Avoiding Imposters and Delinquents: Adversarial Crowdsourcing and Peer Prediction

Sree Priyanka Uppu
(uppu@usc.edu)

Aim

The problem addressed is that of a crowdsourcing model in which n workers are asked to rate the quality of m items previously generated by the workers. Let αn be the set of reliable workers while the remaining may be arbitrary or adversarial. The Manager of the experiment can evaluate some items which are of high quality. The result is that the dataset can be curated with $\tilde{\mathcal{O}}(\frac{1}{\beta\alpha^3\epsilon^4})$ ratings per worker and $\tilde{\mathcal{O}}(\frac{1}{\beta\epsilon^2})$ ratings per manager to output the set of high quality items.

1 Introduction

The main idea of the paper is ensure that we obtain reliable information from humans. In order to ensure quality of the raters, earlier work included the use of gold sets which has the correct answers to all the questions. But, this approach fails in the sense that, if the answers were in terms of sentences translating a para can be interpreted in multiple ways and hence, there will not be a right way to judge a rater. Also, this approach fails to prevent collusion in peer grading for online MOOCs.

Challenge: The main challenge is to use the crowdsourced human ratings to accurately and efficiently evaluate a large dataset of content. For example, in case of peer grading for MOOCs the goal would be to accurately evaluate each and every student. Similarly, in curation of large dataset the goal would be to output a high quality subset.

However, in this problem there are various issues as well. Firstly, the raters are unreliable and evaluation are uncorrelated with the actual item quality. Reliable raters may be harsher or lenient and similarly the items may be harder to evaluate. Lastly, the reliable raters may collude or hack

the system. So, the main important question to answer would be to find out the info from reliable raters without knowing who they are.

The author has considered some weak assumptions like that the majority of workers are reliable and their ratings are approximate to the true ratings of the items. There exists a gold set of items with known ratings presented to each user. Unreliable workers can be adversarial, collude among themselves, etc.

2 Algorithm, Setting and Intuition

Below is the setting of the problem and the explanation of the variables used in the algorithm:

n = number of raters
m = number of items to be evaluated. Assigned rating or quality level for each item $\in [0, 1]$
αn = the minimum number of reliable workers (Reliable workers are defined as the ones who's rating significantly matches with the rating given by the manager or in this case it is "us").
k = maximum number of randomly selected items assigned to each worker for their rating
k_0 = maximum number of randomly selected items assigned us for the rating.

$\tilde{A} \in [0, 1]^{n \times m}$ = Ratings matrix. The ratings of each rater are stored in this matrix with the rows denoting the rater and column denoting the items and hence the size of the matrix is $n \times m$. If the rater i does not rate item j then, $A_{ij} = 0$ else assign the rating given by the rater i for the item j$\in [0, 1]$
Note: $\tilde{A}$ is very sparse since each rater rates only few items. If we do not rate item j then, $A_{ij} = 0$ else assign the rating given us for the item j$\in [0, 1]$

$\tilde{r} \in [0, 1]^{1 \times m}$: Rating matrix given by "us" for k_0 items. If we do not rate item j then, $\tilde{r_j} = 0$ else assign the rating given us for the item j$\in [0, 1]$

$r^* = \in [0, 1]^m$ denote the vector of true ratings of the items.
<a,b> = For any vector a and b, this denotes the dot product between them.

Goal: Recover $\beta - quantile$: the set T^* of the βm highest quality items according to "my" rating.
Intuition:
The algorithm proposed is based on the intuition that the relaible raters ap-

proximately agree on the ranking of the items. So, if we cluster rows of $\tilde{A}$, then the cluster pertaining to the reliable raters will be the single very large cluster of size αn. Hence, we can say that there may be atmost $1/\alpha$ disjoint clusters. We can identify the reliable raters cluster by randomly selecting ratings from each cluster and checking their agreement with "our" own ratings and then choose the best one.

