An Algorithm for Crowdsourcing with Hard and Easy Tasks

R. Srikant

University of Illinois at Urbana-Champaign

Coauthors

Seo Taek Kong

Saptarshi Mandal

Dimitrios Katselis

Outline

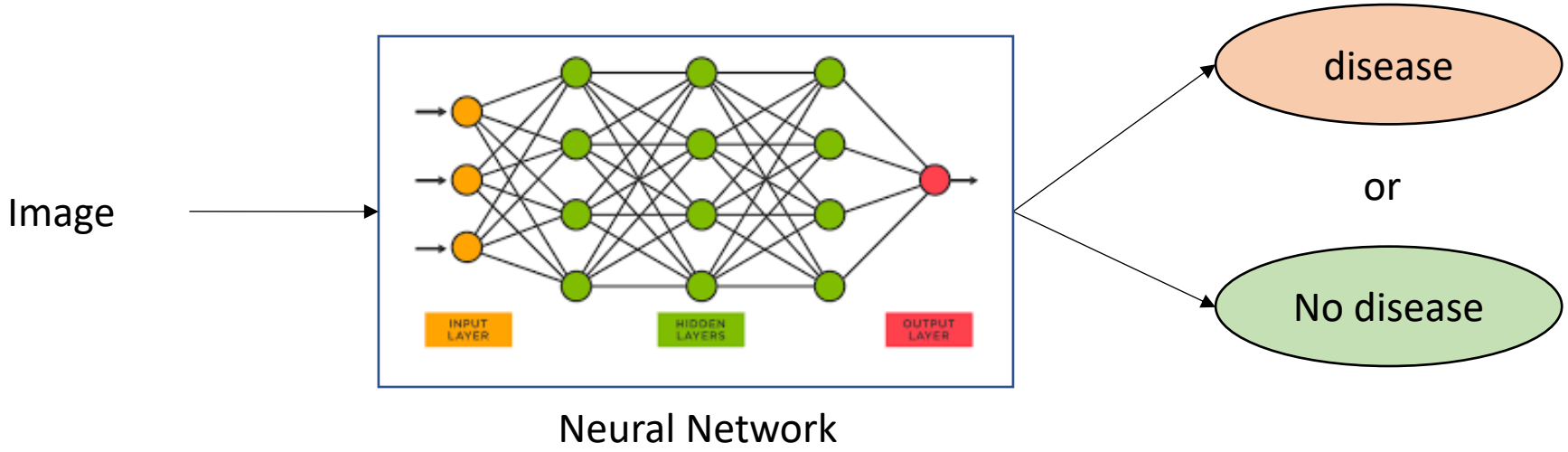
- Background
- Our Model and Solution
- Experiments
- Conclusions

Outline

- Background
- Our Model and Solution
- Experiments
- Conclusions

Supervised Machine Learning

- Classification tasks
  - Train a neural network to classify images as +1 or -1
  - We will only consider binary classification here
  - Example: +1: dog, -1: cat
  - Example: medical images, +1: disease, -1: no disease

