Learning Finite-State Machines

Statistical and Algorithmic Aspects

Borja de Balle Pigem

Thesis Supervisors: Jorge Castro and Ricard Gavaldà
Department of Software — PhD in Computing

Thesis submitted to obtain the qualification of Doctor
from the Universitat Politècnica de Catalunya

LARCA. Laboratory for Relational Algorithmics, Complexity and Learning
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

The present thesis addresses several machine learning problems on generative and predictive models on sequential data. All the models considered have in common that they can be defined in terms of finite-state machines. On one line of work we study algorithms for learning the probabilistic analog of Deterministic Finite Automata (DFA). This provides a fairly expressive generative model for sequences with very interesting algorithmic properties. State-merging algorithms for learning these models can be interpreted as a divisive clustering scheme where the "dependency graph" between clusters is not necessarily a tree. We characterize these algorithms in terms of statistical queries and a use this characterization for proving a lower bound with an explicit dependency on the distinguishability of the target machine. In a more realistic setting, we give an adaptive state-merging algorithm satisfying the stringent algorithmic constraints of the data streams computing paradigm. Our algorithms come with strict PAC learning guarantees. At the heart of state-merging algorithms lies a statistical test for distribution similarity. In the streaming version this is replaced with a bootstrap-based test which yields faster convergence in many situations. We also studied a wider class of models for which the state-merging paradigm also yield PAC learning algorithms. Applications of this method are given to continuous-time Markovian models and stochastic transducers on pairs of aligned sequences. The main tools used for obtaining these results include a variety of concentration inequalities and sketching algorithms.

In another line of work we contribute to the rapidly growing body of spectral learning algorithms. The main virtues of this type of algorithms include the possibility of proving finite-sample error bounds in the realizable case and enormous savings on computing time over iterative methods like Expectation-Maximization. In this thesis we give the first application of this method for learning conditional distributions over pairs of aligned sequences defined by probabilistic finite-state transducers. We also prove that the method can learn the whole class of probabilistic automata, thus extending the class of models previously known to be learnable with this approach. In the last two chapters we present works combining spectral learning with methods from convex optimization and matrix completion. Respectively, these yield an alternative interpretation of spectral learning and an extension to cases with missing data. In the latter case we used a novel joint stability analysis of matrix completion and spectral learning to prove the first generalization bound for this type of algorithms that holds in the non-realizable case. Work in this area has been motivated by connections between spectral learning, classic automata theory, and statistical learning; tools from these three areas have been used.

Preface

From Heisenberg's uncertainty principle to the law of diminishing returns, essential trade-offs can be found all around in science and engineering. The Merriam–Webster on-line dictionary defines *trade-off* as a noun describing "a balancing of factors all of which are not attainable at the same time." In many cases where a human actor is involved in the balancing, such balancing is the result of some conscious reasoning and acting plan, pursued with the intent of achieving a particular effect in a system of interest. In general it is difficult to imagine that this reasoning and acting can take place without the existence of some specific items of knowledge. In particular, no reasoning whatsoever is possible without the knowledge of the factors involved in the trade-off and some measurement of the individual effect each of those factors has on the ultimate goal of the balancing.

The knowledge required to reach a trade-off can be conceptually separated into two parts. First there is a qualitative part, which involves the identification of the factors that need to be balanced. Then, there is a quantitative one: the influence exerted by these factors on the system of interest needs to be measured. These two pieces are clearly complementary, and each of them brings with it some actionable information. The qualitative knowledge brings with it intuitions about the underlying works of the system under study which can be useful in deriving analogies and generalizations. On the other hand, quantitative measurements provide the material needed for principled mathematical reasoning and fine-tuning of the trade-off.

In computer science there are plenty of trade-offs that perfectly exemplify this dichotomy. Take, for example, the field of optimization algorithms. Let k denote the number of iterations executed by an iterative optimization algorithm. Obviously, there exists a relation between k and the accuracy of the solution found by the algorithm. This is a general intuition that holds along all optimization algorithms, and one that can be further extended to other families of approximation algorithms. On the other hand, it is well known that for some classes of optimization problems the distance to the optimal solution decreases like $O(1/k)$, while for some others the rate $O(1/\sqrt{k})$ is not improvable. These particular bits of knowledge exemplify two different quantitative realizations of the iterations/accuracy trade-off. But even more importantly, this kind of quantitative information can be used to approach meta-trade-offs like the following. Suppose you are given two models for the same problem, one that is closer to reality but can only be solved at a rate of $O(1/\sqrt{k})$, and another that is less realistic but can be solved at a rate of $O(1/k)$. Then, given a fixed computational budget, should one compute a more accurate solution to an imprecise model, or a less accurate solution in a more precise model? Being able to answer such questions is what leads to informed decisions in science and engineering, which is, in some sense, the ultimate goal for studying such trade-offs in the first place.

Learning theory is a field of computer science that abounds in trade-offs. Those arise not only from the computational nature of the field, but also from a desire to encompass, in a principled way, many aspects of the statistical process that takes data from some real world problem and produces a meaningful model for it. The many aspects involved in such trade-offs can be broadly classified as either computational (time and space complexity), statistical (sample complexity and identifiability), or of a modeling nature (realizable *vs* non-realizable, proper *vs* improper modeling). This diversity induces a rich variety of situations, most of which, unfortunately, turn out to be computationally intractable as soon as the models of interest become moderately complex. The understanding and identification of all these trade-offs has lead in recent years to important advances in the theory and practice of machine learning, data mining, information retrieval, and other data-driven computational disciplines.

In this dissertation we address a particular problem in learning theory: the design and analysis of algorithms for learning finite-state machines. In particular, we look into the problem of designing efficient learning algorithms for several classes of finite-state machines under different sets of modelling assumptions. Special attention is devoted to the computational and statistical complexity of the proposed methods. The leitmotif of the thesis is to try answering the question: *what is the larger class of finite-state machines that is efficiently learnable from a computational and statistical perspective?* Of course, a short answer would be: *there is a trade-off.* The results presented in this thesis provide several longer answers to this question, all of which illuminate some aspect of the trade-offs implied by the different modelling assumptions.

Acknowledgements

Needless to say is that this is the last page one writes on a thesis. It is thus the time to thank everyone whose help, in one way or another, has made it possible to reach this final page. Help has come to me in many forms: mentoring, friendship, love, and mixtures of those. To me, these are the things which make life worth living. And so, it is with great pride and pleasure that I thank the following people for their help.

First and foremost, I am eternally thankful to my advisors Jorge and Ricard for their encouragement, patience, and guidance throughout these years. My long-time co-authors Ariadna and Xavier, for their kindness, and for sharing their ideas and listening to mine, deserve little less that my eternal gratitude. During my stay in New York I had the great pleasure to work with Mehryar Mohri; his ideas and points of view still inspire me to this day.

Besides my closer collaborators, I have interacted with a long list of people during my research. All of them have devoted their time to share and discuss with me their views on many interesting problems. Hoping I do not forget any unforgettable name, I extend my most sincere gratitude to the following people: Franco M. Luque, Raphaël Bailly, Albert Bifet, Daniel Hsu, Dana Angluin, Alex Clark, Colin de la Higuera, Sham Kakade, Anima Anandkumar, David Sontag, Jeff Sorensen, Marta Àrias, Ramon Ferrer, Doina Precup, Prakash Panangaden, Denis Thérien, Gabor Lugosi, Geoff Holmes, Bernhard Pfahringer, and Osamu Watanabe.

I would never have gotten into machine learning research if it was not for Pep Fuertes and Enric Ventura who introduced me to academic research, José L. Balcázar whose lectures on computability sprang my interest on computer science, and Ignasi Belda who mentored my first steps into the fascinating world of artificial intelligence. I thank all of them for giving some purpose to my sleepless nights.

Sometimes it happens that one mixes duty with pleasure and work with friendship. It is my duty to acknowledge that working alongside my good friends Hugo, Joan, Masha, and Pablo has been a tremendous pleasure. Also, this years at UPC would no have been the same without the fun and tasteful company of the omegacix crew and satellites: Leo, David, Oriol, Dani, Joan, Marc, Cristina, Lídia, Edgar, Pere, Adrià, Miquel, Montse, Sol, and many others. A special thank goes to all people who work at the LSI secretariat, for their well-above-the-mean efficiency and friendliness.

Now I must confess that, on top of all the work, these have been some fun years. But it is not entirely my fault. Some of these people must take part of the blame: my long-time friends Carles, Laura, Ricard, Esteve, Roger, Gemma, Txema, Agnès, Marcel, and the rest of the Weke crew; my roomies Marc, Neus, Lluís, Milena, Laura, Laura, and Hoyt; my newly found friends Eulàlia, Rosa, Marçal, and Esther; my tatami partners Sergi, Miquel, and Marina, and my aikido sensei Joan.

Never ever I would have finished this thesis without the constant and loving support from my family, specially my mother Laura and my sister Júlia. To them goes my most heartfelt gratitude.

Contents

Introduction

Finite-state machines (FSM) are a well established computational abstraction dating back to the early years of computer science. Roughly speaking, a FSM sequentially reads a string of symbols from some input tape, writes, perhaps, another string on an output tape, and either accepts or rejects the computation. The main constraint is that at any intermediate step of this process the machine can only be in one of finitely many states. From a theoretical perspective, the most appealing property of FSM is that they provide a seamlessly easy way to define *uniform* computations, in the sense that the same machine expresses the computation to be performed on input strings of all possible lengths. This is in sharp contrast with other simple computational devices like boolean formulae (CNF and DNF) and circuits, which can only represent computations over strings of some fixed length.

Formal language theory is a field where FSM have been extensively used, specially due to their ability to represent uniform computations. Since a formal language is just a set of finite strings, being able to compute membership in such a set with a simple and compact computational device is a great aid both for theoretical reasoning and practical implementation. Already in the 1950's the Chomsky hierarchy identified several important classes of formal languages, with one of them, *regular languages*, being defined as all languages that can be recognized by the class of FSM known as *acceptors*. This class of machines is characterized by the fact that it does not write on an input tape during the computation; its only output is a binary decision – acceptance or rejection – that is observed after all the symbols from the input tape have been read. Despite being equivalent from the point of view of the class of languages they can recognize, finite acceptors are usually classified into either *deterministic finite automata* (DFA) or *non-deterministic finite automata* (NFA). These two classes have different algorithmic properties which are transferred, and even amplified, to some of their generalizations. Such differences will play a central role in the development of this thesis.

Finite-state machines are also an appealing abstraction from the point of view of *dynamical systems*. In particular, FSM are useful devices for representing the behavior of a system which takes a sequence of inputs and produces a sequence of outputs by following a set of rules determined by the internal state of the system. From this point of view the emphasis sometimes moves away from the input-output relation defined by the machine and focuses more on the particular state transition rules that define the system. The relation between input and output sequences can then be considered as observable evidence of the inner workings of the systems, specially in the case where the internal state of the system cannot be observed but the input and output sequences are.

A key ingredient in the modelling of many dynamical systems is *stochasticity*. Either because exact modelling is beyond current understanding for a particular system or because the system is inherently probabilistic, it is natural and useful to consider dynamical systems exhibiting a (partially) probabilistic behavior. In the particular case of systems modelled by FSM, this leads to the natural definition of probabilistic finite-state machines. The stochastic component of probabilistic FSM usually involves the transitions between states *and* the generation of observed outputs, both potentially conditioned by the given input stimulus. A special case with intrinsic interest is that of *autonomous systems*, where the machine evolves following a set of probabilistic rules for transtitioning between states and producing output symbols, without the need for an input control signal.

All these probabilistic formulations of FSM give rise to interesting *generative models* for finite strings and transductions, i.e. pairs of strings. Taking them back to the world of computation, one can use them as devices that assign probabilities to (pairs of) strings, yielding simple models for computing probability

distributions over (pairs of) strings. Furthermore, in the domain of language theory, probabilistic FSM define natural probability distributions over regular languages which can be used in applications, e.g. modelling stochastic phenomena in natural language processing. Besides the representation of computational devices, probabilistic FSM can also be used as *probabilistic generators*. That is, models that specify how to randomly generate a finite string according to some particular distribution. Both as generators and devices for computing probability distributions over strings, probabilistic FSM yield a rich class of *parametric statistical models* over strings and transductions.

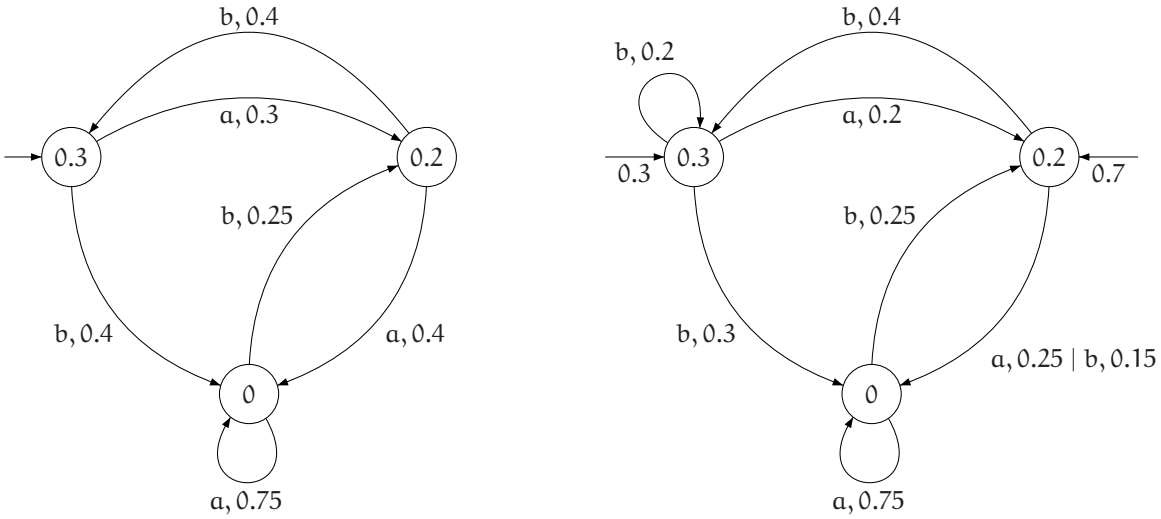
Synthesizing a finite-state machine with certain prescribed properties is, in most cases, a difficult yet interesting problem. Either for the sake of obtaining an interpretable model, or with the intent of using it for predicting and planning future actions, algorithms for obtaining FSM that conform to some behavior observed in a real-world system provide countless opportunities for applications in fields like computational biology, natural language processing, and robotics. The complexity of the synthesis problem varies a lot depending on several particular details, like: what class of FSM can the algorithm produce as hypothesis, and in what form are the constraints given to the algorithm. Despite this variety, most of these problems can be naturally cast into one of the frameworks considered in *learning theory*, a field at the intersection of computer science and statistics that studies the computational and statistical aspects of algorithms that learn models from data. Broadly speaking, the goal of the statistical side of learning theory is to argue that the model resulting from a learning process will not only represent the training data accurately, but that it will also be a good model for *future* data. The computational side of learning theory deals with the algorithmic problem of choosing, given a very large, possibly infinite set of hypotheses, a (quasi-)optimal model that agrees, in a certain sense, with the training data.

Designing and analyzing learning algorithm for several classes of FSM under different learning frameworks is the object of this thesis. The probabilistic analogs of DFA and NFA – known as PDFA and PNFA – are the central objects of study. The algorithmic principles used to learn those classes are also generalized to other classes, including probabilistic transducers and generalized acceptors whose output is real-valued instead of binary valued. Our focus is on efficient algorithms, both from the computational and statistical point of view, and on proving formal guarantees of efficiency in the form of polynomial bounds on the different complexity measures involved.

The first part of the thesis discusses algorithms for learning *Probabilistic Deterministic Finite Automata* (PDFA) and generalizations of those. These objects define probability distributions over sets of strings by means of a Markovian process with finite state space and partial observability. The statement of our learning tasks will generally assume that the training sample was actually generated by some unknown PDFA. In this setting, usually referred to as the *realizable case*, the goal of a learning algorithm is to recover a distribution close to the one that generated the sample using as few examples and running time as possible. If we insist, and that will be the case here, that the learned distribution is represented by a PDFA as well, then learning is said to be *proper*. In machine learning, the problem of using randomly sampled examples to extract information about the probabilistic process that generates them falls into the category of *unsupervised learning* problems. Here, the term unsupervised refers to the fact that no extra information about the examples is provided to the learning process; specially, we assume that the sequence of states that generated each string is not observed by the algorithm. Thus, we will be focusing on the computational and statistical aspects of realizable unsupervised proper learning of PDFA.

Despite the fact that PDFA are not among the best known models in machine learning, there are several arguments in favor of choosing distributions generated by PDFA as a target and hypothesis class. The main argument is that PDFA offer a nice compromise between expressiveness and learnability, which will be seen in the chapters to come. The fact that the parameters characterizing learnability in PDFA have been identified means that we also understand which of them are easy to learn, and which ones are hard. Furthermore, PDFA offer two other advantages over other probabilistic models on sequences: inference problems in PDFA are easy due to their deterministic transition structure, and they are much easier to interpret by humans than non-deterministic models.

The most natural approach to learning a probability distribution from a sample is, perhaps, the maximum likelihood principle. This principle states that among a set of possible models, one should choose the one that maximizes the likelihood of the data. Grounded in the statistical literature, this



(a) A PDFA over $\Sigma = \{a, b\}$

(b) A PNFA over $\Sigma = \{a, b\}$

Figure 1: Examples of probabilistic automata. In each case we have states with stopping probabilities and transitions between them labeled with symbols from Σ and transition probabilities. In the case of PDFA there is only one initial state and for each state and symbol there is at most one outgoing transition. In contrast, PNFA have a distribution over initial states, and states can have several outgoing transitions labeled with the same symbol.

method is proved to be consistent in numerous situations, a very desirable property when learning probability distributions. However, maximizing the likelihood over hypotheses classes of moderate size can be computationally intractable. Two alternatives exist in such situations: one can resort to efficient heuristics that try to maximize (an approximation of) the likelihood, or one can seek alternative learning methods that do not rely on likelihood maximization. The Expectation–Maximization (EM) algorithm is an heuristic that falls into the former class: it is an iterative algorithm for approximately maximizing the likelihood. Though it is not guaranteed to converge to the true optimum, EM is widely used because in practice it performs quite well and is easy to adapt to new probabilistic models. The second class of alternatives are usually more model-specific, since in general they rely on particular properties of the model class and formulate learning goals that do not involve the likelihood of the training data directly. The algorithms we will give for learning PDFA fall into this second class. In particular, they heavily rely on the fact that the target is defined by a DFA with transition and stopping probabilities. The general principle will be to learn an approximation of this DFA and then estimate the corresponding probabilities. Though this does no involve likelihood in any way, it is not a straightforward task either. The general principle on which these algorithms are based is called *state-merging*, and though it resembles an agglomerative clustering scheme, it has its own peculiarities.

An essential obstruction is that learning PDFA is known to be a computationally hard problem in the worst case. Abe and Warmuth showed in [AW92] that learning probabilistic automata in time polynomial in the size of the alphabet is not possible unless $\mathsf{NP} = \mathsf{RP}$. Nonetheless, they also showed that learning with exponential time is possible given only a sample of size polynomial in the number of states and the alphabet size – the barrier is computational, not information-theoretic. Later on, Kearns et al. [KMRRSS94] gave evidence that even for alphabets of size two learning PDFA may be difficult: it is shown there that noisy parities, a class supposed to be hard to PAC learn, can be represented as PDFA.

With these restrictions in mind, the key for obtaining polynomial time bounds on PDFA learning is to realize that the size of the alphabet and the number of states do not represent accurately the hardness of learning a particular automaton. A third parameter called *distinguishability* was introduced in [RST98] for obtaining polynomial bounds for learning acyclic PDFA. In the specific hard cases identified by

previous lower-bounds this distinguishability is exponential in the number of states. And in many other cases it is just constant or polynomial on the number of states and alphabet size. Thus, introducing this third parameter yields upper bounds that scale with the complexity of learning any particular PDFA – and in the worst cases, this upper bounds usually match the previous lower bounds. It should be noted here that this is neither a trick nor an artifact. In fact, it responds to a well established paradigm in *parametrized complexity* that seeks to identify parametrizations of computationally hard problems yielding insights into how the complexity scales between the different sub-classes of that problem.

A surprising fact about PNFA is that they can represent probability distributions that cannot be represented by any PDFA. This is in sharp contrast with the acceptor point of view in which DFA and NFA can represent exactly the same class of languages. Though the same lower bounds for learning PDFA apply to learning PNFA as well, the panorama with the upper bounds is more complicated. In particular, since PNFA are strictly more general and have an underlying non-deterministic transition system, state-merging algorithms for learning PDFA do not provide a practical solution for learning PNFA. This is because, although one can approximate every PNFA by a large enough PDFA, the obvious learning algorithm based on this reduction ends up not being very efficient. Said differently, the clustering-like approach which lies at the heart of state-merging algorithms is not useful for PNFA because clustering states in the latter yields a number of cluster much larger than the number of states as a FSM. In fact, almost all existing efficient algorithms for learning non-deterministic FSM rely on algebraic and spectral principles very distant from the state-merging paradigm used in learning algorithms for PDFA.

This striking difference between the techniques employed for learning PDFA and PNFA will establish a central division on the contributions of this thesis. While state-merging algorithms for learning PDFA are in essence greedy and local, spectral methods for learning PNFA perform global operations to infer a state-space all at once. Tough these two approaches share some intuitions at different levels, the content and organization of this thesis will be mostly shaped by the differences between them. In particular, the following three key factors will drive our explorations and determine their limits. The first factor is the difference on the types of FSM that can learned with these two techniques. While state-merging algorithms can be extended to any machine with an underlying deterministic structure that somehow can be inferred from statistical evidence, spectral methods are able to construct a non-deterministic state space defining a machine that matches a set of concrete observations with a particular algebraic structure. The second factor arises from the first one by observing that both methods differ not only on the type of machines they can learn, but also on the kind of information they require about the target. The state-merging paradigm is based upon telling states apart using a sufficiently large amount of information generated from each states. On the other hand, the spectral approach requires a particular set of observations to be (approximately) known. This difference will set a distinction between the kinds of learning frameworks where each of the methods can be applied. The last is a formal factor: different learning algorithms require different analysis techniques. In this respect, the difference between the two methods is abysmal. Statistical tests and inductive greedy arguments are the main tools used to reason about state-merging algorithms. In contrast, the analysis of spectral methods and its variations relies on perturbation results from linear algebra and convex optimization. There is, however, a common tool used in both types of analyses: concentration inequalities of probability on product spaces. In essence, these technical disparities can be summarized by saying that state-merging algorithms are combinatorial in nature, while spectral methods are heavily based on linear algebra primitives. This is also reflected in the way both methods are usually implemented: while state-merging tend to require specialized libraries for graph data structures, spectral methods are easy to implement inside common mathematical software packages for matrix computations. Using the problem of learning FSM as an underlying plot, we present results along these two very different lines of work in a single thesis with the hope that it will help researchers in each of these separate areas become acquainted with techniques used in the other area for solving similar problems.

In the next two sections we provide a summary of the contributions presented in this thesis and a detailed account of previous work directly related to the topics addressed here.

Overview of Contributions

In the first four chapters of this dissertation we study the problem of learning PDFA defining probability distributions over some free monoid $\Sigma^\star$ by using an algorithmic paradigm called *state-merging*. The main idea behind state-merging algorithms is that, since the underlying transition graph of the target is a DFA, one can recover such DFA by starting from the initial state, greedily conjecture new states reached from previous states by a single transition, and test whether these are really new or correspond to already discovered states – in which case a *merge* of states occurs.

Chapter 1 begins by describing how to implement a simple state-merging algorithm in a learning framework based on *statistical queries*. In such framework, instead of a sample containing i.i.d. examples from the target distribution, the learning algorithm is given access to an oracle that can give approximate answers to simple queries about the target distribution. Among others, this has the advantage of separating algorithmic and statistical issues in learning proofs. In particular, using this method we are able to give a full proof of the learnability of PDFA which is much shorter than other existing proofs and can actually be taught in a single session of a graduate course on learning theory or grammatical inference. This simplicity comes at the price of our proof being slightly less general than previous PDFA learning results, in that our learning guarantees are only in terms of total variation distances instead of relative entropy, and that our algorithm requires one more input parameter than previous ones. Though these two weaknesses can be suppressed by using much longer proofs, we chose to present a simple, short, and self-contained proof instead. In the second part of Chapter 1 we take on the characterization of state-merging algorithms in terms of statistical queries and use it to proof a lower bound on the total number and accuracy of the queries required by an algorithm capable of learning the class of all PDFA. This is the first lower bound on learning PDFA where the distinguishability of the target appears explicitly. Furthermore, the fact that a key ingredient in constructing our lower bound are PDFA realizing distributions over parities sheds light on an interesting fact: though previous lower bounds have suggested that learning PDFA is hard because they can realize *noisy* parities, an obstruction to the efficiency of state-merging methods is actually realized by *noiseless* parities. The main conclusion one can draw from this result is that any general approach for learning the underlying structure of a PDFA will essentially face these same limitations. Results presented in this chapter have been published in the conference paper [BCG10a] and the journal paper [BCG13].

The main role of statistical queries in the process of learning a PDFA via a state-merging approach is to test whether two states in the hypothesis correspond to the same state in the target. Answering this question using examples drawn from the target distribution involves testing whether two samples of strings were generated by the same distribution. Chapter 2 addresses the problem of designing statistically and computationally efficient statistical tests for answering this type of question, and proving formal guarantees on the success of such tests. We derive different tests for solving this task, all of them using a variant of the distinguishability called *prefix-distinguishability* which yields more statistically efficient distinction in most cases. The simplest versions of our tests are based on the Vapnik–Chervonenkis (VC) theory of uniform convergence of empirical frequencies to their expected values. These tests are not new but we give a simple and unified treatment. Motivated by a weakness of VC-based tests on certifying *equality*, we propose to use tests based on Efron's *bootstrap* to improve the statistical efficiency on this task. We demonstrate the advantages of our approach with an extensive set of experiments, and provide formal proofs showing that in the worst case bootstrapped-based tests will perform similarly to VC-based tests. Our analyses provide the first finite-sample bounds for confidence intervals of bootstrap estimators from distributions over strings. In the second part of this chapter we address an essential algorithmic question of distribution similarity testing: namely, whether the whole samples needs to be stored in memory in order to conduct statistically correct tests. We show that accurate testing can be performed by just storing a summary of both samples which can be constructed on-line by observing each example only once, storing its relevant information in the summary, and forgetting it. Data structures performing these operations are called *sketches* and have been extensively studied in the *data streams* computing paradigm. Assuming the existence of such data structures, we show how to achieve a reduction of a square root factor on the total memory needed to test distribution similarity. A by-product of this reduction in memory usage is also a reduction of the time needed for testing due to a reduction of the number of comparisons in testing routines. We give similarity tests with formal guarantees based on

sample summaries using both VC bounds and the bootstrap principle. Our bootstrap-based tests using summarized samples are the first of their kind and could be useful in many other tasks as well. Some of these results have been published without proofs in a conference paper [BCG12b].

Similarity tests that use information stored in sketches built from an on-line stream of examples are the first step towards a full on-line implementation of the state-merging paradigm for learning PDFA from data streams. Building such an implementation on top of the tests from Chapter 2 is the goal of Chapter 3. The main result of that chapter is a learning algorithm for PDFA from examples arriving in a data stream format. This requires the algorithm to process each new item in constant time and use an amount of memory that is only allowed to grow slowly with the number of processed items. We show that our algorithm satisfies these stringent requirements; one of the key contributions to achieve this result is the design of new sketching data structures to find frequent prefixes in a stream of strings. In addition, we show that if the stream was generated from a PDFA then our algorithm is a proper PAC learner. Since in a real streaming environment the data-generating process might not be stationary, we also give a change detector for discovering changes in the stream so that our learning algorithm can be adapted to react whenever one of such changes occur. We also provide an algorithm for efficiently searching for the correct parameters – number of states and distinguishability – required by the state-merging procedure. A preliminary version of these results appeared in a conference paper [BCG12b]. An extended journal version of this work is currently under review [BCG12a].

Chapter 4 is the last one of the first part of the thesis. In it we extend the state-merging paradigm to *Generalized PDFA* (GPDFA). Roughly speaking, these are defined as any probabilistic finite state-machine defining a probability distribution over a free monoid of the form $(\Sigma \times \mathcal{X})^\star$, where Σ is a finite alphabet and $\mathcal{X}$ is an arbitrary measure space, and whose state transition behavior is required to be deterministic when projected on $\Sigma^\star$. Examples include: machines that model the time spent in each transition with some real random variable, in which case $\mathcal{X} = \mathbb{R}$; or a sub-class of sequential transductions, in which case $\mathcal{X} = \Delta$ is a finite output alphabet. We begin by taking an axiomatic approach assuming that we can test whether two states in a hypothesis GPDFA are equal and that we can estimate the probability distributions associated with $\mathcal{X}$ in each transition. Under such assumptions, we show a full PAC learning result for GPDFA in terms of relative entropy. Finally we give specific implementations of this algorithm for the two examples discussed above. This chapter is based on some preliminary results presented in [BCG10b].

In the second part of this dissertation we study learning algorithms based on spectral decompositions of Hankel matrices. A Hankel matrix encodes a function $f : \Sigma^\star \to \mathbb{R}$ in such a way that the algebraic structure of the matrix is intimately related to the existence of automata computing f. Since this is true for non-deterministic automata and even more general classes of machines, these methods can be used to derive learning algorithms for problems where state-merging algorithms are not useful.

In total, this part comprises five chapters. In the first of those, Chapter 5, the notion of Hankel matrix is introduced. We then use this concept to establish a duality principle between factorizations of Hankel matrices and minimal Weighted Finite Automata (WFA), from which a derivation of the spectral method follows immediately. The algorithmic key to spectral learning – and the key to its naming as well – is the use of the singular vector decomposition (SVD) for factorizing Hankel matrices. Perturbation bounds known for SVD can be used to obtain finite sample error analyses for this type of learning algorithms. In the last part of Chapter 5 we give several technical tools that will be useful for obtaining such bounds in subsequent chapters. The purpose of collecting these tools and their proofs here is twofold. The first is to provide a comprehensive source of information on how these technical analyses work. This is useful because all the published analyses so far are for very specific situations and it is difficult sometimes to grasp the common underlying principles. The second reason for re-stating all these results here is to give them in a general form that can be used for deriving all the bounds proved in subsequent chapters. Some of this material have appeared in the conference papers [BQC12; BM12].

The first application of the spectral method is given in Chapter 6. In it we derive a PAC learning algorithm for PNFA that can also be applied to other classes of stochastic weighted automata. A full proof of the result is given. This involves obtaining concentration bounds on the estimation accuracy of arbitrary Hankel matrices for probability distributions over $\Sigma^\star$, and applying bounds derived in the previous chapter for analyzing the total approximation error on the output of the spectral method. Our

analysis gives a learning algorithm for the full class of PNFA and in this sense it is the most general result to date on PAC learning of probabilistic automata. We also show how to obtain reductions for learning distributions over prefixes and substrings using the same algorithm. Though the full analysis is novel and unpublished, some technical results have been published in a conference paper [LQBC12] and some other are currently under review for publication in a journal [BCLQ13].

Chapter 7 gives a less straightforward application of spectral learning to probabilistic transducers. There we show how to use the spectral method for learning a conditional distribution on input-output pairs of strings of the same length over $(\Sigma \times \Delta)^\star$. Our main working assumption is that these conditional probabilities can be computed by a probabilistic FSM with input and output tapes. We use most of the tools from the two previous chapters while at the same time having to deal with a new actor: the probability distribution D_X that generates the input sequences in the training sample. In a traditional distribution-free PAC learning setting we would let D_X be an arbitrary distribution. We were not able to prove such general results here; and in fact, the general problem can become insurmountably difficult without making any assumptions on the input distribution. Therefore, our approach consists on imposing some mild conditions on the input distribution in order to make the problem tractable. In particular, we identify two non-parametric assumptions which guarantee that the spectral method will succeed with a sample of polynomial size. A preliminary version of these results was published in [BQC11] as a conference paper.

The algebraic formulation of the spectral method is in sharp contrast with the analytic formulation of most machine learning algorithms where some optimization problem is solved. Despite its simplicity, the algebraic formulation has one big drawback: it is not clear how to enforce any type of prior knowledge into the algorithm besides the number of states in the hypothesis. On the other hand, a very intuitive way to enforce prior knowledge in optimization-based learning algorithms is by adding regularization terms to the objective function. This facts motivate Chapter 8, where we design an optimization-based learning algorithm for WFA founded upon the same modeling principles of the spectral method. In particular, we give a non-convex optimization algorithm that shares many statistical properties with the spectral method, and then show how to obtain a convex relaxation of this problem. These results are mostly based on those presented in [BQC12].

Chapter 9 explores the possibility of using the spectral method for learning non-stochastic WFA. We formalize this approach in a regression-like setting where the training dataset contains pairs of strings and real labels. The goal is to learn a WFA that approximates the process assigning real labels to strings. Our main result along these lines is a family of learning algorithms that combine spectral learning with a matrix completion technique based on convex optimization. We prove the first generalization bound for an algorithm in this family. Furthermore, our bound holds in the agnostic setting in contrast with all previous analyses of the spectral method. This chapter is entirely based on the conference paper [BM12].

Though the main concerns of this thesis are the theoretical aspects of learning finite-state machines, most of the algorithms presented in this thesis have been implemented and tested on real and synthetic data. The tests from Chapter 2 in the data streams context and the state-merging algorithm for learning PDFA from data streams of Chapter 3 have all been implemented by myself and will be published as open source shortly. The spectral learning algorithms from Chapters 6, 7, and 8 have been extensively used in experiments reported in the papers [BQC11; LQBC12; BQC12; BCLQ13]. These experiments have not been included in the present thesis because the implementations and analyses involved were carried out almost entirely by my co-authors.

Related Work

Countless definitions and classes of finite-state machines have been identified, sometimes the differences being of a very subtle nature. A general survey about different classes of probabilistic automata that will be used in this thesis can be found in [VTHCC05a; VTHCC05b; DDE05]. These papers also include references to many formal and heuristic learning algorithms for these types of machines. Good references on weighted automata and their algorithmic properties are [BR88; Moh09]. Depending on the context, weighted automata can also be found in the literature under other names; the most notable examples are Observable Operator Models (OOM) [Jae00], Predictive State Representations (PSR) [LSS01], and

rational series [BR88]. For definitions, examples, and properties of transducers one should consult the standard book [Ber79].

The problem of learning formal languages and automata under different algorithmic paradigms has been extensively studied since Gold's seminal paper [Gol67]. Essentially, three different paradigms have been considered. The *query learning* paradigm is used to model the situation where an algorithm learns by asking questions to a teacher [Ang87]. An algorithm *learns in the limit* if given access to an infinite stream of examples converges to a correct hypothesis after making a finite amount of mistakes [Gol67]; this model makes no assumption on the computational cost of processing each element in the stream. In the *Probably Approximately Correct* (PAC) learning model an algorithm has access to a finite random sample and has to produce a hypothesis which is correct on most of the inputs [Val84]. Since the literature on these results if far too large to survey here, we will just give some pointers to the most relevant papers from the point of view of the thesis; the interested reader can dwell upon this field starting with [Hig10] and references therein. In addition, the synthesis of FSM with some prescribed properties was also studied from a less algorithmic point of view even before the formalization of learning theory; see [TB73] for a detailed account of results along these lines.

The state-merging approach – and its "dual" state-splitting – for learning automata have been around the literature for a long time. They have been used in formal settings for learning sub-classes of regular languages like k-reversible [Ang82] and k-testable [GV90] languages. They are also the basis for several successful heuristic algorithms for learning DFA like RPNI [OG92] and Blue-Fringe [LPP98]. In the case of probabilistic automata, a key milestone is the state-merging ALERGIA algorithm [CO94] for learning PDFA in the limit. Further refinements of this approach lead to the PAC learning algorithms for PDFA on which this thesis builds upon. The first result along these lines is the algorithm for learning *acyclic* PDFA in terms of relative entropy from [RST98]; this is where the *distinguishability* of a PDFA was first formally defined. Latter on Clark and Thollard [CT04a] extended this result to the class of all PDFA; in addition to the distinguishability, their PAC bounds included the maximum expected length of strings generated from any state. By using the total variation instead of relative entropy as their error measure, the authors of [PG07] showed that a similar result could be obtained without any dependence on the expected length. Learning PDFA with a state-merging strategy based on alternative definitions of distinguishability was the topic investigated in [GVW05]. Despite working in polynomial time, these PAC learning algorithms for PDFA were far from practical because they required a lot if input parameters and access to a sampling routine to retrieve as many examples as needed. In [GKPP06; CG08] both of this restrictions were relaxed while still providing algorithms with PAC guarantees. In particular, their algorithms needed much less input parameters and could be executed with a sample of any size – their results state that the resulting PDFA is a PAC hypothesis once the input sample is large enough; on the quantitative side, the sample bounds in [CG08] are also much smaller than previous ones.

In general, learning finite automata from random examples is a hard problem. It was proved in [PW93] that finding a DFA or NFA consistent with a sample of strings labeled according to their membership to a regular language, and whose size approximates the size of a minimal automaton satisfying such constrain, is an NP-hard problem. In addition, the problem of PAC learning regular languages using an arbitrary representation is as hard as some presumably hard cryptographic problems according to [KV94]. These results deal with learning arbitrary regular languages in a *distribution independent* setting like the PAC framework. On the other hand, [CT04b] showed that it is possible to learn a DFA under a probability distribution generated by a PDFA whose structure is "compatible" with that of the target. Together with the mentioned lower bounds this suggests that focusing on particular distributions may be the only way to obtain positive learning results for DFA. It is not surprising that with some extra effort Clark and Thollard were able to extend their result to an algorithm that actually learns PDFA [CT04a]. Actually, since state-merging algorithms for learning PDFA output a DFA together with transition probabilities, one motivation for studying such algorithms is because they provide tools for learning DFA under certain distributions.

Deciding whether to merge two states or not is the basic operation performed in state-merging algorithms. ALERGIA and successive state-merging algorithms for PAC learning PDFA relied on statistical tests based on samples of strings. The basic question is whether two samples were drawn from the same distribution; this is sometimes known as the *two-sample* problem in the literature. Although this prob-

lem has been extensively studied in the statistics literature, almost all existing approaches rely on very restrictive modelling assumptions and provide asymptotic results which are not useful for obtaining PAC bounds. Thus, all mentioned PAC learning algorithms relied on ad-hoc tests built around concentration bounds like Hoeffding's inequality. In this thesis we give a more systematic treatment of this problem. A few similar studies have recently appeared in the machine learning literature, though none of them have been directly applied to probability distributions over strings. From a computational perspective, the paper [BFRSW13] and references therein study the problem of distinguishing two distributions in terms of total variation distance and mean square distance. In [GBRSS12] the authors provide three different statistical tests for the two-sample problem with distributions over reproducing kernel Hilbert spaces; two of them come with finite sample bounds derived from concentration inequalities. They also show how to apply a sub-sampling strategy to reduce the running time of their testing algorithms, an idea that could be applied to data streams as well. See references therein for previous results on kernel-based approaches to the two-sample testing problem. Efron's bootstrap [Efr79] is a generic technique for statistical inference that, among many other things, can be used for constructing two-sample tests; we refer to the excellent book [Hal92] for an in-depth statistical treatment of the many uses and theoretical properties of the bootstrap. Learning algorithms have found uses of the bootstrap in many contexts. For example, to define the *bagging* method for ensemble learning [Bre96], and to construct penalty terms for model selection with guarantees [Fro07]. In the context of testing similarity distributions we have only found the paper [FLLRB12]; in it the authors combine kernel-methods with the bootstrap for obtaining two-sample tests with finite sample bounds.

The *data streams* computational model provides an interesting framework for algorithmic problems dealing with huge amounts of data that cannot be stored for later processing [Agg07]. In this model each data item in an infinite stream is presented to the algorithm once, which has to process it in almost constant time and discard it; the total memory used by the algorithm is constrained to grow very slowly with the number of processed items. In recent years the data mining community has embraced the model to represent many big data applications related to industry, social networks, and bioinformatics. The list of mining and learning algorithms that have been ported to this new algorithmic paradigm is too long to be surveyed here; instead we refer to [Gam10; Bif10] for extensive treatments on this subject. Most algorithmic advances on the efficiency of data streams algorithms have been driven forth by the design of sketching data structures; see [Mut05] for a comprehensive list of examples. The problem of learning PDFA has not been addressed before in the data streams framework. However, the problem of testing whether two streams are generated from the same distribution, for which we give testing algorithms, has been considered in [AB12]. The test given by the authors of this manuscript comes without finite sample guarantees, and is not based on the bootstrap like ours. The only usage of the bootstrap in a big data scenario we have found in the literature is the recent paper [KTSJ12]; in it the authors derive bootstrap-like resampling strategies for large amounts of data.

