Combinatorial Methods for Learning and Information Theory

by

Thomas Jacob Maranzatto
B.A., New College of Florida, 2020

THESIS

Submitted in 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, 2024

Chicago, Illinois

Defense Committee:
Prof. Lev Reyzin, Chair and Advisor
Prof. Gyorgy Turan
Prof. Dhruv Mubayi
Prof. Jan Verschelde
Prof. Will Perkins, The Georgia Institute of Technology

To Mom and Dad.

ACKNOWLEDGMENT

I've been fortunate to have many people support me academically and socially in my pursuit of this thesis. First, I'd like to thank my advisor Lev Reyzin for his support the past 4 years. His role in shaping the way I think about mathematics and research is profound, and I couldn't have completed this thesis without his guidance. I'd also like to thank Marcus Michelen for taking time to introduce me to fascinating problems and building my probabilistic intuition. Special thanks to my committee, Will Perkins, Gyuri Turan, Dhruv Mubayi, and Jan Verschelde, with whom I've enjoyed great topics courses, independent studies, and discussions.

I have many of my peers to thank. Sayok Chakravarty for great discussions and combinatorial insights in our office, and Karline Dubin for sleepless study sessions and being a great research partner - both are dear friends. The graduate department wouldn't be the same without my good friend Nick Christo's organizational prowess and humor. I have to thank Sam Wallace, Trevor Teolis, and Anish Chedalavada for introducing me to rock climbing, which has greatly improved my well being. I have very fond memories of our time in Utah and drive to the east coast. I'd also like to thank Sam Dodds, Jenny Vaccaro, Amy Pompillo, Clay Mizgerd, and Katie Kruzan for being great friends.

I'm lucky to have Jacob, Jacob, Harrison, and Joe as very close friends outside of the mathematics department. Talking and joking with them are the high points in my week. Our trips across the country are some of my favorite times.

ACKNOWLEDGMENT (Continued)

My siblings, Hanna and Tyler, have pushed me to be my best in every aspect, and I hope to be a model for them. My Mom Teresa always supported us three: none of us would be where we are without her sacrifices. My dad José taught me how to do good, honest work from a young age, and has more grit than anyone I know.

Finally, I'd like to thank my Fiancé Diana, for supporting me since New College. I can't put into words what her relationship means to me; I love her dearly.

TJM

CONTRIBUTIONS OF AUTHORS

- Chapter 2 represents the paper *Age of Gossip in Random and Bipartite Networks* by Thomas Maranzatto (1). The content in section 2.7 was done after the submission of the paper, jointly with Marcus Michelen.
- Chapter 3 represents the paper *Tree Trace Reconstruction - Reductions to String Trace Reconstruction* by Thomas Maranzatto (2). The results in sections 3.3.2 through 3.4.3 appeared in my undergraduate thesis, and are included as motivation and background.
- Chapter 4 represents the paper *A Unified Analysis of Dynamic Interactive Learning* by Xing Gao, Thomas Maranzatto, and Lev Reyzin (3). Xing Gao contributed Sections 4.3 and 4.4, and proved Lemma 11. I contributed 4.3.1 and 4.3.2. The remaining results including introduction, formulation of definitions, background, literature review, algorithms and theorems were done jointly with the coauthors.
- Chapter 5 represents joint work with Karoline Dubin, Marcus Michelen, and Lev Reyzin. All content including introduction, formulation of definitions, theorems, and algorithms were done jointly with the coauthors.

TABLE OF CONTENTS

CHAPTER			PAGE
1	**INTRODUCTION**		1
	1.1	Introduction	1
	1.2	Notation	2
	1.2.1	Data Structures	2
	1.2.2	Glossary of Graph Notation	3
	1.2.3	Asymptotic Notation	5
	1.3	Background	5
2	**AGE OF INFORMATION IN GOSSIPING NETWORKS**		10
	2.1	Introduction	10
	2.2	System Model and Background	11
	2.2.1	Version Age of Information	11
	2.2.2	Random Graphs	12
	2.3	Notation and Summary of Results	13
	2.4	Bipartite graphs	16
	2.5	Random Regular Graphs	21
	2.6	Erdős-Reyni Random Graphs	23
	2.7	General Bound on Version Age	28
	2.7.1	Lower Bound	28
	2.7.2	Upper Bound	30
	2.7.3	The Upper Bound is tight: Δ-regular tree	32
	2.7.4	Application of Theorem 4 to Open Problems	34
	2.8	Remarks	35
	2.9	Monotonicity of vAoI	36
	2.10	Proof of Lemma 3	38
3	**STRING AND TREE RECONSTRUCTION**		40
	3.1	Introduction	40
	3.1.1	String Trace Reconstruction	40
	3.1.2	Tree Trace Reconstruction	41
	3.2	Related Work	43
	3.3	Tree Reconstruction Lower Bounds	44
	3.3.1	Recovering Unknown Tree Topologies	44
	3.3.2	Left-Propagation	47
	3.4	Tree Reconstruction Upper Bounds	49
	3.4.1	Labelled Trees with known Topology	49
	3.4.2	Trees with Large Degree	50

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

	3.4.3	Trees with Leaf Labels	51
	3.5	Combinatorics of String Reconstruction	52
	3.5.1	Infinite Strings	53
	3.5.2	Fixed Length Traces	56
4	**DYNAMIC INTERACTIVE LEARNING**		60
	4.1	Introduction	60
	4.2	Preliminaries	61
	4.2.1	Static model	61
	4.2.2	Dynamic model	63
	4.2.2.1	Shifting target	64
	4.2.2.2	Drifting target	65
	4.3	A unified model	66
	4.3.1	Shortest path	71
	4.3.2	m-Neighborhood	72
	4.4	Query complexity lower bound	73
	4.5	Efficient algorithm for low diameter graphs	75
	4.5.1	Cliques: graphs with diameter 1	76
	4.5.2	Stars: graphs with diameter 2	78
	4.5.3	Graphs with diameter o(log n)	80
	4.5.4	Paths: graphs with diameter $\mathtt{n}$	84
	4.6	Acknowledgements	85
	4.7	Proof of Lemma 11	85
	4.8	Quasi-stars: graphs with diameter $\mathtt{d}$	87
5	**LEARNING AUTOMATA FROM RANDOM WALKS**		89
	5.1	Introduction	89
	5.2	Notation and Problem Statement	91
	5.2.1	Problem Statement	91
	5.3	Learning with large transition probabilities	93
	5.4	Learning with Small Transition Probabilities: Random DFA	96
	5.4.1	Identifying 0-Cycles	97
	5.4.2	Learning Random DFA in Stages	102
	5.4.2.1	Stage 1	106
	5.4.2.2	Stage 2 and Beyond	109
	CITED LITERATURE		111

LIST OF FIGURES

FIGURE		PAGE
1	The generic picture before (left) and after (right) node w is removed from tree R in the TED model. The subtrees 3 and 4 are inserted as children of 1 when w is removed.	42
2	Example configuration of a leaf node $\mathfrak{a}$ and the leaves nearest it. The survival of $\mathfrak{a}$ is dependent on the nodes in its 2-neighborhood.	46
3	Trees P_n and R_n described above	47
4	The Markov chain under consideration.	56

SUMMARY

Combinatorial structures are ubiquitous in Computer Science - graphs, trees, strings, and automata are all used in various subfields as modeling tools and objects of study in their own right. Networks can represent connections between wireless routers, or relations between data in a server. Trees are a natural way of encoding data at a high level, and strings are the low-level equivalent stored in physical memory. Automata are a natural definition of limited computational devices.

This thesis is concerned with these structures in a learning-theoretic setting. We will study four notions of learning. In Chapter 2, we analyze a message passing protocol over graphs, where nodes in the graph are attempting to track the data leaving a source. In this setting, the data at the source is the learned quantity. In chapter 3, we study the sample complexity of recovering arbitrary strings and trees passed through a lossy channel. Here, the the original string/tree is the object to be learned. In chapter 4 we study a search game on a graph, where one player moves a token between vertices, and the other must learn the location of this token using limited feedback. Finally, in Chapter 5 we consider a learning protocol over deterministic finite automata, where the learner is given access only to bits generated from a random walk on the automata, and must learn the structure from these bits.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Given access to some noisy source of data, it is a ubiquitous problem in the natural sciences, statistics, and machine learning to produce a model that describes the source. This thesis is concerned with the general theme of reconstruction and recovery problems when the source is a combinatorial structure. Defined informally and broadly, given a combinatorial structure $\mathcal{A}$ (e.g. strings, trees, graphs...) and some property $\mathcal{P}$ of that structure, our goal is to either approximately or exactly produce $\mathcal{P}$ given noisy access to $\mathcal{A}$. This informal definition applies to a host of real-world problems. For instance MRI scanners take 2-dimensional pictures of the body, and an algorithm then has to stitch these together to form a 3-D object. Another example is designing DNA sequencing algorithms that output a gene given many noisy fragments of that gene. A third is observing traffic patterns and producing a schedule for buses or ride-sharing services. The access to $\mathcal{A}$ is highly problem dependent as seen by these examples. Some cases of access to $\mathcal{A}$ could for instance be a time-stamped packet, a random sub-structure, an edge in a graph, or even just a single bit of information. In fact, these last four examples parallel the four chapters of this thesis.

Chapter 2 focuses on a simple distributed algorithm for a network of users tracking a time-varying source; the users wish to recover the source at any time by some communication

protocol. An application to this is tracking a repository on GitHub where updates are made at a rate much faster than users can access, and share information with each other on their knowledge of the repository. Chapter 3 considers the sample complexity for recovering binary strings and trees when randomly chosen sub-strings or sub-graphs are the only data provided to the algorithm. This has immediate applications to the DNA sequencing problem referenced above. Chapter 4 concerns a search problem on graphs where an algorithm attempts to locate a moving target node by querying vertices - the only feedback given to the algorithm is an edge on the shortest path from the query to the target. This is an abstract model for recommender systems, and this chapter gives upper and lower bounds for the number of queries needed to 'recommend' the right service or product. Finally, chapter 5 discusses a problem for learning deterministic finite automata (a type of directed graph) given only local information of the graph from a random walk. This has applications to robotics and environment learning.

Each of these chapters is self contained, and the purpose of the rest of this section is to set up notation, review background, and discuss some related work.

1.2 Notation

1.2.1 Data Structures

Throughout we will be dealing primarily with the structures graphs, trees, and strings. A **graph** G is a collection of vertices V along with edges E, where an edge is an ordered pair of vertices (u, v). Sometimes the edges have associated weights, and so we can define a function $w : E \to \mathbb{R}$ that assigns weights to edges, and will write $G = (V, E, w)$ for the weighted graph. We say a weighted graph is **undirected** if $(u, v) \in E \implies (v, u) \in E$ and $w(u, v) = w(v, u)$.

The distance between two vertices is length of the shortest path connecting u and v and is denoted $\text{dist}_G(u, v)$

A **cycle** in a graph is a collection of vertices $v_1, ..v_k$ such that the edges $\{(v_i, v_{i+1})\}_1^k$ exist in G, where the sum is $\mod k$. A graph is **connected** if there is a path between any two vertices. A **tree** is a connected undirected graph with no cycles. A **rooted** tree is a tree with a distinguished vertex r. All other vertices are said to be **descendants** of r. v is a **child** of u if (u, v) exists and $\text{dist}\, G(r, u) < \text{dist}\, G(r, v)$. Two nodes are **siblings** if they are children of the same node.

A **string** s over alphabet Ω with length k is an ordered tuple $s = (\omega_1, \omega_2, ..., \omega_k)$ with $\omega_i \in \Omega$. We write $s \in \Omega^k$ to mean s is an ordered tuple of length k where each symbol is from Ω. We write $s \in \Omega^*$ to mean s has arbitrary finite length. We can allow infinite strings and use the notation $s \in \Omega^\infty$ where $\Omega^\infty := \Omega^{\mathbb{N}}$. A **subsequence** s' of s is an ordered collection of symbols from s taken in order. For ordered index set $\mathcal{I} \subset [1, ..., k]$ we can let $s' = s[\mathcal{I}_0, \mathcal{I}_1, ...]$. A **substring** of s is a subsequence where the index set is contiguous.

1.2.2 Glossary of Graph Notation

Graph theory has a zoology of terminology and notation for different graphs. We briefly characterize some of these terms here. For every graph below, V is the set of vertices and we just characterize the edge sets. We start with a few deterministic graphs.

Complete Graph - $K_n = \{(u, v) : u, v \in V\}$ with $|V| = n$

Complete Bipartite Graph - $K_{L,R} = \{(u, v) : u \in V_1 \text{ and } v \in V_2\}$ and $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, $|V_1| = L$, $|V_2| = R$

Star Graph - $S_n = K_{1,n}$

Path Graph - $P_n = \{(i, i+1) : i \in [n]\}$ with $V = [n]$

Cycle Graph - $C_n = \{(i, (i+1) \mod n) : i \in [n]\}$ with $V = [n]$

k-regular Graph - Graph where all vertices have exactly k neighbors.

k-ary tree - Rooted tree where each vertex has at most k children

k-regular tree - k-ary tree which is also k-regular

There are three classes of random graphs this thesis discusses. The first, and most classical, is the **Erdős-Reyni random graph** model. Here every possible edge (u, v) is included with probability p independent of all other edges. This induces a measure on the set of all n-vertex graphs, which is denoted $G(n, p)$. The second type we consider is the **random regular graph**. In this model, the measure is uniform on all d-regular graphs, and is denoted $G(n, d)$. There is no ambiguity between $G(n, p)$ and $G(n, d)$ as $p \in [0, 1]$ but $d \in \mathbb{N}$. The final class of graphs are the **d-dimensional random geometric graph**, which are briefly discussed in Section 2.7.4. Now vertices are points in $\mathbb{R}^d$ distributed uniformly at random. Two vertices are neighbors if they have Euclidean distance at most γ, and this measure is denoted $G^d(\gamma, n)$.

Finally we introduce some notation for deterministic finite automata (DFA). A DFA D is a a tuple (V, τ, γ, n_0) where V is the set of states, $\tau : V \times \{0, 1\} \to V$ is the transition function, $\gamma : V \to \{+, -\}$ is the accept/reject behavior, and v_0 is the start state. We think of a D as a directed labeled graph where each vertex has out-degree 2. The two edges leaving any vertex are labeled 0 and 1, and each vertex has a label in $\{+, -\}$. For any string $s \in \{0, 1\}^*$, the accept/reject

behavior of D on s is defined by starting at q_0 and following the labeled edges that agree with s in order; the final state reached has some label and this is output. We write $\nu_0 s$ for the symbol output by the above procedure.

1.2.3 Asymptotic Notation

Throughout this thesis we will count the number of operations various algorithms make before they terminate. We use conventional Big-O, and Little-o notations.

Definition 1 (Big-O Notation). *We say that* $f(n) = O(g(n))$ *if there exists* $c_0, n_0 > 0$ *such that for all* $n > n_0$, $f(n) \leq c_0 g(n)$

Definition 2 (Little-o Notation). *We say that* $f(n) = o(g(n))$ *if for every* $\varepsilon > 0$ *there exists* $n_0 > 0$ *such that for all* $n > n_0$, $f(n) \leq \varepsilon g(n)$

Definition 3 (Big-Ω Notation). *We say that* $f(n) = \Omega(g(n))$ *if there exists* $c_0, n_0 > 0$ *such that for all* $n > n_0$, $f(n) \geq c_0 g(n)$

Definition 4 (Little-ω Notation). *We say that* $f(n) = \omega(g(n))$ *if for every* $\varepsilon > 0$ *there exists* $n_0 > 0$ *such that for all* $n > n_0$, $f(n) \geq \varepsilon g(n)$

Definition 5 (Big-Θ Notation). *We say that* $f(n) = \Theta(g(n))$ *if* $f(n) = O(g(n))$ *and* $f(n) = \Omega(g(n))$

1.3 Background

The purpose of this section is to synthesize some concepts in the remaining four chapters, and show how information theory, combinatorics, and learning are used to solve various problems.

We start with random graph theory, which makes appearances in chapters 2 and 5. Broadly, graphs on n vertices are endowed with a probability measure, and the aim is to study typical behavior and deviations of graphs chosen from this measure. A central question in this area is when a graph property holds under different model parameters. A graph property for n-vertex graphs is a function $f : \{0,1\}^{\binom{n}{2}} \to \{\text{True}, \text{False}\}$ which is invariant under permuting the vertex labels. Properties can usually be understood colloquially (eg. connectivity, 3-colorability, triangle-freeness). A nice class of properties are *monotone* graph properties; these are the properties where adding edges cannot make the property go True to False.

The first random graph model was studied by Erdős and Reyni (4), where the measure is uniform on all graphs with exactly M edges. We denote this graph by $\mathcal{G}(n, M)$ A related model was introduced in parallel by Gilbert (5), where edges of the graph are included independently with probability p. We keep our notation from the introduction and denote this $G(n, p)$. These two models are basically equivalent when $M \sim pn$ and $p(1-p)n \to \infty$. Here if Q is a convex graph property ($F \subset G \subset H$) and F and H have Q, then so does G), and almost surely (a.s.) $G(n, p)$ has Q, then so does $\mathcal{G}(n, M)$ (6). Here 'almost surely' means with probability tending to 1 as $n \to \infty$, the rate of convergence is not relevant.

A historic question in random graph theory is to determine when monotone graph properties hold almost surely. First noticed by Erdős and Reyni for the $\mathcal{G}(n, M)$ model and later for $G(n, p)$, these models undergo a *threshold* at some critical edge density; below this density the random graph does not have the property a.s., and above the density it has the property a.s. We briefly note that monotonicity is crucial here, as this phenomena is a consequence of the

monotone version of the Margulis-Russo Formula for monotone Boolean functions. Friedgut and Kalai (7) showed under the extra assumption that the graph property is also *symmetric*, then the property has a *sharp* threshold. We discuss this phenomena a bit more in Chapter 2.

We can discuss properties of random DFA, which appear in Chapter 5. These are related to two other models, random d-regular graphs and random mappings. In both models the random structure is chosen from the uniform distribution on the support; all d-regular graphs or all mappings $f : [n] \to [n]$. It was shown by Bollobas (8) that the random regular graph has great connectivity properties if $d > 2$ (if $d = 2$ the graph is a collection of disjoint cycles). On the other hand, a random DFA is a random out-degree regular graph with $d_{out} = 2$. If we isolate one of the out-edge sets, say the edges labeled with a 0, then the resulting structure is precisely a random mapping, which is a collection of disjoint trees rooted on disjoint cycles. There are various studies looking into the properties of random mappings, such as size of the largest component, number of vertices on cycles, number of vertices on trees and more (9; 10). Unlike the $G(n, p)$ model, there is no parameter p to vary, so these properties hold a.s. for all random mappings. We discuss random mappings and random DFA more in chapter 5.

Somewhat related to random graphs are random walks on graphs. These appear in chapters 2, 4, and 5. For the purposes of this thesis, the underlying graph G will be fixed for the random walk, and a particle will be placed at some starting vertex v_0. For each discrete time $t \in \mathbb{N}$, the particle uniformly chooses a random neighbor of the current vertex and moves there. In chapters 2 and 4 we analyze time the particle takes to first visit a distinguished state s, known as the *hitting time* for s. For gossip networks, this corresponds to the time a data packet takes

to travel from the source to s. For dynamic learning, this corresponds to the time it takes for a simple learning protocol to observe the moving target under noisy observations. Chapter 5 uses the random walk as a source of data collection on a DFA, and we compute upper bounds for the time to hit a state k times. Random walks on graphs are a special type of discrete-time Markov Chain, and we will see a simple example of a Markov Chain in Chapter 3, which we use to generate random binary strings.

We conclude by discussing some background on Learning Theory. Chapter 4 refines the *Multiplicative Weights* (MW) algorithm for Online Learning. In this setting there are an unbounded number of rounds $r_1, r_2, ...r_n, ...$, and a collection of M 'experts' $e_1, ..., e_M$. Our goal is to predict some time-varying Boolean function $f(t)$ at round r_t given advice from each expert during that round. After every round we are told if our prediction was correct or not. We know expert e_j is correct with some fixed probability p_j, so the question is how to appropriately utilize the advice from the experts.

The MW algorithm proceeds as follows. Fix some weight parameter $\varepsilon < 1$. After each round, expert e_j have a weight w_j assigned to him, and our prediction will be the choice with the higher total sum of expert weights. Once we learn what the truth was for the round, we multiply the weights of the correct experts by $1 - \varepsilon$, and the incorrect by ε. Then it can be proven that after T rounds, the number of mistakes $M_T$this algorithm makes is bounded by $M_T \leq 2(1-\varepsilon)\max_i p_i + \frac{2\ln n}{\varepsilon} + o(1)$. This access to experts is a specialization of *query* learning, where the learning algorithm has access to some device to possibly noisy query function. The motivation for Chapter 5 stems from Angluin's (11; 12) results on query learning DFA. Her

L^* algorithm learns using membership and counterexample queries, which we discuss in the introduction of Chapter 5.

CHAPTER 2

AGE OF INFORMATION IN GOSSIPING NETWORKS

2.1 Introduction

We live in a world of connectivity, and randomized communication networks are becoming more ubiquitous. From IoT applications (13) to federated machine learning topologies (14) to connectivity in drone swarms (15), it is clear that analysis of information flow in random networks is needed.

In this chapter we consider the version age of information (vAoI) metric over various gossip network topologies. Herein, a source node observing a process randomly shares its status with a network $G = (V, E)$, and the nodes in the network randomly share (or 'gossip') their statuses amongst each other. A central question in gossip networks is to characterize how out of date the status of the network is compared to the source, and the vAoI is just one among many metrics used to measure 'freshness'. Others include the age of information (16), age of incorrect information (17), and age of synchronization (18).

Yates (19) initialized the study of version age by using formula (2.1) to show that the age of information in the complete gossiping network K_n scales as $\Theta(\log n)$, and the age of gossip in the disconnected network $\overline{K_n}$ scales as $\Theta(n)$. Yates also gave a conjecture that the cycle C_n should have version age $O(\sqrt{n})$ which was subsequently proven by Srivastava and Ulukus (20). Other works have studied the 2-d lattice (21) and generalizations of cycles (20). For an overview

of the problem and many variants, we refer the reader to the recent survey by Kaswan, Mitra, Srivastava, and Ulukus (22).

This chapter investigates the question, 'how does the vAoI evolve as we interpolate between K_n and $\overline{K_n}$'? This is a natural question to ask, and was stated in other works ((21; 22)) and partially answered in the context of generalized cycles (20). We choose to focus on complete bipartite graphs $K_{L,R}$ and random graph models. For bipartite graphs, we investigate how version age changes when we vary the partition sizes. For random graphs, we show a phase transition occurs where the graph moves from rational to logarithmic version age. We also study d-regular graphs and show under the uniform distribution, these almost surely have logarithmic version age. Precise statements of our results are presented in section 2.3.

2.2 System Model and Background

2.2.1 Version Age of Information

We consider a source node n_0 sending updates to a network $G = (V, E)$ over n nodes. We let $V = [1, .., n]$. The source updates itself via a Poisson process with rate λ_e. The source also sends updates to each $v \in G$ as separate Poisson processes with rates $\frac{\lambda}{n}$. In all graphs we consider, an undirected edge $ij \in E$ facilitates two-way communication between nodes i and j, with $\lambda_i(j) = \frac{\lambda}{\deg(i)}$ denoting the Poisson rate from i to j, and $\lambda_j(i) = \frac{\lambda}{\deg(j)}$ denoting the Poisson rate from j to i. In general $\lambda_i(j) \neq \lambda_j(i)$. If node i has no neighbors then for all j, $\lambda_i(j) = 0$.

The source and every node in the network have internal counters; when a node $i \in V \cup \{n_0\}$ communicates to a neighbor j (because i's Poisson process updated) i sends its current counter value. The counter for n_0 increments if and only if the process for n_0 updates. Contrast this

with $j \in V$ whose counter increments if and only if j receives a newer version from one of its neighbors. Let $X_j(t)$ be the number of versions node j is behind n_0 at time t, and for any subset S of vertices let $X_S(t) = \min_{i \in S} X_i(t)$. Then the limiting average version age of S is $\nu_G(S) = \lim_{t \to \infty} \mathbb{E} X_S(t)$.

