Resilient Structures and Robust Machine Learning Algorithms

by

Shelby Lynn Heinecke
B.S., Massachusetts Institute of Technology, 2013
M.S., Northwestern University, 2015

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mathematics
in the Graduate College of the
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:
Prof. Lev Reyzin, Chair and Advisor
Prof. Gyorgy Turan
Prof. Yu Cheng
Prof. Will Perkins
Prof. Tanya Berger-Wolf, The Ohio State University

To my husband and our cat.

ACKNOWLEDGMENTS

I have been blessed to have mentors, family, and friends whose support empowered me to complete this thesis. First and foremost, I thank my advisor Lev Reyzin for taking me under his wing and guiding me in my research. I am thankful for his patience with me and for his unending optimism that always reignited my motivation. He went above and beyond in supporting my research and career development, encouraging me to participate in conferences, workshops, and internships. From his keen mathematical intuition to his ability to purify complicated problems to their core, his technical prowess inspires me and I hope to one day achieve his level of mastery. I thank György Turán for inspiring me to pursue research topics in theoretical computer science through his excellent graduate courses and our research discussions. In addition to being one of the most thorough and talented researchers, György is also kind and encouraging. His welcoming attitude was a light in my life in times of doubt. I thank Tanya Berger-Wolf for being a role model and for giving me honest academic and career advice I needed to hear. I thank Tanya for her thoughtful research feedback that helped me to consider new potential research directions. I thank the other members of my committee Yu Cheng and Will Perkins for their support and feedback. Finally, I thank Avrim Blum for elucidating and insightful research discussions.

My family has given me unconditional love and support throughout my PhD and I would not have been able to complete this thesis without them. Thanks to my parents for their support of my ambitious academic endeavors throughout my entire life. They recognized my

potential first and went out of their way to ensure I was in the right environments to realize my potential. I would like to thank my husband Will for believing in me, encouraging me everyday, and making me laugh. I thank my in-laws for their unwavering love and support. Finally, I'd like to thank my cat Chipotle for bringing joy and laughter to our home everyday.

I am lucky to have supportive and brilliant friends who push and inspire me. Amanda Bower has been one of the strongest positive forces in my life during my PhD. Her friendship, encouragement, and advice during difficult times were critical to the completion of my PhD. I thank my best friends from UIC, Karly Brinla and Mojgan Mirzaei, for their friendship and for productive study sessions early in our PhD. I thank my best friend Sarah Van Allen for collaborating on math research with me years ago and for cheering me on during my graduate school years. I thank my friend Horalia Armas for her constant enthusiastic support and our regular motivational discussions that energize me to continue pursuing my dreams.

During my PhD, I participated in internships that were vital to my immersion in machine learning research and data science. I thank Intel for three internships spanning an array of applied data science problems. These internships gave me hands on experience with machine learning which helped solidify and deepen my understanding. In particular, I'd like to thank Devin Cornish whose mentorship and ingenious career advice pushed me to excel technically even beyond these internships. I thank IBM Research, and specifically, my mentor Naoki Abe, for thought provoking research discussions and exposure to new topics in machine learning. I thank Salesforce Research and my mentor Huan Wang for the opportunity to extend my

research interests to new and exciting topics in AI. These internships expanded my horizons and reinvigorated my passion for machine learning and AI.

A huge thanks to my undergraduate institution, MIT, for the challenges that made me stronger and for the supportive programs that helped me to be successful at MIT and beyond. In particular, the Laureates & Leaders program run by the Office of Minority Education was crucial to my pursuit of graduate school. The program demystified graduate school, removed financial barriers, and set me on a path to make graduate school a reality. Without this program, I may not have made it this far. I thank mentors from my MIT years, such as David Jordan, Jeremy Orloff, and Richard Melrose, who believed in me and gave me opportunities to continue developing as a mathematician.

SLH

CONTRIBUTION OF AUTHORS

Chapter 2 represents the paper *On the Resilience of Bipartite Networks* by Shelby Heinecke, Will Perkins, and Lev Reyzin [1]. Will Perkins and Lev Reyzin completed a preprint of this work under the same title [2]. I added to their preprint by contributing Theorem 3, Section C, and the code written for computations (summarized in Appendix A), resulting in the joint publication [1]. The full joint publication, including both their contributions and mine, is represented in Chapter 2.

Chapter 3 represents the paper *Crowdsourced PAC Learning under Classification Noise* by Shelby Heinecke and Lev Reyzin [3]. All content including the introduction, literature review, formulations of definitions, theorems, algorithms, and writing of the manuscript was done jointly with the coauthor.

Chapter 4 represents joint work with Avrim Blum and Lev Reyzin. The content including the introduction, literature review, formulations of definitions, theorems, and algorithms was done jointly with the coauthors.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CN	Classification Noise
ERM	Empirical Risk Minimizer
IoT	Internet of Things
MAB	Multi-Armed Bandits
PAC	Probably Approximately Correct

SUMMARY

Networks and learning algorithms are key themes in artificial intelligence (AI). The nodes and edges in networks can be used to model the connections of a vast community of internet-of-things (IoT) devices, the internet, and distributed databases, among other important computing contexts for AI. Learning algorithms, which generate trained models, can be strategically designed to learn from a variety of data-rich networks, such as sensor networks, device networks, or even networks of humans working on crowdsourcing tasks. In real-world settings, noise generated from sources such as device inaccuracy or human error are unavoidable. Likewise, attacks in real-world networks, such as the injection of malware, are inevitable due to their ever-evolving sophistication.

In this thesis, we study the intrusions of noise and malicious attacks on networks and learning from networks. We devise structural and algorithmic solutions for mitigating the effects of these unwanted intrusions. In Chapter 2, we consider viral propagation under the independent cascade model of infection spread on half-regular bipartite networks and characterize the most resilient structures. We then shift to the learning setting in Chapters 3 and 4, where we study learning from data-rich networks in the presence of noise. In Chapter 3, we study learning and generalizing from a network of crowd workers, where crowd workers provide erroneous labels to unlabelled data at fixed, unknown error rates, known as the classification noise model. In this setting, we develop a three-step probably approximately correct (PAC) algorithm that incorporates majority voting, pure-exploration bandits, and noisy-PAC learning and demonstrate

our algorithm's improvement over baseline approaches. In Chapter 4, we study learning from a network of participants, each with their own distribution on the unlabelled data, under the classification noise model. We develop collaborative PAC algorithms robust to classification noise and prove sample complexity bounds. We also study the communication complexity of collaborative PAC learning, with and without classification noise, and develop communication efficient algorithms in both settings.

CHAPTER 1

INTRODUCTION

Networks and learning algorithms are key themes in AI. A network consists of nodes connected by either directed or undirected edges. In the context of AI, nodes can represent people, objects, entities, or devices, among other representations. The work in this thesis considers networks where nodes represent sources of data. Edges in a network can represent physical connections, such as computers connected by cables, or abstract connections, such as humans belonging to the same social network. Therefore, networks can model the connections of a vast community of IoT devices, the internet, and distributed databases, among other important computing contexts for AI. Attacks in real-world networks, such as the injection of malware, are inevitable due to their ever-evolving sophistication. In Chapter 2, we consider viral propagation in a particular class of networks, half-regular bipartite networks, and determine which half-regular bipartite structures yield the smallest expected fraction of infected nodes. The results in this chapter inform on how to best arrange nodes and edges of a network in the presence of viral propagation. Chapter 2 is essentially self-contained and an introduction of the necessary background is deferred to that chapter.

Chapters 3 and 4 focus on algorithms that learn from networks in the presence of noise. Learning algorithms, which generate trained models, can be strategically designed to learn from a variety of data-rich networks, such as sensor networks, device networks, or even networks of humans working on crowdsourcing tasks. In real-world settings, noise generated from sources

such as device inaccuracy or human error are unavoidable. The work in these chapters depend on fundamentals of learning theory. We review the necessary background in this section. A more thorough foundation in learning theory can be found in [4, 5].

1.1 Notation

We begin with basic notation used throughout this thesis. Let X denote the instance space consisting of all possible examples. Let Y denote the set of all possible labels for the data in X. In this thesis, $Y = \{0, 1\}$ since we deal primarily with binary classification. A *concept*, or *hypothesis*, h, is a mapping from X to Y. A concept class C is a set of concepts that a learning algorithm attempts to learn. Let $c^* \in C$ denote the target concept. A hypothesis class H is a set of hypotheses given to a learning algorithm. In general, the goal of the learning algorithm is to select some $h \in H$ that approximates $c^* \in C$. The precise learning criteria of interest in this thesis is discussed in the next sections.

To evaluate the cost of a learning algorithm, we focus on the resources consumed asymptotically with respect to the relevant parameters. We use the conventional Big-O and Big-Omega notation to describe the quantity of resources consumed asymptotically.

Definition (Big-O Notation [5])**.** *We say that* $f(n) = O(g(n))$ *if there exists* $c_0, n_0 > 0$ *such that for all* $n \geq n_0$, $f(n) \leq c_0 g(n)$.

Definition (Big-Omega Notation [5])**.** *We say that* $f(n) = \Omega(g(n))$ *if there exists* $c_0, n_0 > 0$ *such that for all* $n \geq n_0$, $f(n) \geq c_0 g(n)$.

Definition (Big-Theta Notation [5]). *We say that* $f(n) = \Theta(g(n))$ *if* $f(n) = O(g(n))$ *and* $f(n) = \Omega(g(n))$. *If* $f(n) = \Theta(g(n))$, *then we say that the bounds for* $f(n)$ *are asymptotically tight.*

We sometimes use the notation $\tilde{O}$ to indicate Big-O notation with hidden polylogarithmic factors of present parameters. The main resource we will be concerned with is the number of samples consumed by an algorithm. This is defined precisely as the sample complexity and we review the related notions in the next section.

1.2 Sample Complexity

We now review the definitions of sample complexity of a learning algorithm.

Definition (Sufficient Sample Size [5]). *A* sufficient sample size *for a learning algorithm* A *is a number of samples* $m_0 \in \mathbb{N}$ *so that for sample size* $m \geq m_0$ *the learning criteria is satisfied for any distribution and any target function.*

Definition (Sample Complexity [5]). *The* sample complexity, m_A, *of a learning algorithm* A *is the smallest number of samples sufficient for learning; that is,* $m_A = \min\{m | m = m_0\}$.

Definition (Sample Complexity Lower Bound [5]). *A* sample complexity lower bound, m_L, *is a lower bound on the number of samples required to learn* H *regardless of the learning algorithm used; that is,* $m_L = \min_A m_A$.

In the next section, we discuss the details of the learning model used in this thesis, the PAC learning model.

1.3 PAC Learning

The probably approximately correct (PAC) learning model outlines criteria for successfully learning a concept class C using a hypothesis class H given any distribution D on X and any error rate ϵ and confidence $1-\delta$. We will assume that H and C are large, so we analyze H and C in terms of their VC-dimensions, defined below.

Definition (Shatter [4]). *A set of functions H* shatters *a set of points $S \subseteq X$ if H realizes all labelings of points in S.*

Definition (VC-dimension [4]). *The* VC-dimension *of a set H is the size of the largest set that can be fully shattered by H.*

We let d denote the VC-dimension of the hypothesis class H. Before defining PAC learning, we review two notions of error needed to evaluate a hypothesis h - the empirical error and the generalization error.

Definition (Empirical Error [4]). *Let h^* denote the target hypothesis. Let $S \sim D^m$ denote a sample of size m drawn from distribution $D \sim X$. The* empirical error *of hypothesis h is defined as*

$$\hat{err}_D(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{h^*(x_i)\neq h(x_i)}$$

Definition (Generalization Error [4]). *Let h^* denote the target hypothesis. The* generalization error *of hypothesis h is defined as*

$$err_D(h) = \Pr_{S\sim D^m}[h^*(x) \neq h(x)].$$

We now define the PAC learning criteria. There are two settings of PAC learning considered in this thesis. The first setting is the realizable setting, where the target function is in the hypothesis set.

Definition (Realizable PAC Learning [4,5]). *Let H be a hypothesis class of finite VC-dimension d. Then, algorithm A is an* realizable PAC learning algorithm *for C using H if for any target $c^* \in C$, for any distribution $D \sim X$ and any $\epsilon, \delta > 0$, A returns $h \in H$ so that with probability $1 - \delta$,*

$$err_D(h) \leq \epsilon,$$

with sample complexity m_A polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

We note that in the realizable setting, H and C can be different sets. In this thesis, we generally assume that $H = C$, known as the proper PAC learning setting. The second setting of PAC learning is the agnostic, or non-realizable setting, where the target function is not necessarily in the hypothesis set.

Definition (Agnostic, Non-Realizable, PAC Learning [4, 5]). *Let H be a hypothesis class of finite VC-dimension d. Then, algorithm A is an* agnostic, or non-realizable, PAC learning algorithm *using H if for any distribution $D \sim X \times \{0, 1\}$ and any $\epsilon, \delta > 0$, A returns $h \in H$ so that with probability $1 - \delta$,*

$$err_D(h) \leq \min_{h' \in H} err_D(h') + \epsilon,$$

with sample complexity m_A polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

1.3.1 Sample Complexity Bounds of PAC Learning

Upper and lower sample complexity bounds of realizable and non-realizable PAC learning are well known. We rely on these bounds heavily in our algorithm analyses. Let $m_{\epsilon,\delta}$ denote the sample complexity of PAC learning. We first consider a sample complexity upper bound in the realizable setting. A sufficient method for PAC learning is identifying a hypothesis in H that is consistent with the sample $S \sim D^m$. The size of S is outlined in the following result.

Theorem 1 (Upper Bound, Realizable [4–6])**.** *Let H be a hypothesis class of finite VC-dimension d. By [7], for any PAC learning algorithm that finds a consistent hypothesis,*

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

The following upper bound is sometimes better, and due to [8]:

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

In this thesis we default to the second upper bound, $O(\frac{d}{\epsilon}\ln(\frac{1}{\delta}))$, in the above theorem as the sample complexity upper bound of PAC learning. We now consider the sample complexity upper bound of non-realizable (agnostic) PAC learning. Recall that in the non-realizable setting, the target concept may not be in H. Therefore, finding a consistent hypothesis may not be possible. However, it has been shown that the generalization of a consistent hypothesis, an *empirical risk minimizing (ERM)* hypothesis, is sufficient for PAC learning.

Theorem 2 (Upper Bound, Non-realizable [4, 5])**.** *Let H be a hypothesis class of finite VC-dimension d. For any PAC learning algorithm that finds an ERM,*

$$m_{\epsilon,\delta} = O\left(\frac{d}{\epsilon^2}\ln\left(\frac{1}{\delta}\right)\right).$$

Finally, we recall sample complexity lower bounds in both the realizable and non-realizable PAC learning settings.

Lemma 3 (Lower Bound, Realizable [4–6])**.** *Let H be a hypothesis class of finite VC-dimension d. For any PAC learning algorithm, there exists a distribution D and a target function $h^* \in H$ such that for any $\epsilon < \frac{1}{8}$ and $\delta < \frac{1}{100}$,*

$$m_{\epsilon,\delta} = \Omega\left(\frac{d}{\epsilon}\right).$$

Furthermore, if H contains at least three functions, then

$$m_{\epsilon,\delta} = \Omega\left(\frac{1}{\epsilon}\ln\left(\frac{1}{\delta}\right)\right),$$

for all $0 < \epsilon < \frac{3}{4}$ and $0 < \delta < 1$. Together, these results imply that for all $0 < \epsilon \leq \frac{1}{8}$ and $0 < \delta \leq \frac{1}{100}$,

$$m_{\epsilon,\delta} = \Omega\left(\frac{d}{\epsilon} + \frac{1}{\epsilon}\ln\left(\frac{1}{\delta}\right)\right).$$

1.3.2 PAC Learning with Classification Noise

We review PAC learning in the presence of classification noise (CN), introduced in [9]. We will sometimes refer to this setting as *noisy-PAC learning*. We recall the main result in [9], optimized and adapted for hypothesis classes of finite VC-dimension in [10], which prescribes the number of samples that must be drawn from a noisy oracle in order to PAC learn H. Let $m_{\epsilon,\delta,\eta}$ denote the sample complexity of PAC learning with CN setting.

Theorem 4 ([9, 10])**.** *Let H denote a hypothesis class with finite VC-dimension d. Let D be a distribution on X and $\eta < \frac{1}{2}$. Let $EX_\eta(\cdot)$ denote an oracle that returns $(x, h^*(x))$ with probability $1-\eta$ or $(x, \neg h^*(x))$ with probability η. Given any sample S drawn from EX_η, an algorithm A that produces a hypothesis $h \in H$ that minimizes disagreements with S satisfies the PAC criterion, i.e. for any $\epsilon, \delta > 0$ and any distribution D on X,*

$$\Pr_{S \sim D^m}[err_D(h) \geq \epsilon] \leq \delta,$$

with sample complexity

$$m_{\epsilon,\delta,\eta} = O\left(\frac{d}{\epsilon(1-2\eta)^2} \ln\left(\frac{1}{\delta}\right)\right).$$

We now recall the sample complexity lower bound for learning in the presence of random classification noise.

Theorem 5 ([11, 12]). *Let C be a concept class of finite VC-dimension $d \geq 2$. By [12], for all $\epsilon \leq 1/3$, $\delta < 1/120$ and $\eta \geq 29/60$, PAC learning C in the presence of classification noise requires at least*

$$m_{\epsilon,\delta,\eta} = \Omega\left(\frac{1}{\epsilon(1-2\eta)^2}\ln\left(\frac{1}{\delta}\right)\right)$$

samples. By [11], at least

$$m_{\epsilon,\delta,\eta} = \Omega\left(\frac{d}{\epsilon(1-2\eta)^2}\right)$$

samples are required to PAC learning in the presence of classification noise. Together, the lower bound is then,

$$m_{\epsilon,\delta,\eta} = \Omega\left(\frac{d}{\epsilon(1-2\eta)^2} + \frac{1}{\epsilon(1-2\eta)^2}\ln\left(\frac{1}{\delta}\right)\right).$$

1.4 Concentration Inequalities

We now review concentration inequalities used in some of our proofs.

Theorem 6 (Markov's Inequality [4]). *Let X denote a non-negative random variable with $\mathbb{E}[X] < \infty$. Let $k > 0$. Then,*

$$\Pr[X \geq k\mathbb{E}[X]] \leq \frac{\mathbb{E}[X]}{k}.$$

Theorem 7 (Hoeffding's Inequality [4]). *Let $X_1, ..., X_n$ be independent random variables taking values $X_i \in [a_i, b_i]$. Let $X = \sum_{i=1}^{n} X_i$. For any $\epsilon > 0$,*

$$\Pr_{X \sim D^n}[X - \mathbb{E}(X) \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right).$$

Theorem 8 (Multiplicative Chernoff Bounds [5, 13])**.** *Suppose $X_i, ..., X_n$ are independent random variables taking values $X_i \in \{0, 1\}$ and $\Pr(X_i = 1) = p_i$ for all $i \in [1, n]$. Let $X = \sum_{i=1}^{n} X_i$ and $s \in (0, 1)$. Then,*

$$\Pr[X \leq (1 - s)\mathbb{E}(X)] \leq \exp\left(\frac{-s^2\mathbb{E}(X)n}{2}\right)$$

$$\Pr[X \geq (1 + s)\mathbb{E}(X)] \leq \exp\left(\frac{-s^2\mathbb{E}(X)n}{3}\right).$$

If $s \geq 1$,

$$\Pr[X \leq (1 + s)\mathbb{E}(X)] \leq \exp\left(\frac{-s\mathbb{E}(X)n}{3}\right).$$

CHAPTER 2

BIPARTITE NETWORK RESILIENCE

This chapter was previously published as On the Resilience of Bipartite Networks by Shelby Heinecke, Will Perkins, and Lev Reyzin [1]. The code used for computations can be found in Appendix A.

2.1 Introduction

The goal of our work is to study the resilience of bipartite networks to the spread of diseases, viruses, or other contagion. In our case, the bipartite networks will represent an interaction between two types of agents. Examples of such networks include clients and servers or persons and drinking wells. In the former, one may need to connect clients to servers in order to minimize the propagation of computer viruses; in the latter, one may want to direct people to drinking wells as to minimize the spread of infections.

Our main motivation, however, comes from the study of the spread of sexually transmitted diseases in heterosexual contact networks. This problem has been studied in the economics community, with the assumption that each gender has some (possibly asymmetric) partner distribution. An influential paper in economics [14] shows that in a mean-field model of HIV infection, strategic behavior on the part of individuals can lead to two extreme equilibria, one in which all individuals have a moderate number of partners and one in which some individuals

have very few partners and other individuals have very many partners. We study the same problem in the setting of finite networks.

Namely, the model we employ has been used by Blume et al. [15, 16] to study the network resilience problem in uniform-degree graphs, though we note that related models have been recently considered, e.g. [17]. In a variant of the Blume et al. model, vertices represent agents in the network and edges represent pairwise interactions among the agents. Each agent has an independent probability of being initially infected and can further infect neighboring agents with some probability (see Section 2.2 for details).

Moreover, to correspond to the motivation above, we require the interaction graph to be bipartite as well as have minimum degree on one side of the bipartition. The degree restriction is weaker than that of Blume et al. [16] and allows for a larger class of graphs.

We study extremal and computational aspects of the model. Among our results, we show the following:

- We extend the analysis of the susceptibility of networks to infection to the bipartite case, motivated by problems in which there are two types of agents, such as computer terminals/servers, human sexual networks, and maps of shared resources.
- We show that the objective function, the expected fraction of infected individuals in the network, corresponds for specific choices of parameters to the expectation of natural functions under independent edge percolation, a widely studied model in probability and combinatorics.