Deterministic classes of transducers and automata with timed transitions are all susceptible to state-merging learning algorithms. A classical algorithm for learning subsequential transducers based on the state-merging paradigm is OSTIA [OGV93]; see [Hig10] for a detailed account of several variations of this algorithm in different contexts. Several models of automata with timing information have been considered in the literature, including: acceptors where transitions can specify clock guards imposing restrictions on the time spent on each state [AD94], and stochastic generators modelling the time spent on each state with an exponential random variable [Par87]. The thesis [Ver10] contains a comprehensive list of learning results for these and other related models, most of which are based on state-merging algorithms of some sort; though most are just heuristics, some of them come with learning in the limit guarantees. We have not been able to find PAC learning results in the literature for any of these models.

Learning algorithms based on linear-algebraic principles like the spectral method have been around for quite a long time. The most similar approach in the literature on learning theory of finite automata is the algorithm in [BBBKV00] for learning multiplicity automata using membership and equivalence queries. In the field of control theory, subspace methods are a popular family of algorithms for identification of linear dynamical systems [Nor09]. The basic problem in this setting is to identify a matrix governing the evolution of a discrete-time dynamical system from observations which are linearly related to the state-vector of the system. From an automata point of view, this problem corresponds to learning a

FSM with a single output symbol and many states, from which at each step we can observe a linear transformation of its state vector. A landmark on this topic that shares several ideas with our spectral method is the N4SID algorithm [VODM94] which is implemented in many software toolkits for system identification. The literature on OOM has also produced some learning algorithms with features common to SVD-based methods [ZJT09]. These models are almost identical to the weighted automata we will consider in this thesis, thought the original motivation for studying them was to understand the possible dynamics of linear systems with partial observability. Statisticians addressing the identifiability problem of HMM and phylogenetic trees have been using linear algebra in a similar way for quite a while [Cha96]; an algorithmic implementation of these ideas was presented in [MR05].

Taking on these results, the spectral method as we understand it in this thesis was described and analyzed for the first time in [HKZ09] in the context of HMM and in [BDR09] in the context of stochastic weighted automata. The derivation based on a duality principle that we give here is original but essentially yields the same algorithm. A significant difference with [HKZ09] is that, in deriving their algorithm, they make a set of assumptions on the target automaton to be learned which we do not need. With respect to [BDR09], our analysis yields much stronger learning guarantees.

Since these two seminal papers were published in 2009, the field of spectral learning algorithms for probabilistic models has exploded. Variations on the original algorithm for learning probabilistic models on sequences include: [SBG10] on learning Reduced Rank HMM where, though still assuming a factorization between transition and emission, the rank restriction from [HKZ09] was relaxed; [SBSGS10] on learning Kernel HMM that model FSM with continuous observations; in [BSG11] a consistent spectral learning algorithm for PSR was given; in order to force the output of the learning algorithm to be a proper probability distribution, [Bai11b] gave an algorithm for learning stochastic quadratic automata; the authors of [FRU12] gave a new analysis of spectral learning for HMM with improved guarantees by using a tensorized approach and making some extra assumptions on the target machine. Besides variations of HMM, spectral learning algorithms for several classes of graphical models have been given following a very similar approach. Notable examples include: [PSX11; ACHKSZ11] on learning the parameters and structure of tree-shaped latent variable models; algorithms for Latent Dirichlet Allocation [AFHKL12b; AFHKL12a] and other topic models [DRFU12]; and, several classes of mixture models [AHK12a; AHHK12; AHK12b]. A recent line of work tries to replace decompositions of matrices with decompositions of tensors in spectral algorithms [AGHKT12]. From a computational linguistics point of view, a natural idea is trying to generalize algorithms for learning automata to algorithms for learning context-free formalisms. Works along these lines include: [BHD10] on learning stochastic tree-automata; the results in [LQBC12] on learning split-head automata grammars; an application of spectral learning to dependency parsing [DRCFU12]; and, a learning algorithm for probabilistic context-free grammars [CSCFU12]. Models arising from spectral learning have been used in [CC12] for improving the computational efficiency of parsing.

Part I

State-merging Algorithms

Chapter 1

State-Merging with Statistical Queries

In this first chapter we study the problem of learning PDFA in the Statistical Query model, where learning algorithms are given access to oracles that provide answers to questions related to the target distribution. Studying learning problems in this computational model has certain advantages over the usual PAC model where the learner is presented with a random sample generated by the target distribution. The most interesting of these advantages is, perhaps, the fact that learning proofs in this model are simpler and much more conceptual than in the PAC model, where learning proofs usually contain a great deal of probability concentration arguments that tend to blur the main ideas behind the *why* and *how* the algorithm works. Learning results in the PAC model can be easily obtained from learning results in query models once it is shown that those queries can be successfully simulated using randomly sampled examples.

One contribution of this chapter is introducing a Statistical Query framework for learning probability distributions and giving an algorithm for learning PDFA in this model. Despite the fact that the result is already known in the literature, we believe that our learning proof – which is just three pages long – illuminates the most important and subtle points in previous learning proofs in the PAC model, and at the same time condenses the facts that make the class of PDFA learnable via state-merging algorithms. Our proof may be taught in a single lecture in a graduate course on learning theory or grammatical inference, a feature that will hopefully make the result more accessible to students and researchers in the field. On the other hand, we give efficient simulations of the queries used in our model, which yield yet another proof of the PAC-learnability of PDFA.

Query models are also useful for proving unconditional lower bounds on the complexity of learning problems. The idea is that when calls to an oracle are the only source of information a learning algorithm has about its target, it is possible to control how much information an algorithm is able to gather after a fixed number of queries. In contrast, when learning from examples it is extremely difficult to control how much information an algorithm can extract from a sample in a given amount of computational time. Thus, another contribution of this chapter is proving a lower bound on the complexity of learning PDFA in the statistical query model which confirms the intuition that the difficulty of the problem depends on the *distinguishability* of the target distribution. Further consequences and interpretations of our lower bound are discussed at the end of the chapter.

1.1 Strings, Free Monoids, and Finite Automata

The set $\Sigma^\star$ of all strings over a finite alphabet Σ is a *free monoid* with the concatenation operation. Elements of $\Sigma^\star$ will be called strings or words. Given $x, y \in \Sigma^\star$ we will write either $x \cdot y$ or simply xy to denote the concatenation of both strings. Concatenation is an associative operation. We use λ to denote the empty string which satisfies $\lambda \cdot x = x \cdot \lambda = x$ for all $x \in \Sigma^\star$. The length of a string $x \in \Sigma^\star$ is denoted

by $|x|$. The empty string is the only string with $|\lambda| = 0$. For any $k \geq 0$ we denote by Σ^k and $\Sigma^{\leq k}$ the sets all strings of length k and all strings of length at most k, respectively. We use Σ^+ to denote $\Sigma^\star \setminus \{\lambda\}$.

A *prefix* of a string $x \in \Sigma^\star$ is a string u such that there exists another string v satisfying $x = uv$. String v is a *suffix* of x. We will write $u \sqsubseteq_0 x$ and $v \sqsubseteq_\infty x$ to denote that u and v are respectively prefixes and suffixes of x. A prefix (suffix) is called *proper* if $|u| < |x|$ ($|v| < |x|$). If $x = uv$, the pair (u, v) is a *partition* of x.

The concatenation operation is extended to sets of strings as follows: given $W_1, W_2 \subseteq \Sigma^\star$, we define their concatenation as $W_1W_2 = \{w_1w_2 \mid w_1 \in W_1 \wedge w_2 \in W_2\}$. In the case of a singleton $\{x\}$ and a set W, we write xW instead of $\{x\}W$. A set $W \subseteq \Sigma^\star$ is *prefix-free* if for every $x \in W$ and any proper prefix $u \sqsubseteq_0 x$ we have $u \notin \Sigma^\star$. Another way to say the same is that for every $x \in W$ we have $W \cap x\Sigma^\star = \{x\}$. An important property is that if W is a prefix-free set of strings and W' is an arbitrary set of strings, then for each $w' \in W'$ there is at most one $w \in W$ such that $w \sqsubseteq_0 w'$.

A string w is a *substring* of x if there exist a prefix $u \sqsubseteq_0 x$ and a suffix $v \sqsubseteq_\infty x$ such that $x = uwv$. Note that, contrary to other formulations, in our definition the symbols of w must be contiguous in x. The notation $|x|_w$ will be used to denote the number of times that w appears as a substring on x, meaning

$$|x|_w = |\{(u, v) \mid u \sqsubseteq_0 x \wedge v \sqsubseteq_\infty x \wedge x = uwv\}| \quad .$$

Given a string $x = x_1 \cdots x_t$ we write $x_{i:j}$ to denote the substring $x_i \cdots x_j$.

A Deterministic Finite Automaton (DFA) over Σ is given by a tuple $A = \langle \Sigma, Q, q_0, \tau, \phi \rangle$, where Q is a finite set of *states*, $q_0 \in Q$ is a distinguished *initial state*, $\tau : Q \times \Sigma \to Q$ is a partial *transition function*, and $\phi : Q \to \{0, 1\}$ is the *termination function*. The number of states $|Q|$ of A is sometimes denoted by $|A|$ or simply n. Note that for some pairs (q, σ) the transition $\tau(q, \sigma)$ might not be defined. When τ is a total function we say that A is *complete*. Giving ϕ is equivalent to giving the set $F = \phi^{-1}(1)$ of *accepting states*. The transition function can be inductively extended to a partial function $\tau : Q \times \Sigma^\star \to Q$ by setting $\tau(q, \sigma x) = \tau(\tau(q, \sigma), x)$, where the value is undefined whenever $\tau(q, \sigma)$ is not defined, and $\tau(q, \lambda) = q$ for all $q \in Q$. The *characteristic function* of A is $f_A : \Sigma^\star \to \{0, 1\}$ with $f_A(x) = \phi(\tau(q_0, x))$, where we assume that ϕ returns 0 on an undefined output of τ. The *language* of A is the set $\mathcal{L}(A) = f_A^{-1}(1) \subseteq \Sigma^\star$. It is easy to modify A into a complete DFA with the same language by adding a rejecting state, with every transition going to itself, and all previous undefined transitions going to this new state.

Give $q \in Q$ we define the set of all strings that go from q_0 to q, and the set of all strings that go from q_0 to q without using q in any intermediate step:

$$A[\![q]\!] = \{x | \tau(q_0, x) = q\}$$
$$A[q] = \{x | \tau(q_0, x) = q \wedge \forall y, z : x = yz \to (|z| = 0 \vee \tau(q_0, y) \neq q)\}$$

Note that by definition $A[q]$ is always a prefix-free set.

A *Probabilistic Deterministic Finite Automaton* (PDFA) over Σ is a tuple $A = \langle \Sigma, Q, q_0, \tau, \gamma, \phi \rangle$, where Q, q_0, and τ are defined as in a DFA. In addition we have the *transition probability* function $\gamma : Q \times \Sigma \to [0, 1]$ and the *stopping probability* function $\phi : Q \to [0, 1]$. These functions must satisfy the following constraints: if $\tau(q, \sigma)$ is undefined, then $\gamma(q, \sigma) = 0$; and for all $q \in Q$, we have $\phi(q) + \sum_{\sigma \in \Sigma} \gamma(q, \sigma) = 1$. In a similar way we did for a DFA, we can define an extension $\gamma : Q \times \Sigma^\star \to [0, 1]$ as follows: $\gamma(q, \lambda) = 1$, and $\gamma(q, \sigma x) = \gamma(q, \sigma) \cdot \gamma(\tau(q, \sigma), x)$. We note that if $\tau(q, x)$ is undefined, then we will get $\gamma(q, x) = 0$. The function computed by A is $f_A : \Sigma^\star \to [0, 1]$ defined as $f_A(x) = \gamma(q_0, x) \cdot \phi(\tau(q_0, x))$, where $\phi(\tau(q_0, x)) = 0$ for an undefined $\tau(q_0, x)$. It is well known that if for all $q \in Q$ there exists some $x \in \Sigma^\star$ such that $\phi(\tau(q, x)) > 0$, then f_A is a probability distribution over $\Sigma^\star$; that is, $\sum_{x \in \Sigma^\star} f_A(x) = 1$. Given any $q \in Q$ we write $A_q = \langle \Sigma, Q, q, \tau, \gamma, \phi \rangle$ to denote the PDFA obtained from A by taking q to be the starting state. If D is a distribution over $\Sigma^\star$ that can be realized by a PDFA – that is, there exists A such that $f_A = D$ – then we use A_D to denote any minimal PDFA for D. Here, minimal means a PDFA with the smallest number of states. The smallest non-zero stopping probability of a PDFA A will be denote by $\pi_A = \min_{q \in \phi^{-1}((0,1])} \phi(q)$. The expected length of strings generated by a PDFA A is $L_A = \sum_{x \in \Sigma^\star} |x| f_A(x)$. If D is an arbitrary distribution over strings, we shall write L_D to denote its expected length.

An important structural result about distributions generated by PDFA is that the length of strings they generate always follows a sub-exponential distribution. See Appendix A.1 for more information on sub-exponential distributions.

Lemma 1.1.1. *For any PDFA A there exists a c_A such that $\mathbb{P}_{x\sim A}[|x| \geq t] \leq \exp(-c_A t)$ holds for all $t \geq 0$.*

Proof. Let n be the number of states of A and let q be any state. Starting from state q and before generating n new alphabet symbols, there is a probability $\pi > 0$ of stopping. Thus,

$$\mathbb{P}_{x\sim A}[|x| \geq t] \leq \mathbb{P}_{x\sim A}[|x| \geq \lfloor t/n \rfloor n] \leq (1-\pi)^{\lfloor t/n \rfloor} \leq \exp(-\pi \lfloor t/n \rfloor) \quad . \qquad \square$$

When D is a distribution realized by a PDFA, we may write c_D instead of c_{A_D} because being sub-exponential is clearly a property that depends only on the distribution.

If D is any probability distribution over $\Sigma^\star$ and $W \subseteq \Sigma^\star$ is a set of strings, the probability of W is defined as $D(W) = \sum_{x\in W} D(x)$. The probability under D of generating a word with $x \in \Sigma^\star$ as a prefix is $D(x\Sigma^\star)$.

Given distributions D and D' over $\Sigma^\star$, their *supremum distance* is defined as

$$\mathrm{L}_\infty(D, D') = \max_{x\in\Sigma^\star} |D(x) - D'(x)| \quad .$$

The *total variation distance* between D and D' is given by

$$\mathrm{L}_1(D, D') = \sum_{x\in\Sigma^\star} |D(x) - D'(x)| \quad .$$

Another commonly used measure of discrepancy between probability distributions os the *relative entropy*, also known as *Kullback–Leibler divergence*, which is not an actual distance because it is not symmetric and does not satisfy the triangular inequality. It is given by the following expression:

$$\mathrm{KL}(D\|D') = \sum_{x\in\Sigma^\star} D(x) \log_2 \left(\frac{D(x)}{D'(x)}\right) \quad ,$$

where the $\log_2$ means logarithm with base two. Note that $\mathrm{KL}(D\|D')$ can be infinite if $D'(x) = 0$ for some x such that $D(x) > 0$.

The L_∞-*distinguishability* of a PDFA A is the minimum supremum distance between the distributions generated from each of its states; that is,

$$\mu_A = \min_{q,q'\in Q} \mathrm{L}_\infty(f_{A_q}, f_{A_{q'}}) \ ,$$

where the minimum is taken over all pairs such that $q \neq q'$. Without loss of generality we shall always assume that $\mu_A > 0$, since otherwise there are two identical states in A which can be merged to obtain a smaller PDFA defining the same distribution; iterating this process as many times as required we always obtain a PDFA with positive L_∞-distinguishability. In most cases we shall just write μ when A is clear from the context. For the rest of this chapter we shall use the simpler term *distinguishability* to refer to this quantity, since we will not consider other distances to compare states in a PDFA.

An important class of PDFA are those defining uniform distributions over parity concepts. Let us fix $\Sigma = \{0, 1\}$ and equip it with addition and multiplication via the identification $\Sigma \simeq \mathbb{Z}/(2\mathbb{Z})$. Given $h \in \Sigma^n$, the *parity function* $P_h : \Sigma^n \to \Sigma$ is defined as $P_h(u) = \bigoplus_{i\in[n]} h_i u_i$, where $\oplus$ denotes addition modulo 2. Each h defines a distribution D_h over Σ^{n+1} whose support depends of P_h as follows: for $u \in \Sigma^n$ and $c \in \Sigma$ let

$$D_h(uc) = \begin{cases} 2^{-n} & \text{if } P_h(u) = c \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

It was shown in [KMRRSS94] that for every $h \in \Sigma^n$ the distribution D_h can be realized by a PDFA with $\Theta(n)$ states. See Figure 1.1 for an illustrative example where each state but the last one has stopping probability 0. It is not hard to see that each of these PDFA has distinguishability $\Theta(2^{-n})$.

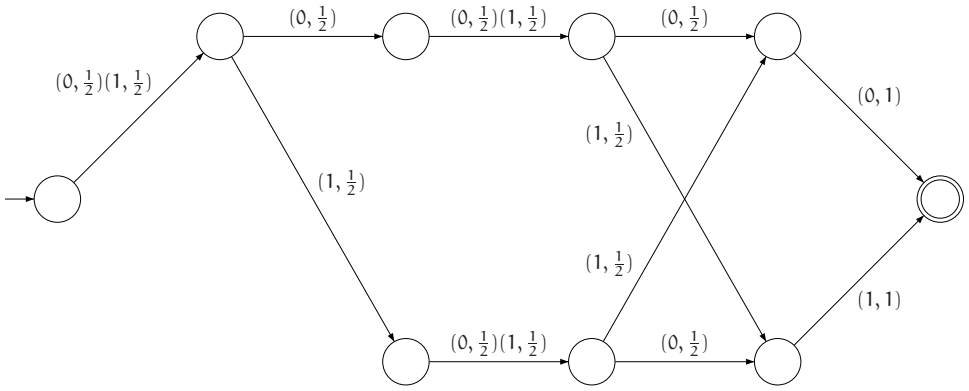


Figure 1.1: PDFA for the distribution D_h with $h = 0101$.

We will need some notation to refer to subclasses of all the distributions that can be realized by PDFA. To start with, we use $\mathcal{PDFA}$ to denote the whole of these distributions. Now we consider several parameters: Σ, n, μ, L, π, and c. Then we define a class of distributions $\mathcal{PDFA}(\Sigma, n, \mu, L, \pi, c)$ that can be realized by some PDFA satisfying the constraints given by these parameters. That is, any $D \in \mathcal{PDFA}(\Sigma, n, \mu, L, \pi, c)$ is $D = f_A$ for some A over Σ such that: $|A| \leq n$, $\mu_A \geq \mu$, $L_A \leq L$, $\pi_A \geq \pi$, and $c_A \geq c$. We may also consider superclasses where only some of the parameters are restricted. For example, $\mathcal{PDFA}(\Sigma, n)$ is the class of distributions over Σ that can be realized by PDFA with at most n states.

1.2 Statistical Query Model for Learning Distributions

The original Statistical Query learning model was introduced in [Kea98]. Algorithms in this framework are given access to an oracle that can provide reasonable approximations to the probabilities of certain events. The model has been used to study trade-offs between computational and information-theoretic limitations of learning algorithms. In particular, the model proved useful as a means of studying resistance against classification noise in the training sample, as well as a means of proving unconditional lower bounds for learning some concept classes.

In this section we introduce the *statistical query* model for learning probability distributions. Our model corresponds to a natural extension of Kearns' original SQ model. Furthermore, in the same way Kearns' model abstracts the behavior of many algorithms in Valiant's well-known PAC framework for learning from labeled examples, our model can be considered an abstraction of the PAC model learning for probability distributions [KMRRSS94].

We begin with a brief recall of the classical SQ framework. In this learning model an algorithm that wants to learn a concept over some set $\mathcal{X}$ can ask queries of the form (χ, α) where χ is (an encoding of) a predicate $\mathcal{X} \times \{0,1\} \to \{0,1\}$ and $0 < \alpha < 1$ is some tolerance. Given a distribution D over $\mathcal{X}$ and a concept $f : \mathcal{X} \to \{0,1\}$, a query $\mathsf{SQ}_f^D(\chi, \alpha)$ to the oracle answers with an α-approximation $\hat{p}_\chi$ of $p_\chi = \mathbb{P}_{x \sim D}[\chi(x, f(x)) = 1]$; that is, $|p_\chi - \hat{p}_\chi| \leq \alpha$. Kearns interprets this oracle as a proxy to the usual PAC oracle EX_f^D that returns pairs $(x, f(x))$ where $x \sim D$ is drawn independently in each successive call. According to him, the oracle SQ_f^D abstracts the fact that learners usually use examples only to measure statistical properties about f under D. The obvious adaptation of statistical queries for learning distributions over $\mathcal{X}$ is to do precisely the same, but without labels.

Let D be a distribution over some set $\mathcal{X}$. A *statistical query for* D is a pair (χ, α) where $\chi : \mathcal{X} \to \{0,1\}$ is (an encoding of) a predicate and $0 < \alpha < 1$ is some *tolerance* parameter. The query $\mathsf{SQ}^D(\chi, \alpha)$ returns an α-approximation $\hat{p}_\chi$ of $p_\chi = \mathbb{P}_{x \sim D}[\chi(x) = 1]$. Since χ is the characteristic function of some subset of $\mathcal{X}$, learners can in principle ask the oracle an approximation to the probability of any event; we will usually identify χ and its characteristic set. However, we will impose a restriction on the possible

characteristic sets that can be queried: we require that $\chi(x)$ can be evaluated efficiently for all $x \in \mathcal{X}$. We will say that an SQ algorithm is γ-*bounded* if all the queries it makes have tolerance $\Omega(\gamma)$. This γ may be a function on the inputs of the algorithm or parameters of the target distribution.

The following result gives a simple simulation of statistical queries using examples drawn i.i.d. from D.

Proposition 1.2.1. *For any distribution D over $\mathcal{X}$, a statistical query $\mathsf{SQ}_f^D(\chi, \alpha)$ can be simulated with error probability smaller than δ using $O(\alpha^{-2}\log(1/\delta))$ examples drawn i.i.d. from D.*

Proof. By Hoeffding, if $\hat{p}_\chi = \hat{\mathbb{E}}_m[\chi(x)]$ with $m \geq (1/2\alpha^2)\ln(2/\delta)$, then one has $|p_\chi - \hat{p}_\chi| \leq \alpha$ with probability at least $1 - \delta$. □

As in Kearns' SQ model, our model can also be used to measure any statistical property of D that can be approximated from random a sample. However, from an algorithmic perspective sometimes these queries fall short on the side of simplicity, in the sense that learning algorithms may need to combine the result of several of queries in order to approximate a relevant parameter or make a correct decision. This has led in the past to the definition of more elaborate and informative queries that can themselves be simulated using statistical queries. For example, the *change of distribution* queries used by [AD98] in their implementation of boosting in the SQ model. Next section introduces a new family of queries that later on will prove useful for implementing state merging methods for learning PDFA.

1.3 The L_∞ Query for Split/Merge Testing

This section introduces a new kind of statistical query called L_∞ query. These queries are only defined in the cases where the domain of the target distribution has a particular structure; that is, when the support of D is contained in a free monoid $X = \Sigma^\star$. In next section it will be shown how state-merging algorithms for learning PDFA can be described rather naturally using such queries. After introducing L_∞ queries, we show how these queries can be simulated either using examples or standard statistical queries.

Let D be a distribution over $\Sigma^\star$ and $A \subseteq \Sigma^\star$ a prefix-free subset. We use D^A to denote the distribution over suffixes conditioned on having a prefix in A:

$$D^A(y) = \mathbb{P}_{x\sim D}[x = zy \mid z \in A] = \frac{D(Ay)}{D(A\Sigma^\star)} \ .$$

An L_∞ query is a tuple of the form (A, B, α, β), where $A, B \subseteq \Sigma^\star$ are (encodings of) disjoint and prefix-free sets, and $0 < \alpha, \beta < 1$ are, respectively, the *tolerance* and *threshold* parameters of the query. Let $\mu = \mathrm{L}_\infty(D^A, D^B)$ denote the supremum distance between distributions D^A and D^B. The oracle DIFF_∞^D answers queries $\mathsf{DIFF}_\infty^D(A, B, \alpha, \beta)$ according to the following rules:

1. If either $D(A\Sigma^\star) < \beta$ or $D(B\Sigma^\star) < \beta$, it answers `'?'`.
2. If both $D(A\Sigma^\star) > 3\beta$ and $D(B\Sigma^\star) > 3\beta$, it answers with some α-approximation $\hat{\mu}$ of μ.
3. Otherwise, the oracle may either answer `'?'` or an α-approximation of μ, arbitrarily.

To be precise, the algorithm asking a query will provide A and B in the form of oracles deciding the membership problems for $A\Sigma^\star$ and $B\Sigma^\star$. We will say that a L_∞ query algorithm is (α, β)-*bounded* if all the queries it makes have tolerance $\Omega(\alpha)$ and threshold $\Omega(\beta)$.

A couple of remarks regarding the definition of L_∞ queries are necessary. The first one is about the role of β that will also be clarified by the simulation given in Proposition 1.3.1. The second observation dispatches computational issues about the encoding of A and B.

Remark 1.1 (The role of β). An algorithm asking an L_∞ query may not know a priori the probability under D of having a prefix in A. It could happen that the region $A\Sigma^\star$ had very low probability, and this might indicate that a good approximation of D in this region is not necessary in order to obtain a good estimate of D. Furthermore, a large number of examples might be required to obtain a good absolute

approximation to probabilities of the form $D^A(x)$. Thus, β allows a query to fail – i.e. return `'?'` – when at least one of the regions being compared has low probability.

Remark 1.2 (Representation of A and B). From now on we will concentrate on the information-theoretic aspects of L_∞ queries. Our focus will be on the number of queries needed to learn certain classes of distributions, as well as the required tolerances and thresholds. We are not concerned with how A and B are encoded or how membership to them is tested from the code: the representation could be a finite automaton, a Turing machine, a hash table, a logical formula, etc. In practice, the only requirement one would impose is that membership testing can be done efficiently.

1.3.1 Simulation from Examples

Similar to what happens with standard Statistical Queries, any L_∞ query can be easily simulated with high probability using examples as shown by the following result.

Proposition 1.3.1. *For any distribution D over $\Sigma^\star$, a call $\mathsf{DIFF}^D_\infty(A, B, \alpha, \beta)$ can be simulated with error probability smaller than δ using $\tilde{O}(\alpha^{-2}\beta^{-2}\log(1/\delta))$ examples drawn i.i.d. from D.*

Proof. The simulation begins by taking a sample S of m_0 i.i.d. examples from D, where

$$m_0 = \max\left\{\frac{48}{\alpha^2\beta}\ln\frac{528}{\alpha^2\beta\delta}, \frac{1}{2\beta^2}\ln\frac{8}{\delta}\right\} \ .$$

Sample S is then used to obtain a sample S^A from D^A as follows. For each word $x \in S$, check whether $x = yz$ with $y \in A$. If this is the case, add z to S^A. The multiset obtained, $S^A = \{z : yz \in S \text{ and } y \in A\}$, is a sample from D^A. Note that since A is prefix-free, each word in S contributes at most one word to S^A, and thus all examples in S^A are mutually independent. Similarly, a sample S^B from D^B is obtained. We let m_A and m_B denote the respective sizes of S^A and S^B and write $\hat{p}_A = m_A/m_0$ and $\hat{p}_B = m_B/m_0$. Note that since $m_0 \geq (1/2\beta^2)\ln(8/\delta)$, by Chernoff $\hat{p}_A$ and $\hat{p}_B$ are β-approximations of $D(A\Sigma^\star)$ and $D(B\Sigma^\star)$ respectively with probability at least $1 - \delta/2$. Based on these approximations, our simulation answers `'?'` if either $\hat{p}_A < 2\beta$ or $\hat{p}_B < 2\beta$. If this is not the case, then we must have $m_A, m_B \geq 2\beta m_0$. In this situation the simulation uses S^A and S^B to compute $\hat{\mu} = \mathrm{L}_\infty(\hat{D}^A, \hat{D}^B)$ and returns this approximation. The probability that $\hat{\mu}$ is not an α-approximation of μ is bounded as follows:

$$\begin{aligned}
&\mathbb{P}[|\hat{\mu}-\mu| > \alpha] \\
&\quad\leq \mathbb{P}\left[\exists x\ |\hat{D}^A(x) - D^A(x)| > \alpha/2 \vee |\hat{D}^B(x) - D^B(x)| > \alpha/2\right] \\
&\quad\leq \mathbb{P}\left[\sup_{x\in\Sigma^\star} |\hat{D}^A(x) - D^A(x)| > \alpha/2\right] + \mathbb{P}\left[\sup_{x\in\Sigma^\star} |\hat{D}^B(x) - D^B(x)| > \alpha/2\right] \\
&\quad\leq 4(2m_A + 1)\exp(-m_A\alpha^2/32) + 4(2m_B + 1)\exp(-m_B\alpha^2/32) \\
&\quad\leq 8(2m_0 + 1)\exp(-m_0\alpha^2\beta/16) \leq \delta/2 \ ,
\end{aligned}$$

where the bounds follow respectively from: the triangle inequality, the union bound, Vapnik–Chervonenkis inequality (A.2), the bounds $2\beta m_0 \leq m_A$ and $m_B \leq m_0$, and $m_0 \geq (48/\alpha^2\beta)\ln(528/\alpha^2\beta\delta)$ together with Lemma A.3.2. □

1.3.2 Simulation from Statistical Queries

Now we show that L_∞ queries can also be simulated using Statistical Queries. To bound the running time of the simulation we will use the following simple isoperimetric inequality giving a geometric relation between surface and volume of connected sets in trees, see e.g. [HLW06]. Think of $\Sigma^\star$ as a tree and let $\Gamma \subseteq \Sigma^\star$ be a finite connected subset. We denote by $\partial\Gamma$ the *boundary* of Γ; that is, the set of all vertices in $\Sigma^\star \setminus \Gamma$ that are adjacent to some vertex in Γ.

Lemma 1.3.2. *Any finite connected $\Gamma \subseteq \Sigma^\star$ satisfies $|\Gamma| \leq |\partial\Gamma|/(|\Sigma| - 1)$.*

Let D be a distribution over $\Sigma^\star$ and $A, B \subseteq \Sigma^\star$ disjoint and prefix-free. Furthermore, let $\alpha, \beta \in (0,1)$. The following theorem shows how to find an approximation of $\mathrm{L}_\infty(D^A, D^B)$ using statistical queries. Its strategy is similar to that of the algorithm used by [KM93] to find the heavy Fourier coefficients of a decision tree using membership queries.

Theorem 1.3.3. *An L_∞ query $\mathsf{DIFF}^D_\infty(A, B, \alpha, \beta)$ can be simulated with $O(|\Sigma|L/\alpha^2)$ calls to SQ^D with tolerance $\Omega(\alpha\beta)$, where L is the maximum of the expected lengths of D^A and D^B.*

Proof. The simulation begins by checking whether $\mathsf{SQ}^D(A\Sigma^\star, \beta) < 2\beta$ or $\mathsf{SQ}^D(B\Sigma^\star, \beta) < 2\beta$. If this is the case, then either having a prefix in A or in B has probability at most 3β under D and the simulation returns `'?'`. Otherwise, the simulation will compute an approximation $\hat{\mu}$ of $\mu = \mathrm{L}_\infty(D^A, D^B)$ such that $|\hat{\mu} - \mu| \leq \alpha$.

Note that if $\mu \leq \alpha$, then $\hat{\mu} = 0$ is a safe approximation. In contrast, if $\mu > \alpha$, then it must be the case that $\mu = |D^A(x) - D^B(x)|$ for some x such that either $D^A(x)$ or $D^B(x)$ is larger than α. Therefore, it is enough to find a set of strings T that contains every x such that either $D^A(x) > \alpha$ or $D^B(x) > \alpha$ and return the maximum of $|D^A(x) - D^B(x)|$ over T. Now we show how to implement this strategy using statistical queries.

Note that hereafter we can assume $D(A\Sigma^\star) \geq \beta$ and $D(B\Sigma^\star) \geq \beta$, and for any string x let us write

$$f(x) = \frac{\mathsf{SQ}^D(Ax\Sigma^\star, \alpha\beta/9)}{\mathsf{SQ}^D(A\Sigma^\star, \alpha\beta/9)} \qquad \text{and} \qquad g(x) = \frac{\mathsf{SQ}^D(Ax, \alpha\beta/9)}{\mathsf{SQ}^D(A\Sigma^\star, \alpha\beta/9)} \ .$$

Note that f and g are not deterministic functions but rather "derived" oracles. By Lemma A.3.1 we know that $f(x)$ and $g(x)$ return $\alpha/3$-approximations of $D^A(x\Sigma^\star)$ and $D^A(x)$ respectively. In the first place, our simulation builds a set T^A as follows. Starting with an empty T^A and beginning with the root λ, recursively explore the tree $\Sigma^\star$, and at each node x do: stop exploring if $f(x) \leq 2\alpha/3$; otherwise, add x to the current T^A if $g(x) > 2\alpha/3$, and recursively explore the nodes $x\sigma$ for each $\sigma \in \Sigma$. Now we prove the following two claims about the output of this procedure: (i) for all $x \in T^A$ we have $D^A(x) > \alpha/3$, and (ii) if $D^A(x) > \alpha$, then $x \in T^A$. We will also bound the size of T^A. For (i) observe that the inclusion condition $g(x) > 2\alpha/3$ implies $D^A(x) > \alpha/3$; this implies $|T^A| < 3/\alpha$. On the other hand, let x be a string with $D^A(x) > \alpha$. For any prefix y of x we have $D^A(y\Sigma^\star) > \alpha$ and $f(y) > 2\alpha/3$. Thus, our stopping condition guarantees that for each $\sigma \in \Sigma$ the nodes $y\sigma$ will be explored. In addition, $D^A(x) > \alpha$ implies $g(x) > 2\alpha/3$. Hence, when x is reached it will be added to T^A; this is claim (ii). Another set T^B for D^B with similar guarantees is built in the same way.

The next step is to show how to use $T = T^A \cup T^B$ to answer the L_∞ query. Let us define a new query

$$h(x) = \left| \frac{\mathsf{SQ}^D(Ax, \alpha\beta/6)}{\mathsf{SQ}^D(A\Sigma^\star, \alpha\beta/6)} - \frac{\mathsf{SQ}^D(Bx, \alpha\beta/6)}{\mathsf{SQ}^D(B\Sigma^\star, \alpha\beta/6)} \right| \ ,$$

and note that, by Lemma A.3.1, a call to $h(x)$ returns an α-approximation to $|D^A(x) - D^B(x)|$. The simulation returns $\hat{\mu} = \max_{x \in T} h(x)$, where $\hat{\mu} = 0$ if T is empty. This can be computed using at most $O(1/\alpha)$ queries with tolerance at least $\Omega(\alpha\beta)$.

Finally, we bound the number of statistical queries used by the simulation, and their tolerances. Let R^A be the set of all vertices of $\Sigma^\star$ explored when constructing T^A; note that R^A is connected. The number of statistical queries used to build T^A is obviously $O(|R^A|)$. Now let $\hat{S}^A$ denote the set of *stopping* vertices of the process that builds T^A:

$$\hat{S}^A = \{x\sigma \mid f(x) > 2\alpha/3 \wedge f(x\sigma) \leq 2\alpha/3\} \ .$$

It is clear that $\hat{S}^A \subseteq R^A$. Furthermore, any vertex in ∂R^A is adjacent to some vertex in $\hat{S}^A$, and each vertex in $\hat{S}^A$ is adjacent to at most $|\Sigma|$ vertices in ∂R^A. Thus, by Lemma 1.3.2, we have $|R^A| \leq |\hat{S}^A||\Sigma|/(|\Sigma| - 1) = O(|\hat{S}^A|)$. We claim that $|\hat{S}^A| = O(|\Sigma|L/\alpha^2)$. To prove the claim, first note that by definition $\hat{S}^A \subseteq S^A$, where

$$S^A = \{x\sigma \mid D^A(x\Sigma^\star) > \alpha/3 \wedge D^A(x\sigma\Sigma^\star) \leq \alpha\} \ .$$

Now consider the following set

$$Q^A = \{y \mid D^A(y\Sigma^\star) > \alpha/3 \wedge \forall \sigma\, D^A(y\sigma\Sigma^\star) \leq \alpha\} \ ,$$

and observe that it is prefix-free; this implies $|Q^A| < 3/\alpha$. Now we establish the following correspondence between S^A and Q^A: each $x \in S^A$ can be uniquely mapped to some $y \in Q^A$, and via this mapping each $y \in Q^A$ is assigned at most $|\Sigma| + (|\Sigma| - 1)|y|$ elements of S^A. Indeed, given $x\sigma \in S^A$ one can see that either $x \in Q^A$ or $xw \in Q^A$ for some suffix w; any ambiguity is resolved by taking the first such w in lexicographical order. Furthermore, given $y \in Q^A$, any element of S^A that is assigned to y by this mapping is of the form $y\sigma$ or $z\sigma$ for some (possibly empty) proper prefix z of $y = z\sigma' w$ with $\sigma \neq \sigma'$. The claim follows from the observation that Markov's inequality implies $|y| \leq 3L/\alpha$ for all $y \in Q^A$. Therefore, to build T we explore at most $O(|\Sigma|L/\alpha^2)$ nodes, making at most $O(|\Sigma|L/\alpha^2)$ statistical queries with tolerance $\Omega(\alpha\beta)$. This concludes the proof. □

The following corollary gives a specialization of the previous simulation for the case when a bound on the length of the words generated by D is known.

Corollary 1.3.4. *If* $\mathrm{supp}(D) \subseteq \Sigma^{\leq n}$*, then a call to* $\mathsf{DIFF}^D_\infty(A, B, \alpha, \beta)$ *can be simulated using* $O(|\Sigma|n/\alpha)$ *calls to* SQ^D *with tolerance* $\Omega(\alpha\beta)$*.*

Proof. Just inspect the previous proof and use the fact that $|x| > n$ implies $D^A(x\Sigma^\star) = 0$ and $D^B(x\Sigma^\star) = 0$. □

1.4 Learning PDFA with Statistical Queries

In this section we describe an algorithm for learning PDFA using statistical and L_∞ queries. We provide a full proof of this result, as well as bounds on the number of queries used by the algorithm and its respective tolerance and threshold parameters. The accuracy between the target PDFA and the hypothesis PDFA produced by the algorithm is measured in terms of L_1, the total variation distance. Two key ideas in the proof are: first, that states in the target which have very low probability of being visited while generating a random string can be safely ignored by a learning algorithm; and second, that having a hypothesis that has small relative error on all short string is enough because the probability assigned by any PDFA to long strings decreases exponentially with their length.

Our algorithm falls into the well-known category of state-merging algorithms. The idea is that at any given time the algorithm maintains a transition graph – which can be regarded as a DFA – with two types of nodes: *safes* and *candidates*. The former and any transitions between them are assumed to be correctly inferred. The latter are recursively explored to decide whether they represent new safes, ways to reach already existing candidates, or are inexistent or infrequent enough to be safely ignored. Information obtained from queries is used in order to make such decisions. Once the transition structure is learned, the DFA is converted into a PDFA by estimating the transition and stopping probabilities corresponding to each state. In this step, a sink state is introduced to represent all non-existing transitions in the inferred DFA. As input the algorithm receives statistical query and L_∞ query oracles, as well as the desired learning accuracy ε and several parameters describing the target distribution: number of states n, alphabet Σ, distinguishability μ, smallest positive stopping probability π, and expected length L. A discussion of why these parameters are needed is deferred to Section 1.6. Full pseudo-code for the algorithm is provided in Algorithm 1.