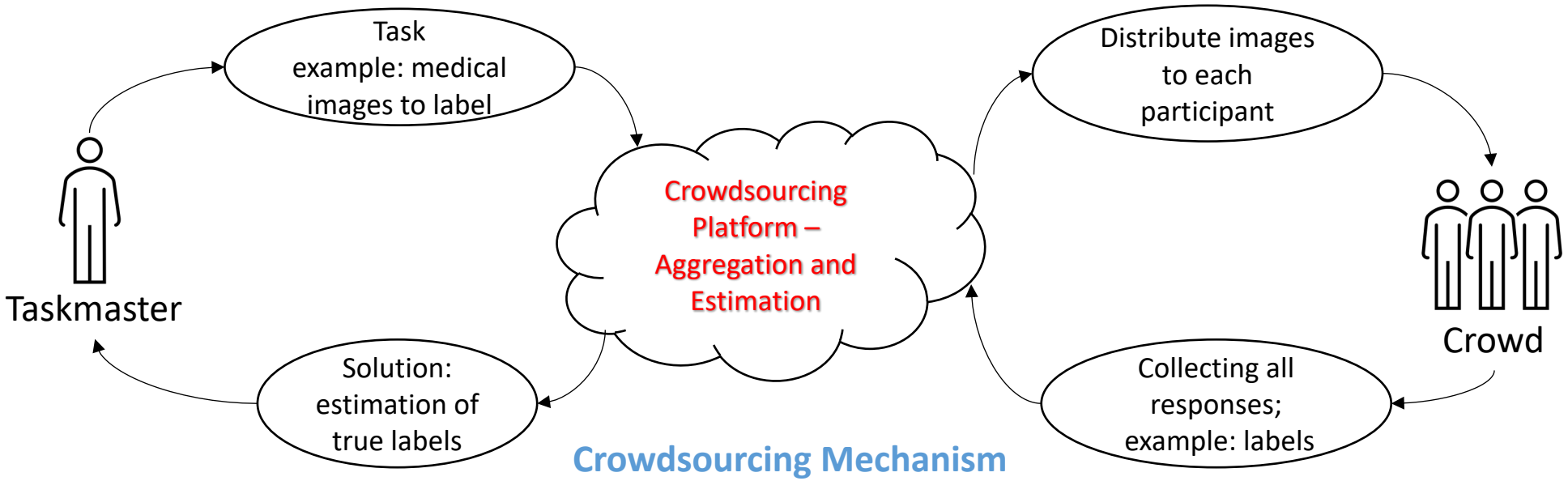


Training the Network Needs Labeled Data

- Learning the function in a supervised framework needs a labeled dataset
- Dataset : $D = (x_i, y_i)_1^n$; $x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$
- x_i corresponds to an image and y_i is its label
- Today's talk: How do we get these labels?

Crowdsourcing

- One popular technique : Crowdsourcing
- Crowdsourcing example :
  - We might want each medical image to be labeled by 10 doctors
  - Then estimate the true label for that image from those (noisy) labels.



Crowdsourcing Mechanism

Crowdsourcing Data

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Worker 1	+1	-1	-1	+1	-1	+1	-1	-1
Worker 2	-1	-1	+1	+1	+1	+1	+1	-1
Worker 3	-1	-1	-1	+1	-1	-1	+1	+1
Worker 4	-1	+1	+1	-1	-1	+1	+1	-1

- Example: Task: Image, Worker: Doctor
- Images may be hard to classify even for doctors. So how do we infer the true labels?
- Simplest and most natural scheme: Majority vote
- In this case, that would yield -1, -1, +/-1, +1, -1, +1, +1, -1
- Question: can we do better?

Dawid-Skene Model (Dawid and Skene, 1979)

- Suppose that each worker i correctly labels a task with probability p_i
  - Models the fact that some workers may be better at labeling tasks than others
- Suppose we know the p_is, what should we do?
- Weighted majority voting: if O_{ij} is the label given by worker i to task j, estimate the true label to be

$$\operatorname{sgn}\left(\sum_i w_i O_{ij}\right)$$

- We should give higher weights to the labels of the better workers
  - But what should the weights be? Assume $\frac{1}{n}\sum_i p_i > \frac{1}{2}$

Maximum Likelihood Estimator

- If p_is are known and the goal is maximum likelihood estimation, then the weights are given by (*Nitzan and Paroush,1982*) :

$$w_i = \log\left(\frac{p_i}{1-p_i}\right)$$

- MLE finds the labels that maximize the probability of observing the given dataset

- But that may not be the objective we are interested in

Objective

- We want as many correct labels as possible

- Minimize the fraction of incorrect labels :

$$\mathbb{E}(\mathcal{E}_{CS}) = \mathbb{E}\left(\frac{1}{d}\sum_{j=1}^{d} \mathbb{I}_{\hat{y}_j \neq y_j}\right) = \frac{1}{d}\sum_{j=1}^{d} \mathbb{P}(\hat{y}_j \neq y_j)$$