- We characterize optimal graphs when one side of the bipartition has uniform degree 1 and for higher degree give optimal graphs for extremal choices of parameters. (Theorems 1 and 2).
- We show that the two optimally resilient "extreme" graphs in the $d = 1$ case are not sufficient in the $d = 2$ case (Theorem 3).
- We show that if we instead force a connectivity requirement in lieu of a one-sided degree requirement, we again find that the two obvious "extremes" are not sufficient.
- We show that finding an optimal subnetwork of an arbitrary graph is NP-hard even when the one-sided degree restriction is $d = 1$. (Theorem 4).

2.2 Model

In this work, we are concerned with balanced bipartite graphs on $2n$ nodes. In a balanced bipartite graph $G = (V, E)$, we have $V = L \cup R$, with $|L| = |R| = n$. Our graphs will also have the following asymmetric degree restriction: all vertices in R have degree at least[1] $d > 0$.

On such a graph G, the following infection process occurs. Each node v becomes infected independently at random with probability μ 'by nature'. Then, infected nodes spread their infections independently to adjacent uninfected nodes with probability p. As each new node becomes infected, they have one chance to infect their uninfected neighbors. This is known as the independent cascade model in the literature [18].

---

[1]For the task of finding structures most resilient to the spread of infections, the optimal graphs will have the property that all vertices in R will have degree exactly d.

Given that the above is a random process, we analyze the expected number of infected nodes for a given choice of n, μ, p, and graph G. The goal of our work is to examine which networks among all bipartite graphs of a minimal degree on one side are most resilient to the spread of infections, i.e. which networks have the fewest infected nodes in expectation. We also consider the computational hardness of determining the optimal subnetwork of one-sided minimal degree d of an arbitrary bipartite graph.

Blume et al. [15] study this model with respect to a cost/benefit analysis. They consider strategic vertices who receive utility for each link formed but are penalized if they become infected. They show a gap between the optimal graphs with respect to social welfare and graphs which satisfy conditions for strategic equilibria. In this work we are solely concerned with socially optimal graphs and do not consider strategic behavior.

One way to interpret the model and the quantity we are minimizing is with respect to independent edge percolation. For a fixed graph G on n vertices, let each edge be present independently with probability p. Let $|C(v)|$ denote the size of the (random) connected component containing the vertex v. Then, one can easily calculate that

$$I(G) := 1 - \frac{1}{n}\mathbb{E}\left[\sum_{v \in G}(1-\mu)^{|C(v)|}\right]$$

is exactly the expected fraction of infected nodes in the (μ, p) model.

Independent edge percolation on finite graphs is widely studied in probability and combinatorics. If G is the complete graph on n vertices, the model is the Erdős-Rényi random

graph. Edge percolation on regular lattices is the topic of percolation theory in probability, and edge percolation on more general graphs has also been studied [19–21], but typically in the context of strong conditions (the 'triangle condition,' conditions on expansion) that ensure certain behavior at the phase transition.

One topic in this field that has not been considered in depth is extremal graphs with respect to percolation properties. Network design to minimize the spread of infections is one example of such a problem, but many more can be imagined. In fact, several other quantities can be interpreted with regard to the spread of infections. For example, let the random variable

$$S(G) = \frac{1}{n} \sum_{v} |C(v)|$$

be the average component size of a graph after p-edge percolation. This quantity, known as the susceptibility, is fundamental in the study of random graphs (e.g. [20], [22]). It is not hard to show that the graph in a family of n-vertex graphs that minimizes $\mathbb{E}[S(G)]$ also minimizes the expected number of infected individuals in a single-origin model of infection in which one vertex at random is infected by nature, and then the infection spreads across edges with probability p.

In a different model, that of general thresholds as studied in [16], half-regular bipartite graphs are already extremely rich. It can be shown that for $d = 1$ every possible graph can be optimal under some choice of settings (Proposition 1 in Section 2.5).

2.3 Independent Cascade on Bipartite Graphs

As in the work of Blume et al. [16], we solve the problem of finding the optimal network satisfactorily for the smallest non-trivial degree bound ($d = 1$ for half-regular bipartite graphs, $d = 2$ for regular graphs), and for higher d we exhibit two graphs that can be optimal.

First we characterize the $d = 1$ case, which is the simplest case for this model. We first show that, depending on the settings of μ and p, different graph structures become optimal. Moreover, we can characterize the set of optimal solutions – namely, the network structure that minimizes $I(G)$, the expected fraction of infected nodes, must always be a matching or a star. Finally, we will point out that despite the optimality of one of the two extreme cases, there is non-monotonic behavior with respect to the size of the star.

2.3.1 Half-Regular Graphs with $d = 1$

Theorem 1. *For $d = 1$, all n, and all settings of μ and p, either the perfect matching or an n-star (with $n - 1$ isolated vertices) minimizes $I(G)$.*

Proof. We observe that each feasible graph is a collection of stars with (possibly) some isolated vertices in L. We therefore compute the expected fraction of infected individuals in the union of a k-star and $k - 1$ isolated vertices, call this $\mathbb{E}[I_k]$:

$$\mathbb{E}[I_k] = \frac{L_k + (k-1)L_0 + kR_k}{2k},$$

where L_j is the probability that a vertex of degree j in L is infected, and R_j is the probability that a vertex in R joined to a vertex of degree j is infected. Note that the expected fraction of

infected individuals in a perfect matching is exactly $\mathbb{E}[I_1]$ and the expected fraction in an n-star with $n-1$ isolated vertices in L is $\mathbb{E}[I_n]$. We will show that for $k \in [1, n]$, $\mathbb{E}[I_k]$ is minimized at either $k = 1$ or $k = n$, and since any feasible graph is a union of stars, this shows that either the perfect matching or n-star is optimal.

We calculate

$$L_j = 1 - (1-\mu)(1-\mu p)^j$$

and

$$R_j = \mu + p - \mu p - (1-\mu)^2 p(1-\mu p)^{j-1},$$

giving

$$\begin{aligned}\mathbb{E}[I_k] =& \frac{1-(1-\mu)(1-\mu)^k + (k-1)\mu}{2k} \\ &+ \frac{\mu + p - \mu p - (1-\mu)^2 p(1-\mu p)^{k-1}}{2}.\end{aligned}$$

Now define

$$\begin{aligned}Q(k) &:= \frac{2(\mathbb{E}[I_k] - \mathbb{E}[I_1])}{1-\mu} + 2\mu p - p \\ &= \frac{1-(1-\mu p)^k}{k} - (1-\mu)p(1-\mu p)^{k-1} \\ &= \frac{1-\alpha^k}{k} - \beta\alpha^k,\end{aligned}$$

where we define $\alpha = 1 - \mu p$ and $\beta = \frac{(1-\mu)p}{1-\mu p}$.

We will show that whenever $\frac{dQ}{dk} \geq 0$, $\frac{d^2Q}{dk^2} < 0$, which shows that Q is a unimodal function of k on the interval $[1, n]$ for any n, and in particular takes its minimum at one of its endpoints. Because Q is a linear function of $\mathbb{E}[I_k]$, this shows that $\mathbb{E}[I_k]$ takes its minimum at either $k = 1$ or $k = n$. We can assume $\mu \in (0, 1)$ and $p > 0$, since otherwise all $\mathbb{E}[I_k]$ is equal for all k.

We compute

$$\frac{dQ}{dk} = -\frac{(1-\alpha^k) + k(1+\beta k)\alpha^k \log \alpha}{k^2}$$

and

$$\frac{d^2Q}{dk^2} = \frac{2(1-\alpha^k) + 2k\alpha^k \log(\alpha) - k^2(1+\beta k)\alpha^k \log^2 \alpha}{k^3}$$

and so

$$2k^2 \frac{dQ}{dk} + k^3 \frac{d^2Q}{dk^2} = -\alpha^k k^2 \log \alpha (2\beta + \log \alpha + \beta k \log \alpha).$$

Since $\log \alpha < 0$, this is negative when $2\beta + \log(\alpha) + \beta k \log(\alpha)$ is negative, i.e. when $k > -\frac{2}{\log \alpha} - \frac{1}{\beta}$, and so for such k we have that whenever $\frac{dQ}{dk} \geq 0$, $\frac{d^2Q}{dk^2} < 0$. If $-\frac{2}{\log \alpha} - \frac{1}{\beta} < 1$, then we are done, since we need Q to be unimodal on $[1, n]$.

Otherwise, for $2\beta + \log(\alpha) + \beta k \log(\alpha) \geq 0$, we show directly that $\frac{d^2Q}{dk^2}$ is negative. From (2.3.1), we see that if

$$\begin{aligned} H(k) &:= 2(1-\alpha^k) + 2k\alpha^k \log(\alpha) - k^2(1+\beta k)\alpha^k \log^2 \alpha \\ &< 0, \end{aligned}$$

then $\frac{d^2Q}{dk^2} < 0$. We compute $H(0) = 0$ and

$$\frac{dH}{dk} = -k^2\alpha^k \log^2 \alpha(3\beta + \log(\alpha) + bk\log(\alpha)),$$

which is negative when $k > 0$ and $3\beta + \log(\alpha) + bk\log(\alpha) > 0$, which is true by assumption for this range of k since $\beta > 0$. □

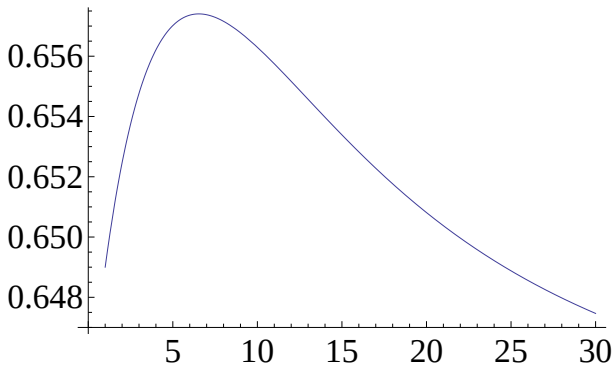

Figure 1: Average infection probability as a function of the degree of a star, for $\mu = 0.55$ and $p = 0.4$.

Note that for n large enough, the matching is better than the star if and only if $\mu < 1/2$. However, there are already surprising effects in the $d = 1$ case – for instance, while a star can be better than a matching, a decomposition into smaller stars can be worse than either. In Figure 1, for the fixed parameters $\mu = .55, p = .4$, we plot the expected fraction of infected vertices in a k-star with $k - 1$ isolated vertices for various values of k.

2.3.2 Half-Regular Graphs with $d \geq 2$

For $d \geq 2$, we first show two possibilities for optimal graphs. We will prove the following proposition by solving appropriate extremal percolation problems:

Theorem 2. *Both a collection of $K_{d,d}$'s and $K_{d,n}$ with $n - d$ isolated vertices can be optimal d-half-regular bipartite graphs. In particular,*

1. *For any p and any $d \geq 1$, for large enough n, there exists μ close enough to 1 so that $K_{d,n}$ with $n - d$ isolated vertices is optimal.*
2. *For any d and large enough n, there is a μ close enough to 0, there exist p's close enough to 0 and to 1 so that a collection of $K_{d,d}$'s is optimal.*

Proof. We prove the two parts separately:

1. If we set $\mu = 1 - n^{-2}$, the RHS in 2.2 becomes

$$1 - n^{-3}\mathbb{E}[X_0(G)] + O(n^{-4}),$$

where $X_0(G)$ is the number of isolated vertices after p-edge percolation (each edge of the graph is deleted independently with probability $1-p$). So for large enough n, minimizing $I(G)$ becomes equivalent to maximizing the expected number of isolated vertices in a graph after p-edge percolation. Since every vertex in R has the same probability of being isolated due to the degree restriction, we wish to maximize the fraction of vertices in L which are isolated. The $K_{d,n}$ configuration has $n-d$ vertices which are isolated with probability 1, and for n large enough the contribution of the remaining d vertices becomes negligible.

2. Set $\mu = n^{-2}$. Then $I(G)$ in 2.2 becomes

$$n^{-3}\mathbb{E}\left[\sum_v |C(v)|\right] + O(n^{-3}),$$

and so minimizing $I(G)$ becomes equivalent to minimizing $\mathbb{E}[S(G)]$ from 2.2. For $p = 1$, we keep all the edges and so we need to minimize

$$\sum_v |C(v)| = \sum_C |C|^2 \leq \sum_{C \in \mathcal{C}_R} |C|^2,$$

where the first sum is over all vertices, the second over all components, and the third over all components containing a vertex in R. Since a collection of $K_{d,d}$'s has no isolated vertices in L, showing that such a graph minimizes $\sum_{C \in \mathcal{C}_R} |C|^2$ suffices. Considering all components containing a vertex in R, we note that each component has at least d vertices from L, and the sum of

the number of vertices from R in all components equals n. Under these conditions, minimizing with Lagrange multipliers gives each component of size $2d$, which is the $K_{d,d}$ configuration.

For $p \to 0$, set $\mu = n^{-3}, p = n^{-2}$. A similar calculation to the above shows that minimizing $I(G)$ in this case is equivalent to minimizing $\sum_C |E(C)|^2$, where the sum is over all connected components and $|E(C)|$ is the number of edges in a component C. Again we can relax the minimization since $K_{d,d}$'s will have no isolated L vertices, and show that a collection of $K_{d,d}$'s minimizes $\sum_{C \in \mathcal{C}_R} |E(C)|^2$. There are at most n/d components in $\mathcal{C}_R$, and the total number of edges is nd. Therefore n/d components of d^2 edges each minimizes $\sum |E(C)|^2$, which completes the proof. □

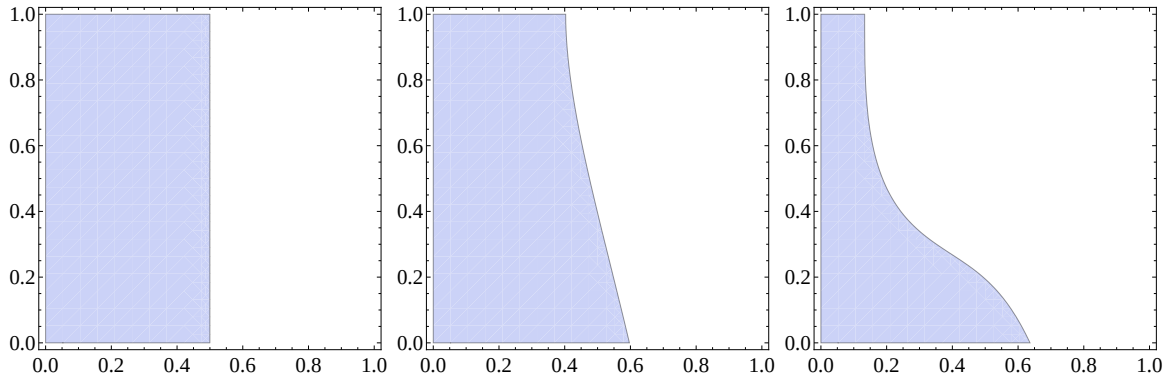

Figure 2: The graphs are for $d = 1$ (left), $d = 2$ (center), and $d = 3$ (right), for $n \to \infty$. The x-axes are values of μ, and the y-axes are values of p. The colored regions are where a $K_{d,d}$ decomposition has a lower average infection rate than $K_{d,n}$ with $n - d$ isolated vertices.

In Figure 2, after solving the cases exactly, we indicate the regions in the parameter space for which $K_{d,d}$ and $K_{d,n}$ are better than one another in the large n limit. It is straightforward to show that as $d \to \infty$, the cut-off for $p = 1$ tends to 0, and for $p \to 0$ the cut-off tends to 1.

Given the results above, we might conjecture that for all $d \geq 1$ and $0 \leq \mu, p \leq 1$, either a $K_{d,d}$ decomposition or $K_{d,n}$ with $n - d$ isolated vertices would be the optimal d-half-regular, balanced bipartite graph on $2n$ vertices. Presently, however, we disprove such a conjecture.

Theorem 3. *For $d = 2$, there exist 2-half-regular graphs on $2n$ nodes that are more resilient than either a $K_{2,2}$ decomposition or a $K_{2,n}$ with $n - 2$ isolated vertices.*

Proof. We take $n = 4$ and consider the 2-half-regular graph on 8 vertices composed of a union of a $K_{3,2}$ and a $K_{1,2}$, with the degree requirement satisfied by the 3 vertices on one side of the partition in the $K_{3,2}$ together with the 1 vertex in the $K_{1,2}$.

For the values $\mu = .302$ and $p = .801$, this graph is more resilient than either two copies of $K_{2,2}$ or the $K_{2,4}$ with two isolated vertices. For these parameter settings, the average infection probabilities for the three graphs are approximately[1] .7197, .7207, and .7199, respectively. This counterexample graph was discovered via a careful computer search, using 2.2, over all half-regular graphs and a chosen set of settings for the parameters μ and p. □

2.3.3 A Note on Connected Graphs

We now briefly turn our attention back to the general model and consider what would happen if we dispose of any degree restriction and instead force the graphs to be connected.

---

[1] We give approximate values to sufficient precision to illustrate the difference in resilience.

We show that with this different restriction, a similar phenomenon occurs as in the $d \geq 2$ case, with optimally resilient graphs again not lying on "extremes." Connected graphs are interesting in models where edges can be used for passing information, as well as disease. There, finding connected resilient graphs preserves the ability to spread information throughout the network while being as resilient as possible to the spread of disease.

If we try to find the optimally resilient connected graph for the μ, p model, we know that an optimal graph is always a tree, since any graph with cycles can have an edge removed without hurting resiliency. It is also interesting to note that, because of this, connectivity naturally gives us a different restriction on bipartite graphs than half-regularity.

A connectivity requirement is somewhat different than the regular or half-regular case. For example, $K_{d,d}$ decompositions, which are sometimes optimal in the half-regular case, are no longer allowed if the graph must be connected. Similarly, for d-regular graphs, Blume et al. [16] show that the optimal 2-regular finite graph on $3n$ nodes is always a triangle decomposition; this is again not connected.

It is then natural to begin by considering the path and the star graphs.[1] In the case of infinite graphs, it is easy to exactly find the expected infection probability of both the infinite star and the infinite path. For the case of the infinite star, we can assume the center is infected (as long as μ, p are constants > 0), and therefore the probability of infection for a leaf is simply

$$\mu + (1 - \mu)p.$$

---

[1]We note that Blume et al. [16] show that the infinite path can be the optimal 2-regular graph.

In the case of the infinite path, 2.2 gives an average infection rate of

$$\sum_{i=1}^{\infty} i(1-(1-\mu)^i)p^i(1-p)^2 = \frac{\mu p - \mu p^3 + \mu^2 p^3}{p(1-p+\mu p)^2}.$$

It is also easy to see that the quantities in 2.3.3 and 2.3.3 are upper bounds for finite stars and paths, respectively, yet either of these can be optimal depending on the settings of μ and p.

The natural question again arises whether a star or a path must always be the most resilient graph, and the answer is, perhaps by now, unsurprisingly, no.

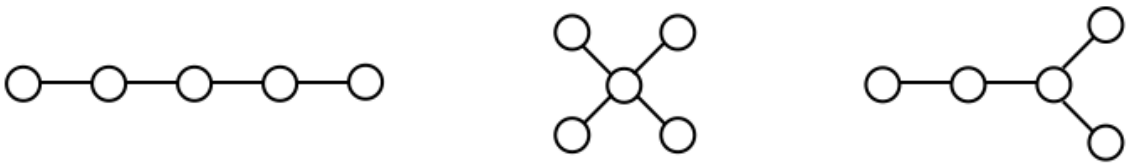

Figure 3: Left to right: the path, star, and fork graphs on 5 nodes. These graphs comprise all the trees on 5 nodes, up to isomorphism. Hence, the most resilient 5-node connected graph must come from this set of graphs, $\forall\ 0 \leq u, p \leq 1$.

For $n = 5$, we compare the 5-path to the star on 5 nodes to a 5-node "fork graph" (Figure 3), and we show that a fork graph can be more resilient than either one of the two "extremes." For the values $\mu = .63$ and $p = .7$, the average infection probabilities for the star, path, and fork graphs are approximately .8906, .8907, and .8905, respectively. Figure 4, computed from

plotting the exact infection rates on the three graphs shows the narrow region where the fork is more resilient than the other two extreme graphs.

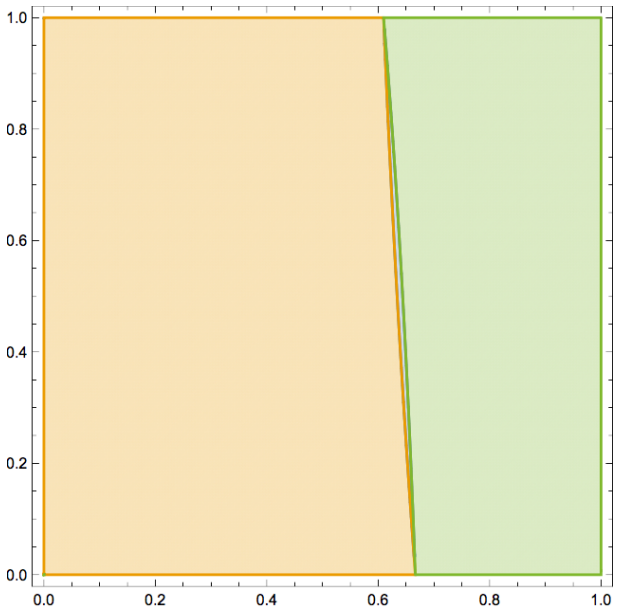

Figure 4: The orange region is where the 5-path is the most resilient 5-node connected graph; the green region is where the star on 5 nodes is the most resilient 5-node connected graph; the small blue region in the center is where the fork is the most resilient 5-node connected graph. μ runs along the horizontal axis and p runs on the vertical axis.