Algorithm 1: Algorithm for Learning PDFA

Input: n, μ, π, L, Σ, ε, oracles DIFF^D_∞ and SQ^D
Output: A PDFA H

Find the smallest k such that $\mathsf{SQ}^D(\mathbb{1}_{|x|>\lceil 2^k \ln(12/\varepsilon)\rceil}, \varepsilon/24) \leq \varepsilon/8$;
Let $\alpha \leftarrow \mu/2$, $\beta \leftarrow \varepsilon/24n|\Sigma|$, and $\ell \leftarrow \lceil 2^k \ln(12/\varepsilon)\rceil$;
Let $\theta \leftarrow (7\varepsilon\beta/72(|\Sigma|+1)(\ell+1)) \cdot \min\{\beta/(L+1), \pi\}$;

Initialize the graph of H with a safe q_λ and candidates q_σ for $\sigma \in \Sigma$;
while *H contains candidates* **do**
  Choose a candidate q_x maximizing $\mathsf{SQ}^D(H[q_x]\Sigma^\star, \beta)$;
  foreach *safe q* **do**
    Make a call to the oracle $\mathsf{DIFF}^D_\infty(H[q_x]\Sigma^\star, H[q]\Sigma^\star, \alpha, \beta)$;
    if *the answer is* **'?'** **then** remove q_x from the list of candidates and **break**;
    if *the answer $\hat{\mu} < \mu/2$* **then** merge q_x and q and **break**;
  if *q_x is still candidate* **then** promote to safe and add candidates $q_{x\sigma}$ for each $\sigma \in \Sigma$;

Add a *sink* state q_∞ with $\tau_H(q_\infty, \sigma) \leftarrow q_\infty$ and uniform probabilities;

foreach *safe q* **do**
  foreach $\sigma \in \Sigma$ **do**
    let $\hat{\gamma}_{(q,\sigma)} \leftarrow \mathsf{SQ}^D(H[\![q]\!]\sigma\Sigma^\star, \theta)/\mathsf{SQ}^D(H[\![q]\!]\Sigma^\star, \theta)$
  Let $\hat{\varphi}_q \leftarrow \mathsf{SQ}^D(q, \theta)/\mathsf{SQ}^D(q\Sigma^\star, \theta)$;
  if $\hat{\varphi}_q < \pi - 3\theta/\beta$ **then** let $\hat{\varphi}_q \leftarrow 0$;
  Let $N_q \leftarrow \hat{\varphi}_q + \sum_\sigma \hat{\gamma}_{(q,\sigma)}$, $\varphi_H(q) \leftarrow \hat{\varphi}_q/N_q$;
  foreach $\sigma \in \Sigma$ **do**
    if *(q,σ) is defined* **then** $\gamma_H(q,\sigma) \leftarrow \hat{\gamma}_{(q,\sigma)}/N_q$;
    else let $\tau_H(q,\sigma) \leftarrow q_\star$ and $\gamma_H(q,\sigma) \leftarrow \hat{\gamma}_{(q,\sigma)}/N_q$;

Theorem 1.4.1. *If the oracles* DIFF_∞^D *and* SQ^D *given to Algorithm 1 correspond to a distribution* $D \in \mathcal{PDFA}(\Sigma, n, \mu, L, \pi)$*, then the execution of the algorithm:*

1. *runs in time* $\mathsf{poly}(n, |\Sigma|, \log(1/c_D))$*,*
2. *makes* $O(n^2|\Sigma|^2 \log(1/c_D))$ *SQ queries with tolerance* $\tilde{\Omega}(\varepsilon^3 \pi c_D / n^2|\Sigma|^3 L)$*,*
3. *makes* $O(n^2|\Sigma|)$ L_∞ *queries with tolerance* $\Omega(\mu)$ *and threshold* $\Omega(\varepsilon/n|\Sigma|)$*, and*
4. *returns a hypothesis PDFA* H *such that* $\mathrm{L}_1(D, f_H) \leq \varepsilon$*.*

1.4.1 Analysis

The first step in our analysis is identifying sufficient conditions for learning a distribution in terms of the total variation distance. A set of these are given in the following lemma, which basically states that a relative approximation on a massive set is enough.

Lemma 1.4.2. *Let* D *and* D' *be distributions over some set* X*. If there exists a set* $T \subseteq X$ *such that* $D(T) \geq 1 - \varepsilon_1$ *and* $|D(x) - D'(x)| \leq \varepsilon_2 D(x)$ *for every* $x \in T$*, then* $\mathrm{L}_1(D, D') \leq 2(\varepsilon_1 + \varepsilon_2)$*.*

Proof. First note that $\sum_{x \in T} |D(x) - D'(x)| \leq \varepsilon_2 \sum_{x \in T} D(x) \leq \varepsilon_2$. Then, we have $D'(T) \geq 1 - (\varepsilon_1 + \varepsilon_2)$ because

$$D(T) - D'(T) \leq |D(T) - D'(T)| \leq \sum_{x \in T} |D(x) - D'(x)| \leq \varepsilon_2 \ .$$

Finally we see that $\mathrm{L}_1(D, D') \leq \varepsilon_2 + (2\varepsilon_1 + \varepsilon_2)$ since

$$\sum_{x \in \bar{T}} |D(x) - D'(x)| \leq D(\bar{T}) + D'(\bar{T}) \leq \varepsilon_1 + (\varepsilon_1 + \varepsilon_2) \ . \qquad \square$$

Next step is to analyze the structure of the DFA H produced by Algorithm 1. We begin with a technical lemma that will be used in the analysis.

Lemma 1.4.3. *Let* D *be a PDFA with expected length* L_D*. For any DFA* A *we have* $\sum_{q \in Q_A} D(A[\![q]\!]\Sigma^\star) \leq L_D + 1$*. In particular,* $\sum_{q \in Q_{A_D}} D(A_D[\![q]\!]\Sigma^\star) \leq L_D + 1$*.*

Proof. If we define the support of A as the set

$$\mathrm{supp}(A) = \{x \in \Sigma^\star \mid \tau_A(q_0, x) \text{ is defined}\},$$

then we see that the collection of sets $\{A[\![q]\!]\}_{q \in Q_A}$ forms a partition of $\mathrm{supp}(A)$. Furthermore, since for any $x \in \Sigma^\star \setminus \mathrm{supp}(A_D)$ we have $D(x) = 0$, then $D(\Sigma^\star \setminus \mathrm{supp}(A_D)) = D(\mathrm{supp}(A) \setminus \mathrm{supp}(A_D)) = 0$, which in turn implies that

$$\sum_{q \in Q_A} D(A[\![q]\!]) \leq \sum_{q \in Q_{A_D}} D(A_D[\![q]\!]) \ .$$

Now note that for any $x \in \mathrm{supp}(A_D)$ we have $x \in A_D[\![\tau_{A_D}(q_0, u)\ ]\!]\Sigma^\star$ for any $u \sqsubseteq_0 x$. Therefore, any such x will appear at most $|x| + 1$ times in the sum $\sum_{q \in Q_{A_D}} D(A_D[\![q]\!])$. Using the definition of expected length we finally get

$$\sum_{q \in Q_{A_D}} D(A_D[\![q]\!]) \leq \sum_{x \in \Sigma^\star} (|x| + 1) D(x) = L_D + 1 \ . \qquad \square$$

Now let $H' = H \setminus q_\infty$ be the transition graph of H where the sink state q_∞ and all its incoming transitions are removed; that is, H' is the graph obtained right after the while loop ends in Algorithm 1. We will show that this transition structure is identical to a substructure of the target PDFA that contains all states that are traversed with at least some probability when generating a random string.

Lemma 1.4.4. *Under the same hypotheses of Theorem 1.4.1, if* H' *is the DFA defined above, then the following hold:*

1. *H' is isomorphic to a subgraph of A_D,*
2. *$D(\mathrm{supp}(H')) \geq 1 - \varepsilon/6$,*
3. *no transition from A_D with $\gamma_{A_D}(q, \sigma) < \beta/(L_D + 1)$ is present in H', and*
4. *for all states q in H' we have $D(H'[\![q]\!]\Sigma^\star) \geq \beta$.*

Proof. For convenience we write $A = A_D$. We will use H_k to denote the graph built by the algorithm after the kth iteration in the while loop, where H_0 denotes the initial graph. The notation H'_k will be used to denote the subgraph of H_k obtained by removing all candidate nodes and all edges from safe to candidate nodes.

Fact 1: We begin by proving that for all H'_k is a graph isomorphic to a subgraph of A for all $k \geq 0$; we proceed by induction on the number of edges in this graph. Before the first iteration, the graph H'_0 contains a single safe node and no edges, and is clearly isomorphic to the subgraph of A containing only the initial state. At the beginning of the $(k+1)$th iteration the graph H'_k contains at most k edges between safe nodes which, by induction, are assumed to yield a graph isomorphic to a subgraph of A. Let $q_{y\sigma}$ be the candidate chosen during the $(k+1)$th iteration. We must consider three different situations:

1. If the candidate has been discarded, the graph remains unchanged: $H'_{k+1} = H'_k$.
2. If candidate $q_{y\sigma}$ is merged to some safe q, then a new transition from q_y to q labeled with σ is added to the graph; since the $\hat{\mu}$ returned by the oracle certifies $\mathrm{L}_\infty(D^{H_k[q_{y\sigma}]}, D^{H_k[q]}) < \mu$ we conclude that $D^{H_k[q_{y\sigma}]} = D^{H_k[q]}$, thus the new edge must be present in A as well because q_y and q have representatives in A by the inductive hypothesis and $\gamma_A(q_y, \sigma) > 0$ because the L_∞ query did not return `'?'`.
3. If the candidate is promoted to be a new safe node, we known from the oracle's answers that $D(H_k[q_{y\sigma}]\Sigma^\star) > \beta > 0$ and for each safe q already in the graph $\mathrm{L}_\infty(D^{H_k[q_{y\sigma}]}, D^{H_k[q]}) > 0$; thus $q_{y\sigma}$ is a state present in A which is different from the rest of safes and by induction there must be a transition in A from q_y to the new safe labeled by $y\sigma$.

Therefore, the graph of safes H'_{k+1} after the $(k+1)$th iteration is isomorphic to a subgraph of A, and so is H' because it is the graph obtained at the end of the iteration in the while loop that removes the last candidate.

Fact 2: In order to show that $D(\mathrm{supp}(H')) \geq 1 - \varepsilon/6$, we consider two cases. If no candidate is discarded during the while loop, then the graph H' is complete, and thus $D(\mathrm{supp}(H')) = D(\Sigma^\star) = 1$. Now we show that if some candidate is discarded in the while loop, then $D(\mathrm{supp}(H')) \geq 1 - \varepsilon/6$. This will follow from the following claim: for any $k \geq 0$, if $D(\mathrm{supp}(H'_k)) < 1 - \varepsilon/6$ then the candidate q_x chosen during the $(k+1)$th iteration satisfies $D(H_k[q_x]\Sigma^\star) > 3\beta$ and will not be discarded. We prove the claim by induction on k. For $k = 0$ we have $D(\mathrm{supp}(H'_0)) = D(\lambda) < 1 - \varepsilon/6$, and since $D(\lambda) + \sum_\sigma D(q\Sigma^\star) = 1$, we must have $D(\sigma\Sigma^\star) > \varepsilon/6|\Sigma| \geq 4\beta$ for some σ. Therefore, the candidate selection rule will select a candidate q_σ with $D(H_0[q_\sigma]\Sigma^\star) = D(\sigma\Sigma^\star) > 3\beta$ during the first iteration, which will not be discarded because $D(H_0[q_\lambda]\Sigma^\star) = 1 > 3\beta$. For the general case, suppose the claim is true for $k' < k$ and $D(\mathrm{supp}(H'_k)) < 1 - \varepsilon/6$. Since we have $D(\mathrm{supp}(H'_{k'})) \leq D(\mathrm{supp}(H'_k))$ because $\mathrm{supp}(H'_{k'}) \subseteq \mathrm{supp}(H'_k)$, by induction no candidate has been discarded so far and every safe q in H'_k satisfies $D(H_k[q]\Sigma^\star) > 3\beta$. Then, taking into account that there will be at most $n|\Sigma|$ candidates in the graph, the same argument used in the base case shows that the candidate q_x chosen in the $(k+1)$th iteration will satisfy $D(H_k[q_x]\Sigma^\star) > 3\beta$ and will not be discarded.

Fact 3: Suppose that $q_{y\sigma}$ is the candidate considered in the kth iteration of the while loop. We know that safe q_y must correspond to some state in A. Then we have that

$$D(H_k[q_{y\sigma}]\Sigma^\star) = D(H_k[q_y]\ \Sigma^\star)\gamma_A(q_y, \sigma) \leq D(H_k[\![q_y]\!]\Sigma^\star)\gamma_A(q_y, \sigma)\ ,$$

and note that Lemma 1.4.3 implies $D(H_k[\![q_y]\!]\Sigma^\star) \leq (L_D + 1)$. Therefore, if $\gamma_D(q_y, \sigma) < \beta/(L_D + 1)$, then $D(H_k[q_{y\sigma}]\Sigma^\star) < \beta$, the candidate will be discarded, and transition (q_y, σ) will no be present in H'.

Fact 4: It is obvious that any $q \neq q_\lambda$ in H' must have been a candidate which was considered at some iteration k and was not discarded. Therefore, it must be the case that $D(H'[q]\Sigma^\star) \geq D(H_k[q]\Sigma^\star)\beta$. □

From now on we will identify H' with the corresponding subgraph of A_D. Next result gives some information about the parameter ℓ computed by our algorithm. In particular, how many queries are needed to compute it and what is the probability under D of strings longer than ℓ.

Lemma 1.4.5. *ℓ is such that $D(\Sigma^{>\ell}) \leq \varepsilon/6$, the number of queries needed to find ℓ is $O(\log(1/c_D))$, and ℓ is at most $O(c_D^{-1} \ln 1/\varepsilon)$.*

Proof. By definition we have $D(\Sigma^{>\ell}) \leq \varepsilon/8 + \varepsilon/24 = \varepsilon/6$. Because D is a PDFA, Lemma 1.1.1 implies that for all i we have

$$D(\Sigma^{>\lceil 2^i \ln(12/\varepsilon)\rceil}) \leq \exp(-c_D \lceil 2^i \ln(12/\varepsilon)\rceil) \leq (\varepsilon/12)^{c_D 2^i} \ .$$

The number k of queries needed to compute ℓ is upper bounded by the largest i such that $D(\Sigma^{>\lceil 2^i \ln(1/\varepsilon)\rceil}) > \varepsilon/12$, because for all such i the oracle can return $\mathsf{SQ}^D(\mathbb{1}_{|x|>\lceil 2^k \ln(1/\varepsilon)\rceil}, \varepsilon/24) = D(\Sigma^{>\lceil 2^i \ln(1/\varepsilon)\rceil}) + \varepsilon/24 > \varepsilon/8$. From equation above we deduce that $k = O(\log(1/c_D))$, which implies $\ell = O(c_D^{-1} \ln 1/\varepsilon)$. □

Next two lemmas show that the probability assigned to short strings under H has small relative error with respect to D. This will allow us to apply Lemma 1.4.2 to proof the desired approximation bound.

Lemma 1.4.6. *Let $A = A_D$. For every transition $(q,\sigma) \in H'$ we have $|\gamma_A(q,\sigma) - \gamma_H(q,\sigma)| \leq 7\varepsilon\gamma_A(q,\sigma)/8(\ell+1)$. Furthermore, for all $q \in H'$ we have $|\varphi_A(q) - \varphi_H(q)| \leq 7\varepsilon\varphi_A(q)/8(\ell+1)$, and in particular if $\varphi_A(q) = 0$ then $\varphi_H(q) = 0$.*

Proof. First note that for all $q \in H'$ and all $\sigma \in \Sigma$ we have $|\gamma_A(q,\sigma) - \hat{\gamma}_{(q,\sigma)}| \leq 3\theta/\beta$ by Lemmas A.3.1 and 1.4.4. For stopping probabilities $|\varphi_A(q) - \hat{\varphi}_q| \leq 3\theta/\beta$ as well; note that $\hat{\varphi}_q$ is set to zero if and only if $\varphi_A(q) < \pi$, which by definition of π implies $\varphi_A(q) = 0$. The bounds derived so far imply that $|N_q - 1| \leq (|\Sigma|+1)3\theta/\beta$ holds for each $q \in H'$. Now suppose $(q,\sigma) \in H'$. By Lemma 1.4.4 we know that $\gamma_A(q,\sigma) \geq \beta/(L_D+1)$. Therefore, Lemma A.3.1 yields $|\gamma_A(q,\sigma) - \gamma_H(q,\sigma)| \leq 9(|\Sigma|+1)(L_D+1)\theta/\beta^2 \leq 7\varepsilon\gamma_A(q,\sigma)/8(\ell+1)$. In addition, if $\hat{\varphi}_q > 0$ then $\varphi_A(q) \geq \pi$ and we have $|\varphi_A(q) - \varphi_H(q)| \leq 7\varepsilon\varphi_A(q)/8(\ell+1)$ as well. □

Lemma 1.4.7. *For any $x \in \Sigma^{\leq\ell} \cap \mathrm{supp}(H')$ one has $|1 - f_H(x)/D(x)| \leq \varepsilon/6$*

Proof. Let $x \in \Sigma^{\leq\ell} \cap \mathrm{supp}(H')$ and $A = A_D$. By Lemma 1.4.6 we have $|\ln(f_H(x)/f_A(x))| \leq (\ell+1)\ln(1 + 7\varepsilon/8(\ell+1)) \leq 7\varepsilon/8$ since the computation of $f_H(x)$ uses only transitions in H' and $\ln(1+x) \leq x$ for $x \geq 0$. Therefore, Lemma A.3.3 with $c = 1/3$ implies $|1 - f_H(x)/f_A(x)| \leq \varepsilon/6$. □

Now we can finally combine all the results above in order to prove the main theorem of this section.

Proof of Theorem 1.4.1. Let $S = \mathrm{supp}(H')$ and $T = S \cap \Sigma^{\leq\ell}$. Note that $\bar{T} \subseteq \bar{S} \cup \Sigma^{>\ell}$. By Lemmas 1.4.4 and 1.4.5 we have $D(T) \geq 1 - D(\bar{S}) - D(\Sigma^{>\ell}) \geq 1 - \varepsilon/3$. Thus, by Lemmas 1.4.7 and 1.4.2 we get $\mathrm{L}_1(D, f_H) \leq 2(\varepsilon/3 + \varepsilon/6) = \varepsilon$. The rest of the bounds follow trivially by inspection and Lemma 1.4.5. □

1.5 A Lower Bound in Terms of Distinguishability

In this section we prove a lower bound on the number of L_∞ queries that any (α,β)-bounded algorithm learning the class of all PDFA must make. In order to do so, we show such a lower bound for a class of distributions defined using parities that can be modeled by PDFA. Our lower bound is unconditional, and applies to proper and improper learning algorithms that try to learn either with respect to relative entropy (KL) or total variation distance (L_1). From the main result we deduce another lower bound in terms of the distinguishability that describes a trade-off between the number of queries and the tolerance and threshold parameters.

In this section we fix $\Sigma = \{0,1\}$ and let $\mathcal{D}_n$ denote the class of all distributions D_h constructed using parity codes $h \in \Sigma^n$ as defined in Section 1.1. We already know that every distribution in $\mathcal{D}_n$ has support in Σ^{n+1} and can be represented with a PDFA of $\Theta(n)$ states.

Kearns [Kea98] proved that learning parity *concepts* is a difficult problem in the statistical query model. More precisely, he showed the following.

Theorem 1.5.1 ([Kea98]). *Any γ-bounded statistical query algorithm that learns the class of all parity concepts must make at least $\Omega(2^n\gamma^2)$ queries.*

It is not difficult to translate this result into a lower bound for algorithms that try to learn the class $\mathcal{D}_n$ *as distributions* using our version of statistical queries. On the other hand, if one is interested in learning the class $\mathcal{D}_n$ using a state-merging algorithm it is enough to learn the structure of the target parity, since by definition all transition probabilities are uniform. In principle this can be done with an algorithm using *only* L_∞ queries, by a simple adaptation of Algorithm 1. For this task, a reduction argument combining Kearns's lower bound with the simulation given by Corollary 1.3.4 yields the following lower bound.

Corollary 1.5.2. *Any (α,β)-bounded L_∞ query algorithm that learns the class $\mathcal{D}_n$ with accuracy $\varepsilon \leq 1/9$ must make at least $\Omega(2^n\alpha^3\beta^2/n)$ queries.*

In this section we will prove an improvement on this bound using a direct proof. In particular, we are able to show the following lower bound, which improves the dependence on n and α in the previous one.

Theorem 1.5.3. *Any (α,β)-bounded L_∞ query algorithm that learns the class $\mathcal{D}_n$ with accuracy $\varepsilon \leq 1/9$ must make at least $\Omega(2^n\alpha^2\beta^2)$ queries.*

An immediate consequence of this theorem is that the same lower bound applies to algorithms that learn the class of all PDFA.

Corollary 1.5.4. *Any (α,β)-bounded L_∞ query algorithm that learns the class $\mathcal{PDFA}(n)$ with accuracy $\varepsilon \leq 1/9$ must make at least $\Omega(2^n\alpha^2\beta^2)$ queries.*

Theorem 1.5.3 is proved by devising an adversary that answers in a malicious way each query asked by the learning algorithm. The argument is similar in nature to that used in [Kea98] to prove that parities cannot be learned in the Statistical Query model. Basically, we use a probabilistic argument to show that the number of distributions in $\mathcal{D}_n$ that are inconsistent with each answer given by the adversary is at most $O(1/(\alpha^2\beta^2))$. Since there are 2^n distributions in $\mathcal{D}_n$, any learning algorithm is unable to distinguish the correct target with $o(2^n\alpha^2\beta^2)$ queries. Since all distributions in $\mathcal{D}_n$ are far apart from each other, the adversary can always find a distribution which is consistent with every answer given to the learner but still has large error with respect to the hypothesis provided by the learner. The details of the proof are deferred to the end of this section. Now we discuss some corollaries of Theorem 1.5.3.

An interesting consequence of our lower bound in terms of α and β is that, by a simple padding argument, it allows us to derive a lower bound in terms of the distinguishability. The dependence on ε in the statement is ignored.

Corollary 1.5.5. *Suppose $2\log(1/\mu)+3 \leq n \leq (1/\mu)^{o(1)}$. Let $a,b,c \geq 0$ be constants and take $\alpha = \mu^a \cdot n^{O(1)}$ and $\beta = \mu^b \cdot n^{O(1)}$. If an (α,β)-bounded L_∞ query algorithm learns the class $\mathcal{PDFA}(n,\mu)$ using at most $n^{O(1)}/\mu^c$ queries, then necessarily $2a+2b+c \geq 1$.*

Proof. Recall the class of distributions $\mathcal{D}_k$ from Theorem 1.5.3. For every positive integers m and k, define the class of distributions $\mathcal{C}_{m,k}$ as follows: for every distribution D in $\mathcal{D}_k$, there is a distribution in $\mathcal{C}_{m,k}$ that gives probability $D(x)$ to each string of the form $0^m x$, and 0 to strings not of this form. Every distribution in $\mathcal{D}_k$ is generated by a PDFA with at most $2k+2$ states and distinguishability 2^{1-k}. It follows that every distribution in $\mathcal{C}_{m,k}$ is generated by a PDFA with at most $m+2k+2$ states and distinguishability 2^{-k}. Assuming without loss of generality that $k = \log(1/\mu)$ is an integer, for $m = n-2k-2$ we have $\mathcal{C}_{m,k} \subseteq \mathcal{PDFA}(n,\mu)$.

Now note that, by an immediate reduction, any (α,β)-bounded L_∞ query algorithm that learns $\mathcal{P}_{n,\mu}$ using at most $n^{O(1)}/\mu^c$ queries can learn the class $\mathcal{D}_k$ with the same number of queries. Therefore, Theorem 1.5.3 implies that necessarily $n^{O(1)}/\mu^c = \Omega(\alpha^2\beta^2/\mu)$. Substituting for α and β, and using that $1/n = \mu^{o(1)}$, the bound yields

$$\frac{1}{\mu^{c+o(1)}} = \Omega\left(\frac{\mu^{2a+2b}}{\mu^{1+o(1)}}\right) \quad .$$

Therefore, since $1/\mu = \omega(1)$, we conclude that $2a+2b+c \geq 1$. □

Next corollary collects the extreme cases of previous one. These results illustrate the trade-off that algorithms for learning PDFA must face: either ask a lot of queries with moderate tolerance and threshold, or ask a few queries with very small values of tolerance and threshold.

Corollary 1.5.6. *Suppose* $2\log(1/\mu) + 3 \leq n \leq (1/\mu)^{o(1)}$. *Then the following hold:*

1. *If* $\alpha, \beta = \mu^{o(1)} \cdot n^{O(1)}$, *then any* (α, β)*-bounded* L_∞ *query algorithm that learns* $\mathcal{PDFA}(n, \mu)$ *must make at least* $\Omega(1/\mu^{1-\nu})$ *queries for any constant* $\nu > 0$.
2. *If an* (α, β)*-bounded* L_∞ *query algorithm learns* $\mathcal{PDFA}(n, \mu)$ *using* $n^{O(1)}$ *queries, then necessarily* $\alpha^2\beta^2 = O(\mu \cdot n^{O(1)})$.

Proof. Both statements follow by inspection from the previous proof. □

1.5.1 Technical Lemmata

Before getting to the proof of Theorem 1.5.3 we must go over a chain of lemmas. We begin with two simple bounds which are stated without proof.

Lemma 1.5.7. *Let* X *be a real random variable and* $\tau, \gamma \in \mathbb{R}$. *The following hold:*

1. *if* $\tau > |\gamma|$, *then* $\mathbb{P}[|X| > \tau] \leq \mathbb{P}[|X - \gamma| > \tau - |\gamma|]$,
2. $\mathbb{P}[|X| < \tau] \leq P[|X - \gamma| > \gamma - \tau]$.

The following technical lemma bounds the probability that the maximum of a finite set of random variables deviates from some quantity in terms of the individual variances. Let $\{X_i\}_{i\in[m]}$ be a finite collection of random variables with $\gamma_i = \mathbb{E}[X_i]$ and $\gamma_1 \geq \ldots \geq \gamma_m$. Let $Z = \max_{i\in[m]} |X_i|$ and $\gamma = \max_{i\in[m]} |\gamma_i|$.

Lemma 1.5.8. *If* $\gamma = \gamma_1$, *then for all* $t > 0$

$$\mathbb{P}[|Z - \gamma| > t] \leq \frac{\mathbb{V}[X_1]}{t^2} + \sum_{i\in[m]} \frac{\mathbb{V}[X_i]}{t^2} \ .$$

Proof. On the first place, note that by (1) in Lemma 1.5.7 and the definition of γ we have, for any $i \in [m]$, $\mathbb{P}[|X_i| > \gamma + t] \leq \mathbb{P}[|X_i - \gamma_i| > t]$. Thus, by the union bound and Chebyshev's inequality,

$$\mathbb{P}[Z > \gamma + t] \leq \sum_{i\in[m]} \frac{\mathbb{V}[X_i]}{t^2} \ . \tag{1.1}$$

On the other side, for any $i \in [m]$ we have $\mathbb{P}[Z < \gamma - t] \leq \mathbb{P}[|X_i| < \gamma - t] \leq \mathbb{P}[|X_i - \gamma_i| > \gamma_i - \gamma + t]$, where the last inequality follows from (2) in Lemma 1.5.7. Choosing $i = 1$ we obtain

$$\mathbb{P}[Z < \gamma - t] \leq \frac{\mathbb{V}[X_1]}{t^2} \ . \tag{1.2}$$

The result follows from (1.1) and (1.2). □

For any $A \subseteq \Sigma^*$, $c \in \Sigma$ and $h \in \Sigma^n$, we write $A_n = A \cap \Sigma^n$ and $A_h^c = \{x \in A_n \mid P_h(x) = c\}$. Let $A, B \subseteq \Sigma^{1:n} = \Sigma^{\leq n} \cap \Sigma^+$ be disjoint and prefix-free. Our next lemma shows that, for any $D_h \in \mathcal{D}_n$, one can write $\mathrm{L}_\infty(D_h^A, D_h^B)$ as the maximum of $2n$ easily computable quantities. Let $p_A = D_h(A\Sigma^*)$ and $p_B = D_h(B\Sigma^*)$, and note that we have $p_A = \sum_{k=1}^n |A_k|/2^k$ and $p_B = \sum_{k=1}^n |B_k|/2^k$; that is, these probabilities are independent of h. Now define the following quantities:

$$X_k^c = \frac{|A_{h_{1:k}}^c|}{p_A} - \frac{|B_{h_{1:k}}^c|}{p_B} \ . \tag{1.3}$$

These quantities can be used to easily compute the supremum distance between D^A and D^B.

Lemma 1.5.9. *With the above notation, the following holds:*

$$\mathrm{L}_\infty(D_h^A, D_h^B) = \max_{k\in[n],c\in\Sigma} |X_k^c|/\,2^n \ .$$

Proof. First we write, for $1 \le k \le n$, $Y_k = \max_{y\in\Sigma^{n+1-k}} |D_h^A(y) - D_h^B(y)|$, and note that $\mathrm{L}_\infty(D_h^A, D_h^B) = \max_{1\le k\le n} Y_k$. Now we show that, for $k \in [n]$, $Y_k = \max\{|X_k^0|, |X_k^1|\}/2^n$. First observe that for $y \in \Sigma^{n-k}$ and $z \in \Sigma$ we have $D_h^A(yz) = \sum_{x\in A\cap\Sigma^k} D_h(xyz)/p_A$. Furthermore, $D_h(xyz) = 2^{-n}$ if and only if $P_{h_{1:k}}(x) \oplus P_{h_{k+1:n}}(y) = z$. Thus, for fixed y and z, we have

$$\sum_{x\in A\cap\Sigma^k} D_h(xyz) = \begin{cases} |A^0_{h_{1:k}}|/2^n & \text{if } P_{h_{k+1:n}}(y) = z \ , \\ |A^1_{h_{1:k}}|/2^n & \text{if } P_{h_{k+1:n}}(y) \neq z \ . \end{cases}$$

This concludes the proof. □

Now we will consider the quantities X_k^c as random variables when $h \in \Sigma^n$ is taken uniformly at random. We are interested in the expectation and variance of these quantities.

Lemma 1.5.10. *When h is taken uniformly at random, the following hold:*

1. $\mathbb{E}[X_k^c] = |A_k|/(2p_A) - |B_k|/(2p_B) + O(1)$, *and*
2. $\mathbb{V}[X_k^c] = O(|A_k|/p_A^2 + |B_k|/p_B^2 + (|A_k| + |B_k|)/(p_A p_B))$.

Proof. To begin note that taking $g \in \Sigma^k$ uniformly at random, the random variables $|A^c_{h_{1:k}}|$ and $|A^c_g|$ follow the same distribution. Thus, we can use g instead of $h_{1:k}$ in the definition of X_k^c.

We start by computing $\mathbb{E}[|A_g^c|]$, $\mathbb{E}[|A_g^c|^2]$ and $\mathbb{E}[|A_g^c||B_g^c|]$. Using indicator variables one has

$$\mathbb{E}[|A_g^c|] = \sum_{x\in A_k} \mathbb{P}[P_g(x) = c] = \frac{|A_k'|}{2} + \mathbb{1}_{\bar 0\in A_k\wedge c=0} = \frac{|A_k|}{2} + O(1) \ , \tag{1.4}$$

where $A_k' = A_k \setminus \{\bar 0\}$. The second term models the fact that $P_g(\bar 0) = 0$ for all g. This implies (1). To compute $\mathbb{E}[|A_g^c|^2]$ we will use that for different $x, y \in \Sigma^k \setminus \{\bar 0\}$ a standard linear algebra argument shows $\mathbb{P}[P_g(x) = c \wedge P_g(y) = c] = 1/4$. Furthermore, note that if either $x = \bar 0$ or $y = \bar 0$, but not both, then $\mathbb{P}[P_g(x) = c \wedge P_g(y) = c] = 1/2$ if $c = 0$, and it is zero if $c = 1$. Hence, we have

$$\begin{aligned} &\sum_{x\in A_k} \sum_{y\in A_k\setminus\{x\}} \mathbb{P}[P_g(x) = c \wedge P_g(y) = c] \\ &\qquad = \frac{|A_k'|^2}{4} - \frac{|A_k'|}{4} + \mathbb{1}_{\bar 0\in A_k\wedge c=0}(|A_k| - 1) \ . \end{aligned} \tag{1.5}$$

Now, combining (1.4) and (1.5) we obtain

$$\begin{aligned} \mathbb{E}[|A_g^c|^2] &= \sum_{x\in A_k} \sum_{y\in A_k\setminus\{x\}} \mathbb{P}[P_g(x) = c \wedge P_g(y) = c] + \sum_{x\in A_k} \mathbb{P}[P_g(x) = c] \\ &= \frac{|A_k|^2}{4} + O(|A_k|) \ . \end{aligned} \tag{1.6}$$

Furthermore, since $\mathbb{1}_{\bar 0\in A_k}\mathbb{1}_{\bar 0\in B_k} = 0$ because A and B are disjoint, similar arguments as before yield

$$\begin{aligned} \mathbb{E}[|A_g^c||B_g^c|] &= \sum_{x\in A_k} \sum_{y\in B_k} \mathbb{P}[P_g(x) = c \wedge P_g(y) = c] \\ &= \frac{|A_k'||B_k'|}{4} + \mathbb{1}_{c=0}\left(\mathbb{1}_{\bar 0\in B_k}\frac{|A_k|}{2} + \mathbb{1}_{\bar 0\in A_k}\frac{|B_k|}{2}\right) \\ &= \frac{|A_k||B_k|}{4} + O(|A_k| + |B_k|) \ . \end{aligned} \tag{1.7}$$

Finally, (1.4), (1.6) and (1.7) can be used to compute $\mathbb{V}[X_g^c]$. The bound in (2) follows from the observation that all terms having $|A_k|^2$, $|B_k|^2$ or $|A_k||B_k|$ are cancelled. □

1.5.2 Proof of the Lower Bound

Finally, we combine the preceding lemmas to prove Theorem 1.5.3.

Proof of Theorem 1.5.3. Suppose $\mathtt{L}$ is an (α, β)-bounded L_∞ query algorithm that learns the class $\mathcal{D}_n$. Now we describe and analyze an adversary answering any L_∞ query asked by $\mathtt{L}$.

Let $\mathsf{DIFF}_\infty^D(A, B, \alpha, \beta)$ be a query asked by $\mathtt{L}$ where we assume without loss of generality that it uses the smallest tolerance and threshold values allowed by the bounding restriction. Suppose that $h \in \Sigma^n$ is taken uniformly at random and let $p_A = \sum_{k=1}^n |A_k|/2^k$, $p_B = \sum_{k=1}^n |B_k|/2^k$, $\gamma_k^c = \mathbb{E}[X_k^c]/2^n$, where we use the quantities defined in (1.3). The adversary uses the expressions given by Lemma 1.5.10 to compute these quantities in terms of A and B. It then answers as follows: if either $p_A \leq 2\beta$ or $p_B \leq 2\beta$, then answer `'?'`; otherwise, answer $\hat{\mu} = \max_{k\in[n],c\in\Sigma} |\gamma_k^c|$. We claim that in each case the number of distributions in $\mathcal{D}_n$ inconsistent with these answers is at most $O(1/(\alpha^2\beta^2))$. If the adversary answers `'?'`, then every distribution in $\mathcal{D}_n$ is consistent with this answer because p_A and p_B are independent of h. In the other case, we use a probabilistic argument to prove the bound.

Suppose the adversary answers $\hat{\mu} = \max_{k\in[n],c\in\Sigma} |\gamma_k^c|$, and note that, interchanging A and B if necessary, we can assume $\hat{\mu} = \max_{k\in[n],c\in\Sigma} \gamma_k^c$. In this case, the number of inconsistent distributions is $2^n \,\mathbb{P}[|\,\mathrm{L}_\infty(D_h^A, D_h^B) - \hat{\mu}| \geq \alpha]$. We can combine Lemmas 1.5.8 and 1.5.9 to get

$$\mathbb{P}[|\,\mathrm{L}_\infty(D_h^A, D_h^B) - \hat{\mu}| \geq \alpha] \leq \frac{1}{2^{2n}\alpha^2}\left(\mathbb{V}[X_M] + \sum_{k,c}\mathbb{V}[X_k^c]\right) \ ,$$

where $X_M \in \{X_1^0, \ldots, X_n^1\}$ is such that $\hat{\mu} = \mathbb{E}[X_M]/2^n$. Since A and B are disjoint and thus $\sum_{k=1}^n (|A_k| + |B_k|) \leq 2^n$, using the bound (2) from Lemma 1.5.10 we obtain

$$\mathbb{V}[X_M] + \sum_{k,c}\mathbb{V}[X_k^c] = O\left(\frac{2^n}{p_A^2} + \frac{2^n}{p_B^2} + \frac{2^n}{p_A p_B}\right) = O\left(\frac{2^n}{\beta^2}\right) \ ,$$

were last inequality uses that $p_A, p_B > 2\beta$. Therefore, we can conclude that the number of distributions inconsistent with the answer $\hat{\mu}$ is at most $O(1/(\alpha^2\beta^2))$.

Let I denote the maximum number of distributions inconsistent with each query issued by $\mathtt{L}$ and Q the number of queries it makes. Now we show that if $I \cdot Q \leq 2^n - 2$, the algorithm necessarily produces a hypothesis with large error.

Note that the relative entropy between any two distributions in $\mathcal{D}_n$ is infinite because they have different supports. Since $I \cdot Q \leq 2^n - 2$ implies that there are at least two distributions in $\mathcal{D}_n$ consistent with every answer given by the adversary, if $\mathtt{L}$ outputs a hypothesis in $\mathcal{D}_n$, the adversary can still choose a consistent distribution with infinite error with respect to the hypothesis produced by $\mathtt{L}$. Recalling that for each pair of distributions in $\mathcal{D}_n$ we have $\mathrm{L}_1(D_f, D_g) = 1$, we also get a lower bound for learning $\mathcal{D}_n$ with respect to the total variation distance. Furthermore, the following argument shows that $\mathtt{L}$ still has large error even if its output is not restricted to a distribution in $\mathcal{D}_n$.

Assume $\mathtt{L}$ outputs some distribution $\hat{D}$, not necessarily in $\mathcal{D}_n$, such that $\mathrm{KL}(D_h\|\hat{D}) \leq \varepsilon$ for some $D_h \in \mathcal{D}_n$. Then it follows from Pinsker's inequality [CBL06] that $\mathrm{KL}(D_g\|\hat{D}) \geq (1/2\ln 2)(1 - \sqrt{2\ln 2\varepsilon})^2$ for any other distribution D_g different from D_h. Since $\varepsilon \leq 1/9$, we then have $\mathrm{KL}(D_g\|\hat{D}) > 2/9$. Therefore, if $I \cdot Q \leq 2^n - 2$ the hypothesis produced by the learner will have large error. We can conclude that if $\mathtt{L}$ learns $\mathcal{D}_n$ with Q queries, then necessarily $Q > (2^n - 2)/I = \Omega(2^n\alpha^2\beta^2)$. □

1.6 Comments

We devote this section to make some remarks about the meaning of the results presented in this chapter.

1.6.1 Consequences of SQ Learnability

We begin by discussing the implications of showing that state-merging algorithms can be cast in a statistical query framework. It is well known that statistical query algorithms for learning concepts can

be easily converted to PAC algorithms capable of learning from noisy examples. In the case where the goal is to learn a probability distribution, noise takes the form of outliers. The following result shows that state-merging algorithms can learn PDFA even in the presence of a small fraction of outliers.

Let D be a distribution over some set $\mathcal{X}$. For $0 < \eta < 1$, an *η-perturbation* of D is a distribution over $\mathcal{X}$ of the form $D_\eta = (1-\eta)D + \eta\tilde{D}$, where $\tilde{D}$ is some distribution of outliers over $\mathcal{X}$. We say that D can be *learned under a proportion of outliers η* if there exists an algorithm that can learn D when it is only given access to some η-perturbation of D.

Theorem 1.6.1. *Suppose D can be learned by an algorithm that makes statistical queries with tolerance at least γ, and let $\eta < \gamma/2$. Then D can be learned under a proportion of outliers η using a $(\gamma - 2\eta)$-bounded SQ algorithm.*

Proof. In the algorithm that learns D we will replace every statistical query with another query with a smaller tolerance to obtain an algorithm that learns D under a proportion of outliers η. We can assume without loss of generality that all queries in the original algorithm are of the form $\mathsf{SQ}^D(\chi, \gamma)$. In the new algorithm each of these queries is replaced by a query of the form $\mathsf{SQ}^{D_\eta}(\chi, \gamma - 2\eta)$. We only need to show that any answer $\hat{p}_\chi$ to any of the new queries is a γ-approximation to $p_\chi = D(A_\chi)$, where $A_\chi = \chi^{-1}(1)$. Indeed, since by definition $|\hat{p}_\chi - D_\eta(A_\chi)| \leq \gamma - 2\eta$, we have

$$\begin{aligned}
|\hat{p}_\chi - p_\chi| &\leq |\hat{p}_\chi - D_\eta(A_\chi)| + |D_\eta(A_\chi) - D(A_\chi)| \\
&\leq (\gamma - 2\eta) + |(1-\eta)D(A_\chi) + \eta\tilde{D}(A_\chi) - D(A_\chi)| \\
&\leq (\gamma - 2\eta) + \eta(D(A_\chi) + \tilde{D}(A_\chi)) \leq \gamma \ .
\end{aligned}$$

□

Combining the learning result of Theorem 1.4.1 with the simulation of Theorem 1.3.3 we can show that PDFA can be learned under some proportion of outliers.