The information flow from i into set S is denoted by $\lambda_i(S) = \sum_{j \in S} \lambda_i(j) = \frac{\lambda |N(i) \cap S|}{\deg(j)}$. Similarly $\lambda_0(S) = \frac{\lambda |S|}{n}$. Then the main result of Yates (19) is that the stochastic quantity $\nu_G(S)$ can be computed combinatorially by the recursion:

$$\nu_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) \nu_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)} \tag{2.1}$$

In any network where λ and λ_e are constant, $\lambda_0(S) = \frac{\lambda |S|}{n}$, and $\lambda_i(j) = \frac{\lambda}{\deg(i)}$, the version age is monotonic and bounded as $\Omega(\log n) = \nu_G(S) = O(n)$ as shown in Section 2.9. We use this fact numerous times throughout and will attempt to be explicit when it is applied.

2.2.2 Random Graphs

A graph is d-regular if all vertices have degree d. We let $G(n, d)$ denote the uniform probability distribution over all n-node d regular graphs. The Erdős-Reyni model $G(n, p)$ denotes the distribution on n-node graphs where each edge is chosen i.i.d. with probability p. We say a graph property Q holds asymptotically almost surely (a.a.s.) under $G(n, d)$ if $\mathbb{P}[G(n, d) \in Q] \to 1$ as $n \to \infty$, and likewise for $G(n, p)$.

A graph property is called monotone if adding edges to a graph does not change the property and the property is invariant to permuting the labels of vertices (e.g. connectivity is monotone, 3-colorability is not). Recall the definition of a random graph threshold:

Definition 6. *A function* $p^* = p^*(n)$ *is a threshold for graph property* R *in* $G(n,p)$ *if*

$$\lim_{n\to\infty} \mathbb{P}[G(n,p) \in R] = \begin{cases} 0 & \text{if } p/p^* \to 0 \\ 1 & \text{if } p/p^* \to \infty \end{cases}$$

A *sharp threshold* is defined similarly, where the limits to 0 and ∞ can be bounded by $1-\varepsilon$ and $1+\varepsilon$ respectively, for every $\varepsilon > 0$. (See eg. (7) for a more formal definition of sharp thresholds.)

2.3 Notation and Summary of Results

Throughout we assume λ_0 and λ are constants and let $n \to \infty$. All inequalities are meant to hold for n sufficiently large. We say a graph property Q holds asymptotically almost surely (a.a.s.) under $G(n,d)$ if $\mathbb{P}[G(n,d) \in Q] \to 1$ as $n \to \infty$, and likewise for $G(n,p)$. We use $\nu_G(S)$ to denote the version age of $S \subseteq V$, and when there is no ambiguity about the graph the subscript G will be dropped. The graph $K_{L,R}$ is the complete bipartite graph with vertex bipartition $V = L \cup R$ and $E(G) = \{uv : u \in L, v \in R\}$. For any $S \subset V$, the neighborhood of S is $N(S) = \{v \in V : \mathrm{dist}_G(v,S) = 1\}$ and its edge boundary is $\partial S = \{ij \in E : i \in S, j \notin S\}$. Let $B_m(v) := \{u : \mathrm{dist}_G(u,v) \le m\}$ be the ball of radius m about v. Define $m_*(v) := \min\{m :$

$|B_m(v)|m \geq n\}$ and $\Phi_m(v) := \frac{|B_m(v)|}{|B_{m-1}(v)|}$. Finally we use the notation $\tilde{\Theta}(\cdot)$ to suppress logarithmic factors. Our main results are presented below.

Theorem 1. *Let* $L := L(n)$ *and* $R := R(n)$ *be non-decreasing functions such that* $|L| + |R| = n$ *and for all* n, $|L| < |R|$. *For* $K_{L,R}$, *if* $j \in R$ *then,*

1. $L = \Theta(1) \implies \nu(\{j\}) = \Theta(n)$
2. $L = f(n)$ *and* $f(n) = o(n) \implies \nu(\{j\}) = \Omega(n/f(n))$
3. $L = n^\alpha$ *for* $\alpha \in (0,1) \implies \nu(\{j\}) = \tilde{\Theta}(n^{1-\alpha})$
4. $L = \Theta(n) \implies \nu(\{j\}) = \Theta(\log n)$

Theorem 2. *For any fixed* $d \geq 3$, *a.a.s the worst-case version age of any vertex in* $G(n,d)$ *is* $\Theta(\log n)$.

Theorem 3. *Let* $\varepsilon > 0$. *If* $p = \frac{(1-\varepsilon)\log n}{n}$ *then a.a.s. the average version age of a vertex in* $G(n,p)$ *is* $\Omega(n^{\varepsilon - o(1)})$. *If* $p = \frac{(100+\varepsilon)\log n}{n}$ *then the average version age of a vertex is* $\Theta(\log n)$. *Furthermore there are constants* $\alpha > 0$ *and* $1 \leq c^* \leq 100$ *such that* $p = \frac{c^* \log n}{n}$ *is a threshold for the graph property "*G *has average version age less than* $\alpha \log n$*"*

Theorem 4. *There exists constants* C_1, C_2 *such that for any gossip network where* $\lambda_e = \lambda = 1$ *with minimum degree* δ *and maximum degree* Δ, *every vertex* $v \in V$, *the average expected age satisfies*

$$C_1 \min\left\{\frac{\delta}{\Delta}, \frac{1}{\Phi_{m_*}(v)}\right\} \leq \frac{\mathbb{E}X_v(t)}{m_*(v)} \leq C_2\Delta$$

To summarize in words, the worst-case version age in complete bipartite graphs is inversely related to the size of the smaller component. This intuitively makes sense, as $K_{0,n} = \overline{K_n}$ is the empty graph and has linear version age, and the balanced bipartite graph $K_{n/2,n/2}$ is very dense so information should flow readily (and we will show has the same version age scaling as K_n up to constant factors). Theorem 1 makes this intuition precise. Our concern with rational partition sizes in $K_{L,R}$ is to compare these graphs to generalized rings with rational degree as studied in (20): in both cases rational degree distribution leads to rational version age scaling.

Perhaps surprising is the contrast between Theorem 2 and Theorem 3, where d-regular graphs have logarithmic version age and $G(n,p)$ only achieves logarithmic version age for expected degree $\sim \log n$. An explanation for this is that for any fixed d, $G(n,d)$ is a very good expander, but $G(n,p)$ only achieves connectivity once $p = \frac{\log n}{n}$ and so below this point is a worst-case expander. Expansion is a good measure for long-range connectivity in graphs, and connectivity should be a condition for the quick spread of gossip. In fact the proof for Theorem 2 relies heavily on the expansion properties of $G(n,d)$, and likewise the lower bound of Theorem 3 relies on counting isolated vertices in $G(n,p)$.

Theorem 4 gives a general result for the average version age of an arbitrary graph. The value m_* is the combinatorial quantity that controls the version age scaling. Intuitively this can be though of as the time it takes a new packet from the source to land 'close' to vertex v, and then for this packet to be forwarded to v by the gossiping protocol. We use Theorem 4 to re-prove some results above and solve open problems in section 2.7.4

2.4 Bipartite graphs

We prove theorem 1. By symmetry of the complete bipartite graph $K_{L,R}$, the version age of a subset only depends on the number of nodes in the left and right parts. Therefore for any subset $S \subset V$ with $|S \cap L| = i, |S \cap R| = j$ define $\nu(i,j) := \nu_{K_{L,R}}(S)$ and likewise for $\lambda_w(i,j)$. Finally define $u_{K_{L,R}}(i,j) = \frac{\lambda}{\lambda_e}\nu_{K_{L,R}}(i,j)$.

Lemma 1. *Let $K_{L,R}$ be a complete bipartite graph on n vertices. Then for any $S \subset V$ with $S \cap L = i$, $S \cap R = j$,*

$$u_{K_{L,R}(i,j)} = \frac{1 + \frac{(|L|-i)j}{|R|}u(i+1,j) + \frac{(|R|-j)i}{|L|}u(i,j+1)}{\frac{i+j}{n} + \frac{(|L|-i)j}{|R|} + \frac{(|R|-j)i}{|L|}}.$$

Proof. Rearranging equation (2.1) and using $\nu(V) = \frac{\lambda_e}{\lambda}$,

$$\nu(V)\lambda = \lambda_0(S)\nu(S) + \sum_{i \notin S} \lambda_i(S)(\nu(S) - \nu(S \cup \{i\}))$$

Note that $u(S) = \frac{\nu(S)}{\nu(V)}$, and $\lambda_0(S) = \frac{\lambda|S|}{n}$. By symmetry of the network we can split the sum and simplify,

$$\begin{aligned}
1 &= \frac{i+j}{n}u(i,j) + \sum_{(S\cap L)^c} \frac{j}{|R|}(u(i,j) - u(i+1,j)) \\
&\quad + \sum_{(S\cap R)^c} \frac{i}{|L|}(u(i,j) - u(i,j+1)) \\
&= u(i,j)\left(\frac{i+j}{n} + \frac{(|L|-i)j}{|R|} + \frac{(|R|-j)i}{|L|}\right) \\
&\quad - \frac{(|L|-i)j}{|R|}u(i+i,j) - \frac{(|R|-j)}{|L|}u(i,j+1)
\end{aligned}$$

Solving for $u(i,j)$ completes the proof □

Proof. (Theorem 1) We split the proof into four parts corresponding to the four regimes in the Theorem. Since $u(S)$ is a constant multiple of $\nu(S)$, we are content to bound the former. If $L = \Theta(1)$, then there are constants $C_1 < \liminf L(n)$ and $C_2 > \limsup L(n)$. Then for large enough n,

$$\frac{\lambda}{\lambda_e}\nu(0,1) = u(0,1) \geq \frac{1 + \frac{C_1}{n-C_1}u(1,1)}{\frac{1}{n} + \frac{C_2}{n-C_2}} \geq \frac{n-C_2}{C_2+1} = \Theta(n)$$

Where the second inequality uses the fact that for all graphs $u_G(S) > 0$. The upper bound follows since any graph has at most linear version age.

For the second case that $L = f(n) = o(n)$,

$$u(0,1) \geq \frac{1 + \frac{f(n)}{n-f(n)}u(1,1)}{\frac{1}{n} + \frac{f(n)}{n-f(n)}} \geq \frac{n-f(n)}{f(n)+1} = \Omega\left(\frac{n}{f(n)}\right)$$

where the last equality follows from the upper bound on f. For the third case when $L = n^{\alpha}$, we have the lower bound $\Omega(n^{1-\alpha})$ from the above observation. We also have

$$u(0,1) = \frac{1 + \frac{n^{\alpha} u(1,1)}{n - n^{\alpha}}}{\frac{1}{n} + \frac{n^{\alpha}}{n - n^{\alpha}}} = O\left(n^{1-\alpha}\right) + O\left(u(1,1)\right)$$

so it is enough to show $u(1,1) = \tilde{O}\left(n^{1-\alpha}\right)$. To that end we state and prove the following lemma.

Lemma 2. *For any complete bipartite graph* $K_{L,R}$*,*

$$u_{K_{L,R}}(1,1) \leq \min\{|R|(\log(|L|)+1), |L|(\log(|R|)+1)\}.$$

Proof. For clarity we write L and R instead of $|L|$ and $|R|$ and drop the subscript on $u(\cdot,\cdot)$. By Lemma 1

$$\begin{aligned} u(k,1) &\leq \frac{1 + \frac{L-k}{R} u(k+1,1) + \frac{k(R-1)}{L} u(k,2)}{\frac{L-k}{R} + \frac{k(R-1)}{L}} \\ &\leq \frac{1 + \frac{L-k}{R} u(k+1,1) + \frac{k(R-1)}{L} u(k,1)}{\frac{L-k}{R} + \frac{k(R-1)}{L}} \\ &= \frac{RL + L(L-k)u(k+1,1) + kR(R-1)u(k,1)}{L(L-k) + kR(R-1)} \end{aligned}$$

Where the second inequality follows since increasing the size of a set can only decrease its version age. Letting $D = L(L-k) + kR(R-1)$, we have

$$u(k,1)\left(\frac{L(L-k)}{D}\right) \leq \frac{RL}{D} + \frac{L(L-k)}{D}u(k+1,1)$$
$$\implies u(k,1) \leq \frac{R}{L-k} + u(k+1,1)$$

Starting at $u(1,1)$ and applying this inequality recursively L times, followed by expanding the right partition yields $u(1,1) \leq R + \sum_{i=1}^{L} \frac{R}{L-i} \leq R(\log(L)+1)$. An analogous argument holds for the quantity $L(\log(R)+1)$ by instead expanding $u(1,k)$. □

Applying Lemma 2 gives $u(1,1) \leq n^{\alpha}(\log(n-n^{\alpha})+1)$ and completes the proof for the third case.

For the fourth case when $L = \Theta(n)$, similar to case one we could find constants C_1, C_2 so that the tail of $L(n)$ is bounded as $nC_1 \leq L(n) \leq nC_2$. For clarity we are content to let $L = cn$, and $K(n) := K_{cn,(1-c)n}$ for some absolute constant c. We also drop floors and ceilings on cn when this isn't an integer; this obviously doesn't change the asymptotics. We briefly prove the following fact.

Fact 1. $u_{K(n)}(1,2) \leq u_{K(2n)}(1,2)$ *and* $u_{K(n)}(2,1) \leq u_{K(2n)}(2,1)$

Proof. When we move from $K_{cn,(1-c)n}$ to $K_{2cn,2(1-c)n}$, every arc in the larger network has half the capacity of a corresponding arc in the smaller network. Therefore by symmetry of the network

any subset $(A, B) \subset K_{2cn,2(1-c)n}$ will have the same version age scaling as $(A/2, B/2) \subset K_{cn,(1-c)n}$. The result follows from monotonicity (Appendix A, Lemma 5). □

Now there is a $\delta = \delta(c) > 0$, and for any $\varepsilon > 0$ and n large enough,

$$\begin{aligned}
u_{K(n)}(1,1) &= \frac{1 + \frac{nc-1}{n(1-c)} u_{K(n)}(2,1) + \frac{n(1-c)-1}{nc} u_{K(n)}(1,2)}{\frac{2}{n} + \frac{nc-1}{n(1-c)} + \frac{n(1-c)-1}{nc}} \\
&\leq \frac{1}{\frac{1}{n} + \frac{nc-1/2}{(1-c)n} + \frac{n(1-c)-1/2}{nc}} \times \\
&\quad \left(1 + \frac{nc-1}{n(1-c)} u_{K(2n)}(2,1) \right. \\
&\quad \left. + \frac{n(1-c)-1}{nc} u_{K(2n)}(1,2)\right) + \varepsilon \\
&\leq \frac{1}{\frac{1}{n} + \frac{nc-1/2}{(1-c)n} + \frac{n(1-c)-1/2}{nc}} \times \\
&\quad \left(1 + \frac{nc-1/2}{n(1-c)} u_{K(2n)}(2,1) \right. \\
&\quad \left. + \frac{n(1-c)-1/2}{nc} u_{K(2n)}(1,2)\right) + \varepsilon - \delta \\
&= u_{K(2n)}(1,1) + \varepsilon - \delta
\end{aligned}$$

In the first inequality we applied Fact 1 and noted the limits are the same. In the second inequality we used the fact that the version age of a subset in any network is at most linear. Therefore $u_{K(n)}(1,1) \leq u_{K(2n)}(1,1) + \varepsilon - \delta$, setting $\varepsilon < \delta$ implies $u_{K(n)}(1,1) = O(\log n)$. The lower bound follows since the version age of the clique K_n is $\Theta(\log n)$. □

2.5 Random Regular Graphs

The proof for Theorem 2 uses similar techniques to (21) for bounding the age of gossip on a grid. Therein a key part of the argument is understanding expansion properties of the network; how many edges exist in any cut of G. We recall the edge expansion number for a graph (also known as the Cheeger constant, or isoperimetric number).

Definition 7. *For any graph G, the edge expansion number $h(G)$ is given by*

$$h(G) := \min_{|S| \leq n/2} \frac{|\partial S|}{|S|}$$

where ∂S is the set of edges in the cut spanning S and S^c.

Recall that $G(n, d)$ is the uniform probability distribution over all d-regular graphs. A result by Bollobás (8) shows for constant d, $G(n, d)$ generates good edge expanders. We need a weaker version of his result,

Theorem 5. *(Bollobás(8)) For every fixed $d \geq 3$. Then there is a constant $c_d < \frac{1}{2}$ such that*

$$\mathbb{P}[h(G(n, d)) \geq dc_d] \to 1 \text{ as } n \to \infty$$

Proof. (Theorem 2) Recalling identity (4) from (21) which is just a rearrangement of equation (2.1),

$$\lambda_e = \lambda_0(S)\nu(S) + \sum_{i \notin S} \lambda_i(S)(\nu(S) - \nu(S \cup \{i\})) \tag{2.2}$$

Since $G(n, d)$ is regular, we can partition ∂S into sets $A_1, ..., A_d$ where $A_j = \{v \notin S : |N(v) \cap S| = j\}$. Then,

$$\begin{aligned}
\lambda_e &= \lambda_0(S)\nu(S) + \sum_{i=1}^{d} \sum_{j \in A_i} \lambda_j(S)(\nu(S) - \nu(S \cup j)) \\
&\geq \lambda_0(S)\nu(S) + \sum_{i=1}^{d} |A_i| \frac{i\lambda}{d} \min_{j \in A_i} (\nu(S) - \nu(S \cup j)) \\
&\geq \lambda_0(S)\nu(S) + \left(\frac{\lambda}{d} \sum_{i=1}^{d} i|A_i| \right) (\nu(S) - \max_{i \in N(S)} (\nu(S \cup i)))
\end{aligned}$$

Since $\sum_{i=1}^{d} i|A_i| = \partial S$, by Theorem 5, when $|S| < n/2$ a.a.s. $G(n, d)$ satisfies:

$$\begin{aligned}
\lambda_e &\geq \frac{\lambda |S|}{n} \nu(S) + \frac{\lambda}{d} c_d d |S| (\nu(S) - \max_{i \in N(S)} \nu(S \cup i)) \\
\implies \nu(S) &\leq \frac{\frac{\lambda_e}{\lambda} + c_d |S| \max_{i \in N(S)} \nu(S \cup i)}{\frac{|S|}{n} + c_d |S|}
\end{aligned} \tag{2.3}$$

By an analogous argument for when $|S| > n/2$:

$$\nu(S) \leq \frac{\frac{\lambda_e}{\lambda} + c_d (n - |S|) \max_{i \in N(S)} \nu(S \cup i)}{\frac{|S|}{n} + c_d (n - |S|)} \tag{2.4}$$

Therefore when unrolling recursion (2.2) for $\nu(\{i\})$, if $|S| < n/2$ we use inequality (2.3), otherwise we use (2.4). To that end let X be the sum corresponding to small subset size and letting $j := |S|$,

$$\begin{aligned} X &\leq \frac{\lambda_e}{\lambda}\left(\frac{1}{c_d + \frac{1}{n}}\right)\left(1 + \sum_{i=1}^{n/2}\prod_{j=1}^{i}\frac{c_d j}{\frac{j+1}{n} + c_d(j+1)}\right) \\ &\leq \frac{\lambda_e}{c_d\lambda}\left(1 + \sum_{i=1}^{n/2}\prod_{j=1}^{i}\frac{c_d j}{\frac{j+1}{n} + c_d(j+1)}\right) \\ &\leq \frac{\lambda_e}{c_d\lambda}\left(1 + \sum_{i=1}^{n/2}\prod_{j=1}^{i}\frac{j}{j+1}\right) \\ &= \frac{\lambda_e}{c_d\lambda}\left(1 + \sum_{i=1}^{n/2}\frac{1}{i+1}\right) = O(\log n) \end{aligned}$$

Letting Y be the terms corresponding to $|S| > n/2$, it can be shown that

$$\begin{aligned} Y &\leq \frac{\lambda_e}{\lambda} + \frac{\lambda_e}{\lambda}\left(1 + \sum_{i=n/2}^{n-2}\prod_{j=n/2}^{i}\frac{c_d(n-j)}{\frac{j}{n} + c_d(n-j-1)}\right) \\ &\quad \times \prod_{j=1}^{n/2-1}\frac{c_d j}{\frac{j}{n} + c_d(j+1)} \times \frac{1}{1/2 + c_d n/2} \\ &\leq C'\frac{\lambda_e}{\lambda}\log n = O(\log n) \end{aligned}$$

Where we omitted computations that are analogous to those found in the previous step and in (21). Finally noting that for any network $\nu(\{i\}) = \Omega(\log n)$ completes the proof. □

2.6 Erdős-Reyni Random Graphs

We need the following lemmas on properties of $G(n, p)$,

Lemma 3. *The following holds a.a.s.*

1. *If* $d < 1$ *and* $p = d/n$*, then* $G(n,p)$ *has no component of size larger than* $O(\log n)$
2. *If* $1/2 < d < 1$ *and* $p = \frac{d \log n}{n}$ *then* $G(n,p)$ *has* $\Omega(n^{1-d-o(1)})$ *isolated vertices*
3. *If* $d > 100$ *and* $p = \frac{d \log n}{n}$ *then* $G(n,p)$ *is connected and for any* $0 < \varepsilon < 1$*, every* $v \in G(n,p)$ *satisfies* $\deg(v) \in ((1-\varepsilon)np, (1+\varepsilon)np)$

Proof. Items 1 and 3 are easily verified ((23), (24)). We leave item 2 for Section 2.10. □

Lemma 4. *Let* $0 < \delta < 1$ *and* $p \geq \frac{30 \log n}{\delta^2 n}$*. Then a.a.s. all subsets* S *of* $G(n,p)$ *with* $|S| < \frac{n}{2}$ *satisfies*

$$\mathbb{P}\left(|\partial S| \notin (1 \pm \delta)\mathbb{E}[|\partial S|]\right) = o(1)$$

Proof. For any $S \subset V$ where $|S| = k$ and $\alpha = \delta\sqrt{k(n-k)p}$, define $A_S := \mathbb{P}\left[|\partial S| \notin (1 \pm \delta)\mathbb{E}[|\partial S|]\right]$. Since in $G(n,p)$ edges are included i.i.d, the Chernoff bound states

$$\mathbb{P}[A_S] \leq 3\exp\left(\frac{-\alpha^2}{8}\right).$$

then by the choice of p, for n large,

$$\begin{aligned}\frac{9\log\left(n\binom{n}{k}\right)}{\delta^2 k(n-k)} &\leq \frac{9\log(\frac{ne}{k})}{\delta^2(n-k)} + \frac{9\log(n)}{\delta^2(n-1)} \\ &\leq \frac{9(1+\log n)}{\delta^2\frac{n}{2}} + \frac{10\log n}{\delta^2 n} \leq p\end{aligned}$$

So that by the definition of α,

$$\mathbb{P}\left(\bigcup_{\substack{S\subset V\\ |S|\le n/2}} A_S\right) \le 3\sum_{k=1}^{n/2}\binom{n}{k}\exp\left(\frac{-\delta^2 k(n-k)p}{8}\right)$$
$$\le 3\sum_{k=1}^{n/2} o\left(\frac{1}{n}\right) = o(1)$$

□

Proof. (Theorem 3) Let $\varepsilon > 0$ and suppose $p \le \frac{(1-\varepsilon)\log n}{n}$. Then by Lemma 3.2, there are $O(n^{\varepsilon - o(1)})$ isolated vertices. By equation (2.1) any isolated vertex i has version age $\nu(\{i\}) = \frac{\lambda_e n}{\lambda} = \Theta(n)$. For any i that is not isolated,

$$\nu_{G(n,p)}(\{i\}) = \Omega(\log n).$$

Then the average version age of a vertex is greater than $\frac{1}{n}(\Theta\left(n^{1+\varepsilon-o(1)}\right) + \Omega\left((n-n^{\varepsilon})\right)\log n) = \Omega(n^{\varepsilon-o(1)})$, which is a rational function for all $\varepsilon > 0$.