2.4 Optimal Subnetworks of Arbitrary Graphs

In this section we consider the problem of finding an optimal bipartite subnetwork of arbitrary bipartite graphs.

Let $G = (V, E)$ be a bipartite graph with $V = L \cup R$ with degree $\geq d$ for vertices in R. We call the problem of finding a subgraph of G, $G' = (V, E')$, with minimum degree d for vertices in R, as to minimize $I(G')$, the **optimal bipartite subnetwork problem**.

Theorem 4. *For all $d \geq 1$ the optimal bipartite subnetwork problem is NP-hard.*

Proof. For $d = 1$ we reduce from exact set cover. An instance of exact set cover is a family of subsets $\mathcal{F}$ of a ground set U. The goal is to find a subcollection of sets $\mathcal{F}' \subseteq F$ such that each element in U appears in exactly one set in F'. This problem is NP-hard [23]. We will assume w.l.o.g. that all sets in F are the same size, k (we can append new elements to smaller sets).

For our reduction, we construct an instance of the optimal bipartite subnetwork problem as follows. The graph G will contain vertices $L \cup R$, with $R = U$ and $L = \mathcal{F}$. We form an edge $(l, r) \in E$, where $l \in \mathcal{F}$ and $r \in U$ if $r \in l$. Applying 2.3.2, there is a setting of μ and p such that the optimal network will maximize the number of isolated vertices, subject to our constraints.

It is clear that if an exact cover exists, there will be subgraph of G with $|\mathcal{F}| - |U|/k$ isolated vertices – namely the one that uses all edges from the cover. On the other hand, if there is no exact cover, the number of isolated vertices will be $\leq |\mathcal{F}| - |U|/k - 1$.

For $d = 2$, we use Theorem 2, part 2, that there exist settings for μ and p such that a $K_{d,d}$ decomposition is optimal in any graph if it exists. The problem of decomposing a bipartite graph into vertex-disjoint $K_{2,2}$ is NP-hard [24]

For $d \geq 3$ we reduce from the problem of finding a d-clique decomposition of an arbitrary graph, known to be NP-hard [25]. An instance of a d-clique decomposition problem is a graph $G = (V, E)$ and a solution is a partition of G into vertex-disjoint d-cliques.

For this reduction we make a bipartite graph $\hat{G} = (\hat{V}, \hat{E})$ with $\hat{V} = L \cup R$ and $|L| = |R| = |V|$ and $(l_i, r_j) \in \hat{E}$ if $(v_i, v_j) \in E$ or $i = j$. Again, by Theorem 2, part 2, there exist μ and p such that a $K_{d,d}$ decomposition is optimal. Such a decomposition will exist in our case if and only if the original graph G had a d-clique decomposition. □

2.5 General Threshold Model

Blume et al. [16] consider a generalization of the (μ, p) model which we will call the general threshold model. In this model, each vertex is assigned a non-negative integer i which represents the number of infected neighbors required to infect that vertex. If $i = 0$, the vertex is infected 'by nature'. We assign these integers randomly and independently according to some common distribution, where $\Pr[i] =: \mu_i$, and $\sum \mu_i = 1$. The sequence $\{\mu_i\}$ comprises the parameters for the model. The μ, p model is a special case of the cascade model with

$$\mu_i = \begin{cases} \mu & \text{if } i = 0 \\ \\ \mu_i = (1-\mu)p(1-p)^{i-1} & \text{if } i \geq 1. \end{cases}$$

In the case of d regular graphs, [16] shows that for $d = 2$, the optimal graphs[1] are collections of disjoint triangles or the n-cycle. For $d \geq 3$, they show that both collections of disjoint $(d+1)$-cliques and the infinite d-regular tree can be optimal, but there are choices of parameters for which neither is optimal.

For half-regular bipartite graphs, already the case $d = 1$ shows the richness of this model: each k-star can be optimal under some choice of parameters:

Proposition 1. *For every $k \geq 1$ there exists ϵ small enough so that for the choice of parameters $\mu_0 = .6$, $\mu_1 = \epsilon$, and $\mu_{k+1} = .4 - \epsilon$ in the general threshold model, the k-star is the optimal $1-$half-regular bipartite graph.*

Proof. Set the parameters of the general threshold model as above. For $j \leq k$, the expected fraction of infected individuals in a j-star with $j - 1$ isolated vertices is:

$$\begin{aligned}\mathbb{E}[I_j] &= \frac{1}{2j}\left[.6 \cdot 2j + \epsilon(1 - .4^j) + .6\epsilon j + O(j\epsilon^2)\right] \\ &= .6 + .3\epsilon + \frac{1 - .4^j}{2j}\epsilon + O(\epsilon^2)\end{aligned}$$

---

[1]Their choice of objective function is slightly different: they minimize the maximum probability of infection over all vertices.

The function $\frac{1-.4^j}{2j}$ is a strictly decreasing function of j, so for small enough ϵ the k-star is better than any j-star with $j < k$. And for $j > k$,

$$\mathbb{E}[I_j] \geq .6 + .3\epsilon + \frac{1-.4^j}{2j}\epsilon + \frac{(.4-\epsilon)q_{j,k+1}}{2j} + \frac{\epsilon(.4-\epsilon)}{2j}\sum_{i=k+1}^{j-1}(j-1)p_{j,i}$$

where $q_{j,k+1} = \Pr[\text{Bin}(j,.6) \geq k+1]$ and $p_{j,i} = \Pr[\text{Bin}(j,.6) = i]$. For $j \leq 2k$, and ϵ sufficiently small, $\frac{(.4-\epsilon)q_{j,k+1}}{2j} > \frac{1-.4^j}{2j}\epsilon$ and so $\mathbb{E}[I_j] > \mathbb{E}[I_k]$. For $j > 2k$, the term $\frac{\epsilon(.4-\epsilon)}{2j}\sum_{i=k+1}^{j-1}(j-1)p_{j,i}$ is bounded below by ϵ times a constant independent of j, and so the k-star is optimal. □

CHAPTER 3

CROWDSOURCED PAC LEARNING IN THE PRESENCE OF CLASSIFICATION NOISE

This chapter was previously published as Crowdsourced PAC Learning under Classification Noise by Shelby Heinecke and Lev Reyzin [3].

3.1 Introduction and Previous Work

3.1.1 Overview

In this paper, we study the problem of learning a classifier from data labeled by a crowd of workers. In our model, we make the assumption that each worker has his or her own error rate, independent of the data. In this framework, we give a flexible three-step algorithm that achieves the PAC learning criterion. First, a subset of data points is chosen from X, and sufficiently many workers are asked to label each point, so that with high probability, majority votes on each point are correct. This gives a "ground truth" set of points on which workers can be evaluated, so that in the second step, we can estimate their individual error rates and identify good workers – this can be done in many ways, for example by running pure-exploration bandit algorithms. In the final step, the workers selected in the previous step are assigned to label sufficiently many new points so that a PAC-classifier can be trained efficiently. While each part of our approach comes from known results, combining all these steps into a streamlined procedure is, to our knowledge, new. We also illustrate the flexibility of our approach herein.

Instead of relying on random workers to produce labels, the goal of our approach is to quickly identify good workers and assign the main labeling task to them. Our algorithms work especially well when there are a few expert workers in a large crowd, and when they are difficult to pre-screen. Such scenarios can often occur when specialized knowledge is needed, e.g. in the case of using crowdsourced labels to training a classifier to identify cat breeds, where most people presumably don't know anything about cats, but a few people in any large crowd will be adept at it.

3.1.2 Previous Work

3.1.2.0.1 Classification Noise.

We assume that workers in the crowd are imperfect. In particular, each worker w_i has an individual, hidden noise rate $0 \leq \eta_i < 1/2$ so that each data point has an independent and equal chance of being mislabeled, conditioned on the worker. We build our algorithm and analysis around this noise model but show that our analysis can be adapted to handle the case where the noise rates are conditioned on class membership. These noise models have been extensively studied in crowdsourcing literature [26–31] and are usually attributed to Dawid and Skene [32].

The PAC learning model was extended by Angluin and Laird [9] to capture a simple notion of noise, which they termed "classification noise." In their extension, labels of samples are flipped independently with probability $0 \leq \eta < 1/2$ by the noisy oracle, and the learner's runtime and sample complexity must also have a polynomial dependence on $\frac{1}{1-2\eta}$. For part of our work, we will adapt the results of Angluin and Laird [9] to our noise setting. Note that our noise setting is similar in that a label of a data point is flipped independently with probability η_i from worker

w_i; in other words, each worker functions as a noisy oracle in our setting. Since our noise model is a generalization, we refer to our noise model as classification noise throughout this paper.

3.1.2.0.2 Majority Voting in Crowdsourcing.

Since worker skill can be unknown and varying in crowdsourcing, entities posting data points to be labeled on crowdsourcing platforms may require that each data point be labeled by multiple workers. Majority voting is the most obvious method for aggregating the labels from multiple workers. Li, Yu, and Zhou [29] establish error rate bounds of generalized hyperplane rules of which majority voting is a special case. While they assume the same model of classification noise as our work, their analysis is limited to establishing error bounds of these hyperplane aggregation rules and not on PAC learning. Wang and Zhou [30] establish error bounds for majority voting under different assumptions, but they also do not focus on PAC learning. Sheng et al. [33] explore various strategies of utilizing multiple noisy labels based on majority voting and pairing, but they do not focus on PAC learning. Awasthi et al. [34], who focus on PAC learning from crowdsourced labels as we do, note that majority voting is not ideal because the number of worker labels needed to produce an accurate majority vote with probability $1 - \delta$ scales with the size of the data set. We arrive to this same conclusion with our noise model, but we find it beneficial to still use majority voting on a small subset of the unlabeled data set to establish a "high probability" ground-truth training set which helps to eliminate the need for queries to an expert oracle as their algorithm requires.

3.1.2.0.3 PAC Learning in Crowdsourcing.

Feng et al. [27], in very recent work, develop PAC-style bounds for the cost complexity of learning an aggregation function that fits a crowd of workers with varying reliabilities. They focus on using PAC learning to train an aggregation function for the workers' labels; we, however, focus on using PAC learning to train a classifier that generalizes from worker labels. Wang and Zhou [35] also develop PAC-style bounds for the cost complexity of learning a classifier in a similar crowdsourcing setting but their algorithm is similar to our baseline approach. Concurrently, Zhang and Conitzer [36] develop a PAC learning framework for aggregating agents' judgments in a similar setting as ours. However, they focus on recovering the target classifier exactly and employ methods similar to our baseline approach with additional assumptions. Awasthi et al. [34] develop PAC learning algorithms in the crowdsourcing setting that generalize from worker labels but their assumptions on the crowd differ from ours. On the one hand, they assume nothing about the workers' label distribution (this is the agnostic learning setting), but on the other hand they assume some fraction α of the crowd are perfect performing workers. While this assumption is reasonable in some settings (for example, if the crowd is curated), there may exist settings where this assumption would not hold since even the best performing workers are capable of making a mistake. Thus, we instead assume that each worker has a hidden error rate η_i. Second, in the case that the fraction of perfect performing workers is less than $1/2$, their algorithm requires queries to an expert oracle. Our algorithm, however, does not require any expert oracle queries.

3.1.2.0.4 Multiarmed Bandits for Crowdsourcing.

There is substantial progress in multiarmed bandit (MAB) literature regarding identifying the best arms in the vanilla MAB setting [28,31,37–41]. In our work, each arm will represent a worker and we build upon these previous results to train a classifier. In this work, we restrict our attention to the fixed confidence setting of best arm vanilla MAB - we want to identify the best arms with confidence $1 - \delta$.

More recently, MAB have been adapted to crowdsourcing settings [26, 28, 31, 41–43]. These previous works use bandit techniques to strategically assign tasks to workers under assumptions that are realistic to crowdsourcing including limited budgets, limited worker availability, and limited worker loads. Our algorithm builds upon these works to ultimately output a trained classifier. In particular, [26, 31] suggest using their MAB top-K arm algorithms to identify good workers, but they assume there exists a set of accurately labeled points from which to learn (ground truth set). Similarly, Liu and Liu [42] suggest using many bandit algorithms with the same limitation. Our algorithm does not require a ground truth set of data because in practice, ground truth sets may not be available or can be expensive to obtain. Although some previous algorithms [28, 42] do not assume a ground truth set of points and instead estimate the correct label for points online, these algorithms describe an optimal selection policy for assigning tasks rather than training a classifier like ours. Other work [41, 43] likewise focuses on a task assignment policy rather than training a classifier. Other more applied works also consider similar exploration/exploitation trade-offs, e.g. [44].

3.2 Model and Preliminaries

Given a hypothesis class $\mathcal{C}$ of finite VC dimension d and parameters $\epsilon > 0$ and $\delta > 0$, we want to PAC-learn $\mathcal{C}$ using data points with labels gathered from workers in a crowd. Let $W = \{w_i \mid i \in [1, n]\}$ denote the set of all workers, $|W| = n$. Each worker w_i has an individual noise rate $0 \leq \eta_i \leq 1/2$ and will correctly label any given example with probability $1 - \eta_i$. The noise is assumed to be persistent, so a worker asked to label the same example a second time will deterministically produce the same label again. In particular, for any target function $c \in \mathcal{C}$, for any $x \in X$. The worker w_i acts as follows: for all x, $\Pr[w_i(x) \neq c(x)] = \eta_i$.

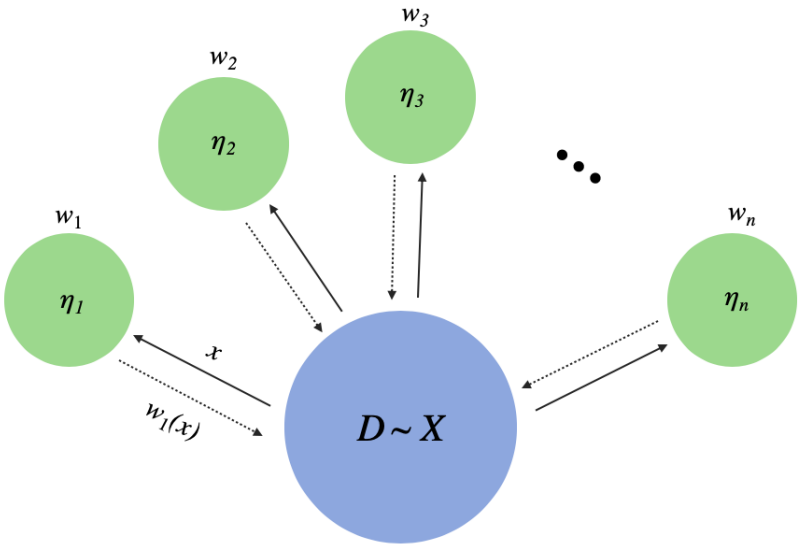

Figure 5: Crowdsourced PAC Setting

Similar to Amazon Mechanical Turk, we define a task as a single data point that needs a label. The goal, then, is to PAC-learn $\mathcal{C}$ while minimizing the number of tasks labeled by workers; in other words, we want to minimize the number of times we query the crowd. This modeling requirement is due to the fact that in most realistic settings, workers are paid per task, and a natural goal is to train a good classifier while expending as little as possible. Also note that this model corresponds to PAC learning from data that has been labeled by workers, each of which is a classifier operating under classification noise [9].

We will ultimately derive upper bounds on the number of tasks labeled by workers to PAC-learn $\mathcal{C}$. We now define several parameters that come into play throughout this paper and in our final bounds. We define

$$\bar{\eta}_W = \frac{1}{|W|} \sum_{\{j|w_j \in W\}} \eta_j$$

denotes the average error rate of workers in W. Let $\bar{\eta}^*_{K,W}$ denote the average error rate of the best K workers in W. As a special case, $\bar{\eta}^*_{1,W}$ denotes the error rate of the single best worker in W.

To obtain reliable labels, our algorithm will identify approximately good workers. Let $0 \leq \Delta \leq 1/2$. We define a Δ-optimal worker and Δ-optimal set of K workers.

Definition (Δ-Optimal Worker [37])**.** *A worker $w_i \in W$ is said to be Δ-optimal if $\eta_i \leq \bar{\eta}^*_{1,W} + \Delta$.*

Definition (Δ-Optimal Set of K Workers [31])**.** *Let $S \subseteq W$ and $|S| = K$. The set of workers S is Δ-optimal if $\bar{\eta}_S \leq \bar{\eta}^*_{K,W} + \Delta$.*

3.2.1 Baseline Approaches

We now describe two baseline approaches and their corresponding task complexities. For the first baseline approach, we plug $\bar{\eta}_W$ into the classification noise bound of Angluin and Laird [9] (see Theorem 4 for details) to get an algorithm that solicits

$$\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2\bar{\eta}_W)^2}\right)$$

labels in total, from worker pool W of N workers, where workers are selected at random to label the points.

Another baseline approach is to obtain a large perfectly labeled set of data points with high confidence via majority voting and use the proper noiseless PAC bound, which requires $m = \mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon}\right)$ examples. By Theorem 10 (which appears in a later section), for each of m datapoints we need a majority vote of $\mathcal{O}\left(\frac{\log(m/\delta)}{(1-2\bar{\eta}_W)^2}\right)$ workers to get a perfectly labeled set with high probability. Combining these two bounds gives a total sample complexity of

$$\tilde{\mathcal{O}}\left(\frac{d\log(1/\delta)}{\epsilon(1-2\bar{\eta}_W)^2}\right),$$

which is actually slightly worse (with respect to suppressed polylogarithmic terms) than the bound in 3.2.1.

Since we want to learn to arbitrarily small errors ϵ, we do not want the d/ϵ dependence to be multiplied by the factor of $\frac{1}{(1-2\bar{\eta}_W)^2}$, which could be large for $\bar{\eta}_W$ close to $1/2$. This is the dependence this paper aims to avoid.

3.3 Crowdsourced Learning Algorithm

Our algorithm proceeds in three parts. First, we choose a small, randomly chosen set of data points to have labeled by multiple workers. This allows us to know the true labels of these data with high confidence using majority voting. Second, we use these labeled data points to identify the approximately best workers. Third, we use these workers to label additional data points from which to train a classifier.

Algorithm 1: Crowdsourcing PAC Algorithm (Informal)

Input: n workers, unlabeled data points X

Output: classifier $h \in \mathcal{C}$

1 take a majority vote with workers on small subset of unlabeled tasks yielding a set of accurately labeled tasks with high confidence

2 using the ground-truth data from Step 1, identify the approximate top worker(s) (e.g. using MAB algorithms [26, 31, 37, 38])

3 assign tasks at random among worker(s) identified in Step 2 to perform noisy-PAC learning [9], returning hypothesis $h \in \mathcal{C}$ that is an empirical risk minimizer (ERM) with the labels of the approximate top worker(s)

We now proceed to analyze each part of the algorithm separately.

3.3.1 Majority Voting by Workers with Classification Noise

Our algorithm begins by getting a set of points for which the labels need to be known, thereby creating a "ground truth" set on which the workers' error rates can be tested. This is done by a majority vote of the labels of randomly selected workers. For this we need the following lemma, which is a simple consequence of the Hoeffding inequality (it is also proved in a more general setting in Li, Yu, and Zhou [29]).

Lemma 9. *Let $\mathcal{L}(x) = \{w_i(x) \mid w_i \in W\}$ be the labels from workers in W, for some $x \in X$. Suppose majority voting over the n labels in $\mathcal{L}(x)$ is applied and the winning label is the final label corresponding to x. Then, the error of the majority vote can be upper bounded as follows:*

$$\Pr[\mathrm{MAJ}(\mathcal{L}(x)) \neq c(x)] \leq 2e^{-n(1-2\bar{\eta}_W)^2/2}.$$

As a consequence, we can derive the following theorem.

Theorem 10. *Let $Y \subseteq X$ and $|Y| = T$. Suppose we want to get true labels for data points in Y with probability $1-\delta$ using majority voting with the crowd of workers W. Then for each $y \in Y$, it is sufficient to solicit*

$$\mathcal{O}\left(\frac{\log(T/\delta)}{(1-2\bar{\eta}_W)^2}\right)$$

labels from the crowd.

Proof. Follows from Lemma 1 and the union bound. □

Since Theorem 10 scales poorly, it is not prudent to rely solely on majority voting for gathering a labeled data set. However, we find that using majority voting on a small enough data set can be useful because it can eliminate the assumption of a ground truth set and instead generate an ground-truth set with high probability. In this way, we also eliminate the need for expert oracle queries used in Awasthi et al. [34]. [1]

3.3.2 Identifying Top Performing Workers

Using the ground-truth training labeled data set acquired from the previous section, we now identify one approximately good worker. We also examine the case where we want to identify a set of approximately good workers.

3.3.2.1 Identifying One Δ-Optimal Worker.

The naive approach to identifying a Δ-optimal worker with probability $1-\delta$ is to sample each arm $\mathcal{O}\left(\frac{1}{\Delta^2}\log(n/\delta)\right)$ times and return the arm with the largest empirical average.

Theorem 11. *Identifying a Δ-optimal worker with probability at least $1-\delta$ can be done in $\mathcal{O}\left(\frac{n}{\Delta^2}\log(n/\delta)\right)$ arm trials.*

Since each ground-truth data point can be used to test all n workers, we need $\mathcal{O}\left(\frac{1}{\Delta^2}\log(n/\delta)\right)$ ground-truth data points to find a Δ-optimal worker. If we introduce the assumption that there

---

[1] This of course relies on access to a sufficiently large crowd, and hence we assume that $N = \tilde{\Omega}\left(\frac{\log(T/\delta)}{(1-2\bar{\eta}_W)^2}\right)$, so that at this stage each worker will be assigned at most one labeling task, to get the label of each point. Additionally, notice that the number of labels in Theorem 10 scales as a function of the number of data points T for which we want labels, as noted by Awasthi et al. [34]. Our bound in Theorem 10 is also a function of $\bar{\eta}_W$ because of our classification noise model, which differs from Awasthi et al. [34].

is at least one perfect performing worker in the crowd, then the number of arm trials to identify a Δ-optimal worker decreases.