Corollary 1.6.2. *Let $D \in \mathcal{PDFA}(\Sigma, n, \mu, L, \pi)$ and $\gamma = \min\{\mu, \varepsilon^2\pi c_D/n|\Sigma|^2 L\}$ for some $\varepsilon > 0$. Then D can be learned with accuracy ε with respect to total variation distance under a proportion of outliers $O(\varepsilon\gamma/n|\Sigma|)$ using statistical queries.*

The same arguments we have just used above can be given a different, equally interesting interpretation. Fix some $\eta > 0$. If $\mathcal{D}$ is some class of probability distributions, we let $\mathcal{D}_\eta$ denote the class of all distributions that are at L_1 distance at most η of some distribution in $\mathcal{D}$:

$$\mathcal{D}_\eta = \{D \ \mid \ \mathrm{L}_1(D, D') \leq \eta \text{ for some } D' \in \mathcal{D}\} \ .$$

Theorem 1.6.3. *Suppose that every distribution in a class $\mathcal{D}$ can be learned with accuracy ε with respect to L_1 distance using statistical queries of tolerance at least γ. Then, for any $\eta < \gamma$, the class $\mathcal{D}_\eta$ can be learned with accuracy $\varepsilon + \eta$ with respect to L_1 distance using statistical queries of tolerance at least $\gamma - \eta$.*

Proof. Suppose we want to learn a distribution $D \in \mathcal{D}_\eta$ given access to oracle SQ^D. We run the algorithm for learning $\mathcal{D}$ up to accuracy ε and replace every query $\mathsf{SQ}(\chi, \gamma)$ by a query of the form $\mathsf{SQ}^D(\chi, \gamma - \eta)$. Denote by $\hat{D}$ the output of this algorithm. Then the same argument used above shows that $\mathrm{L}_1(\hat{D}, D') \leq \varepsilon$ for some $D' \in \mathcal{D}$ such that $\mathrm{L}_1(D, D') \leq \eta$, and the result follows from the triangle inequality. □

Therefore, it turns out that our state-merging algorithm can learn distributions that are close to being generated by PDFA.

Corollary 1.6.4. *For any $\varepsilon > 0$, the class of distributions $\mathcal{PDFA}_\eta(\Sigma, n, \mu, L, \pi, c)$ for some $\eta = O(\varepsilon\gamma/n|\Sigma|)$ can be learned with accuracy $O(\varepsilon(1 + \gamma/n|\Sigma|))$ with respect to L_1 distance using statistical queries, where $\gamma = \min\{\mu, \varepsilon^2\pi c/n|\Sigma|^2 L\}$.*

In conclusion, when a class of distributions can be learned using statistical queries, then distributions in that class can also be learned in the presence of outliers, and distributions that are "almost" in the class can be learned using the same learning algorithm. Of course, these results combined with the simulation in Proposition 1.2.1 imply that similar results hold when learning algorithms are provided with a large enough sample from the target distribution (plus maybe some outliers). It seems that in the particular case of PDFA these results were not previously known.

1.6.2 Input Parameters to Algorithm 1

We will now devote a few lines to discuss the input parameters required by our algorithm for learning PDFA using statistical queries. Recall that these are: desired accuracy ε, alphabet Σ, number of states n, distinguishability μ, expected length L, and smallest non-zero stopping probability π. Of these, ε can be arbitrarily chosen by the user. The rest are assumed to correctly represent the target distribution. That is, the oracles SQ^D and DIFF^D_∞ provided to the algorithm correspond to a distribution $D \in \mathcal{PDFA}(n, \Sigma, \mu, L, \pi)$.

Parameters n, Σ, and μ are usually required by most algorithms in the literature for PAC learning PDFA, the only exception being [CG08], where μ appears in the sample bounds but is not an input to their algorithm. Of course, Σ can in principle be confidently estimated given a large enough sample; in the case of SQ algorithms it needs to be given as input because the algorithm can not see examples drawn from D. Our lower bounds show that μ is indispensable, though it may be possible to devise a cross-validation-like scheme to infer a lower bound on the true μ. Something of this sort will be done in Chapter 3 in a different context. Designing such strategy in the SQ framework is left as an open problem. This same reasoning can, in principle, be applied to the number of states n.

Asking for the expected length L of D is very convenient for two reasons. First, estimating L using only probabilities of events may be possible but not really practical. If the SQ framework is understood as a proxy to information extracted from samples, then it is reasonable to ask as input something that can be easily estimated from a sample but that can be cumbersome to estimate using only statistical queries. Using that the length of strings generated by a PDFA is sub-exponential (Lemma 1.1.1) and the concentration inequality for sub-exponential random variables given in Lemma A.1.5, it is not difficult to give a bound on the number of examples needed to obtain an accurate estimate of L. The second reason is that using L in the learning proof greatly simplifies some of the bounds. Palmer and Goldberg in [PG07] show that PDFA can be learned in terms of total variation distance without the need to know L. Theirs is a technically interesting result, but given that L can be easily estimated from a sample and its use greatly simplifies the learning bounds, we have decided to use it for the sake of simplicity. It is worth mentioning here that it was shown in [CT04a] that a bound on the expected length of strings generated from *any* state in a PDFA is required for PAC learning PDFA in terms of relative entropy. Though unfortunately they also call their parameter L, our L is just the expected length of strings generated from the *initial* state.

The role of (a lower bound to) the smallest non-zero stopping probability π may appear less clear at first sight. This parameter has not been used in any previous analysis, and may in principle be difficult to estimate. However, it turns out that allows for a simpler analysis which stresses the thesis of Lemma 1.4.2: a relative approximation in a large enough set is enough for learning in terms of total variation distance. The need for π as an input can be relaxed at the expense of a slightly more complex algorithm and an increase in the length of our learning proof. Since this results only in a technical improvement bearing no new insights into the problem, we have opted for using π in our presentation. If the algorithm is presented in the course of a lecture, the problem of removing π from the input can be left as homework.

1.6.3 Meaning and Tightness of the Bounds

The main message of this chapter is that all state-merging algorithms in the literature for PAC learning PDFA can be interpreted as SQ algorithms. Together with the lower bounds in Section 1.5 we see that, despite the observation in [KMRRSS94] that PDFA contain the hard to learn class of noisy parities, it is parities themselves that are hard to learn with state-merging methods. That is, state-merging methods have a great difficulty in learning PDFA that contain deep highly symmetric structures; precisely those which yield exponentially small distinguishabilities.

On the other hand, the results are somewhat reassuring, in the sense that μ provides a rather fine-grained measure of complexity for the whole class of PDFA. Examples with very small μ seem to be pathological cases which are hard to learn – and remember that such cases must exist due to the lower bounds – or simply cases where some other method different from state-merging might do better. In contrast, using state-merging methods in cases where the distinguishability is large seems the best option,

and yields a fast, statistically efficient method, which has the advantage of producing as output a PDFA guaranteed to contain the relevant substructures from the target. It will become clear in the second part of this dissertation that proper learning of complex probability distributions is not always possible. This means that sometimes it is easier to find a function $f : \Sigma^\star \to \mathbb{R}$ that approximates the target distribution when we do not impose that f is given by a probabilistic automaton. However, whenever it is possible, proper learning is always a desirable property.

Of course, it is not difficult to construct examples with exponentially small distinguishability which are easy to learn using other methods based on statistical queries other than state-merging. For every $h \in \{-1, 0, +1\}^n$ define the *conjunction* $C_h : \{0,1\} \to \{0,1\}$ as $C_h(x) = \bigwedge_{i\in[n]} x_i^{h_i}$, with the conventions $x_i^0 = 1$, $0^{-1} = 1$ and $1^{-1} = 0$. We construct the class of distributions $\mathcal{C}_n$ following the same schema used for defining $\mathcal{D}_n$, but using conjunctions instead of parities. Note that each $D_h \in \mathcal{C}_n$ can be represented with a PDFA of $\Theta(n)$ states and distinguishability $\Theta(1/2^n)$. However, it is well known that the class of all conjunctions over n variables can be efficiently learned using statistical queries, and a trivial reduction shows that $\mathcal{C}_n$ can also be learned using $\mathsf{poly}(n)$ statistical queries with tolerance $\mathsf{poly}(1/n)$. That is, our lower bound of $\Omega(1/\mu^{1-c})$ for every $c > 0$ applies when trying to learn the class $\mathcal{PDFA}(n, \mu)$ of all PDFA with n states and distinguishability μ, but not when trying to learn an arbitrary subclass of $\mathcal{PDFA}(n, \mu)$.

The conclusion is that μ is a measure of complexity well suited to measure hardness of learning with state-merging methods, but it does not characterize hardness of learning in general. In this sense μ resembles the VC dimension used in statistical learning, though with some reserves. For example, VC dimension characterizes the statistical complexity of learning with any algorithm, while μ measures the computational complexity of finding the underlying DFA of a PDFA by greedily distinguishing states. We note here that this problem is known to be easy in statistical terms, since it was shown in [AW92] that a sample of polynomial size suffices to determine the best transition structure with a fixed number of states. Ideally, however, one would like a measure of complexity, say $\tilde{\mu}$, that provably characterizes the complexity of learning any subclass of $\mathcal{PDFA}$. Namely, if $\mathcal{D} \subseteq \mathcal{PDFA}$ is any large enough subclass, we would be interested in a result saying that any algorithm that learns $\mathcal{D}$ uses at least $\Omega(\tilde{\mu}_\mathcal{D})$ examples (or queries), where $\tilde{\mu}_\mathcal{D}$ measures the complexity of the class $\mathcal{D}$. A possible line of work in this direction would be adapt one of the several notions of *dimension* used in concept learning to distribution learning. Furthermore, finding learning algorithms that provably match this hypothetical lower bound in terms of $\tilde{\mu}$ would be another interesting, practically relevant problem. A (small) step in this direction will be described in the following chapter with the introduction of prefix-distinguishability.

At a more quantitative level, we want to remark that there is still a gap between our upper and lower bounds on the complexity of learning PDFA with statistical queries. In particular, we can compare Theorem 1.4.1 with Corollary 1.5.6. Note that our algorithm is $(\mu, 1/n)$-bounded – we ignore the dependence on $|\Sigma|$ and ε here – and makes $O(n^2)$ L_∞ queries. According to the lower bound it must have $\alpha^2\beta^2 = O(\mu n^{O(1)})$ but it turns out that our algorithm has $\alpha^2\beta^2 = O(\mu^2 n^{O(1)})$ which is smaller by a factor of μ. It is an open question whether this gap is an artifact in our lower bound or whether more efficient algorithms in terms of μ can be designed for learning PDFA.

Chapter 2

Implementations of Similarity Oracles

One of the key observations from last chapter is that being able to compare two distributions over $\Sigma^\star$ and determine whether they are equal or not is the cornerstone of state-merging algorithms. Though we showed how to simulate L_∞ queries using samples in Proposition 1.3.1, broadly speaking last chapter's approach was mostly axiomatic with respect to the testing procedure, in the sense that its existence was assumed and given in the form of an oracle. In this chapter we dwell upon the particular problem of testing similarity between two distributions D and D' given i.i.d. samples S and S' from both distributions.

We begin with a general description of the problem in the context of property testing. Instead of making comparisons based on the L_∞ distance we used in previous chapter, we introduce a less coarse distance measure denoted $\mathrm{L}_\infty^{\mathrm{p}}$ which in many cases will lead to more sample-efficient tests. The problem of testing similarity between distributions is reduced in Section 2.1 to that of constructing confidence intervals for the true distance $\mathrm{L}_\infty^{\mathrm{p}}(D, D')$.

The rest of the chapter gives constructions of such confidence intervals and provides an experimental evaluation of the tests obtained using different constructions. In particular, we provide a simple construction based on uniform convergence bounds. After observing an asymmetry in this construction which makes decisions on equality harder to make, we propose a test based on bootstrapped confidence intervals that improves on this particular aspect. We finish the chapter with constructions of tests that work only with summaries of the samples S and S' instead of the samples themselves. This yields tests whose memory requirements drops from $O(1/\mu^2)$ to $O(1/\mu)$ and that can process samples on-line. These tests will be useful in next chapter when we design state-merging algorithms for the data stream scenario.

2.1 Testing Similarity with Confidence Intervals

The main problem that we face in this chapter is to decide whether two distributions D and D' are equal or not based on observing a sample drawn from each distribution. We will begin this first section by fixing some notation that will be used throughout the rest of the chapter. Then we introduce a notion of distance between probability distributions over $\Sigma^\star$ which is less coarse than the L_∞ used in the previous chapter. In general this will allow us to design tests that in some cases can make accurate decisions with smaller samples. Finally we will describe a general strategy for constructing the desired tests based on confidence intervals. The rest of the chapter will then be devoted to different methods for obtaining such confidence intervals.

Suppose D and D' are two arbitrary distributions over $\Sigma^\star$. Let S be sample of m i.i.d. examples drawn from D and S' a sample of m' i.i.d. examples drawn from D'. For any event $A \subseteq \Sigma^\star$ we will use $S(A)$ to denote the *empirical probability of A under sample S*, which should in principle be an approximation to the probability $D(A)$. More specifically, if for any $x \in \Sigma^\star$ we let $S[x]$ denote the number of times that x

appears in $S = (x^1, \ldots, x^m)$, then

$$S(A) = \frac{1}{m} \sum_{x \in A} S[x] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{x_i \in A} \ .$$

The sizes of S and S' will come into play in the statement of several results. The two following related quantities will be useful:

$$M = m + m', \qquad \text{and} \qquad M' = \frac{mm'}{(\sqrt{m} + \sqrt{m'})^2} \ .$$

In order to make quantitative statements about the distinguishing power of a certain statistical test we need a measure of distance between probability distributions. Instead of the supremum distance L_∞ used in Chapter 1, here we will use the so-called *prefix*-L_∞ *distance* which we denote by $\mathrm{L}_\infty^{\mathrm{p}}$. It is defined as follows:

$$\mathrm{L}_\infty^{\mathrm{p}}(D, D') = \sup_{x \in \Sigma^\star} |D(x\Sigma^\star) - D'(x\Sigma^\star)| \ .$$

It is shown in Appendix A.5 that $\mathrm{L}_\infty^{\mathrm{p}}$ is indeed a metric. Proposition A.5.1 in that same section shows that in general $\mathrm{L}_\infty(D, D') \leq (2|\Sigma|+1)\, \mathrm{L}_\infty^{\mathrm{p}}(D, D')$. This shows that using $\mathrm{L}_\infty^{\mathrm{p}}$ instead of L_∞ as a measure of distinguishability one may, in the worst case, lose a constant factor in the sample bounds that will be given in the rest of the chapter. Furthermore, we also provide in Appendix A.5 a class of distributions generated by PDFA where L_∞ is exponentially smaller that $\mathrm{L}_\infty^{\mathrm{p}}$. Altogether, these two bounds justify the use of $\mathrm{L}_\infty^{\mathrm{p}}$ as a metric for designing similarity tests on distributions over $\Sigma^\star$. This particular measure was used in [RST98] for learning acyclic PDFA and then in [CG08] for learning general PDFA.

The goal of our tests will be to decide whether S and S' come from the same distribution – $D = D'$ – or from different distributions. This problem can be arbitrarily difficult if D and D' are allowed to be different but arbitrarily close to each other. In order to formalize this intuition, one can parametrize the complexity of the problem in terms of the smallest distance between D and D' that the test must be able to distinguish. A natural framework for such formalization which is used frequently in property testing is the notion of a *promise problem* [Gol06].

Let $\mu > 0$ be a *distinguishability* parameter. Then given S and S' a $\mathrm{L}_\infty^{\mathrm{p}}$ *similarity test* has to decide whether $D = D'$ or $D \neq D'$ under the promise that if $D \neq D'$ then $\mathrm{L}_\infty^{\mathrm{p}}(D, D') \geq \mu$. Introducing the promise on μ makes the design and analysis of correct testing algorithms much easier. In practice one assumes that the algorithm can answer arbitrarily when the promise is not satisfied. Sometimes it is also useful to introduce a third possible output representing the fact that the test is unable to decide which is the case given the current samples. This may happen, for example, when S or S' are too small. Of course, it may also happen sometimes that by chance the samples S and S' are misleading. Thus, testing algorithms usually accept a confidence parameter δ and certify that their answer (if any) is correct with probability at least $1 - \delta$ over the sampling procedure and any internal randomization in the test.

However, this setup requires a bit of caution. Asking for an algorithm that does not fail very often and that is allowed to answer "I don't know" may result in a very conservative test that does not make a decision until it is absolutely certain about the correct answer. Though perfectly correct, such a test might not be very useful for practical purposes, where one is interested in deciding confidently *as soon as possible*. In the literature on statistical hypothesis testing this leads to the notion of the *power* of a test. Informally, we say that a test is more powerful than another if the first test is always able to make a correct decision using less examples than the second test. In general, bounding the power of a test may be very difficult, and even when it is possible the bounds may not be tight enough for practical purposes. Since power comparisons are usually motivated by the need to choose a particular test for practical applications, a set of experiments providing such comparisons in an empirical setting will be presented in Section 2.6. In most cases these experiments are much more informative than theoretical bounds. That is because in tasks where *uniformly most powerful tests* do not exist, some tests may be better than others depending on the distributions D and D'. And this can be observed better through experimentation.

The main tool we will use to build similarity tests are confidence intervals. Suppose $\mu_\star = \mathrm{L}_\infty^{\mathrm{p}}(D, D')$ and let $0 < \delta < 1$ be a confidence parameter. An *upper confidence limit* for $\mu_\star$ at confidence level

δ computed from S and S' is a number $\hat{\mu}_U = \hat{\mu}_U(S, S', \delta)$ which for any two distributions D and D' satisfies $\mu_\star \leq \hat{\mu}_U$ with probability at least $1 - \delta$. Similarly, we define a *lower confidence limit* $\hat{\mu}_L$ satisfying $\hat{\mu}_L \leq \mu_\star$ with the same guarantees. Using these two quantities one can define a *confidence interval* $[\hat{\mu}_L, \hat{\mu}_U]$ which will contain $\mu_\star$ with probability at least $1 - 2\delta$.

Given a confidence interval $[\hat{\mu}_L, \hat{\mu}_U]$ for $\mu_\star$ and a distinguishability parameter μ it is simple to construct a similarity test as follows. If $\hat{\mu}_U < \mu$ then decide that $D = D'$ since we know that (with high probability) $\mu_\star < \mu$ which, by the promise on μ, implies $\mu_\star = 0$. If $\hat{\mu}_L > 0$ then decide that $D \neq D'$ since we then know that (with high probability) $\mu_\star > 0$. If none of the conditions above hold, then answer "I don't know".

The problem of similarity testing has thus been reduced to that of obtaining confidence intervals for $\mu_\star$ with a desired level of confidence. We will give several constructions of those in the rest of the chapter.

2.2 Confidence Intervals from Uniform Convergence

In this section we give confidence limits for empirical estiamtions of $\mathrm{L}^{\mathrm{p}}_\infty(D, D')$ using Vapnik–Chervonenkis bound on the uniform rate of convergence of empirical probabilities to their expectations over a fixed (possibly infinite) set of events. From our perspective, VC bounds provide a combinatorial framework for applying a union bound argument to bound the probability that none of an infinite family of events occur, provided that the family of interest satisfies certain properties. See Appendix A.1 for more details and Appendix A.5 for relevant calculations regarding shattering coefficients. VC theory has other numerous applications in statistical learning theory, see [Vap99] for example. Bounds in the same spirit as the ones given here, but proved using different techniques, have appeared in the context of PDFA learning in the works [RST98; CT04a; PG07; CG08].

Let D and D' be two distributions over $\Sigma^\star$ with $\mu_\star = \mathrm{L}^{\mathrm{p}}_\infty(D, D')$. Suppose we have access to two i.i.d. samples S and S' drawn from D and D' respectively, where $m = |S|$ and $m' = |S'|$. We will use the empirical estimate $\hat{\mu} = \mathrm{L}^{\mathrm{p}}_\infty(S, S')$ to determine whether D and D' are equal or different. In particular, we will give a confidence interval for $\mu_\star$ centered around $\hat{\mu}$. For any $0 < \delta < 1$ define

$$\Delta(\delta) = \sqrt{\frac{8}{M'} \ln\left(\frac{16M}{\delta}\right)} \ . \tag{2.1}$$

Then we can prove the following two propositions giving confidence limits for $\mu_\star$ of the form $\hat{\mu} \pm \Delta(\delta)$.

Proposition 2.2.1. *With probability at least $1 - \delta$ we have $\mu_\star \leq \hat{\mu} + \Delta(\delta)$.*

Proof. Let us write $\Delta = \Delta(\delta)$ for some fixed δ. The result follows from a standard application of the Vapnik–Chervonenkis inequality showing that $\mathbb{P}[\hat{\mu} < \mu_\star - \Delta] \leq \delta$. First note that by the triangle inequality we have $\mathrm{L}^{\mathrm{p}}_\infty(S, S') \geq \mathrm{L}^{\mathrm{p}}_\infty(D, D') - \mathrm{L}^{\mathrm{p}}_\infty(D, S) - \mathrm{L}^{\mathrm{p}}_\infty(D', S')$. Therefore, $\hat{\mu} < \mu_\star - \Delta$ implies $\mathrm{L}^{\mathrm{p}}_\infty(D, S) + \mathrm{L}^{\mathrm{p}}_\infty(D', S') > \Delta$. Now for any $0 < \gamma < 1$ we have

$$\begin{aligned}
\mathbb{P}[\hat{\mu} < \mu_\star - \Delta] &\leq \mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(D, S) + \mathrm{L}^{\mathrm{p}}_\infty(D', S') > \Delta] \\
&\leq \mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(D, S) > \gamma\Delta] + \mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(D', S') > (1-\gamma)\Delta] \\
&\leq 16me^{-m\gamma^2\Delta^2/8} + 16m'e^{-m'(1-\gamma)^2\Delta^2/8} \\
&= 16Me^{-M'\Delta^2/8} = \delta \ ,
\end{aligned}$$

where we choose γ such that $m\gamma^2 = m'(1-\gamma)^2$. □

Proposition 2.2.2. *With probability at least $1 - \delta$ we have $\mu_\star \geq \hat{\mu} - \Delta(\delta)$.*

Proof. The argument is very similar to the one used in Proposition 2.2.1. Write $\Delta = \Delta(\delta)$ for some fixed δ as well. We need to see that $\mathbb{P}[\mu_\star < \hat{\mu} - \Delta] \leq \delta$. Since $\mathrm{L}^{\mathrm{p}}_\infty(S, S') \leq \mathrm{L}^{\mathrm{p}}_\infty(D, D') + \mathrm{L}^{\mathrm{p}}_\infty(D, S) + \mathrm{L}^{\mathrm{p}}_\infty(D, D')$, then $\mu_\star < \hat{\mu} - \Delta$ implies $\mathrm{L}^{\mathrm{p}}_\infty(D, S) + \mathrm{L}^{\mathrm{p}}_\infty(D, D') > \Delta$. Thus, the conclusion follows from the same bound we used before. □

The conclusion is that $[\hat{\mu}-\Delta(\delta/2), \hat{\mu}+\Delta(\delta/2)]$ is a confidence interval for $\mu_\star$ at level δ. It is interesting to note that $\hat{\mu}$ depends on the contents of S and S', but the size of the interval $2\Delta(\delta/2)$ only depends on the sizes m and m'. When $m = m'$, the size of this interval behaves like $O(\sqrt{\ln(m)/m})$, which is known to be optimal up to the $O(\sqrt{\ln(m)})$ factor. Bounds with optimal asymptotic behavior can be obtained using different techniques, e.g. chaining and covering numbers [DL01]. However such bounds are rarely of practical use for testing algorithms because the asymptotic advantage is only observed on astronomical sample sizes due to the constants involved.

Propositions 2.2.1 and 2.2.2 can also be used to quantify how many examples are needed at most to make a decision with a test based on these bounds. Let $m = m'$ again. Assume first that $\mu_\star = 0$. Then with probability at least $1 - \delta/2$ we will have $\hat{\mu} \leq \Delta(\delta/2)$. Therefore, $\hat{\mu}_U = \hat{\mu} + \Delta(\delta/2) \leq 2\Delta(\delta/2)$. Thus, with high probability a decision will be made when $2\Delta(\delta/2) < \mu$ or sooner, which implies $m = \tilde{O}(1/\mu^2)$. On the other hand, assume that $\mu_\star \geq \mu$. Similar arguments show that with high probability $\hat{\mu}_L = \hat{\mu} - \Delta(\delta/2) \geq \mu_\star - 2\Delta(\delta/2)$. Therefore, a decision will be made at the latest when $2\Delta(\delta/2) < \mu_\star$, which implies $m = \tilde{O}(1/\mu_\star^2)$.

These bounds expose a particular feature of the test which is often encountered in practice. Namely, that deciding similarity is usually harder than deciding dissimilarity. This is because in the former case one is always playing against the worst case $\mu_\star = \mu$, while in the latter larger values for $\mu_\star$ require smaller sample sizes. In some sense, the test is adaptive to the true (unknown) value of $\mu_\star$ when it comes to deciding dissimilarity. This will motivate the alternative upper confidence limit presented in next section. The key idea is to obtain an alternative to $\hat{\mu} + \Delta$ with a greater dependence on the *contents* of S and S', with the hope that this will help us make similarity decisions with smaller samples when enough evidence of similarity is available.

2.3 Bootstrapped Confidence Intervals

In this section we describe an alternative upper confidence bound for $\mu_\star$. This will be based on the use of a common technique in computational statistics: the bootstrap. The basic idea is to take a sample S from some distribution D and simulate the process of generating new examples from D by sampling with replacement from S. Via this process new *bootstrapped* samples are obtained, and these are then used to obtain statistical information about a certain empirical estimate. This technique was introduced by Efron [Efr79] and is very popular in non-parametric statistics. The asymptotic theory of the bootstrap has been well studied in the statistical literature, see e.g. Hall's book [Hal92]. However, general finite sample analysis of Efron's bootstrap seem to be inexistent.

We begin by introducing some notation and describing Efron's construction of bootstrapped upper confidence limits. This is basically the approach one would use in practice. Here, however, we deviate a little from the classic approach and propose a modification to the method for which we can given strict formal justifications and finite sample analyses. Both methods are compared in the experiments of Section 2.6.

In the bootstrap setting, given an i.i.d. sample S from D with $m = |S|$, the first step is to construct several bootstrapped samples by resampling S uniformly with replacement. In particular, we let $B_1, \ldots, B_r$ be independent resamples from S, each of size m. The same is done for a sample S' of size m' from D', yielding resamples $B'_1, \ldots, B'_r$ of size m'. From these resamples we construct r bootstrapped estimates for $\mu_\star$: let $\hat{\mu}_i = \mathrm{L}^{\mathrm{p}}_\infty(B_i, B'_i)$ for $1 \leq i \leq r$.

For the purpose of testing it is useful to consider these estimates sorted increasingly. Thus we will write ω to denote a permutation of $[r]$ satisfying $\hat{\mu}_{\omega(1)} \leq \cdots \leq \hat{\mu}_{\omega(r)}$. Efron's idea is then to use $\hat{\mu}_{\omega(\lceil(1-\delta)r\rceil)}$ as an upper confidence limit for $\mu_\star$ at level δ. This is based on the observation that if B_i and B'_i were independent samples from D and D' respectively, the estimates $\hat{\mu}_i$ would be a sample from the distribution of the estimator $\hat{\mu} = \mathrm{L}^{\mathrm{p}}_\infty(S, S')$. In which case, enough estimates $\hat{\mu}_i$ would yield an accurate histogram of this particular distribution, from which confidence upper limits for $\mu_\star$ can be obtained because the estimator $\hat{\mu}$ is asymptotically unbiased; that is, $\mathbb{E}_{S,S'}[\hat{\mu}] \to \mu_\star$ when m and m' grow to infinity.

Of course, the justification we have just given for the bootstrap is an asymptotic one. But since we are interested in provably correct tests that work with finite samples, we need to quantify the error

probability of such tests for given sample sizes. In the following we show that it is possible to conduct a finite sample analysis of an alternative test which is also based on the bootstrap estimates $\hat{\mu}_i$. The test relies on upper confidence limit for $\mu_\star$ constructed using bootstrap estimates. The main technical tool is Theorem 2.3.1, whose direct consequence is Corollary 2.3.2 which gives the desired upper confidence limit.

The setup is as we just described. Let $0 < \delta < 1$ be a confidence parameter and $1 \le k \le r$ the index of an estimate in the series $\hat{\mu}_{\omega(1)} \le \cdots \le \hat{\mu}_{\omega(r)}$. Let us define the following two quantities:

$$\Theta_k(\delta) = \max_{0<x<\delta-e^{-2k^2/r}} \min\left\{x, \frac{k}{r} - \sqrt{\frac{1}{2r}\ln\left(\frac{1}{\delta - x}\right)}\right\} \ ,$$

$$\Delta_k(\delta) = \sqrt{\frac{32}{M'}\ln\left(\frac{16M}{\Theta_k(\delta)}\right)} \ .$$

Theorem 2.3.1. *Let $\sqrt{(r/2)\ln(1/\delta)} < k \le r$. With probability at least $1-\delta$ we have $\mu_\star \le \hat{\mu}_{\omega(k)} + \Delta_k(\delta)$.*

The proof is given in Section 2.3.1. The following corollary follows from a simple union bound.

Corollary 2.3.2. *With probability at least $1-\delta$ it holds that*

$$\mu_\star \le \min\left\{\hat{\mu}_{\omega(k)} + \Delta_k(\delta/k) \;\middle|\; \sqrt{(r/2)\ln(r/\delta)} < k \le r\right\} \ .$$

This corollary provides an alternative with finite-sample guarnatees to Efron's upper bound $\mu_\star \le \hat{\mu}_{\omega(\lceil(1-\delta)r\rceil)}$.

2.3.1 Proof of Theorem 2.3.1

Write $\Delta = \Delta_k(\delta)$ and for $1 \le i \le r$ define an indicator random variable $Y_i = \mathbb{1}_{\hat{\mu}_i \le \mu_\star - \Delta}$. It is clear that if $\hat{\mu}_{\omega(k)} \le \mu_\star - \Delta$, then $Z = Y_1 + \cdots + Y_r \ge k$.

The key observation for analyzing the bootstrap test described in this section is that once the sample S is given the resamples $B_1, \ldots, B_r$ are mutually independent and identically distributed. Therefore, conditioned on S and S', the estimates $\hat{\mu}_1, \ldots, \hat{\mu}_r$ are independent and Z is a sum of i.i.d. random variables.

The first step in the proof is to analyze the expectation of these random variables given samples satisfying a particular property. We will say that an i.i.d. sample S from D is t-good if $\mathrm{L}^{\mathrm{p}}_\infty(D, S) \le t$. Then we write $\mathfrak{G}(t, t')$ to denote the event "S is t-good and S' is t'-good".

Lemma 2.3.3. *For any $1 \le i \le r$ and $t, t' > 0$ such that $t + t' < \Delta$ it holds that*

$$\mathbb{E}\left[Y_i \,|\, \mathfrak{G}(t, t')\right] \le 16Me^{-M'(\Delta - t - t')^2/8} \ .$$

Proof. Since all Y_i are i.i.d. given S and S', we consider only the case $i = 1$. Let $B = B_1$, $B' = B'_1$, and $Y = Y_1$. First we observe that events $\mathfrak{G}(t, t')$ and $\hat{\mu}_i \le \mu_\star - \Delta$ imply that necessarily $\mathrm{L}^{\mathrm{p}}_\infty(S, B) + \mathrm{L}^{\mathrm{p}}_\infty(S', B') \ge \Delta - t - t'$. Considering the samples S and S' fixed and applying the Vapnik–Chervonenkis inequality to the resampling processes from which B and B' are obtained we can see that, for any $0 < \gamma < 1$,

$$\begin{aligned}\mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S, B) + \mathrm{L}^{\mathrm{p}}_\infty(S', B') \ge \Delta - t - t'] \\ &\le \mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S, B) \ge \gamma\Delta - t] + \mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S', B') \ge (1-\gamma)\Delta - t'] \\ &\le 16me^{-m(\gamma\Delta - t)^2/8} + 16me^{-m'((1-\gamma)\Delta - t')^2/8} \ .\end{aligned}$$

Thus, choosing γ such that $m(\gamma\Delta - t)^2 = m'((1-\gamma)\Delta - t')^2$ yields

$$\mathbb{E}\left[Y_i \,|\, \mathfrak{G}(t, t')\right] = \mathbb{P}\left[\hat{\mu}_i \le \mu_\star - \Delta \,|\, \mathfrak{G}(t, t')\right] \le 16Me^{-M'(\Delta - t - t')^2/8} \ . \qquad \square$$

Proof of Theorem 2.3.1. For any $t > 0$ let $t' = t\sqrt{m/m'}$. We denote by $\bar{\mathfrak{G}}(t,t')$ the complementary event of $\mathfrak{G}(t,t')$: "S is not t-good or S' is not t'-good". Then, by the Vapnik–Chervonenkis inequality,

$$\mathbb{P}[\bar{\mathfrak{G}}(t,t')] \leq 16me^{-mt^2/8} + 16m'e^{m't'^2/8} = 16Me^{-mt^2/8} \ ,$$

Now let $0 < x_\star < \delta - e^{-2k^2/r}$ be the solution to equation $x_\star = k/r - \sqrt{(1/2r)\ln(1/(\delta - x_\star))}$, which maximizes the expression in $\Theta_k(\delta)$. Taking $t = \sqrt{(8/m)\ln(16M/x_\star)}$ we get $\mathbb{P}[\bar{\mathfrak{G}}(t,t')] \leq x_\star$.

On the other hand, writing $k = \eta r$, Hoeffding's inequality combined with Lemma 2.3.3 yields

$$\mathbb{P}[Z \geq \eta r \mid \mathfrak{G}(t,t')] \leq e^{-2r(\eta - 16me^{-M(\Delta - t - t')^2/8})^2} \ .$$

Now note that our choice for t' implies $t + t' = t\sqrt{m/M'}$. Together with our choice for t this implies that

$$\Delta - (t + t') = \Delta/2 = \sqrt{\frac{8}{M'}\ln\left(\frac{16M}{\eta - \sqrt{\frac{1}{2r}\ln\left(\frac{1}{\delta - x_\star}\right)}}\right)} \ .$$

Thus, we see that $\mathbb{P}[Z \geq \eta r \mid \mathfrak{G}(t,t')] \leq \delta - x_\star$, which yields

$$\mathbb{P}[Z \geq k] \leq \mathbb{P}[\bar{\mathfrak{G}}(t,t')] + \mathbb{P}[Z \geq k \mid \mathfrak{G}(t,t')] \leq \delta \ . \qquad \square$$

2.4 Confidence Intervals from Sketched Samples

A common drawback to all tests described so far is their need to store the whole sample in memory in order to compute some statistic. Since in the case $D = D'$ samples of size $\Omega(1/\mu^2)$ are needed before a decision can be made confidently, this implies a lower bound of the same order on the amount of memory used by such tests. This rises the question of whether using this much memory is absolutely necessary for confidently testing similarity. And the answer is no. It turns out that tests using just $O(1/\mu)$ memory can be designed. The key to such constructions is using statistics that can be computed from *sketches* of the sample instead of the sample itself.

Sketches are basically data structures that receive items in the sample one at a time and process them sequentially. After processing the whole sample, a sketch keeps a summary that contains only its most relevant features. This summary can hen be used to efficiently compute some statistics of interest. Interestingly, the sketch can process every item in the sample in constant time and the memory used by the sketch only depends on the accuracy required to compute the desired statistics, i.e. it is independent of the sample size. Concrete implementations of sketches will be discussed in the next chapter. In the present section we shall adopt an axiomatic approach by just assuming that sketches exist and satisfy a certain property. Then we will concentrate on how to implement similarity testing using information contained in a sketch.

Let D be some distribution over $\Sigma^\star$ and S a sample of i.i.d. examples drawn from D. The empirical distribution defined by S can be interpreted as a function $S : 2^{\Sigma^\star} \to [0,1]$, where for $A \subseteq \Sigma^\star$ the value $S(A)$ is the empirical probability of event A under sample S. A sketch is basically an approximation of this function over a particular family of subsets of $\Sigma^\star$. Let $0 < \nu < 1$. A *ν-sketch for S with respect to* $\mathrm{L}^{\mathrm{p}}_\infty$ is a function $\hat{S} : 2^{\Sigma^\star} \to [0,1]$ such that $\mathrm{L}^{\mathrm{p}}_\infty(S, \hat{S}) \leq \nu$.

In some cases, using ν-sketches for testing similarity of probability distributions is rather trivial. Suppose S and S' are samples from distributions D and D' respectively. If instead of S and S' we only have access to their ν-sketches $\hat{S}$ and $\hat{S}'$, then we can compute a statistic $\hat{\mu}_\nu = \mathrm{L}^{\mathrm{p}}_\infty(\hat{S}, \hat{S}')$. By definition of ν-sketch we have $|\hat{\mu}_\nu - \mathrm{L}^{\mathrm{p}}_\infty(S, S')| \leq 2\nu$. Therefore, trivial modifications of the proofs of Propositions 2.2.1 and 2.2.2 yield the two following results for testing with sketches using this statistic. The definition of $\Delta(\delta)$ is the same of Equation (2.1).

Corollary 2.4.1. *With probability at least $1 - \delta$ we have $\mu_\star \leq \hat{\mu}_\nu + 2\nu + \Delta(\delta)$.*

Corollary 2.4.2. *With probability at least $1 - \delta$ we have $\mu_\star \geq \hat{\mu}_\nu - 2\nu - \Delta(\delta)$.*

Based on these results, it is easy to obtain a provably correct similarity test for $\mu_\star$ using memory $O(1/\mu)$ by choosing, for example, $\nu = \mu/4$; that is because there exist ν-sketches using $O(\nu)$ memory. Details will be given in Section 3.3. Now we concentrate on the possibility of adapting the bootstrap approach from Section 2.3 to a sketching setting.

Deriving a test based on the bootstrap using sketches is much less straightforward than the case based on uniform convergence bounds. The main obstruction is the need to sample with replacement from a sample S to obtain bootstrapped samples $B_1, \ldots, B_r$. Obviously, the whole sample S needs to be stored in order to perform (exact) resampling. The rest of this section describes a method to obtain bootstrapped sketches $\hat{B}_1, \ldots, \hat{B}_r$ and derives a similarity test based on those sketches. We also state a theorem about confidence intervals built from bootstrapped sketches; full proofs are given in the next section.

Let r be some fixed integer. Suppose that an i.i.d. sample $S = (x_1, \ldots, x_m)$ from some distribution D is presented to an algorithm one example at a time. The algorithm will construct r sketch-bootstrapped samples $\hat{B}_1, \ldots, \hat{B}_r$ as follows. For each element $x_t \in S$ we drawn r indices $i_1, \ldots, i_r \in [r]$ independently at random from the uniform distribution over $[r]$. Then a copy of x_t is added to the sketch $\hat{B}_{i_j}$ for $1 \leq j \leq r$ – repetitions are allowed. Note that this sampling process matches several first order moments of the original bootstrap using sampling with replacement. In particular, the expected number of elements introduced in each sketch is m, and the expected number of occurrences of each string x in a particular sketch is $S[x]$. The same process is repeated with a sample S' from distribution D' and sketch-bootstrapped samples $\hat{B}'_1, \ldots, \hat{B}'_r$ are obtained.

Instead of r statistics like in the original bootstrap setting, in the sketched bootstrap we are going to compute r^2 statistics. Since the main motivation for considering the use of sketches is to save memory, and it turns out that memory usage grows linearly with r, this approach allows us to obtain more statistics with less memory, at the price of increasing the dependencies between them. Thus, for any $i, j \in [r]$ we compute statistic $\hat{\mu}_{i,j} = \mathrm{L}^{\mathrm{p}}_\infty(\hat{B}_i, \hat{B}'_j)$. As before, we will sort these statistics increasingly and obtain $\hat{\mu}_{\omega(1)} \leq \cdots \leq \hat{\mu}_{\omega(r^2)}$, where now ω is a bijection between $[r^2]$ and $[r] \times [r]$.

In the same way we did before, an heuristic based on Efron's bootstrap can be applied to this collection of statistics to say that $\hat{\mu}_{\omega(\lceil (1-\delta)r^2 \rceil)}$ is an upper confidence limit for $\mu_\star$ at level δ. However, in order to obtain formally justified bounds based on finite sample analyses, we proceed in a different direction. Let $1 \leq k \leq r^2$, $0 < \delta < 1$, and define the following.

$$\Delta_k(\delta) = \sqrt{\frac{8}{M'} \ln\left(\frac{32M}{\delta}\right)} + \sqrt{\frac{192}{M'} \ln\left(\frac{800r^3M^2}{\delta k^2}\right)} \ .$$

Then the following theorem gives an upper confidence limit for $\mu_\star$ based on the sketch-bootstrapped statistic $\hat{\mu}_{\omega(k)}$.

Theorem 2.4.3. *Let $1 \leq k \leq \min\{r^2, 30r^{10/7}\delta^{-4/7}M\}$. With probability at least $1-\delta$ we have $\mu_\star \leq \hat{\mu}_{\omega(k)} + 2\nu + \Delta_k(\delta)$.*

Using this result, a union bound argument automatically yields the following upper confidence limit based on the whole set of statistics.