Optimality of NP-WMV for

- Gao et al.(2016) prove that the best error rate that any algorithm can achieve is given by the following error bound:

$$\mathbb{E}(\mathcal{E}_{CS}) \geq \exp(-nI(p))$$

  - $I(p)$ can be thought of as the collective ability of the workers given by:

$$I(p) = -\frac{1}{n}\sum_i \log\left(2\sqrt{p_i(1-p_i)}\right)$$

- NP-WMV achieves this optimal error rate asymptotically (*Bonald and Combes, 2017*)

But we don't know p_i …

- All we have is the observation matrix (a.k.a. data matrix)

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Worker 1	+1	-1	-1	+1	-1	+1	-1	-1
Worker 2	-1	-1	+1	+1	+1	+1	+1	-1
Worker 3	-1	-1	-1	+1	-1	-1	+1	+1
Worker 4	-1	+1	+1	-1	-1	+1	+1	-1

Some Algorithms for Crowdsourcing

One class of algorithms:

1. First, estimate the p_is from data
2. Then plug in those estimates of p_is to estimate true labels
   - Bonald and Combes, 2017 (TE)
   - Dalvi et al., 2013 (Spectral estimation of p_is)
   - Zhang et al, 2014 (Spectral+EM)
   - Ma, Olshevsky, Saligrama and Szepesvari 2020 (Low-rank matrix estimation)
   - Karger, Oh and Shah, 2011 (Message passing)
   - …

Outline

- Background
- Our Model and Solution
- Experiments
- Conclusions

Our Model

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Worker 1	+1	-1	-1	+1	-1	+1	-1	-1
Worker 2	-1	-1	+1	+1	+1	+1	+1	-1
Worker 3	-1	-1	-1	+1	-1	-1	+1	+1
Worker 4	-1	+1	+1	-1	-1	+1	+1	-1

- Recall that to do better than simple majority voting, we assumed worker i labeled a task correctly w.p. p_i
- But what if some tasks are harder to classify than others for the workers
  - In medical images, the size of a nodule may make it hard to detect cancer
- So, we use an extended Dawid-Skene model

An Extended D-S Model

- Assume there are two types of tasks (hard and easy)
  - For simplicity, we will assume there are only two task types.

- Worker i labels a hard task correctly with probability p_{hi} and an easy task correctly with probability p_{ei}
  - Assume $\frac{1}{n}\sum_i p_{ei} > \frac{1}{2}$ and $\frac{1}{n}\sum_i p_{hi} > \frac{1}{2}$

- The same task may be hard for someone and easy for someone else

What do hard and easy mean?

- Reliability of a worker for an easy task is defined as $r_{ei} = (2p_{ei} - 1)$
  - Let r_e be the reliability vector for easy tasks and r_h be for hard tasks

- Assumption: $\|r_e\| > \|r_h\|$
  - "On average", across workers, we assume some tasks are harder than others

- Given this model, as before, we will assume more information and try to solve the problem. Then, we will solve it in a data-driven fashion

Data Matrix Plus Side Information

- As before, let's first consider the case where we assume more information than we have, obtain the solution for that, and then consider the real problem

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Worker 1	+1	-1	-1	+1	-1	+1	-1	-1
Worker 2	-1	-1	+1	+1	+1	+1	+1	-1
Worker 3	-1	-1	-1	+1	-1	-1	+1	+1
Worker 4	-1	+1	+1	-1	-1	+1	+1	-1
	H	**E**	**H**	**E**	**E**	**E**	**E**	**H**

- Suppose we know which tasks are easy and which tasks are hard
- We would apply a single-type DS algorithm to the easy tasks and a single-type DS algorithm for the hard tasks
- But we don't know which tasks are easy and which are hard, what do we do?

Spectral Clustering

- Let O be the observation matrix
- It is easy to show that $E(O^T O)$ has rank 2 and the principal eigenvector has a special structure (modulo sign determined by the labels) with $v_e > v_h$

$$\begin{bmatrix} v_h \\ v_e \\ v_h \\ v_e \\ v_e \\ v_e \\ v_e \\ v_h \end{bmatrix}$$

The Algorithm for Hard-Easy Model

- Separation step : Cluster the tasks by type as follows:
  1. Get the principal eigen-vector : $v\left(\frac{1}{n}O^{\top}O\right)$
  2. Estimate the threshold for clustering : $\hat{\mu} = \frac{1}{d}\sum_j |v_j|$
  3. Clustering: $type_j = \begin{cases} e, & if\ |v_j| \geq \hat{\mu} \\ h, & else \end{cases}$