Lemma 12. *If there is at least one worker in the crowd who performs perfectly, then* $\mathcal{O}(\frac{n}{\Delta}\log(n/\delta))$ *samples are sufficient to identify a* Δ*-optimal worker with probability* $1-\delta$*.*

Proof. The probability that a worker who was observed to be perfect on t examples has error $\geq \Delta$ is bounded by $(1-\Delta)^t \leq e^{-\Delta t}$. For the union bound, we need to set this to $\leq \delta/k$, which yields the result. □

Corollary 13. *Acquiring accurate labels for data points with probability* $1-\delta$ *so that a* Δ*-optimal worker can be identified requires at most* $\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta(1-2\bar{\eta}_W)^2}\right)$ *worker labels if there is at least one perfect performing worker in the crowd and* $\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta^2(1-2\bar{\eta}_W)^2}\right)$ *otherwise.*

Proof. The number of arm trials to identify a Δ-optimal worker is given in Theorem 11 and Lemma 12. We sample all arms uniformly, so $\mathcal{O}\left(\frac{1}{\Delta^2}\log(n/\delta)\right)$ and $\mathcal{O}\left(\frac{1}{\Delta}\log(n/\delta)\right)$ accurately labeled points are needed in order to compute the reward for each arm trial, respectively. We acquire an accurately labeled point with high confidence as in Theorem 10, where we set $T = \mathcal{O}\left(\frac{1}{\Delta^2}\log(n/\delta)\right)$ and $T = \mathcal{O}\left(\frac{1}{\Delta}\log(n/\delta)\right)$. We then multiply by T to get the total number of worker labels needed to acquire an ground-truth set of size T. □

The problem of identifying the best workers can also be solved with sophisticated methods that employ pure-exploration stochastic multi-armed bandit algorithms [26, 31]; for example, `OptMAI` (see Theorem 14) improves the dependence on n in logarithm, even in the case of finding the approximately-best worker.

In our crowdsourcing setting, each worker is an arm in the bandit setting with mean reward $1 - \eta_i$. When we select a worker/arm, the reward is 1 if the worker's label is correct and 0 otherwise. In order to compute rewards, many bandit algorithms require a ground-truth set of points [26, 31, 37, 38]. Instead, we use the set we gathered from the majority voting step as a proxy for a ground-truth set. Thus, we are able to make use of many MAB algorithms, but for now we focus on vanilla MAB.

3.3.2.2 Identifying the Top K Workers.

Let $K \leq n$. The following sample complexity bound of identifying a set of the approximate top K workers is known.

Theorem 14 (Zhou, Chen, and Li 2014)**.** *For $K \leq \frac{n}{2}$, `OptMAI`(n, K, q) identifies a Δ-optimal set of K arms with probability $1 - \delta$ using*

$$q = \mathcal{O}\left(\frac{n}{\Delta^2}\left(1 + \frac{\log(1/\delta)}{K}\right)\right)$$

arm trials.[1]

An upper bound on the number of trials per arm is given, as well.

---

[1] *For $K \geq n/2$, `OptMAI`(n,K, q) identifies a Δ-optimal set of K arms with probability $1 - \delta$ using*

$$q = \mathcal{O}\left(\left(\frac{(n-K)n}{K\Delta^2}\right)\left(\frac{(n-K)}{K} + \frac{\log(1/\delta)}{K}\right)\right)$$

arm trials.

Theorem 15 (Zhou, Chen, and Li 2014)**.** *In* `OptMAI`*(n, K, q), each arm is sampled at most*

$$s = \mathcal{O}\left(\frac{q}{n^{.3}}\right)$$

times, where q is set according to 14.

Now, we will use `OptMAI` from Zhou, Chen, and Li [31] to efficiently learn a set of the approximate top K workers in our crowdsourcing model. A worker labeling a datapoint will function as an arm pull. Hence, a number of correctly labeled datapoints as in 15 will be sufficient to implement this strategy.

Corollary 16. *Acquiring accurate labels for s data points (as per 15) with probability $1-\delta$ so that a Δ-optimal set of K workers can be identified requires at most*

$$\tilde{\mathcal{O}}\left(\frac{n^{.7}\log(1/\delta)\left(1+\frac{\log(1/\delta)}{K}\right)}{\Delta^2(1-2\bar{\eta}_W)^2}\right)$$

total tasks assigned to workers to label points.

Proof. The number of arm trials to identify a Δ-optimal set of K workers is given by q in Theorem 14. Each arm is sampled at most $\mathcal{O}\left(\frac{q}{n^{.3}}\right)$ times (Theorem 15), thus we need this many accurately labeled points in order to compute the reward for each arm trial. We acquire an accurately labeled point with high confidence as in Theorem 10, where we set $T = \mathcal{O}\left(\frac{q}{n^{.3}}\right)$. We then multiply by T to get the total number of worker labels needed to acquire a ground-truth set of size T. □

It is also clear that for various extensions and variants of our problem, we can also use more sophisticated bandit algorithms. For example, if different sets of workers are available during different rounds, we can use sleeping bandits [45], etc. The variety of known bandit algorithms working under various assumptions further illustrates the flexibility of our modular approach.

3.3.3 PAC Learning under Label Noise

Now that the algorithm has identified good workers, we use those workers to label more tasks needed to PAC learn the concept class $\mathcal{C}$. In this step, each task consists of labeling a distinct data point; in other words, each data point is labeled only once by one of the good workers we identified in the previous step. To perform the PAC learning we use the algorithm from [9] in which the learner queries a noisy oracle sufficiently many times and returns the hypothesis $h \in \mathcal{C}$ that has the minimal number of disagreements with the results from the noisy oracle. We assume that finding this hypothesis can be done efficiently.

We adapt Theorem 4 to the two types of approximately good workers identified in the previous section so that either one Δ-optimal worker functions as the noisy oracle or the Δ-optimal set of K workers sampled i.i.d. function as the noisy oracle. In the former case, the oracle noise rate η becomes $\bar{\eta}^*_{1,W} + \Delta$, so by Theorem 4 we assign to the Δ-optimal worker at most

$$\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2(\bar{\eta}^*_{1,W}+\Delta))^2}\right)$$

additional points to label. In the latter case, the oracle noise rate becomes $\bar{\eta}^*_{K,W} + \Delta$, so we assign the Δ-optimal set of K workers

$$\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2(\bar{\eta}^*_{K,W}+\Delta))^2}\right)$$

points to label.

3.3.4 Total Task Complexity

We combine the task bounds established for majority voting, identifying good workers, and PAC-learning with good workers to derive the total task complexity of our algorithm. To satisfy the PAC criterion (Theorem 3.3.3), we set the failure rate for each part of our algorithm to be at most $\delta/3$ (so that the total failure rate is bounded by δ). We then add the three task bounds. Notice that the bounds in 3.3.3 and 3.3.3 adapted from Theorem 4 and the arm trial bounds from Section 3.3.2 are a function of Δ. We parameterize Δ as a function of either best worker's error rate or the average error rate of the best set of K workers. For Theorems 17 and 18, which follow, we set

$$\Delta = \frac{1/2 - \bar{\eta}^*_{1,W}}{2}.$$

The following theorem gives an upper bound on the number of tasks required by our algorithm in order to PAC-learn $\mathcal{C}$.

Theorem 17. *Let $\epsilon, \delta > 0$. Suppose that in Step 2 of the algorithm, we identify one approximately good worker. Then*

$$\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{(1-2\bar{\eta}^*_{1,W})^2(1-2\bar{\eta}_W)^2} + \frac{\left(n+\frac{d}{\epsilon}\right)\log(1/\delta)}{(1-2\bar{\eta}^*_{1,W})^2}\right)$$

tasks can be labeled by workers in order to efficiently PAC learn $\mathcal{C}$.

Proof. We sum the task complexity from each step in the algorithm. We first sample the crowd $\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta^2(1-2\bar{\eta}_W)^2}\right)$ times (Corollary 13) in order to gather a ground truth set with probability $1-\delta$. Using the ground truth set as the training set, we sample the crowd $\mathcal{O}\left(\frac{n}{\Delta^2}\log(n/\delta)\right)$ times (Theorem 11) in order to identify an approximately good worker. We then use the approximately good worker to label $\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2(\bar{\eta}^*_{1,W}+\Delta))^2}\right)$ points (bound in 3.3.3). Summing these components gives

$$\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta^2(1-2\bar{\eta}_W)^2}\right) + \mathcal{O}\left(\frac{n}{\Delta^2}\log(n/\delta)\right) + \mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2(\bar{\eta}^*_{1,W}+\Delta))^2}\right).$$

Setting $\Delta = \frac{1/2-\bar{\eta}^*_{1,W}}{2}$ and simplifying yields the task complexity. □

Recall that from Lemma 12, if we assume there is one perfect worker in the crowd, the task complexity improves. We see the improvement in the overall task complexity below. In this case, since $\bar{\eta}^*_{1,W} = 0$, we set $\Delta = \frac{1}{4}$.

Theorem 18. *Let $\epsilon, \delta > 0$. Suppose that in Step 2 of the algorithm, we identify one approximately good worker and we assume there exists at least one perfect performing worker in the crowd. Then*

$$\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{(1-2\bar{\eta}_W)^2} + \left(n + \frac{d}{\epsilon}\right)\log(1/\delta)\right)$$

tasks can be labeled by workers in order to efficiently PAC learn $\mathcal{C}$.

Proof. We sum the task complexity from each step in the algorithm. We first sample the crowd $\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta(1-2\bar{\eta}_W)^2}\right)$ times (Corollary 13) in order to gather a ground truth set with probability $1-\delta$. Using the ground truth set as the training set, we sample the crowd $\mathcal{O}\left(\frac{n}{\Delta}\log(n/\delta)\right)$ times (Lemma 12) in order to identify an approximately good worker. We then use this approximately good worker to label $\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2(\bar{\eta}^*_{1,W}+\Delta))^2}\right)$ points (3.3.3). Since there is a perfect worker in the crowd, $\bar{\eta}^*_{1,W} = 0$. Summing these components gives

$$\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta(1-2\bar{\eta}_W)^2} + \frac{n}{\Delta}\log(n/\delta) + \frac{d\log(1/\delta)}{\epsilon(1-2\Delta)^2}\right).$$

Setting $\Delta = 1/4$ and simplifying yields the task complexity. □

As discussed in Section 3.3.2, an alternative to identifying one approximately good worker is to identify K approximately good workers to limit the burden of tasks for workers. The maximum number of tasks a single worker must complete is referred to as the load [34]. In the case of one approximately good worker, that worker must label all the tasks prescribed by the bound in 3.3.3. In the case of K approximately good workers, the workers can evenly split the tasks prescribed by the bound in 3.3.3, reducing the load. If load is a priority in a particular

crowdsourcing setting, then we have the following task upper bound for our algorithm. To derive this bound, we set

$$\Delta = \frac{1/2 - \bar{\eta}^*_{K,W}}{2}.$$

Theorem 19. *Let $\epsilon, \delta > 0$. Let K denote the number of workers identified in Step 2 of the algorithm and assume $K \leq \frac{n}{2}$. Then,*

$$\tilde{\mathcal{O}}\left(\frac{n^{\cdot 7}\log(1/\delta)(1+\frac{1}{K}\log(1/\delta))}{(1-2\bar{\eta}^*_{K,W})^2(1-2\bar{\eta}_W)^2}\right) + \mathcal{O}\left(\frac{(\frac{n}{K}+\frac{d}{\epsilon})\log(1/\delta)+n}{(1-2\bar{\eta}^*_{K,W})^2}\right)$$
$$\subset \tilde{\mathcal{O}}\left(\frac{n\log^2(1/\delta)}{(1-2\bar{\eta}^*_{K,W})^2(1-2\bar{\eta}_W)^2} + \frac{d\log(1/\delta)}{\epsilon(1-2\bar{\eta}^*_{K,W})^2}\right).$$

tasks can be labeled by workers in order to efficiently PAC learn $\mathcal{C}$.

Proof. Again, we sum the task complexity from each step in the algorithm. We first sample the crowd $\tilde{\mathcal{O}}\left(\frac{n^{\cdot 7}\log(1/\delta)(1+\frac{\log(1/\delta)}{K})}{\Delta^2(1-2\bar{\eta}_W)^2}\right)$ times (Corollary 16) in order to gather a ground truth set with probability $1-\delta$. Using the ground truth set as the training set, we sample the crowd q times in order to identify a set of K approximately good workers (Theorem 14). We then use these approximately good workers to label $\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2(\bar{\eta}^*_{K,W}+\Delta))^2}\right)$ points (bound in 3.3.3). Summing these components gives

$$\tilde{\mathcal{O}}\left(\frac{n^{\cdot 7}\log(1/\delta)(1+\frac{\log(1/\delta)}{K})}{\Delta^2(1-2\bar{\eta}_W)^2}\right) + \mathcal{O}\left(\frac{n}{\Delta^2}\left(1+\frac{\log(1/\delta)}{K}\right)\right) + \mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2(\bar{\eta}^*_{K,W}+\Delta))^2}\right).$$

Setting $\Delta = \frac{1/2-\bar{\eta}^*_{K,W}}{2}$ and simplifying yields the task complexity. □

3.3.5 Comparison to Baseline and to Other Work

In the bounds established above, the term $\frac{1}{(1-2\bar{\eta}_W)^2}$ is not multiplied the d/ϵ term, which is the improvement over the baseline described in Section 3.2. In particular, in Theorem 17, the d/ϵ term is multiplied by a factor of

$$\frac{1}{(1-2\bar{\eta}^*_{1,W})^2}$$

which is function of the error rate of the best worker $\bar{\eta}^*_{1,W}$ in W instead of the average of all workers in W, $\bar{\eta}_W$, as in the baseline. Similarly, in Theorem 19, the d/ϵ term is multiplied by

$$\frac{1}{(1-2\bar{\eta}^*_{K,W})^2}$$

which is function of the error rate of the best K workers in W, $\bar{\eta}^*_{K,W}$, instead of $\bar{\eta}_W$. Theorem 18 shows further improvement from the baseline as the d/ϵ is multiplied only by a factor of $\log(1/\delta)$. Note that in all three Theorems, the task complexity can get arbitrarily bad as any of the crowd parameters approaches random guessing, i.e. as $\bar{\eta}^*_{1,W}$, $\bar{\eta}_W$, or $\bar{\eta}^*_{K,W}$ approach $1/2$.

Unlike the baseline, there are additional terms in each of the bounds above that are not multiplied by d/ϵ. While these terms indeed add to the task complexity, as ϵ becomes arbitrarily small they become negligible, thus, the term multiplied by d/ϵ is most important.

We now discuss how our results compare to the work of Awasthi et al. [34]. Recall that Awasthi et al. [34] assume that a fraction α of workers are perfect performers with no assumptions on the rest of the crowd. Like Awasthi et al. [34], our algorithm is a PAC learning algorithm but ours does not rely on or require an assumption of perfect workers in the crowd. Instead,

our algorithm assumes everyone has an individual noise rate. When we do consider perfect workers, we find that even just one perfect worker in the crowd improves our task complexity bound. When the fraction of perfect workers is below 1/2, the algorithm in Awasthi et al. [34] requires "golden queries", queries to an expert oracle. Note that none of our PAC bounds are dependent on access to an expert oracle.

3.4 Variants and Extensions

We now demonstrate a few ways in which our model and algorithm can be easily adapted to fit different crowdsourcing settings.

3.4.1 Asymmetric Classification Noise

In some settings, workers may perform differently depending on the true label of the data point. This assymetric noise model is attributed to Dawid and Skene [32]. For simplicity, we assume the binary classification setting with labels $\{-1, +1\}$. For worker $w_i \in W$, let η_i^+ and η_i^- denote the error rates of positive and negative instances; that is, for each $x \in X$,

$$\eta_i^+ = \Pr[w_i(x) \neq c(x) \mid c(x) = 1]$$

and

$$\eta_i^+ = \Pr[w_i(x) \neq c(x) \mid c(x) = -1].$$

Let $\bar{\eta}_W^+$ and $\bar{\eta}_W^-$ denote the average one-sided error rates among all workers in W. Also let

$$\hat{\eta}_i = \eta_i^+ \Pr[c(x) = 1] + \eta_i^- \Pr[c(x) = -1]$$

and let $i^* = \text{argmin}_i \hat{\eta}_i$.

We now show that our algorithm can be easily adapted to the setting of asymmetric classification noise. We first derive an analogue of Theorem 10 which is also a result of Hoeffding and union bounds.

Theorem 20. *Let $Y \subseteq X$ where $|Y| = T$. Suppose we want to get true labels for data points in Y with probability $1 - \delta$ using majority voting using the crowd of workers W under asymmetric classification noise. Then for each data point $y \in Y$, it is sufficient to solicit*

$$\mathcal{O}\left(\frac{\log(T/\delta)}{(1 - 2\max(\bar{\eta}_W^+, \bar{\eta}_W^-))^2}\right)$$

labels.

The only added bound we need is an asymmetric-noise analogue for the Angluin and Laird [9] bound from Theorem 4.

Corollary 21 (to Theorem 4)**.** *If a learning algorithm that is given at least*

$$\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1 - 2\max(\eta^+, \eta^-))^2}\right)$$

samples labeled under the Dawid-Skene noise model [32] with parameters η^+ and η^- can produce a hypothesis h that minimizes disagreements with the noisy sample, then h satisfies the PAC criterion for the class $\mathcal{C}$, i.e. for any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ on X,

$$\Pr(d(h, h^*) \geq \epsilon) \leq \delta.$$

where $d(h, h^*)$ denotes the rate of disagreement between h and the target concept h^*.

Proof. The simplest proof of this is a reduction to the uniform noise case, as suggested by Blum and Kalai [46] for reducing from one-sided noise to two-sided noise. Without loss of generality, assume $\eta^+ > \eta^-$ (otherwise, we will flip the other label). We will flip each negative label with probability p. Hence, the new noise rates are $\eta'^+ = \eta^+ - \eta^+ p$ and $\eta'^- = \eta^- + (1 - \eta^-)p$. Making $\eta'^- = \eta'^+$ and solving for p yields $p = \frac{\eta^+ - \eta^-}{1+\eta^+ - \eta^-}$. The new symmetric noise rate is now $\eta'^+ = \eta'^- \leq \max(\eta^+, \eta^-)$, so we can apply the bound from Theorem 4 to finish the proof. □

We can now proceed to derive upper bounds on the task complexity of our algorithm adapted to this new setting.

Theorem 22. *Let $\epsilon, \delta > 0$. Suppose we identify one approximately good worker in Step 2 of the algorithm per label. Then*

$$\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{(1-2\max(\bar{\eta}_W^+, \bar{\eta}_W^-))^2(1-2\hat{\eta}_{i^*})^2}\right) + \mathcal{O}\left(\left(n + \frac{d}{\epsilon}\right)\frac{\log(1/\delta)}{(1-2\max(\eta_{i^*}^+, \eta_{i^*}^-))^2}\right)$$

tasks can be labeled by workers in order to efficiently PAC learn $\mathcal{C}$.

Proof. We sum the task complexity from each step in the algorithm. As before, we first sample the crowd $\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta(1-2\max(\eta_W^+, \eta_W^-))^2}\right)$ times (Theorem 11 and Theorem 20) in order to gather a ground truth set with probability $1 - \delta$. Using the ground truth set as the training set, we sample the crowd $\mathcal{O}\left(\frac{n}{\Delta^2}\log(n/\delta)\right)$ times (Theorem 11) in order to identify an approximately

good worker. We then use the approximately good worker to label $\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2\max(\eta_{i^*}^+,\eta_{i^*}^-))^2}\right)$ points (Corollary 21). Summing these components gives

$$\tilde{\mathcal{O}}\left(\frac{\log^2(n/\delta)}{\Delta(1-2\max(\bar{\eta}_W^+,\bar{\eta}_W^-)^2)}\right)+\mathcal{O}\left(\frac{n}{\Delta^2}\log(n/\delta)\right)+\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2\max(\eta_{i^*}^+,\eta_{i^*}^-))^2}\right).$$

Setting $\Delta=\frac{1/2-\hat{\eta}_{i^*}}{2}$ and simplifying yields the task complexity. □

The baseline approach in this setting would be to substitute the average one-sided error rates of workers into the bound from Corollary 21, yielding an upper bound of

$$\mathcal{O}\left(\frac{d\log(1/\delta)}{\epsilon(1-2\max(\bar{\eta}_W^+,\bar{\eta}_W^-))^2}\right).$$

With only slight adjustments to our algorithm and analysis, the bound we derive in Theorem 22 for this asymmetric noise setting is still an improvement on the baseline since the d/ϵ term is multiplied by a term that is a function of $\bar{\eta}_{i^*}^+$ and $\bar{\eta}_{i^*}^-$ instead of $\bar{\eta}_W^+$ and $\bar{\eta}_W^-$.