Now let $p \ge \frac{100\log n}{n}$. Define $A_j = \{v \in V(G) : |N(v)\cap S| = j\}$ and $u(S) = \frac{\nu(S)}{\nu(V)}$. Applying Lemma 3.3 with $\varepsilon = 1/2$ and Lemma 4 with $\delta = 1/3$, a.a.s. for every $k = |S| \subset V$ we have

$$\nu(V)\lambda = \lambda_0(S)\nu(S) + \sum_{i\notin S}\lambda_i(S)(\nu(S) - \nu(S\cup\{i\}))$$
$$= \frac{\lambda k}{n}\nu(S) + \sum_{i\notin S}\frac{|N(i)\cap S|}{\deg(i)}(\nu(S) - \nu(S\cup\{i\}))$$

Where the first line is identity (2.2). Then a.a.s.

$$n \geq ku(S) + \frac{2n}{3\log n} \sum_{\ell < 3\log n} A_\ell (u(S) - u(S \cup \{i\})) \tag{2.5}$$

$$\geq ku(S) + \frac{n}{2\log n} \times \frac{k(n-k)\frac{2}{3}\log n}{n}$$

$$\times (u(S) - \max_{i \in N(S)} u(s \cup i)) \tag{2.6}$$

$$\geq ku(S) + \frac{1}{3}k(n-k)(u(S) - \max_{i \in N(S)} u(s \cup i))$$

Where (2.5) uses Lemma 3 and connectivity, and (2.6) uses Lemma 4 and symmetry. Rearranging,

$$u(S) \leq \frac{n + \frac{1}{3}k(n-k)\max u(S \cup \{i\})}{k + \frac{1}{3}k(n-k)} \tag{2.7}$$

Let $P_i = \prod_{j=i}^{n-1} \frac{\frac{1}{3}j(n-j)}{j+\frac{1}{3}j(n-j)}$. Expanding a single vertex $\ell \in V$,

$$u(\{\ell\}) \leq \frac{n}{1+\frac{n-1}{3}} + \frac{\frac{1}{3}(n-1)}{1+\frac{1}{3}(n-1)} + \sum_{i=1}^{n} \frac{n}{(i+1)+\frac{1}{3}(i+1)(n-i-1)} P_i$$

Observing that $P_i = \prod_{j=i}^{n-1} \frac{n-j}{n-j+3} \leq \prod_{j=i}^{n-1} \frac{n-j}{n-j+1} = \frac{1}{n-i+1}$,

$$u(\{\ell\}) \leq C_1 + 3n \sum_{i=1}^{n} \frac{1}{i(n-i+\frac{1}{3})} \tag{2.8}$$

Where the $\frac{1}{3}$ is present to avoid diving by zero and facilitate the next step. Briefly recall some properties of the Digamma function (25): for every complex z that is not a negative integer,

the Digamma function has series representation $\psi(1+z) = -\gamma + \sum_{k=1}^{\infty} \frac{z}{k(k+z)}$, where γ is the Euler–Mascheroni constant. Also $\psi(1-x) = \psi(x) + \pi\cot(\pi x)$ and $\psi(1+x) \sim \log x$ if $x \in (0,\infty)$. Therefore (2.8) is bounded as,

$$\begin{aligned} u(\{\ell\}) &\leq C_1 + \sum_{i=1}^{\infty} \frac{3n}{i(n-i+\frac{1}{3})} + \sum_{i=n+1}^{\infty} \frac{3n}{i(i-n-\frac{1}{3})} \\ &\leq C_2 + 3\sum_{i=1}^{\infty} \frac{-(n+\frac{1}{3})}{i(i-(n+\frac{1}{3}))} + 3\sum_{i=1}^{\infty} \frac{n}{i(n+i)} \\ &\leq C_3 + 3\psi(1-(n+\frac{1}{3})) + 3\psi(n+1) \\ &= O(\log n) \end{aligned}$$

We conclude by remarking on the existence of a threshold for super-logarithmic to logarithmic version age in $G(n,p)$. Fix $\varepsilon > 0$, $\delta > 0$ and the sequences $p_0(n) := \frac{(1-\varepsilon)\log n}{n}$ and $p_1(n) := \frac{(100+\varepsilon)\log n}{n}$. Consider the graph property $P =$'G has average version age less than $\alpha \log n$'. We just proved that $\mathbb{P}[G(n,p_0) \in P] \leq \delta$, and $\mathbb{P}[G(n,p_1) \in P] \geq 1-\delta$, when α is suitably large. By Lemma 6 adding edges to any G cannot increase the average version age, P is a monotone graph property invariant under vertex permutations. Therefore for all n, there exists a $p^*(n) \in (p_0(n), p_1(n))$ where $\mathbb{P}[G(n,p^*) \in P] = \frac{1}{2}$. By the shrinking critical window, $\lim_{n\to\infty} \frac{p_1(n)-p_0(n)}{p_c(n)}$ exists, so Definition 6 is satisfied and a threshold exists. □

We briefly note that part 1 of Lemma 3 implies for $d < 1$ and $p < d/n$, the expected version age of a vertex in $G(n,p)$ is $\Omega\left(\frac{n}{\log n}\right)$. For p much greater than in part 3, the version age still scales logarithmically, but the constant factors decrease.

2.7 General Bound on Version Age

Throughout we fix a communication graph $G = (V, E)$ and a distinguished vertex $v \in V$ whose age we wish to track. Let $B_m := \{u : \text{dist}_G(u, v) \leq m\}$ be the ball of radius m about v. Define $m_* := \min\{m : |B_m(v)| m \geq n\}$ and $\Phi_m := \frac{|B_m(v)|}{|B_{m-1}(v)|}$. Recall $X_t(v)$ is the number of versions node v is behind the source, δ is the minimum degree of G and Δ is the maximum degree of G. For consistency with the above notation, we drop the dependence on v for the version age and define $X_t := X_t(v)$. For notational convenience we assume $\lambda_e = \lambda = 1$. In this section we prove Theorem 4

2.7.1 Lower Bound

We start with the lower bound in Theorem 4.

Proof. For C_1 as given in the Theorem statement, let $K = \left(100C_1\left(\frac{\Delta}{\delta} + \Phi_{m_*}\right)\right)^{-1}$. For any $m \in \mathbb{N}$, $0 \leq t$, and $a \geq 0$ define the following events:

- $\mathcal{E}_1$ is the event the source n_0 updates itself at least Km_* times in the interval $[t - Km_*, t]$
- $\mathcal{E}_2$ is the event no vertex in B_{m_*} received an update from the source in the in the interval $[t - Km_*, t]$
- $\mathcal{E}_3$ is the event there is no path from $B_{m_*} \setminus B_{m_*-1}$ to v that updates in sequence in the interval $[t - Km_*, t]$

Note that $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ are mutually independent, so it suffices to show each are bounded below by a constant, so that their intersection has positive constant probability.

The probability of $\mathcal{E}_1$ is at least $1/2$. The probability of $\mathcal{E}_2$ can be bounded as,

$$\mathbb{P}[\mathcal{E}_2] = \exp\left(-\frac{Km_*|B_{m_*}|}{n}\right) \geq \exp\left(-\frac{2K(m_*-1)|B_{m_*-1}|\Phi_{m_*}}{n}\right) \geq \exp\left(-2K\Phi_{m_*}\right)$$

Which is bounded away from 0 by our choice of K. For $\mathcal{E}_3$ we need the following result,

Lemma 5. *For $Y_1, \ldots, Y_N$ i.i.d. exponential random variables with mean μ, there is a constant C such that*

$$\mathbb{P}\left[\sum_{j=1}^{N} Y_j \leq t\right] \leq \left(\frac{Ct\mu}{N}\right)^N$$

For every vertex $w_0 \in B_{m_*} \setminus B_{m_*-1}$, the number of paths of length L from w_0 to v is at most Δ^L. Let $w_0, w_1, ..., w_{L-2}, v$ be such a path. Let T_1 be the time it takes after $t - Km_*$ for the edge w_0w_1 to update. Then T_2 the time after already waiting T_1 for w_1w_2 to update, and so on for all edges in the path. By definition of the model, the T_i are independent exponential random variables with mean at most $1/\delta$ (each edge represents a Poisson process with mean inter-arrival time given by its degree). Therefore define $\{Y_j\}_{j=1}^{L}$ to be i.i.d. standard exponential random variables. We have by Lemma 5,

$$\mathbb{P}[(w_0, w_1, ..., v) \text{ update in sequence}] \leq \mathbb{P}\left(\sum_{i=1}^{L} Y_i \leq Km_*\right) \leq \left(\frac{CKm_*}{L\delta}\right)^L$$

Summing over all paths we obtain a bound of,

$$\begin{aligned}\mathbb{P}[\mathcal{E}_3^c] &\leq |B_{m_*}| \sum_{L=m_*+1}^{\infty} \left(\Delta \frac{CKm_*}{\delta L}\right)^L \\ &= |B_{m_*}| \sum_{L=m_*+1}^{\infty} \left(\Delta \frac{CKm_*|B_{m_*}|^{1/L}}{\delta L}\right)^L \\ &\leq |B_{m_*}| \sum_{L=m_*+1}^{\infty} \left(\Delta \frac{CKm_*|B_{m_*}|^{1/m_*}}{\delta L}\right)^L\end{aligned}$$

Which is bounded away from 0 by our choice of K. Therefore $\mathbb{P}[E_1 \cap E_2 \cap E_3]$ is bounded away from 0, and on this event the age of v is at least Km_* which implies

$$\mathbb{E}X_t \geq C_1 m_* \min\left\{\frac{\delta}{\Delta}, \frac{1}{\Phi_{m_*}}\right\}$$

□

2.7.2 Upper Bound

We prove the upper bound in Theorem 4.

Proof. For any $m \in \mathbb{N}$, $0 \leq t$, and $a \geq 0$ define the following events:

- $\mathcal{E}_4$ is the event that B_m received a source update in the time interval $\left[t - \frac{2a}{3}, t - \frac{a}{3}\right]$
- Among all vertices in B_m that received an update from the source in $\left[t - \frac{2a}{3}, t - \frac{a}{3}\right]$, let w be one such vertex chosen uniformly at randomly. $\mathcal{E}_5$ is the event that there is a path of

vertices from w to v so that the directed edges of the path update in sequence during the times $\left[t - \frac{a}{3}, t\right]$.

- $\mathcal{E}_6$ is the event that the source updates fewer than a times in $\left[t - \frac{2a}{3}, t\right]$.

We have the following fact,

Fact 2. $\mathbb{P}[X_t > a] \leq \mathbb{P}[\mathcal{E}_4^c] + \mathbb{P}[\mathcal{E}_5^c] + \mathbb{P}[\mathcal{E}_6^c]$

Proof. Observe that on the event $\mathcal{E}_4 \cap \mathcal{E}_5 \cap \mathcal{E}_6$ we have $X_t \leq a$, then the fact is immediate from the union bound. □

It then suffices to bound each of the events. By the model definition, the number of updates from the source to any subset $S \subset V$ in the interval $[t - t_0, t]$ is a Poisson random variable with parameter $\frac{t_0|S|}{n}$. Therefore for $\mathcal{E}_4^c$,

$$\mathbb{P}(\mathcal{E}_4^c) \leq \exp\left(-\frac{a|B_m|}{3n}\right) \leq \exp\left(-\frac{a}{3m_*}\right).$$

For $\mathcal{E}_5^c$, it suffices to bound the probability an individual path from an updated node $w_0 \in B_m$ to v updates in sequence. Note this path has length at most $m \leq m_*$. Define $\{Y_i\} i = 1^m$ to be i.i.d. standard exponential random variables with mean $1/\Delta$. By a similar argument for bounding $\mathcal{E}_3^c$ in the previous section,

$$\mathbb{P}[\mathcal{E}_5^c] \leq \mathbb{P}\left[\sum_{j=1}^{m} Y_j \geq \frac{a}{3\Delta}\right] \leq \mathbb{P}\left[\sum_{j=1}^{m_*} Y_j \geq \frac{a}{3\Delta}\right] \leq \mathbb{P}\left[\frac{1}{m_*}\sum_{j=1}^{m_*}(Y_j - 1) \geq \frac{a}{3\Delta m_*} - 1\right] \leq \exp\left(\frac{-c_0 a}{3\Delta m_*}\right)$$

For some absolute constant c_0, where the last inequality follows from Bernstein's inequality.

Finally we have $\mathcal{E}_6^c$ is exactly the probability a Poisson variable with mean $2a/3$ is larger than a, which is bounded by,

$$\mathbb{P}[\mathcal{E}_6^c] \leq \exp(-c_1 a)$$

For some absolute constant c_1. Let $c := \max\{c_0, c_1\}$. Then we have,

$$\begin{aligned}\mathbb{P}[X_t > a] &\leq \exp\left(-\frac{a}{3m_*}\right) + \exp\left(-\frac{ca}{3\Delta m_*}\right) + \exp\left(-ca\right) \\ &\leq 3\exp\left(-\frac{ca}{3\Delta m_*}\right)\end{aligned}$$

So summing over all a and letting C be a large constant,

$$\begin{aligned}\mathbb{E}X_t &= \sum_{a\geq 0} \mathbb{P}[X_t > a] \\ &\leq 3\sum_{a\geq 0} \exp\left(-\frac{ca}{3\Delta m_*}\right) \\ &\leq C\int_0^\infty \exp\left(-\frac{cx}{3\Delta m_*}\right) dx \\ &= C_2 \Delta m_*\end{aligned}$$

□

2.7.3 The Upper Bound is tight: Δ-regular tree

Let $T_\Delta(n)$ be the complete Δ-regular tree on n vertices (n chosen appropriately, Δ is constant). Note that for the root vertex r we have $m_*(r) = \log_\Delta(n)$, so the following result matches the upper bound in Theorem 4.

Theorem 6. *There is some constant* $C := C(\Delta)$ *such that for the root* r *of the* Δ*-regular tree, the average expected age of* r *is at least* $C\Delta \log_\Delta(n)$

Proof. The proof follows similarly from Section 2.7.1, but specialized to T_Δ. Let $K = \frac{\Delta}{100}$, and $d = \frac{m_*}{2} = \frac{\log_\Delta(n)}{2}$. For any $t \geq 0$ define the following events.

- $\mathcal{E}_1$ is the event the source updates at least Km_* times in $[t - Kd, t]$
- $\mathcal{E}_2$ is the event no vertex in $B_d(r)$ receives an update from the source in $[t - Kd, t]$
- $\mathcal{E}_3$ is the event no path from $B_d(r) \setminus B_{d-1}(r)$ updates in sequence in $[t - Kd, t]$

Then the probability of $\mathcal{E}_1$ is bounded below by a non-zero constant. For $\mathcal{E}_2$, note $|B_d(r)| \sim \sqrt{n}$. Therefore,

$$\mathbb{P}[\mathcal{E}_2] = \exp\left(-\frac{Kd|B_d(r)|}{n}\right) \sim \exp\left(-\frac{\Delta \log_\Delta(n)}{200\sqrt{n}}\right)$$

which is also bounded below by a non-zero constant.

For $\mathcal{E}_3$, we can apply Lemma 5 to bound the probability a path $(w_0, w_1, ..., r)$ of length L updates in sequence in the time window as,

$$\mathbb{P}[(w_0, w_1, ..., r) \text{ updates in sequence}] \leq \mathbb{P}\left[\sum_{i=1}^{L} Y_i \leq kd\right] \leq \left(\frac{CKd}{L\Delta}\right)^L$$

Where the Y_i are i.i.d. copies of exponential random variables with mean Δ. We can then sum over all such paths and bound,

$$\mathbb{P}[\mathcal{E}_3^c] \leq |B_d(r)| \sum_{L=d+1}^{\log_\Delta(n)} \Delta^L \left(\frac{CKd}{L\Delta}\right)^L \leq |B_d(r)| \sum_{L=d+1}^{\infty} \left(\frac{CKd}{L}\right)^L$$

Which is bounded away from 1 by some constant that depends on Δ. Therefore on the event $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$ is bounded from 0 as the tree grows, and on this event the age is at least $km_* = \frac{1}{100}\Delta \log_\Delta(n)$.

□

We briefly comment that as $d \to \infty$, $\Phi_{m_*}(\mathbb{Z}^d) \to 1$, and we believe for this case the extra factor $\Delta = 2d$ can be removed so that the d-dimensional lattice is a tight example for the lower bound.

2.7.4 Application of Theorem 4 to Open Problems

We start with the observation that the diameter of a graph G is greater than $m_*(G)$, where $m_*(G) := \max_u\{m_*(u)\}$ is the largest value of m_* achieved in G. We now apply Theorem 4 to some open problems. All graphs below are assumed to have n vertices.

1. If $\mathbb{Z}_n^d$ is the d-dimensional toroidal lattice ($n = \ell^d$), then $m_* = n^{\frac{1}{d+1}}$, so there is some large constant C_d depending only on the dimension such that $\frac{1}{C_d} n^{\frac{1}{d+1}} \le \mathbb{E}X_t \le 2C_d d n^{\frac{1}{d+1}}$. This solves an open problem from (21) when d is constant or slowly growing with n.

2. If $G = C_n$ is a cycle, then this is just a specialization of the above to 1-dimension, so $\mathbb{E}X_t \sim n^{1/2}$

3. If $G = C + \mathcal{M}$, is the union of a cycle and a random matching, then a classic result by Bollobas and Chung (26) gives the diameter of G is with high probability $O(\log n)$, therefore $X_t = \Theta(\log n)$. This shows a curious property - adding $O(n)$ edges can reduce the version age of a graph from $O(\sqrt{n})$ to asymptotically optimal.

4. For constant d, let $G = \mathcal{G}(n, d)$ be a d-regular graph chosen uniformly at random from all possible d-regular graphs on n vertices. Bollobas and Vega (27) showed the diameter of G is $O(\log(n \log n))$, so $X_t = \Theta(\log n)$. This is another proof of Theorem 2.

5. Let $G = G^d(\gamma, n)$ be the d-dimensional random geometric graph, where vertices are vectors $x \in \mathbb{R}^d, \|x\|_2 < 1$ distributed uniformly at random in the unit ball and $uv \in G \iff \|u, v\|_2 < \gamma$. For $\varepsilon > 0$ Suppose that $\gamma > (1+\varepsilon)\left(\frac{\log n}{n}\right)^{1/d}$, then Ellis, Martin, and Yan (28) showed this is sufficient for G to be connected, and furthermore G has diameter scaling as $O\left(\frac{1}{1+\varepsilon}\left(\frac{n \log\log n}{\log^2 n}\right)^{1/d}\right)$. It can be shown that the degree distribution in G concentrates, so with high probability no vertex has degree greater than $C_d(1+\varepsilon)^d \log^2 n$ for some constant C_d depending only on the dimension. Therefore $X_t = O\left((1+\varepsilon)^{d-1} \log^2 n (n \log\log n)^{1/d}\right) = \tilde{O}((1+\varepsilon)^{d-1} n^{1/d})$, which agrees with item (1) as d gets large up to logarithmic factors. This, along with item (1) gives evidence that connected graphs embedded in a metric space have version age governed by some global property of the metric space.

2.8 Remarks

We have analyzed vAoI from the combinatorial perspective of Yates (19), as well as a novel probabilistic approach in section 2.7. There are still open problems in this area. One question is posed at the end of item 5 above: if G is a graph embedded in a metric space, so that edges now have weights proportional to their distance in the space, how does version age scale? Would long-range connections with high weights improve version age? This doesn't follow from the Monotonicity of vAoI, as the model would need modifying.

2.9 Monotonicity of vAoI

We show the vAoI function is monotone, which was extensively used up to section 2.7. We assume λ, λ_e are constants, and $\lambda_i(j) = \frac{\lambda}{\deg(i)}$.

Lemma 6. *For any graph G, let G' be obtained from G by adding an edge (uv). Then for any $S \subset G$,*

$$\nu_{G'}(S) \leq \nu_G(S)$$

Proof. Recalling equality 2.1, note that if $u \in S$ and $v \in S$, then the version age of S in G and G' are identical. Therefore we only need to consider the cases when one of u, v are in S, or neither are in S. We prove these cases separately via reverse induction on $S \subset G$. To that end, the base case is $S = \mathcal{N}$ and the statement is true by the observation above. For the inductive hypothesis, suppose the claim is true for $|S'| = k > 1$. We now consider sets $|S| = k - 1$.

Case 1 is that $u \in S$ but $v \notin S$. Then,

$$\begin{aligned}
\nu_{G'(S)} &= \frac{\frac{\lambda_e}{\lambda} + \sum_{i \notin S \cup \{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \nu_{G'}(S \cup \{i\}) \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)} \nu_G(S \cup \{v\})}{\frac{|S|}{n} + \sum_{i \notin S \cup \{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)}} \\
&\leq \frac{\frac{\lambda_e}{\lambda} + \sum_{i \notin S \cup \{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \nu_G(S \cup \{i\}) \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)} \nu_G(S \cup \{v\})}{\frac{|S|}{n} + \sum_{i \notin S \cup \{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)}} \\
&\leq \left(\frac{\frac{|S|}{n} + \sum_{i \notin S \cup \{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right)}{\frac{|S|}{n} + \sum_{i \notin S \cup \{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)}} \right) \nu_G(S)
\end{aligned}$$

$$
\begin{aligned}
&+\left(\frac{\frac{|S\cap N_{G'}(v)|}{\deg_{G'}(v)}}{\frac{|S|}{n}+\sum_{i\notin S\cup\{v\}}\left(\frac{|S\cap N_G(i)|}{\deg_{G'}(i)}\right)+\frac{|S\cap N_{G'}(v)|}{\deg_{G'}(v)}}\right)\nu_G(S\cup\{v\})\\
\leq&\left(\frac{\frac{|S|}{n}+\sum_{i\notin S\cup\{v\}}\left(\frac{|S\cap N_G(i)|}{\deg_{G'}(i)}\right)}{\frac{|S|}{n}+\sum_{i\notin S\cup\{v\}}\left(\frac{|S\cap N_G(i)|}{\deg_{G'}(i)}\right)+\frac{|S\cap N_{G'}(v)|}{\deg_{G'}(v)}}\right)\nu_G(S)\\
&+\left(\frac{\frac{|S\cap N_{G'}(v)|}{\deg_{G'}(v)}}{\frac{|S|}{n}+\sum_{i\notin S\cup\{v\}}\left(\frac{|S\cap N_G(i)|}{\deg_{G'}(i)}\right)+\frac{|S\cap N_{G'}(v)|}{\deg_{G'}(v)}}\right)\nu_G(S)\\
=&\,\nu_G(S)
\end{aligned}
$$

Where the first inequality is from the inductive hypothesis, the second is multiplying by 1 and applying equation 2.1, the third is the fact that adding a vertex to a set cannot increase the version age, and the fourth is just rearranging the coefficients in the large brackets.

Case 2 is when neither u or v are in S. Notice that the only terms obtained by unrolling equation 2.1 that are different in G and G' are those when either u or v are neighbors of the S; otherwise the extra edge uv doesn't play a role in the expansion, or the edge is present in S so the terms are identical. Therefore we only need to consider those sets which are distance 1 from $\{u, v\}$. For completeness we can then perform analogous computations to above.

$$
\nu_{G'(S)} =
\frac{\frac{\lambda_e}{\lambda}+\sum_{i\in N_{G'}(S)\setminus\{u\}\setminus\{v\}}\left(\frac{|S\cap N_G(i)|}{\deg_{G'}(i)}\nu_{G'}(S\cup\{i\})\right)+\frac{|S\cap N_{G'}(v)|}{\deg_{G'}(v)}\nu_{G'}(S\cup\{v\})+\frac{|S\cap N_{G'}(u)|}{\deg_{G'}(u)}\nu_{G'}(S\cup\{u\})}{\frac{|S|}{n}+\sum_{i\in N_{G'}(S)\setminus\{u\}\setminus\{v\}}\left(\frac{|S\cap N_G(i)|}{\deg_{G'}(i)}\right)+\frac{|S\cap N_{G'}(v)|}{\deg_{G'}(v)}+\frac{|S\cap N_{G'}(u)|}{\deg_{G'}(u)}}
$$

$$
\begin{aligned}
&\leq \frac{\frac{\lambda_e}{\lambda} + \sum_{i \in N_{G'}(S)\setminus\{u\}\setminus\{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} v_G(S \cup \{i\}) \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)} v_G(S \cup \{v\}) + \frac{|S \cap N_{G'}(u)|}{\deg_{G'}(u)} v_G(S \cup \{u\})}{\frac{|S|}{n} + \sum_{i \in N_{G'}(S)\setminus\{u\}\setminus\{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)} + \frac{|S \cap N_{G'}(u)|}{\deg_{G'}(u)}} \\
&\leq \left(\frac{\frac{|S|}{n} + \sum_{i \in N_{G'}(S)\setminus\{u\}\setminus\{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right)}{\frac{|S|}{n} + \sum_{i \in N_{G'}(S)\setminus\{u\}\setminus\{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)} + \frac{|S \cap N_{G'}(u)|}{\deg_{G'}(u)}} \right) v_G(S) \\
&\quad + \left(\frac{\frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)}}{\frac{|S|}{n} + \sum_{i \in N_{G'}(S)\setminus\{u\}\setminus\{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)} + \frac{|S \cap N_{G'}(u)|}{\deg_{G'}(u)}} \right) v_G(S \cup \{v\}) \\
&\quad + \left(\frac{\frac{|S \cap N_{G'}(u)|}{\deg_{G'}(u)}}{\frac{|S|}{n} + \sum_{i \in N_{G'}(S)\setminus\{u\}\setminus\{v\}} \left(\frac{|S \cap N_G(i)|}{\deg_{G'}(i)} \right) + \frac{|S \cap N_{G'}(v)|}{\deg_{G'}(v)} + \frac{|S \cap N_{G'}(u)|}{\deg_{G'}(u)}} \right) v_G(S \cup \{u\}) \\
&\leq v_G(S)
\end{aligned}
$$

□

We have the easy corollary, which follows from above and results by (19).