Corollary 2.4.4. *With probability at least $1-\delta$ it holds that*

$$\mu_\star \leq \min\left\{\hat{\mu}_{\omega(k)} + 2\nu + \Delta_k(\delta) \ \middle|\ 1 \leq k \leq r^2\right\} \ .$$

2.5 Proof of Theorem 2.4.3

Our goal is to prove that for any $1 \leq k \leq \min\{r^2, 30r^{10/7}\delta^{-4/7}M\}$ it holds that $\mathbb{P}[\mu_\star > \hat{\mu}_{\omega(k)} + \Delta'] \leq \delta$, where $\Delta' = 2\nu + \Delta$ and $\Delta = \Delta_k(\delta)$. For $1 \leq i, j \leq r$ we write $Y_{i,j} = \mathbb{1}_{\hat{\mu}_{i,j} \leq \mu_\star - \Delta'}$ and $Z = \sum_{i,j} Y_{i,j}$. Since $\hat{\mu}_{\omega(1)} \leq \ldots \leq \hat{\mu}_{\omega(r^2)}$ are a reordering of $\hat{\mu}_{1,1}, \ldots, \hat{\mu}_{r,r}$ we have that by definition $\hat{\mu}_{\omega(k)} \leq \mu_\star - \Delta'$ implies $Z \geq k$.

Recall the definition $\hat{\mu}_{i,j} = \mathrm{L}^{\mathrm{p}}_{\infty}(\hat{B}_i, \hat{B}'_j)$ and suppose $t \geq \mathrm{L}^{\mathrm{p}}_{\infty}(D, S)$ and $t' \geq \mathrm{L}^{\mathrm{p}}_{\infty}(D', S')$. Then by the triangle inequality and the definition of ν-sketch we have, for any $1 \leq i, j \leq r$:

$$\mu_\star \leq \mathrm{L}^{\mathrm{p}}_{\infty}(S, B_i) + \mathrm{L}^{\mathrm{p}}_{\infty}(S', B'_j) + 2\nu + t + t' \ .$$

Therefore, if $\hat{\mu}_{i,j} \leq \mu_\star - \Delta'$, then $\mathrm{L}^{\mathrm{p}}_{\infty}(S, B_i) + \mathrm{L}^{\mathrm{p}}_{\infty}(S', B'_j) \geq \Delta - (t + t')$, which in turns implies that at least one of $\mathrm{L}^{\mathrm{p}}_{\infty}(S, B_i) \geq \gamma\Delta - t$ or $\mathrm{L}^{\mathrm{p}}_{\infty}(S', B'_j) \geq (1-\gamma)\Delta - t'$ holds, where $0 < \gamma < 1$ is arbitrary. For fixed γ, let us define indicator variables $X_i = \mathbb{1}_{\mathrm{L}^{\mathrm{p}}_{\infty}(S,B_i) \geq \gamma\Delta - t}$ and $X'_j = \mathbb{1}_{\mathrm{L}^{\mathrm{p}}_{\infty}(S',B'_j) \geq (1-\gamma)\Delta - t'}$. Then, the chain of implication we just derived can be rewritten as $Z_{i,j} \leq X_i + X'_j$. Furthermore, we get $Z \leq r(\sum_i X_i + \sum_j X'_j) = r(X + X')$.

Like we did in the proof of Theorem 2.3.1, for any t and t' we let $\mathfrak{G}(t, t')$ denote the event "$\mathrm{L}^{\mathrm{p}}_{\infty}(D, S) \leq t$ and $\mathrm{L}^{\mathrm{p}}_{\infty}(D', S') \leq t'$". Note that this event depends only on the sampling from D and D' used to obtain samples S and S' and is independent of the process generating the sketched bootstrapped samples once S and S' are given. Thus, we have shown that for any t, t', $0 < \gamma < 1$, and $0 < \beta < 1$, we have

$$\begin{aligned}
\mathbb{P}[\mu_\star > &\hat{\mu}_{\omega(k)} + \Delta' \mid \mathfrak{G}(t, t')] \\
&\leq \mathbb{P}[Z \geq k \mid \mathfrak{G}(t, t')] \\
&\leq \mathbb{P}[r(X + X') \geq k \mid \mathfrak{G}(t, t')] \\
&\leq \mathbb{P}[X \geq \beta k/r \mid \mathfrak{G}(t, t')] + \mathbb{P}[X' \geq (1-\beta)k/r \mid \mathfrak{G}(t, t')]
\end{aligned} \tag{2.2}$$

In order to bound the terms in this last expression we will need the two following lemmas.

Suppose S is a fixed sample of size m over $\Sigma^\star$ and that $B = B_i$ is a stream-bootstrapped sample.

Lemma 2.5.1. *The following holds for any $\rho \geq \sqrt{(48 \ln 2)/m}$:*

$$\mathbb{P}[\mathrm{L}^{\mathrm{p}}_{\infty}(S, B) > \rho] \leq 12m \exp\left(-\frac{m\rho^2}{48}\right) \ .$$

Proof. Note in the first place that since S contains at most m different strings, a shattering argument shows that the function $f(x) = |S(x\Sigma^\star) - B(x\Sigma^\star)|$ can take at most $2m$ different values when x runs over all possible strings in $\Sigma^\star$. Therefore by a union bound we have

$$\mathbb{P}[\mathrm{L}^{\mathrm{p}}_{\infty}(S, B) > \rho] \leq 2m \max_{x \in \Sigma^\star} \mathbb{P}[|S(x\Sigma^\star) - B(x\Sigma^\star)| > \rho] \ .$$

Next we bound the probability in the right hand side for an arbitrary string x such that $S[x\Sigma^\star] > 0$ – since otherwise the probability is zero. We begin by writing

$$\begin{aligned}
\mathbb{P}[|B(x\Sigma^\star) - &S(x\Sigma^\star)| > \rho] \\
&= \mathbb{P}[B(x\Sigma^\star) > S(x\Sigma^\star) + \rho] + \mathbb{P}[B(x\Sigma^\star) < S(x\Sigma^\star) - \rho] \ .
\end{aligned} \tag{2.3}$$

Let $\gamma > 0$. We bound the first term as

$$\begin{aligned}
\mathbb{P}[B(x\Sigma^\star) &> S(x\Sigma^\star) + \rho] \\
&\leq \mathbb{P}[B(x\Sigma^\star) > S(x\Sigma^\star) + \rho \mid |B| > (1-\gamma)m] + \mathbb{P}[|B| < (1-\gamma)m] \ ,
\end{aligned}$$

where the second term is at most $\exp(-\gamma^2 m/2)$ by the Chernoff bounds. The first term can be bounded by

$$\begin{aligned}
\mathbb{P}[B(x\Sigma^\star) &> S(x\Sigma^\star) + \rho \mid |B| > (1-\gamma)m] \\
&\leq \mathbb{P}[B[x\Sigma^\star] > (1-\gamma)S[x\Sigma^\star] + \rho(1-\gamma)m \mid |B| > (1-\gamma)m] \\
&\leq \frac{\mathbb{P}[B[x\Sigma^\star] > (1-\gamma)S[x\Sigma^\star] + \rho(1-\gamma)m]}{\mathbb{P}[|B| > (1-\gamma)m]} \ .
\end{aligned}$$

By Chernoff the denominator is at least $1 - \exp(-\gamma^2 m/2)$. Now, assuming that $\gamma \leq (1/2)(1 + S[x\Sigma^\star]/\rho m)^{-1}$, we have $\gamma(1 + \rho m/S[x\Sigma^\star]) \leq \rho m/2S[x\Sigma^\star]$. Therefore, using $S[x\Sigma^\star] \leq m$, Chernoff bounds yield

$$\mathbb{P}[B[x\Sigma^\star] > (1-\gamma)S[x\Sigma^\star] + \rho(1-\gamma)m] \leq e^{-m\rho^2/12} \ .$$

Thus, choosing $\gamma = \rho/4$ we obtain that for $\rho^2 \geq (32 \ln 2)/m$ it holds

$$\mathbb{P}[B(x\Sigma^\star) > S(x\Sigma^\star) + \rho] \leq 3e^{-m\rho^2/32} \ .$$

A similar argument applied to the other term in (2.3) shows that when $\rho^2 \geq (48 \ln 2)/m$ one has

$$\mathbb{P}[B(x\Sigma^\star) < S(x\Sigma^\star) - \rho] \leq 3e^{-m\rho^2/48} \ . \qquad \square$$

Suppose once again that S is a fixed arbitrary sample of size m. For $1 \leq i \leq r$ let B_i be stream-bootstrapped samples of S. Pick $\theta > 0$ and write $X_{i,\theta} = \mathbb{1}_{\mathrm{L}^{\mathrm{p}}_\infty(S,B_i) \geq \theta}$. Then define $X_\theta = \sum_i X_{i,\theta}$.

Lemma 2.5.2. *For any $\kappa > 0$ and $\theta \geq \sqrt{(48/m)\ln(12\sqrt{2}m/\kappa)}$ one has*

$$\mathbb{P}[X_\theta > \kappa r] \leq \frac{200m^2}{\kappa^2 r} \exp\left(-\frac{m\theta^2}{192}\right) \ .$$

Proof. The proof proceeds by bounding the variance of X_θ using the Efron–Stein inequality, and then plugging the bound into the Chebyshev–Cantelli inequality.

For $s \in [r]$ and $t \in [m]$ let $U_{s,t}$ be i.i.d. uniform random variables on $[r]$. Then we can think of $X_\theta = X_\theta(U_{1,1}, \ldots, U_{r,m})$ as a function of these random variables representing the tosses of strings in S into the bootstrapped sketches. Furthermore, for $s \in [r]$ and $t \in [m]$ let $U'_{s,t}$ denote an independent copy of each $U_{s,t}$. We write $X'_{\theta,s,t} = X_\theta(U_{1,1}, \ldots, U'_{s,t}, \ldots, U_{r,m})$ to denote the value of the function where $U_{s,t}$ has been replaced by $U'_{s,t}$. We will use the notation $B'_{i,s,t}$ to denote the stream-bootstrapped samples obtained by these alternative tossings.

Note that $X'_{\theta,s,t}$ is the number of sketches such that $\mathrm{L}^{\mathrm{p}}_\infty(S,B) > \theta$, where only one of the tosses has changed with respect to X_θ. If we write $U_{s,t} = i$ and $U'_{s,t} = j$, then only the values in $X_{i,\theta}$ and $X_{j,\theta}$ may have changed. Therefore, by the Efron–Stein inequality [BLB04] and symmetry with respect to the choice of s and t, we have

$$\begin{aligned}
\mathbb{V}[X_\theta] &\leq \frac{1}{2}\sum_{s,t} \mathbb{E}[(X_\theta - X'_{\theta,s,t})^2] \\
&= \frac{mr}{2}\mathbb{E}[(X_{i,\theta} + X_{j,\theta} - X'_{i,\theta} - X'_{j,\theta})^2] \\
&\leq 2mr(\mathbb{P}[X_{i,\theta} \neq X'_{i,\theta}] + \mathbb{P}[X_{j,\theta} \neq X'_{j,\theta}]) \ .
\end{aligned}$$

To bound the quantity above observe that if $X_{i,\theta} \neq X'_{i,\theta}$ then $\mathrm{L}^{\mathrm{p}}_\infty(S,B_i)$ must be close enough to the threshold θ in order to cross it by removing only one example. To see how close to the threshold $\mathrm{L}^{\mathrm{p}}_\infty(S,B_i)$ needs to be note that by the triangle inequality we have

$$|\,\mathrm{L}^{\mathrm{p}}_\infty(S,B_i) - \mathrm{L}^{\mathrm{p}}_\infty(S,B'_i)| \leq \mathrm{L}^{\mathrm{p}}_\infty(B_i,B'_i) \ ,$$

where now B'_i denotes the ith stream-bootstrapped sample obtained by a tossing that differs only in a coordinate that was originally assigned to B_i. Thus, since B'_i is obtained from B_i by removing a copy of x_t, it is easy to see that $\mathrm{L}^{\mathrm{p}}_\infty(B_i,B'_i) \leq 1/(|B_i|-1)$. Therefore, $X_{i,\theta} \neq X'_{i,\theta}$ necessarily implies $|\,\mathrm{L}^{\mathrm{p}}_\infty(S,B_i) - \theta| \leq 1/(|B_i|-1)$, and in particular

$$\mathbb{P}[X_{i,\theta} \neq X'_{i,\theta}] \leq \mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S,B_i) \geq \theta - 1/(|B_i|-1)] \ .$$

Let $\gamma > 0$ and $\tau = ((1-\gamma)m - 1)^{-1}$. We can bound the probability above as

$$\mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S,B_i) \geq \theta - 1/(|B_i|-1)] \leq \mathbb{P}[|B_i| < (1-\gamma)m] + \frac{\mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S,B_i) \geq \theta - \tau]}{\mathbb{P}[|B_i| \geq (1-\gamma)m]} \ .$$

Now, by Chernoff bounds the first term is at most $\exp(-m\gamma^2/2)$ and the denominator in the second term is at least $1 - \exp(-m\gamma^2/2)$. Furthermore, assuming $\gamma \leq (m - 1 - 2\theta^{-1})/m$, we have $\tau \leq \theta/2$ and Lemma 2.5.1 yields

$$\mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S,B_i) \geq \theta - \tau] \leq 12me^{-m(\theta/2)^2/48} \ ,$$

when $\theta^2/4 \geq (48 \ln 2)/m$. Since this last restriction implies $m \geq 4/\theta + 2$, we can choose $\gamma = 1/2$ and obtain

$$\mathbb{P}[X_{i,\theta} \neq X'_{i,\theta}] \leq 25me^{-m(\theta/2)^2/48} \ .$$

Altogether, the same arguments yield an identical bound for $\mathbb{P}[X_{j,\theta} \neq X'_{j,\theta}]$, from which it follows that

$$\mathbb{V}[X_\theta] \leq 100m^2re^{-m(\theta/2)^2/48} \ ,$$

for $\theta^2/4 \geq (48 \ln 2)/m$.

Now, by linearity of expectation and Lemma 2.5.1 we have

$$\mathbb{E}[X_\theta] = r\mathbb{P}[\mathrm{L}^{\mathrm{p}}_\infty(S, B) > \theta] \leq 12mre^{-m\theta^2/48} \ ,$$

when $\theta^2 \geq (48 \ln 2)/m$. Furthermore, if $\theta^2 \geq (48/m) \ln(12\sqrt{2}m/\kappa)$, then $\mathbb{E}[X_\theta] \leq \kappa r/\sqrt{2}$. Thus, under this condition, the Chebyshev–Cantelli inequality yields

$$\mathbb{P}[X_\theta > \kappa r] \leq \frac{\mathbb{V}[X_\theta]}{(\kappa r - \mathbb{E}[X_\theta])^2} \leq \frac{200m^2}{\kappa^2 r} e^{-m(\theta/2)^2/48} \ . \qquad \square$$

Note that the events considered in Lemmas 2.5.1 and 2.5.2 depend only on the process that samples the stream-bootstrapped samples B_i and B'_j. Thus, we can proceed to bound the terms in (2.2) as follows:

$$\mathbb{P}[X \geq \beta k/r \mid \mathfrak{G}(t, t')] \leq \mathbb{P}[X_{\gamma\Delta - t} \geq (\beta k/r^2)r] \leq \frac{200m^2r^3}{\beta^2k^2} e^{-m(\gamma\Delta - t)^2/192} \ .$$

Obviously, we also have

$$\mathbb{P}[X' \geq (1 - \beta)k/r \mid \mathfrak{G}(t, t')] \leq \frac{200m'^2r^3}{(1 - \beta)^2k^2} e^{-m'((1-\gamma)\Delta - t')^2/192} \ .$$

Now, by setting $t' = t\sqrt{m/m'}$, $\gamma = \sqrt{m'}/(\sqrt{m} + \sqrt{m'})$, and $\beta = m/(m + m')$, we obtain

$$\mathbb{P}[\mu_\star > \hat{\mu}_{\omega(k)} + \Delta' \mid \mathfrak{G}(t, t')] \leq \frac{400r^3M^2}{k^2} e^{-M'(\Delta - \tau)^2/192} \ ,$$

where $\tau = t/\gamma$ and we require that $\Delta \geq \tau + \sqrt{(48/M') \ln(12\sqrt{2}r^2M/k)}$. Finally, recalling Equation (2.3.1) from the proof of Theorem 2.3.1 and choosing $\tau = \sqrt{(8/M') \ln(32M/\delta)}$, the definition of Δ yields

$$\begin{aligned}\mathbb{P}[\mu_\star < \hat{\mu}_{\omega(k)} + \Delta'] &\leq \mathbb{P}[\bar{\mathfrak{G}}(t, t')] + \mathbb{P}[\mu_\star < \hat{\mu}_{\omega(k)} + \Delta' \mid \mathfrak{G}(t, t')] \\ &\leq 16Me^{-M'\tau^2/8} + \frac{400r^3M^2}{k^2} e^{-M'(\Delta - \tau)^2/192} = \delta \ .\end{aligned}$$

The condition on k follows from the constraint

$$\sqrt{\frac{192}{M'} \ln \frac{800r^3M^2}{\delta k^2}} \geq \sqrt{\frac{48}{M'} \ln \frac{12\sqrt{2}r^2M}{k}} \ .$$

2.6 Experimental Results

In this section we report some experimental results showing that bootstrap-based tests for similarity testing behave very well in practice. In particular we compare a test based on upper confidence bounds obtained with Efron's bootstrap with the tests derived from Propositions 2.2.1 and 2.2.2.

We begin by describing the experimental setup. First we fixed an alphabet Σ with $|\Sigma| = 4$ and defined four PDFA with two states realizing four different distributions over $\Sigma^\star$: D_0, $D_{0.05}$, $D_{0.1}$, and $D_{0.25}$. Those were designed in such a way that $\mathrm{L}^{\mathrm{p}}_\infty(D_0, D_{0.05}) = 0.05$, $\mathrm{L}^{\mathrm{p}}_\infty(D_0, D_{0.1}) = 0.1$, and $\mathrm{L}^{\mathrm{p}}_\infty(D_0, D_{0.25}) = 0.25$. Using samples generated from these distributions we tested the power of bootstrap tests under

different distinguishability conditions. Specifically, for different values of m, we obtained samples S_0 from D_0 and S_μ from D_μ with $|S_0| = |S_\mu| = m$ for all $\mu \in \{0.25, 0.1, 0.05, 0\}$. Then we computed the empirical estimate $\hat{\mu} = \mathrm{L}^{\mathrm{p}}_{\infty}(S_0, S_\mu)$. Furthermore, for some fixed r we also obtained bootstrapped samples $B_1, \ldots, B_r$ from S_0 and $B'_1, \ldots, B'_r$ from S_μ. Then we used those to obtain bootstrapped estimates $\hat{\mu}_i = \mathrm{L}^{\mathrm{p}}_{\infty}(B_i, B'_i)$ for $i \in [r]$; we assume without loss of generality that $\hat{\mu}_1 \leq \cdots \leq \hat{\mu}_r$. Using all this information, we then picked some confidence parameter $0 < \delta < 1$ and computed upper bounds for $\mathrm{L}^{\mathrm{p}}_{\infty}(D_0, D_\mu)$ of the form $\hat{\mu} + \Delta(\delta)$ and $\hat{\mu}_{\lceil (1-\delta)r \rceil}$. We also computed the lower bound $\hat{\mu} - \Delta(\delta)$. All these was done for all values of $r \in \{20, 50, 100\}$ and $\delta \in \{0.25, 0.1, 0.05\}$. The values of m depended on each choice of μ. In particular, we picked $m = t \cdot m_\mu$ for $t \in \{1, 2, \ldots, 25\}$, where $m_{0.25} = 800$, $m_{0.1} = 5600$, $m_{0.05} = 10000$, and $m_0 = 4000$. These m_μ where chosen to ensure that a test based on VC bounds is able to make the correct decision for $m = 25m_\mu$. For each of these experiments, we made $K = 20$ independent repetitions. Results are given in Table 2.1 and Figure 2.1.

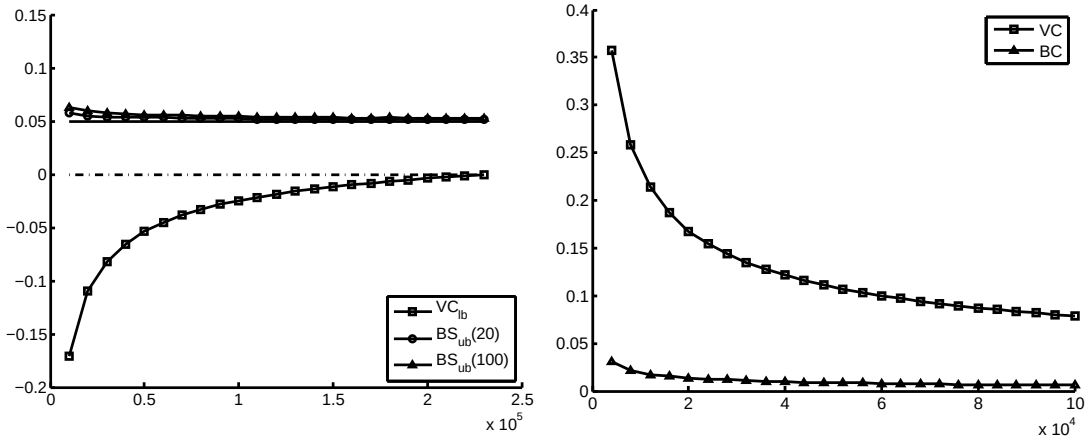
For the case $\mu > 0$ we wanted to study the rate of false positives of the following test based on bootstrap estimates: decide that two distributions are similar when $\hat{\mu}_{\lceil (1-\delta)r \rceil} < \mu$. That is, we are interested in seeing how often do the bootstrap upper bound fall below the true μ. In particular, this test has to compete against the following test based on VC bounds: decides that two distributions are different when $\hat{\mu} - \Delta(\delta) > 0$. In this scheme, we see a false positive whenever $\hat{\mu}_{\lceil (1-\delta)r \rceil} < \mu$ happens before $\hat{\mu} - \Delta(\delta) > 0$. In particular, for a sequence of estimates $\hat{\mu} - \Delta(\delta)$ with growing m, we found $m_{\mathrm{cut}} = t_{\mathrm{cut}} \cdot m_\mu$ such that the average of $\hat{\mu} - \Delta(\delta)$ over the K experiments was positive. Then we counted how many times, for $1 \leq t \leq t_{\mathrm{cut}}$ and $1 \leq k \leq K$, the upper bound $\hat{\mu}_{\lceil (1-\delta)r \rceil}$ fell below μ; we denote this count of "fails" by F_μ. Furthermore, we wanted to know how smaller than μ can $\hat{\mu}_{\lceil (1-\delta)r \rceil}$ become. Thus, we computed the minimum $\hat{\mu}_{\min}$ of this quantity over the same range of t and k. These results are detailed in Table 2.1. Also, Figure 2.1a shows the evolution of the average of $\hat{\mu}_{\lceil (1-\delta)r \rceil}$ over the K experiments for different values of m. Overall we observe that the behavior of the bootstrap tests is very good, even for small values of r. In particular, we observe in almost all cases a difference of one order of magnitude between the confidence parameter and the estimated false positive rate. We also see, as expected, that for smaller values of μ the test tends do commit more errors. However, such errors are never very significant for small values of δ, since we see in the table that $\hat{\mu}_{\min}$ is never much smaller than μ in those cases.

When $\mu = 0$ we were interested in comparing the upper bounds provided by bootstrapped estimates with the ones obtained via VC bounds. We can see in Figure 2.1b that bootstrap bounds are much tighter and converge faster to zero by a factor of about 10. Thus, they provide a tool for deciding that μ is below a certain threshold with less examples than required by VC bounds. In Figures 2.1c and 2.1d we compare the effect of r and δ in the behavior of these upper bounds. We observe that all cases have similar variances which decrease when the sample sizes grows. In terms of the mean we see that $r = 20$ yields slightly less stable estimates than $r = 100$.

Overall, we can conclude that tests based on Efron's bootstrap are a promising tool for testing distribution similarity in state-merging algorithms. We also note that such tests behave very well for moderate parameter choices, which allows for efficient testing with moderate sample sizes.

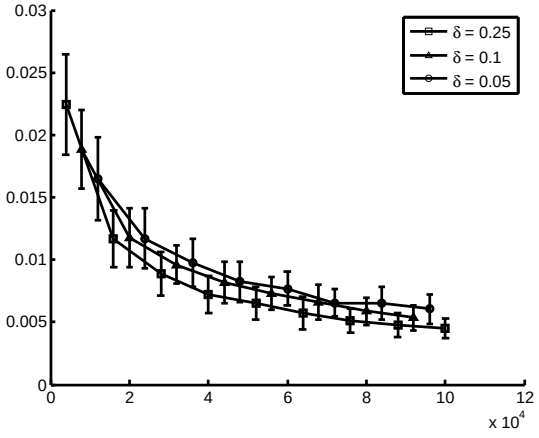
		μ	m_{cut}	$\hat{\mu}_{\text{min}}$	F_μ (Kt_{cut})	F_μ/Kt_{cut}
$r = 20$	$\delta = 0.25$	0.25	6400	0.235	7 (160)	0.044
		0.10	44800	0.097	13 (160)	0.081
		0.05	210000	0.045	25 (420)	0.060
	$\delta = 0.10$	0.25	7200	0.242	4 (180)	0.022
		0.10	50400	0.098	5 (180)	0.028
		0.05	220000	0.047	11 (440)	0.025
	$\delta = 0.05$	0.25	7200	0.243	4 (180)	0.022
		0.10	50400	0.098	3 (180)	0.017
		0.05	230000	0.047	8 (460)	0.017
$r = 50$	$\delta = 0.25$	0.25	6400	0.239	4 (160)	0.025
		0.10	44800	0.097	8 (160)	0.050
		0.05	210000	0.044	23 (420)	0.055
	$\delta = 0.10$	0.25	7200	0.246	5 (180)	0.028
		0.10	50400	0.099	3 (180)	0.017
		0.05	220000	0.047	6 (440)	0.014
	$\delta = 0.05$	0.25	7200	0.250	0 (180)	0.000
		0.10	50400	0.099	1 (180)	0.006
		0.05	230000	0.048	3 (460)	0.007
$r = 100$	$\delta = 0.25$	0.25	6400	0.235	6 (160)	0.037
		0.10	44800	0.096	11 (160)	0.069
		0.05	210000	0.046	24 (420)	0.057
	$\delta = 0.10$	0.25	7200	0.247	5 (180)	0.028
		0.10	50400	0.099	3 (180)	0.017
		0.05	220000	0.047	10 (440)	0.023
	$\delta = 0.05$	0.25	7200	0.250	1 (180)	0.006
		0.10	50400	0.100	0 (180)	0.000
		0.05	230000	0.049	5 (460)	0.011

Table 2.1: Experimental results comparing different parameter choices for bootstrapped upper bounds.

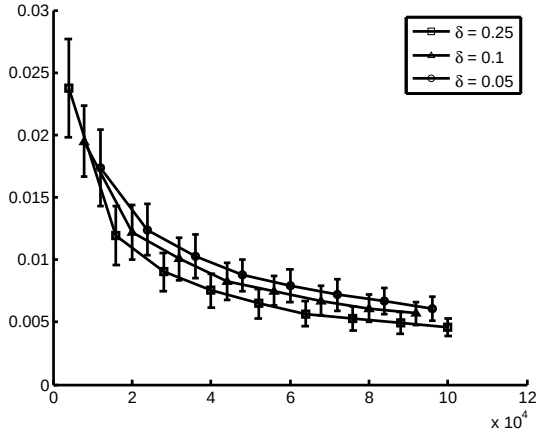


(a) $r = 20, 100$, $\mu = 0.05$, $\delta = 0.05$

(b) $r = 100$, $\mu = 0$, $\delta = 0.05$



(c) $r = 20$, $\mu = 0$



(d) $r = 100$, $\mu = 0$

Figure 2.1: Experimental results comparting VC and bootstrap bounds. Averages and standard deviations over 20 repetitions. a. Lower bound with VC bounds and upper bounds with bootstrap $r = 20, 100$, for $\mu = 0.05$ and $\delta = 0.05$. b. Upper bounds with VC and bootstrap with $r = 100$, $\mu = 0$, and $\delta = 0.05$. c. Bootstrap upper bounds with $r = 20$ and $\mu = 0$. d. Bootstrap upper bounds with $r = 100$ and $\mu = 0$.

Chapter 3

Learning PDFA from Data Streams Adaptively

In a data streams scenario, algorithms can only access data in the form of a sequence of examples that must be processed on-line using very little time and memory per example. It turns out that this computational model describes a rather natural framework in which several real-world problems related to network traffic analysis, social web mining, and industrial monitoring can be cast.

Most algorithms in the streaming model fall into one of the following two classes: a class containing primitive building blocks, like change detectors and sketching algorithms for computing statistical moments and finding frequent items; and a class containing full-featured data mining algorithms, like frequent itemsets miners, decision tree learners, and clustering algorithms. A generally valid rule is that primitives from the former class can be combined for building algorithms in the latter class.

In this chapter we design a new state-merging algorithm for learning PDFA in this demanding algorithmic paradigm. In particular, we will describe algorithms in both of the categories mentioned above, and use them to design a complete learning system that, in addition to learning PDFA using very little memory and processing time per example, will be able to detect changes in the distribution that generates the stream and adapt the learning process accordingly. Regarding the state-merging component, the two main contributions of this chapter are the design of an efficient and adaptive scheduling policy to perform similarity tests as soon as enough information is available, and the use of sketching methods to find frequent prefixes in streams of strings which yield a PAC learning algorithm for PDFA using $O(1/\mu)$ memory, in contrast with the usual $O(1/\mu^2)$ required by batch methods. Other novel contributions include a parameter search strategy that discovers the number of states and distinguishability of an unknown target PDFA, and a change detector module that can determine when the distribution generating the stream has changed.

The chapter begins by recalling the basic facts about the data stream model, and then proceeds to give a high-level description of our system for learning PDFA in this model. Details on the individual building blocks of this system come next. In particular, we present a sketch for finding frequent prefixes in a stream of strings, an adaptive state-merging algorithm, a parameter selection strategy fulfilling the restrictions of the data stream paradigm, and a change detector with guarantees on the number of false negatives and positives.

3.1 The Data Stream Framework

The *data stream* computation model has established itself in the last fifteen years for the design and analysis of algorithms on high-speed sequential data [Agg07]. It is characterized by the following assumptions:

1. The input is a potentially infinite sequence of items $x_1, x_2, \ldots, x_t, \ldots$ from some (large) universe X.

2. Item x_t is available only at the tth time step and the algorithm has only that chance to process it, probably by incorporating it to some summary or sketch; that is, only one pass over the data is allowed.

3. Items arrive at high-speed, so the processing time per item must be very low – ideally, constant time, but most likely, logarithmic in t and $|X|$.

4. The amount of memory used by algorithms (the size of the sketches alluded above) must be sublinear in the data seen so far at every moment; ideally, at time t memory must be polylogarithmic in t – for many computations, this is impossible and memory of the form t^c for constant $c < 1$ may be required. Logarithmic dependence on $|X|$ is also desirable.

5. Approximate, probabilistic answers are often acceptable.

A large fraction of the data stream literature discusses algorithms working under worst-case assumptions on the input stream, e.g. compute the required (approximate) answer at all times t for every possible values of $x_1, \ldots, x_t$ [LZ08; Mut05]. For example, several sketches have been proposed for computing an ε-approximation to the number of distinct items seen in the stream using memory $O(\log(t|X|)/\varepsilon)$. In machine learning and data mining, this is often not the problem of interest: one is interested in modeling the current "state of the world" at all times, so the current items matter much more than those from the far past [Bif10; Gam10]. An approach is to assume that each item x_t is generated by some underlying distribution D_t over X, that varies over time, and the task is to track the distributions D_t (or its relevant information) from the observed items. Of course, this is only possible if these distributions do not change too wildly, e.g. if they remain unchanged for fairly long periods of time ("distribution shifts", "abrupt change"), or if they change only very slightly from t to $t + 1$ ("distribution drift"). A common simplifying assumption (which, though questionable, we adopt here) is that successive items are generated independently, i.e. that x_t depends only on D_t and not on the outcomes x_{t-1}, x_{t-2}, etc.

In our case, the universe X will be the infinite set $\Sigma^\star$ of all string over a finite alphabet Σ. Intuitively, the role of $\log(|X|)$ will be replaced by a quantity such as the expected length of strings under the current distribution.

3.2 A System for Continuous Learning

Before getting into the technicalities of our algorithm for learning PDFA from data streams, we begin with a general description of a complete learning system capable of adapting to changes and make predictions about future observations. In the following sections we will describe the components involved in this system in detail and prove rigorous time and memory bounds.

The goal of the system is to keep at all times a hypothesis – a PDFA in this case – that models as closely as possible the distribution of the strings currently being observed in the stream. For convenience, the hypothesis PDFA will be represented as two separate parts: the DFA containing the inferred transition structure, and tables of estimations for transition and stopping probabilities. Using this representation, the system is able to decouple the state-merging process that learns the transition structure of the hypothesis from the estimation procedure that computes transition and stopping probabilities. This decomposition is also useful in terms of change detection. Change in transition or stopping probabilities can be easily tracked with a simple sliding window technique. On the other hand, changes in the structure of a PDFA are much harder to detect, and modifying an existing structure to adapt to this sort of changes is a very challenging problem. Therefore, our system contains a block that continually estimates transition and stopping probabilities, and another block that detects changes in the underlying structure and triggers a procedure that learns a new transition structure from scratch. A final block that uses the current hypothesis to make predictions about the observations in the stream can also be integrated into the system. Since this block will depend on the particular application, it will not be discussed further here. The structure we just described is depicted in Figure 3.1.

The information flow in the system works as follows. The structure learning block – containing a state-merging algorithm and a parameter search block in charge of finding the correct n and μ for the

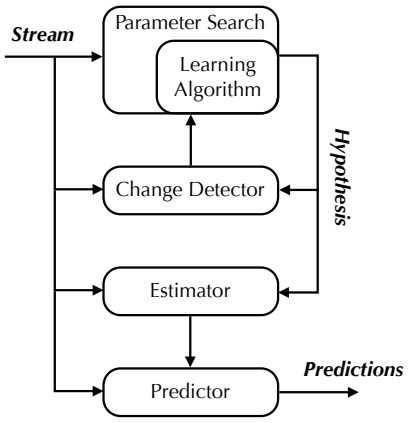


Figure 3.1: System for Continuous Learning from Data Streams

target – is started and the system waits until it produces a first hypothesis DFA. This DFA is fed to the probability estimation block and the change detector. From now on, these two blocks run in parallel, as well as the learning block, which keeps learning new structures with more refined parameters. If at some point a change in the target structure is found, a signal is emitted and the learning block restarts the learning process.[1] In parallel, the estimator block keeps updating transition and stopping probabilities all the time. It may be the case that this adaptation procedure is enough to track the current distribution. However, if the structure learning procedure produces a new DFA, transition probabilities are estimated for this new structure, which then takes the place of the current hypothesis. Thus, the system will recover much faster from a change that only affects transition probabilities than from a change in the structure of the target.

3.3 Sketching Distributions over Strings

In this section we describe sketches that can be used by our state-merging algorithm in data streams. The basic building block of these sketches is the Space-Saving algorithm [MAA05]. We will use this sketch to keep information about *frequent prefixes* in a stream of strings over $\Sigma^\star$. This information will then be used to compute the statistics required by the similarity tests described in Section 2.4.

We begin by recalling the basic properties of the Space-Saving sketch introduced in [MAA05]. Given a number of counters K, the Space-Saving sketch $\texttt{SpSv}(K)$ is a data structure that uses memory $O(K)$ at all times and has two basic operations. The first is an *insertion* operation that receives an element and adds it to the sketch in time $O(1)$. We use f_x to denote the number of times an element x has been added to the sketch. The number of elements added to the sketch up to some point is $m = \sum_x f_x$. The second is a *retrieval* operation that given some $\varepsilon \geq 1/K$ takes time $O(1/\varepsilon)$ and returns a set of at most K pairs of the form $(x, \hat{f}_x)$ such that $0 \leq \hat{f}_x - f_x \leq m/K$, and which is guaranteed to contain every x whose $f_x \geq m\varepsilon$. Using these operations an algorithm can maintain a summary of the most frequent elements seen in a stream together with an approximation to their current absolute or relative frequency.

In order to be able to retrieve frequent prefixes from a stream of strings, some modifications have to be made to this sketch. A first observation is that whenever we observe a string x of length $|x|$ in the stream, we must insert $|x| + 1$ prefixes to the sketch. This is another way of saying that under a distribution D over $\Sigma^\star$ events of the form $x\Sigma^\star$ and $y\Sigma^\star$ are not independent if $x \sqsubseteq y$. In fact, it can be shown that $\sum_x D(x\Sigma^\star) = L + 1$, where $L = \sum_x |x| D(x)$ is the expected length of D [CT04a]. In practice, a good estimate for L can be easily obtained from an initial fraction of the stream. It is easy

[1] Here one has a choice of keeping the current parameters in or restarting them to some initial values; prior knowledge about the changes the algorithm will be facing can help to make an informed decision in this point.

to see that a Space-Saving sketch with $O(KL)$ counters can be used to retrieve prefixes with relative frequencies larger than some $\varepsilon \geq 1/K$ and approximating these frequencies with error at most $O(1/K)$. When computing relative frequencies, the absolute frequency obtained via a retrieval operation needs to be divided by the number of strings added to the sketch so far (instead of the number of prefixes).

We encapsulate all this behavior into a *Prefix-Space-Saving* sketch $\texttt{SpSv}^{\mathrm{P}}(K)$, which is basically a Space-Saving sketch with K counters where when one string is inserted, each proper prefix of the string is inserted into the sketch as well. A string x is processed in time $O(|x|)$. Such a sketch can be used to keep information about the frequent prefixes in a stream of strings, and the information in two Prefix-Space-Saving sketches corresponding to streams generated by different distributions can be used to approximate their $\mathrm{L}_\infty^{\mathrm{P}}$ distance.

We now analyze the error introduced by the sketch on the empirical $\mathrm{L}_\infty^{\mathrm{P}}$ distance between the "exact" empirical distribution corresponding to a sample S and its sketched version which we denote by $\hat{S}$. Fix $K > 0$. Given a sequence $S = (x_1, \ldots, x_m)$ of strings from $\Sigma^\star$, for each prefix $x \in \Sigma^\star$ we denote by $\hat{S}[x\Sigma^\star]$ the absolute frequency returned for prefix x by a Prefix-Space-Saving sketch $\texttt{SpSv}^{\mathrm{P}}(K)$ that received S as input; that is, $\hat{S}[x\Sigma^\star] = \hat{f}_x$ if the pair $(x, \hat{f}_x)$ was returned by a retrieval query with $\varepsilon = 1/K$, and $\hat{S}[x\Sigma^\star] = 0$ otherwise. Furthermore, $\hat{S}(x\Sigma^\star)$ denotes the relative frequency of the prefix x in $\hat{S}$: $\hat{S}(x\Sigma^\star) = \hat{S}[x\Sigma^\star]/m$. The following result analyzes the maximum of the differences $|\hat{S}(x\Sigma^\star) - S(x\Sigma^\star)|$.

Lemma 3.3.1. *Let $S = (x_1, \ldots, x_m)$ be a sequence of i.i.d. examples from a PDFA D and $\hat{S}$ a $\texttt{SpSv}^{\mathrm{P}}(K)$ sketch where each element of S has been inserted. Then with probability at least $1 - \delta$ the following holds:*

$$\mathrm{L}_\infty^{\mathrm{P}}(S, \hat{S}) \leq \frac{L+1}{K} + \sqrt{\frac{32e^2}{mK^2c_D^2} \ln\left(\frac{1}{\delta}\right)} \ .$$

Proof. In the first place note that the number of elements inserted into the underlying space-saving sketch is $M = \sum_{i=1}^{m}(|x_i| + 1)$. This is a random variable whose expectation is $\mathbb{E}[M] = m(L+1)$. We claim that for any $x \in \Sigma^\star$ we have $|S(x\Sigma^\star) - \hat{S}(x\Sigma^\star)| \leq M/Km$. If $\hat{S}[x\Sigma^\star] = 0$, that means that $S[x\Sigma^\star] < M/K$, and therefore the bound holds. On the other hand, the guarantee on the sketch gives us $|S[x\Sigma^\star] - \hat{S}[x\Sigma^\star]|/m \leq M/Km$. Now, since $Z = M - m$ is the sum of m i.i.d. subexponential random variables by Lemma 1.1.1, we can apply Lemma A.1.5 and obtain

$$\mathbb{P}[Z - \mathbb{E}[Z] \geq mt] \leq \exp\left(-\frac{m}{8e^2} \min\left\{\frac{t^2c_D^2}{4}, \frac{tc_D}{2}\right\}\right) \ .$$

The bound follows from choosing $t = \sqrt{(32e^2/mc_D^2)\ln(1/\delta)}$. □

Thus, with the notation of Section 2.4, a Prefix-Space-Saving sketch with K counters that receives as input strings generated by a PDFA D is a ν-sketch, where

$$\nu = \frac{L+1}{K} + \sqrt{\frac{32e^2}{mK^2c_D^2} \ln\left(\frac{1}{\delta}\right)} \ .$$

Plugging this value into the results from Section 2.4 one obtains confidence intervals for $\mu_\star = \mathrm{L}_\infty^{\mathrm{P}}(D, D')$ using Prefix-Space-Saving sketches.