3.4.2 Per-worker Task Limits

In practice, the load per worker may be limited. In Section 3.4 we discuss how identifying the top K workers instead of one good worker in Step 2 of the algorithm reduces the load on workers. In this extension, we consider a different setting of limited worker loads where each worker can complete no more than B tasks. This setting is useful because in reality, workers will have time and energy limitations that will bound the number of tasks they can realistically complete. Suppose that $B>0$ denotes the task limit for each worker. We assume that B is

greater than the number of trials per arm required to identify top workers in Step 2 so that each worker has the capacity to help with labeling additional tasks, to some extent, in Step 3. We assume that in this setting, for Step 2 of the algorithm, we identify a Δ-optimal set of K workers instead of one Δ-optimal worker. We take this approach because we can use the limited worker load constraint and our knowledge of the number of tasks to be completed in Step 3 of the algorithm to determine how many workers K to identify in Step 2. We first determine the capacity remaining per worker after running the top-K MAB algorithm. We recall that

$$q = \mathcal{O}\left(\frac{n}{\Delta^2}\left(1 + \frac{\log(1/\delta)}{K}\right)\right)$$

from Theorem 3.6.

Lemma 23. *After implementing* `OptMAI(n,K,q)` *to identify the top* $K \leq n/2$ *workers, the number of tasks remaining per worker is at least*

$$B - \frac{4n^{.7}}{(1/2 - \bar{\eta}^*_{K,W})^2}\left(1 + \frac{\log(1/\delta)}{K}\right).$$

Proof. From Theorem 15, each arm is pulled at most $\mathcal{O}(q/n^{.3})$ times. We subtract this from the task limit B and set

$$\Delta = \frac{1/2 - \bar{\eta}^*_{K,W}}{2},$$

and simplify. $\square$

Recall that the bound in 3.3.3, derived from Theorem 4, is the number of data points that need to labeled by the selected K workers in Step 2 to complete the PAC learning algorithm. Dividing the bound from 3.3.3 by the capacity remaining per worker yields the number of top workers, K, that need to be identified in Step 2.

Lemma 24. *In Step 2 of the algorithm,*

$$K = \mathcal{O}\left(\left(\frac{d}{\epsilon} + n^{\cdot 7}\right) \frac{\log(1/\delta)}{B(1-2\bar{\eta}^*_{K,W})^2 - n^{\cdot 7}}\right)$$

workers will be identified.

Proof. The number of workers K is the number of data points that need to be labeled, as prescribed by the bound in 3.3.3, divided by the tasks remaining per worker, established in Lemma 23. After setting $\Delta = \frac{1/2-\bar{\eta}^*_{K,W}}{2}$,

$$K = \frac{\frac{2d}{\epsilon(1-2\bar{\eta}^*_{K,W})^2}\log(1/\delta)}{B - \frac{2n^{\cdot 7}}{(1-2\bar{\eta}^*_{K,W})^2}\left(1 + \frac{\log(1/\delta)}{K}\right)}.$$

The theorem follows from solving for K. □

Theorem 19 can now be extended to upper bound the number of tasks labeled by workers in this new setting by simply letting K be defined as in Lemma 24.

3.4.3 Agnostic PAC

It is possible that our symmetric or asymmetric classification noise model does not model the behavior of all workers. For instance, there may be workers who behave maliciously or

workers with error rates $\eta_i > 1/2$. On one end of the spectrum, each worker may have a fixed error rate. On the other end of the spectrum, there may be no assumptions on worker behavior at all, and this case is referred to as the agnostic setting.

PAC learning in the agnostic setting is usually computationally hard [47]. Hence, Awasthi et al. [34] assume an α fraction of workers are perfect performers and places no assumptions on the behavior of the remaining $1-\alpha$ workers. We would like to begin bridging the two ends of the spectrum in a similar way to account for workers that cannot be modeled by classification noise. Here, we begin to do this by showing a simple extension of the result of Awasthi et al. [34].

We let α denote the fraction of workers that can be modeled by persistent classification noise with $\eta_i \leq 1/2$. As in the work of Awasthi et al. [34], there are no assumptions on the behavior of the remaining $1-\alpha$ fraction of workers. PAC learning can be achieved in this setting by adapting the proof of Theorem 4.3 from Awasthi et al. [34]; in particular, in our proposed setting, the probabilistic guarantees of their Lemma 4.6 still hold. Thus, their algorithm still the extended setting we proposed, as we state in the following corollary.

Corollary 25 (to Theorem 4.3 of Awasthi et al. 2017)**.** *Let W be the subset of workers that can be modeled by persistent classification noise with average noise rate $\bar{\eta}_W$, and let α denote the fraction all workers that are in W. Then when*

$$\alpha(1 - \bar{\eta}_W) \geq 1/2,$$

a concept class $\mathcal{C}$ can be efficiently PAC learned from the crowd given the ability to efficiently find an ERM over $\mathcal{C}$.

CHAPTER 4

COLLABORATIVE PAC LEARNING IN THE PRESENCE OF CLASSIFICATION NOISE

4.1 Introduction

In this chapter, we study collaboratively learning classifiers that generalize well on set of distributions in the presence of classification noise. We first recall the noiseless collaborative PAC learning setting defined in Blum et al. [48]. In the collaborative PAC setting, there are k distributions on data set X, referred to as *players*, and a center node that orchestrates the learning process (see Figure 6). The goal of collaborative PAC learning is to learn classifiers from data provided by the players that generalize well on each of players' distributions simultaneously. Note that this differs from distributed PAC learning, whose goal is to learn classifiers that perform well on the mixture of players' distributions [49]. There are two styles of collaborative PAC learning. The first is the *personalized learning setting*, where the goal is to learn a classifier for each player with generalization error less than ϵ, with probability $1 - \delta$. The second setting is *centralized learning setting*, where the goal is learn a single classifier with generalization error less than ϵ on each players' distribution with probability $1 - \delta$. The efficiency of a collaborative learning algorithm is assessed by its *overhead*, defined as the ratio of the sample complexity of learning in the collaborative setting to the sample complexity of learning in the single player setting. An overhead of at least k indicates that the collaborative learning algorithm offers no

sample complexity benefit over individual PAC learning. An overhead less than k indicates that the collaborative algorithm is more sample efficient than individual PAC learning.

In real-world settings of this model, players may be noisy. In this chapter, we consider the classification noise setting. Classification noise (CN) is integrated into the collaborative PAC learning setting as follows. Each player has an individual noise rate $\eta_i < \frac{1}{2}$. When the center node requests a point from player i, the player sends instance-label pair (x, y) where $x \sim D_i$ and $y = f^*(x)$ with probability $1 - \eta_i$ or $y = \neg f^*(x)$ with probability η_i. We develop collaborative PAC learning algorithms robust to CN.

This chapter is organized as follows. First, we discuss the previous work and background. Second, we develop personalized and centralized collaborative learning algorithms robust to classification noise and prove sample and overhead bounds. Finally, we develop personalized learning algorithms with improved communication complexity in both the noiseless and classification noise settings.

4.2 Previous Work

The work in this chapter builds primarily on [13, 48, 50]. The collaborative PAC framework and the first algorithms were introduced in [48]. In [48], they develop an algorithm in the personalized setting with $O(\ln(k))$ overhead and an algorithm in the centralized setting with $O(\ln^2(k))$ overhead. They prove that the overhead lower bound in both the personalized and centralized setting when $k = d$ is $\Omega(\ln(k))$, thus, showing their personalized algorithm to be asymptotically optimal when $k = d$. We recall their personalized algorithm as Algorithm 2 and state their sample complexity result below.

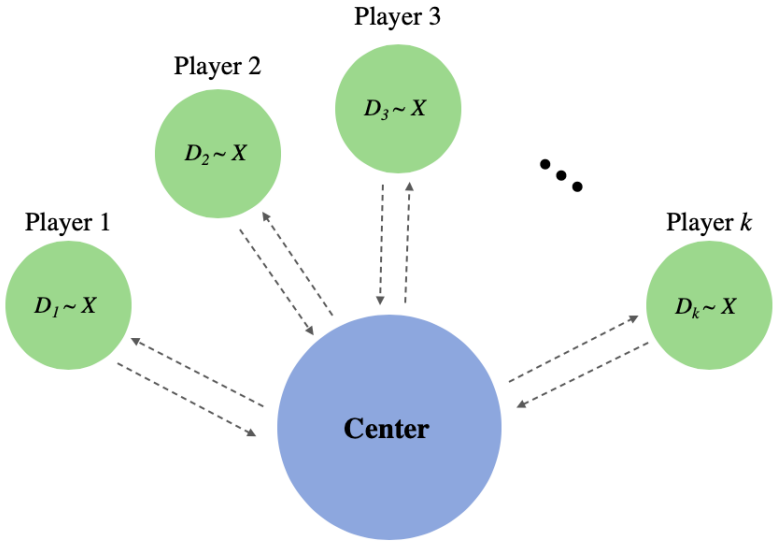

Figure 6: Collaborative PAC Setting

Theorem 26 ([48]). *For any $\epsilon, \delta > 0$, and hypothesis class H of finite VC-dimension d, the sample complexity of personalized collaborative PAC learning is*

$$m = O\left(\frac{\ln(k)}{\epsilon}\left((d+k)\ln\left(\frac{1}{\epsilon}\right) + k\ln\left(\frac{k}{\delta}\right)\right)\right).$$

When $k\ln(k) = O(d)$, the overhead is $O(\ln(k))$.

Subsequent works [13, 50] develop optimal centralized algorithms using a multiplicative weights approach. We recall their optimal centralized algorithm in Algorithm 3. In [50], they extend the sample complexity lower bound to any k and d, showing $\Omega(\max\{d\ln(k), k\ln(d)\}/\epsilon)$ samples are needed for personalized and centralized collaborative learning. This chapter builds

on the works of [13,48,50] by adapting their collaborative learning algorithms to handle classification noise. We also develop communication efficient alternatives in the personalized learning setting.

Regarding robustness, the previous work of [51] considers the collaborative PAC learning setting under the noise model where $1 - \alpha$ fraction of players behave truthfully while the remaining α fraction of players behave adversarially. They show that centralized learning is impossible in their setting. Our classification noise setting differs from theirs since we assume noise occurs only in the label of a sample and we assume every player has their own noise rate. Unlike [51], both personalized and centralized learning are possible in our noise setting.

We note that no previous work has addressed the communication complexity of collaborative PAC learning.

Algorithm 2: Personalized Collaborative PAC Learning [48]

Input: H, k distributions $D_i \sim X$, $\delta' = \delta/2\log(k)$, $\epsilon > 0$

Output: $f_1, ..., f_k \in H$

Let $N_1 = \{1, ..., k\}$;

for $j = 1, ..., \lceil \log(k) \rceil$ **do**

  Draw sample S of size $m_{\epsilon/4,\delta'}$ from mixture $D_{N_j} = \frac{1}{|N_j|} \sum_{i \in N_j} D_i$;

  Select consistent hypothesis $h_j \in H$ on S;

  $G_j \leftarrow \texttt{TEST}(h_j, N_j, \epsilon, \delta')$;

  $N_{j+1} = N_j \setminus G_j$;

  for $i \in G_j$ **do**

    $f_i \leftarrow h_j$;

  end

end

return $f_1, ..., f_k$

Procedure $\texttt{TEST}(h, N, \epsilon, \delta)$

  for $i \in N$ **do**

    Draw sample of size $T_i = O\left(\frac{\ln(\frac{|N|}{\epsilon\delta})}{\epsilon}\right)$ from D_i;

  end

  return $\{i \mid err_{T_i}(h) \leq \frac{3\epsilon}{4}\}$

Algorithm 3: Centralized Collaborative PAC Learning [13, 50]

Input: H, k distributions $D_i \sim X$, $\delta' = \delta/4t$, $\epsilon' = \epsilon/6$, $t = 150\lceil \log(\frac{k}{\delta}) \rceil$

Output: $h = \text{MAJORITY}(\{h_i\}_{i=1}^{t})$

Initialize $w_{i,0} = 1$ for all $i \in [1, k]$;

for $j = 1, ..., t$ **do**

Draw sample S of $m_{\epsilon'/16,\delta'}$ samples from mixture $D_j = \frac{1}{\sum_{i=1}^{k} w_{i,j}} \sum_{i=1}^{k} w_{i,j} D_i$;

Select consistent hypothesis $h_j \in H$ on S;

$G_j \leftarrow$ `FAST-TEST`$(h_j, N_j, \epsilon, \delta')$;

for $i = 1, ..., k$ **do**

$$w_{i,j+1} = \begin{cases} 2w_{i,j} & \text{if } i \notin G_j \\ w_{i,j} & \text{if } i \in G_j \end{cases};$$

end

end

return $h = MAJORITY(\{h_i\}_{i=1}^{t})$

Procedure `FAST-TEST`(h, k, ϵ, δ)

for $i = 1, ..., k$ **do**

Draw sample of size $T_i = O(\frac{1}{\epsilon})$ from D_i;

end

return $\{i \mid err_{T_i}(h) \leq \frac{3\epsilon}{4}\}$

4.3 Background

We now formally define notation and key concepts used in this chapter. Let X denote the instance space and $Y = \{0, 1\}$ denote the set of possible labels. Let H denote a hypothesis class with finite VC-dimension d. We will assume the realizable setting of learning, where the target hypothesis h^* is in the hypothesis class H. The collaborative learning scenario with CN is similar to the original formulation [48] but with the addition of CN. Our setting consists of k players, each with their own distributions $D_i \sim X$ and classification noise rates $\eta_i < \frac{1}{2}$. The center node orchestrating the learning process knows the players' noise rates but does not know the players' distributions. In the CN model, we treat each player as an oracle, denoted by $\mathrm{EX}_{\eta_i}(\cdot)$, that returns an instance-label pair (x, y), where $x \sim D_i$, and with probability $1 - \eta_i$, $y = h^*(x)$, or with probability η_i, $y = \neg h^*(x)$.

A learned classifier will be evaluated by its error on an individual player's distributions. We review the notions of error used in the CN setting. Let $err_T(\mathrm{EX}_{\eta_i}(\cdot), h)$ denote the empirical error of concept $h \in H$ on T points generated from $\mathrm{EX}_{\eta_i}(\cdot)$. The definition of empirical error is standard and defined as

$$err_T(\mathrm{EX}_{\eta_i}(\cdot), h) = \frac{1}{T} \sum_{j=1}^{T} \mathbf{1}_{\mathrm{EX}_{\eta_i}(x_j) \neq h(x_j)}.$$

The generalization error in more subtle in the CN setting. There are two types of generalization errors of h to consider. The first is the error of h on the *noisy distribution*, that is, the distribution D_i in the presence of label noise. The second is the error of h on the underlying

clean distribution, that is, the distribution D_i without label noise. In the CN setting, the learner only has access to samples from the noisy distribution, but the goal of learning in this setting is to generalize well with respect to the clean distribution. With access only to the noisy distribution, we use the generalization error with respect to the noisy distribution as a stepping stone in our analysis. The generalization error on the noisy data distribution, $\mathrm{EX}_{\eta_i}(\cdot)$, is defined as

$$err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h) = \mathbb{E}_{T\sim D_i^T}[err_T(\mathrm{EX}_{\eta_i}(\cdot), h)].$$

The generalization error of concept h on the clean data distribution, D_i, denoted $err_{D_i}(h)$, and defined as follows,

$$err_{D_i}(h) = \mathbb{E}_{T\sim D_i^T}[err_T(h)] = \Pr_{x\sim D_i}[h(x) \neq h^*(x)].$$

We now formally define the collaborative PAC learning criteria for personalized and centralized learning in the presence of classification noise.

Definition (Personalized Learning with Classification Noise)**.** *A learning algorithm A is a* personalized learning algorithm in the presence of classification noise *for concept class H if for any target $h^* \in H$, any k distributions $\{D_1, ...D_k\}$ on X each with noise rate $n_i < \frac{1}{2}$, and for any $\epsilon, \delta > 0$, A returns a concept $h_i \in H$ for each D_i (not necessarily distinct) so that with probability $1 - \delta$, the following is true*

$$\wedge_{i=1}^{k}(err_{D_i}(h_i) < \epsilon),$$

and A has sample complexity $m_A = poly(\frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{(1-2\eta_{MAX})})$.

Definition (Centralized Learning with Classification Noise)**.** *A learning algorithm A is a* centralized learning algorithm in the presence of classification noise *for concept class H if for any target* $h^* \in H$*, any k distributions* $\{D_1, ...D_k\}$ *on X each with noise rate* $n_i < \frac{1}{2}$*, and for any* $\epsilon, \delta > 0$*, A returns a single concept* $h \in H$ *so that with probability* $1 - \delta$*, the following is true*

$$\wedge_{i=1}^{k}(err_{D_i}(h) \leq \epsilon),$$

and A has sample complexity $m_A = poly(\frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{(1-2\eta_{MAX})})$.

In addition to considering sample complexity, we will consider the overhead of our algorithms, formally defined as follows.

Definition (Overhead [48])**.** *The* overhead *of a collaborative learning algorithm is the ratio of the sample complexity of collaborative PAC learning in the presence of noise with k players to the sample complexity of traditional PAC learning in the presence of noise in the single player setting.*

We compute the overhead in our classification noise setting with respect to the greatest error rate among all k players, denoted as η_{MAX}. In our algorithms and analysis, we use the sample complexity results of traditional PAC learning with classification noise, Theorem [9] and Theorem [12]. Let $m_{\epsilon,\delta,\eta} = O\left(\frac{d\log(1/\delta)}{\epsilon(1-2\eta)^2}\right)$ denote the sample complexity upper bound prescribed by Theorem [9]. This also represents the sample complexity of learning with classification noise

in the single player setting, so this will be used in our computations of overhead. Finally, for $N \subseteq \{1, ..., k\}$, we let $\bar{\eta}_N = \frac{1}{|N|} \sum_{i=1}^{|N|} \eta_i$ denote the average error rate of all players in N.

4.4 Personalized Learning with Classification Noise

Before presenting our algorithm, we first consider the baseline approach to personalized learning with CN. Naively, the center can sample $m_{\epsilon,\delta,\eta_i}$ from each player and learn an empirical risk minimizer (ERM), following exactly as in traditional PAC learning with CN outlined in Theorem 4. In this case, the sample complexity is $\sum_{i=1}^{k} m_{\epsilon,\delta,\eta_i} = O(k m_{\epsilon,\delta,\eta_{\text{MAX}}})$. The sample complexity for learning with a single player is $m_{\epsilon,\delta,\eta_{\text{MAX}}}$, so the overhead of the baseline is $O(k)$. The goal is to develop algorithms with overhead less than $O(k)$.

Our algorithm for personalized learning in the presence of CN, Algorithm 4, improves upon the overhead of the baseline, with overhead of $O(\log(k)\log(\log(k))$. We summarize how our algorithm differs from noiseless personalized learning, Algorithm 2. In the first and second step of our algorithm, the center draws $m_{\epsilon/4,\delta',\bar{\eta}_{N_j}}$ points in total from the mixture of players and returns an ERM. In contrast, in Algorithm 2, the center draws $m_{\epsilon/4,\delta'}$ samples and returns a consistent hypothesis. In the setting of CN, the existence of a hypothesis in H consistent with a sample generated from a noisy distribution is not guaranteed. Therefore, in our algorithm we find instead the more general ERM. By Theorem 4 the ERM has error $\epsilon/4$ when trained on $m_{\epsilon/4,\delta',\bar{\eta}_{N_j}}$ samples. The final step of our algorithm, the `CN-TEST` step, differs from the `TEST` step in Algorithm 2 in that ours accounts for the individual noise rates of the players. Essentially, players draw a factor of $\frac{1}{(1-2\eta_i)}$ more samples in `CN-TEST` than in `TEST` and the testing criterion

is adjusted to reflect the relationship between drawing from noisy distribution and generalizing on the clean distribution.

We now formally prove the correctness of our algorithm. Besides proving the correctness of our adjustments to handle CN outlined above, most of the proof will follow immediately from the correctness results of the original personalized learning (Algorithm 2) proved in [48]. In the following lemmas, we prove the correctness of our adjustments. We start by showing formally that the first and second steps in our algorithm yield a classifier that performs with error $\epsilon/4$ on the mixture.

Lemma 27. *The ERM h_j established in Step 2 of Algorithm 4 has error no more than $\frac{\epsilon}{2}$ on at least half of the distributions in N_j.*

Proof. First, note that the mixture D_{N_j} has expected error rate $\bar{\eta}_{N_j}$. By Theorem 4, the ERM h_j trained on $m_{\epsilon/4,\delta',\eta_{\bar{N}_j}}$ samples drawn from D_{N_j} has $err_{D_{N_j}}(h_j) \leq \frac{\epsilon}{4}$. Now, as shown in [48], Markov's inequality yields the result,

$$\Pr\left[err_{D_{N_j}}(h_j) \leq 2\left(\frac{\epsilon}{4}\right)\right] \geq \frac{1}{2}.$$

□

Next, we prove the correctness of the `CN-TEST` routine. We use the following lemma and proof from [9] that connects the generalization error of a concept h on the noisy distribution to the generalization error of h on the underlying clean distribution.