Corollary 1. *For any graph* G *and any* $i \in G$*,*

$$\Omega(\log n) \leq v_G(\{i\}) \leq O(n)$$

2.10 Proof of Lemma 3

Proof. Let X be the random variable counting the number of isolated vertices in $G(n, p)$ for $p = \frac{(1-\varepsilon)\log n}{n}$. In a classical result, Barbour, Karoński, and Ruciński (29) show $\frac{X - \mathbb{E}X}{\sqrt{\operatorname{Var} X}} \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1)$. That is, X appropriately scaled and shifted converges in distribution to the standard Gaussian.

We can compute for n large, $\mathbb{E}X \approx n\exp(-(1-\varepsilon)\log n) \approx n^{\varepsilon}$ and $\operatorname{Var} X \approx n^{\varepsilon} + 2\binom{n}{2}\exp(-2(1-\varepsilon)\log n) \approx n^{\varepsilon} + n^{-2\varepsilon}$.

Then by the Portmanteau Lemma and tail bounds on the standard Gaussian,

$$\begin{aligned}
\lim_{n\to\infty} \mathbb{P}\left(\frac{X - n^{\varepsilon}}{\sqrt{n^{\varepsilon} + n^{-2\varepsilon}}} \le -\sqrt{\log n}\right) &= \mathbb{P}\left(\mathcal{N}(0,1) \le \sqrt{\log n}\right) \\
&\le \frac{1}{2n} \\
\iff \mathbb{P}\left(X \le n^{\varepsilon} - \left(\log n(n^{\varepsilon} + n^{-2\varepsilon}\right)^{1/2}\right) &\le \frac{1}{2n}
\end{aligned}$$

So in particular, a.a.s. $X \ge n^{\varepsilon - o(1)}$

□

CHAPTER 3

STRING AND TREE RECONSTRUCTION

3.1 Introduction

In this chapter, we study the complexity of reconstructing trees from traces under the tree edit distance model, as defined by Davies et al. (30). Tree trace reconstruction is a generalization of the traditional string trace reconstruction problem, where we observe noisy samples of a binary string after passing it through a deletion channel several times (31). Trace reconstruction and tree trace reconstruction have a variety of applications including in archival DNA storage, sensor networks, and linguistic reconstruction (32; 33; 34; 35).

3.1.1 String Trace Reconstruction

A trace of a string s on n bits is obtained by sending the string through a deletion channel that removes each bit of s independently with probability q. Each trace is generated independently of other traces. The *string trace reconstruction problem* is to recover s with probability at least $1 - \delta$ using as few traces as possible. Denote by $T(n, q, \delta)$ the minimum number of traces needed to reconstruct a string with n bits with deletion rate q and success probability $1 - \delta$. We sometimes hide the dependency on q by writing $T(n, \delta)$ when q is known.

This problem has been studied extensively, yet there is still an exponential gap in the upper and lower bounds on trace sample complexity. (36) recently proved that $\exp(\tilde{O}(n^{1/5}))$ traces

are sufficient to reconstruct a string, improving on the previous bound of $\exp(O(n^{1/3}))$ traces that was independently proved by (37; 38). The current best lower bound is $\tilde{\Omega}(n^{3/2})$ (39).

Many variants on trace reconstruction have been considered. In the average-case setting, the unknown input string is chosen uniformly at random, and improved upper bounds are known in this case (31; 40; 41; 42). (41) showed that $O(\exp(C \log^{1/3} n))$ traces are sufficient to recover the string with high probability.

Other settings which yield improved bounds are in the smoothed-analysis setting where the unknown string is perturbed before generating traces (43), in the coded setting where the string is confined to a pre-determined set (44; 45; 46), and in the approximate setting where the goal is to output a string that is close to the original in edit-distance (47).

The motivation for developing reconstruction theory is also pragmatic, reconstruction has direct application of trace reconstruction to DNA data storage where algorithms are deployed to recover the data from noisy samples (32; 33; 48). Clearly there is a practical need to reduce the sample complexity as this impacts the time and cost of retrieving archival data stored with DNA. For example, biologists were able to store a 53,426 word book, 11 JPG images, and a JavaScript program (a total of 5.27 megabits) in a DNA medium (33).

3.1.2 Tree Trace Reconstruction

Robust branching DNA structures for storage have recently been created by biologists and bio-engineers (34; 49). This naturally leads to a modelling question for reconstructing these structures. In the *tree trace reconstruction problem*, a trace of an ordered tree R on n nodes is obtained by removing each node of R independently with probability q. Each trace is generated

independently of the other traces. An ordered tree is a rooted tree where the children of each vertex is assigned a total order. In our diagrams the ordering of children is depicted left-to-right. Given a tree R taken from the set of all trees on n nodes, the goal is to reconstruct R with probability at least $1-\delta$ using as few traces as possible. There are many ways to define removing a node in a tree. We follow the tree edit distance and left-propagation models introduced by (30). In the *tree edit distance* (TED) model when a node w is removed, the children of w become the children of w's parent.

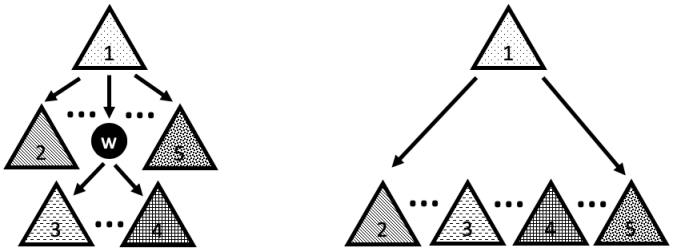

Figure 1: The generic picture before (left) and after (right) node w is removed from tree R in the TED model. The subtrees 3 and 4 are inserted as children of 1 when w is removed.

The *left-propagation* model is discussed in 3.3.2. Now when a node w is deleted, recursively replace every node (together with its label) in the left-only path defined by the total orderings starting at w with its child in the path. The ordering of children stays the same, except that now the left-most child might change for some vertices. Note that in both models the order of deletions doesn't matter, only the set of nodes that are removed.

3.2 Related Work

The problem of string-trace reconstruction has received considerable attention since it was first introduced by Batu et. al. (31). Despite this there is still an exponential gap in the upper and lower bounds for the number of traces needed to reconstruct an arbitrary input string (39; 36; 38; 40; 37). The best lower bound thus far is due to Chase (39), where he showed that $\tilde{\Omega}(\log^{5/2} n)$ traces are necessary, and the best upper bound is due to Chase (36), where the same author showed that $\exp(\tilde{O}(n^{1/5}))$ traces are sufficient. Previous to this, Nazarov and Peres (37) and De et al. (38) independently achieved the upper bound of $\exp(O(n^{1/3}))$ traces.

Variants of trace reconstruction have been proposed in order to develop new techniques to close the exponential gap in the original problem formulation. Davies et al. (30) investigated a new problem of reconstructing trees from their traces, which this chapter builds upon. Those authors investigated two classes of trees; complete k-ary trees and spider graphs. They also introduced the "tree edit distance" (TED) and "left-propagation" deletion channels.

For a complete k-ary tree T the authors considered the general case for arbitrary k, as well as when k is 'large' compared to the total number of nodes. Under TED, if $k \geq O(\log^2(n))$, then a complete k-ary tree can be reconstructed using $\exp(O(\log_k n)) \cdot T(k, 1/n^2)$ traces generated from T. With no restrictions on k, $\exp(O(k \log_k n))$ traces suffice to reconstruct T. Under left-propagation, if $k \geq O(\log\ n)$ then $T(O(\log_k n + k), 1/n^2))$ traces suffice to reconstruct T. For arbitrary k then $O(n^\gamma \log n)$ traces suffice, where $\gamma = \ln\left(\frac{1}{1-q}\right)\left(\frac{c'k}{\ln n} + \frac{1}{\ln k}\right)$. The authors use primarily combinatorial arguments to achieve these bounds, which is in contrast to the

current methods used in string trace reconstruction. It was given as an open problem to extend these combinatorial arguments to general trees.

The case of spider graphs was broken into small leg-depth and large leg-depth. Note that for spider graphs, TED and left-propagation are identical deletion channels. For depth $d \leq \log_{1/q} n$ and $q > 0.7$, then $\exp(O(d(nq^d)^{1/3}))$ traces suffice to reconstruct the spider. For depth $d \geq \log_{1/q} n$ and all $q > 1$, then $2T(d, 1/2n^2)$ traces suffice to reconstruct the spider. The algorithms proposed here generalize the mean-based methods of De et al. (38) and Nazarov and Peres (37) to multiple independent strings, where some strings have a chance of being completely removed.

3.3 Tree Reconstruction Lower Bounds

The goal of this section is to give lower bounds for tree reconstruction via reductions to string reconstruction. As stated in 3.1.2 all trees are ordered. Theorem 7 gives a reduction from learning trees under TED to the string reconstruction problem. Corollary 1 shows recovery of arbitrary trees in the left-propagation model require exponentially many traces.

3.3.1 Recovering Unknown Tree Topologies

Given an ordered tree R on vertex set $\mathcal{N}$, a labelling of R is a function $f : \mathcal{N} \to \{0, 1\}$. This section is concerned with recovery of R when the labelling is uniform, i.e. $f \equiv 0$. We will call such trees *unlabelled*.

Begin by considering any binary string $s \in \{0, 1\}^n$, $q \in (0, 1)$ and $\delta > 0$. In the next paragraph we construct a tree $R_s := R(s, q, \delta)$ that naturally 'encodes' s. Given the deletion rate q and failure probability δ this will allow an emulation of the string reconstruction problem.

First, create a path with $n + 2\ell$ vertices, where $\ell = \Theta\left(\frac{\ln(1/\delta)+\ln(T(n,q,\delta))}{\ln(1/q)}\right)$. Next, for each bit $s^{(i)} \in s$, if $s^{(i)} = 0$ add a left child to the node in position $(\ell + i)$ in the path. Otherwise $s^{(i)} = 1$ so add a right child to node $(\ell + i)$ in the path.

Lemma 7. *Let $s \in \{0,1\}^n$ be a string and R_s be the tree encoding described above. Let $\delta > 0$ and $q \in (0,1)$. Under the TED deletion channel with deletion rate q, reconstructing R_s with probability greater than $1 - \delta$ requires at least $\Omega(T(n, q^2, \delta))$ traces.*

Proof. Begin by generating $T(n, q^2, \delta)$ i.i.d. TED traces of R_s. Call this (random) set of traces $\mathcal{T}$. Recall that the data available to any reconstruction algorithm is just $\mathcal{T}$, δ, and q. We will strengthen the algorithm by informing it that the set of possible trees that could have generated $\mathcal{T}$ are precisely those R_s generated from any $s \in \{0,1\}^n$; note that this cannot increase the lower bound.

Observe that by the choice of ℓ above, with probability $1 - \delta$ for any $R' \in \mathcal{T}$ at least one of the nodes in the path below ℓ and above $n + \ell$ survives. Using $O(n \log \frac{1}{\delta})$ traces, by Hoeffding's inequality with probability $1 - \delta$ at least one trace will include a path of length $n + 2\ell$. Therefore what remains is to determine the positions of the leaves in R_s.

Noting figure Figure 2, say a leaf and its parent are *completely removed* from R_s if both are deleted in the TED channel (e.g. a is completely removed if a and b are deleted). If a leaf is completely removed this corresponds to a complete deletion of a bit in s. Otherwise there are residual nodes in the trace that don't correspond to bits in s. This suggests the following procedure for generating a trace equivalent to TED. First with probability q^2 each leaf-parent pair is removed i.i.d. (if a node does not have a leaf child it is ignored). Then with

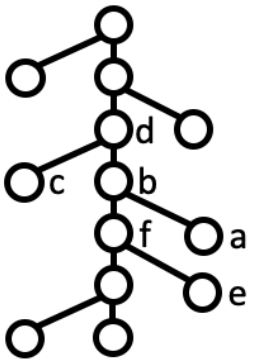

Figure 2: Example configuration of a leaf node a and the leaves nearest it. The survival of a is dependent on the nodes in its 2-neighborhood.

probability $q - q^2$ each node is removed i.i.d. (a node ignored in the previous step is removed with probability q).

We consider the modified TED channel where only the first deletion step is performed; this cannot increase the lower bound. By the above observations any trace generated from this modified channel will contain a path of some length, then an encoding of a string trace generated from s, then another path of some length. Since we used Hoeffding's inequality to recover the length of the longest path in R_s, we may apply the string trace reconstruction algorithm to recover the leaf positions in the weakened TED channel. Therefore recovery of R_s under the TED channel is at least as hard as recovering s under the deletion channel. □

Note that the choice of ℓ reduces to $O(n)$ if q and δ are constants, so the lemma above shows tree reconstruction under TED is at least as hard as string reconstruction up to some constant factors.

Theorem 7. *Let* $q \in (0,1)$ *and* $\delta > 0$ *be constants. Then there exists* $C > 0$ *such that for any* $r, s \in \{0,1\}^{Cn}$*, at least* $\Omega(T(n, q^2, \delta))$ *TED traces are needed to distinguish* R_r *and* R_s *with probability at least* $1 - \delta$*.*

3.3.2 Left-Propagation

We briefly describe an easy construction to show that reconstruction of arbitrary trees under the left-propagation model requires at least exponentially many traces. Consider the unlabelled trees P_n and R_n on n vertices in figure Figure 3. Tree P_n is a path on n vertices. Tree R_n is a path on $n-2$ vertices, with the final node in the path having 2 children. As in the previous section we consider the case that all vertices have the same label 0, and will refer to these as 'unlabelled' trees.

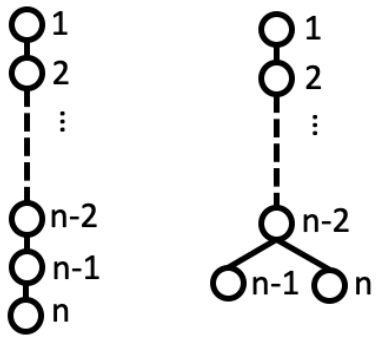

Figure 3: Trees P_n and R_n described above

Consider the following protocol for distinguishing distributions. An algorithm A is given as input a positive integer $n > 0$. A returns a value $f(n)$, the number of samples drawn from known distribution $\mathcal{D}_1$ or $\mathcal{D}_2$. It is then provided $f(n)$ i.i.d samples from one of these distributions. We say A is a *distinguishing algorithm* for $\mathcal{D}_1$ and $\mathcal{D}_2$ if with probability greater than $2/3$, A outputs the correct distribution the samples were drawn from. We say $f(n)$ is the *sample size* A uses to distinguish the distributions.

Lemma 8. *The expected number of traces needed for any algorithm to distinguish* P_n *and* R_n *under the left-propagation model is at least* $\Omega\left((1-q)^{-n}\right)$

Proof. Let $\mathcal{D}_P$ be the distribution of traces generated from the left-propagation channel applied to P_n, and similarly for $\mathcal{D}_R$. If k nodes are removed from either P_n or R_n, then the resulting trace is isomorphic to P_{n-k}. Therefore the support of $\mathcal{D}_P$ and $\mathcal{D}_R$ are identical except for the tree generated when no nodes are removed. Furthermore the probability mass on elements in the intersection of the supports are identical.

Because the distributions are identical conditioned on removing at least one node, distinguishing $\mathcal{D}_P$ and $\mathcal{D}_R$ requires at least one sample where no nodes have been removed. The expected number of samples needed for this to happen is exactly $\frac{1}{(1-q)^n}$. □

Distinguishing between the distributions generated by P_n and R_n is a prerequisite to reconstructing arbitrary trees. Therefore we conclude.

Corollary 1. *For any* $\delta > 0$ *the number of traces needed to reconstruct arbitrary ordered trees under the left-propagation model with probability at least* $1-\delta$ *is at least* $\Omega\left((1-q)^{-n}\right)$.

3.4 Tree Reconstruction Upper Bounds

The goal of this section is to give upper bounds for tree reconstruction. Subsection 3.4.1 shows the labels of a known tree can be recovered via a reduction to string reconstruction. Subsection 3.4.2 proves trees with large degree can be reconstructed with polynomially many traces. Subsection 3.4.3 shows if trees are encoded in a special way, then they can be reconstructed with polynomially many traces.

3.4.1 Labelled Trees with known Topology

We start by showing that learning the labels of a tree under TED is no harder than the string reconstruction problem.

Lemma 9. *Let* $\mathtt{R}$ *be an ordered tree and* $\mathtt{R}'$ *be an ordered tree obtained from* $\mathtt{R}$ *via an arbitrary set of deletions under the tree edit distance model. If* u, v *are any two nodes in* $\mathtt{R}'$ *such that a pre-order traversal of* $\mathtt{R}'$ *visits* u *before* v *then a pre-order traversal of* $\mathtt{R}$ *will also visit* u *before visiting* v*.*

Proof. Given the deletion of an arbitrary node w, we consider how it effects the traversal order of the remaining nodes. Figure 1 shows the general case when a node w is removed, with the subtrees numbered in the order they are visited according to a pre-order traversal.

If u and v fall in the same subtree, their visitation order is clearly not affected. Otherwise, if u is in a tree that's visited earlier than when the tree of v is visited on the left of Figure 1, then it is also visited earlier on the right. This simple illustration captures all of the cases (i.e. pairs of trees where u and v can occur) □

This immediately gives the following.

Theorem 8. *Let* R *be any ordered labelled tree on* n *nodes and* $\delta > 0$*. Given only* $T(n, q, \delta)$ *traces from the TED deletion channel, the labels of* R *ordered by pre-order traversal can be reconstructed with probability at least* $1 - \delta$*.*

Applying the results of Chase (36) for recovering any string from $\tilde{O}(\exp(n^{1/5}))$ traces when q is constant, there is an algorithm for reconstructing a tree from traces if the structure of the tree is given to the algorithm.

Corollary 2. *Let* R *be any ordered labelled tree on* n *nodes, and* $q, \delta > 0$*. Given the structure of* R *and* $\tilde{O}(\exp(n^{1/5}))$ *traces from the TED deletion channel,* R *can be reconstructed with probability at least* $1 - \delta$*.*

3.4.2 Trees with Large Degree

For any tree, call a leaf terminal if all of its siblings are leaves. We define a *partially complete tree of degree m* as a tree where all terminal leaves have at least $m - 1$ siblings.

Theorem 9. *Let* $q, \delta > 0$*. Given a partially complete ordered tree* R *of degree at least* $\log_{\frac{1}{q}}(nT(n, \delta))$ *on* n *nodes, , we can reconstruct* R *using* $T(n, \delta)$ *traces with probability at least* $1 - \delta$*.*

Proof. Sample $T(n, \delta)$ traces from R. For TED trace R', construct two binary strings $s_0^{R'}, s_1^{R'}$ from a pre-order traversal of R'. $s_0^{R'}$ uses the alphabet $\{0, 2\}$ and $s_1^{R'}$ uses the alphabet $\{2, 1\}$. $s_1^{R'}$ is constructed by writing a 1 when the traversal moves down an edge, and a 2 when the

traversal encounters a leaf. Analogously $s_0^{R'}$ is constructed by writing a 0 when the traversal moves up an edge, and a 2 when the traversal encounters a leaf.

Because of the assumption that R is partially complete, for any trace the probability that there exists a node with all children deleted is less than $\frac{1}{nT(n,\delta)}$. By the union bound the probability that any trace contains a node where all children are deleted is less than $1/n$.

R has two unknown strings s_0 and s_1 constructed analogously to the trace strings above. It is easy to see that a node deletion in R corresponds to a single bit deletion in s_0 and s_1 where the ordering among the remaining bits is maintained. This property holds for multiple deletions as well. By the argument above, with high probability no node will have all its children removed. Because only one edge is represented by each bit in the two strings, only one edge is removed in a TED deletion, and all children of a leaf are not deleted with high probability, we conclude that with high probability deletions in s_0 and s_1 behave as *i.i.d.* deletions in the string deletion channel model. □

3.4.3 Trees with Leaf Labels

The following theorem shows certain labellings allow for a natural reduction from tree reconstruction to string reconstruction. We use the labelling set $\{L, I\}$ in place of $\{0, 1\}$ below.

Theorem 10. *For any ordered tree* R *on* n *nodes, and* $q, \delta > 0$*, if the leaves of* R *have label 'L' and internal nodes have label 'I', under the TED deletion channel a.a.s. we can reconstruct* R *and its labelling using* $\exp(\tilde{O}(n^{1/5}))$ *traces.*

Proof. The proof is similar to Theorem 9. Sample $\mathsf{T}(\mathfrak{n}, \delta)$ TED traces generated from R. For TED trace R' construct two binary strings $s_0^{\mathsf{R}'}$, $s_1^{\mathsf{R}'}$ from a pre-order traversal of $\mathfrak{m}$. $s_0^{\mathsf{R}'}$ uses the alphabet $\{0, 2\}$ and $s_1^{\mathsf{R}'}$ uses the alphabet $\{2, 1\}$. $s_1^{\mathsf{R}'}$ is constructed by writing a 1 when the traversal moves down an edge and neither endpoint has value 'L', and a 2 when the traversal encounters an 'L' node. Analogously $s_1^{\mathsf{R}'}$ is constructed by writing a 0 when the traversal moves up an edge and does not encounter an 'L' node, and a 2 when the traversal encounters an 'L' node.

A single node deletion in R corresponds to a single bit deletion in $s_0^{\mathsf{R}'}$ and $s_1^{\mathsf{R}'}$ where the ordering among the remaining bits is maintained per lemma 9. Further these deletions are *i.i.d.* As ordering of nodes is invariant under deletions, independence would only be violated if a bit is flipped in the two trace encoding. A bit is flipped if and only if it represents an internal node in R, and that node became a leaf in the trace. However when we transcribed $s_0^{\mathsf{R}'}$ and $s_1^{\mathsf{R}'}$, the states of the nodes were captured by the node labels, therefore even if an internal node becomes a leaf, the proper bit is transcribed. Therefore bit deletions are *i.i.d.* and so behave as in the string deletion channel model. Applying (39) to the two sets of pre-order strings gives the desired result. □

3.5 Combinatorics of String Reconstruction

This section introduces new combinatorial and probabilistic identities for the string deletion channel. Subsection 3.5.2 studies the distribution of traces of a fixed length. A combinatorial identity is given for the expectation of any function over this distribution. Subsection 3.5.1 gives a generating function expansion for the probability that the $\mathfrak{i}$'th bit is a 1 or a 0.

3.5.1 Infinite Strings

We can discuss the deletion channel applied to infinite strings. This could for instance model a continuous datastream where bits are lost with some probability. For an infinite string $t \in \{0,1\}^\infty$, define $\mathcal{D}_t : \{0,1\}^\infty \to \mathbb{R}$ as the distribution on infinite strings obtained by removing each bit from t independently with probability q. Note the probability the deletion process generates a finite string with probability 0 so we can restrict the domain to infinite strings.

For a string $s \in \{\pm 1\}^n$, let $s^{(i)}$ be the i'th bit of s, with indexing starting from 0. Define the event $A_j = \{x \in \{0,1\}^\infty : x^{(j)} = 1\}$. We can exactly characterize the probability of A_j based on the j'th derivatives of the generating function of the parent string defining the distribution:

Theorem 11. *Let $t \in \{0,1\}^\infty$, and suppose the 1's occur at indices $\mathcal{I} \subset \mathbb{N}$. Consider the generating function for t, $f(t;x) := \sum_{i \in \mathcal{I}} x^i$. Then,*

$$\mathbb{P}_{\mathcal{D}_t}[A_j] = \frac{1}{j!}(p)^{j+1} \frac{\partial^j f(t;\cdot)}{\partial x^j}\bigg|_{(1-p)}$$

Proof. $\frac{\partial^j f(k;x)}{\partial x^j} = \sum_{i \in \mathcal{I}} \frac{(i)!}{(i-j)!} x^{i-j}$. Also, $\mathbb{P}_{\mathcal{D}_t}[A_j] = \sum_{i \in \mathcal{I}} \binom{i}{j} p^{j+1}(1-p)^{i-j}$ by inspecting each 1 in t and noting the probability it ends up at position j. For every $i \in \mathcal{I}$, there are exactly $\binom{i}{j}$ ways for the bit to end up at position j, so exactly $i-j$ bits are removed with probability $(1-p)$, and

$j+1$ bits including i are retained with probability p. Further notice that no bit $\mathcal{I} \ni i < j+1$ contributes anything to the sum, and by convention $i < j \implies \binom{i}{j} = 0$. Therefore,

$$\begin{aligned}\mathbb{P}_{\mathcal{D}_t}[A_j] &= \frac{1}{j!}(p)^{j+1} \sum_{i \in \mathcal{I}} \frac{i!}{(i-j)!}(1-p)^{i-j} \\ &= \frac{1}{j!}(p)^{j+1} \frac{\partial^j f(k;\cdot)}{\partial x^j}(1-p)\end{aligned}$$

□

When the string t is well structured we can hope to evaluate the j'th derivative of the generating function explicitly, as the next corollary shows.