- Label Estimation:
  1. Separate the observation matrix by estimated types
  2. Use any standard single-type algorithm for each type separately

The Algorithm for Hard-Easy Model (example)

- A simple example: Two workers, Four tasks

- Suppose the observation matrix given to us is

$$O = \begin{bmatrix} +1, +1, -1, +1 \\ +1, -1, -1, -1 \end{bmatrix}$$

The Algorithm for Hard-Easy Model (example)

$$O = \begin{bmatrix} +1, +1, -1, +1 \\ +1, -1, -1, -1 \end{bmatrix} \rightarrow \frac{1}{2} O^{\top} O = \begin{bmatrix} +1, +0, -1, +0 \\ +0, +1, +0, +1 \\ -1, +0, +1, +0 \\ +0, +1, +0, +1 \end{bmatrix} \rightarrow v\left(\frac{1}{n} O^{\top} O\right) = [0.71, 0.00, -0.71, 0.00]^{\top}$$

Cluster the entries of $|v|$

Easy set = {1,3} ;
Hard set = {2,4}

$\widehat{O^e} = \begin{bmatrix} +1, -1 \\ +1, -1 \end{bmatrix}$ ← Apply TE+WMV

$\widehat{O^h} = \begin{bmatrix} +1, +1 \\ -1, -1 \end{bmatrix}$ ← Apply TE+WMV

Theorem 1

Error rate for our algorithm:

- Let k_j be the type of task j

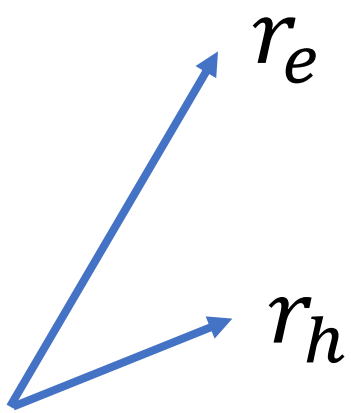
$$P\left(\frac{1}{d}\sum_j I\left(k_j \neq \hat{k}_j\right) \geq \delta\right) \leq d\exp(-n\gamma K(r_h, r_e)\delta)$$

- For the tasks whose types are estimated correctly,

$$\mathbb{P}\left(\hat{y}(j) \neq y_j \middle| \tilde{k}(j) = k(j)\right) \leq \exp(-n(1-\gamma)I(r_k)(1-o(1)))$$

Take-Away Messages

- We need the Clustering error, i.e., task type estimation error, to be small



- The more different the norms are or the more orthogonal the vectors are, the better it is for clustering

Take-Away Messages

- For the tasks whose types are estimated correctly,

$$\mathbb{P}\left(\hat{y}(j) \neq y_j \middle| \tilde{k}(j) = k(j)\right) \leq \exp(-n(1-\gamma)I(r_k)(1-o(1)))$$

- Each task type's label error is limited by the ability of workers to correctly label this type of task

Take-Away Messages

- For the tasks whose types are estimated correctly,

$$\mathbb{P}\left(\hat{y}(j) \neq y_j \middle| \tilde{k}(j) = k(j)\right) \leq \exp(-n(1-\gamma)I(r_k)(1-o(1)))$$

- Each task type's label error is limited by the ability of workers to correctly label this type of task

- Next, we will see how this is better than an algorithm in which we don't try to separate the task types

What if we use a single weight for each user?