In the state merging algorithm described in next section, a sketch will be associated with each state in the hypothesis in order to keep an approximation to a sample of the distribution generated from that state. The particular form of the sketch will depend on the similarity test used by the algorithm, but all sketches will use the Prefix-Space-Saving sketch as a basic building block. In particular, if a tests based on VC bounds is used (cf. Corollaries 2.4.1 and 2.4.2), then to each state q we will associate a sketch $\hat{S}_q = \texttt{SpSv}^{\mathrm{P}}(\lceil 1/\alpha \rceil)$ for some parameter α that will depend on the distinguishability and expected length of the target PDFA. In this case, each state will use memory $O(1/\alpha)$ and each insertion operation will take time $O(|x|)$. Furthermore, given two sketches $\hat{S}$ and $\hat{S}'$, the statistics $\mathrm{L}_\infty^{\mathrm{P}}(\hat{S}, \hat{S}')$ can be computed in time $O(1/\alpha)$ because it only requires two retrieval operations and computing a maximum among $O(1/\alpha)$ items.

$$
\begin{aligned}
\alpha_0 &= 128 \\
\alpha &= 2 \\
\beta_0 &= (64n|\Sigma|/\varepsilon)\ln(2/\delta') \\
\beta &= (64n|\Sigma|/\varepsilon)\ln 2 \\
\theta &= (3\varepsilon)/(8n|\Sigma|) \\
\delta' &= \delta/2|\Sigma|n(n+2) \\
\delta_i &= 6\delta'/\pi^2 i^2
\end{aligned}
$$

Table 3.1: Definitions used in Algorithm 2

If a bootstrap similarity test is used, then each state will have a composite sketch of the form $\hat{S}_q = (\texttt{SpSv}^{\mathrm{p}}(\lceil 1/\alpha_1 \rceil), \hat{B}_1, \ldots, \hat{B}_r)$. Items will be assigned to each $\hat{B}_i = \texttt{SpSv}^{\mathrm{p}}(\lceil 1/\alpha_2 \rceil)$ following the bootstrapping scheme described in Section 2.4. The sketch with parameter α_1 will receive every item from the multiset S_q and will be used to test dissimilarity using Corollary 2.4.2. The bootstrapped sketches $\hat{B}_1, \ldots, \hat{B}_r$ will be used to test similarity according to Corollary 2.4.4. Now each state will use memory $O(1/\alpha_1 + r/\alpha_2)$ and each insertion operation will take time $O(r|x|)$. Given two such sketches, upper and lower confidence bounds will be computed in time $O(1/\alpha_1 + r^2/\alpha_2)$.

3.4 State-merging in Data Streams

In this section we present an algorithm for learning distributions over strings from a data stream. The algorithm learns a PDFA by adaptively performing statistical tests in order to discover new states in the automaton and merge similar states. We focus on the state-merging aspect of the algorithm, which is in charge of obtaining the transition structure between states. This stage requires the use of sketches to store samples of the distribution generated from each state in the PDFA, and a testing sub-routine to determine whether two states are equal or distinct based on the information contained in these sketches. Since both of these components have already been discussed in detail elsewhere, here we will just assume that components satisfying certain assumptions are used in the algorithm, without giving further implementation details. After finding the transition structure between states, a second stage in which transition and stopping probabilities are estimate takes place. The implementation of this stage is routine and will not be discussed here.

The algorithm will read successive strings over Σ from a data stream and, after some time, output a PDFA. Assuming the distribution generating the stream is stationary and generated from a PDFA, we will show that then the output will be accurate with high probability. The procedure requires as input a guess on the number of states in the target.

We begin with an informal description of the algorithm, which is complemented by the pseudo-code in Algorithm 2. The algorithm follows a structure similar to other state-merging algorithms, like the one described in Section 1.4, though here tests to determine similarity between states are performed adaptively as examples arrive.

Our algorithm requires some parameters as input: the usual accuracy ε and confidence δ, a finite alphabet Σ, and a number of states n. Some quantities defined in terms of these parameters that are used in the algorithm are given in Table 3.1. The algorithm, which is called `StreamLearner`, reads data from a stream of strings over Σ. At all times it keeps a hypothesis represented by a DFA. States in the DFA are divided into three kinds: *safes*, *candidates*, and *insignificants*, with a distinguished *initial* safe state denoted by q_λ. Candidate and insignificant states have no out-going transitions. To each string $w \in \Sigma^\star$ we may be able to associate a state by starting at q_λ and successively traversing the transitions labeled by the symbols of w in order. If all transitions are defined, the last state reached is denoted by

q_w, otherwise q_w is undefined – note that by this procedure different strings w and w' may yield $q_w = q_{w'}$.

For each state q_w, `StreamLearner` keeps a multiset S_w of strings. These multisets grow with the number of strings processed by the algorithm and are used to keep statistical information about a distribution D_{q_w}. In fact, since the algorithm only needs information from frequent prefixes in the multiset, it does not need to keep the full multiset in memory. Instead, it uses sketches to keep the relevant information for each state. We use $\hat{S}_w$ to denote the information contained in these sketches associated with state q_w, and $|\hat{S}_w|$ to denote the number of strings inserted into the sketch associated with state q_w.

Execution starts from a DFA consisting of a single safe state q_λ and several candidates q_σ, one for each $\sigma \in \Sigma$. All states start with an empty sketch. Each element x_t in the stream is then processed in turn: for each prefix w of $x_t = wz$ that leads to a state q_w in the DFA, the corresponding suffix z is added to that state's sketch $\hat{S}_{q_w}$. During this process, similarity and insignificance tests are performed on candidate states following a certain schedule; the former are triggered by the sketch's size reaching a certain threshold, while the latter occur at fixed intervals after the state's creation. In particular, t_w^0 denotes the time state q_w was created, t_w^s is a threshold on the size $|\hat{S}_w|$ that will trigger the next round of similarity tests for q_w, and t_w^u is the time the next insignificance test will occur. Parameter i_w keeps track of the number of similarity tests performed for state q_w; this is used to adjust the confidence parameter in those tests.

Insignificance tests are used to check whether the probability that a string traverses the arc reaching a particular candidate is below a certain threshold; it is known that these transitions can be safely ignored when learning a PDFA [CT04a; PG07]. Similarity tests use statistical information provided by the candidate's sketch to determine whether it equals some already existing safe or it is different from all safes in the DFA. These tests can return three values: `equal`, `distinct`, and `unknown`. These answers are used to decide what to do with the candidate currently being examined.

A candidate state will exist until it is promoted to safe state, merged to another safe, or declared insignificant. When a candidate is merged to a safe, the sketches associated with that candidate are discarded. The algorithm will end whenever there are no candidates left, or when the number of safe states surpasses the limit n given by the user.

3.4.1 Analysis

Now we proceed to analyze the `StreamLearner` algorithm. We will consider memory and computing time used by the algorithm, as well as accuracy of the hypothesis produced in the case when the stream is generated by a PDFA. Our analysis will be independent of the particular sketching methodology and similarity test used in the algorithm. In this respect, we will only require that the particular components used in `StreamLearner` to that effect satisfy the following assumptions.

Assumption 1. *Algorithms* `Sketch` *and* `Test` *algorithm used in* `StreamLearner` *satisfy the following:*

1. *each instance of* `Sketch` *uses memory at most* M_s,
2. *a* `Sketch.insert`(x) *operation takes time* $O(|x|T_\mathrm{s})$,
3. *any call* `Test`$(\hat{S}, \hat{S}', \delta)$ *takes time at most* T_t,
4. *there exists a* N_u *such that if* $|\hat{S}|, |\hat{S}'| \geq N_\mathrm{u}$, *then a call* `Test`$(\hat{S}, \hat{S}', \delta)$ *will never return* `unknown`,
5. *there exists a* N_e *such that if a call* `Test`$(\hat{S}, \hat{S}', \delta)$ *returns* `equal`, *then necessarily* $|\hat{S}| \geq N_\mathrm{e}$ *or* $|\hat{S}'| \geq N_\mathrm{e}$,
6. *when a call* `Test`$(\hat{S}, \hat{S}', \delta)$ *returns either* `equal` *or* `distinct`, *then the answer is correct with probability at least* $1 - \delta$.

Our first result is about memory and number of examples used by the algorithm.

Theorem 3.4.1. *The following hold for any call to* `StreamLearner`$(n, \Sigma, \varepsilon, \delta)$*:*

1. *The algorithm uses memory* $O(n|\Sigma|M_\mathrm{s})$

Algorithm 2: `StreamLearner` procedure

Input: Parameters $n, \Sigma, \varepsilon, \delta, \texttt{Sketch}, \texttt{Test}$
Data: A stream of strings $x_1, x_2, \ldots \in \Sigma^\star$
Output: A hypothesis H

initialize H with safe q_λ and let $\hat{S}_\lambda \leftarrow \texttt{Sketch}$;
foreach $\sigma \in \Sigma$ **do**
    add a candidate q_σ to H and let $\hat{S}_\sigma \leftarrow \texttt{Sketch}$;
    $t^0_\sigma \leftarrow 0$, $t^s_\sigma \leftarrow \alpha_0$, $t^u_\sigma \leftarrow \beta_0$, $i_\sigma \leftarrow 1$;
foreach *string* x_t *in the stream* **do**
    foreach *decomposition* $x_t = wz$, *with* $w, z \in \Sigma^\star$ **do**
        if q_w *is defined* **then**
            $\hat{S}_w.\texttt{insert}(z)$;
            if q_w *is a candidate* **and** $|\hat{S}_w| \geq t^s_w$ **then**
                foreach *safe* $q_{w'}$ *not marked as distinct from* q_w **do**
                    Call $\texttt{Test}(\hat{S}_w, \hat{S}_{w'}, \delta_{i_w})$;
                    if `Test` *returned* `equal` **then** merge q_w to $q_{w'}$;
                    else if `Test` *returned* `distinct` **then** mark $q_{w'}$ as distinct from q_w;
                    else $t^s_w \leftarrow \alpha \cdot t^s_w$, $i_w \leftarrow i_w + 1$;
                if q_w *is marked distinct from all safes* **then**
                    promote q_w to safe;
                    foreach $\sigma \in \Sigma$ **do**
                        add a candidate $q_{w\sigma}$ to H and let $\hat{S}_{w\sigma} \leftarrow \texttt{Sketch}$;
                        $t^0_{w\sigma} \leftarrow t$, $t^s_{w\sigma} \leftarrow \alpha_0$, $t^u_{w\sigma} \leftarrow t + \beta_0$, $i_{w\sigma} \leftarrow 1$;
    foreach *candidate* q_w **do**
        if $t^u_w \geq t$ **then**
            if $|S_w| < \theta \cdot (t - t^0_w)$ **then** declare q_w insignificant;
            else $t^u_w \leftarrow t^u_w + \beta$;
    if H *has more than* n *safes* **or** *there are no candidates left* **then return** H;

2. *The expected number of elements read from the stream is at most* $O(n^2|\Sigma|^2 N_{\mathrm{u}})$

3. *Each item* x_t *in the stream is processed in* $O(|x_t|^2 T_{\mathrm{s}})$ *amortized time*

4. *If a merge occurred, then at least* N_{e} *elements were read from the stream*

Proof. The memory bound follows from the fact that at any time there will be at most n safe states in the DFA, each with at most $|\Sigma|$ candidates attached, yielding a total of $n(|\Sigma|+1)$ states, each with an associated sketch using memory M_{s}. By assumption, a candidate will be either promoted or merged after collecting N_{u} examples, provided that every safe in the DFA has also collected N_{u} examples. Since this only matters for states with probability at least $\varepsilon/4n|\Sigma|$ because the rest of the will be marked as insignificant (see Lemma 3.4.4), in expectation the algorithm will terminate after reading $\tilde{O}(n^2|\Sigma|^2 N_{\mathrm{u}}/\varepsilon)$ examples from the stream. The time for processing each string depends only on its length: suffixes of string x_t will be inserted into at most $|x_t|+1$ states, at a cost $O(|x_t|T_{\mathrm{s}})$ per insertion, yielding a total processing time of $O(|x_t|^2 T_{\mathrm{s}})$. It remains, though, to amortize the time used by the tests among all the examples processed. Any call to `Test` will take time at most T_{t}. Furthermore, for any candidate, time between successive tests grows exponentially; that is, if t strings have been added to some $\hat{S}_w$, at most $O(\log t)$ tests on $\hat{S}_w$ have taken place due to the scheduling used. Thus, taking into account that each possible candidate may need to be compared to every safe during each testing round, we obtain an expected amortized processing time per string of order $O(|x_t|^2 T_{\mathrm{s}} + n^2|\Sigma|T_{\mathrm{t}}\log(t)/t)$. Finally, note that for a merge to occur necessarily some call to `Test` must return `equal`, which means that at least N_{e} have been read from the stream. □

We want to remark here that Item 3 above is a direct consequence of the scheduling policy used by `StreamLearner` in order to perform similarity tests adaptively. The relevant point is that the ratio between executed tests and processed examples is $O(\log(t)/t) = o(1)$. In fact, by performing tests more often while keeping the tests/examples ratio to $o(1)$, one could obtain an algorithm that converges slightly faster, but has a larger (though still constant with t) amortized processing time per item.

Our next theorem is a PAC-learning result. It says that if the stream is generated by a PDFA then the resulting hypothesis will have small error with high probability when transition probabilities are estimated with enough accuracy. Procedures to perform this estimation have been analyzed in detail in the literature. Furthermore, the adaptation to the streaming setting is straightforward. We use an analysis from [Pal08] in order to prove our theorem.

Theorem 3.4.2. *Suppose we give to* `StreamLearner`$(n', \Sigma, \varepsilon, \delta)$ *a stream generated from a PDFA* D *with* $n \leq n'$ *states. Let* H *denote the DFA returned by* `StreamLearner` *and* $\hat{D}_H$ *a PDFA obtained from* H *by estimating its transition probabilities using* $\tilde{O}(n^4|\Sigma|^4/\varepsilon^3)$ *examples. Then with probability at least* $1-\delta$ *we have* $\mathrm{L}_1(D, \hat{D}_H) \leq \varepsilon$.

The proof of Theorem 3.4.2 is similar in spirit to that of Theorem 1.4.1, and to other learning proofs in [CT04a; PG07; CG08; BCG13]. Therefore, we only discuss in detail those lemmas involved in the proof which are significantly different from the batch and SQ setting. In particular, we focus on the effect of the adaptive test scheduling policy. The rest of the proof is quite standard: first show that the algorithm recovers a transition graph isomorphic to a subgraph of the target containing all relevant states and transitions, and then bound the overall error in terms of the error in transition probabilities (see Section 1.4). We note that by using a slightly different notion of insignificant state and applying a smoothing operation after learning a PDFA, our algorithm could also learn PDFA under the more strict KL divergence.

The next two lemmas establish the correctness of the structure recovered: with high probability, merges and promotions are correct, and no significant candidate state are marked as insignificant.

Lemma 3.4.3. *With probability at least* $1-n(n+1)|\Sigma|\delta'$*, all transitions between safe states are correct.*

Proof. We will inductively bound the error probability of a merge or promotion by assuming that all the previous ones were correct. If all merges and promotions performed so far are correct, there is a transition-preserving bijection between the safe states in H and a subset of states from target A_D;

therefore, for each safe state q_w the distribution of the strings added to $\hat{S}_w$ is the same as the one in the corresponding state in the target. Note that this also holds before the very first merge or promotion.

First we bound the probability that next merge is incorrect. Suppose `StreamLearner` is testing a candidate q_w and a safe $q_{w'}$ such that $D_w \neq D_{w'}$ and decides to merge them. This will only happen if, for some $i \geq 1$ a call $\texttt{Test}(\hat{S}_w, \hat{S}_{w'}, \delta_i)$ returns `equal`. For fixed i this happens with probability at most δ_i; hence, the probability of this happening for some i is at most $\sum_i \delta_i \leq \delta'$. Since there are at most n safes, the probability of next merge being incorrect is at most $n\delta'$.

Next we bound the probability that next promotion is incorrect. Suppose that we promote a candidate q_w to safe but there exists a safe $q_{w'}$ with $D_w = D_{w'}$. In order to promote q_w to a new safe the algorithms needs to certify that q_w is distinct from $q_{w'}$. This will happen if a call $\texttt{Test}(\hat{S}_w, \hat{S}_{w'}, \delta_i)$ return `distinct` for some i. But again, this will happen with probability at most $\sum_i \delta_i \leq \delta'$.

Since a maximum of $n|\Sigma|$ candidates will be processed by the algorithm, the probability of an error in the structure is at most $n(n+1)|\Sigma|\delta'$. □

Following Palmer and Goldberg [PG07], we say a state in a PDFA is *insignificant* if a random string passes through that state with probability less than $\varepsilon/2n|\Sigma|$; the same applies to transitions. It can be proved that a subgraph from a PDFA that contains all its non-insignificant states and transitions fails to accept a set of strings accepted by the original PDFA of total probability at most $\varepsilon/4$.

Lemma 3.4.4. *With probability at least $1 - n|\Sigma|\delta'$ no significant candidate will be marked insignificant and all insignificant candidates with probability less than $\varepsilon/4n|\Sigma|$ will be marked insignificant during its first insignificance test.*

Proof. First note that when insignificance test for q_w is performed, it means that $T_j = (64n|\Sigma|/\varepsilon)\ln(2^j/\delta')$ examples have been processed since its creation, for some $j \geq 1$. Now suppose q_w is a non-insignificant candidate, i.e. it has probability more than $\varepsilon/2n|\Sigma|$. Then, by the Chernoff bounds, we have $|\hat{S}_w|/T_j < 3\varepsilon/8n|\Sigma|$ with probability at most $\delta'/2^j$. Thus, q_w will be marked insignificant with probability at most δ'. On the other hand, if q_w has probability less than $\varepsilon/4n|\Sigma|$, then $|\hat{S}_w|/T_1 > 3\varepsilon/8n|\Sigma|$ happens with probability at most $\delta'/2$. Since there will be at most $n|\Sigma|$ candidates, the statement follows by a union bound. □

Though the algorithm would be equally correct if only a single insignificance test was performed for each candidate state, the scheme followed here ensures the algorithm will terminate even when the distribution generating the stream changes during the execution and some candidate that was significant w.r.t. the previous target is insignificant w.r.t. to the new one.

With the results proved so far we can see that, with probability at least $1-\delta/2$, the set of strings in the support of D not accepted by H have probability at most $\varepsilon/4$ w.r.t. D_T. Together with the guarantees on the probability estimations of $\hat{D}_H$ provided by Palmer [Pal08], we can see that with probability at least $1-\delta$ we have $\mathrm{L}_1(D, \hat{D}_H) \leq \varepsilon$.

Structure inference and probabilities estimation are presented here as two different phases of the learning process for clarity and ease of exposition. However, probabilities could be incrementally estimated during the structure inference phase by counting the number of times each arc is used by the examples we observe in the stream, provided a final probability estimation phase is run to ensure that probabilities estimated for the last added transitions are also correct.

3.5 A Strategy for Searching Parameters

Besides other parameters, a full implementation of `StreamLearner`, `Sketch`, and `Test` as used in the previous section require a user to guess the number of states n and distinguishability μ of the target PDFA in order to learn it properly. These parameters are a priori hard to estimate from a sample of strings. And though in the batch setting a cross-validation-like strategy can be used to select these parameters in a principled way, the panorama in a data streams setting is far more complicated. This is not only because storing a sample to cross-validate the parameters clashes with the data streams paradigm, but also because when the target changes over time the algorithm needs to detect these changes and react

accordingly. Here we focus on a fundamental part of this adaptive behavior: choosing the right n and μ for the current target.

We will give an algorithm capable of finding these parameters by just examining the output of previous calls to `StreamLearner`. The algorithm has to deal with a trade-off between memory growth and time taken to find the correct number of states and distinguishability. This compromise is expressed by a pair of parameters given to the algorithm: $\rho > 1$ and $\phi > 0$. Here we assume that `StreamLearner` receives as input just the current estimations for n and μ. Furthermore, we assume that there exists an unknown fixed PDFA with $n_\star$ states and distinguishability $\mu_\star$ which generates the strings in the stream. The rest of input parameters to `StreamLearner` – Σ, ε, and δ – are considered fixed and ignored hereafter. Our goal is to identify as fast as possible (satisfying some memory constraints) parameters $n \geq n_\star$ and $\mu \leq \mu_\star$, which will allow `StreamLearner` to learn the target accurately with high probability. For the sake of concreteness, and because they are satisfied by all the implementations of `Sketch` and `Test` we have considered, in addition to Assumption 1, we make the following assumptions.

Assumption 2. *Given a distinguishability parameter μ for the target PDFA, algorithms* `Sketch` *and* `Test` *satisfy Assumption 1 with $M_{\mathrm{s}} = \Theta(1/\mu)$ and $N_{\mathrm{e}} = \Theta(1/\mu^2)$.*

Our algorithm is called `ParameterSearch` and is described in Algorithm 3. It consists of an infinite loop where successive calls to `StreamLearner` are performed, each with different parameters n and μ. `ParameterSearch` tries to find the correct target parameters using properties from successive hypothesis produced by `StreamLearner` as a guide. Roughly, the algorithm increments the number of states n if more than n states were discovered in the last run, and decreases distinguishability μ otherwise. However, in order to control the amount of memory used by `ParameterSearch` distinguishability needs to be decreased sometimes even if the last hypothesis' size exceeded n. This is done by imposing an invariant that is maintained throughout the whole execution: $n \leq (1/\mu)^{2\phi}$. This invariant is key to proving the following results.

Theorem 3.5.1. *After each call to* `StreamLearner` *where at least one merge happened, the memory used by* `ParameterSearch` *is $O(t^{1/2+\phi})$, where t denotes the number of examples read from the stream so far.*

Proof. First note that, by the choice of ρ', the invariant $n \leq (1/\mu)^{2\phi}$ is maintained throughout the execution of `ParameterSearch`. Therefore, at all times $n/\mu \leq (1/\mu)^{1+2\phi} \leq (1/\mu^2 + c)^{1/2+\phi}$ for any $c \geq 0$. Suppose that a sequence of $k \geq 1$ calls to `StreamLearner` are made with parameters n_i, μ_i for $i \in [k]$. Write t_i for the number of elements read from the stream during the ith call, and $t = \sum_{i \leq k} t_i$ for the total number of elements read from the stream after the kth call. Now assume a merge occurred in the process of learning the kth hypothesis, thus $t_k = \Omega(1/\mu_k^2)$ by Theorem 3.4.1 and Assumption 2. Therefore we have $t^{1/2+\phi} = (\sum_{i<k} t_i + \Omega(1/\mu_k^2))^{1/2+\phi} = \Omega(n_k/\mu_k)$. Since by Theorem 3.4.1 the memory in use after the kth call to `StreamLearner` is $O(n_k M_{\mathrm{s}}) = O(n_k/\mu_k)$, we are done. □

Note the memory bound does not apply when `StreamLearner` produces tree-shaped hypotheses because in that case the algorithm makes no merges. However, if the target is a not a tree-like PDFA, then merges will always occur. In particular, the bound holds even when the distribution generating the stream does not correspond to a PDFA. On the other hand, if the target happens to be tree-like, our algorithm will learn it quickly (because no merges are needed). A stopping condition for this situation could be easily implemented, thus restricting the amount of memory used by the algorithm in this situation.

The next theorem quantifies the overhead `ParameterSearch` pays for not knowing a priori the parameters of the target. This overhead depends on ρ and ϕ, and introduces a trade-off between memory usage and time until learning.

Theorem 3.5.2. *Assume $\phi < 1/2$. When the stream is generated by a PDFA with $n_\star$ states and distinguishability $\mu_\star$,* `ParameterSearch` *will find a structurally correct hypothesis after making at most $O(\log_\rho(n_\star/\mu_\star^{2\phi}))$ calls to* `StreamLearner` *and reading in expectation at most $\tilde{O}(n_\star^{1/\phi}\rho^{2+1/\phi}/\mu_\star^2)$ elements from the stream.*

Algorithm 3: `ParameterSearch` algorithm

Input: Parameters ρ, ϕ
Data: A stream of strings $x_1, x_2, \ldots \in \Sigma^\star$

$\rho' \leftarrow \rho^{1/2\phi}$;
$n \leftarrow \rho$, $\mu \leftarrow 1/\rho'$;
while true do
  $H \leftarrow$ `StreamLearner`(n, μ);
  if $|H| < n$ **then** $\mu \leftarrow \mu/\rho'$;
  else
    $n \leftarrow n \cdot \rho$;
    if $n > (1/\mu)^{2\phi}$ **then** $\mu \leftarrow \mu/\rho'$;

Proof. For convenience we assume that all $n_\star$ states in the target are important – the same argument works with minor modifications when $n_\star$ denotes the number of important states in the target. By Theorem 3.4.2 the hypothesis will be correct whenever the parameters supplied to `StreamLearner` satisfy $n \geq n_\star$ and $\mu \leq \mu_\star$. If n_{k+1} and μ_{k+1} denote the parameters computed after the kth call to `StreamLearner` we will show that $n_{k+1} \geq n_\star$ and $\mu_{k+1} \leq \mu_\star$ for some $k = O(\log_\rho(n_\star/\mu_\star^{2\phi}))$.

Given a set of k calls to `StreamLearner` let us write $k = k_1 + k_2 + k_3$, where k_1, k_2 and k_3 respectively count the number of times n, μ, or n *and* μ are modified after a call to `StreamLearner`. Now let $k_n = k_1 + k_3$, $k_\mu = k_2 + k_3$. Considering the first k calls to `StreamLearner` in `ParameterSearch` one observes that $n_{k+1} = \rho^{1+k_n}$ and $1/\mu_{k+1} = \rho^{(1+k_\mu)/2\phi}$. Thus, from the invariant $n_{k+1} \leq (1/\mu_{k+1})^{2\phi}$ maintained by `ParameterSearch` (see the proof of Theorem 3.5.1), we see that $k_1 \leq k_2$ must necessarily hold.

Now assume k is the smallest integer such that $n_{k+1} \geq n_\star$ and $\mu_{k+1} \leq \mu_\star$. Then write $k = k' + k''$, where $k' \geq 0$ is the first call to `StreamLearner` after which $\mu_{k'+1} \leq \mu_\star$. By definition of k' we must have $\mu_{k'+1} > \mu_\star/\rho'$, and $1/\mu_{k'+1} = \rho^{(1+k'_\mu)/2\phi}$, hence $k'_\mu < \log_\rho(1/\mu_\star^{2\phi})$. Therefore we see that $k' = k'_1 + k'_2 + k'_3 \leq 2k'_2 + k'_3 \leq 2k'_\mu < 2\log_\rho(1/\mu_\star^{2\phi})$.

Next we observe that if $\mu \leq \mu_\star$ and n are fed to `StreamLearner` then it will return a hypothesis with $|H| \geq n$ whenever $n < n_\star$. Thus we have $k''_2 = 0$ and $k'' = k''_n$; that is, after the first k' calls, n is incremented at each iteration. Therefore we must have $k'' \leq \log_\rho n_\star$, and together with the bound above, we get $k = O(\log_\rho(n_\star/\mu_\star^{2\phi}))$.

It remains to bound the number of examples used in these calls. Note that once the correct μ is found, it will only be further decreased in order to maintain the invariant; hence, $1/\mu_{k+1} \leq \rho^{1/2\phi} \cdot \max\{1/\mu_\star, n_\star^{1/2\phi}\}$. Furthermore, if the correct μ is found before the correct n, the latter will never surpass $n_\star$ by more than ρ. However, it could happen that n grows more than than really needed when $\mu > \mu_\star$; in this case the invariant will keep μ decreasing. Therefore, in the end $n_{k+1} \leq \rho \cdot \max\{n_\star, 1/\mu_\star^{2\phi}\}$. Note that since $\phi < 1/2$ we have $\max\{1/\mu_\star, n_\star^{1/2\phi}\} \cdot \max\{n_\star, 1/\mu_\star^{2\phi}\} = O(n_\star^{1/2\phi}/\mu_\star)$. Thus, by Theorem 3.4.1 we see that in expectation `ParameterSearch` will read at most $O(n_{k+1}^2 N_\mathrm{u} \log_\rho(n_\star/\mu_\star^{2\phi})) = \tilde{O}(n_\star^{1/\phi}\rho^{2+1/\phi}/\mu_\star^2)$ elements from the stream. □

Note the trade-off in the choice of ϕ: small values guarantee little memory usage while potentially increasing the time until learning.

3.6 Detecting Changes in the Target

Now we describe a change detector `ChangeDetector` for the task of learning distributions from streams of strings using PDFA. Our detector receives as input a DFA H, a change threshold γ, and a confidence parameter δ. It then runs until a change relevant w.r.t. the structure of H is observed in the stream distribution. This DFA is *not* required to model the structure of current distribution, though the more

accurate it is, the more sensitive to changes will the detector be. In the application we have in mind, the DFA will be a hypothesis produced by `StreamLearner`.

We will use notation from Section 1.1 and define some new one. Given a DFA H and a distribution D on $\Sigma^\star$, $D(H[q]\Sigma^\star)$ is the probability of all words visiting state q at least once. Given a sample from D, we denote by $\hat{D}(H[q]\Sigma^\star)$ the relative frequency of words passing through q in the sample. Furthermore, we denote by $\mathbf{D}$ the vector containing $D(H[q]\Sigma^\star)$ for all states q in H, and by $\hat{\mathbf{D}}$ the vector of empirical estimations.

Our detector will make successive estimations $\hat{\mathbf{D}}_0$, $\hat{\mathbf{D}}_1$, ... and decide that a change happened if some of these estimations differ too much. The rationale behind this approach to change detection is justified by next lemma, showing that a non-negligible difference between $D(H[q]\Sigma^\star)$ and $D'(H[q]\Sigma^\star)$ implies a non-negligible distance between D and D'. A converse to this lemma would mean that our detector can detect all possible kinds of change, though in general we believe that to be false; that is, that one can find distributions which are indistinguishable with respect to some DFA though they are far apart in the total variation distance.

Lemma 3.6.1. *If for some state $q \in H$ we have $|D(H[q]\Sigma^\star) - D'(H[q]\Sigma^\star)| > \gamma$, then $\mathrm{L}_1(D, D') > \gamma$.*

Proof. First note that $\mathrm{L}_1(D, D') \geq \sum_{x \in H[q], y \in \Sigma^\star} |D(xy) - D'(xy)|$. Then, by triangle inequality this is at most $|\sum_{x \in H[q], y \in \Sigma^\star} D(xy) - D'(xy)| = |D(H[q]\Sigma^\star) - D'(H[q]\Sigma^\star)| > \gamma$. □

More precisely, our change detector `ChangeDetector` works as follows. It begins by reading the first $m = (8/\gamma^2)\ln(4|H|/\delta)$ examples from the stream and uses them to estimate $\hat{\mathbf{D}}_0$ in H. Then, for $i > 0$, it makes successive estimations $\hat{\mathbf{D}}_i$ using $m_i = (8/\gamma^2)\ln(2\pi^2 i^2 |H|/3\delta)$ examples, until $\|\hat{\mathbf{D}}_0 - \hat{\mathbf{D}}_i\|_\infty > \gamma/2$, at which point a change is detected.

We will bound the probability of false positives and false negatives in `ChangeDetector`. For simplicity, we assume the elements on the stream are generated by a succession of distributions $D_0, D_1, \ldots$ with changes taking place only between successive estimations $\hat{\mathbf{D}}_i$ of state probabilities. The first lemma is about false positives.

Lemma 3.6.2. *If $D = D_i$ for all $i \geq 0$, with probability at least $1 - \delta$ no change will be detected.*

Proof. We will consider the case with a single state q; the general case follows from a simple union bound. Let $p = D(H[q]\Sigma^\star)$. We denote by $\hat{p}_0$ an estimation of p obtained with $(8/\gamma^2)\ln(4/\delta)$ examples, and by $\hat{p}_i$ an estimation from $(8/\gamma^2)\ln(2\pi^2 i^2/3\delta)$ examples. Recall that $p = \mathbb{E}[\hat{p}_0] = \mathbb{E}[\hat{p}_i]$. Now, change will be detected if for some $i > 0$ one gets $|\hat{p}_0 - \hat{p}_i| > \gamma/2$. If this happens, then necessarily either $|\hat{p}_0 - p| > \gamma/4$ or $|\hat{p}_i - p| > \gamma/4$ for that particular i. Thus, by Hoeffding's inequality, the probability of a false positive is at most $\mathbb{P}[|\hat{p}_0 - p| > \gamma/4] + \sum_{i>0} \mathbb{P}[|\hat{p}_i - p| > \gamma/4] \leq \delta$. □

Next we consider the possibility that a change occurs but is not detected.

Lemma 3.6.3. *If $D = D_i$ for all $i < k$ and $|D_{k-1}(H[q]\Sigma^\star) - D_k(H[q]\Sigma^\star)| > \gamma$ for some $q \in H$, then with probability at least $1 - \delta$ a change will be detected. Furthermore, if the change occurs at time t, then it is detected after reading at most $O(1/\gamma^2 \ln(\gamma t/\delta))$ examples more.*

Proof. As in Lemma 3.6.2, we prove it for the case with a single state. The same notation is also used, with $p_k = D_k(H[q]\Sigma^\star)$. The statement will not be satisfied if a change is detected before the kth estimation or no change is detected immediately after it. Let us assume that $|\hat{p}_0 - p| \leq \gamma/4$. Then, a false positive will occur if $|\hat{p}_0 - \hat{p}_i| > \gamma/2$ for some $i < k$, which by our assumption implies $|\hat{p}_i - p| > \gamma/4$. On the other hand, a false negative will occur if $|\hat{p}_0 - \hat{p}_k| \leq \gamma/2$. Since $|p - p_k| > \gamma$, our assumption implies that necessarily $|\hat{p}_k - p_k| > \gamma/4$. Therefore, the probability that the statement fails can be bounded by

$$\mathbb{P}[|\hat{p}_0 - p| > \gamma/4] + \sum_{0<i<k} \mathbb{P}[|\hat{p}_i - p| > \gamma/4] + \mathbb{P}[|\hat{p}_k - p_k| > \gamma/4] \ ,$$

which by Hoeffding's inequality is at most δ.

Now assume the change happened at time t. By Stirling's approximation we have

$$t = \Theta\left(\sum_{i \le k} \frac{1}{\gamma^2} \ln \frac{k^2}{\delta}\right) = \Theta\left(\frac{k}{\gamma^2} \ln \frac{k}{\delta}\right) \quad .$$

Therefore $k = O(\gamma^2 t)$. If the change is detected at the end of the following window (which will happen with probability at least $1 - \delta$), then the response time is at most $O(1/\gamma^2 \ln(\gamma t/\delta))$. □

Chapter 4

State-Merging on Generalized Alphabets

In this chapter we generalize previous state-merging algorithms for learning distributions over $(\Sigma \times \mathcal{X})^\star$, where Σ is again a finite alphabet, but $\mathcal{X}$ is an arbitrary measure space. To model such distributions we introduce a new type of automata which we call Generalized PDFA (GPDFA). In such an automaton, the transition between states is deterministic in terms of Σ, but does not depend on the particular elements from $\mathcal{X}$ generated alongside the elements of Σ. It is this particular restriction what helps us generalize state-merging algorithms to this new model. For example, taking $\mathcal{X} = \mathbb{R}$, GPDFA can mode continuous-time markov processes with discrete observations from Σ which are observed for some random period of time. Thus, we get a string of observations of the form (σ, t), where t is period of time during which symbol σ was observed.

Like in previous chapters, we analyze the sample and time complexity of state-merging algorithms for learning GPDFA in the PAC learning model. In particular, we show what conditions are required on $\mathcal{X}$ in order for such learning to be possible and efficient. Another difference with previous analyses given in this thesis is that here we give learning results in terms of relative entropy instead of total variation distance. These are much stronger that what we have considered before. Thus, the proofs follow a slightly different approach, closer in some sense to the original PAC analysis in [CT04a].

An interesting feature of GPDFA is that they encompass seamlessly several useful models which may seem a priori unrelated. In the last section of this chapter we shall give two important applications of GDPFA: one to Markov models with timed transitions as mentioned above, and another to a particular class of transducers from $\Sigma^\star$ to $\Delta^\star$. To the best of our knowledge, these are the first PAC results for learning such models in terms of relative entropy.

4.1 PDFA over Generalized Alphabets

Our model for generalized PDFA is a FSM for defining probability distributions over the free monoid $(\Sigma \times \mathcal{X})^\star$, where Σ is a finite alphabet and $\mathcal{X}$ is some arbitrary measure space. We will discuss the learnability of such objects in a rather general setting. However, we will ignore most measure-theoretic issues that may arise in our definitions. We do so in purpose, and for two reasons: first, because we do not want such issues to blur the general ideas behind the model and the learning algorithm; and second, because despite the general treatment we give here, the two particular examples we give in Section 4.6 do not suffer from these measure-theoretic complications. That said, we can proceed with our main definition.

A *Generalized PDFA* (GPDFA) consists of a tuple $A = \langle Q, \Sigma, \tau, \gamma, q_0, \xi, \mathcal{D} \rangle$, where: Q is a finite set of states; Σ is a finite alphabet; $q_0 \in Q$ is the initial state; ξ is a symbol not in Σ used to denote the end of a string, and we will write $\Sigma' = \Sigma \cup \{\xi\}$ for convenience; $\tau : Q \times \Sigma \to Q \cup \{\perp\}$ is the transition (partial) function; $\gamma : Q \times \Sigma' \to [0, 1]$ is a transition probability function such that $\sum_{\sigma \in \Sigma'} \gamma(q, \sigma) = 1$ for all

$q \in Q$, and satisfying: $\tau(q,\sigma) = \perp \rightarrow \gamma(q,\sigma) = 0$ for all $\sigma \in \Sigma$; $\mathcal{D} : Q \times \Sigma' \rightarrow \mathcal{M}_1^+(\mathcal{X}) \cup \{\perp\}$ is a partial function that assigns a probability distribution $\mathcal{D}(q,\sigma) = D_{q,\sigma}$ over $\mathcal{X}$ to every defined transition event. Sometimes we shall abuse this notation and write $\mathcal{D}$ for the set $\mathcal{D}(Q \times \Sigma')$ of all distributions over $\mathcal{X}$ appearing in transitions from A. It is immediate to note that the tuple $\langle Q, \Sigma, \tau, \gamma, q_0, \xi \rangle$ actually defines a PDFA, where instead of stopping probabilities like in Chapter 1, we have now an end-of-string symbol ξ which is observed whenever a string is terminated. Extensions $\tau : Q \times \Sigma^\star \rightarrow Q$ and $\gamma : Q \times (\Sigma')^\star \rightarrow [0,1]$ are defined in the usual way.

The process by which a GPDFA generates an observation is very similar to the corresponding process for standard PDFA. Starting from state q_0, at each step in the generation process the GPDFA is in some state $q \in Q$. At every step, a symbol $\sigma \in \Sigma'$ is drawn according to the probability distribution $\gamma(q, \bullet)$ and then an element from $\mathcal{X}$ is drawn from distribution $D_{q,\sigma}$. Then, the process stops if $\sigma = \xi$; otherwise the next state is set to $\tau(q,\sigma)$. The output of the process is a string $z \in (\Sigma \times \mathcal{X})^\star \times (\{\xi\} \times \mathcal{X})$ which we write as $z = (x\xi, y)$ with $x = x_1 \cdots x_t \in \Sigma^t$ and $y = y_1 \cdots y_{t+1} \in \mathcal{X}^{t+1}$.

Recall that PDFA define a probability distribution if and only if for any state $q \in Q$ there is a non-negative probability of reaching some state q' such that $\gamma(q', \xi) > 0$; we shall assume that this always holds for the GPDFA we consider. For convenience we may sometimes say that GPDFA define probability distributions over $(\Sigma' \times \mathcal{X})^\star$, though it will always be implicitly assumed that any string outside $(\Sigma \times \mathcal{X})^\star \times (\{\xi\} \times \mathcal{X})$ has probability zero. This formulation contrasts a little bit with the PDFA defined in Chapter 1 which did not use a symbol to denote the end of a string. In the present formulation we allow the GPDFA to generate a last element from $\mathcal{X}$ during the "stop" event. As such, this formulation is slightly more general than one which only defines probability distributions over $(\Sigma \times \mathcal{X})^\star$. Furthermore, the introduction of an end-of-string maker will be convenient to bound the relative entropy between a target and a hypothesis GPDFA as will become clear in the following sections. We shall write f_A to denote the probability distribution defined by GPDFA A. Like we did for PDFA, we let A_q denote the GPDFA obtained from A by taking $q \in Q$ as a starting state.

When a GPDFA generates a string starting from some state q, we can obtain a marginal distribution over $\Sigma' \times \mathcal{X}$ by just looking at the first symbol in the sequence. We call such a distribution a *local distribution*. In particular, for $q \in Q$ we use D_q to denote the associated local distribution. Thus, if $E \subseteq \Sigma' \times \mathcal{X}$ is an event whose fibers on $\mathcal{X}$ with respect to Σ' are denoted by E_σ, then we have

$$D_q(E) = \sum_{\sigma \in \Sigma'} \gamma(q,\sigma) D_{q,\sigma}(E_\sigma) \ .$$

We note here that we can think of $D_{q,\bullet}$ as a function from Σ' to the space $\mathcal{M}_1^+(\mathcal{X})$ of probability distributions over $\mathcal{X}$; this object is sometimes called a *stochastic kernel* in measure-theoretic probability [Pol03].