Corollary 3. *Let* $s = (1,0)^\infty$. *Then,*

$$\mathbb{P}_{\mathcal{D}_s}[A_n] = \frac{1}{2}\left(1 - \left(-\frac{p}{q-2}\right)^{j+1}\right)$$

Proof. Using the identity that $\sum_{k=0}^{\infty} x^{2k} = \frac{1}{1-x^2}$ corresponds to the infinite series $1,0,1,0,...$ and differentiating both sides j times yields

$$\sum_{k=0}^{\infty} \frac{2k!}{(2k-j)!} x^{2k-j} = \frac{j!}{2}\left(\frac{1}{(1-x)^{j+1}} - \frac{1}{(-1-x)^{j+1}}\right)$$

Therefore via Theorem 11,

$$\begin{aligned}\mathbb{P}_{\mathcal{D}_s}[A_n] &= \frac{1}{j!}p^{j+1}\frac{j!}{2}\left(\frac{1}{p^{j+1}} - \frac{1}{(q-2)^{j+1}}\right) \\ &= \frac{1}{2}\left(1 - \left(\frac{p}{q-2}\right)^{j+1}\right)\end{aligned}$$

□

Corollary 3 shows that the probability an individual bit in a trace is 1 converges to $1/2$ as the index of the bit increases, so much of the information about the string is contained in the prefixes of the traces.

For this particular string $s = (1,0)^\infty$, if $p = q = 1/2$ we can find a much simpler description of the distribution $\mathcal{D}_s$. Consider a 2-state Markov chain with states 'flip' and 'stay' ($\mathcal{F}$ and $\mathcal{S}$ respectively). The transition from $\mathcal{F}$ to $\mathcal{S}$ has weight 1/3, the transition from $\mathcal{S}$ to $\mathcal{F}$ has weight 2/3, and the self-weights are such that the out-edges have sum 1 (see Figure 4). Now start with the string $r = (1)$, and start a random walk on the Markov chain starting from state $\mathcal{F}$. At discrete time i read bit $r[i]$. If the chain is at state $\mathcal{F}$, append $r[i] - 1 \mod 2$ to r. Else the chain is in state $\mathcal{S}$, so append $r[i]$ to r. That is, if in state $\mathcal{F}$ append the flipped bit, otherwise append the same bit to r. It's not difficult to see the same probability in Corollary 3 applies to this chain by a simple inductive argument.

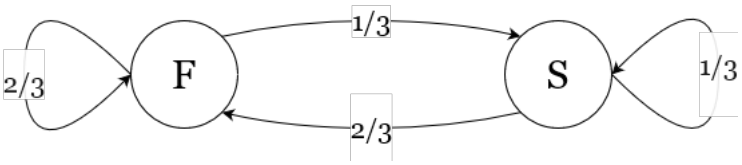

Figure 4: The Markov chain under consideration.

3.5.2 Fixed Length Traces

Throughout we will assume the deletion rate $q = \frac{1}{2}$ unless otherwise stated. Let $s^{i\to j}$ be the substring of s starting at the i'th bit and ending at the $j-1$'th bit. Extend this notation so that $s^{i\to} := s^{i\to n}$. Let $k < n$ represent the index of a bit in s. If $I \in \{0,1\}^n, |I| \leq k$, define $s[I]$ as a string in $\{\pm 1\}^{|I|}$ as the subsequence of s on indices where $I = 1$. I can be thought of as the 'indexing' set where bits from s are extracted.

For $s \in \{\pm 1\}^n$, let $\mathcal{D}_s$ be the distribution over binary strings $t \in \{\pm 1\}^{\leq n}$ induced by the deletion channel with $q = p = 1/2$. Let $R(s,t) = |\{t \text{ is a subsequence of } s\}|$; then $\mathcal{D}_s(t) = \frac{1}{2^n} R(s,t)$ is the measure of t under $\mathcal{D}_s$.

Define $\mathcal{Z}(s,k) = \sum_{t\in\{\pm 1\}^k} R(s,t)$. Then let $\mathcal{D}_s^k(x) = \frac{1}{\mathcal{Z}(s,k)} \cdot R(s,x) \cdot \mathbb{1}\{|x| = k\}$ be the restriction of $\mathcal{D}_s$ to only those traces with length k. ($\mathcal{Z}$ is just a normalizing function).

We restate a commonly known combinatorial identity for the deletion channel, see for example (50) for more detail.

Observation 1. $R(s,t) = R(s^{1\to}, t) + \mathbb{1}\{s[0] = t[0]\} \cdot R(s^{1\to}, t[1:])$

We also have $\mathcal{Z}(s,k) = \binom{n}{k}$. To see this, each trace is obtained by removing $n-k$ bits from s, and summing over all traces there are exactly $\binom{n}{n-k} = \binom{n}{k}$ ways of removing the bits.

It is desirable to see on average how often a trace appears over all input strings of a given length. That is, letting $\mathcal{D}^k := \mathop{\mathbb{E}}_{s\sim\mathcal{U}(\{\pm1\}^n)}[\mathcal{D}^k_s]$. This turns out to be the uniform distribution.

Lemma 10. *$\mathcal{D}^k$ is a distribution on $\{\pm1\}^k$ and is equal to the uniform distribution $\mathcal{U}(\{\pm1\}^k)$*

Proof. Let $x \in \{\pm1\}^k$. Then,

$$\begin{aligned}
\mathcal{D}^k(x) &= \frac{1}{2^n}\sum_{s\in\{\pm1\}^n}\left(\frac{R(s,x)}{\mathcal{Z}(s,k)}\right)\\
&= \frac{1}{2^n}\sum_{s\in\{\pm1\}^n}\frac{R(s,x)}{\binom{n}{k}}\\
&= \frac{1}{2^n\binom{n}{k}}\sum_{s\in\{\pm1\}^n}R(s,x)\\
&= \frac{1}{2^n\binom{n}{k}}\frac{2^n\binom{n}{k}}{2^k} = \frac{1}{2^k}
\end{aligned}$$

□

Instead of taking expectations over the uniform distribution, we may want to take expectations over a particular $\mathcal{D}^k_s$. We generalize lemma 10 by taking the expectation over real-valued functions. If $h : \{\pm1\}^k \to \mathbb{R}$ is real valued, we can define $h^s_I := h(s[I], \cdot, \cdot, ..., \cdot)$ as a function from $\{\pm1\}^{k-|I|}$ to the reals. This is just replacing the first $|I|$ arguments of h by $s[I]$, and allowing the unfilled arguments to vary. We have the following recursive identity.

Theorem 12. *Let $h : \{\pm1\}^k \to \mathbb{R}$ be any function, and $s \in \{\pm1\}^n$ any string. Then for every $0 \le k \le n$,*

$$\mathbb{E}_{\mathcal{D}_s^k}[h] = \frac{1}{\binom{n}{k}} \sum_{I \in \{0,1\}^k} \left(\binom{n-k}{k-|I|} \cdot \mathbb{E}_{\mathcal{D}_{s^{k\rightarrow}}^{k-|I|}}[h_I^s] \right)$$

Proof. Recall Observation 1. Then,

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}_s^k}[h] &= \sum_{x \in \{\pm 1\}^k} h(x) \cdot \frac{R(s,x)}{\mathcal{Z}(s,k)} \\
&= \frac{1}{\mathcal{Z}(s,k)} \sum_{x \in \{\pm 1\}^k} h(x) \big(R(s^{1\rightarrow}, x) \\
&\quad + \mathbb{1}\{s[0] = x[0]\} \cdot R(s^{1\rightarrow}, x[1:]) \big) \\
&= \frac{\mathcal{Z}(s^{1\rightarrow},k)}{\mathcal{Z}(s,k)} \mathbb{E}_{\mathcal{D}_{s^{1\rightarrow}}^k}[h] + \sum_{y \in \{\pm 1\}^{k-1}} h_1^s(y) R(s^{1\rightarrow}, y)) \\
&= \frac{\mathcal{Z}(s^{1\rightarrow},k)}{\mathcal{Z}(s,k)} \mathbb{E}_{\mathcal{D}_{s^{1\rightarrow}}^k}[h] + \frac{\mathcal{Z}(s^{1\rightarrow},k-1)}{\mathcal{Z}(s,k)} \mathbb{E}_{\mathcal{D}_{s^{1\rightarrow}}^{k-1}}[h_1^s]
\end{aligned}$$

Recursively applying this equality 2^k times proves the lemma. That is, each leaf of the complete binary tree of depth k corresponds to one term in the sum in the theorem statement. □

In particular Theorem 12 gives a way to compute the expected value of a real-valued function h over the distribution D_s^k. This evaluation only involves the substring $s^{k\rightarrow}$ and its traces. The values of the first k bits of s are then used when evaluating h_1^s. This could be useful for doing discrete Fourier analysis on the string trace distribution.

We conclude this chapter with the open questions: Do there exist other strings with simple distribution descriptions (clearly $(1)^\infty$ does)? Could we use these as a tool for proving lower bounds on the string reconstruction problem?

CHAPTER 4

DYNAMIC INTERACTIVE LEARNING

4.1 Introduction

The problem of recommending products or media is ubiquitous in many practical settings such as search engines, online marketplaces, or media streaming services (e.g. Google search, Amazon, Spotify, etc.). In such settings any algorithm that tries to optimize recommendations receives implicit feedback from users in the system. This feedback is then used to refine future queries.

Drawing inspiration from earlier work on query learning by (12), as well as more recent models for interactive clustering (51; 52; 53), Emamjomeh-Zadeh and Kempe (54) considered such product recommendation problems from the perspective of combinatorial learning, where specific orderings of recommendations are nodes in a (very large) digraph. In this graph there is a distinguished node that corresponds to the ordering that the learner wishes to discover. This can be thought of as the 'ideal' ordering of products in a marketplace, or the 'best' sequence of recommendations in a streaming service. A directed edge exists from node s to s' if the user might propose s' as a response to s. For example, a user might want to swap two items from an ordered list s to get the more preferred ordering s'. If the learner proposes a node and that is not the target, it receives noisy (random (54) or even adversarial (55)) feedback in the form of

an edge on the shortest path from the proposed node and the target. This form of feedback is similar to the correction queries from query learning (56).

(57) subsequently considered cases when the combinatorial structure itself can evolve over time – as they noted, some of these settings resembled earlier work on shifting bandits (58). These dynamic settings are also where our results lie, and among our results, we generalize the work of (57) and solve some of their open problems herein.

4.2 Preliminaries

(54) first introduced a static graph model for robust interactive learning, where there is one fixed concept in the concept class, and the learner is trying to learn under noisy feedback. Later (57) extended the model to dynamic interactive learning, where the target concept can change during learning. Our work is based on the same framework, so we will briefly describe previously defined models and results here.

4.2.1 Static model

For clarity we first state the static learning model from (54), as it is a foundation for later work on dynamic models.

Definition 8 (Feedback graph (54))**.** *Define a weighted (directed or undirected) graph* $\mathsf{G} = (\mathsf{V}, \mathsf{E}, w)$*, where the vertices represent a set of* $\mathfrak{n} = |\mathsf{V}|$ *candidate concepts. The edge set* E *captures all possible corrections a learner can receive: edge* $(\mathsf{s}, \mathsf{s}')$ *exists if the user is allowed to propose* s' *in response to* s*. The edge weights* w *are given to the learning algorithm, satisfying a key property: if the learner proposes* s *and the ground truth (target) is* $\mathsf{t} \neq \mathsf{s}$*, then every correct user feedback* s' *lies on a shortest path from* s *to* t *with respect to edge weight* w*.*

Note that we assume that the weighted graph is given to the algorithm and faithfully represents the underlying problem.

For an undirected graph G, let $N_G(v)$ denote the neighborhood of v in G. For a directed graph G, let $N_D^{in}(v)$ be the in-neighborhood of v in digraph D (including self loops) and $N_D^{out}(v)$ be the out-neighborhood of v in digraph D (including self loops).

In the static model there exists a fixed target vertex $t \in V$ that the algorithm is attempting to learn over multiple rounds. In each round the learner proposes a query vertex $q \in V$ and receives a feedback vertex z which is noisy with probability p. Specifically, if $q = t$, with probability $1 - p$ the learner receives feedback q indicating the query is correct, and with probability p it receives an incorrect feedback z which is adversarially chosen from $N_G(q)$; if $q \neq t$ the algorithm is given a feedback $z \in N_G(q)$ which is incorrect with probability p. Crucially both correct and incorrect feedback is adversarial. As discussed in (54) this implies learning is only feasible when $p < 1/2$.

Other important definitions used throughout include the collection of concepts that are consistent with a particular feedback, or the version space for a query-feedback pair, as well as the weighted median of the feedback graph, which can be interpreted as the 'center of mass' of the graph.

Definition 9 (Version space (54))**.** *If the learner proposes* q *and receives feedback* z*, let* $S_G(q, z)$ *be the collection of concepts (nodes) that are consistent with the feedback. Formally,* $S_G(q, z) = \{v \mid z$ *lies on a shortest weighted path from* q *to* $v\}$.

Definition 10 (Weighted median (59)). *Let* $L : V \to \mathbb{R}^{\geq 0}$ *be a function that assigns likelihood to every vertex in the feedback graph* $G = (V, E, w)$. *A weighted median* u *is a vertex that minimizes* $\sum_{v \in V} L(v) \cdot w(u, v)$.

(54) presented a multiplicative weight update algorithm, which assigns likelihoods for each vertex in the feedback graph, and repeatedly queries the weighted median, which has the property of halving the total likelihood of its version space each round.

4.2.2 Dynamic model

Our paper is concerned with dynamic interactive learning where the target t is allowed to move. Assume that over the R rounds of learning the target moves at most B times and at round r the target is located at some node t_r. Without further assumption on target evolution, (57) showed the following general mistake upper bound. For the remainder of the paper, denote the entropy as $H(p) = p \log \frac{1}{p} + (1 - p) \log \frac{1}{1-p}$.

Theorem 13 ((57)). *Assume the total number of rounds* R *is known beforehand. Let* $A = V^R$ *be the set of all node sequences of length* R *and let* $a^* = \langle t_1, \ldots, t_R \rangle$ *be the sequence of true targets throughout the* R *rounds. Let* $\lambda : A \to \mathbb{R}^{\geq 0}$ *be a function that assigns non-negative weights to these sequences, such that* $\sum_{a \in A} \lambda(a) \leq 1$. *There is an online learning algorithm that makes at most*

$$\frac{1}{1 - H(p)} \cdot \log \frac{1}{\lambda(a^*)}$$

mistakes in expectation.

While this is a positive result for the mistake bound, it does not guarantee an efficient algorithm as it has to keep track of weights for all sequences V^R, and in the worst case the number of sequences is $O(n^R)$ where $n = |V|$. Relatively efficient implementations exist for the following two models without explicitly constructing the λ map for each sequence.

4.2.2.1 Shifting target

In the Shifting Target model there exists an unknown subset of vertices $S \subseteq V$ where $|S| \leq k$, and the learner knows k. Target transition is restricted within S, viz. $t_r \in S$ for every round r. Previous work by (57) proved the following theorem.

Theorem 14 ((57)). *Under the Shifting Target model, there is a deterministic algorithm that runs in time* $O(n^k \mathrm{poly}(n))$ *and makes at most*

$$\frac{1}{1 - H(p)} \cdot (k \log n + (B + 1) \log k + R \cdot H(B/R))$$

mistakes in expectation. Furthermore, there exists a graph such that every algorithm makes at least

$$\min \left\{ \frac{1}{1 - H(p)} \cdot [k \log n + (B - 2k + 1) \cdot (\log k)] - o(\log n) - B \cdot o(\log k) \, , \, R - o(R) \right\}$$

mistakes in expectation.

Note in particular the exponential dependence on k in the runtime of the algorithm.

4.2.2.2 Drifting target

In the Drifting Target model, it is assumed that the target slowly evolves over time, and the evolution can be modeled by following the edges of some transition graph defined below.

Definition 11 (Transition graph (57))**.** *There exists a known unweighted digraph* $G' = (V, E')$*, where in particular* G' *and* G *have the same vertex set but their edge sets can be different. The target is only allowed to move along edges of* G'*. Formally for every round* r *the model requires* $t_{r+1} \in \{t_r\} \cup N^{out}_{G'}(t_r)$*. Let* Δ *be the maximum degree in* G'*.*

Previous work by (57) proved the following theorem.

Theorem 15 ((57))**.** *Under the Drifting Target model, there is a deterministic algorithm that runs in time* $poly(n)$ *and makes at most* $\frac{1}{1-H(p)} \cdot (\log n + B \cdot \log \Delta + R \cdot H(B/R))$ *mistakes in expectation..*

Furthermore, there exists a graph such that every algorithm makes at least

$$\min\left\{R - o(R), \frac{1}{1-H(p)} \cdot (\log n + B \log \Delta) - o(\log n) - B \cdot o(\log \Delta)\right\}$$

mistakes in expectation.

Notice that for both Shifting Target and Drifting Target models, there is a gap of $R \cdot H(B/R)$ between the mistake upper bound and lower bound, which remains an open problem in (57). We will show a new lower bound and close the gap in Section 4.

4.3 A unified model

Our first contribution is to define a more generalized model inspired by the original Drifting Target model, and show that the results from Theorems 14 and 15 are both valid under this generalization, thus unifying previous models. The technique used in (57) to efficiently implement the Shifting Target model was to keep track of likelihoods for each subset of nodes instead of each sequence, reducing computational complexity from $O(n^R)$ to $O(n^k)$. Similarly, their efficient implementation for the Drifting Target model keeps track of likelihoods for each node, reducing computational complexity to $O(\mathtt{poly}(n))$. This method requires customization for each transition model, and can become tricky as the models become more complicated. A key motivation for a general model is that it unifies a wide class of transition models, and allows us to easily obtain mistake upper bounds and runtime guarantees based on a single algorithm.

As above let $G = (V, E, w)$ be the graph representing the candidate models (the feedback graph), and $G' = (V', E', \pi)$ be a directed transition graph, representing all possible ways the target might change over time. The key difference is that for each vertex $i \in V$, V' contains possibly duplicated vertices corresponding to the same vertex i, denoted by $V'_i := \{u \in V' \mid u$ corresponds to $i \in V\}$. Define $n' = |V'|$, Δ' as the max degree of G', and π_{ij} as the transition probability in G', where $\pi_{ii} = (1 - b)$ with $b = \frac{B}{R}$, and $\pi_{ij} = \pi_{out} := \frac{b}{\Delta'}$ for $i \neq j$ under a uniform transition assumption.

We present a modified version of the algorithm from (57). In the r^{th} round, we keep track of the likelihoods for each vertex $u \in V'$ as $L'_r(u)$, and for each vertex $i \in V$ its likelihood $L_r(i)$ is aggregated from L'_r as the summation over all of i's duplicates in V'. The median of the feedback

Algorithm 1 Interactive learning likelihood update

Initialize $L'_1(u)\ \forall u \in V'$
for $1 \le r \le R$ **do**
  $\forall i \in V : L_r(i) \leftarrow \sum_{u \in V'_i} L'_r(u)$ {Aggregate L_r from L'_r over the duplicates}
  $q_r \leftarrow \arg\min_{i \in V} \sum_{j \in V} L_r(j) \cdot w(i,j)$ {Query the weighted median}
  $z_r \leftarrow$ feedback from adversary
  $\forall i \in V$and $\forall u \in V'_i$:
    $P(i) \leftarrow (1-p) \cdot \mathbb{1}[i \in S_G(q_r, z_r)] + p \cdot \mathbb{1}[i \notin S_G(q_r, z_r)]$ {Weight update multiplier for i}
    $P(u) \leftarrow P(i)$ {Same weight update for each duplicate u of i}
  $\forall u \in V' : L'_{r+1}(u) = \sum_{v \in N^{in}_{G'}(u)} P(v) \cdot L'_r(v) \cdot \pi_{vu}$ {Apply weight update and transition}
end for

graph G is then calculated based on L_r. We update the likelihoods L'_{r+1} for all corresponding nodes in G' based on each node's consistency with the feedback using the same rules as (57).

Theorem 16. *Assuming the first target is chosen uniformly at random from V', Algorithm 1 runs in time $O(\Delta' \cdot n' + \mathrm{poly}(n))$, uses space $O(n')$, and has query complexity*

$$\frac{1}{1 - H(1-p)} \cdot \Big(\log n' + B \cdot \log \Delta' + R \cdot H(B/R) \Big).$$

Alternatively writing the bound using transition probabilities instead of maximum degree, Algorithm 1 runs in time $O\left(\frac{1}{n'} \cdot \pi_{out}^{B} \cdot (1-b)^{R-B}\right)$, uses space $O(n')$, and has query complexity

$$\frac{1}{1 - H(1-p)} \cdot \Big(\log n' + B \cdot \log(b/\pi_{out}) + R \cdot H(b) \Big).$$

Proof. We wish to show that the likelihood of the ground truth sequence a^* is at least

$$\lambda(a^*) = \frac{1}{n' \cdot \Delta'^B \cdot \binom{R}{B}},$$

or alternatively

$$\lambda(a^*) = \frac{1}{n'} \cdot \pi_{\text{out}}^B \cdot (1-b)^{R-B}.$$

Note that these two expressions correspond to two equivalent interpretations of the transition model: 1, the target changes at most B times during R rounds; 2, the target changes with probability at most $b = B/R$ at each round.

For the first interpretation, we provide the expression for $\lambda(a^*)$ with a combinatorics argument: there are n' choices for the first node, and the next node differs from the previous node at most B times, each time with Δ' choices, and these changes can occur at $\binom{R}{B}$ locations in the sequence. Thus the total number of valid sequences is $n' \cdot \Delta'^B \cdot \binom{R}{B}$. The initial likelihoods are assigned uniformly among all sequences, so dividing 1 by the total number of sequences gives us $\lambda(a^*)$.

For the second interpretation, we have a probability argument: the sequence starts with any particular node in the transition graph with probability $\frac{1}{n'}$, and the next node changes to one of the neighbors with probability $\pi_{\text{out}} = \frac{b}{\Delta'}$ for B times, and stays the same with probability $1-b$ for $R-B$ times. Taking the product of these probabilities gives the result.

The bound on query complexity follows by substituting $\lambda(a^*)$ into Theorem 5 of (57). Note that for large R we approximate $\binom{R}{B}$ by $\left(\frac{R}{B}\right)^B \cdot \left(\frac{R}{R-B}\right)^{(R-B)}$, which contributes to the term $R \cdot H(B/R) = R \log R - B \log B - (R-B)\log(R-B)$ after taking logarithm.

For algorithmic complexity, steps 3 (aggregating likelihoods) and 6 (computing weight multiplier) both take time $O(n')$, step 4 (computing median) takes time $O(n^3)$, and step 7 (updating weight and transition) takes time $O(\Delta' \cdot n')$, and L_r, L'_r takes space n and n' respectively. □

Under our generalized model, any dynamic interactive learning problem can be reduced to defining the feedback graph G to represent the concept class, and defining the transition graph G' to represent the concept evolution. Specifically, the Shifting Target model and Drifting Target model studied in the original paper can be shown as special cases under this general model, and we will show that the general bounds agree with the original results.

Corollary 4. *In the Drifting Target model, Algorithm 1 runs in time* $O(\Delta \cdot \mathrm{poly}(n))$*, uses space* $O(n)$*, and makes at most*

$$\frac{1}{1 - H(1-p)} \cdot \Big(\log n + B \cdot \log \Delta + R \cdot H(B/R) \Big)$$

mistakes in expectation.

Proof. The transition graph G' is the same as the feedback graph G, and transition probability is assumed to be uniform. Thus $n' = n$, and $\Delta' = \Delta$. Plugging into Theorem 16 gives the result. □

Corollary 5. *In the Shifting Target model, Algorithm 1 runs in time* $O(k^2 \cdot n^k)$*, uses space* $O(k \cdot n^k)$*, and makes at most*

$$\frac{1}{1 - H(1-p)} \cdot \Big(k \cdot \log n + (B+1) \cdot \log k + R \cdot H(B/R) \Big)$$

mistakes in expectation.