Challenge:

The biggest challenge is that to implement the intuition stated above, we would be needing a fully filled ratings matrix, but in our setting since each rater only rates few items we get a ratings matrix $\tilde{A}$ which is very sparse. Hence, any two rows in the ratings matrix would most probably have no ratings in common. The author views the problem as a noisy matrix completion. Imagine a matrix A^* in which all the ratings have been filled. We define M^* that indicated the top βm items in each row of A^* such that: M^*_{ij} $= 1$ if item j is one of the top βm ratings from rater i and $M^*_{ij} = 0$, otherwise. From the M^* we can obtain the set T.

The author has described 3 algorithms to solve the problem in our above defined setting. Below is the summary of each algorithm.

Algorithm 1. To construct the ratings matrix $\tilde{A}$ and $\tilde{r}$

1. Input: n,m,k,k_0
2. Initially, assign each rater to each item with probability =k/m.
3. For each rater $>$ 2k items, unassign items arbitrarily until there are 2k items for each rater.
4. For each item $>$ 2k raters, unassign items arbitrarily until there are 2k raters for each item.

5. Have the raters submit their ratings of the assigned items, and let $\tilde{A}$ be the resulting ratings matrix. If the rater i does not rate item j then, $A_{ij} = 0$ else assign the rating given by the rater i for the item j$\in [0, 1]$
6. Similarly, generate $\tilde{r}$ by rating items with probability k_0/m. If we do not rate item j then, $\tilde{r_j} = 0$ else assign the rating given us for the item j$\in [0, 1]$.
7. Output $\tilde{A}$ and $\tilde{r}$.

Algorithm 2. To construct the $\beta - quantile$ matrix $\tilde{M}$ using the ratings matrix $\tilde{A}$

---

1. Parameters: $\alpha, \beta, \epsilon(tolerance), n, m$
2. Input: noisy ratings matrix $\tilde{A}$
3. Let $\tilde{M}$ be the solution of the optimization problem:

$$\begin{gathered} \text{maximize } <\tilde{A},\text{M}>, \\ \text{subject to } 0 \leq M_{ij} \leq 1, \forall i, j \\ \textstyle\sum_j M_{ij} \leq \beta m \ \forall j, \ ||M||_* \leq \frac{2}{\alpha\epsilon}\sqrt{\alpha\beta nm}, \\ \text{where, } ||.||_* \text{ denotes the nuclear norm.} \end{gathered}$$

4. Output $\tilde{M}$.

To avoid adversarial behavior, the author has used nuclear norm and row-normalization. The importance of nuclear norm is that, consider the case when, $\alpha = \beta \quad and \quad m = n$. And there are adversarial raters who assigns 1 to his assigned set of items, and this generated $1/\alpha$ symmetrical blocks.

Since, we would not know the good blocks, we need to have them in solution M^*. Here, $||M^*||_* = n,\sqrt{\alpha\beta nm} = \sqrt{\alpha^2 n^2} = \alpha n$. Hence, the nuclear norm is atleast $1/\alpha$ times larger that $\sqrt{\alpha\beta nm}$ to capture the solution M^*.

If the row normalization constant is not chosen, $\sum_j M_{ij} \leq \beta m$, this might happen if instead of getting all items of quality above a given quantile, we sought all items above a given threshold say $1/2$. It may lead to focusing on quality thresholds rather than quantile thresholds which in turn loses the robustness to monotonic transformations.

Algorithm 3. To recover an accurate $\beta - quantile$ T from the $\beta - quantile$ matrix $\tilde{M}$

---

1. Parameters: $\alpha, \epsilon(tolerance)$
2. Input: Matrix $\tilde{M}$ of approximate $\beta - quantile$, "our" ratings $\tilde{r}$
3. Select $2log\frac{2/\delta}{\alpha}$ indices $i \in [n]$at random.
4. Score each row of $\tilde{M}$ for the selected i as $\sum_j \tilde{M}_{ij} and \tilde{r_j}$.
5. Select T_0= the row with the highest score as calculated from above.
6. Round $T_0 \in [0,1]^m$ to $T \in \{0,1\}^m$ using Randomzied Rounding Algorithm.
4. Output T.