Algorithm 4: Personalized Learning with CN

Input: H, k distributions $D_i \sim X$ with error rates $\eta_i < \frac{1}{2}$, $\delta' = \delta/2\log(k)$, $\epsilon > 0$

Output: $f_1, ..., f_k \in H$

Let $N_1 = \{1, ..., k\}$;

for $j = 1, ..., \lceil \log(k) \rceil$ **do**

  Draw sample S of size $m_{\epsilon/4,\delta',\bar{\eta}_{N_j}}$ from mixture $D_{N_j} = \frac{1}{|N_j|}\sum_{i \in N_j} D_i$;

  Select ERM hypothesis $h_j \in H$ on S;

  $G_j \leftarrow$ `CN-TEST`$(h_j, N_j, \epsilon, \delta')$;

  $N_{j+1} = N_j \setminus G_j$;

  for $i \in G_j$ **do**

    $f_i \leftarrow h_j$;

  end

end

return $f_1, ..., f_k$

Procedure `CN-TEST`(h, N, ϵ, δ)

  for $i \in N$ **do**

    Draw sample of size $T_i = O\left(\frac{\ln(\frac{|N|}{\delta})}{\epsilon(1-2\eta_i)}\right)$ from D_i;

  end

  return $\{i \mid err_{T_i}(EX_{\eta_i}, h_j) \leq \eta_i + \frac{3\epsilon}{4}(1-2\eta_i)\}$

Lemma 28 ([9])**.** *Let D denote a distribution on X. Let η_i denote the CN error rate and $h^* \in H$ denote the target function. Then,*

$$err_D(EX_{\eta_i}(\cdot), h) = \eta_i + err_D(h)(1 - 2\eta_i).$$

Proof. There are two ways in which h can disagree with $\mathrm{EX}_{\eta_i}(\cdot)$ on a point $x \in X$:

1. $\mathrm{EX}_{\eta_i}(\cdot)$ labels x correctly with probability $1 - \eta_i$ and h disagrees with h^*, or
2. $\mathrm{EX}_{\eta_i}(\cdot)$ labels x incorrectly with probability η_i and h agrees with h^*.

The probability that either of these two events occurs is $(1 - \eta_i)err_D(h) + \eta_i(1 - err_D(h)) = \eta_i + err_D(h)(1 - 2\eta_i)$. □

We prove the correctness of `CN-TEST` in the following lemmas. The first direction of correctness is proved as follows.

Lemma 29. *With probability $1 - \delta'$, if h_j passes `CN-TEST` then $err_{D_i}(EX_{\eta_i}(\cdot), h_j) \leq \eta_i + (1 - 2\eta_i)\epsilon$. Hence, $err_{D_i}(h_j) \leq \epsilon$.*

Proof. As in [13, 48, 50], we use multiplicative Chernoff bounds (Theorem 8) to prove the performance of `CN-TEST`. Our Chernoff bounds are adjusted to handle each player's noise rate η_i. By Lemma 28, we scale the generalization error on the noisy distribution to the generalization error on clean distribution.

Assume that $err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) \geq \eta_i + (1 - 2\eta_i)\epsilon$ and that $T_j \geq \frac{32}{\epsilon(1-2\eta_i)} \ln\left(\frac{|N|}{\delta'}\right)$. Let $P = err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i$. We use the Chernoff bound on the random variable $err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j)$,

the empirical error of h_j of T_j samples. If h_j passes `CN-TEST` for player i, then we have the following inequality:

$$\Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) \leq \eta_i + \frac{3\epsilon}{4}(1-2\eta_i)\right] = \Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \leq \frac{3\epsilon}{4}(1-2\eta_i)\right]$$

Computing the expected value of $err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j)$, we have

$$\begin{aligned}
\mathbb{E}_{T_j \sim D_i^{T_j}}[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i] &= \mathbb{E}_{T_j \sim D_i^{T_j}}[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j)] - \eta_i \\
&= err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \\
&\geq (1-2\eta_i)\epsilon.
\end{aligned}$$

Applying the Chernoff bound gives:

$$\begin{aligned}
\Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) \leq \eta_i + \frac{3\epsilon}{4}(1-2\eta_i)\right] &= \Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \leq \frac{3\epsilon}{4}(1-2\eta_i)\right] \\
&\leq \Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \leq (1-\frac{1}{4})P\right] \\
&\leq \exp\left(-\frac{1}{2}\left(\frac{1}{4}\right)^2(1-2\eta_i)\epsilon T_j\right) \\
&\leq \frac{\delta'}{|N|}
\end{aligned}$$

By the union bound over $|N|$ players, if $err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) \geq \eta_i + (1-2\eta_i)\epsilon$ then h_j passes `CN-TEST` with probability at most δ'. Hence, `CN-TEST` is correct with probability $1-\delta'$. By Lemma 28, this implies $err_{D_i}(h_j) \leq \epsilon$. □

To prove the second direction of correctness, we again employ multiplicative Chernoff bounds, but the multiplicative factor is more subtle, as in [13].

Lemma 30. *Let* $s = \frac{\epsilon(1-2\eta_i)}{4(err_{D_i}(EX_{n_i}(\cdot),h_j)-\eta_i)}$. *Suppose* $err_{D_i}(EX_{\eta_i}(\cdot), h_j) - \eta_i \leq \frac{\epsilon}{2}(1-2\eta_i)$. *Then,*

$$(1+s)(err_{D_i}(EX_{\eta_i}(\cdot), h_j) - \eta_i) \leq \frac{3\epsilon}{4}(1-2\eta_i).$$

Proof. The proof follows as in [13], but casted to our CN setting. Suppose the claim is true, $\frac{3\epsilon}{4}(1-2\eta_i) \geq (1+s)(err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j)$. Then,

$$\begin{aligned}
\frac{3\epsilon(1-2\eta_i)}{4(err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i))} &\geq 1+s \\
\frac{3\epsilon(1-2\eta_i)}{4(err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i))} &\geq 1 + \frac{(1-2\eta_i)\epsilon}{4(err_{D_i}(h_j, \mathrm{EX}_{n_i}) - \eta_i)} \\
\frac{\epsilon(1-2\eta_i)}{2(err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i)} &\geq 1
\end{aligned}$$

The last inequality is true since by assumption $err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \leq (1-2\eta_i)\frac{\epsilon}{2} \implies 2(err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i) \leq \epsilon(1-2\eta_i)$. □

Lemma 31. *With probability* $1-\delta'$, *if* $err_{D_i}(EX_{\eta_i}(\cdot), h_j) \leq \eta_i + (1-2\eta_i)\frac{\epsilon}{2}$, *then* h_j *passes* `CN-TEST`. *Hence, if* $err_{D_i}(h_j) \leq \frac{\epsilon}{2}$, *then* h_j *passes* `CN-TEST`.

Proof. As in [13], we invoke our CN-analogue, Lemma 30, and use the multiplicative Chernoff bounds in Theorem 8. Let s be defined as in Lemma 30. We consider two cases, (1) when

$err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \geq (1-2\eta_i)\frac{\epsilon}{4}$ and (2) when $err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \leq (1-2\eta_i)\frac{\epsilon}{4}$. First, suppose $err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \geq (1-2\eta_i)\frac{\epsilon}{4}$. Let $P = err_{D_i}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i$. Then, $s < 1$ and,

$$\Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \geq (1-2\eta_i)\frac{3\epsilon}{4}\right] \leq \Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \geq (1-2\eta_i)(1+s)\right].$$

By multiplicative Chernoff bounds and Lemma 30,

$$\begin{aligned}
\Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \geq (1-2\eta_i)\frac{3\epsilon}{4}\right] &= \Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \geq 3(1-2\eta_i)\frac{\epsilon}{4}\right] \\
&\leq \Pr\left[err_{T_j}(\mathrm{EX}_{\eta_i}(\cdot), h_j) - \eta_i \geq (1+s)(1-2\eta_i)\frac{\epsilon}{4}\right] \\
&\leq \exp\left(-\frac{1}{3}\left(\frac{(1-2\eta_i)\epsilon}{4(err_{D_i}(\mathrm{EX}_{n_i}(\cdot), h_j) - \eta_i)}\right)^2 PT_j\right) \\
&\leq \exp\left(-\frac{1}{12}(1-2\eta_i)\epsilon T_j\right) \\
&\leq \frac{\delta'}{|N|}
\end{aligned}$$

when $T_j = O\left(\frac{\ln(\frac{|N|}{\delta'})}{\epsilon(1-2\eta_i)}\right)$. The second case follows by a symmetric argument. In both cases, by the union bound over all $|N|$ players, h_j fails `CN-TEST` with probability less than δ'. Since $err_{D_i}(h_j, \mathrm{EX}_{\eta_i}) \leq \eta_i + (1-2\eta_i)\frac{\epsilon}{2}$, Lemma 28 implies $err_{D_i}(h_j) \leq \frac{\epsilon}{2}$. □

We combine the above lemmas to prove the correctness of our algorithm (Algorithm 4).

Theorem 32. *Algorithm 4 is a personalized collaborative learning algorithm in the presence of CN.*

Proof. Analogous to [48], the first and second step in our algorithm yield a classifier that performs with error $\epsilon/4$ on the mixture of distributions, proven in Lemma 27. By Lemmas 29, 30, and 31, if $err_{D_i}(h_j) \leq \frac{\epsilon}{2}$, then h_j passes `CN-TEST` and if h_j passes `CN-TEST`, then $err_{D_i}(h_j) \leq \epsilon$. Our `CN-TEST` exhibits the same performance as `TEST` step in the noiseless personalized learning algorithm (Algorithm 2). Therefore, the rest of the proof of correctness follows directly from [48], where they prove that in each round, at least half of the players are removed with probability $1-\frac{\delta}{\log(k)}$. Therefore, after at most $\log(k)$ rounds, each player is assigned a hypothesis with generalization error less than ϵ, with respect to their clean distribution, with probability $1-\delta$. Therefore, the learning criteria is satisfied. □

Now, we consider the sample complexity and overhead of our algorithm.

Theorem 33. *The sample complexity of personalized learning with CN, Algorithm 4, is*

$$O\left(\log(k)\left(\frac{k\ln(\frac{k\log(k)}{\delta})}{\epsilon(1-2\eta_{MAX})}+\frac{d\ln(\frac{\log(k)}{\delta})}{\epsilon(1-2\eta_{MAX})^2}\right)\right).$$

When $k\ln(k) = O(d)$, the overhead is $O(\log(k)\log(\log(k)) = \tilde{O}(\log(k))$.

Proof. Recall that $\delta' = O(\frac{\delta}{\log(k)})$. Our algorithm implements `CN-TEST` for $\log(k)$ rounds on at most $|N| = k$ players, using

$$O\left(\log(k)\frac{k\ln(\frac{k\log(k)}{\delta})}{\epsilon(1-2\eta_{\text{MAX}})}\right)$$

samples. Let $K = \{1, ..., k\}$. The algorithm learns at most $\log(k)$ ERMs via Theorem 4, using

$$O(\log(k) m_{\epsilon/4, \delta', \bar{\eta}_K}) = O\left(\log(k) \frac{d \ln(\frac{\log(k)}{\delta})}{\epsilon(1 - 2\bar{\eta}_K)^2}\right)$$

samples. Note that $\bar{\eta}_K \leq \eta_{\text{MAX}}$. Summing gives the sample complexity result. Simplifying the above sample complexity with respect to constant ϵ, δ, and η_{MAX}, and using the fact that $k \ln(k) = O(d)$, we have $O(d \log(k) \log(\log(k))$. Simplifying the sample complexity of single-player PAC learning with CN under the same conditions gives sample complexity $O(d)$. Therefore, the overhead of our algorithm is $O(\log(k) \log(\log(k))$. □

We have shown that personalized collaborative PAC learning with CN can be performed with nearly the same overhead as in the noiseless setting (Theorem 26). Next, we move on to the centralized collaborative PAC learning setting in the presence on CN.

4.5 Centralized Learning with Classification Noise

We now consider centralized collaborative learning in the presence of CN, where the goal is to learn a single classifier that generalizes well on each player's distribution. We recall the noiseless centralized learning algorithm in Algorithm 3 and present our algorithm robust to CN in Algorithm 5. As in [13, 50], our centralized learning algorithm does not iteratively remove players from consideration, but rather, re-weights the players' distributions at each round,

reminiscent of boosting. Let $\bar{\eta}_j$ denote the weighted average of noise rates determined by the distributions weights in round j,

$$\bar{\eta}_j = \frac{1}{\sum_{i=1}^{k} w_{i,j}} \sum_{i=1}^{k} w_{i,j}\eta_i.$$

To handle CN, we make adjustments to the noiseless centralized learning (Algorithm 3) algorithm that resemble the adjustments we made to noiseless personalized learning algorithm in the previous section. In summary, our algorithm differs from the noiseless centralized learning algorithm in two ways. First, we sample $m_{\epsilon'/16,\delta',\bar{\eta}_j}$ from the weighted mixture to account for the classification noise (by Theorem 4). Second, we use `CN-FAST-TEST`, a modification of `FAST-TEST`, to account for classification noise. Aside from the proof of correctness of our adjustments to handle CN, the proof of correctness of our algorithm will follow immediately from [13, 50].

We start by showing the correctness of Steps 1 and 2.

Lemma 34. *The ERM h_j established in Step 2 of Algorithm 5 satisfies $err_{D_j}(h_j) \leq \frac{\epsilon'}{16}$.*

Proof. The expected error rate of weighted distribution D_j is $\bar{\eta}_j$. Therefore, the result follows by Theorem 4. □

We now prove the correctness of `CN-FAST-TEST`.

Lemma 35. *(1) With probability at least .99, if h_j passes `CN-FAST-TEST` then $err_{D_i}(h_j, EX_{\eta_i}) \leq \eta_i + (1-2\eta_i)\epsilon$. Hence, $err_{D_i}(h_j) \leq \epsilon$. (2) With probability at least .99, if $err_{D_i}(h_j, EX_{\eta_i}) \leq \eta_i + (1-2\eta_i)\frac{\epsilon}{2}$, then h_j passes `CN-FAST-TEST`. Hence, if $err(h_j) \leq \frac{\epsilon}{2}$ then h_j passes `CN-FAST-TEST`.*

Algorithm 5: Centralized Learning with CN

Input: H, k distributions $D_i \sim X$, $\delta' = \delta/4t$, $\epsilon' = \epsilon/6$, $t = 150\lceil \log(\frac{k}{\delta}) \rceil$

Output: $h = \text{MAJORITY}(\{h_i\}_{i=1}^{t})$

Initialize $w_{i,0} = 1$ for all $i \in [1, k]$ **for** $j = 1, ..., t$ **do**

Draw sample S of $m_{\epsilon'/16,\delta',\bar{\eta}_j}$ samples from mixture $D_j = \frac{1}{\sum_{i=1}^{k} w_{i,j}} \sum_{i=1}^{k} w_{i,j} D_i$;

Select ERM hypothesis $h_j \in H$ on S;

$G_j \leftarrow$ `CN-FAST-TEST`$(h_j, N_j, \epsilon', \delta')$;

for $i = 1, ..., k$ **do**

$$w_{i,j+1} = \begin{cases} 2w_{i,j} & \text{if } i \notin G_j \\ w_{i,j} & \text{if } i \in G_j \end{cases};$$

end

end

return $h = MAJORITY(\{h_i\}_{i=1}^{t})$

Procedure `CN-FAST-TEST`(h, N, ϵ, δ)

for $i \in N$ **do**

Draw sample of size $T_i = O(\frac{1}{\epsilon'(1-2\eta_i)})$ from D_i;

end

return $\{i \mid err_{T_j}(h, EX_{\eta_i}) \leq \eta_i + \frac{3\epsilon}{4}(1 - 2\eta_i)$

Proof. Follows from multiplicative Chernoff bounds (Theorem 8). □

Recall that in the centralized learning algorithms, players are never eliminated from the algorithm. Instead, their weights are increased or decreased according to the performance of h_j on their distributions. Over t rounds, this gives have t classifiers with varying performance on the players. As in Algorithm 3, we take the majority vote of these classifiers as our final classifier. Thus, we need to verify that the majority vote is sufficiently accurate. We use the following claim directly from [13, 50] which also holds in our setting with CN.

Lemma 36 ([13, 50])**.** *For each player, the number of classifiers that have error more than ϵ' is less than .4t with probability $1 - \delta$.*

Therefore, by the above lemma and analysis in [13, 48, 50], with probability $1 - \delta$, the error of the final hypothesis h is less than ϵ for every player's distribution. Combining Lemmas 34, 35, 36 proves the correctness of our algorithm.

Theorem 37. *Algorithm 5 is a centralized collaborative PAC algorithm in the presence of CN.*

We compute the sample complexity of our algorithm.

Theorem 38. *The sample complexity of Algorithm 5 is*

$$O\left(\log\left(\frac{k}{\delta}\right)\left(\frac{d\log(\frac{\log(k/\delta)}{\delta})}{\epsilon(1-2\eta_{MAX})^2}+\frac{k}{\epsilon(1-2\eta_{MAX})}\right)\right).$$

When $k\ln(k) = O(d)$, the overhead is $O(\log(k)\log(\log(k)))$.

Proof. Recall that $\delta' = O\left(\frac{\delta}{\log(k/\delta)}\right)$. The algorithm runs for a total of $t = O(\log(\frac{k}{\delta}))$ rounds. In each round, an ERM is learned on the mixture of players using

$$m_{\frac{\epsilon'}{16},\delta',\bar{\eta}_j} = O\left(\frac{d\log(\frac{\log(k/\delta)}{\delta})}{\epsilon(1-2\bar{\eta}_j)^2}\right)$$

samples. Then, the algorithm implements `CN-FAST-TEST` at each round for all k players, costing $O(\frac{k}{\epsilon(1-2\eta_i)})$ samples. Therefore, over all t rounds, the sample complexity is

$$O\left(\log\left(\frac{k}{\delta}\right)\left(\frac{d\log(\frac{\log(k/\delta)}{\delta})}{\epsilon(1-2\eta_{\text{MAX}})^2} + \frac{k}{\epsilon(1-2\eta_{\text{MAX}})}\right)\right).$$

Setting ϵ, δ, and η_{MAX} to constants and setting $k\ln(k) = O(d)$, the sample complexity simplifies to $O(d\log(k)\log(\log(k)))$ indicating $O(\log(k)\log(\log(k)))$ overhead. □

4.6 Communication Complexity of Personalized Learning

Thus far, we have considered only the sample complexity and overhead of collaborative PAC learning algorithms with and without noise. Likewise, related work focuses only on sample complexity and overhead. In this section, we shift our focus to the communication complexity of personalized collaborative learning. We define the communication complexity as the total number of samples and bits communicated to the center node during the execution of the algorithm. The communication complexity is dependent on the implementation of an algorithm. Therefore, to compute communication costs accurately and consistently, we carefully outline the implementation assumptions of our collaborative learning model. First, we assume that the

learning criteria is personalized collaborative learning and that the algorithm is complete when each player is in possession of a classifier that has error $< \epsilon$. Second, we assume that each player has computing power, access to the hypothesis class H, and access to learning parameters ϵ, δ, and k. Third, we assume the broadcast model of communication as in [49], which means that all players can observe all samples or bits sent to the center. Fourth, for clarity in computations and analysis, we assume δ is a constant.

We first revisit the baseline approach of personalized learning. Previously in this chapter, we described the personalized baseline as the approach where each player sends $O(\frac{d}{\epsilon})$ samples from their own distributions to the center. The communication efficient personalized baseline is the same but implemented differently. Namely, each player draws $O(\frac{d}{\epsilon})$ locally and individually learns their own classifier. This implementation requires no communication to the center. Although both baselines have the same sample complexity, in light of the model assumptions and communication costs, the latter baseline is most competitive and will serve as our default baseline in this section.

The sample complexity of the personalized learning algorithm (Algorithm 2) is proved in Theorem 26. We compute the communication complexity of their algorithm in the following theorem.

Theorem 39. *The communication complexity of personalized collaborative learning, Algorithm 2, is*

$$\tilde{O}\left(\log(k)\frac{d}{\epsilon}\right)$$

samples plus $O(k\log(k))$ additional bits of communication.

Proof. We describe the implementation details for personalized learning. Consider round j. In the first step, each player in N_j sends $m_{\epsilon/4,\delta'}/|N_j|$ points to the center to learn a common consistent hypothesis, h_j, costing $\tilde{O}(\frac{d}{\epsilon})$ samples of communication. Note that each player sending exactly $m_{\epsilon/4,\delta'}/|N_j|$ samples is the same, in expectation, as drawing from the multinomial distribution on the players' distributions and therefore, the hypothesis learned in the next step will have the same generalization guarantees. By assumption of the broadcast model, each player can see the samples transmitted by other players so all players can learn a consistent hypothesis locally, costing no communication in this step. After learning the consistent hypothesis h_j, each player implements `TEST` locally, costing no communication. Afterwards, they communicate a single bit to the center indicating whether or not `TEST` passed with h_j, costing $O(k)$ bits of communication. Therefore, the total communication over $\log(k)$ rounds is $\tilde{O}(\log(k)\frac{d}{\epsilon})$ samples plus additional $O(k\log(k)) = \tilde{O}(k)$ bits of communication. □

The main contribution of personalized collaborative learning (Algorithm 2) is the exponential improvement in the sample complexity with respect to the baseline; in particular, the sample complexity improves to logarithmic dependence on k in (Algorithm 2), a drastic improvement for large k. We now consider the scenario where k is large *and* players want to learn classifiers with low error. In this scenario, ϵ will be very small, so improving ϵ dependence is crucial. In this section, we show that the dependence on ϵ can be improved exponentially in communication complexity. Our communication efficient personalized learning algorithm

achieves the same sample complexity as personalized learning but with $O(\log(\frac{1}{\epsilon}))$ dependence in communication complexity.

Table I summarizes the sample and communication complexities of the baseline approach, the personalized learning algorithm from [48] (Algorithm 2), and in bold, our algorithm (Algorithm 7). The table also tracks the number of additional bits needed to orchestrate the algorithms. However, we are primarily concerned with the number of samples communicated (since this can be expensive for large d) and therefore, we aim to reduce this cost.