When the goal is to learn GPDFA it may be convenient to restrict the possible kernels that may arise from in $\mathcal{D}$. In general we may require that all distributions in $\mathcal{D}$ are absolutely continuous with respect to some measure $\mathfrak{m}$ on $\mathcal{X}$. Sometimes we may require more stringent constraints, like that all distributions over $\mathcal{X}$ belong to a particular parametric family. For example, when $\mathcal{X} = \mathbb{R}$ and $\mathfrak{m}$ is the Lebesgue measure, we may restrict the distributions in $\mathcal{D}$ to belong to the exponential family, or even to be exponential or normal distributions. These and other examples will be examined in detail in Section 4.5.

Any GPDFA has an underlying deterministic automaton the same way a PDFA does. Since this deterministic component is the key to the state-merging paradigm for learning such objects, we want to study to what extent can this paradigm be extended to learn families of GPDFA. Intuitions built up through the study of state merging methods in previous chapters steadily point to two sub-problems that need to be addressed by these hypothetical extensions: first, distinguishing distributions over $(\Sigma \times \mathcal{X})^\star$; and second, learning the local distributions of A accurately. In the following sections we will study these two problems in detail. As before, our focus will be on provably correct solutions that require only samples of polynomial size.

4.2 Generic Learning Algorithm for GPDFA

In this section we describe a state-merging algorithm for learning GPDFA over an arbitrary $\mathcal{X}$. The algorithm will make use of two subroutines that depend on the particular choice of $\mathcal{X}$; these will be described in detail later. For now it will suffice to assume that we have at our disposal the following two procedures:

- A procedure `Test` that given two samples from $(\Sigma' \times \mathcal{X})^\star$ determines if they certainly come from two distinct distributions – in which case returns `distinct` – or not – in which case returns `not clear`.
- A procedure `Estimate` that given a sample from $\Sigma' \times \mathcal{X}$ returns an estimate $\hat{D}$ of its distribution.

Our main working assumption, which we shall formalize later, is that when given large enough samples, `Test` and `Estimate` will return correct answers with high probability.

Now we describe the algorithm in detail. Pseudocode can be found in Algorithm 4. As input the algorithm takes a finite alphabet Σ, an upper bound on the number of states n, and a confidence parameter δ. The algorithm also receives as input a sample S containing m strings from $(\Sigma \times \mathcal{X})^\star \times (\{\xi\} \times \mathcal{X})$. Basically, the algorithm works in two stages: in the first stage it builds a DFA, where each state q has an associated sample S_q from $(\Sigma \times \mathcal{X})^\star \times (\{\xi\} \times \mathcal{X})$; in the second stage the information from samples S_q is used to estimate local distributions for each state in the DFA and convert it into a GPDFA. While building the DFA states are separated into two categories: safe and candidate. Safe states correspond to states the algorithm confidently believes that belong to the target GPDFA. Candidates can become new safes at some point or be merged to already existing safes. In addition, candidates have no outgoing transitions. When a state is declared safe, its associated sample will be fixed and never modified again. The process begins with a simple DFA with a single safe q_λ with sample $S_{q_\lambda} = S$, and candidates q_σ for each $\sigma \in \Sigma$; we also add transitions $\tau(q_\lambda, \sigma) = q_\sigma$. Then the process executes a `while`loop which for each iteration is guaranteed to either promote one candidate to safe or merge one candidate to an existing safe. This will go on until the DFA is complete or it contains more than n safes.

The first step in each iteration is to clear the samples associated with each candidate and refill them again; this ensures the samples associated with each candidate adequately reflect the current structure of the DFA. To do so we *parse* each string in the sample S until we hit a candidate, and then add the remaining suffix to the sample of that candidate. More formally, if (x, y) is an example in S, we seek, if it exists, a prefix $u \sqsubseteq_0 x$ such that $q_u = \tau(q_\lambda, u)$ is a candidate. In this case, we add $(x_{|u|+1:|x|}, y_{|u|+1:|y|})$ to the sample S_{q_u}; otherwise the example is discarded. Once all examples from S have been parsed, we select a candidate q maximizing $|S_q|$. For this candidate, we perform a series of tests to determine if the sample S_q comes from the same distribution as any of the samples associated to the existing safes. If we find no matches, q is promoted to be a safe and new candidates $\tau(q, \sigma)$ are added for each $\sigma \in \Sigma$. Otherwise, q is merged to a matching safe q'; this means that q is erased from the DFA, and the incoming transition into q is now incoming into q'. After the `while`loop terminates, a postprocessing stage is conducted in case the DFA is not complete. In this stage we add a new state q_∞ with empty multiset and make sure all missing transitions in the DFA are directed towards q_∞.

In the second part of the algorithm we use the sample S_q for each state q to estimate a local distribution over $\Sigma' \times \mathcal{X}$. We do so by applying procedure `Estimate` to local(S_q) which returns a sample over $\Sigma' \times \mathcal{X}$ from a sample over $(\Sigma' \times \mathcal{X})^\star$ by keeping only the first $z_1 = (x_1, y_1)$ symbol from every string $z = (x, y) \in S_q$. This distribution is then used in the obvious way to define a GPDFA using the DFA constructed by the previous part as its transition structure.

In the following section we will analyze the statistical aspects of Algorithm 4. Its running time is given in the next result, where the running times of `Test`and `Estimate` are considered variables. In particular, we suppose a call `Test`(S, S', δ) takes time $T_\mathrm{t} = T_\mathrm{t}(|S|, |S'|, \delta)$, and a call to `Estimate`$(S)$ takes time $T_\mathrm{e} = T_\mathrm{e}(|S|)$. Furthermore, given a sample $S = (z^1, \ldots, z^m)$ from $(\Sigma' \times \mathcal{X})^\star$, we use $\|S\|$ to denote the sum of the length of the strings in S; that is, $\|S\| = \sum_{i=1}^m |x^i|$, where $z^i = (x^i, y^i)$.

Theorem 4.2.1. *The running time of Algorithm 4 is bounded by* $O(n|\Sigma|\|S\| + n^2|\Sigma|T_\mathrm{t} + nT_\mathrm{e})$.

Algorithm 4: Learn GPDFA

Input: parameters n, Σ, δ
Data: sample $S = (z^1, \ldots, z^m)$
Output: hypothesis GPDFA H

initialize H with safe q_λ and let $S_{q_\lambda} \leftarrow S$;
foreach $\sigma \in \Sigma$ **do** add a candidate to H reached from q_λ with σ;
while *H is not complete* **and** *H has n or less safes* **do**
    clear multisets of all candidates in H;
    foreach $z = (x, y) \in S$ **do**
        if *there exists $u \sqsubseteq_0 x$ such that q_u is a candidate of H* **then**
            add $(x_{|u|+1} \cdots x_{|x|}, y_{|u|+1} \cdots y_{|y|})$ to S_{q_u};
    let q be the candidate in H with largest $|S_q|$;
    foreach *safe q' in H* **do** run `Test`$(S_q, S_{q'}, \delta)$;
    if *all calls to* `Test` *returned* `distinct` **then**
        promote q to safe with multiset S_q;
        foreach $\sigma \in \Sigma$ **do** add a candidate to H reached from q with σ;
    else
        select a candidate q' for which `Test` returned `not clear`;
        merge q to q' in H;
if *H still contains candidate states* **then**
    create a ground state q_∞ and let $S_{q_\infty} \leftarrow \varnothing$;
    merge all candidates in H to q_∞;
    foreach $\sigma \in \Sigma$ **do** add a transition from q_∞ to itself labeled with σ;
foreach *state q in H* **do**
    compute $\hat{D}_q \leftarrow$ `Estimate`(local(S_q));
    let $\hat{D}_q$ be the local distribution of q in H;

Proof. As a first observation note that since a GPDFA with n states over Σ can have at most $n|\Sigma|$ transitions between its states. Since each iteration of the `while`loop adds one transition between two safe states to the DFA, the algorithm will runs for at most $O(n|\Sigma|)$ iterations. Each of these iterations parses the full sample S, which takes time $O(\|S\|)$, and performs as many calls to `Test` as safes in the current DFA, of which there are at most n. Thus, the first part of the algorithm takes time $O(n|\Sigma|(\|S\| + nT_{\mathrm{t}}))$. In the second part we make a call to `Estimate` for each state in the DFA which gives us an additional running time of order $O(nT_{\mathrm{e}})$. □

4.3 PAC Analysis

Now we proceed to analyze Algorithm 4. We will give PAC learning guarantees under some assumptions on the `Test` and `Estimate` subroutines. Our analysis generalizes that of [CT04a] and follows a similar approach. First we give an expression for the relative entropy between two GPDFA in terms of their local distributions. Based on this formula, we show that identifying the frequent states and transitions in a target GPDFA is enough to correctly approximate its probability distribution. Finally we show that if the input sample is large enough, then Algorithm 4 will construct a hypothesis whose transition graph contains these frequent states and transitions.

4.3.1 Relative Entropy between GPDFA

We begin by defining some concepts related to the frequency with which states are visited in a GPDFA. Most of these are analogous to those defined in [CT04a] for PDFA. Let $A = \langle Q, \Sigma, \tau, \gamma, q_0, \xi, \mathcal{D}\rangle$ be a GPDFA over Σ. Recall from Chapter 1 the notations

$$A[\![q]\!] = \{x | \tau(q_0, x) = q\} \ ,$$
$$A[q] = \{x | \tau(q_0, x) = q \wedge \forall y, z : x = yz \rightarrow (|z| = 0 \vee \tau(q_0, y) \neq q)\} \ ,$$

for the sets of words that reach a state q and reach a state q for the first time, respectively. With this notation we can define the *weight* of a state $q \in Q$ as

$$W(q) = \sum_{x \in A[\![q]\!]} \gamma(q_0, x) \ ,$$

which corresponds to the expected number of times state q will be visited during the generation of a string by A. We also defined the *probability* of a state $q \in Q$ as

$$P(q) = \sum_{x \in A[q]} \gamma(q_0, x) \ ,$$

which corresponds to the probability that A will be in state q during the generation of a string. Recall from [CT04a] that for any $q \in Q$, the expected length of strings generated from q is given by

$$L(q) = \sum_{x \in \Sigma^\star} |x| \gamma(q, x\xi) = \sum_{x \in \Sigma^\star} \gamma(q, x) - 1 \ .$$

The following relations between $W(q)$, $P(q)$, and $L(q)$ were established in [CT04a] for PDFA. Since the definitions of weight and probability of states rely only on the underlying PDFA of a GPDFA, they hold as well for GPDFA.

Lemma 4.3.1. *For any $q \in Q$ we have $W(q) \leq (L(q) + 1)P(q)$. Furthermore, $\sum_{q \in Q} W(q) \leq L(q_0) + 1$.*

Let $A' = \langle Q', \Sigma, \tau', \gamma', q'_0, \xi, \mathcal{D}'\rangle$ be another GPDFA over Σ. We can define joint weights for pairs of states $q \in Q$ and $q' \in Q'$ by coupling the path taken by both automata:

$$W(q, q') = \sum_{x \in A[\![q]\!] \cap A'[\![q']\!]} \gamma(q_0, x) \ .$$

This weight computes the expected number of times we will find A in state q and A' in state q' in the process where strings are generated by A and parsed by A'. This quantity will play a central role in the expression for the relative entropy between A and A'. The intuition behind this fact is that if the transition structure of A and A' is very similar, for each state q in A will exist a state q' of A' such that $W(q, q')$ is close to $W(q)$ and $W(q, q'')$ is very small for any other $q'' \neq q'$. In fact, it is immediate to see that we have

$$\sum_{q' \in Q'} W(q, q') \leq W(q) \ , \tag{4.1}$$

with equality whenever the transition structure of A' is complete; i.e. the transition function τ' is total.

Now we are ready to state the main result of this section. The following theorem gives an expression for the relative entropy – also known as Kullback–Leibler divergence – between two GPDFA which generalizes the one given in [Car97] for PDFA. The proof relies on several measure-theoretic observations and is given in Section 4.7.

Theorem 4.3.2. *Let A and A' be GPDFA over Σ. The relative entropy between distributions f_A and $f_{A'}$ is given by the following expression:*

$$\mathrm{KL}(f_A \| f_{A'}) = \sum_{q \in Q} \sum_{q' \in Q'} W(q, q') \, \mathrm{KL}(D_q \| D_{q'}) \ ,$$

where

$$\mathrm{KL}(D_q \| D_{q'}) = \sum_{\sigma \in \Sigma'} \gamma(q, \sigma) \left[\log \frac{\gamma(q, \sigma)}{\gamma'(q', \sigma)} + \mathrm{KL}(D_{q,\sigma} \| D'_{q',\sigma}) \right] \ .$$

We note that in this chapter we use the relative entropy defined in terms of the *natural logarithm* for convenience. The same learning results can be obtained with the logarithm in base 2 by adjusting a constant term in our bounds.

4.3.2 Frequent Core of a GPDFA

Now we prove the main technical result that we will use to show that Algorithm 4 is a PAC learner for GPDFA with respect to the Kullback–Leibler divergence. Roughly speaking, the result will say that if we can produce a hypothesis that contains a subgraph isomorphic to that formed by all *frequent* states and transitions of the target GPDFA, then the joint weights $W(q, q')$ in the formula from Theorem 4.3.2 will be small whenever q is not frequent or q is frequent but q' is not the image of q in the hypothesis. The particular form of our result yields a slight improvement over Lemma 13 from [CT04a]. Since the proof published there is not correct and no erratum has been published so far [Cla09], we include a new full proof for completeness.

Let $A = \langle Q, \Sigma, \tau, \gamma, q_0, \xi, \mathcal{D} \rangle$ be a GPDFA over Σ. Given some $\varepsilon_{\mathrm{s}} > 0$ we define the set of *frequent states* in A as

$$F_{\mathrm{s}} = \{ q \in Q \mid W(q) > \varepsilon_{\mathrm{s}} \} \ .$$

Furthermore, given $\varepsilon_{\mathrm{t}} > 0$ we define the set of *frequent transitions* in A as

$$F_{\mathrm{t}} = \{ (q, \sigma) \in Q \times \Sigma \mid q \in F_{\mathrm{s}} \wedge \gamma(q, \sigma) > \varepsilon_{\mathrm{t}} \} \ .$$

The *frequent core* of A is the subgraph of the underlying DFA of A containing all frequent states and transitions. The set of *frequent words* of A is defined as

$$F_{\mathrm{w}} = \{ x \in \Sigma^\star \mid \forall y \sqsubseteq_0 x \ \ \tau(q_0, y) \in F_{\mathrm{s}} \wedge \forall y\sigma \sqsubseteq_0 x \ \ (\tau(q_0, y), \sigma) \in F_{\mathrm{t}} \} \ .$$

These are all the strings that can be recognized by the frequent core of A.

Now we consider $\bar{F}_{\mathrm{s}}$, the complementary of F_{w}, formed by all strings $x = x_1 \cdots x_t$ such that the state path $q_0, \tau(q_0, x_1), \ldots, \tau(q_0, x_1 \cdots x_t)$ leaves the frequent core of A at some point. By definition, we can write

$$\bar{F}_{\mathrm{s}} = \{ x \in \Sigma^\star \mid \exists y \sqsubseteq_0 x \ \ \tau(q_0, y) \notin F_{\mathrm{s}} \vee \exists y\sigma \sqsubseteq_0 x \ \ (\tau(q_0, y), \sigma) \notin F_{\mathrm{t}} \} \ .$$

If follows immediately from this definition that if $x \in \bar{F}_\mathrm{s}$, then $x\Sigma^\star \subseteq \bar{F}_\mathrm{s}$; that is, if x leaves the frequent core, any extension of x leaves the frequent core as well. Using this property, we can characterize the structure of $\bar{F}_\mathrm{s}$ as follows.

Lemma 4.3.3. *There exists a prefix-free set U such that $\bar{F}_\mathrm{s} = \cup_{x\in U} x\Sigma^\star$.*

Proof. Let $U_0 = \varnothing$ and $V_0 = \bar{F}_\mathrm{s}$. For $i \geq 0$ we define inductively $U_{i+1} = U_i \cup \{x_i\}$ and $V_{i+1} = V_i \setminus x_i\Sigma^\star$, where x_i is the smallest word in V_i in lexicographical order. Then we take $U = \cup_{i\geq 0} U_i$. By construction and our previous observation we have $U \subset \bar{F}_\mathrm{s}$ and $\cup_{x\in U} x\Sigma^\star \subseteq \bar{F}_\mathrm{s}$. To see that $\bar{F}_\mathrm{s} \subseteq \cup_{x\in U} x\Sigma^\star$ take any $y \in \bar{F}_\mathrm{s}$ and observe that $\bar{F}_\mathrm{s}$ contains at most $k = \sum_{i=0}^{|y|} |\Sigma|^i$ strings less or equal than y in lexicographical order, which implies that $y \in \cup_{x\in U_k} x\Sigma^\star$. Finally, observe that U_0 is prefix-free and suppose U_i too. Then, since no word in V_i has a prefix in U_i by construction, then U_{i+1} is also prefix-free, and so is U. □

Given set U from previous lemma, it is easy to see that for any $x \in U$ all of its proper prefixes belong to F_w. Thus, for any $x \in U$, it must be the case that either $\tau(q_0, x) \notin F_\mathrm{s}$ or $x = y\sigma$ and $(\tau(q_0, y), \sigma) \notin F_\mathrm{t}$. Accordingly, we define the following two sets:

$$U_\mathrm{s} = \{x \in U \mid \tau(q_0, x) \notin F_\mathrm{s}\} \ , \tag{4.2}$$

$$U_\mathrm{t} = \{x \in U \mid x = y\sigma \wedge (\tau(q_0, y), \sigma) \notin F_\mathrm{t}\} \ . \tag{4.3}$$

Note that by construction we have $U = U_\mathrm{s} \cup U_\mathrm{t}$.

Now let $A' = \langle Q', \Sigma, \tau', \gamma', q'_0, \xi, \mathcal{D}'\rangle$ be another GPDFA over the same alphabet Σ. We assume that A' has a subgraph isomorphic to a subgraph of A containing its frequent core. More formally, suppose there exist $Q_\mathrm{f} \subseteq Q$, $Q'_\mathrm{f} \subseteq Q'$, and a bijection $\Phi : Q_\mathrm{f} \to Q'_\mathrm{f}$ satisfying the following:

1. $q_0 \in Q_\mathrm{f}$, $q'_0 \in Q'_\mathrm{f}$, and $\Phi(q_0) = q'_0$,
2. if $q' \in Q'_\mathrm{f}$ and $\tau'(q', \sigma) \in Q'_\mathrm{f}$, then $\tau(\Phi^{-1}(q'), \sigma) = \Phi^{-1}(\tau'(q', \sigma))$,
3. $F_\mathrm{s} \subseteq Q_\mathrm{f}$,
4. if $(q, \sigma) \in F_\mathrm{t}$, then $\tau(q, \sigma) \in Q_\mathrm{f}$ and $\tau'(\Phi(q), \sigma) = \Phi(\tau(q, \sigma))$.

Note that the first two conditions enforce that Φ is a proper isomorphism on the graph structure and the other two that the isomorphism is defined over the frequent core of A. Whenever A' satisfies these conditions we will say that A' *captures the frequent core* of A.

Theorem 4.3.4. *Let A be a GPDFA with n states and $L(q) \leq L$ for all $q \in Q$. Suppose A' captures the frequent core of A for some ε_s and ε_t. Then the following holds:*

$$\sum_{q\in F_\mathrm{s}} \sum_{q'\in Q'\setminus\Phi(q)} W(q, q') \leq n(L+1)\varepsilon_\mathrm{s} + (L+1)^2\varepsilon_\mathrm{t} \ .$$

Proof. We start by observing that if $x \in F_\mathrm{w}$ then $\tau(q_0, x) = q \in F_\mathrm{s}$ and $\tau'(q'_0, x) = \Phi(q)$, where the second fact holds because A' captures the frequent core of A. Considering the contrapositive one can show the following inequality:

$$\begin{aligned}\sum_{q\in F_\mathrm{s}} \sum_{q'\in Q'\setminus\Phi(q)} W(q, q') &= \sum_{q\in F_\mathrm{s}} \sum_{q'\in Q'\setminus\Phi(q)} \sum_{x\in A[\![q]\!]\cap A'[\![q']\!]} \gamma(q_0, x) \\ &\leq \sum_{x\in\bar{F}_\mathrm{s}} \gamma(q_0, x) \ .\end{aligned}$$

Now, by Lemma 4.3.3 and the definition of $L(q)$ we have

$$\begin{aligned}\sum_{x\in\bar{F}_\mathrm{s}} \gamma(q_0, x) &= \sum_{x\in U} \sum_{y\in\Sigma^\star} \gamma(q_0, x)\gamma(\tau(q_0, x), y) \\ &= \sum_{x\in U} \gamma(q_0, x)(L(\tau(q_0, x)) + 1) \\ &\leq (L+1) \sum_{x\in U} \gamma(q_0, x) \ .\end{aligned}$$

Next we split the sum over U in two terms using $U = U_{\mathrm{s}} \cup U_{\mathrm{t}}$ and bound the resulting terms as follows. First, using (4.2) and the definition of F_{s} we have

$$\begin{aligned}\sum_{x \in U_{\mathrm{s}}} \gamma(q_0, x) &= \sum_{q \in Q} \sum_{x \in U_{\mathrm{s}} \cap A[\![q]\!]} \gamma(q_0, x) \\ &= \sum_{q \in \bar{F}_{\mathrm{s}}} \sum_{x \in U_{\mathrm{s}} \cap A[\![q]\!]} \gamma(q_0, x) \\ &\leq \sum_{q \in \bar{F}_{\mathrm{s}}} W(q) \\ &\leq n\varepsilon_{\mathrm{s}} \ .\end{aligned}$$

Then, using (4.3) we get

$$\begin{aligned}\sum_{y\sigma \in U_{\mathrm{t}}} \gamma(q_0, y\sigma) &= \sum_{y\sigma \in U_{\mathrm{t}}} \gamma(q_0, y)\gamma(\tau(q_0, y), \sigma) \\ &\leq \varepsilon_{\mathrm{t}} \sum_{y\sigma \in U_{\mathrm{t}}} \gamma(q_0, y) \\ &\leq \varepsilon_{\mathrm{t}}(L(q_0) + 1) \ .\end{aligned}$$

Applying $L(q_0) \leq L$ to this last bound yields the desired inequality. □

4.3.3 Identifying the Frequent Core

Now we will proof that when the sample S given to Algorithm 4 is large enough, with high probability the hypothesis GPDFA H captures the frequent core of the target A. To prove this result we will need to make the following assumption on the function `Test`.

Assumption 1. *Function* $\texttt{Test}(S_q, S_{q'}, \delta)$ *satisfies the following:*

1. *For any* $\delta > 0$ *and any* $q \in Q$*, if* S_q *and* S'_q *are two i.i.d. samples from distribution* f_{A_q}*, then* $\texttt{Test}(S_q, S'_q, \delta)$ *returns* `not clear` *with probability at least* $1 - \delta$*.*
2. *There exists* $N_{\mathrm{t}} = N_{\mathrm{t}}(\delta)$ *such that for any* $\delta > 0$ *and any different* $q, q' \in Q$*, if* S_q *and* $S_{q'}$ *are i.i.d. samples from* f_{A_q} *and* $f_{A_{q'}}$*, and* $\min\{|S_q|, |S_{q'}|\} \geq N_{\mathrm{t}}$*, then* $\texttt{Test}(S_q, S_{q'}, \delta)$ *returns* `distinct` *with probability at least* $1 - \delta$*.*

In words, the first assumption implies that `Test` rarely returns `distinct` when given two samples from the same distribution. The second assumption makes sure that `Test` can distinguish two distinct distributions when large enough samples are available.

Now we proceed to analyze the graph constructed by the main `while`loop in Algorithm 4. We will call each iteration of that loop a *stage*. We denote by H_k the hypothesis graph obtained after by the algorithm at the end of the kth stage. To distinguish these H_k from the output of Algorithm 4 we shall call them *intermediate hypotheses*. A stage will be called *good* if the selected candidate q satisfies $|S_q| \geq N_{\mathrm{t}}$. The following result shows that an initial run of good stages will yield a subgraph of H isomorphic to a subgraph of A.

Lemma 4.3.5. *Suppose the first* k *stages are all good. Then, with probability at least* $1 - kn\delta$*, the subgraph of* H_k *formed by all its safe states is isomorphic to a subgraph of* A*.*

Proof. We proceed by induction on k. The base case corresponds to $k = 0$ where no iteration of the `while`loop has taken place yet. In this case there is only the safe q_λ and the corresponding subgraph is clearly isomorphic to the subgraph of A containing only q_0. Now suppose that the statement is true for some positive k and that stage $k + 1$ is good. Let q be the candidate selected on the $(k + 1)$th stage and S_q the associated sample. Conditioned on the subgraph of candidates at the beginning of stage $k + 1$ being isomorphic to a subgraph of A – which happens with probability at least $1 - kn\delta$ by hypothesis

– we have that S_q is an i.i.d. sample from some $f_{A_{q'}}$ with $q' \in Q$. And since the stage is good we have $|S_q| \geq N_{\mathrm{t}}$. Furthermore, since all previous stages were good, the samples associate with all safes in the current hypothesis also have size at least N_{t} and each can be associated to some state in A because the subgraph of safes is isomorphic to a subgraph of A. To bound the probability of an error at this stage we consider two cases. First suppose that q is promoted to be a new safe. In this case an error occurs if the distribution of S_q is the same as that of one of the safes and the corresponding test returned `distinct`. By Assumption 1 this happens with probability at most δ. Now suppose that q is merged to some already existing safe. In this case an error means that for some safe whose distribution is different from that of S_q the test returned `not clear`. By Assumption 1 and the hypothesis on the size of the sample this happens with probability at most δ for any fixed safe. Since there are at most n safes in the graph, the probability of an error in the $(k+1)$th stage conditioned on that no previous error has occurred is at most $n\delta$. Thus, the probability that the subgraph of safes at the end of the $(k+1)$th stage is isomorphic to a subgraph of A is at least $1-(k+1)n\delta$. □

The next step is to ensure that as long as some frequent state or transition is not part of the hypothesis, the next stage will be good. To define this formally let H_k be some intermediate hypothesis. We say that a frequent state $q \in F_{\mathrm{s}}$ from A *has no representative* in H_k if for all $x \in A[q]$ state q_x in H_k is either a candidate or undefined. Similarly, we say that a frequent transition $(q,\sigma) \in F_{\mathrm{t}}$ from A *has no representative* in H_k if for all $x \in A[q]$ state $q_{x\sigma}$ is either a candidate or undefined.

Lemma 4.3.6. *Suppose there exists a frequent state in A that has no representative in H_k. For any $\delta > 0$, if $|S| \geq (8(L+1)/\varepsilon_{\mathrm{s}})\ln(1/\delta)$ then the selected candidate q in the current stage will have a sample of size $|S_q| \geq |S|\varepsilon_{\mathrm{s}}/(2(L+1)n|\Sigma|)$ with probability at least $1-\delta$.*

Proof. Let $q' \in Q$ denote the frequent state not represented in H_k so far. Note that the probability of a string in S passing through q' is at least $\varepsilon_{\mathrm{s}}/(L+1)$. Thus, by the Chernoff bound, with probability at least $1-\delta$ there will be at least $|S|\varepsilon_{\mathrm{s}}/2(L+1)$ strings in S passing through q', each of which will be parsed to a candidate in H_k. Therefore, since there are at most $n|\Sigma|$ candidates, with probability at least $1-\delta$ there will be some candidate q with $|S_q| \geq |S|\varepsilon_{\mathrm{s}}/(2(L+1)n|\Sigma|)$. □

Mimicking this last proof one can also establish the following result for frequent transitions.

Lemma 4.3.7. *Suppose there exists a frequent transition in A that has no representative in H_k. For any $\delta > 0$, if $|S| \geq (8(L+1)/\varepsilon_{\mathrm{s}}\varepsilon_{\mathrm{t}})\ln(1/\delta)$ then the selected candidate q in the current stage will have a sample of size $|S_q| \geq |S|\varepsilon_{\mathrm{s}}\varepsilon_{\mathrm{t}}/(2(L+1)n|\Sigma|)$ with probability at least $1-\delta$.*

Now we can combine previous lemmas with a union bound to show that if S is large enough, then with high probability Algorithm 4 will find a hypothesis H that captures the frequent core of A.

Theorem 4.3.8. *For any $\delta > 0$, if $|S| \geq (2(L+1)/\varepsilon_{\mathrm{s}}\varepsilon_{\mathrm{t}})\cdot\max\{N_{\mathrm{t}}(\delta)n|\Sigma|, 4\ln(1/\delta)\}$, then with probability at least $1-n(n|\Sigma|^2/2+3|\Sigma|/2+1)\delta$ the hypothesis H produced by Algorithm 4 captures the frequent core of A.*

Proof. Let k denote the number of good stages from the first one until the first non-good stage. By Lemma 4.3.5 the subgraph of safes of H_k is isomorphic to a subgraph of A with probability at least $1-k(k+1)n\delta/2$. Next, by Lemmas 4.3.6 and 4.3.7 and the assumption on $|S|$ we see that each stage in which a frequent state or transition is missing from H will be good with probability at least $1-\delta$. Hence, a union bound shows that H will capture the frequent core of A with probability at least $1-(k(k+1)n/2+|F_{\mathrm{s}}|+|F_{\mathrm{t}}|)\delta$. Now note that each stage of Algorithm 4 creates a new transition between safe states in H. Since a GPDFA with n states over Σ can have at most $n|\Sigma|$ transitions, the algorithm will run for at most $n|\Sigma|$ stages. Thus, we have $k \leq n|\Sigma|$. Combining this observation with the trivial bounds $|F_{\mathrm{s}}| \leq n$ and $|F_{\mathrm{t}}| \leq n|\Sigma|$ yields the desired result. □

4.3.4 Bounding the Error

So far we have seen how to compute the relative entropy between two GPDFA, the effect the frequent core of the target GPDFA has on this quantity, and that Algorithm 4 can identify with high probability

a hypothesis that contains an isomorphic image of that frequent core. The next and last step is to see how these pieces fit together in order to produce an accurate hypothesis. Doing so requires an analysis of the local distributions produced by the `Estimate` function used in Algorithm 4. Like we did in the case of the `Test` function, we will make an axiomatic assumption about `Estimate` here and defer the details to later sections. In particular, we will see that the properties of the `Estimate` function will influence the choice of parameters ε_{s} and ε_{t} defining the frequent core of A.

Assumption 2. *Let S be a sample of examples from $(\Sigma' \times \mathcal{X})$, and $\hat{D}$ the local distribution returned by* `Estimate`(S). *Then the following hold for every local distribution D_q of A and every $\varepsilon, \delta \in (0,1)$:*

1. *There exists $N_{\mathrm{e}} = N_{\mathrm{e}}(\varepsilon, \delta)$ such that if S is drawn i.i.d. from D_q and $|S| \geq N_{\mathrm{e}}$, then $\mathrm{KL}(D_q \| \hat{D}) \leq \varepsilon$ holds with probability at least $1 - \delta$.*
2. *There exists some $\varepsilon_{\mathrm{e}} = \varepsilon_{\mathrm{e}}(|S|) > 0$ such that $\mathrm{KL}(D_q \| \hat{D}) \leq \varepsilon_{\mathrm{e}}$ holds for any S.*

In words, the first assumption above says that any local distribution in A can be accurately estimated in terms of relative entropy provided a large enough sample is available. The second assumptions guarantees that distributions returned by `Estimate` have been smoothed in a way that they will never be too far from any local distribution in A. We note that this is important because the relative entropy between two distributions is a priori unbounded. For this reason we call ε_{e} the *smoothing coefficient* of `Estimate`.

We begin our analysis with a simple consequence of Theorem 4.3.8 that follows by using the same proof with a different sample bound.

Corollary 4.3.9. *Fix some $\delta > 0$ and suppose the following holds:*

$$|S| \geq \frac{2(L+1)}{\varepsilon_{\mathrm{s}}\varepsilon_{\mathrm{t}}} \max\left\{N_{\mathrm{e}}(\varepsilon, \delta)n|\Sigma|, N_t(\delta)n|\Sigma|, 4\ln(1/\delta)\right\} \quad .$$

Then, with probability at least $1 - n(n|\Sigma|^2/2 + 3|\Sigma|/2 + 1)\delta$ the hypothesis H produced by Algorithm 4 captures the frequent core of A and every state q in H which corresponds to a frequent state in A has a sample of size $|S_q| \geq N_{\mathrm{e}}(\varepsilon, \delta)$.

Now we are in a position to state a PAC learning result for GPDFA under Assumptions 1 and 2 on the behavior of `Test` and `Estimate`. The result follows from combining Theorem 4.3.2 and Corollary 4.3.9 with an adequate choice of ε_{s} and ε_{t}.

Theorem 4.3.10. *Suppose A is a GPDFA over Σ with at most n states and that Assumptions 1 and 2 hold. For any $\delta > 0$ and $\varepsilon > 0$, let $\delta' = \delta/(n(n|\Sigma|^2/2 + 3|\Sigma|/2 + 2))$, $\varepsilon_{\mathrm{s}} = \varepsilon/(3n(L+2)\varepsilon_{\mathrm{e}})$, $\varepsilon_{\mathrm{t}} = \varepsilon/(3(L+1)^2\varepsilon_{\mathrm{e}})$, and $\varepsilon_{\mathrm{g}} = \varepsilon/(3(L+1))$. If $|S| \geq (2(L+1)/\varepsilon_{\mathrm{s}}\varepsilon_{\mathrm{t}}) \cdot \max\{N_{\mathrm{e}}(\varepsilon_{\mathrm{g}}, \delta')n|\Sigma|, N_t(\delta')n|\Sigma|, 4\ln(1/\delta')\}$, then with probability at least $1 - \delta$ the hypothesis H returned by Algorithm 4 with input parameters Σ, n, and δ', satisfies $\mathrm{KL}(f_A \| f_H) \leq \varepsilon$.*

Proof. First note that by Corollary 4.3.9 and Assumption 1 , with probability at least $1 - \delta$ we have that H captures the frequent core of A, each frequent state q in A has a representative $\Phi(q)$ in H with associated sample of size at least $N_{\mathrm{e}}(\varepsilon_{\mathrm{g}}, \delta)$, and $\mathrm{KL}(D_q \| \hat{D}_{\Phi(q)}) \leq \varepsilon_{\mathrm{g}}$. Now take the formula from Theorem 4.3.2 and decompose the sum into three terms as follows:

$$\begin{aligned}
\mathrm{KL}(f_A \| f_H) &= \sum_{q \in Q_A} \sum_{q' \in Q_H} W(q, q') \, \mathrm{KL}(D_q \| \hat{D}_{q'}) \\
&= \sum_{q \in F_s} W(q, \Phi(q)) \, \mathrm{KL}(D_q \| \hat{D}_{\Phi(q)}) \\
&+ \sum_{q \in F_s} \sum_{q' \in Q_H \setminus \Phi(q)} W(q, q') \, \mathrm{KL}(D_q \| \hat{D}_{q'}) \\
&+ \sum_{q \in Q_A \setminus F_s} \sum_{q' \in Q_H} W(q, q') \, \mathrm{KL}(D_q \| \hat{D}_{q'}) \quad .
\end{aligned}$$

Using (4.1) and Lemma 4.3.1 we can bound the first term as

$$\sum_{q \in F_s} W(q, \Phi(q)) \operatorname{KL}(D_q \| \hat{D}_{\Phi(q)}) \leq (L(q_0) + 1)\varepsilon_{\mathrm{g}} \leq \frac{\varepsilon}{3} \ .$$

The second term can be bounded using Theorem 4.3.4 and Assumption 2, yielding

$$\begin{aligned} \sum_{q \in F_s} \sum_{q' \in Q_H \setminus \Phi(q)} W(q, q') \operatorname{KL}(D_q \| \hat{D}_{q'}) &\leq (n(L+1)\varepsilon_{\mathrm{s}} + (L+1)^2 \varepsilon_{\mathrm{t}})\varepsilon_{\mathrm{e}} \\ &= \frac{(L+1)\varepsilon}{3(L+2)} + \frac{\varepsilon}{3} \ . \end{aligned}$$

Finally, the third term is bounded by (4.1) and Assumption 2 as

$$\sum_{q \in Q_A \setminus F_s} \sum_{q' \in Q_H} W(q, q') \operatorname{KL}(D_q \| \hat{D}_{q'}) \leq \varepsilon_{\mathrm{s}} \varepsilon_{\mathrm{g}} = \frac{\varepsilon}{3(L+2)} \ .$$

Combining the three bounds above yields $\operatorname{KL}(f_A \| f_H) \leq \varepsilon$. □

4.4 Distinguishing Distributions over Generalized Alphabets

In this section we present a generic approach for implementing the `Test` function used in Algorithm 4 together with some examples. Our approach generalizes the use of L_∞ and $\mathrm{L}_\infty^{\mathrm{p}}$ distances for distinguishing states in algorithms for learning PDFA (see Chapter 2 for details). Basically, we define a measure of distinguishability by taking the supremum over some collection of events of the difference between probabilities assigned by two distributions to these events. In particular, we will take the set of events in such a way that lets us use VC bounds to obtain finite-sample guarantees like the ones required by Assumption 1.

We begin by recalling the definition of L_∞ and $\mathrm{L}_\infty^{\mathrm{p}}$ distances between distributions on $\Sigma^\star$. This will easily hint to the correct generalization in the case of distributions over $(\Sigma \times \mathcal{X})^\star$. Given distributions D and D' over $\Sigma^\star$, the supremum distance L_∞ is defined as

$$\mathrm{L}_\infty(D, D') = \sup_{x \in \Sigma^\star} |D(x) - D'(x)| \ .$$

The prefix supremum distance $\mathrm{L}_\infty^{\mathrm{p}}$ is defined as

$$\mathrm{L}_\infty^{\mathrm{p}}(D, D') = \sup_{x \in \Sigma^\star} |D(x\Sigma^\star) - D'(x\Sigma^\star)| \ .$$

Looking at these definitions it is immediate to see that both these distances are particular instances of a general measure of discrepancy between distributions given by

$$\mathrm{L}_\infty^{\mathfrak{E}}(D, D') = \sup_{E \in \mathfrak{E}} |D(E) - D'(E)| \ ,$$

where $\mathfrak{E}$ is a collection of events over $\Sigma^\star$. In particular, $\mathfrak{E} = \{\{x\} : x \in \Sigma^\star\}$ in the case of L_∞, and $\mathfrak{E} = \{x\Sigma^\star : x \in \Sigma^\star\}$ in the case of $\mathrm{L}_\infty^{\mathrm{p}}$. Now, since there is nothing special about $\Sigma^\star$ in the definition of $\mathrm{L}_\infty^{\mathfrak{E}}$, we can easily use this general form to define discrepancy measures between distributions on $(\Sigma \times \mathcal{X})^\star$.

Note that in general, $\mathrm{L}_\infty^{\mathfrak{E}}$ is not necessarily a distance: if $\mathfrak{E}$ is not large enough, there may exist distributions $D \neq D'$ such that $\mathrm{L}_\infty^{\mathfrak{E}}(D, D') = 0$. On the other hand, we will see that very large sets of events $\mathfrak{E}$ yield worse statistical performance when estimating $\mathrm{L}_\infty^{\mathfrak{E}}(D, D')$ from samples, in addition to the larger computational effort required to compute such estimations. We will thus need to balance this trade-off if we want to obtain efficient learning algorithms for large classes of distributions on $(\Sigma \times \mathcal{X})^\star$ using tests based on this principle. In the most general case, when defining $\mathfrak{E}$ one should also take into account measurability issues that may arise if $\mathcal{X}$ is a continuous space, for example $\mathcal{X} = \mathbb{R}$ equipped

with a Lebesgue measure. We will however ignore such considerations because measurability problems will not appear in the applications we have in mind.