Summary: In summary, we first generate the noisy matrix $\tilde{A}$ and $\tilde{r}$ using Algorithm-1. Then, with $\tilde{A}$, we run Algorithm-2 to recover $\beta - quartile\tilde{M}$ for each rater. And finally, we run Algorithm-3 to recover our personal $\beta - quartile$ using $\tilde{M}$ which is the set T^*.

3 Assumptions and Approach

We can query:

1. a rater $i \in [n]$ and item $j \in [m]$ to obtain a rating $\tilde{A}_{ij} \in [0,1]$.
2. an item j rated by us, to get a $\tilde{r}_j$ such that $E[\tilde{r}_j] = r_j^*$.

Let $\mathcal{C} \subset [n]$ be the set of reliable raters, such that $|\mathcal{C}| \geq \alpha n$.Below are the assumptions made by the author:

1. Independence: Reliable raters make independent errors. When we query either for the rater and the item or query the item from the rating matrix by us, the queries are independent of all other queries asked so far.This allows that the unreliable raters can depend on all previous ratings or collusion among themselves.

We now need a way to identify that reliable raters agree with us. This can be done by defining the monotonic property. The idea is that if we think that one item say a, is significantly better than item b, then even the reliable raters should think the same and the ratings given by them should match with us by (L, ϵ) monotonic property. In the case, where the reliable ratings are not monotonic, this property fails to identify the set of reliable raters.

Formerly defining the same, We say that the reliable raters are (L, ϵ) monotonic if and only if,

$$r_j^* - r_{j'}^* \leq L.(A_{ij} - A_{ij'}) + \epsilon \tag{1}$$

whenever $r_j^* > r_{j'}^*$ for all $i \in C$ and all $j, j^{'} \in [m]$.

Theorem 2. *Let $m \geq n$, Suppose Assumption 1 holds, and the reliable raters are (L, ϵ) monotonic and we run Algorithm 1 to obtain noisy ratings. Then there is $k = \mathcal{O}(\frac{log^3(2/\delta)}{\beta\alpha^3\epsilon^4}\frac{m}{n})$ and $k_0 = \mathcal{O}(\frac{log(2/\alpha\beta\epsilon\delta)}{\beta\epsilon^2})$ such that with probability $1 - \delta$, Algorithm 2 and Algorithm 3 output a set T with:*

$$\frac{1}{\beta}(\sum_{j \in T^*} r_j^* - \sum_{j \in T} r_j^*) \leq (2L + 1).\epsilon + 2\epsilon_0 \tag{2}$$

The dependence on m/n ensures that more work is assigned to a worker if there are more items than workers, ensuring that all items are evaluated by atleast one worker.

Below is an outline on how to prove the Theorem 2:

1. Analyze Algorithm 2, recovering $\tilde{M}$
2. Analyze Algorithm 3, recovering T

4 Recovering $\tilde{M}$ Algorithm 2

In this part the author shows that the nuclear norm constraint imparts sufficient noise robustness while at the same time it does not allow the adversary to have too much influence. Consider a concept class $\mathcal{C}$ which has all the reliable raters.

The author conveys through *Proposition -1* that the row $\tilde{M}_i$ is good according to the rater i's ratings A_i^* (which is the gold set) and this implies that it is also good according to r^*. It can be proved by using Lipschitz bound.

The above proposition can be proved by showing that (a) nuclear norm constraint imparts noise robustness. (b) The constraint does not allow the adversaries to influence $\tilde{M}_C$ too much.

(a)Noise robustness: The author constructs a matrix B such that $B_C = \frac{k}{m}A^*_C, B_{\overline{C}} = \tilde{A}_{\overline{C}}$. Let R be a gold set ratings matrix of true values. Then, T^* will be approximately close to R. Ideally, for the concept class covering all the reliable raters $M_{\mathcal{C}} = R_{\mathcal{C}}$. Which also means that M^* is also close to T^*. Hence, if $\tilde{M}$ performs nearly as good as the M^* then the following inequality holds from Holder's inequality, uniform deviation result, matrix concentration inequality bounding and Theorem 3:

$$< B, M > \quad \geq \quad < B, M^* > -\epsilon\alpha\beta kn \tag{3}$$

(b) Bounding the influence of adversaries: Say if the nuclear norm constraint is not present, then the adversaries would have no influence on $M^*_{\mathcal{C}}$. Using Lagrangian duality(This allows to replace constraint with appropriate modifications), we can show the effect of nuclear norm constraint.