- Our algorithm separates the task types and uses weighted majority voting for each type
- What if we do not separate the tasks and use a single weighted majority vote?
- Consider the following example:

p_{ij}	$Task\ 1\ (easy)$	$Task\ 2\ (hard)$	$Task\ 3\ (easy)$	$Task\ 4\ (hard)$
$Worker\ 1$	**0.5**	**0.8**	**0.5**	**0.8**
$Worker\ 2$	**0.9**	**0.4**	**0.9**	**0.4**

What if we use a single weight for each user?

p_{ij}	$Task\ 1\ (easy)$	$Task\ 2\ (hard)$	$Task\ 3\ (easy)$	$Task\ 4\ (hard)$
$Worker\ 1$	0.5	0.8	0.5	0.8
$Worker\ 2$	0.9	0.4	0.9	0.4

- If we use a single weight across both task types, the following situation occurs:
  - More weight to worker 1, the performance will not be good on easy tasks
  - More weight to worker 2, the performance will not be good in hard tasks
  - This is strictly worse than using optimal weights for each type separately

Performance of Type-Agnostic Algorithms

Theorem 2

For plug-in single-type algorithms:

$$\mathbb{E}(\mathcal{E}_{CS}) \geq exp\left(-n \min_k \varphi(w^*, r_k)(1 + o(1))\right)$$

where, $\min_k \varphi(w^*, r_k)$ acts as the average ability of the crowd given by

$$\varphi(w, r_k) = -\min_{t>0} \frac{1}{n} \sum_{i=1}^{n} \log\left(\frac{1}{2}\exp(tw_i)(1 - r_{ki}) + \frac{1}{2}\exp(-tw_i)(1 + r_{ki})\right)$$

$$w^* = arg \max_w \min_k \varphi(w, r_k)$$

Parsing the Theorem

- Recall the optimal error rate possible for any type k is given by the exponent of $I(p) = -\frac{1}{n}\sum \log\left(2\sqrt{p_i(1-p_i)}\right)$
- We can show that the optimal error rate for an easy task is smaller than the hard ones, i.e, $I(p_e) > I(p_h)$
- We can also show that $I(p_e) > I(p_h) \geq \min_k \varphi(w^*, r_k)$
  - This implies if we use a single-type DS-based algorithm for Hard-Easy model, the labeling error rate is even worse than $\exp(-nI(p_h))$ which is the rate we have if we have only hard tasks

Proof Sketch of Theorem 2

- We are using weighted majority voting with weight w_i for worker i
- Notice we express the error probability in labeling as

$$\mathbb{P}\left(\hat{y}_j \neq y_j\right) = \mathbb{P}\left(\sum_{i=1}^{n} w_i G_{ij} > 0\right)$$

$$\text{where } G_{ij} = \begin{cases} 1, & with\ probability\ \frac{1}{2}(1 - r_{k_j i}) \\ -1, & with\ probability\ \frac{1}{2}(1 + r_{k_j i}) \end{cases}$$

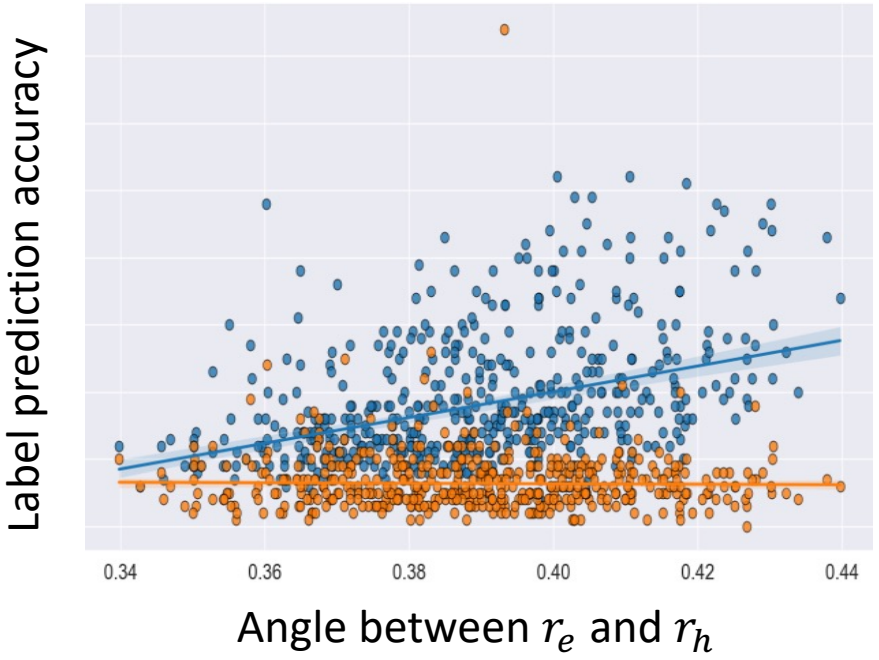
- Next use standard large deviation lower-bound technique (Cramer-Chernoff theorem) to bound this expression