Proof. The transition graph G' consists of $\binom{n}{k}$ disconnected sub-graphs, where each sub-graph is a clique of size k, corresponding to a subset of k vertices in V. Each round the target might shift within a k-clique, and each clique represents a possible choice of the k-subset of targets. Thus $n' = \binom{n}{k} \cdot k$ and $\Delta' = k$. Plugging this into Theorem 16 gives the result. □

Since the query and computational upper bounds mostly depend on the size of transition graph, namely n' and Δ', minimality of the transition graph is crucial for query and computational efficiency. We want to find the worst case query upper bound, which can be used as a benchmark when modeling various types of transitions. A trivial upper bound on query complexity occurs in the case that the learner does not have any information about how target might change over time, thus the transition graph G' is a complete graph on $n' = n$ vertices, and $\Delta' = n$. Plugging into Theorem 16 gives the following result.

Corollary 6. *The worst case query complexity using Algorithm 1 is*

$$\frac{1}{1 - H(1-p)} \cdot \Big((B+1) \cdot \log n + R \cdot H(B/R) \Big),$$

and runs in time $O(\mathrm{poly}(n))$ *and space* $O(n)$.

In the following sections, we will discuss a few examples of other transition models, showing a hierarchy of query complexity.

4.3.1 Shortest path

Given two vertices $s, t \in G$, define $S_G^B(s,t)$ to be the collection of all subsets of $S_G(s,t)$ that contain at most B vertices. Formally, $S_G^B(s,t) = \{H \subseteq S_G(s,t) : |H| \leq B\}$. In the Shortest Path model we insist the target can only move along a shortest path in G.

We can describe this model in the language of our generalized framework. The transition graph G' consists of many disconnected directed paths, each corresponding to some element of $S_G^B(s,t)$ for some $s, t \in G$. This procedure overcounts, so we also restrict G' to only include one copy of any subset of vertices in a path in G. Finally the vertices in any subgraph of G' are connected with $B-1$ arcs that correspond to the ordering imposed by traversing $S_G(s,t)$ from s to t.

The number of vertices in G' is bounded as $n' \leq B \cdot \binom{n}{B}$. We can't hope to do better than this, as there are classes of graphs with exponentially many shortest paths between two distinguished vertices. The maximum degree of G' is 2, as all disconnected components are paths.

This model is a variation of the Shifting Target model. If the target can move B times, then the target can only move in one direction in each valid path. We can still apply Thm. 16 and get a naive mistake upper bound that runs in time $n^B \cdot \mathrm{poly}(n)$. We can achieve the $\binom{n}{B}$ bound on the number of subsets when G is a path with n vertices. However, the target can only move

in one direction along the path, so it's natural to think a better algorithm can be developed at least for this case.

Corollary 7. *In the Shortest-path model, Algorithm 1 runs in time* $O(n^B)$*, uses space* $O(B \cdot n^B)$*, and makes at most*

$$\frac{1}{1 - H(p)} \cdot (B \cdot \log n + (B + 1) \cdot \log B + R \cdot H(B/R))$$

mistakes in expectation.

4.3.2 m-Neighborhood

Let $N_G^m(v)$ denote the set of vertices in G that have a shortest path of length m to v. In the m-Neighborhood model, the target can move within N_G^m, and m is known to the learner. This model is a variation of the Drifting Target model, and note that $m = 1$ is exactly the case when $G = G'$ in the original Drifting Target model. The transition graph G' is constructed by including an arc from every $v \in V$ to every node in its m-Neighborhood. Note that $n' = n$ and $\Delta' \leq \Delta^m$. Applying Theorem 16 gives the following mistake bound for the m-Neighborhood model:

Corollary 8. *In the m-Neighborhood model, Algorithm 1 runs in time* $O(\Delta^m \cdot n)$*, uses space* $O(n)$*, and makes at most*

$$\frac{1}{1 - H(p)} \cdot (\log n + B \cdot m \cdot \log(\Delta) + R \cdot H(B/R))$$

mistakes in expectation.

To complete our hierarchy, in descending query complexity, we have: Shortest Path model, the original Shifting Target model, the m-Neighborhood model, and the original Drifting Target model.

4.4 Query complexity lower bound

In this section we close the gap between upper and lower bounds on query complexity, which remained an open problem in (57). We show a query complexity lower bound that matches the upper bound asymptotically.

Our result requires some background on the noisy binary search problem. Here there is a distinguished integer t from the set $\{1, ..., m\}$. In each round r, the learner queries some integer x. If $x = t$ then the item has been found and the procedure stops. Otherwise with probability $1 - p$ the learner receives correct feedback of the form $x > t$ or $x < t$. We make use of the following lower bound.

Theorem 17 ((57))**.** *Every algorithm for the noisy binary search problem requires at least* $\frac{\log m}{1-H(p)} - o(\log m)$ *queries in expectation.*

The idea of our proof is to establish a reduction from noisy binary search: given a noisy binary search problem where the target is uniformly random among m items, we will reduce it to a specific Drifting Target problem under our dynamic interactive learning model. Thus the lower bound on noisy binary search is also a lower bound on interactive learning.

Theorem 18. *For every n and Δ', there exists a Drifting Target problem such that every algorithm makes at least*

$$\frac{1}{1-H(1-p)} \cdot \Big(\log n + B \cdot \log \Delta' + R \cdot H(B/R)\Big) - o\left(\log n + B \cdot \log \Delta' + R \cdot H(B/R)\right)$$

mistakes in expectation.

Proof. We reduce a noisy binary search problem over m items to an interactive learning problem defined in the following way: choose n and R such that $m \leq n^R$ so that each of the m items can be represented as a base n encoding/enumeration of a sequence with R digits. The feedback graph G for the learning problem is a simple path on n vertices, ordered in the same way as in the encoding: the left end is the smallest digit while the right end is the largest. The transition graph G' is defined on the same set of vertices as in G so $n' = n$. G' includes all the edges in G as bi-directional edges, potentially with additional edges up to some degree Δ'.

For example, to search among $m \leq 1000$ items, we can choose $n = 10$ and $R = 3$, so that each item can be encoded by a length-3 sequence between $\langle 0, 0, 0\rangle$ and $\langle 9, 9, 9\rangle$, which are our familiar base-10 natural numbers up to 999. The graph G consists of vertices $0, 1, 2, \ldots, 9$ on a path, and interactive learning continues for 3 rounds. Suppose the target item is encoded as the sequence $\langle 1, 1, 2\rangle$, then the ground truth target locations during the 3 rounds are vertices $1, 1, 2$ respectively (target shifted once from vertex 1 to 2).

In the r^{th} round of the interaction, the learner queries vertex $i \in [n]$, which can be interpreted as guessing the r^{th} digit of the target's encoding sequence. Without loss of generality, suppose

the adversary's feedback is some vertex j to the left of vertex i, which is interpreted as the r^{th} digit of the target sequence is less than the value i. The learner updates likelihoods for sequences whose r^{th} digit is less than i by a factor of $1-p > \frac{1}{2}$, and the other sequences by a factor of $p < \frac{1}{2}$. According to Lemma 6 from (57), the likelihood of the target item's encoding sequence (ground truth) decreases exponentially more slowly than the rest of the sequences and will eventually prevail.

To establish the lower bound for a Drifting Target problem with arbitrary n and Δ', there exists a noisy binary search problem on $m = n \cdot \Delta'^{B} \cdot \binom{R}{B}$ items that reduces to the Drifting Target problem. The encoding is restricted such that after the first digit is chosen, the remaining digits can change at most B times among Δ' choices (all other sequences are initialized with 0 likelihoods). Plugging in our value of m into Theorem 17 gives a mistake lower bound of $\frac{1}{1-H(1-p)} \cdot \Big(\log n + B \cdot \log \Delta' + R \cdot H(B/R) \Big) - o(\log n + B \cdot \log \Delta' + R \cdot H(B/R))$, as desired. □

4.5 Efficient algorithm for low diameter graphs

While Algorithm 1 emphasizes on bounding the number of mistakes for general interactive learning problems, its computation can be inefficient in each round and deteriorates as the transition model becomes more complex. We realize that the computational complexity mainly comes from keeping track of the likelihoods under all possible transitions, so we consider an alternative approach where the learner ignores the transition model completely and simply follows the adversary's feedback each round. After the initial query, the algorithm requires no computation. In this section, we study this simple algorithm's performance on low diameter graphs. We formally present this algorithm below.

Algorithm 2 'Follow the Feedback' Procedure for Interactive Learning

$q_1 \leftarrow \text{argmin}_{i \in V} \sum_{j \in V} w(i,j)$ {Start with a 'center' vertex}
for $1 \leq r \leq R$ **do**
  $z_r \leftarrow$ feedback from adversary after querying q_r
  $q_{r+1} \leftarrow z_r$ {Follow the feedback for next round}
end for

4.5.1 Cliques: graphs with diameter 1

A clique is the most symmetric graph, where each vertex can be considered the center, and the graph has diameter 1. This means no matter which node the learner queries, a correct feedback from the adversary will reveal the true target at each round. Therefore after each mistake, the learner will keep querying the correct node until the target's next move. The mistake upper bound is stated in the theorem below.

Theorem 19. *If the feedback graph* G' *is fully-connected (a clique on the concept class), Algorithm 2 makes at most* $B + p(R - B)$ *mistakes in expectation.*

Proof. By assumption the learner queries a node then receives a feedback, and the target may move at any point during this process. To help with the analysis in this case, we break down the chain of events in the following way: in each round we assume that at first the target moves (or stays put), then the learner makes a query and receives a new feedback. Notice that in every round where the target moves the learner will make a mistake regardless of the correctness of the previous feedback. If we assume target can move at most B times, this leads to B mistakes. For the $R - B$ rounds where the target doesn't move, the learner makes a mistake if and only if

the previous feedback is incorrect. As feedback is noisy with probability p, this leads to $p(R-B)$ mistakes. So the expected number of mistakes over the course of R rounds is:

$$E[M] = B + p(R - B)$$

In anticipation of discussing other classes of graphs we present a second analysis of Algorithm 2 on cliques. Assume that each round the target moves with probability $b = \frac{B}{R}$. We can model the process as a Markov Chain where the states $\{0, 1\}$ represent the learner's distance from the target at each round. Note that these states do not represent the learner's position in the graph. Now we break down the chain of events in a slightly different way: first, the learner queries the node received from previous feedback and receives a new feedback, then target either moves or stays put for the next round.

The new feedback is correct with probability $1 - p$, and the target stays at the same vertex with probability $1 - b$, so in the next round, the learner queries the correct vertex (transitions to state 0) with probability $(1 - p)(1 - b)$. If either the new feedback is incorrect or the target moves at the end of this round, the learner will make a mistake next round (transitions to state

1) with probability $p + b - pb$, assuming noise of feedback and target evolution are independent. The state transition matrix is:

$$P = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} (1-p)(1-b) & p+b-pb \\ (1-p)(1-b) & p+b-pb \end{pmatrix} \end{array}$$

Each row in the transition matrix is already in its stationary distribution $\pi = (\pi_0, \pi_1)$. The expected number of mistakes over the course of R rounds is: $E[M] = R(1-\pi_0) = B+p(R-B)$. □

4.5.2 Stars: graphs with diameter 2

A simple star graph is a graph of diameter 2, with one center vertex connecting to all the other vertices (leaf nodes). After querying the center vertex, a correct feedback will reveal the true target, so an efficient strategy is to query the center first then follow the feedback. We assume the target only moves among the leaf nodes, because the learner will make no more mistakes in the case that the target can move to the center: if the learner queries a wrong leaf, it takes at least 2 queries if target is on another leaf, and takes 1 query if the target is at the center.

Theorem 20. *If the feedback graph G' is a star then Algorithm 2 makes at most $2B + p(R - B) + p^2(R - B)$ mistakes in expectation.*

Proof. We again break down the chain of events in this order: first, the target either moves or stays put, then the learner queries the previous feedback received and the adversary provides a

new feedback. For the B rounds that the target shifts, the learner will make 1 mistake each time. For the $R-B$ rounds when the target doesn't move, if the previous feedback was incorrect, the learner will make 1 mistake; if the previous feedback was correct, the learner will make a mistake if the feedback pointed to the center, which means the previous query was a wrong leaf. Another case that the feedback points to the center is when the learner queried the correct leaf, but received an incorrect feedback pointing to the center.

Let x, y, z represent the number of times the learner queries the correct leaf, the wrong leaf, and the center respectively. Based on the analysis above, we can set up a system of linear equations:

$$x + y + z = R \tag{4.1}$$

$$px + (1-p)y = z \tag{4.2}$$

$$B + p(R-B) + (1-p)(R-B)(y/R) = R - x \tag{4.3}$$

Equation 4.1 is trivial; equation 4.2 represents the number of times the learner queries the center as a function of queries to correct/incorrect leaf nodes; equation 4.3 is the expected number of mistakes, which is the number of times the learner does not query the correct leaf. After elimination, equation 4.3 becomes:

$$\begin{aligned} E[M] &= B + p(R-B) + \left([B + p^2(R-B)] \cdot \frac{(1-p)(R-B)}{B + p^2(R-B) + (1-p)R} \right) \\ &\leq 2B + p(R-B) + p^2(R-B) \end{aligned}$$

Alternatively, if we assume each round the target moves with probability $b = \frac{B}{R}$, we can model the process using a Markov Chain with states $\{0, 1, 2\}$ representing the learner's distance from the target at each round. Similar to the analysis of the clique, we have state transition matrix:

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} (1-p)(1-b) & p & (1-p)b \\ (1-p)(1-b) & 0 & p+b-pb \\ 0 & 1-p & p \end{pmatrix} \end{array}$$

This is a fully-connected Markov Chain, and the stationary distribution $\pi = (\pi_0, \pi_1, \pi_2)$ can be calculated numerically. The expected number of mistakes over the course of R rounds is $R(1 - \pi_0)$. It can be verified that the numerical solution agrees with the analytical solution above. □

In Appendix 4.8, we extend this analysis for "quasi-stars" with a central vertex connecting otherwise disjoint paths of length $d/2$ (for even d).

4.5.3 Graphs with diameter o(log n)

From our previous analysis, we notice that the mistake bound does not depend on the number of nodes in the feedback graph, but rather the diameter, which is the largest distance from any node to the target. Therefore we consider general graphs bounded by diameter d. The mistake bound is stated as the theorem below.

Theorem 21. *If the feedback graph has diameter* d*, then Algorithm 2 makes at most*

$$\frac{1}{1-p} \cdot \left(dB - \frac{pB}{1-2p} + pR \right)$$

mistakes in expectation.

We model the learning process as a random walk on a Markov Chain with states $\{0, \ldots, d\}$. However, now we reverse the meaning of the states: state 0 means the query node is distance d from the true target, and state d means the query node is the target. This change does not affect the result of analysis, but greatly simplifies the notation. Every time the target moves, the random walk restarts at state 0 and moves towards state d: the learner moves 1 step forward upon every correct feedback, and moves 1 step backward upon every noisy feedback. There are two types of mistakes during the random walk: before reaching the target for the first time, every query contributes a mistake; once the learner reaches the target, it will circle around it due to noise probability $p < 1/2$, and occasionally misses the target.

The first type of mistake is captured by the hitting time of random walk on the Markov Chain from state 0 to state d. We have the following lemma (see Appendix 4.7 for the proof):

Lemma 11. *Let* $p < 1/2$*, for a Markov Chain on a path of length* $d+1$*, the random walk with forward probability* $1-p$ *and backward probability* p *has a hitting time*

$$h_{0,d} \leq \frac{d}{1-2p} - \frac{p}{(1-2p)^2}.$$

Next we consider the second type of mistake. Once the learner reaches the target, it will keep reporting the correct node unless it receives noisy feedback and is misguided to move away from the target, which will cause a mistake for the next query. We bound the fraction of time the learner misses the target with the following lemma.

Lemma 12. *Let* $p < 1/2$*, after reaching state* d *and before the next target transition, the expected fraction of time the learner wanders away from state* d *is bounded by*

$$T_{off} \leq \frac{p}{1-p}.$$

Proof. Once the random walk reaches state d (learner queried the correct target), let T_d denote the expected time spent at state d, we have the following recurrence relations:

$$p \cdot T_d = (1-p) \cdot T_{d-1} \implies T_{d-1} = r \cdot T_d \text{ where } r = \frac{p}{1-p}$$

$$T_{d-1} = (1-p) \cdot T_{d-2} + p \cdot T_d \implies T_{d-2} = r \cdot T_{d-1}$$

$$\dots$$

$$p \cdot T_1 = (1-p) \cdot T_0 \implies T_0 = r \cdot T_1$$

$$\text{For } i = 0 \dots d : T_i = r^{d-i} \cdot T_d$$

The expected fraction of time not spent at state d:

$$T_{\text{off}} = 1 - \frac{T_d}{\sum_{i=0}^{d} T_i} = 1 - \frac{1-r}{1-r^{d+1}} \leq r = \frac{p}{1-p},$$

which finishes the proof. □

Note that the hitting time $h_{0,d}$ is linear in d, and T_{off} is positively related to entropy $H(p)$. Combining the results above, we can prove our theorem:

Proof of Theorem 21. Assume every time the random walk restarts at state 0, state d can be reached before the next restart. This means every time the target moves, the learner is able to reach the target before its next transition. Since the learner makes a mistake every round spent on the hitting time, this is the worst case assumption because the learner is forced to make all the mistakes possible for each target transition. Combining the two types of mistakes from previous lemmas, the total expected number of mistakes is:

$$\begin{aligned} E[M] &= B \cdot h_{0,d} + (R - B \cdot h_{0,d}) \cdot T_{off} \\ &\leq B \cdot \left(\frac{d}{1-2p} - \frac{p}{(1-2p)^2} \right) \cdot \frac{1-2p}{1-p} + \frac{pR}{1-p} \\ &= \frac{1}{1-p} \cdot \left(dB - \frac{pB}{1-2p} + pR \right), \end{aligned}$$

which completes the proof. □

In the case that $d = 2$, the bound in Theorem 21 for a general diameter-2 graph is slightly larger than the bound from Theorem 20 for the star graph. This makes sense because a star is the best case diameter-2 graph, with a center node that when queried provides information to the true target.

In the case that $d = o(\log n)$ and $p = o(H(B/R))$, we notice that the result from Theorem 21 is comparable to the trivial upper bound of Algorithm 1 as stated in Corollary 6. This

means that if the learner has very limited information on target transition, or the transition model is complex, and the graph is bounded by low diameter, then Algorithm 2 makes a huge improvement on computational efficiency without too much sacrifice on query complexity. Note that a complex transition model is often correlated with a low diameter feedback graph: highly connected graphs tend to have low diameters, and potentially complex transitions due to the close relationships between concepts.

4.5.4 Paths: graphs with diameter n

We also note that while path graphs seem like an easy case, they actually present difficulties due to their large diameter.

An upper bound on noisy binary search was given by (60). Their algorithm returns the correct element with probability $(1-\delta)$ with an expected

$$\frac{(1-\delta)}{1-H(p)} \cdot \big(\log n + O(\log\log n) + O(\log(1/\delta))\big)$$

queries. This can be implemented in poly-time.

A naive algorithm for the shifting target case is to run their algorithm k times, setting δ appropriately small, for example, $\delta = 1/\log(kn)$. Then as both k and n go to infinity, the probability of failure goes to 0.

If $k \approx \log(n)$, $B \approx k$, and the number of rounds is much larger than the expected number of queries, this naive algorithm essentially matches the mistake bound from (54). The difference is the $k\log k$ vs. $k^2 \log k$ and $R \cdot H(B/R)$ vs. $\log(\log(kd))$ terms.

4.6 Acknowledgements

This work was supported in part by the National Science Foundation grant CCF-1934915.

4.7 Proof of Lemma 11

Proof. The transition probabilities of the Markov Chain are:

$$P_{ij} = \begin{cases} 1-p & j = i+1 \text{ (move towards target)} \\ p & j = i-1 \text{ (move away from target)} \\ 1-p & j = i = d \text{ (self-loop on the target)} \\ p & j = i = 0 \text{ (self-loop on nodes furthest from the target)} \end{cases}$$

Let $r = \frac{p}{1-p}$. It follows from the assumption $p < \frac{1}{2}$ that $r < 1$. The hitting time analysis follows:

For $i = 1 \dots d$: $h_{i,i+1} = (1-p) \cdot 1 + p \cdot (1 + h_{i-1,i+1}) = 1 + p \cdot (h_{i-1,i} + h_{i,i+1})$

We solve the recurrence: $h_{i,i+1} = \frac{1+p\cdot h_{i-1,i}}{1-p}$ with base cases: $h_{0,1} = \frac{1}{1-p}, \quad h_{1,2} = \frac{1+\frac{p}{1-p}}{1-p} = \frac{1}{(1-p)^2}$.

For $i \geq 2$:

$$
\begin{aligned}
h_{i,i+1} &= \frac{\left(\sum_{j=0}^{i-2}(1-p)^{i-j}\cdot p^j\right) + p^{i-1}}{(1-p)^{i+1}} \\
&= \frac{(1-p)^i \cdot \sum_{j=0}^{i-2}\left(\frac{p}{1-p}\right)^j + p^{i-1}}{(1-p)^{i+1}} \\
&= \frac{1}{(1-p)^{i+1}} \cdot \left((1-p)^i \cdot \frac{1-r^{i-1}}{1-r} + p^{i-1}\right) \\
&= \frac{1}{(1-p)^{i+1}} \cdot \left((1-p)^{i+1} \cdot \frac{1-r^{i-1}}{1-2p} + p^{i-1}\right) \\
&= \frac{1}{1-2p} - \frac{r^{i-1}}{1-2p} + \frac{p^{i-1}}{(1-p)^{i+1}} \\
&= \frac{1}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right)\cdot r^{i-1} \\
h_{2,d} &= \sum_{i=2}^{d-1} h_{i,i+1} \\
&= \frac{d-2}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right)\cdot \sum_{i=2}^{d-1} r^{i-1} \\
&= \frac{d-2}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right)\cdot \frac{r-r^{d-1}}{1-r} \\
&= \frac{d-2}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right)\cdot \frac{p}{1-2p} \\
h_{0,d} &= h_{0,1} + h_{1,2} + h_{2,d} \\
&\leq \frac{1}{1-p} + \frac{1}{(1-p)^2} + \frac{d-2}{1-2p} + \left(\frac{1}{(1-p)^2} - \frac{1}{1-2p}\right)\cdot \frac{p}{1-2p} \\
&= \frac{d-2}{1-2p} - \frac{p}{(1-2p)^2} + \frac{1}{1-p} + \frac{1}{(1-p)^2} + \frac{p}{(1-p)^2(1-2p)} \\
&= \frac{d}{1-2p} - \frac{p}{(1-2p)^2}
\end{aligned}
$$

□

4.8 Quasi-stars: graphs with diameter d

Now we consider a star graph where each branch is a path of length greater than one, and we have diameter $d > 2$. We can generalize the Markov Chain with states $\{0, 1, ..., d\}$, representing the distance from query node to the true target. Further assume that every time the target moves, it moves for a distance of at least 2, with uniform probability of landing at any distance (≥ 2) to the target. The $(d+1)$ by $(d+1)$ transition matrix P can be approximated as follows:

$$P_{ij} = \begin{cases} 0 & j = i-2, \text{ moves 2 steps closer} \\ (1-p)(1-b) & j = i-1, \text{ moves 1 step closer} \\ 0 & j = i, \text{ distance to target does not change} \\ p(1-b) & j = i+1, \text{ moves 1 step further} \\ p' & \text{all remaining probabilities sum to 1 uniformly} \end{cases}$$

With the exception that $P_{00} = (1-p)(1-b)$. For example, for $d = 4$:

$$P = \begin{pmatrix} (1-p)(1-b) & p(1-b) & b/3 & b/3 & b/3 \\ (1-p)(1-b) & 0 & p(1-b) & b/2 & b/2 \\ 0 & (1-p)(1-b) & 0 & p(1-b) & b \\ b & 0 & (1-p)(1-b) & 0 & p(1-b) \\ \frac{(p+b-pb)}{2} & \frac{(p+b-pb)}{2} & 0 & (1-p)(1-b) & 0 \end{pmatrix}$$

With stationary distribution $\pi = (\pi_0, ..., \pi_d)$, we get expected total mistakes as $E[M] = R(1-\pi_0)$, which can be computed numerically.