From the Lagrangian multiplier $Z_{\mathcal{C}}$ which bounds the amount that <B,M> can increase due to changing M outside of $\mathcal{C}$.It can be seen that the below Lemma 2 bounds the effect that the adversaries can have on $\tilde{M}_{\mathcal{C}}$ by localizing $< B, M^* - M >$ to $\mathcal{C}$.

$$< B_{\mathcal{C}} - Z_{\mathcal{C}}, M^*_{\mathcal{C}} - M_{\mathcal{C}} > \quad \leq \quad < B, M^* - M > \tag{4}$$

5 Recovering T Algorithm 3

In this part the author proves by using of concentration inequalities and a standard randomized rounding idea. The goal is to get a set T which is a close approximation of T^* (This $1Xm$ matrix has the true ratings of all items and it is not given to us).

Proposition 2. *Suppose Assumption 1 holds. For some* $k_0 = \mathcal{O}(\frac{log(2/\alpha\beta\epsilon\delta)}{\beta\epsilon^2})$ *with probability* $1-\delta$*, Algorithm 3 outputs a set T satisfying,*

$$< T^* - T, r^* > \leq \frac{2}{|\mathcal{C}|}(\sum_{i\in C} < T^* - \tilde{M}_i, r^* >) + \epsilon\beta m \tag{5}$$

This can be proved in 2 parts:

1. Establish concentration bound showing that $\sum_j \tilde{M}_{ij}\tilde{r}_j$ is close to $\frac{k_0}{m}\sum_j \tilde{M}_{ij}r*_j$ for any fixed i.

2. By union bound, the above bound should hold for any randomly selected i from $\frac{2log(2/\delta)}{\alpha}$ indices selected as part of Algorithm 3.

Lemma 3. *Let i^* be the selected row from Algorithm-3 step 5. For some $k_0 = \mathcal{O}(\frac{log(2/\alpha\beta\epsilon\delta)}{\beta\epsilon^2})$ with probability $1-\delta$ we have,*

$$< T^* - \tilde{M}_{i^*}, r^* > \leq \frac{2}{|\mathcal{C}|}(\sum_{i \in \mathcal{C}} < T^* - \tilde{M}_i, r^* >) + \frac{\epsilon}{4}\beta m \tag{6}$$

Proof This is an application of Bernstein's inequality.

We get $T_0 = \tilde{M}_{i^*}$, and $T_0 \in [0,1]^m$. We need to convert T_0 into a binary vector, $T \in \{0,1\}^m$ by using a randomized rounding algorithm.

Algorithm 4. Randomized rounding algorithm matrix $\tilde{A}$

---

1. Input: T_0
2. Let s= vector of partial sums of $T_0, s_j = (T_0)_1 + (T_0)_2 + ... + (T_0)_j$
3. Sample u Uniform[0,1]
4. Initialize vector T to be filled with 0's of size 1 X m
5. For z=0 to $\beta m - 1$, Find j such that, $u + z \in [s_{j-1}, s_j)$ set $T_j = 1 else skip$
6. return T

The above is a generic randomized rounding algorithm.

6 Open Directions

1. Improve the dependence of k on α, β, ϵ.
2. Scaling in n: if n»m, can we get the output with much less work per rater?
3. Adaptivity: work on reducing the amount of work done by each Rater.

7 Comments

Further thought is, I feel that this algorithm cannot be used to curate the reviews from places like Amazon or Yelp, etc which involve Natural Language Processing of the comments. It can be more applied where the numerical ratings are more important like in the case of peer grading of assessment.I felt that providing an implementation of a real world application would have helped in understanding it easier. The author has introduced many variables

and has failed to define some of them in the paper. Overall, it was interesting for me to learn how to extract meaningful information from a noisy sparse matrix by performing matrix completion.