Outline

- Background
- Our Model and Solution
- Experiments
- Conclusions

Simulations : Transition from DS to HE

As we can see in the figure, as the angle between r_e and r_h increases, our algorithm tends to perform significantly better than the DS-based algorithms



Blue : our algorithm
Orange: DS based algorithm(TE+NP-WMV)

Simulations: Performance Gain on error rate

- We compared the performance of different single-type crowdsourcing algorithms with and without separation for different datasets

- The algorithms used for comparisons are :
  1. Majority Voting (MV)
  2. Ratio of Eigen-vectors (ER) (Dalvi et al. , 2013)
  3. Triangular Estimation (TE) (Bonald and Combes, 2017)
  4. Projected Gradient Descent (PGD) (Ma et al., 2022)

Simulations: Performance Gain on error rate

DATASET	MV	ER	TE	PGD
JSRT-2-U	5.65	5.65	4.74	5.06
JSRT-2-S	5.65	4.39	3.16	3.81
GAIN	0.00	1.26	1.58	1.25
JSRT-6-U	10.30	10.30	9.96	9.72
JSRT-6-S	10.30	10.02	9.84	9.76
GAIN	0.00	0.28	0.12	-0.04

- Dataset : The radiography dataset to identify pneumonia reported in Makhnevich et al., 2019
  - 20 workers and 200 images

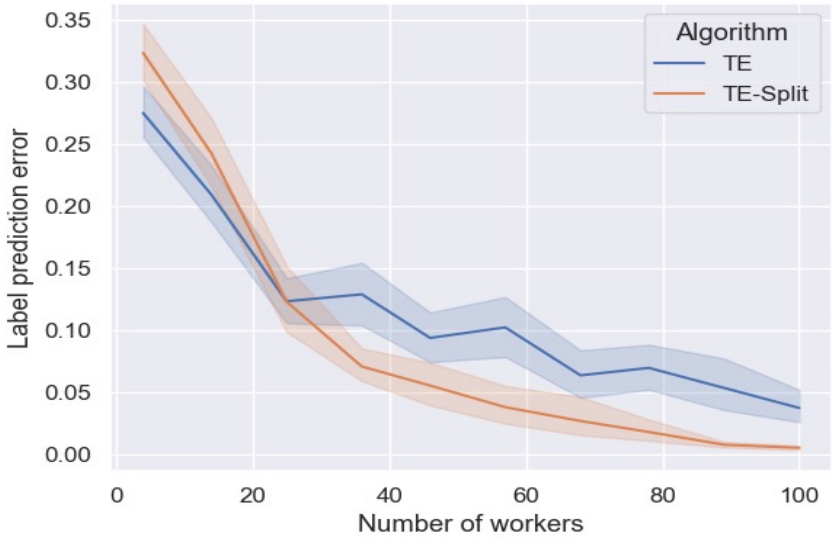
Simulations: Performance Gain on error rate

DATASET	MV	ER	TE	PGD
BIRD-U	24.07	28.70	17.59	25.00
BIRD-S	24.07	11.11	12.96	19.44
GAIN	0.00	16.59	4.63	5.56
DOG-U	23.28	23.32	35.38	20.06
DOG-S	23.28	7.11	14.40	21.72
GAIN	0	16.21	20.98	1.66

- Datasets:
  1. Blue-Bird(Welinder et al., 2010)
     - 39 workers, 108 tasks
  2. Dog(Deng et al., 2009)
     - 78 workers, 807 tasks
     - 4 classes converted into two groups to perform binary classification

Simulations: Performance Gain on error rate

- To compare our algorithm with DS-based algorithms, we used the radiography dataset to identify pneumonia reported in Makhnevich et al., 2019
- For small values of n, TE performs better than our algorithm and for larger n, our algorithm is better than TE (transition at n around 20)



Blue - TE: DS-based algorithm
Orange – TE-split: our algorithm

Pneumonia experiments : A comparison of TE with and without separation.