CHAPTER 5

LEARNING AUTOMATA FROM RANDOM WALKS

5.1 Introduction

Learning from environmental feedback is a ubiquitous problem in algorithms, machine learning, and game theory. We've already seen an example of this in Chapter 4, and this chapter studies a classical problem in this area - learning deterministic finite automata.

For our purposes a deterministic finite automata is described by a tuple $M = (Q, \tau, \gamma, q_0)$. Q is a finite set of states, $\tau : Q \times \{0, 1\} \to Q$ is the transition function, $\gamma : Q \to \{+, -\}$ is the labelling function, and $q_0 \in Q$ is the start state. It is often helpful to think of an automata as a directed graph with labels $\{+, -\}$ on the nodes and labels $\{0, 1\}$ on the directed edges. An edge (i, j) with label ℓ exists if and only if $\tau(i, \ell) = j$. We let G_M be the edge-labelled directed graph induced by M without the state labels.

Learning DFA has a rich history which we briefly summarize here. The first results in this area were pessimistic; Gold (61) and Angluin (11) separately showed the problem of finding the smallest automaton consistent with a set of accept and reject strings is NP-complete. Later Pitt and Warmuth (62; 63) gave an NP-hard result for efficiently approximating the smallest machine. These works imply the intractability of learning finite automata in a variety of passive learning models when the hypothesis representation is an automata, including PAC-learning (64). Kearns and Valiant (65) later proved passive learning in the PAC model by any reasonable

representation is as hard as breaking crypotgraphic protocols. While passive learning in the most general sense is hard, there are positive results for active learning. Angluin (12), extending an algorithm presented by Gold (61), showed finite automata are efficiently learnable when the learner has access to so called 'membership' queries and counterexamples to it's current hypothesis. This is the well known L^* algorithm.

The model we consider was first introduced by Freund et al. (66). In their work, the transition function τ is arbitrary but the labelling γ is chosen uniformly at random. The learner has access to bits generated by a uniform random walk on the states: if the walk is at state q_t at time t, it moves to state $q_{t+1} = \tau(q_t, b)$ at time $t+1$ where b is an unbiased Bernoulli random variable. The learner the makes a prediction on the label of q_{t+1}, and receives feedback $\gamma(q_{t+1})$. The objective in this procedure is for the learner to stop making prediction mistakes at some point, up to some confidence parameter δ that . We formalize this procedure below.

Freund et al. (66) were able to prove the existence of an algorithm that makes polynomially many mistakes for all τ, if the confidence parameter is $\delta = 1/n^2$. Their algorithm relies on a subroutine where the random walk is reset to the starting node q_0. This chapter will only be concerned with this 'reset' model, so we assume the learner has access to a button that when pressed sends the random walk back to q_0 and incurs a mistake. In their analysis, (66) give an algorithm that makes $O(n^5/\delta^2)$ mistakes in the reset model. We will extend their results to the setting where the random walk is not uniform

5.2 Notation and Problem Statement

We borrow some notation from (66). For any state $q \in Q$ and $x \in \{0,1\}^*$, let $qx \in Q$ be the state reached by taking the walk x starting from state q. Let $q\langle x \rangle$ be the string of length $|x|+1$ of $\{+,-\}$ labels observed along this walk, and let $x^{(i)}$ be the first i bits in x.

While we would like to be able learn arbitrary automata topologies under arbitrary labellings γ, due to hardness results this is not feasible. In Section 5.3 we settle to learn arbitrary automata under random labellings γ and refer to 'the adversary' or 'adversarially chosen graph' as the worst-case nature of the choice of G_M. We borrow the following definition from (66):

Definition 12. *Let $P_{n,\delta}$ be any property on n-state finite automata depending only on n and confidence parameter $0 \leq \delta \leq 1$. We say that **uniformly almost all automata** have property $P_{n,\delta}$ if for all $\delta > 0$, for all $n > 0$ and for any n-state graph G_M, when the state labels $\{+,-\}$ are chosen uniformly at random with probability at least $1-\delta$, $P_{n,\delta}$ holds for the resulting automata M*

In section 5.4 we consider DFA topologies chosen uniformly at random.

5.2.1 Problem Statement

Given any adversarially chosen graph G_M and random labelling γ, we would like to learn the automata M induced by the graph and labelling. In the model we consider, the learner has access to bits generated from a p-biased random walk on G_M starting from q_0 and a 'reset switch' which restarts the random walk from q_0.

Formally the learner participates in the following procedure, which is repeated forever:

1. The learner predicts a label $\ell_t \in \{+, -, ?\}$ for the current state $s_t \in Q$ of the random walk
2. The learner is told the correct label $\gamma(s_t) \in \{+, -\}$ of the current state s_t
3. A new state s_{t+1} is chosen:
    - if the learner predicted '?', $s_{t+1} = q_0$
    - otherwise $s_{t+1} = \tau(s_t, 1)$ with probability p. $s_{t+1} = \tau(s_t, 0)$ with probability $1 - p$

That is, when the learner predicts "?" the random walk is reset to the initial state q_0. Otherwise the next state is one of the two out-neighbors of the current state, chosen randomly with bias p.

The learner will inevitably make some mistakes when predicting the label for the current state. In our analysis we distinguish between *prediction mistakes*, when the learner chooses $\ell_t \in \{+, -\}$ but $\ell_t \neq \gamma(s_t)$, and *default mistakes*, when the learner chooses '?'. Our results explicitly characterize the number of prediction and default mistakes made. Finally, in each round we require that the computation needed to predict a label is pseudo-polynomial in n and $1/\delta$, so is of the form $O\left(n^{c \log n}\right)$.

Our interest in psuedo-polynomial mistake bounds is motivated by the section below. Without modifying the core learning algorithm from (66), their analysis gives a psuedo-polynomial mistake bound when $p = n^{-c}$ for any constant c. Below this regime, their algorithm makes at least exponentially many mistakes, and many more if $p << 2^{-n}$ Improving this to a polynomial mistake bound is an interesting open problem, but we chose to focus on the small transition probability regime, where $p = o(n^{-c})$ for every c. In this setting achieving a psuedo-polynomial bound would unify these two regimes.

There is also theoretical reasons to be interested in this regime. A DFA can be thought of as two mappings from $[n]$ to $[n]$ by considering the partial transitions $\tau_b = \{qb : q \in [n]\}$ for $b \in \{0, 1\}$. In the low transition probability regime, the walk is trapped in τ_0 majority of the time, and only occasionally does a transition in τ_1 occur. Because the random walk is assumed to continue forever, all positive probability events will occur, so the learner needs to learn large parts of the DFA that are observed incredibly rarely. This provides a unique landscape for developing novel algorithms.

5.3 Learning with large transition probabilities

In this section, we analyze the **RESET** algorithm from (66) for arbitrary transition probabilities, referenced here as algorithm 3. In the regime where $p > 1/n^2$, the algorithm achieves a pseudo-polynomial mistake bound.

To bound the number of mistakes, we use a combinatorial property on the distinguishing of two states in the machine.

Definition 13. *A string $x \in \{0, 1\}^*$ is a distinguishing string for states q_1 and q_2 if $q_1\langle x\rangle \neq q_2\langle x\rangle$.*

A result proved by (66) is that over random labelings, all pairs of states in almost all automata have short distinguishing strings. Note that the randomness is over the labelling of the states of the automata, not the transition probabilities. This will be useful to us when we bias the random walk.

The main idea for this algorithm is to identify every state with its signature which is assumed to be unique; if we reach the same state often enough, the signature of that state can be

Algorithm 3 RESET algorithm from (66)

1. $Q' = \emptyset, Q'_{inc} = \{q'_0\}$
{Initializing the automata to be empty except the start state}
2. $q' = q'_0$
3. While $q' \notin Q'_{inc}$, after observing b set $q' \leftarrow \tau'(q', b)$
{While we're in a learned part of the automata, predict normally.}
4. Traverse the path through $\Sigma'(q')$ as dictated by the input sequence. At each step predict the label of the current node. Continue until an unlabelled node is reached, or the maximum depth of the tree is exceeded
5. Predict "?". If at an unlabelled node of $\Sigma'(q')$, then label it with the observed output symbol.
6. if $\Sigma'(q')$ is complete, then promote q' via:
(a) $Q'_{inc} \leftarrow Q'_{inc} - \{q'\}$
(b) if $\Sigma'(q') = \Sigma'(r')$ for some $r' \in Q'$:
- find s', b such that $\tau'(s', b) = q'$
- $\tau'(s', b) \rightarrow r'$
(c) else
- $Q' \leftarrow Q' \cup \{q'\}$
- create new states r'_0, r'_1
- $Q'_{inc} \leftarrow Q_{inc} \cup \{r'_0, r'_1\}'$
- partially fill in signatures of $r_0 1, r'_1$ using $\Sigma'(q')$
- $\tau'(q', b) \leftarrow r'_b$ for $b \in \{0, 1\}$
7. GOTO **2.**

discovered, thus its identity can be determined. By identifying the states of the machine, can reconstruct the transition function.

Theorem 22 (Freund et al. (66))**.** *For uniformly almost all automata, every pair of inequivalent states have a distinguishing string of length at most* $2\log_2(\frac{n^2}{\delta})$.

A *signature tree* with depth d for state q is a complete binary tree of depth d obtained by all possible walks $q\langle x\rangle$, $x \in \{0, 1\}^d$. We set $\Sigma(q)$ to be the d-signature tree with $d = 2\log_2\left(\frac{n^2}{\delta}\right)$.

By Theorem 22 for uniformly almost all automata M, every state $q \in Q(M)$ has a unique d-signature tree $\Sigma(q)$.

In Algorithm 3, a partial automata $M' = (Q', \tau', \gamma', q'_0)$ is slowly built by considering partial d-signature trees of hypothesis states $\Sigma'(q')$. Implicit in the analysis of algorithm 3 found in (66) is that regardless of the transition probability p, for uniformly almost all automata the algorithm is correct and for any automata only makes default mistakes. This is because of the conservative nature of the construction of M' and the $\Sigma'(\cdot)$: a new hypothesis node and transition are added only when the algorithm explores the 'frontier' of the signature tree built from the root node. We can exploit this fact and push their analysis further to all values of p:

Theorem 23. *For any transition probability $p := p(n) \in (0, 1)$, the expected number of default mistakes made by Algorithm 3 is $O\left(\frac{n^5}{\delta^2}\left(\frac{n^2}{\delta}\right)^{2\log\lceil 1/p\rceil}\right)$ and Algorithm 3 makes no prediction mistakes.*

Proof. We follow the analysis of (66). By the observations above, Algorithm 3 makes no prediction mistakes, and only makes default mistakes when exploring the leaf nodes of a signature tree for some state q. Since $\Sigma'(q)$ is a partial d-signature the leaves are not necessarily at depth d when explored. Therefore for $0 \le i \le d$, let X_i be the random variable counting the number of default mistakes made at depth i of $\Sigma'(q_0)$. Then for all $q' \neq q$, let $Y_{q'}$ be the random variable counting the number of times q' is reached before $\Sigma'(q')$ is complete.

Then the expected number of mistakes is the expectation of the sum of these random variables, which reduces to the coupon collectors problem. Let $k = \lceil \frac{1}{p} \rceil$ and consider T_k, the complete k-ary tree with depth d. Also consider taking a uniform random walk down T_k and

resetting the walk to the root after i steps. The expected number of resets needed to observe all nodes at depth i of T_k is an upper bound on $\mathbb{E}[X_i]$. A similar argument holds for $\mathbb{E}[Y_{q'}]$. Therefore,

$$\begin{aligned}\sum_{i=1}^{d} \mathbb{E}[X_i] + \sum_{q' \neq q} \mathbb{E}[Y_{q'}] &\leq \sum_{i=1}^{d} k^i(\log_2(k^i) + 1) + knk^d(\log_2 k^d + 1) \\ &\leq k^d(\log k^d + 1) + knk^d(\log k^d + 1) \\ &\leq nk^{\log(n^4/\delta^2)} \log\left(k^{\log(n^4/\delta^2)}\right)\end{aligned}$$

Then re-writing this in terms of p and rearranging finishes the proof, we have

$$\left(\frac{n^5}{\delta^2}\right)(\lceil 1/p \rceil)^{\log(n^4/\delta)} = O\left(\frac{n^5}{\delta^2}\left(\frac{n^2}{\delta}\right)^{2\log\lceil 1/p \rceil}\right).$$

□

This immediately gives a quasi-polynomial result for all sufficiently large p.

Corollary 9. *If $p > 1/n^c$, then the number of default mistakes made by Algorithm 3 is $O\left(\left(\frac{n^c}{\delta}\right)^{2c\log(n)}\right)$.*

5.4 Learning with Small Transition Probabilities: Random DFA

As discussed above, when the transition probability is sub-polynomial the learner spends majority of it's time in the 0-transitions. To see why this could be problematic, consider the DFA where for every state $q_i \in q_0, ..., q_{n-1}$, $q_i 0 = q_0$ and $q_i 1 = q_{i+1 \mod n}$. That is, the 0-transitions

all point to state q_0, and all 1-transitions form a cycle on the n vertices. Then to observe state q_{n-1} requires n 1-transitions to be observed in a row. This is an extremely low probability event, but for any fixed n is still positive. We have not found a way to handle events such as this, so we specialize to the case of random DFA. Here both the labelling γ and the transitions τ are chosen uniformly at random. This will allow us to leverage results from random mappings and develop ideas towards an algorithm.

5.4.1 Identifying 0-Cycles

The purpose of this section is to show the combinatorial result from Theorem 22 can be generalized to 0-cycles in a DFA. Here a 0-cycle is a set of vertices where the 0-transitions form a cycle. We will apply results from this section to random DFA, where the number of 0-cycles is small with high probability.

We specialize the notion of 'uniformly almost all' from definition 12 automata having a property to subsets of DFA,

Definition 14. *Let $P_{n,\delta}$ be any property on n-state automata depending only on n and confidence parameter $0 \leq \delta \leq 1$. Let $\mathcal{S}$ be a subset of all automata that is non-empty for every n. We say that **uniformly almost all** automata from $\mathcal{S}$ have property $P_{n,\delta}$ if for all $\delta > 0$, for all $n > 0$ and for any n-state graph $G_D \in \mathcal{S}$, when the state labels $\{+,-\}$ are chosen uniformly at random with probability at least $1-\delta$, $P_{n,\delta}$ holds for the resulting automata D.*

Definition 15. *A string $s \in \{0,1\}^*$ is called an **n, d-path** if every 1 in s is preceded by at least n 0's, and the number of 1's in s is at least d. Let $\mathcal{S}_{n,d}$ be the set of all such strings. The length of s is denoted by $|s|$, and the number of 1's in s is denoted by $\|s\|$.*

Now let $\mathcal{D}_C$ be the collection of DFA (on any number of nodes) where the 0-transitions form at most C disjoint cycles, and all vertices are on a 0-cycle. Two 0-cycles are said to be **equivalent** if every pair of vertices in the cycles are equivalent (have the same behavior on all strings $x \in \{0,1\}^*$). In general a cycle may not be equivalent to itself.

We state and prove the following structural result. Recall that $q\langle x\rangle$ is the length $|x|+1$ sequence of $\{+,-\}$ labels observed by taking the walk x starting from q.

Definition 16. *For an n-state automata $D \in \mathcal{D}_C$ and states $q_1, q_2 \in D$, an n,d-path x is an* **n-*distinguishing path*** *for q_1 and q_2 if $q_1\langle x\rangle \neq q_2\langle x\rangle$*

Theorem 24. *For uniformly almost all automata on n vertices from $\mathcal{D}_C$, every pair of inequivalent 0- cycles C_1, C_2 contain states $s_1 \in C_1$, $s_2 \in C_2$ such that there is an n-distinguishing path x for s_1, s_2 with norm at most $\|x\| = 2\log\left(\frac{C^2}{\delta}\right)$.*

As in (66), this is proved via a sequence of lemmas. The proof is very similar, except now the length of the distinguishing string is replaced by the norm of the distinguishing path. For completeness, we will present the lemmas and proofs. Throughout, D will be a DFA from the set $\mathcal{D}_C$ on n vertices, and C_1, C_2 will be 0-cycles in D.

Lemma 13. *Let C_1 and C_2 be inequivalent cycles in D, $v_1 \in C_1$, $v_2 \in C_2$ be inequivalent vertices on the cycles, and $x \in \{0,1\}^*$ be an n-distinguishing path for v_1 and v_2 with smallest norm. Let T_1 and T_2 be the sets of 0-cycles passed through on taking an x-walk from v_1 and v_2. Then $|T_1 \cup T_2| \geq \|x\| + 2$*

Proof. First observe that the statement is well-posed; since any 0-cycle has length less than n, any n-distinguishing path will necessarily have to completely pass through a cycle in between any two 1-transitions. Also note that C_1 could be equal to C_2, as a cycle need not be equivalent to itself.

Let R be a set of 0-cycles in D, and let $y \in \{0,1\}^*$. Define the partition of R induced by y to be the subsets of 0-cycles with the same behavior on y; C_1 and C_2 are in the same part if and only if for every $q_1 \in C_1$ and $q_2 \in C_2$, $q_1\langle x\rangle = q_2\langle x\rangle$.

Let x_i be the i'th bit of x and $\ell = |x|$. Let $y_i = x_i, ..., x_\ell$ be the suffix of x starting from position i. For any index i, let i^- be the index of the last 1 before i, and let i^+ be the index of the first 1 after i. We claim that for every i such that $x_i = 1$, the partition of $T_1 \cup T_2$ induced by y_i is a strict refinement of the partition induced by y_{i^+}. For the sake of contradiction suppose that there exists an index j such that the partition induced by y_j is the same as the partition induced by y_{j^+}. Define $r_1 = v_1 x^{(j^-)}$, and $r_2 = v_2 x^{(j^-)}$. Because x distinguishes v_1, v_2 we know $r_1\langle y_j\rangle \neq r_2\langle y_j\rangle$. But then since we assume the partition for y_j is the same as y_{j^+}, $v_1\langle x^{(j^-)} y_{j^+}\rangle \neq v_2\langle x^{(j^-)} y_{j^+}\rangle$, contradicting minimality of x.

Since the number of classes induced by the empty string is two (those vertices with a + label form one class, those with a - form the other), the number of classes in the partition induced by x is at least $\|x\| + 2$. On the other hand, the size of the partition of $T_1 \cup T_2$ induced by any set of strings is at most $|T_1 \cup T_2|$, therefore $\|x\| + 2 \leq |T_1 \cup T_2|$. □

Let x' be a prefix of the shortest n-distinguishing path x for q_1 and q_2 in D, such that the last bit of x' is followed by a 1 in x. Let y be such that $x = x'y$. Let z be new symbol

and define a new automata $D^y = (Q^y, \tau^y, \gamma^y, q_0^y)$ over alphabet $\{0, 1, z\}$ as follows. $q_0^y = q_0$. $Q^y = Q \cup \{q_+, q_-\}$. All transitions in D^y from states in Q on input in $\{0, 1\}$ remain the same. For $q \in Q$, $\tau^y(q, z) = q_+$ if $\gamma(qy) = +$, and q_- otherwise. For $q \in \{q_+, q_-\}$, $\tau^y(q, b) = q_-$ for $b \in \{0, 1, z\}$. Thus the automata D^y is identical to the automata M^y defined in (66), except we now condition that the string x' be in $\mathcal{S}_{n,d}$ for d large enough. In this new setting, all definitions from above will hold over the alphabet $\{0, 1, z\}$ by treating z as if it were a 1. For instance, n-distinguishing path over $0, 1, z$ is a string where every 1 or z is preceeded by at least n 0's.

Lemma 14. *$x'z$ is an n-distinguishing path for q_1 and q_2 in D^y with smallest norm.*

Proof. Clearly $x'z$ is an n-path by construction, and obviously distinguishes q_1 and q_2 in D^y.

Suppose t is a distinguishing path for q_1 and q_2 with smaller norm than $x'z$. Then z cannot appear in the middle of t, as by construction every z transition takes the machine to q_+ or q_-, so the prefix of t containing z would have smaller norm than t. If $t = t'z$, then $t'y$ has smaller norm than $t'x$ in D which is a contradiction. Likewise if $z \notin t$, then t has smaller norm than x in D which is also a contradiction. □

Lemma 15. *Let x' be a prefix of x such that the last bit of x' is followed by a 1 in x. Let $T_1' \subseteq T_1$ and $T_2' \subseteq T_2$ be the sets of states in D passed upon executing x' starting from v_1 and v_2. Then $|T_1' \cup T_2'| \geq \|x'\| + 1$.*

Proof. By Lemma 14, $x'z$ is a shortest distinguishing string for v_1 and v_2 in D^y. The set of cycles (0-cycles and the z self-cycle) passed upon executing $x'z$ starting from q_1 is $T_1' \cup \{q_+\}$, and those

passed starting from q_2 is $T_2' \cup \{q_-\}$. Applying Lemma 13, $|T_1' \cup T_2'| + 2 \geq \|x'z\| + 2 = \|x'\| + 3$, so the result follows. (Here we used the definition of $\|\cdot\|$ posed just above Lemma 14). □

Proof of Theorem 24: Let $D \in \mathcal{D}_C$ be an automata on n vertices where the 0-transitions form cycles. Let $d := 2\log\left(\frac{C^2}{\delta}\right)$. Let C_1 and C_2 be two (possibly the same) 0-cycles. If in every labelling of D, C_1 and C_2 are equivalent, then they are indistinguishable and we are done. Suppose instead that in every labelling there are distinguishable states $v_1 \in C_2$, and $v_2 \in C_2$ such that the norm of the n-distinguishing path is at most d. Then we are also done, because this is true for any random labelling of D.

Otherwise, there exists some labelling of D where C_1 and C_2 are inequivalent but distinguishing them requires an n-distinguishing path with norm greater than d. To be precise, there do not exist a pair of distinguishable states $v_1 \in C_2$, and $v_2 \in C_2$ with an n-distinguishing path containing less than d 1's.

Let $(v_1', v_2') \in (C_1, C_2)$ be a pair of vertices in the 0-cycles with an n-distinguishing path x that has the smallest norm. That is,

$$(v_1', v_2') = \operatorname*{arg\,min}_{(a,b)\in C_1\times C_2} \min_{a\langle y\rangle \neq b\langle y\rangle} \|y\|$$

Define $x(j)$ to be the prefix of x just before the $j+1$'st 1. Consider the $x(d)$ walks taken from v_1' and v_2'. Define $d+1$ state pairs $(r_1^i = r_2^i)$ of D by $(r_1^i = r_2^i) = (v_1 x(j), v_2 x(j))$ for $0 \leq i \leq d$. Now since x is a distinguishing path, for every i $r_1^i \neq r_2^i$. Furthermore Lemma 15 tells us that at least $d+1$ unique 0-cycles are represented in these state pairs. Each cycle contains at least one

vertex and therefore at least one randomly labelled bit. The probability that the string $v_1\langle x(d)\rangle$ is identical to $v_2\langle x(d)\rangle$ is therefore at most $2^{-(d+1)/2}$.

For any fixed pair of 0-cycles C_1, C_2 in D, the probability that C_1 and C_2 are inequivalent but indistinguishable by strings of length d is at most $2^{-(d+1)/2}$. Thus the probability of this occurring for any pair of cycles is bounded by $C^2 2^{-(d+1)/2}$. If $d \geq 2\log\frac{C^2}{\delta}$, this probability is smaller than δ. □

5.4.2 Learning Random DFA in Stages

Consider a DFA D on states $[n]$ with transitions chosen uniformly at random. Then the 0 and 1 transitions form two independent random mappings over $[n]$. Therefore we can talk about the 0-mapping as the random mapping induced by the 0-transitions. Furthermore we say 0-cycles and 0-transitions to mean directed cycles and trees in the 0-mapping. We briefly prove some useful properties of random mappings.

Let $P(n)$ be the number of components in a random mapping, Let $Y(n)$ count the number of vertices on cycles in a random mapping, and let $L(n)$ be the length of the longest cycle in a random mapping. The following holds.

Lemma 16. *There exists a constant* C_1 *such that* a.a.s. $P(n) < C_1 \log n$.

