All}} \setminus \overline{W_I}$.
- Draw a sample set $\overline{W}$ of size $\Theta(m_{\sqrt{\epsilon},\delta})$ from a distribution that equally weights $\overline{W_I}$ and $\overline{W_C}$.
- Let $h_2 = \mathcal{O}_{\mathcal{F}}(\overline{W})$.

Phase 3:
- Let $\overline{S_3} = \text{CORRECT-LABEL}(S_3, \delta/6)$, for a sample set S_3 of size $2m_{\sqrt{\epsilon},\delta/6}$ drawn from $D_{|\mathcal{X}}$ conditioned on $h_1(x) \neq h_2(x)$.
- Let $h_3 = \mathcal{O}_{\mathcal{F}}(\overline{S_3})$.

return $\text{Maj}(h_1, h_2, h_3)$.

CORRECT-LABEL(S, δ)**:**

for $x \in S$ **do**
- Let $L \sim P^k$ for a set of $k = O(\log(\frac{|S|}{\delta}))$ labelers drawn from P and $\overline{S} \leftarrow \overline{S} \cup \{(x, \text{Maj}_L(x))\}$.

end

return $\overline{S}$.

- which has $\Lambda = O(\sqrt{\epsilon} log(\frac{m_{\sqrt{\epsilon},\delta}}{\delta}) + 1)$ label cost. where this is constant if $1/\sqrt{\epsilon} \geq log(...)$

- Insert lots of proofs on why this works.
- First proof is on why Filter behaves well
- Second proof is on why $err(h_2) = O(\sqrt{e})$
- Third proof is on why filter only has few queries
- Final proof is that we have low label cost.

3 4.1 the general case of any α

- Here we generalize the original algorithm to work for any α
- The two main challenges are that we cannot assume that the majority will choose the right label which is important in $CORRECT - LABEL(S, \delta)$
- Also filter does not correctly split the data into H_I
- We overcome this with a few tricks
- **Pruning**
- As alluded to we can utilize certain test tasks to generate a set of perfect labelers.
- This can be used to remove bad labelers.
- If we make golden queries $Maj - size_p(x) \leq \frac{\alpha}{2}$
- we only need to repeat this process $O(\frac{1}{\alpha})$ times
- To measure the majority size to find when to prune we can only query if there is a certain level of uncertainty in the majority prediction
- e.g. there if there is a large majority we keep the label, however, if there is not we test and prune the labelers.
- Robust super sampling.
- the main problem is that any good test case can be filtered into the wrong set.
- Therefore what we do is we find a small set such that is unlikely that the set has a lot of good test cases.

- We then perform filter and because there are not that many test cases our splitting is good enough for the robust super sampling property.
- Whenever we prune the labelers we must reset the algorithm to ensure that our distributions are good.
- Algorithm 3

Algorithm 3 BOOSTING BY PROBABILISTIC FILTERING FOR ANY α

Input: Given a distribution $D_{|\mathcal{X}}$ and P, a class of hypothesis $\mathcal{F}$, parameters ϵ, δ, and α.

Phase 0:

If $\alpha > \frac{3}{4}$, run Algorithm 2 and quit.

Let $\delta' = c\alpha\delta$ for small enough $c > 0$ and draw S_0 of $O(\frac{1}{\epsilon}\log(\frac{1}{\delta'}))$ examples from the distribution D.

PRUNE-AND-LABEL(S_0, δ').

Phase 1:

Let $\overline{S_1}$ = PRUNE-AND-LABEL(S_1, δ'), for a set of sample S_1 of size $2m_{\sqrt{\epsilon},\delta'}$ from D.

Let $h_1 = \mathcal{O}_{\mathcal{F}}(\overline{S_1})$.

Phase 2:

Let S_I = FILTER(S_2, h_1), for a set of samples S_2 of size $\Theta(m_{\epsilon,\delta'})$ drawn from D.

Let S_C be a sample set of size $\Theta(m_{\sqrt{\epsilon},\delta'})$ drawn from D.

Let $\overline{S_{All}}$ = PRUNE-AND-LABEL$(S_I \cup S_C, \delta')$.

Let $\overline{W_I} = \{(x, y) \in \overline{S_{All}} \mid y \neq h_1(x)\}$ and Let $\overline{W_C} = \overline{S_{All}} \setminus \overline{W_I}$.

Draw a sample set $\overline{W}$ of size $\Theta(m_{\sqrt{\epsilon},\delta'})$ from a distribution that equally weights $\overline{W_I}$ and $\overline{W_C}$.

Let $h_2 = \mathcal{O}_{\mathcal{F}}(\overline{W})$.

Phase 3:

Let $\overline{S_3}$ = PRUNE-AND-LABEL(S_3, δ'), for a sample set S_3 of size $2m_{\sqrt{\epsilon},\delta'}$ drawn from D conditioned on $h_1(x) \neq h_2(x)$.

Let $h_3 = \mathcal{O}_{\mathcal{F}}(\overline{S_3})$.

return Maj(h_1, h_2, h_3).

PRUNE-AND-LABEL(S, δ)**:**

for $x \in S$ **do**

Let $L \sim P^k$ for a set of $k = O(\frac{1}{\alpha^2}\log(\frac{|S|}{\delta}))$ labelers drawn from P.

if Maj-size$_L(x) \leq 1 - \frac{\alpha}{4}$ **then**

Get a golden query $y^* = f^*(x)$,

Restart Algorithm 3 with distribution $P \leftarrow P_{|\{(x,y^*)\}}$ and $\alpha \leftarrow \frac{\alpha}{1-\frac{\alpha}{8}}$.

else

$\overline{S} \leftarrow \overline{S} \cup \{(x, \text{Maj}_L(x))\}$.

end

end

return $\overline{S}$.

4 5. No perfect lablers exist

- In the case where no perfect labelers exist. We must find a set of good labelers to label our data so that the majority of labelers are correct.
- This is essentially the agnostic PAC learning framework, and we assume that labelers will be correct with some probability $P(x) > \epsilon$ and bad labelers with $P(x) > 1 - 4 * \epsilon$

- Finally we construct the algorithm that where we try and identify a set of labelers who agree with eachother with some high frequency and and since they should be correct they will form the highest component of a graph

- Algorithm 4

Algorithm 4 GOOD LABELER DETECTION

Input: Given n labelers, parameters ϵ and δ

Let $G = ([n], \emptyset)$ be a graph on n vertices with no edges.

Take set Q of $16\ln(2)n$ random pairs of nodes from G.

1 **for** $(i, j) \in Q$ **do**

  if DISAGREE$(i, j) < 2.5\epsilon$ **then** add edge (i, j) to G;

end

2 Let $\mathcal{C}$ be the set of connected components of G each with $\geq n/4$ nodes.

3 **for** $i \in [n] \setminus \left(\bigcup_{C \in \mathcal{C}} C\right)$ *and* $C \in \mathcal{C}$ **do**

  Take one node $j \in C$, if DISAGREE$(i, j) < 2.5\epsilon$ add edge (i, j) to G.

end

return *The largest connected component of* G

DISAGREE(i, j)**:**

Take set S of $\Theta(\frac{1}{\epsilon}\ln(\frac{n}{\delta}))$ samples from D.

return $\frac{1}{|S|}\sum_{x \in S} \mathbb{1}_{(g_i(x) \neq g_j(x))}$.