(Intuition Behind Proof of) Theorem 1

Error rate for our algorithm:

- Let k_j be the type of task j

$$P\left(\frac{1}{d}\sum_j I\left(k_j \neq \hat{k}_j\right) \geq \delta\right) \leq d\exp(-n\gamma K(r_h, r_e)\delta)$$

- For the tasks whose types are estimated correctly,

$$\mathbb{P}\left(\hat{y}(j) \neq y_j \middle| \tilde{k}(j) = k(j)\right) \leq \exp(-n(1-\gamma)I(r_k)(1-o(1)))$$

Proof Sketch

- First, we characterize the effect of the deviation from the expected case on spectral clustering

- Write $O^\top O$ as

$$\frac{1}{n} O^\top O = \frac{1}{n} \mathbb{E}(O^T O) + noise$$

Proof Sketch

- We know that $\frac{1}{n}\mathbb{E}(O^T O)$ is a rank-2 matrix with the special property of its principal eigenvector v:
  - Each element of $\left|v\left(\frac{1}{n}\mathbb{E}(O^T O)\right)\right|$ is either v_e or v_h with $v_e > v_h$
  - If j^{th} task is easy, j^{th} element of $\left|v\left(\frac{1}{n}\mathbb{E}(O^T O)\right)\right|$ is v_e; else, it is v_h

- Using Davis-Kahan theorem, we can now bound the gap between the eigenvectors of the two matrices : $\frac{1}{n}O^\top O$ and $\frac{1}{n}\mathbb{E}(O^T O)$

Proof Sketch

$$\frac{1}{n}O^{\top}O = \frac{1}{n}\mathbb{E}(O^{T}O) + noise$$

- Using Davis-Kahan Theorem on Matrix Perturbation,

$$\left\| v\left(\frac{1}{n}O^{\top}O\right) - v\left(\frac{1}{n}\mathbb{E}(O^{T}O)\right)\right\| \leq \frac{K\left\|\frac{1}{n}\left(O^{\top}O - \mathbb{E}(O^{T}O)\right)\right\|}{\lambda_1\left(\frac{1}{n}\mathbb{E}(O^{T}O)\right) - \lambda_2\left(\frac{1}{n}\mathbb{E}(O^{T}O)\right)}$$

  - λ_1 and λ_2 are the two largest eigenvalues of $\frac{1}{n}\mathbb{E}(O^{T}O)$

Proof Sketch

$$\frac{1}{n}O^{\top}O = \frac{1}{n}\mathbb{E}(O^{T}O) + noise$$

- Next, we can use Matrix Concentration Inequality to show that

$$\left\|\frac{1}{n}(O^{\top}O - \mathbb{E}(O^{T}O))\right\| \to 0 \text{ exponentially fast in } n$$

Proof Sketch

- The principal eigenvector of $\frac{1}{n} O^\top O$ is exponentially close to its expectation
- From this, we can derive a tail bound on the clustering error:

$$P\left(\frac{1}{d}\sum_j I\left(k_j \neq \hat{k}_j\right) \geq \delta\right) \leq d\exp(-n\gamma K(r_h, r_e)\delta$$

- Clustering performance is good if the gap $|v_e - v_h|$ is large
  - If $|v_e - v_h|$ is large, $K(r_h, r_h)$ is also large
  - It turns out that $K(r_h, r_e)$ is an increasing function of the expression:

$$\left|\frac{\|r_e\|^2 - \|r_h\|^2}{r_e^\top r_h}\right|$$

Outline

- Background
- Our Model and Solution
- Experiments
- Conclusions

Conclusions

- We extended the traditional DS model to be more appropriate for some crowdsourcing applications
  - Allowed more than one type of task
- Proposed a spectral clustering algorithm that clusters tasks by difficulty with an improved performance guarantee
  - Separate types before label estimation
  - Provably better than without separation when there are “many” workers
- What’s next? A data-driven algorithm to determine whether we should use separation or stick to a single-type DS-based algorithm for a given data matrix