TABLE I: Samples and Communication in Personalized Learning

	Sample Complexity	**Samples Communicated**	**Bits Communicated**
Baseline	$\tilde{O}(k\frac{d}{\epsilon})$	$\tilde{O}(1)$	$\tilde{O}(1)$
Algorithm 2	$\tilde{O}(\log(k)\frac{d}{\epsilon})$	$\tilde{O}(\log(k)\frac{d}{\epsilon})$	$\tilde{O}(k)$
Algorithm 7	$\tilde{O}(\log(k)\frac{d}{\epsilon})$	$\tilde{O}(\log(k)d\log(\frac{1}{\epsilon}))$	$\tilde{O}(k\log(d)\log(\frac{1}{\epsilon}))$

To reduce communication costs, we replace the first step in personalized learning with distributed boosting [49]. We recall the distributed boosting algorithm in Algorithm 6. The distributed boosting algorithm is essentially a distributed implementation of AdaBoost ([52]) that requires only $\tilde{O}(\log(\frac{1}{\epsilon}))$ rounds to yield a consistent hypothesis. The learning objective of distributed boosting is to learn a classifier with error less than ϵ on the mixture of distributions, $\frac{1}{k}\sum_i^k D_i$. We recall the communication complexity of distributed boosting below.

Theorem 40 ([49]). *Any class H of finite VC-dimension d can be learned to error ϵ in $\tilde{O}(\log(\frac{1}{\epsilon}))$ rounds and $O(d)$ examples plus $O(k\log(d))$ bits of communication per round using the distributed boosting algorithm, Algorithm 6.*

The sample complexity of distributed boosting was not analyzed in [49]. We derive the sample complexity in the next section, showing that distributed boosting can be implemented with the same sample complexity as boosting.

4.7 Sample Complexity of Distributed Boosting

We first recall the sample complexity of the original implementation of AdaBoost ([52]). The sample complexity is the size of the reservoir of points used in the AdaBoost routine, denoted by S. To compute the sample complexity, we first need the VC-dimension of the hypothesis class H after T rounds of boosting.

Lemma 41 ([52]). *Suppose the weak learner in AdaBoost learns a classifier with constant error in each round. Then, $\tilde{O}(\ln(\frac{1}{\epsilon}))$ rounds of AdaBoost are needed to learn a classifier with zero training error.*

Let d_{boost} denote the VC-dimension of the hypothesis class after T rounds of boosting.

Lemma 42 ([52]). *Let H denote the base class of hypotheses with VC-dimension d. After T rounds of boosting, the resulting hypothesis class has VC-dimension $d_{boost} = O(dT\log(T)) = \tilde{O}(dT)$.*

We now show the known result of the sample complexity of AdaBoost.

Algorithm 6: Distributed Boosting [49]

Input: k players with sample reservoirs $S_i \sim D_i$, $\epsilon > 0$, β-weak learning oracle on H

Initialize player weights: $w_{i,t=0} = 1$ for each player $\implies W_{t=0} = \sum_{i=1}^{k} w_{i,t=0} = k$

Initialize sample weights: $v_{i,j,t=0} = \frac{1}{|S_i|}$ for each point $x_j \in S_i$

Set $\beta = \frac{1}{2}$.

For round $t = 1, ..., T$:

1: [Pre-sampling] Sample $O(\frac{d}{\beta}\log(\frac{1}{\beta}))$ times from the multinomial distribution defined by $\frac{w_{i,t-1}}{W_{t-1}}$ to determine the number of samples to request from each player, $n_{i,t}$. Transmit $n_{i,t}$ to each player.

2: [Sampling] Each player samples $n_{i,t}$ from S_i according to weights $v_{i,j,t}$. All $n_{i,t}$ points are transmitted to the center.

3: [Weak Learning] Each player learns the same $\frac{\beta}{2}$-good classifier simultaneously.

4: [Updating Sample Weights] Each player updates $v_{i,j,t+1}$ according to the underlying boosting algorithm. Set $w_{i,t} = \sum_{j=1}^{|S_i|} v_{i,j,t-1}$ and transmit approximation of $w_{i,t}$ to the center.

Output: $h = \text{sign}(\sum_{i=1}^{T} h_i)$.

Lemma 43. *We have that the sample complexity of AdaBoost is*

$$m_{boost} = O\left(\frac{d_{boost}}{\epsilon}\right) = \tilde{O}\left(\frac{d}{\epsilon}\right).$$

Proof. Follows immediately from Lemma 41, Lemma 42, and PAC sample complexity bounds in Theorem 1. □

We now proceed with the sample complexity of distributed boosting. In distributed boosting (Algorithm 6) the sample complexity is the sum over the players' sample reservoirs, $\sum_{i=1}^{k} S_i$. From classic learning theory we know lower and upper bounds on $\sum_{i=1}^{k} |S_i|$, but the size with which to initialize each individual S_i is unclear. In particular, the issue is that during each round of distributed boosting, the number of samples requested from a player can increase and in the original analysis of Algorithm 6, S_i is assumed to be large enough so each player always has plenty of samples to send when requested. Assuming the worst case that each player has at least $O(d)$ samples in their reservoir leads to sample complexity dependence on $O(kd)$, which will not lead to a competitive sample complexity in our collaborative learning setting. We show that it is not necessary for each player to have a large reservoir S_i. We propose adding the following one-time preprocessing step to Algorithm 6: set each $|S_i| = \lceil \frac{m_{\text{boost}}}{k} \rceil$. By initializing each reservoir in this way, we restrict the sample size to $\sum_{i=1}^{k} |S_i| = m_{\text{boost}}$. By assumption, the players have access to all parameters such as ϵ, δ and k, so they can compute $\lceil \frac{m_{\text{boost}}}{k} \rceil$ and initialize their sets S_i locally.

Theorem 44. *The sample complexity of distributed boosting, Algorithm 6, is*

$$O\left(\frac{d_{boost}}{\epsilon}\right) = \tilde{O}\left(\frac{d}{\epsilon}\right).$$

Proof. The sample complexity of distributed boosting, Algorithm 6, with m_{boost} samples follow from the fact that distributed boosting is equivalent to boosting with a single player with sample $S = \cup_{i=1}^{k} S_i$, and in this case, we know the sample complexity is $\tilde{O}(\frac{d}{\epsilon})$ by Theorem 43. By adding the preprocessing step described above, we restrict the sample complexity of the algorithm to m_{boost}. We now show that distributing m_{boost} points evenly across players as prescribed by the preprocessing step, Algorithm 6 remains correct. The Pre-sampling step remains unaffected. However, in the Sampling step, it is possible that the center requests more points from a player than the player has in their now limited reservoir S_i. This is not a problem as the player simply samples from S_i i.i.d. according to their internal weights, $v_{i,j,t}$. The Weak Learning step is unaffected since it is still receiving a sample drawn i.i.d. from the boosting-weighted mixture of players. And finally, we note that the Updating Sample Weights step is unaffected. Therefore, $\tilde{O}(\frac{d}{\epsilon})$ samples suffice for distributed boosting and adding the preprocessing step to Algorithm 6 achieves the sample complexity. □

In this section, we reviewed the communication complexity of distributed boosting, demonstrating $O(\log(\frac{1}{\epsilon}))$ dependence, and proved the sample complexity of distributed boosting. We use these results in the next section in proving the correctness, sample complexity, and communication complexity of our communication-efficient personalized learning algorithm.

4.8 Communication Efficient Personalized Learning Algorithm

To improve the communication cost of personalized learning, we propose replacing the first step with distributed boosting (Algorithm 6), while leaving the rest of the personalized algorithm intact, as shown in Algorithm 7. We first compute the sample complexity of our proposed algorithm showing that it is equal, up to polylogarithmic terms, to the sample complexity of the original personalized learning algorithm.

Theorem 45. *The sample complexity of communication-efficient personalized learning with boosting (Algorithm 7) is*

$$\tilde{O}\left(\log(k)\frac{d}{\epsilon}\right)$$

when $k\ln(k) = O(d)$.

Proof. By Theorem 44, the sample complexity of distributed boosting is $\tilde{O}(\frac{d}{\epsilon})$. This step is implemented at most $O(\log(k))$ times, totaling $\tilde{O}(\log(k)\frac{d}{\epsilon})$ samples. The `TEST` step is unchanged, so from [48], it uses $O(\log(k)\frac{k}{\epsilon}\ln(\frac{k\log(k)}{\epsilon})) = \tilde{O}(\log(k)\frac{k}{\epsilon})$ samples. Since $k\ln(k) = O(d)$, we have $\tilde{O}(\log(k)\frac{d}{\epsilon})$. □

We now compute the communication complexity of our proposed algorithm, showing that it is indeed an improvement over the original personalized learning.

Theorem 46. *The communication complexity of Algorithm 7 is*

$$\tilde{O}\left(\log(k)\left(d\log\left(\frac{1}{\epsilon}\right)\right)\right)$$

Algorithm 7: Communication Efficient Personalized Learning with Boosting

Input: H, k distributions $D_i \sim X$, $\delta' = \delta/2\log(k)$, $\epsilon > 0$

Output: $f_1, ..., f_k$

Let $N_1 = \{1, ..., k\}$;

for $j = 1, ..., \lceil \log(k) \rceil$ **do**

Run distributed boosting (Algorithm 6) with players in N_j to get candidate h_j;

$G_j \leftarrow \texttt{TEST}(h_j, N_j, \epsilon, \delta')$;

$N_{j+1} = N_j \setminus G_j$;

for $i \in G_j$ **do**

$f_i \leftarrow h_j$;

end

end

return $f_1, ..., f_k$

Procedure $\texttt{TEST}(h, N, \epsilon, \delta)$

for $i \in N$ **do**

Draw sample of size $T_i = O\left(\frac{\ln(\frac{|N|}{\epsilon\delta})}{\epsilon}\right)$ from D_i;

end

return $\{i \mid err_{T_i}(h) \leq \frac{3\epsilon}{4}\}$

samples plus an additional $\tilde{O}(k\log(d)\log(\frac{1}{\epsilon}))$ *bits of communication.*

Proof. We consider a single round of our algorithm. The communication complexity of the first step is given in Theorem 40 as $\tilde{O}(d\log(\frac{1}{\epsilon}))$ examples plus $\tilde{O}(k\log(d)\log(\frac{1}{\epsilon}))$ bits of communication. Recall that each step in distributed boosting, all players learn the same weak learning classifiers locally. Therefore, when the distributed boosting algorithm completes, each player has all $\log(k)$ weak classifiers and can therefore sum them to create the final boosting classifier h_j, costing no communication. Using the boosting classifier in the `TEST` step, there is no communication needed as the players simply need to test the boosting classifier on T_j samples drawn from their own distributions. The players each send one bit of communication to the center indicating if they passed `TEST` or not, costing $O(k)$ bits for all k players. Therefore the total communication complexity over $\log(k)$ rounds is $\tilde{O}(\log(k)(d\log(\frac{1}{\epsilon}))$ samples plus $\tilde{O}(k\log(d)\log(\frac{1}{\epsilon})) + O(k) = \tilde{O}(k\log(d)\log(\frac{1}{\epsilon}))$ additional bits of communication. □

The sample and communication complexities are summarized in Table I. We have shown that by replacing the first step of personalized collaborative learning with distributed boosting, we can improve the number of samples communication exponentially in $\frac{1}{\epsilon}$, which is particularly valuable for collaboratively learning highly accurate classifiers.

4.9 Communication Efficient Personalized Learning with Classification Noise

Thus far in our study of communication efficient collaborative learning we have considered noiseless personalized learning. We now revisit personalized collaborative learning with CN. We propose a communication-efficient variant of our algorithm for personalized learning with CN described in Algorithm 8. The baseline approach is for each player to draw $m_{\epsilon,\delta,\eta_i}$ samples

from their own distributions and learn a classifier locally. The sample sample complexity of this baseline is $O\left(k\frac{d}{\epsilon(1-2\eta_{\text{MAX}})^2}\right)$ and requires no communication.

We revisit our personalized collaborative learning algorithm robust to CN, Algorithm 4, and compute its communication complexity.

Theorem 47. *The communication complexity of personalized learning with CN, Algorithm 4, is*

$$\tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{MAX})^2}\right)$$

samples and $\tilde{O}(k)$ additional bits of communication.

Proof. In the first step, a total of $O\left(\frac{d\log(\log(k))}{\epsilon(1-2\bar{\eta}_K)^2}\right)$ samples are communicated to the center and simultaneously broadcasted to all players so that each player can learn an ERM. Each player than implements the `CN-TEST` locally, costing no communication. However, after implementing `CN-TEST`, the players must send one bit to the center indicating the results of `CN-TEST`, costing $O(k)$ bits. Therefore, over $O(\log(k))$ rounds, the communication complexity is $O\left(\log(k)\frac{d\log(\log(k))}{\epsilon(1-2\bar{\eta}_K)^2}\right) = \tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{\text{MAX}})^2}\right)$ and an additional $O(k\log(k)) = \tilde{O}(k)$ bits of communication. □

Simplifying the sample complexity of personalized learning with CN in Theorem 33 with respect to constant δ and $k\ln(k) = O(d)$ gives $\tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{\text{MAX}})^2}\right)$. Table II summarizes the sample and communication complexities of the baseline approach, personalized learning with CN (Algorithm 4), and in bold, our communication efficient personalized learning algorithm with CN (Algorithm 8). As in the previous section, we consider the scenario where players want

to learn highly accurate classifiers, thus our goal is to develop an algorithm that can improve dependence on $\frac{1}{\epsilon(1-2\eta_{\text{MAX}})}$ in samples communicated.

TABLE II: Samples and Communication in Personalized Learning with Classification Noise

	Sample Complexity	**Samples Communicated**	**Bits Communicated**
Baseline	$\tilde{O}\left(k\frac{d}{\epsilon(1-2\eta_{\text{MAX}})^2}\right)$	$\tilde{O}(1)$	$\tilde{O}(1)$
Algorithm 4	$\tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{\text{MAX}})^2}\right)$	$\tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{\text{MAX}})^2}\right)$	$\tilde{O}(k)$
Algorithm 8	$\tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{\text{MAX}})^2}\right)$	$\tilde{O}\left(\log(k)d\log\left(\frac{1}{\epsilon(1-2\eta_{\text{MAX}})}\right)\right)$	$\tilde{O}\left(kd\log^3\left(\frac{1}{\epsilon(1-2\eta_{\text{MAX}})}\right)\right)$

As in the previous section, we use a boosting approach to improve communication complexity. However, there are some subtleties around boosting in the CN setting. We restrict our attention to boosting the error up to the noise rate η_{MAX}, where $\eta_{\text{MAX}} \leq \epsilon$. It has been shown that boosting past the error rate, where $\eta_{\text{MAX}} > \epsilon$, is hard [54]. By restricting to our attention to boosting up to the noise rate, we can take advantage of agnostic boosting algorithms since CN is a special case of agnostic learning. Therefore, we propose replacing the first step of our personalized learning with CN algorithm (Algorithm 4) with distributed agnostic boosting algorithm in [53], described in Algorithm 8. The distributed agnostic boosting algorithm assumes access to a β-weak agnostic learner, which returns a hypothesis h so that $err_D(h) \leq \min_{h' \in H} err(h') + \beta$ [53]. We recall the sample and communication complexities of distributed agnostic boosting.

Algorithm 8: Communication Efficient Personalized Learning with Agnostic Boosting

Input: H, k distributions $D_i \sim X$ with error rates $\eta_i < \frac{1}{2}$, $\delta' = \delta/2\log(k)$, $\epsilon > 0$

Output: $f_1, ..., f_k \in H$

Let $N_1 = \{1, ..., k\}$;

for $j = 1, ..., \lceil \log(k) \rceil$ **do**

  Use distributed agnostic learning ([53]) on players in N_j to learn h_j;

  $G_j \leftarrow$ `CN-TEST`$(h_j, N_j, \epsilon, \delta')$;

  $N_{j+1} = N_j \setminus G_j$;

  for $i \in G_j$ **do**

    $f_i \leftarrow h_j$;

  end

end

return $f_1, ..., f_k$

Procedure `CN-TEST`(h, N, ϵ, δ)

  for $i \in N$ **do**

    Draw sample of size $T_i = O\left(\frac{\ln(\frac{|N|}{\delta})}{\epsilon(1-2\eta_i)}\right)$ from D_i;

  end

  return $\{i \mid err_{T_i}(EX_{\eta_i}, h_j) \leq \eta_i + \frac{3\epsilon}{4}(1-2\eta_i)\}$

Theorem 48 ([53])**.** *Suppose the distributed agnostic boosting algorithm has access to a β-weak agnostic learner. Then, the sample complexity of distributed agnostic boosting is*

$$\tilde{O}\left(\frac{d}{\epsilon^2(1/2-\beta)^2}\right).$$

The following corollary that holds for the special case of classification noise.

Corollary 49. *Suppose the distributed agnostic boosting algorithm has access to a β-weak agnostic learner. Let β be a fixed constant. The sample complexity of boosting in the presence of classification noise where $\eta_{MAX} \leq \epsilon$ is*

$$\tilde{O}\left(\frac{d}{\epsilon(1-2\eta_{MAX})^2}\right).$$

Proof. Using the standard agnostic learning to classification noise restriction ([55]), the sample complexity for agnostic distributed boosting is $\tilde{O}\left(\frac{d}{\epsilon^2(1-2\eta_{\mathrm{MAX}})^2}\right)$. We then reduce ϵ^2 to ϵ by the argument in [10], which holds specifically for the case of classification noise. □

We now review the communication complexity results of agnostic distributed boosting from [53]. In their work, they track the number of words, instead of bits, communicated. They define words as the length of the vector communicated – a length d vector costs $O(d)$ words to communicate.

Theorem 50 ([53])**.** *Suppose the distributed agnostic boosting algorithm has access to a β-weak agnostic learner. The distributed agnostic boosting algorithm achieves error $\frac{2err_D(H)}{1/2-\beta} + \epsilon$ by using at most*

$$O\left(\frac{\log(\frac{1}{\epsilon})}{(1/2-\beta)^2}\right)$$

rounds, each communicating

$$O\left(\frac{d}{\beta}\log\left(\frac{1}{\beta}\right)\right)$$

samples and

$$\tilde{O}\left(kd\log^2\left(\frac{d}{(1/2-\beta)\epsilon}\right)\right)$$

words of communication.

We derive a corollary that holds specifically for the CN setting. We also convert words to bits. To ensure an approximation within poly$(\frac{d}{\epsilon(1-2\eta_{\mathrm{MAX}})})$, we assume that each element in any vector communicated consists of at most $O\left(\log\left(\frac{d}{\epsilon(1-2\eta_{\mathrm{MAX}})}\right)\right)$ bits.

Corollary 51. *Suppose the distributed agnostic boosting algorithm has access to a β-weak agnostic learner. Let β be a fixed constant. The communication complexity of boosting in the presence of classification noise where $\eta_{MAX} \leq \epsilon$ consists of*

$$O\left(\log\left(\frac{1}{\epsilon(1-2\eta_{MAX})}\right)\right)$$

rounds, each communicating $O(d)$ samples and

$$\tilde{O}\left(kd\log^3\left(\frac{1}{\epsilon(1-2\eta_{MAX})}\right)\right)$$

bits of communication.

Combining the corollaries above with analysis from previous sections, we have the following sample and communication complexities of our algorithm for communication efficient personalized learning with CN (Algorithm 8).