Proof. Set $C_1 = 5$ (this can be optimized). (10) showed the distribution of $P(n)$ converges to a standard gaussian when scaled and centered appropriately. Applying this, the Portmanteau lemma, and the symmetry of the Gaussian,

$$\lim_{n\to\infty} \mathbb{P}\left[\frac{P(n) - \mathbb{E}P(n)}{\sqrt{\frac{1}{2}\log n}} \leq -t\right] = \mathbb{P}[\mathcal{N}(0,1) \leq -t]$$

$$\implies \lim_{n\to\infty} \mathbb{P}\left[\frac{P(n) - \mathbb{E}P(n)}{\sqrt{\frac{1}{2}\log n}} \geq t\right] = \exp\left(\frac{-t^2}{2}\right)$$

Setting $t = 2\sqrt{\log n}$ is enough to guarantee $P(n) \leq C_1 \log n$. □

Lemma 17. *For $\varepsilon > 0$, for n sufficiently large, $\mathbb{P}[L(n) \leq \varepsilon\sqrt{n}] \leq \varepsilon + O\left(\frac{1}{\sqrt{n}}\right)$.*

Proof. Consider the following process. First, for vertex 1 choose a neighbor v_1 uniformly at random from $[n]$. Likewise for all $k > 1$, v_{k+1} is defined by choosing a neighbor uniformly at random from $[n]$. This gives an infinite sequence of vertices $\{1, v_1, v_2, ..., v_i, ..., v_j, v_i, ...v_j, v_i...\}$ where the random substring $v_i, ..., v_j$ repeats forever. Let the random variable $H(n)$ count the length of this repeating segment, and let $F(n) = \min_k\{v_k | v_{k+1} \in \{v_1, ..., v_k\}\}$ be the last index before the repeating segment. Notice that this process defines part of one component of $M(n)$, so bounds on $H(n)$ imply bounds on $L(n)$. We have,

$$\begin{aligned}
\mathbb{P}[L(n) \leq \varepsilon\sqrt{n}] &\leq \mathbb{P}[H(n) \leq \varepsilon\sqrt{n}] \\
&= \mathbb{P}[H(n) \leq \varepsilon\sqrt{n} \wedge F(n) \leq \sqrt{n}] + \mathbb{P}[H(n) \leq \sqrt{n} \wedge F(n) > \sqrt{n}] \\
&= \mathbb{P}[H(n) \leq \varepsilon\sqrt{n} | F(n) \leq \sqrt{n}]\mathbb{P}[F(n) \leq \sqrt{n}] \\
&\quad + \mathbb{P}[H(n) < \varepsilon\sqrt{n} | F(n) > \sqrt{n}]\mathbb{P}[F(n) > \sqrt{n}] \\
&\leq \mathbb{P}[F(n) \leq \sqrt{n}] + \mathbb{P}[H(n) \leq \varepsilon\sqrt{n} | F(n) \leq \sqrt{n}]
\end{aligned}$$

Then notice that $\mathbb{P}[H(n) \leq \varepsilon\sqrt{n} | F(n) \leq \sqrt{n}] \leq \varepsilon$, because conditioning on the length of the non-repeating section $F(n)$ ensures that the probability of forming a cycle is at most ε. To finish the proof it is enough to show $\mathbb{P}[F(n) \leq \sqrt{n}] = O(1/\sqrt{n})$,

$$
\begin{aligned}
\mathbb{P}[F(n) \leq \sqrt{n}] &= \mathbb{P}[F(n) \in 1, ..., \sqrt{n}] \\
&= \sum_{i=1}^{\sqrt{n}} \frac{i}{n} \prod_{j=1}^{i-1} \frac{n-j}{n} \\
&= \sum_{i=1}^{\log(n)-1} \frac{i}{n} \prod_{j=1}^{i-1} \frac{n-j}{n} + \sum_{i=\log n}^{\sqrt{n}} \frac{i}{n} \prod_{j=1}^{i-1} \frac{n-j}{n} \\
&\leq \frac{\log^2 n}{n} + \sum_{k=0}^{\sqrt{n}-\log n} \frac{k+\log n}{n} \prod_{j=1}^{k+\log n-1} \frac{n-j}{n} \\
&= \frac{\log^2 n}{n} + \sum_{k=1}^{\sqrt{n}-\log n} \frac{k+\log n}{n} \frac{1}{n^{k+\log n}} \frac{(n-1)!}{(n-k-\log n)!} \\
&\approx \frac{\log^2 n}{n} + \frac{1}{\sqrt{n}} \sum_{k=0}^{\sqrt{n}-\log n} \frac{1}{n^{k+\log n}} \frac{\sqrt{2\pi n}}{\sqrt{2\pi(n-k-\log n)}} \left(\frac{n}{e}\right)^n \left(\frac{n}{e}\right)^{-n+k+\log n} \\
&= \frac{\log^2 n}{n} + \frac{1}{\sqrt{n}} \sum_{k=1}^{\sqrt{n}-\log n} \frac{\sqrt{2\pi n}}{\exp(k+\log n)} \\
&\leq \frac{\log^2 n}{n} + \sqrt{2\pi}\frac{\sqrt{n}}{n} \\
&= O\left(\frac{1}{\sqrt{n}}\right)
\end{aligned}
$$

Where the second line is by independence of edges, the third line is splitting the sum, the first inequality is bounding the first summation, and the approximation is by Stirling's formula. □

Corollary 10. *For all* $\varepsilon > 0$ *and* n *sufficiently large,* $\mathbb{P}[Y(n) \leq \varepsilon\sqrt{n}] \leq \varepsilon + O\left(\frac{1}{\sqrt{n}}\right)$

If all the 0-components in a random DFA were cycles, an immediate application of Theorem 24 and modification of Algorithm 3 would be sufficient to produce an algorithm for learning when p is sub-polynomial. To see this, instead of building signature trees for each node, we could

instead build signature trees for each cycle with depth $2\log\frac{C^2}{\delta}$. By Algorithm 3, every cycle would have an n-distinguishing tree, and we would therefore only need to connect these with the appropriate 1-transitions to recover the DFA.

Because a Random DFA has 0-paths leading into 0-cycles we need a bit more work. Given a random DFA D, let D' be the automata where 0-transitions not on cycles are contracted to the first vertex that is on a cycle. Then $D' \in \mathcal{D}_C$, and by Corollary 10, $|V(D')| \geq n^{1/3}$ with high probability. By Theorem 24 and Lemma 16, for any cycles C_1, C_2 there are states $s_1 \in C_1, s_2 \in C_2$ such that there is a $n^{1/3}$-distinguishing path x for s_1, s_2 with norm at most $\|x\| = 2\log\left(\frac{\log^2(n)}{\delta}\right)$. This informs the algorithm in the following section, which after completion will not make mistakes on strings of the form $(0^{n^{1/3}}1)^*$.

5.4.2.1 Stage 1

Stage 1 proceeds by building a sub-DFA where no mistakes are made when the input string does not contain two minority-transitions (1-transitions) within distance n of each other.

The subroutine COLLECT(k) predicts arbitrarily for k steps and outputs either -1 if a 1-transition occurs, or it outputs the labels seen on the all 0-transition path of length k.

The subroutine LOLLIPOP(S) takes in a string S, and outputs a labelled directed graph on $|S|$ nodes. The digraph contains a path entering a cycle, where the path is labelled by a prefix of S and the cycle is labelled by the shortest repeating subsequence of the suffix of S. If the length of this suffix is less than $2\log n$, it is increased in size to be at least $2\log n$ by repeating its labelling. (For example if $n = 5$ and $S = 101011$, the output will be a 4-path labelled 1010 leading into a 6-cycle labelled 111111.)

Algorithm 4 STAGE 1

1. $Q_{fin} \leftarrow \emptyset$, $Q_{inc} \leftarrow \{q_0\}$
2. $C_{fin} \leftarrow \emptyset$, $C_{inc} \leftarrow \emptyset$
2a. From the initial node q_0, call $S \leftarrow$ COLLECT(n) until $S \neq -1$, and add the cycle part of LOLLIPOP(S) to C_{inc}
3. $q' \leftarrow q_0$, $C' \leftarrow C(q')$ {$C(q)$ is the 0-cycle in the component containing q. We can assume eg. there is a dictionary mapping nodes to cycles stored}
4. While $q' \notin Q_{fin}$, receive input and predict until encountering an unlabelled node, updating q' to be the current node, and setting $C' \leftarrow C(q')$
5. Call $S \leftarrow$ COLLECT(n). If $S = -1$ goto step 2. Else add the state-output pairs to $\Sigma'(q')$, and add the cycle-part from LOLLIPOP(S) to $\Sigma_T(C')$. Predict '?'
6. If $\Sigma_T(C')$ is complete:
(a) Remove C' from C_{inc}
(b) If $\Sigma_T(C') = \Sigma_T(R')$ for some $R' \in C_{fin}$ then
- for every $q' \in C'$, find appropriate 1-transitions and connect them
(c) Else
- Add C' to C_{fin}
- Add new states $r'_0, ..., r'_\ell$ to Q_{inc} corresponding to the 1-transitions out of the cycle
- Partially fill in the node-signature trees and cycle-signature trees where applicable
- Add 0-transition paths into C' observed on previous calls of step (5) in $\Sigma'(q')$. Root all these paths on a node in C' so that no two paths share a node.

7. GOTO step (3)

Theorem 25. *The expected number of mistakes for Algorithm 4 to learn strings of the form* $(0^n1)^*$ *with probability at least* $(1-\delta)$ *is* $O\left(n \log^3(n) (2\log n)^{2\log \frac{\log^2 n}{\delta}}\right)$. *In particular this is less than* $O\left(n^2 (\log n)^{\log \frac{1}{\delta}}\right)$.

Proof. We first discuss correctness of the algorithm. It keeps track of two quantities, Q_{fin} corresponding to completed states of the machine, and C_{fin} corresponding to completed 0-cycles. It also tracks two signature trees, Σ' and Σ_T for each node and cycle respectively.

A cycle is only added to C_{fin} in stage 6, when its signature tree $\Sigma_T(C')$ is complete. By Theorem 24 this tree will contain some distinguishing path, so every pair of cycles will be distinguished with probability at least $1 - \delta$. Furthermore by the definition of COLLECT(n), any cycle of length less than $2 \log n$ will have its size increased to match $2 \log n$. A simple computation shows that the probability any pair of n randomly labelled cycles on $\geq 2 \log n$ vertices share a labelling is $o(1)$. Therefore, by this blow-up procedure no two cycles with the same labelling will be identified as the same.

Furthermore, a node is included in Q_{fin} only if it has a 0-path to a cycle in C_{fin}, as per (6c). Furthermore by the construction of the signature trees, every 0-path is observed at some point in a Σ', therefore this step will terminate, so the algorithm will form a partial DFA that doesn't make mistakes on strings of the form $(0^n 1)^*$.

For the mistake bound, note that the probability (2a) and (5) takes one step to complete is $O(1 - pn)$, and since $p = o(n^{-c})$ for every c, we will see failure of (2a) and (5) only contributes to lower order terms in the mistake bound. Furthermore each cycle signature tree has depth $2 \log \frac{\log^2 n}{\delta}$. Naively one might expect the branching factor to grow as $O(n)$, but we can reduce this to $2 \log n$ by the following observation. Any 0-cycle with length greater than $2 \log n$ will not share it's labelling with any other cycle with high probability. Therefore we may prune the signature tree whenever encountering a cycle of this length. By the definition of COLLECT(n), all cycles in the tree will have size at least $2 \log n$.

Each cycle therefore has a cycle tree with depth $d = 2 \log \frac{\log^2 n}{\delta}$ and branching factor $b = 2 \log n$, and the algorithm stops making mistakes once all cycle trees are complete. Once

the walk is on a 0-cycle, the probability of exiting any node is very close to uniform, so for simplicity we will assume it is actually uniform. Correcting this assumption will only incur lower order terms to the mistake bound. By a coupon collector argument, the expected number of mistakes before a signature tree is complete is upper bounded as $O(d^b \log(d^b))$. Summing over all cycles is sufficient to complete the proof. □

5.4.2.2 Stage 2 and Beyond

We briefly summarize and discuss some open problems for this chapter. We have seen how to learn a DFA from biased random walks by adapting the core algorithm from Freund et al. (66). This produces a psuedo-polynomial mistake bound when $p = 1/\operatorname{poly}(n)$. For p much smaller, we just described Stage 1 of an algorithm that learns the most probable strings with $\operatorname{poly}(n)$ mistakes in random DFA.

For the following stages, we run into a few difficulties. First, almost certainly the sub-DFA constructed in Stage 1 does not include some small 0-cycles. In particular as n grows there are $\text{Pois}(1/k)$ 0-cycles of constant length k, and $O(\sqrt{n})$ vertices on 0-cycles. As this stage only observes 1-transitions off of 0-cycles, there will be some 0-cycle that is reachable from q_0 but not observed in this stage. Furthermore, a heuristic calculation shows there are at least a constant number of vertices that are only reachable from $O(\log n)$ 1-transitions in a row. Thus, while Stage 1 learns very likely strings, it somehow misses at least constantly many vertices.

Learning these rarely visited vertices poses the following conceptual challenge - how to learn rare events without incurring many mistakes? We have spent some time thinking about this

question and have some partial results. This chapter and progress on algorithms for Random DFA will be represented in future works.

CITED LITERATURE

Bibliography

[1] Maranzatto, T.: Age of gossip in random and bipartite networks, 2024.

[2] Maranzatto, T.: Reconstructing arbitrary trees from traces in the tree edit distance model, 2021.

[3] Gao, X., Maranzatto, T., and Reyzin, L.: A unified analysis of dynamic interactive learning. In 2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, September 2023.

[4] Erdős, P., R. A.: On random graphs i. Publicationes Mathematicae Debrecen, 6, 1959.

[5] Gilbert, E. N.: Random Graphs. The Annals of Mathematical Statistics, 30(4):1141 – 1144, 1959.

[6] Bollobás, B.: The evolution of random graphs. Transactions of the American Mathematical Society, 286(1):257–274, 1984.

[7] Friedgut, E. and Kalai, G.: Every monotone graph property has a sharp threshold. 1996.

[8] Bollobás, B.: The isoperimetric number of random regular graphs. European Journal of Combinatorics, 9(3):241–244, 1988.

[9] Flajolet, P. and Odlyzko, A. M.: Random mapping statistics. In International Conference on the Theory and Application of Cryptographic Techniques, 1990.

[10] Stepanov, V. E.: Limit distributions of certain characteristics of random mappings. Theory of Probability & Its Applications, 14(4):612–626, 1969.

[11] Angluin, D.: On the complexity of minimum inference of regular sets. Inform. and Control, 39(3):337–350, 1978.

[12] Angluin, D.: Queries and concept learning. Mach. Learn., 2(4):319–342, 1987.

[13] Mohamed, M., ElSawy, H., and Mesbah, W.: Optimized degree-aware random patching for thwarting iot botnets. IEEE Networking Letters, 5(1):59–63, 2023.

[14] Isik, B., Pase, F., Gunduz, D., Weissman, T., and Michele, Z.: Sparse random networks for communication-efficient federated learning. In The Eleventh International Conference on Learning Representations, 2023.

[15] Liu, D., Xu, Y., Wang, J., Xu, Y., Anpalagan, A., Wu, Q., Wang, H., and Shen, L.: Self-organizing relay selection in uav communication networks: A matching game perspective. IEEE Wireless Communications, 26(6):102–110, December 2019.

[16] Sun, Y., Kadota, I., Talak, R., and Modiano, E. H.: Age of information: A new metric for information freshness. Synthesis Lectures on Communication Networks, 2019.

[17] Maatouk, A., Kriouile, S., Assaad, M., and Ephremides, A.: The age of incorrect information: A new performance metric for status updates. IEEE/ACM Transactions on Networking, 28(5):2215–2228, October 2020.

[18] Zhong, J., Yates, R. D., and Soljanin, E.: Two freshness metrics for local cache refresh. 2018 IEEE International Symposium on Information Theory (ISIT), pages 1924–1928, 2018.

[19] Yates, R. D.: The age of gossip in networks. In 2021 IEEE International Symposium on Information Theory (ISIT). IEEE, July 2021.

[20] Srivastava, A. and Ulukus, S.: Age of gossip on generalized rings. In MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM). IEEE, October 2023.

[21] Srivastava, A. and Ulukus, S.: Age of gossip on a grid. In 2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, September 2023.

[22] Kaswan, P., Mitra, P., Srivastava, A., and Ulukus, S.: Age of information in gossip networks: A friendly introduction and literature survey, 2023.

[23] Blum, A., Hopcroft, J., and Kannan, R.: Random Graphs, page 215–273. Cambridge University Press, 2020.

[24] Krivelevich, M. and Sudakov, B.: The phase transition in random graphs: A simple proof. Random Structures and; Algorithms, 43(2):131–138, September 2012.

[25] Srivastava, H. and Choi, J.: Zeta and q-zeta functions and associated series and integrals. In Zeta and q-Zeta Functions and Associated Series and Integrals, eds. H. Srivastava and J. Choi, pages 1–140. London, Elsevier, 2012.

[26] Bollobás, B. and Chung, F. R. K.: The diameter of a cycle plus a random matching. SIAM Journal on Discrete Mathematics, 1(3):328–333, 1988.

[27] Bollobás, B. and Fernandez de la Vega, W.: The diameter of random regular graphs. Combinatorica, 2(2):125–134, June 1982.

[28] Ellis, R. B., Martin, J. L., and Yan, C.: Random geometric graph diameter in the unit ball. Algorithmica, 47(4):421–438, February 2007.

[29] Barbour, A., Karoński, M., and Ruciński, A.: A central limit theorem for decomposable random variables with applications to random graphs. Journal of Combinatorial Theory, Series B, 47(2):125–145, 1989.

[30] Davies, S., Rácz, M. Z., and Rashtchian, C.: Reconstructing trees from traces. In COLT, eds. A. Beygelzimer and D. Hsu, volume 99, pages 961–978. PMLR, 2019.

[31] Batu, T., Kannan, S., Khanna, S., and McGregor, A.: Reconstructing strings from random traces. In SODA, pages 910–918. SIAM, 2004.

[32] Bhardwaj, V., Pevzner, P. A., Rashtchian, C., and Safonova, Y.: Trace reconstruction problems in computational biology. IEEE Transactions on Information Theory, page 1–1, 2020.

[33] Church, G. M., Gao, Y., and Kosuri, S.: Next-generation digital information storage in dna. Science, 337(6102):1628–1628, 2012.

[34] He, L., Karau, P., and Tabard-Cossa, V.: Fast capture and multiplexed detection of short multi-arm dna stars in solid-state nanopores. Nanoscale, 11:16342–16350, 2019.

[35] Bouchard-Côté, A., Hall, D., Griffiths, T. L., and Klein, D.: Automated reconstruction of ancient languages using probabilistic models of sound change. Proceedings of the National Academy of Sciences, 110(11):4224–4229, March 2013. Publisher: National Academy of Sciences Section: Physical Sciences.

[36] Chase, Z.: New upper bounds for trace reconstruction, 2020.

[37] Nazarov, F. and Peres, Y.: Trace reconstruction with $\exp(O(n^{1/3}))$ samples. In STOC, pages 1042–1046. ACM, 2017.

[38] De, A., O'Donnell, R., and Servedio, R. A.: Optimal mean-based algorithms for trace reconstruction. Annals of Applied Probability, 29(2):851–874, 2019.

[39] Chase, Z.: New lower bounds for trace reconstruction, 2019.

[40] Holden, N. and Lyons, R.: Lower bounds for trace reconstruction, 2018.

[41] Holden, N., Pemantle, R., and Peres, Y.: Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In Proceedings of the 31st Conference On Learning Theory, eds. S. Bubeck, V. Perchet, and P. Rigollet, volume 75 of Proceedings of Machine Learning Research, pages 1799–1840. PMLR, 06–09 Jul 2018.

[42] Holenstein, T., Mitzenmacher, M., Panigrahy, R., and Wieder, U.: Trace reconstruction with constant deletion probability and related results. In SODA, 2008.

[43] Chen, X., De, A., Lee, C. H., Servedio, R. A., and Sinha, S.: Polynomial-time trace reconstruction in the smoothed complexity model, 2020.

[44] Brakensiek, J., Li, R., and Spang, B.: Coded trace reconstruction in a constant number of traces, 2019.

[45] Cheraghchi, M., Gabrys, R., Milenkovic, O., and Ribeiro, J.: Coded trace reconstruction. IEEE Transactions on Information Theory, 66(10):6084–6103, 2020.

[46] Srinivasavaradhan, S. R., Du, M., Diggavi, S., and Fragouli, C.: On maximum likelihood reconstruction over multiple deletion channels. In 2018 IEEE International Symposium on Information Theory (ISIT), pages 436–440. IEEE, 2018.

[47] Davies, S., Racz, M. Z., Rashtchian, C., and Schiffer, B. G.: Approximate trace reconstruction, 2020.

[48] Organick, L., Ang, S., Chen, Y., Lopez, R., Yekhanin, S., Makarychev, K., Racz, M., Kamath, G., Gopalan, P., Nguyen, B., Takahashi, C., Newman, S., Parker, H., Rashtchian, C., Stewart, K., Gupta, G., Carlson, R., Mulligan, J., Carmean, D., Seelig, G., Ceze, L., and Strauss, K.: Random access in large-scale dna data storage. Nature Biotechnology, 36(3):242–248, March 2018.

[49] Karau, P. and Tabard-Cossa, V.: Capture and translocation characteristics of short branched dna labels in solid-state nanopores. ACS Sensors, 3(7):1308–1315, 2018. PMID: 29874054.

[50] Mitzenmacher, M.: A survey of results for deletion channels and related synchronization channels. Probability Surveys, 6:1–33, 2009.

[51] Awasthi, P., Balcan, M., and Voevodski, K.: Local algorithms for interactive clustering. J. Mach. Learn. Res., 18:3:1–3:35, 2017.

[52] Balcan, M.-F. and Blum, A.: Clustering with interactive feedback. In International Conference on Algorithmic Learning Theory, pages 316–328. Springer, 2008.

[53] Lelkes, Á. D. and Reyzin, L.: Interactive clustering of linear classes and cryptographic lower bounds. In Algorithmic Learning Theory - 26th International Conference, ALT 2015, Banff, AB, Canada, October 4-6, 2015, Proceedings, eds. K. Chaudhuri, C. Gentile, and S. Zilles, volume 9355 of Lecture Notes in Computer Science, pages 165–176. Springer, 2015.

[54] Emamjomeh-Zadeh, E. and Kempe, D.: A general framework for robust interactive learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 7085–7094, 2017.

[55] Dereniowski, D., Tiegel, S., Uznanski, P., and Wolleb-Graf, D.: A framework for searching in graphs in the presence of errors. In 2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA, eds. J. T. Fineman and M. Mitzenmacher, volume 69 of OASICS, pages 4:1–4:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[56] Becerra-Bonache, L., Dediu, A., and Tirnăucă, C.: Learning DFA from correction and equivalence queries. In Grammatical Inference: Algorithms and Applications, 8th International Colloquium, ICGI 2006, Tokyo, Japan, September 20-22, 2006, Proceedings, eds. Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, and E. Tomita, volume 4201 of Lecture Notes in Computer Science, pages 281–292. Springer, 2006.

[57] Emamjomeh-Zadeh, E., Kempe, D., Mahdian, M., and Schapire, R. E.: Interactive learning of a dynamic structure. In Algorithmic Learning Theory, pages 277–296. PMLR, 2020.

[58] Bousquet, O. and Warmuth, M. K.: Tracking a small set of experts by mixing past posteriors. J. Mach. Learn. Res., 3:363–396, 2002.

[59] Emamjomeh-Zadeh, E., Kempe, D., and Singhal, V.: Deterministic and probabilistic binary search in graphs. In Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016, eds. D. Wichs and Y. Mansour, pages 519–532. ACM, 2016.

[60] Ben-Or, M. and Hassidim, A.: The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA, pages 221–230. IEEE Computer Society, 2008.

[61] Gold, E. M.: System identification via state characterization. Automatica J. IFAC, 8:621–636, 1972.

[62] Pitt, L. and Warmuth, M. K.: The minimum consistent DFA problem cannot be approximated within any polynomial. J. Assoc. Comput. Mach., 40(1):95–142, 1993.

[63] Pitt, L. and Warmuth, M. K.: Prediction-preserving reducibility. J. Comput. System Sci., 41(3):430–467, 1990.

[64] Valiant, L. G.: Robust logics. In Annual ACM Symposium on Theory of Computing (Atlanta, GA, 1999), pages 642–651. ACM, New York, 1999.

[65] Kearns, M. and Valiant, L.: Cryptographic limitations on learning Boolean formulae and finite automata. J. Assoc. Comput. Mach., 41(1):67–95, 1994.

[66] Freund, Y., Keams, M., Ron, D., Rubinfeld, R., Schapire, R., and Sellie, L.: Efficient learning of typical finite automata from random walks. pages 315–324, June 1993. 25th Annual ACM Symposium on Theory of Computing, STOC 1993 ; Conference date: 16-05-1993 Through 18-05-1993.