Theorem 52. *The sample complexity of communication efficient personalized learning with noise (Algorithm 8) is*

$$\tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{MAX})^2}\right).$$

Theorem 53. *The communication complexity of communication efficient personalized learning with noise (Algorithm 8) is*

$$\tilde{O}\left(\log(k)\frac{d}{\epsilon(1-2\eta_{MAX})^2}\right)$$

plus $\tilde{O}\left(kd\log^3\left(\frac{1}{\epsilon(1-2\eta_{MAX})}\right)\right)$ *bits of communication.*

APPENDICES

Appendix A

SOURCE CODE FOR CHAPTER 2

```
import networkx as nx
import itertools as it
import random

def powerset(s):
    #CREDIT: This powerset method was taken from stackoverflow, contributed by
    user hughdbrown, source: https://stackoverflow.com/a/1482320
    x = len(s)
    masks = [1 << i for i in range(x)]
    for i in range(1 << x):
        yield [ss for mask, ss in zip(masks, s) if i & mask]

def compute_obj(mu, p, graph_init, R, L):
    '''
    INPUT: any bipartite graph, mu, p, R = set of right nodes, L = set of left
    nodes
    OUTPUT: expected fraction of infected nodes
    '''
    #generate the edges of all subgraphs possible
    #start = time.time()
    subgraphs_powerset = list(powerset(graph_init.edges()))
```

Appendix A (Continued)

```
    subgraphs = [0]*len(subgraphs_powerset)
    original_num_edges = len(graph_init.edges())

    #initialize stuff
    subgraphs_with_probs = {}
    nodes_and_connected_components = {}
    for node in graph_init.nodes():
        nodes_and_connected_components[node] = [0]*(len(graph_init.nodes()))

    #for each edgeset, generate a graph, compute the probability of obtaining
    that graph, and compute the
    #size of the connected components for each vertex of the graph and add the
    graph probability to the
    #appropriate count
    new_graph = nx.Graph()
    for i in range(len(subgraphs_powerset)):
        new_graph.add_nodes_from(R, bipartite = 0)
        new_graph.add_nodes_from(L, bipartite = 1)
        new_graph.add_edges_from(subgraphs_powerset[i])
        conn_comp_size = nx.node_connected_component
        subgraphs_probs = p**(len(new_graph.edges()))*(1-p)**(
    original_num_edges - len(new_graph.edges()))
        for node in graph_init.nodes():
            size = len(conn_comp_size(new_graph, node))
            nodes_and_connected_components[node][size-1] += subgraphs_probs
        new_graph.remove_edges_from(subgraphs_powerset[i])
```

Appendix A (Continued)

```
43
44      #compute the sum of the expected value over all nodes
45      expval = 0
46      for node in graph_init.nodes():
47          sum_over_cv = 0
48          for component_size in range(len(nodes_and_connected_components[node])):
49              sum_over_cv += ((1 - mu)**(component_size+1))*
    nodes_and_connected_components[node][component_size]
50          expval += sum_over_cv
51
52      #compute the objective function = expected fraction of infected nodes
53      obj = 1 - ((expval)/(len(graph_init.nodes())))
54
55      #return result
56      return obj
57
58
59  def get_all_min_indices(list_of_floats):
60      '''Returns a list of all indices that have the lowest value; once indices
    are found, can be used to find
61      corresponding subgraphs that yield min expected fraction of infection.'''
62      smallest = min(list_of_floats)
63      indices_of_smallest = [index for index in range(len(list_of_floats)) if
    list_of_floats[index] == smallest]
64      return indices_of_smallest
65
```

Appendix A (Continued)

```
66
67 def gen_2_half_regular():
68     '''Generates all 2-half-regular bipartite graphs on 8 (4 nodes/side) nodes.
       '''
69     #initialize the complete bipartite graph with 4 nodes per side
70     R = [1, 2, 3, 4]
71     L = ['a', 'b', 'c', 'd']
72     E = [(1,'a'), (1, 'b'), (1, 'c'), (1, 'd'), (2, 'a'), (2, 'b'), (2, 'c'),
       (2, 'd'), (3, 'a'),
73          (3, 'b'), (3, 'c'), (3, 'd'), (4, 'a'), (4, 'b'), (4, 'c'), (4, 'd')]
74     graph_init = nx.Graph()
75     graph_init.add_nodes_from(R, bipartite = 0)
76     graph_init.add_nodes_from(L, bipartite = 1)
77     graph_init.add_edges_from(E)
78
79     two_half_regulars = []
80     append = two_half_regulars.append
81     R_nodes_and_edges = {}
82
83     for i in range(len(R)):
84         R_nodes_and_edges[i+1] = []
85
86     for edge in E:
87         r, l = edge
88         R_nodes_and_edges[r].append(l)
89
```

Appendix A (Continued)

```
90  #generate all possible 2-half-regular graph edge sets: to generate one
    graph, pick TWO edges from each of the
91  #four nodes in R, repeat for all possibilites, i.e. (4 choose 2)^4 = 1296
    graphs
92  edges_2_half_regular = list(it.product(it.combinations(R_nodes_and_edges
    [1], 2), it.combinations(R_nodes_and_edges[2], 2),
93                                          it.combinations(R_nodes_and_edges
    [3], 2), it.combinations(R_nodes_and_edges[4], 2)))
94
95  for i in range(len(edges_2_half_regular)):
96      edges_2_half_regular[i] = list(edges_2_half_regular[i])
97      for j in range(len(edges_2_half_regular[i])):
98          edges_2_half_regular[i][j] = list(edges_2_half_regular[i][j])
99
100 #initialize a bipartite graph object for each graph generated above
101 newgraph = nx.Graph
102 for neighborlist in edges_2_half_regular:
103     E_new_subgraph = []
104     add_on = E_new_subgraph.append
105     for i in range(len(neighborlist)):
106         for j in range(len(neighborlist[i])):
107             add_on((i+1, neighborlist[i][j]))
108
109     graph_new = newgraph()
110     graph_new.add_nodes_from(R, bipartite = 0)
111     graph_new.add_nodes_from(L, bipartite = 1)
```

Appendix A (Continued)

```
112         graph_new.add_edges_from(E_new_subgraph)
113         append(graph_new)
114
115     #return list of 1-half-regular bipartite graph objects with 4 nodes/side
116     return two_half_regulars
117
118
119 #define nodes in R and L
120 R = [1, 2, 3, 4]
121 L = ['a', 'b', 'c', 'd']
122
123 #generate all 2-half-regular graphs and intialize dictionary of all results
124 all_2_half_regulars = gen_2_half_regular()
125 num_2_half_regulars = len(gen_2_half_regular())
126 all_results = {}
127
128
129 n = 10
130 k = 1
131 l = 10
132
133 #for all combinations of mu,p, compute the expected infected fraction of nodes
        explicitly for
134 #each possible 2-half-regular bipartite graph on nodes R, L
135 for mu_times_l in range(0, n, k):
136     all_results[mu_times_l/l] = []
```

Appendix A (Continued)

```
137     for p_times_l in range(0, n, k):
138         iteration = 0
139         print('Starting mu = ', mu_times_l/l, 'p = ', p_times_l/l)
140         p_result = []
141         for graph in all_2_half_regulars:
142             p_result.append(compute_obj(mu_times_l/l, p_times_l/l, graph, R, L)
    )
143             iteration += 1
144         all_results[mu_times_l/10].append(p_result)
145
146
147 all_result_graphs = {}
148
149 #find the smallest expected infected fraction of nodes for each combo of mu and
     p, then find the corresponding
150 #graphs;
151 for mu in all_results.keys():
152     all_result_graphs[mu] = []
153     for p_times_l in range(0, n, k):
154         smallest_obj_graphs_mu = []
155         smallest_indices = get_all_min_indices(all_results[mu][p_times_l])
156         for index in smallest_indices:
157             smallest_obj_graphs_mu.append(all_2_half_regulars[index])
158         all_result_graphs[mu].append(smallest_obj_graphs_mu)
159
160 #print results
```

Appendix A (Continued)

```
161 for mu in all_result_graphs.keys():
162     for p_times_l in range(0, n, k):
163         min_edges = all_result_graphs[mu][p_times_l][0].edges()
164         print('mu = ', mu, 'p = ', p_times_l/l)
165         print(min_edges)
```

Listing A.1: Identifying the 2-Half Regular Bipartite Graphs with Minimal Expected Fraction of Infected Nodes

Appendix B

COPYRIGHT AGREEMENTS

This section contains publication agreements signed by the authors that outline the use policies that allow the original publications to be reproduced in this thesis.

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

On the Resilience of Bipartite Networks
Shelby Heinecke, Will Perkins, Lev Reyzin
2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

You have indicated that you DO wish to have video/audio recordings made of your conference presentation under terms and conditions set forth in "Consent and Release."

CONSENT AND RELEASE

1. In the event the author makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the author, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.
2. In connection with the permission granted in Section 1, the author hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Lev Reyzin	07-10-2018
Signature	**Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by

IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore.

Questions about the submission of the form or manuscript must be sent to the publication's editor.
Please direct all questions about IEEE copyright policy to:
IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966

Appendix B (Continued)

Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

AAAI COPYRIGHT FORM

Title of Article/Paper: Crowdsourced PAC Learning under Classification Noise

Publication in Which Article/Paper Is to Appear: HCOMP-19

Author's Name(s): Shelby Heinecke, Lev Reyzin

Please type or print your name(s) as you wish it (them) to appear in print

PART A – COPYRIGHT TRANSFER FORM

The undersigned, desiring to publish the above article/paper in a publication of the Association for the Advancement of Artificial Intelligence, (AAAI), hereby transfer their copyrights in the above article/paper to the Association for the Advancement of Artificial Intelligence (AAAI), in order to deal with future requests for reprints, translations, anthologies, reproductions, excerpts, and other publications.

This grant will include, without limitation, the entire copyright in the article/paper in all countries of the world, including all renewals, extensions, and reversions thereof, whether such rights current exist or hereafter come into effect, and also the exclusive right to create electronic versions of the article/paper, to the extent that such right is not subsumed under copyright.

The undersigned warrants that they are the sole author and owner of the copyright in the above article/paper, except for those portions shown to be in quotations; that the article/paper is original throughout; and that the undersigned right to make the grants set forth above is complete and unencumbered.

If anyone brings any claim or action alleging facts that, if true, constitute a breach of any of the foregoing warranties, the undersigned will hold harmless and indemnify AAAI, their grantees, their licensees, and their distributors against any liability, whether under judgment, decree, or compromise, and any legal fees and expenses arising out of that claim or actions, and the undersigned will cooperate fully in any defense AAAI may make to such claim or action. Moreover, the undersigned agrees to cooperate in any claim or other action seeking to protect or enforce any right the undersigned has granted to AAAI in the article/paper. If any such claim or action fails because of facts that constitute a breach of any of the foregoing warranties, the undersigned agrees to reimburse whomever brings such claim or action for expenses and attorneys' fees incurred therein.

Returned Rights

In return for these rights, AAAI hereby grants to the above author(s), and the employer(s) for whom the work was performed, royalty-free permission to:

1. Retain all proprietary rights other than copyright (such as patent rights).
2. Personal reuse of all or portions of the above article/paper in other works of their own authorship. This does not include granting third-party requests for reprinting, republishing, or other types of reuse. AAAI must handle all such third-party requests.
3. Reproduce, or have reproduced, the above article/paper for the author's personal use, or for company use provided that AAAI copyright and the source are indicated, and that the copies are not used in a way that implies AAAI endorsement of a product or service of an employer, and that the copies per se are not offered for sale. The foregoing right shall not permit the posting of the article/paper in electronic or digital form on any computer network, except by the author or the author's employer, and then only on the author's or the employer's own web page or ftp site. Such web page or ftp site, in addition to the aforementioned requirements of this Paragraph, shall not post other AAAI copyrighted materials not of the author's or the employer's creation (including tables of contents with links to other papers) without AAAI's written permission.
4. Make limited distribution of all or portions of the above article/paper prior to publication.
5. In the case of work performed under a U.S. Government contract or grant, AAAI recognized that the U.S. Government has royalty-free permission to reproduce all or portions of the above Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract or grant so requires.

In the event the above article/paper is not accepted and published by AAAI, or is withdrawn by the author(s) before acceptance by AAAI, this agreement becomes null and void.

(1) Shelby L Heinecke — 08/22/2019

Author/Authorized Agent for Joint Author's Signature — Date

Employer for whom work was performed — Title (if not author)

(For jointly authored Works, all joint authors should sign unless one of the authors has been duly authorized to act as agent for the others.)

Appendix B (Continued)

Association for the Advancement of Artificial Intelligence

2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

PART B – U.S. GOVERNMENT EMPLOYEE CERTIFICATION

This will certify that all authors of the above article/paper are employees of the U.S. Government and performed this work as part of their employment, and that the article/paper is therefore not subject to U.S. copyright protection. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations.

(2)

_____ _____

U.S. Government Employee Authorized Signature Date

_____ _____

Name of Government Organization Title (if not author)

*(Please read and sign and return Part B **only** if you are a government employee and created your article/paper as part of your employment. If your work was performed under Government contract, but you are not a Government employee, sign only at signature line (1) in Part A and see item 5 under returned rights. Authors who are U.S. government employees should also sign signature line (1) in Part A above to enable AAAI to claim and protect its copyright in international jurisdictions.)*

PART C–CROWN COPYRIGHT CERTIFICATION

This will certify that all authors of the above article/paper are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties , and that the article/paper is therefore subject to Crown Copyright and is not assigned to AAAI as set forth in the first sentence of the Copyright Transfer Section in Part A. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations, and acknowledges that AAAI has the right to publish, distribute, and reprint the Work in all forms and all media.

(3)

_____ _____

British or British Commonwealth Government Employee Authorized Signature Date

_____ _____

Name of Government Organization Title (if not author)

*(Please read and sign and return Part C **only** if you are a British or British Commonwealth Government employee and the Work is subject to Crown Copyright. Authors who are British or British Commonwealth government employees should also sign signature line (1) in Part A above to indicate their acceptance of all terms other than the copyright transfer.)*

CITED LITERATURE

1. Heinecke, S., Perkins, W., and Reyzin, L.: On the resilience of bipartite networks. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* , pages 72–77, 2018.

2. Perkins, W. and Reyzin, L.: On the resilience of bipartite networks. *CoRR* , abs/1306.5720, 2013.

3. Heinecke, S. and Reyzin, L.: Crowdsourced PAC learning under classification noise. *CoRR* , abs/1902.04629, 2019.

4. Mohri, M., Rostamizadeh, A., and Talwalkar, A.: *Foundations of Machine Learning* . The MIT Press, 2012.

5. Anthony, M. and Bartlett, P. L.: *Neural Network Learning: Theoretical Foundations* . Cambridge University Press, 1999.

6. Hanneke, S.: The optimal sample complexity of pac learning. *J. Mach. Learn. Res.* , 17(1):1319?1333, January 2016.

7. Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K.: Learnability and the vapnik-chervonenkis dimension. *J. ACM* , 36(4):929?965, October 1989.

8. Haussler, D., Littlestone, N., and Warmuth, M.: Predicting 0, 1-functions on randomly drawn points. *Information and Computation* , 115(2):248 – 292, 1994.

9. Angluin, D. and Laird, P. D.: Learning from noisy examples. *Machine Learning* , 2(4):343–370, 1987.

10. Laird, P. D.: *Learning from Good and Bad Data* . Norwell, MA, USA, Kluwer Academic Publishers, 1988.

11. Simon, H. U.: General bounds on the number of examples needed for learning probabilistic concepts. *Journal of Computer and System Sciences* , 52(2):239 – 254, 1996.

12. Aslam, J. A. and Decatur, S. E.: On the sample complexity of noise-tolerant learning. *Information Processing Letters* , 57(4):189 – 195, 1996.

13. Nguyen, H. and Zakynthinou, L.: Improved algorithms for collaborative pac learning. In *Advances in Neural Information Processing Systems 31* , eds. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pages 7631–7639. Curran Associates, Inc., 2018.

14. Kremer, M.: Integrating behavioral choice into epidemiological models of aids. *The Quarterly Journal of Economics* , 111(2):549–573, 1996.

15. Blume, L., Easley, D., Kleinberg, J., Kleinberg, R., and Tardos, E.: Network formation in the presence of contagious risk. In *Proc. 12th ACM Conference on Electronic Commerce* , 2011.

16. Blume, L., Easley, D., Kleinberg, J., Kleinberg, R., and Tardos, É.: Which networks are least susceptible to cascading failures? In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on* , pages 393–402. IEEE, 2011.

17. Hota, A. R. and Sundaram, S.: Optimal network topologies for mitigating security and epidemic risks. In *54th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2016, Monticello, IL, USA, September 27-30, 2016* , pages 1129–1136, 2016.

18. Kempe, D., Kleinberg, J., and Tardos, É.: Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* , pages 137–146. ACM, 2003.

19. Alon, N., Benjamini, I., and Stacey, A.: Percolation on finite graphs and isoperimetric inequalities. *Annals of Probability* , pages 1727–1745, 2004.

20. Borgs, C., Chayes, J., Van Der Hofstad, R., Slade, G., and Spencer, J.: Random subgraphs of finite graphs: I. the scaling window under the triangle condition. *Random Structures & Algorithms* , 27(2):137–184, 2005.

21. Nachmias, A.: Mean-field conditions for percolation on finite graphs. *Geometric and Functional Analysis* , 19(4):1171–1194, 2009.

22. Spencer, J. and Wormald, N.: Birth control for giants. *Combinatorica* , 27(5):587–628, 2007.

23. Karp, R. M.: Reducibility among combinatorial problems. In *Complexity of Computer Computations* , pages 85–103, 1972.

24. Feder, T. and Motwani, R.: Clique partitions, graph compression and speeding-up algorithms. *J. Comput. Syst. Sci.* , 51(2):261–272, 1995.

25. Kirkpatrick, D. G. and Hell, P.: On the completeness of a generalized matching problem. In *STOC* , pages 240–245, 1978.

26. Cao, W., Li, J., Tao, Y., and Li, Z.: On top-k selection in multi-armed bandits and hidden bipartite graphs. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* , pages 1036–1044, 2015.

27. Fang, Y., Sun, H., Chen, P., and Huai, J.: On the cost complexity of crowdsourcing. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, , Sweden.* , pages 1531–1537, 2018.

28. Kang, Q. and Tay, W. P.: Task recommendation in crowdsourcing based on learning preferences and reliabilities. *CoRR* , abs/1807.10444, 2018.

29. Li, H., Yu, B., and Zhou, D.: Error rate analysis of labeling by crowdsourcing. In *ICML Workshop: Machine Learning Meets Crowdsourcing. Atalanta, Georgia, USA* , 2013.

30. Wang, W. and Zhou, Z.-H.: Crowdsourcing label quality: a theoretical analysis. *Science China Information Sciences* , 58(11):1–12, 2015.

31. Zhou, Y., Chen, X., and Li, J.: Optimal PAC multiple arm identification with applications to crowdsourcing. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014* , pages 217–225, 2014.

32. Dawid, A. P. and Skene, A. M.: Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics* , pages 20–28, 1979.

33. Sheng, V., Zhang, J., Gu, B., and Wu, X.: Majority voting and pairing with multiple noisy labeling. *IEEE Transactions on Knowledge and Data Engineering* , PP:1–1, 01 2017.

34. Awasthi, P., Blum, A., Haghtalab, N., and Mansour, Y.: Efficient PAC learning from the crowd. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017* , pages 127–150, 2017.

35. Wang, L. and Zhou, Z.-H.: Cost-saving effect of crowdsourcing learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* , IJCAI'16, pages 2111–2117. AAAI Press, 2016.

36. Zhang, H. and Conitzer, V.: A pac framework for aggregating agents' judgements. In *Proceedings of the 2019 AAAI Conference on Artificial Intelligence, Honolulu, Hawaii, USA, January 27 - February 2, 2019* , 2019.

37. Even-Dar, E., Mannor, S., and Mansour, Y.: Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research* , 7:1079–1105, 2006.

38. Jiang, H., Li, J., and Qiao, M.: Practical algorithms for best-k identification in multi-armed bandits. *CoRR* , abs/1705.06894, 2017.

39. Kalyanakrishnan, S. and Stone, P.: Efficient selection of multiple bandit arms: Theory and practice. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel* , pages 511–518, 2010.

40. Mannor, S. and Tsitsiklis, J. N.: The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research* , 5:623–648, 2004.

41. Rangi, A. and Franceschetti, M.: Multi-armed bandit algorithms for crowdsourcing systems with online estimation of workers' ability. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018* , pages 1345–1352, 2018.

42. Liu, Y. and Liu, M.: An online learning approach to improving the quality of crowdsourcing. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Portland, OR, USA, June 15-19, 2015* , pages 217–230, 2015.

43. Zhang, H., Ma, Y., and Sugiyama, M.: Bandit-based task assignment for heterogeneous crowdsourcing. *Neural Computation* , 27(11):2447–2475, 2015.

44. Ipeirotis, P. G. and Gabrilovich, E.: Quizz: targeted crowdsourcing with a billion (potential) users. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014* , pages 143–154, 2014.

45. Kleinberg, R., Niculescu-Mizil, A., and Sharma, Y.: Regret bounds for sleeping experts and bandits. *Machine Learning* , 80(2-3):245–272, 2010.

46. Blum, A. and Kalai, A.: A note on learning from multiple-instance examples. *Machine Learning* , 30(1):23–29, Jan 1998.

47. Feldman, V., Guruswami, V., Raghavendra, P., and Wu, Y.: Agnostic learning of monomials by halfspaces is hard. *SIAM J. Comput.* , 41(6):1558–1590, 2012.

48. Blum, A., Haghtalab, N., Procaccia, A. D., and Qiao, M.: Collaborative pac learning. In *Advances in Neural Information Processing Systems 30* , eds. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pages 2392–2401. Curran Associates, Inc., 2017.

49. Balcan, M. F., Blum, A., Fine, S., and Mansour, Y.: Distributed learning, communication complexity and privacy. In *Proceedings of the 25th Annual Conference on Learning Theory* , eds. S. Mannor, N. Srebro, and R. C. Williamson, volume 23 of *Proceedings of Machine Learning Research* , pages 26.1–26.22, Edinburgh, Scotland, 25–27 Jun 2012. PMLR.

50. Chen, J., Zhang, Q., and Zhou, Y.: Tight bounds for collaborative pac learning via multiplicative weights. In *Advances in Neural Information Processing Systems 31* , eds. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pages 3598–3607. Curran Associates, Inc., 2018.

51. Qiao, M.: Do outliers ruin collaboration? In *Proceedings of the 35th International Conference on Machine Learning* , eds. J. Dy and A. Krause, volume 80 of *Proceedings of Machine Learning Research* , pages 4180–4187, Stockholmsman, Stockholm Sweden, 10–15 Jul 2018. PMLR.

52. Freund, Y. and Schapire, R. E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* , 55(1):119 – 139, 1997.

53. Chen, S.-T., Balcan, M.-F., and Chau, D. H.: Communication efficient distributed agnostic boosting. In *Proceedings of the 19th International Conference on Artificial Intelli-*

gence and Statistics , eds. A. Gretton and C. C. Robert, volume 51 of *Proceedings of Machine Learning Research* , pages 1299–1307, Cadiz, Spain, 09–11 May 2016. PMLR.

54. Kalai, A. and Servedio, R. A.: Boosting in the presence of noise. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing* , STOC ?03, page 195?205, New York, NY, USA, 2003. Association for Computing Machinery.

55. Jabbari, S., Holte, R. C., and Zilles, S.: Pac-learning with general class noise models. In *Proceedings of the 35th Annual German Conference on Advances in Artificial Intelligence* , KI?12, page 73?84, Berlin, Heidelberg, 2012. Springer-Verlag.

VITA

NAME: Shelby Lynn Heinecke

EDUCATION: Ph.D., Mathematics, University of Illinois at Chicago, Chicago, Illinois, 2020.

M.S., Mathematics, Northwestern University, Evanston, Illinois, 2015.

B.S., Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2013.

PUBLICATIONS: Shelby Heinecke, Will Perkins, and Lev Reyzin. On the Resilience of Bipartite Graphs. *Allerton Conference on Communication, Control, and Computing*, 2018.

Shelby Heinecke and Lev Reyzin. Crowdsourced PAC Learning under Classification Noise. *AAAI Conference on Human Computation and Crowdsourcing*, 2019.