An important and desirable condition for $\mathrm{L}_\infty^{\mathfrak{E}}$ is that we can estimate its true value using random samples. We will see that this is a statistical property that is independent of whether $\mathrm{L}_\infty^{\mathfrak{E}}$ defines a proper distance or not. Also note that the statistical possibility of estimating $\mathrm{L}_\infty^{\mathfrak{E}}$ from a finite set of random examples does not a priori imply that such estimation can be performed efficiently. Let D be a distribution over $(\Sigma \times \mathcal{X})^\star$ and let $S = (z^1, \ldots, z^m)$ be a sample of m i.i.d. examples from D. Given an event $E \in \mathfrak{E}$ we define the *empirical probability* of E with respect to sample S as

$$S(E) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{z^i \in E} \ .$$

If S' is an i.i.d. sample with m' examples from another distribution D', we define the *empirical discrepancy* between S and S' as

$$\mathrm{L}_\infty^{\mathfrak{E}}(S, S') = \sup_{E \in \mathfrak{E}} |S(E) - S'(E)| \ .$$

We note that even though S and S' are finite, computing such quantity requires evaluating a maximum over $\mathfrak{E}$. This computation and the statistical accuracy with which we can estimate $\mathrm{L}_\infty^{\mathfrak{E}}(D, D')$ using $\mathrm{L}_\infty^{\mathfrak{E}}(S, S')$ will depend on the *shattering function* of $\mathfrak{E}$ defined in Appendix A.1 and denoted by $\Pi_{\mathfrak{E}}$. Like we did in Chapter 2, we use this function to define, for any $0 < \delta < 1$, the following quantity:

$$\Delta(\delta) = \sqrt{\frac{8}{M'} \ln\left(\frac{4(\Pi_{\mathfrak{E}}(2m) + \Pi_{\mathfrak{E}}(2m'))}{\delta}\right)} \ ,$$

where $M' = mm'/(\sqrt{m} + \sqrt{m'})^2$. Now let us write $\mu_\star = \mathrm{L}_\infty^{\mathfrak{E}}(D, D')$ and $\hat{\mu} = \mathrm{L}_\infty^{\mathfrak{E}}(S, S')$. With these definitions, the following two results follow from the same proofs used in Chapter 2 . Note that probabilities in these results are with respect to the sampling of S and S'.

Proposition 4.4.1. *With probability at least $1 - \delta$ we have $\mu_\star \leq \hat{\mu} + \Delta(\delta)$.*

Proposition 4.4.2. *With probability at least $1 - \delta$ we have $\mu_\star \geq \hat{\mu} - \Delta(\delta)$.*

These confidence intervals can be used to build a procedure `Test` with finite-sample guarantees; see Chapter 2 for details. In particular, one can show the following useful result which quantifies the number of examples needed to make confident decisions using this `Test` in a particular case of interest.

Corollary 4.4.3. *Let S and S' be i.i.d. samples from distributions D and D' respectively, and write $m = |S|$ and $m' = |S'|$. Suppose that $\mathrm{L}_\infty^{\mathfrak{E}}$ is such that $\Pi_{\mathfrak{E}}(m) = \Theta(m)$. If $\mu_\star = \mathrm{L}_\infty^{\mathfrak{E}}(D, D') > 0$, then with probability at least $1 - \delta$ `Test` certifies this fact when $\min\{m, m'\} \geq N = \tilde{O}((1/\mu_\star^2)\ln(1/\delta))$. Furthermore, if $\mu_\star = \mathrm{L}_\infty^{\mathfrak{E}}(D, D') = 0$, then with probability at least $1 - \delta$ `Test` certifies $\mu_\star < \mu$ when $\min\{m, m'\} \geq N' = \tilde{O}((1/\mu^2)\ln(1/\delta))$.*

Now we give a general construction for a set of events $\mathfrak{E}$ which guarantees that $\mathrm{L}_\infty^{\mathfrak{E}}$ is a distance under very mild conditions. Let $\mathfrak{X}$ be a set of events on $\mathcal{X}$, i.e. $\mathfrak{X} \subseteq 2^{\mathcal{X}}$, such that $\mathrm{L}_\infty^{\mathfrak{X}}$ is a distance between distributions over $\mathcal{X}$. Besides the usual L_∞, this setting includes other well known distances. For example, if $\mathcal{X} = \mathbb{R}$ and $\mathfrak{X} = \{(-\infty, x] : x \in \mathbb{R}\}$, then $\mathrm{L}_\infty^{\mathfrak{X}}$ is the *Kolmogorov–Smirnov distance*. Another example is the *total variation distance* which corresponds to taking $\mathfrak{X}$ to be the full σ-algebra of $\mathcal{X}$ as a measure space; in particular $\mathfrak{X} = 2^{\mathcal{X}}$ if $\mathcal{X}$ is finite or countable. Given $x \in \Sigma^+$ with $|x| = t$ and $X \in \mathfrak{X}$ we write $E_{x,X}$ to denote the subset $x\Sigma^\star \times \mathcal{X}^{t-1} X \mathcal{X}^\star$ of $(\Sigma \times \mathcal{X})^\star$ that contains all $z = (w, y)$ such that $x \sqsubseteq_0 w$ and $y_t \in X$. With this notation we define the following set of events on $(\Sigma \times \mathcal{X})^\star$:

$$\mathfrak{E} = \{E_{x,X} : x \in \Sigma^+, X \in \mathfrak{X}\} \ .$$

It is easy to see in this case that $\mathrm{L}_\infty^{\mathfrak{E}}$ is in fact a distance. Since the proof involves the construction of a GPDFA with an infinite number of states, we give it only in the case of distributions over $(\Sigma \times \mathcal{X})^\star \times (\{\xi\} \times \mathcal{X})$ because it fits better into the model of GPDFA we use in this chapter.

Proposition 4.4.4. *Let D and D' be distributions over $(\Sigma \times \mathcal{X})^\star \times (\{\xi\} \times \mathcal{X})$. Suppose $\mathrm{L}_\infty^{\mathfrak{X}}$ is a distance between distribution on $\mathcal{X}$. If $D \neq D'$ then $\mathrm{L}_\infty^{\mathfrak{E}}(D, D') > 0$.*

Proof. We begin by noting that D and D' can be easily realized by two GPDFA with an infinite number of states. Indeed, let $A = \langle Q, \Sigma, \tau, \gamma, q_0, \xi, \mathcal{D} \rangle$ be defined as follows: $Q = \Sigma^\star$; $q_0 = \lambda$; $\tau(x, \sigma) = x\sigma$ for all $x \in Q$; for all $x \in Q$ with $|x| = t$ and $\sigma \in \Sigma'$ let[1]

$$\gamma(x, \sigma) = \frac{D((x_1, \mathcal{X}) \cdots (x_t, \mathcal{X})(\sigma, \mathcal{X})(\Sigma' \times \mathcal{X})^\star)}{D((x_1, \mathcal{X}) \cdots (x_t, \mathcal{X})(\Sigma' \times \mathcal{X})^\star)} \ ;$$

and for $x \in Q$ with $|x| = t$ and any measurable $X \subseteq \mathcal{X}$, define

$$D_{x,\sigma}(X) = \frac{D((x_1, \mathcal{X}) \cdots (x_t, \mathcal{X})(\sigma, X)(\Sigma' \times \mathcal{X})^\star)}{D((x_1, \mathcal{X}) \cdots (x_t, \mathcal{X})(\Sigma' \times \mathcal{X})^\star)} \ .$$

It is immediate to check that A realizes D. Similarly we define an infinite GPFDA A' realizing D'. Now, if $D \neq D'$, then A and A' are not equal and this means there exists some string $x \in \Sigma^\star$ for which the corresponding local distributions on A and A' are different: $D_x \neq D'_x$. If there are many such x, we take the shortest one, resolving ties using a lexicographical order on Σ'. For such x we must either have $\gamma(x, \sigma) \neq \gamma'(x, \sigma)$ for some $\sigma \in \Sigma'$ or $D_{x,\sigma}(X) \neq D_{x,\sigma}(X)$ for some $\sigma \in \Sigma'$ and $X \in \mathfrak{X}$, where this last claim follows from $\mathrm{L}_\infty^{\mathfrak{X}}$ being a distance. This certifies the existence of an event $E_{x,X} \in \mathfrak{E}$ such that $|D(E_{x,X}) - D'(E_{x,X})| > 0$. □

Observe that even if $\mathrm{L}_\infty^{\mathfrak{E}}$ is a distance, the set of events $\mathfrak{E}$ does not necessarily have slowly growing shattering functions. In particular, it is not hard to see that if we allow samples with arbitrarily long strings then $\Pi_{\mathfrak{E}}(m) = \Omega(2^m)$ for any non-trivial $\mathfrak{X}$. Fortunately, since the distributions we will need to distinguish are not arbitrary, there is a workaround to this problem. In particular, if D and D' are distinct distributions over $(\Sigma' \times \mathcal{X})^\star$ realized by GPDFA A_D and $A_{D'}$ with at most n states, then we must have $|D(E_{x,X}) - D'(E_{x,X})| > 0$ for some $x \in \Sigma^{\leq n}$ and $X \in \mathfrak{X}$. Indeed, note that since A_D has at most n states, any transition in A_D can be accessed with a string of length at most n; thus, if D and D' assign the same probability to all events $E_{x,X}$ with $|x| \leq n$, then A_D and $A_{D'}$ must define the same probability distribution. Hence, given n and $\mathfrak{X}$ we define

$$\mathfrak{E}_n = \{E_{x,X} : x \in \Sigma^{1:n}, X \in \mathfrak{X}\} \ .$$

Now we can bound $\Pi_{\mathfrak{E}_n}(m)$ as a function of n and $\Pi_{\mathfrak{X}}(m)$.

Lemma 4.4.5. *The following holds: $\Pi_{\mathfrak{E}_n}(m) \leq 2mn\Pi_{\mathfrak{X}}(m)$.*

Proof. The bound follows from well-known properties of shattering coefficients; see Appendix A.1 for details. Let us define for any $k \geq 1$ the following collection of events on $\mathcal{X}^\star$:

$$\mathfrak{X}^{(k)} = \{\mathcal{X}^{k-1} X \mathcal{X}^\star : X \in \mathfrak{X}\} \ .$$

Writing $\mathfrak{P} = \{x\Sigma^\star : x \in \Sigma^\star\}$ it is immediate to see that we have $\mathfrak{E}_n \subseteq \mathfrak{P} \otimes \mathfrak{X}^{(1:n)}$, where $\mathfrak{X}^{(1:n)} = \cup_{k=1}^n \mathfrak{X}^{(k)}$. Thus, the following holds for any m:

$$\Pi_{\mathfrak{E}_n}(m) \leq \Pi_{\mathfrak{P} \otimes \mathfrak{X}^{(1:n)}}(m) \leq \Pi_{\mathfrak{P}}(m)\Pi_{\mathfrak{X}^{(1:n)}}(m) \leq 2mn\Pi_{\mathfrak{X}}(m) \ ,$$

where we used Lemma A.5.2 and $\Pi_{\mathfrak{X}^{(k)}}(m) = \Pi_{\mathfrak{X}}(m)$. □

Now we turn our attention to the issue of how to actually compute the empirical $\mathrm{L}_\infty^{\mathfrak{E}_n}$ distance between two samples drawn from distributions D and D' over $(\Sigma \times \mathcal{X})^\star$. First we introduce some notation. Recall that if S_Σ is a multiset of strings from $\Sigma^\star$ we use $S_\Sigma(x\Sigma^\star)$ to denote the empirical frequency of prefix x in S_Σ. We use $\mathrm{pref}(S_\Sigma)$ to denote the set of all prefixes of strings in S_Σ. Furthermore, recall that if $S_\mathcal{X}$ is a multiset of elements from $\mathcal{X}$ we use $S_\mathcal{X}(X)$ to denote the empirical frequency of X in $S_\mathcal{X}$. Let $x \in \Sigma^\star$

[1] We adopt the convention $0/0 = 0$ in all these definitions.

with $x = x_1 \cdots x_t$. For any $n \geq 1$ we write $x_{1:n}$ to denote $x_1 \cdots x_n$ if $n \leq t$ and $x_1 \cdots x_t$ otherwise. Next, let $S = (z^1, \ldots, z^m)$ be a multiset from $(\Sigma \times \mathcal{X})^\star$ with $z^i = (x^i, y^i)$, $x^i \in \Sigma^\star$, and $y^i \in \mathcal{X}^{|x^i|}$. We define $S_{\Sigma^\star} = (x^1, \ldots, x^m)$ and $S_{\Sigma^{\leq n}} = (x^1_{1:n}, \ldots, x^m_{1:n})$ for any $n \geq 1$. Furthermore, given $x \in \Sigma^\star$ with $|x| = t$ we write $S_{x,\mathcal{X}}$ to denote the multiset from $\mathcal{X}$ containing all the y^i_t for which i is such that $x \sqsubseteq_0 x^i$.

Now suppose we are given two samples S and S' from distributions over $(\Sigma \times \mathcal{X})^\star$ and let $U = \mathrm{pref}(S_{\Sigma^{\leq n}} \cup S'_{\Sigma^{\leq n}})$. Then we can write $\mathrm{L}_\infty^{\mathfrak{E}_n}(S, S')$ as the following maximum:

$$\mathrm{L}_\infty^{\mathfrak{E}_n}(S, S') = \max_{x \in U} \max_{X \in \mathfrak{X}} |S_{\Sigma^\star}(x\Sigma^\star) S_{x,\mathcal{X}}(X) - S'_{\Sigma^\star}(x\Sigma^\star) S'_{x,\mathcal{X}}(X)| \ .$$

It is obvious that the maximum over U can be computed in time $O(|U|)$ if we know

$$\max_{X \in \mathfrak{X}} |S_{\Sigma^\star}(x\Sigma^\star) S_{x,\mathcal{X}}(X) - S'_{\Sigma^\star}(x\Sigma^\star) S'_{x,\mathcal{X}}(X)|$$

for each $x \in U$. In particular, it is easy to see that we have $|U| = O(n(m+m'))$ by bounding the number of prefixes in $S_{\Sigma^{\leq n}} \cup S'_{\Sigma^{\leq n}}$. Thus, we are left with the problem of computing

$$\max_{X \in \mathfrak{X}} |p S_{\mathcal{X}}(X) - p' S'_{\mathcal{X}}(X)| \ ,$$

where $p, p' \in [0,1]$ are arbitrary and $S_{\mathcal{X}}$ and $S'_{\mathcal{X}}$ are multisets over $\mathcal{X}$. Note that by definition of shattering coefficients the quantity inside the maximum cannot take more than $\Pi_{\mathfrak{X}}(|S_{\mathcal{X}}| + |S'_{\mathcal{X}}|)$ different values. However, how the actual computation of this maximum can be done in each case depends on the particular structure of $\mathcal{X}$ and $\mathfrak{X}$. We now give three illustrative examples on how to perform such computation.

4.4.1 Examples with $\mathcal{X} = \Delta$

We begin by taking $\mathcal{X} = \Delta$ a finite alphabet and $\mathrm{L}_\infty^{\mathfrak{X}}$ to be the usual L_∞ distance; that is, $\mathfrak{X} = \{\{\delta\} : \delta \in \Delta\}$. Hence, if S and S' is a sample from Δ, then we can compute

$$\max_{\delta \in \Delta} |p S(\delta) - p' S'(\delta)|$$

by reading each sample once, storing the frequencies of each symbol in a table, and then taking the maximum over the $|\Delta|$ possible differences. Assuming constant time read-write data structures, this computation takes time $O(|S| + |S'| + |\Delta|)$. Note also that in this case we have $\Pi_{\mathfrak{X}}(m) \leq |\Delta| + 1$.

In our second example we take $\mathcal{X} = \Delta$ again but now let $\mathrm{L}_\infty^{\mathfrak{X}}$ be the total variation distance; that is, $\mathfrak{X} = \{X : X \subseteq 2^\Delta\}$. We note that in this particular example we have $\Pi_{\mathfrak{X}}(m) \leq 2^{|\Delta|}$. In this case, given samples S and S' the naive approach to compute

$$\max_{X \subseteq \Delta} |p S(X) - p' S'(X)|$$

would take time $\Theta(2^{|\Delta|})$. However, we can take advantage of a property that this maximum shares with the usual total variation distance. In particular, using the following result we see that this computation can also be done in time $O(|S| + |S'| + |\Delta|)$.

Lemma 4.4.6. *For any $p, p' \in [0,1]$ and any samples S and S' on Δ the following holds:*

$$\max_{X \subseteq \Delta} |p S(X) - p' S'(X)| = \frac{|p - p'|}{2} + \frac{1}{2} \sum_{\delta \in \Delta} |p S(\delta) - p' S'(\delta)| \ .$$

Proof. Let us assume without loss of generality that $p \geq p'$. Then it is easy to see using a standard argument that the maximum over $X \subseteq \Delta$ is achieved on

$$X = \{\delta : p S(\delta) \geq p' S'(\delta)\} \ .$$

On the other hand, using $p = \sum_\delta pS(\delta)$ and $p' = \sum_\delta p'S'(\delta)$, we have

$$\begin{aligned}\sum_{\delta\in\Delta} |pS(\delta) - p'S'(\delta)| &= \sum_{\delta\in X}(pS(\delta) - p'S'(\delta)) + \sum_{\delta\in\bar{X}}(p'S'(\delta) - pS(\delta)) \\ &= \sum_{\delta\in X} pS(\delta) - (p - \sum_{\delta\in X} pS(\delta)) \\ &\quad - \sum_{\delta\in X} p'S'(\delta) + (p' - \sum_{\delta\in X} p'S'(\delta)) \\ &= p' - p + 2(pS(X) - p'S'(X)) \ .\end{aligned}$$

□

4.4.2 Example with $\mathcal{X} = \mathbb{R}$

In the last example we let $\mathcal{X} = \mathbb{R}$ and $\mathrm{L}_\infty^{\mathfrak{X}}$ be the Kolmogorov–Smirnov distance; that is, $\mathfrak{X} = \{(-\infty, x] : x \in \mathbb{R}\}$. Now note that if S is a sample from $\mathbb{R}$, in order to compute $S((-\infty, x])$ we only need to count how many points in S are less or equal to x. Furthermore, it is obvious that while x can range over $\mathbb{R}$, $S((-\infty, x])$ can only take values of the form $k/|S|$ for $0 \le k \le |S|$. Thus, given two samples S and S', we can compute

$$\max_{x\in\mathbb{R}} |pS((-\infty, x]) - p'S'((-\infty, x])| \ ,$$

by sorting the values in S and S' and then taking the maximum over the points $x \in S \cup S'$. Overall, this computation takes time $O(|S| \log |S| + |S'| \log |S'|)$. Furthermore, it is easy to see that we have $\Pi_{\mathfrak{X}}(m) = m + 1$.

4.5 Learning Local Distributions

In this section we show how to obtain procedures for estimating distributions over $\Sigma \times \mathcal{X}$ that satisfy the requirements of Assumption 2. We write $\Sigma \times \mathcal{X}$ instead of $\Sigma' \times \mathcal{X}$, though there is now difference from the point of view of the results in this section. We begin by showing how to estimate multinomial distributions with smoothing. We will use this estimate for transition and stopping probabilities. Using this multinomial estimation we show how to estimate the entire local distribution provided we have a procedure for smoothed estimation of distributions over $\mathcal{X}$. Finally we give two examples of such estimations for different choices of $\mathcal{X}$.

4.5.1 Smoothed Estimation of Multinomial Distributions

Let Σ be a finite set and D a probability distribution over Σ. Suppose $S = (x^1, \ldots, x^m)$ is a sample of m i.i.d. examples from D. Given a smoothing parameter $0 \le \alpha \le 1/|\Sigma|$ we define a *smoothed empirical distribution* $\hat{D}$ obtained from S as follows:

$$\hat{D}(\sigma) = S(\sigma)(1 - \alpha|\Sigma|) + \alpha \ ,$$

where $S(\sigma) = S[\sigma]/m = (1/m)\sum_{i=1}^m \mathbb{1}_{x^i=\sigma}$ is the empirical frequency of σ in S. In this smoothing scheme we shall take $\alpha = |\Sigma|^{-1}m^{-1/3}$. Now, given accuracy and confidence parameters $\varepsilon, \delta \in (0, 1)$, we define

$$N_{\mathrm{d}} = N_{\mathrm{d}}(\varepsilon, \delta) = \max\left\{\frac{18^{3/2}|\Sigma|^{3/2}}{\varepsilon^3(1+\varepsilon)^{3/2}}\left(\ln\frac{|\Sigma|}{\delta}\right)^{3/2}, \frac{8(1+\varepsilon)^3}{\varepsilon^3}\right\} \ .$$

With this notation, the following result shows that when m is large enough, then $\hat{D}$ is close to D in terms of relative entropy. We also show that $\mathrm{KL}(D\|\hat{D})$ is always bounded and grows very slowly with m.

Theorem 4.5.1. *For any m we have $\mathrm{KL}(D\|\hat{D}) \le \ln(|\Sigma|m^{1/3})$. Furthermore, for any $\varepsilon, \delta \in (0, 1)$, if $m \ge N_{\mathrm{d}}(\varepsilon, \delta)$, then with probability at least $1 - \delta$ we have $\mathrm{KL}(D\|\hat{D}) \le \varepsilon$.*

Proof. Note in the first place that $D(\sigma) \leq 1$ and $\hat{D}(\sigma) \geq \alpha$ imply

$$\mathrm{KL}(D\|\hat{D}) \leq \sum_{\sigma\in\Sigma} D(\sigma) \ln \frac{1}{\alpha} = \ln(|\Sigma|m^{1/3}) \ .$$

Now, assuming $m \geq N_{\mathrm{d}}$, we analyze the quotient $D(\sigma)/\hat{D}(\sigma)$ in two different situations. First suppose that $D(\sigma) \leq \alpha(1+\varepsilon)$. Since by definition we have $\hat{D}(\sigma) \geq \alpha$, we get $D(\sigma)/\hat{D}(\sigma) \leq 1+\varepsilon$. Now note that by Chernoff and a union bound, with probability at least $1-\delta$ the following holds for all $\sigma \in \Sigma$:

$$S(\sigma) \geq D(\sigma)\left(1 - \sqrt{\frac{2}{D(\sigma)m} \ln \frac{|\Sigma|}{\delta}}\right) \ .$$

Thus, if $D(\sigma) > \alpha(1+\varepsilon)$, our assumption on m implies $S(\sigma) \geq D(\sigma)(1-\varepsilon/3)$. On the other hand, our assumption on m also implies $(1-\alpha|\Sigma|) \geq (1+\varepsilon/2)/(1+\varepsilon)$. Therefore, we get

$$\begin{aligned}
\hat{D}(\sigma) &\geq D(\sigma)(1-\varepsilon/3)(1-\alpha|\Sigma|) \\
&\geq D(\sigma)\frac{(1-\varepsilon/3)(1+\varepsilon/2)}{1+\varepsilon} \\
&\geq \frac{D(\sigma)}{1+\varepsilon} \ .
\end{aligned}$$

Combining the bounds obtained in the two situations we see that, with probability at least $1-\delta$, we have

$$\begin{aligned}
\mathrm{KL}(D\ \|\hat{D}) &= \sum_{\sigma\in\Sigma} D(\sigma) \ln \frac{D(\sigma)}{\hat{D}(\sigma)} \\
&\leq \sum_{\sigma\in\Sigma} D(\sigma) \ln(1+\varepsilon) \\
&\leq \sum_{\sigma\in\Sigma} \varepsilon D(\sigma) = \varepsilon \ .
\end{aligned}$$

□

4.5.2 Estimating Local Distributions in Terms of Relative Entropy

The estimate described in last section can be used to approximate the transition and stopping probabilities in a local distribution. Now we introduce a requirement on a procedure to estimate a class of distributions over $\mathcal{X}$ that will help us build an estimator for local distributions. In particular, let $\mathcal{D}$ denote an arbitrary family of probability distributions over $\mathcal{X}$. Given a sample S of examples from $\mathcal{X}$, we denote by $\texttt{Estimate}_{\mathcal{X}}$ an algorithm that on input S returns a distribution $\hat{D}_{\mathcal{X}}$. We shall assume that this algorithm satisfies the following.

Assumption 3. *Suppose S is a sample from $\mathcal{X}$ and let $\hat{D}_{\mathcal{X}} = \texttt{Estimate}_{\mathcal{X}}(S)$. Then, the following hold for any $\varepsilon, \delta \in (0,1)$:*

1. *There exists $N_{\mathrm{x}} = N_{\mathrm{x}}(\varepsilon,\delta)$ such that if S is i.i.d. from some $D_{\mathcal{X}} \in \mathcal{D}$ with $|S| \geq N_{\mathrm{x}}$, then with probability at least $1-\delta$ we have $\mathrm{KL}(D_{\mathcal{X}}\|\hat{D}_{\mathcal{X}}) \leq \varepsilon$,*
2. *There exists some $\varepsilon_{\mathrm{x}} > 0$ such that for any S and any $D_{\mathcal{X}} \in \mathcal{D}$ we have $\mathrm{KL}(D_{\mathcal{X}}\|\hat{D}_{\mathcal{X}}) \leq \varepsilon_{\mathrm{x}}$.*

Now we show how to approximate a local distribution D over $\Sigma\times\mathcal{X}$ using $\texttt{Estimate}_{\mathcal{X}}$. First note that D can be written as $D = D_\Sigma \otimes D_{\mathcal{X}}$, where $D_{\mathcal{X}}$ is a stochastic kernel on Σ. We will use $D_{\sigma,\mathcal{X}}$ to denote the distribution given by $D_{\mathcal{X}}(\sigma,\bullet)$. Thus, we can estimate D_Σ and the $D_{\sigma,\mathcal{X}}$ separately. In particular, given a sample $S = ((x^1,y^1),\ldots,(x^m,y^m))$ from $\Sigma\times\mathcal{X}$ we define the following sub-samples: $S_\Sigma = (x^1,\ldots,x^m)$ is the corresponding sample from Σ, and for each $\sigma \in \Sigma$ we let $S_{\sigma,\mathcal{X}}$ be the sample containing all y^i such that $x^i = \sigma$. With this notation, our estimate for the local distribution D is defined by taking $\hat{D}_\Sigma(\sigma) = S_\Sigma(\sigma)(1-\alpha|\Sigma|)+\alpha$ as per the previous section, and $\hat{D}_{\sigma,\mathcal{X}} = \texttt{Estimate}_{\mathcal{X}}(S_{\sigma,\mathcal{X}})$ for each $\sigma \in \Sigma$.

Our next results bounds the error on $\hat{D}$. Given accuracy and confidence parameters $\varepsilon, \delta \in (0,1)$ let use define

$$N_{\mathrm{l}}(\varepsilon,\delta) = \max\left\{ N_{\mathrm{d}}(\varepsilon/2,\delta/2), \frac{16|\Sigma|\varepsilon_{\mathrm{x}}}{\varepsilon}\ln\frac{4|\Sigma|}{\delta}, \frac{4|\Sigma|\varepsilon_{\mathrm{x}} N_{\mathrm{x}}(\varepsilon/2|\Sigma|,\delta/4|\Sigma|)}{\varepsilon}\right\} \ .$$

Theorem 4.5.2. *Suppose $\mathcal{D}$ is a family of distributions over $\mathcal{X}$ satisfying Assumption 3. Let D be distribution on $\Sigma\times\mathcal{X}$ such that $D_{\sigma,\mathcal{X}} \in \mathcal{D}$ for every $\sigma \in \Sigma$. For any S we have $\mathrm{KL}(D\|\ \hat{D}) \le \ln(|\Sigma|m^{1/3}) + \varepsilon_{\mathrm{x}}$. Furthermore, for any $\varepsilon,\delta \in (0,1)$, if $m \ge N_{\mathrm{l}}(\varepsilon,\delta)$, then with probability at least $1-\delta$ we have $\mathrm{KL}(D\|\hat{D}) \le \varepsilon$.*

Proof. First observe that, by the product rule of relative entropy, for any pair of distribution D and $\hat{D}$ on $\Sigma\times\mathcal{X}$ we have:

$$\begin{aligned}\mathrm{KL}(D\|\hat{D}) &= \sum_{\sigma\in\Sigma} D_\Sigma(\sigma)\left[\ln\frac{D_\Sigma(\sigma)}{\hat{D}_\Sigma(\sigma)} + \mathrm{KL}(D_{\sigma,\mathcal{X}}\|\hat{D}_{\sigma,\mathcal{X}})\right]\\ &= \mathrm{KL}(D_\Sigma\|\hat{D}_\Sigma) + \sum_{\sigma\in\Sigma} D_\Sigma(\sigma)\,\mathrm{KL}(D_{\sigma,\mathcal{X}}\|\hat{D}_{\sigma,\mathcal{X}})\\ &\le \ln(|\Sigma|m^{1/3}) + \varepsilon_{\mathrm{x}}\ .\end{aligned}$$

On the other hand, using our assumption on m we can use Theorem 4.5.1 to show that $\mathrm{KL}(D_\Sigma\|\hat{D}_\Sigma) \le \varepsilon/2$ holds with probability at least $1-\delta/2$. To bound the second term containing a sum over Σ we consider two cases. If $D_\Sigma(\sigma) \le \varepsilon/2|\Sigma|\varepsilon_{\mathrm{x}}$, then Assumption 3 yields

$$D_\Sigma(\sigma)\,\mathrm{KL}(D_{\sigma,\mathcal{X}}\|\hat{D}_{\sigma,\mathcal{X}}) \le \frac{\varepsilon}{2|\Sigma|}\ .$$

On the other hand, let us write $m_\sigma = |S_{\sigma,\mathcal{X}}|$. By Chernoff, with probability at least $1-\delta/4$ the following holds for all $\sigma\in\Sigma$:

$$m_\sigma \ge mD_\Sigma(\sigma)\left(1-\sqrt{\frac{2}{mD_\Sigma(\sigma)}\ln\frac{4|\Sigma|}{\delta}}\right)\ .$$

In this case, by our assumption on m, we get $m_\sigma \ge N_{\mathrm{x}}(\varepsilon/2|\Sigma|,\delta/4|\Sigma|)$ for each $\sigma\in\Sigma$ such that $D_\Sigma(\sigma) > \varepsilon/2|\Sigma|\varepsilon_{\mathrm{x}}$. Which means that, for all those σ, we get

$$D_\Sigma(\sigma)\,\mathrm{KL}(D_{\sigma,\mathcal{X}}\|\hat{D}_{\sigma,\mathcal{X}}) \le \frac{\varepsilon}{2|\Sigma|}$$

with probability at least $1-\delta/4$. Overall, these bounds imply that with probability at least $1-\delta/2$ we get

$$\sum_{\sigma\in\Sigma} D_\Sigma(\sigma)\,\mathrm{KL}(D_{\sigma,\mathcal{X}}\|\hat{D}_{\sigma,\mathcal{X}}) \le \frac{\varepsilon}{2}\ .$$

Hence, $\mathrm{KL}(D\|\hat{D}) \le \varepsilon$ with probability at least $1-\delta$. □

4.5.3 Example with $\mathcal{X} = \Delta$

In this example we assume that $\mathcal{X} = \Delta$ is another finite alphabet different from Σ. In this case we can readily use the multinomial estimation from Theorem 4.5.1 to estimate the distributions $D_{\sigma,\Delta}$ for each $\sigma\in\Sigma$. We use $\hat{D}$ the following distribution obtained from a sample S of size m from $\Sigma\times\Delta$:

$$\hat{D}(\sigma,\delta) = (S_\Sigma(\sigma)(1-\alpha_\Sigma|\Sigma|)+\alpha_\Sigma)\cdot(S_{\sigma,\Delta}(\delta)(1-\alpha_{\sigma,\Delta}|\Delta|)+\alpha_{\sigma,\Delta})\ ,$$

where $\alpha_\Sigma = |\Sigma|^{-1}m^{-1/3}$ and $\alpha_{\sigma,\Delta} = |\Delta|^{-1}m_\sigma^{-1/3}$. Now, a simple calculation yields the following result.

Corollary 4.5.3. *Let D be a distribution over $\Sigma\ \times\Delta$ and S a sample from $\Sigma\times\Delta$ of size m. For any S we have $\mathrm{KL}(D\|\hat{D}) \le \ln(|\Sigma|\ |\Delta|m^{2/3})$. Furthermore, for any $\varepsilon,\delta\in(0,1)$, if $m \ge N = \tilde{O}((|\Sigma|^3|\Delta|^{3/2}/\varepsilon^4)\ln(1/\delta)^{3/2})$, then with probability at least $1-\delta$ we have $\mathrm{KL}(D\|\hat{D}) \le \varepsilon$.*

4.5.4 Example with $\mathcal{X} = \mathbb{R}_{\geq 0}$

Now we give an example of estimation of local distributions with $\mathcal{X} = \mathbb{R}_{\geq 0}$. In particular, we show how to estimate the distribution of exponential random variables in terms of relative entropy when given upper and lower bounds on the possible rate parameters. Then we combine this result with Theorem 4.5.2 to obtain an estimation procedure for local distributions over $\Sigma \times \mathbb{R}_{\geq 0}$ when the marginal distributions over $\mathbb{R}_{\geq 0}$ follow an exponential law.

We begin with some definitions. Given any two real numbers $a < b$ we define the following piecewise-linear function:

$$t_{a,b}(x) = \begin{cases} a & \text{if } x < a \\ x & \text{if } a \leq x \leq b \\ b & \text{if } x > b \end{cases}$$

Now let $S = (x^1, \ldots, x^m)$ be a sample containing m non-negative examples $x^i \in \mathbb{R}_{\geq 0}$. Furthermore assume we are given $0 < \lambda_{\min} < \lambda_{\max}$. Using these quantities we define the following estimate:

$$\hat{\lambda} = t_{\lambda_{\min}, \lambda_{\max}} \left(\frac{m}{\sum_{i=1}^{m} x^i} \right) \ .$$

Our goal is to show that under some conditions $\hat{\lambda}$ is a good estimate of the rate parameter λ of an exponential distribution.

To begin, let us assume that S is a sample of i.i.d. examples drawn from some exponential distribution $D = \mathrm{Exp}(\lambda)$ with $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. Using $\hat{\lambda}$ we define the *smoothed estimation* of D as $\hat{D} = \mathrm{Exp}(\hat{\lambda})$. The next result will show that $\hat{D}$ satisfies the conditions required by Assumption 3. Given accuracy and confidence parameters $\varepsilon, \delta \in (0, 1)$ we define $N_{\mathrm{E}} = N_{\mathrm{E}}(\varepsilon, \delta) = (32/\varepsilon^2) \ln(2/\delta)$.

Theorem 4.5.4. *For any S we have $\mathrm{KL}(D \| \hat{D}) \leq \lambda_{\max}/\lambda_{\min} - 1$. Furthermore, for any $\varepsilon, \delta \in (0, 1)$, if $m \geq N_{\mathrm{E}}(\varepsilon, \delta)$, then with probability at least $1 - \delta$ we have $\mathrm{KL}(D \| \hat{D}) \leq \varepsilon$.*

Proof. We begin by recalling that for $D = \mathrm{Exp}(\lambda)$ and $\hat{D} = \mathrm{Exp}(\hat{\lambda})$, one has

$$\mathrm{KL}(D \| \hat{D}) = \frac{\hat{\lambda}}{\lambda} + \ln \frac{\lambda}{\hat{\lambda}} - 1 \ .$$

Then, since by construction we have $\lambda, \hat{\lambda} \in [\lambda_{\min}, \lambda_{\max}]$, we see that

$$\mathrm{KL}(D \| \hat{D}) \leq \max \left\{ \frac{\lambda_{\max}}{\lambda_{\min}} - 1, \ln \frac{\lambda_{\max}}{\lambda_{\min}} \right\} = \frac{\lambda_{\max}}{\lambda_{\min}} - 1 \ .$$

Now recall that if $X_1, \ldots, X_m$ is are i.i.d. random variables with distribution $\mathrm{Exp}(\lambda)$, then $Y = X_1 + \cdots + X_m$ follows a distribution $\mathrm{Gamma}(m, 1/\lambda)$. Hence, using Theorem A.1.6 we see that the following holds with probability at least $1 - \delta$:

$$\frac{\lambda}{1 + \sqrt{\frac{2}{m} \ln \frac{2}{\delta}} + \frac{1}{m} \ln \frac{2}{\delta}} \leq \hat{\lambda} \leq \frac{\lambda}{1 - \sqrt{\frac{2}{m} \ln \frac{2}{\delta}} - \frac{1}{m} \ln \frac{2}{\delta}} \ .$$

In particular, for $m \geq (1/2) \ln(2/\delta)$ we get:

$$\frac{\hat{\lambda}}{\lambda} \leq \frac{1}{1 - \sqrt{\frac{8}{m} \ln \frac{2}{\delta}}} \ , \tag{4.4}$$

$$\frac{\lambda}{\hat{\lambda}} \leq 1 + \sqrt{\frac{8}{m} \ln \frac{2}{\delta}} \ . \tag{4.5}$$

So, assuming that S is i.i.d. from $\mathrm{Exp}(\lambda)$ and $m \geq N_{\mathrm{E}}$, we consider two separate cases: $\hat{\lambda} \geq \lambda$ and $\hat{\lambda} < \lambda$. In the first case, using (4.4) and our assumption on m we get

$$\mathrm{KL}(D\|\hat{D}) \leq \frac{\hat{\lambda}}{\lambda} - 1 \leq \frac{\sqrt{\frac{8}{m}\ln\frac{2}{\delta}}}{1 - \sqrt{\frac{8}{m}\ln\frac{2}{\delta}}} \leq \varepsilon \ .$$

In the second case, since $\ln(1+x) \leq x$ for all $x \geq 0$, from (4.5) we get

$$\mathrm{KL}(D\|\hat{D}) \leq \ln\frac{\lambda}{\hat{\lambda}} \leq \ln\left(1 + \sqrt{\frac{8}{m}\ln\frac{2}{\delta}}\right) \leq \varepsilon \ . \qquad \square$$

Finally, plugging the estimation procedure $\texttt{Estimate}_{\mathbb{R}_{\geq 0}}(S) = \mathrm{Exp}(\hat{\lambda})$ for a given sample over $\mathbb{R}_{\geq 0}$ into Theorem 4.5.2 we can show the following result.

Corollary 4.5.5. *Let D be a distribution over $\Sigma \times \mathbb{R}_{\geq 0}$ such that every marginal $D_{\sigma,\mathbb{R}} = \mathrm{Exp}(\lambda_\sigma)$ with $\lambda_{\min} \leq \lambda_\sigma \leq \lambda_{\max}$, and write $\tilde{\lambda} = (\lambda_{\max} - \lambda_{\min})/\lambda_{\min}$. Let S be a sample from $\Sigma \times \mathbb{R}_{\geq 0}$ of size m, and denote by $\hat{D}$ the distribution obtained from it using the procedure described in Theorem 4.5.2. For any sample S we have $\mathrm{KL}(D\|\hat{D}) \leq \ln(|\Sigma|m^{1/3}) + \tilde{\lambda}$. Furthermore, for any $\varepsilon, \delta \in (0,1)$, if $m \geq N = \tilde{O}((|\Sigma|^3\tilde{\lambda}/\varepsilon^3)\ln(1/\delta)^{3/2})$, then with probability at least $1-\delta$ we have $\mathrm{KL}(D\ \|\hat{D}) \leq \varepsilon$.*

4.6 Applications of Generalized PDFA

In this section we collect all results we have proved so far about the learnability of GPDFA, the problem of distinguishing their states, and the estimation of local distributions. We do so by providing two PAC learning results for families of FSM which can be defined in terms of GPDFA. In particular, we choose these two examples because they represent (sub-)classes of distributions which have been used in practical applications and for which no PAC learning results were known.

We begin with an application to Continuous Time Markov Chains (CTMC) and Continuous Time Hidden Markovian Models (CTHMM). These type of models have been used in the model-checking community to represent machines that need to be tested for a set of specifications [BHHK03]. When one wants to conduct a large number of tests on a given machine, it is sometimes more efficient to learn a model of that machine and then run the tests on the model instead of the real machine. In this case it is important to ensure that the model accurately represents the machine being checked. This is why PAC learning algorithms for this type of models may be useful.

A usual modelling assumption used in CTMC and CTHMM is that time between observations follow exponential distributions. This is due to the memoryless property of exponential distributions, which is well-known for combining analytic simplicity with some degree of realism [GH98]. In our case, by taking $\mathcal{X} = \mathbb{R}_{\geq 0}$ and GPDFA with exponential distributions over $\mathbb{R}_{\geq 0}$ in their transitions, we obtain a class of models related to CTMC and CTHMM. In particular, we can model every CTMC and some sub-classes of CTHMM. Hence, the following result gives a learning algorithm for these classes of distributions over $(\Sigma \ \times \mathbb{R})^\star$.

Let Σ be a finite alphabet and n a number of states. Let $\mathrm{L}_\infty^{\mathfrak{E}_n}$ denote the discrepancy in $(\Sigma' \times \mathbb{R}_{\geq 0})$ built from the Kolmogorov–Smirnov distance $\mathrm{L}_\infty^{\mathfrak{X}}$ on $\mathbb{R}$ as described in Section 4.4.2. We define the class of all GPDFA $A = \langle Q, \Sigma, \tau, \gamma, q_0, \xi, \mathcal{D}\rangle$ satisfying the following conditions, and denote it by $\mathcal{GPDFA}_{\mathrm{Exp}}(\Sigma, n, L, \mu, \lambda_{\min}, \lambda_{\max})$:

1. $|Q| \leq n$,
2. $L(q) \leq L$ for every $q \in Q$,
3. $D_{q,\sigma} = \mathrm{Exp}(\lambda)$ with $\lambda_{\min} \leq \lambda \leq \lambda_{\max}$ for every $q \in Q$ and $\sigma \in \Sigma'$,
4. $\min_{q \neq q'} \mathrm{L}_\infty^{\mathfrak{E}_n}(f_{A_q}, f_{A_{q'}}) \geq \mu$.

The following PAC learning result holds for this class of GPDFA, where we use the notation $\tilde{\lambda} = (\lambda_{\max} - \lambda_{\min})/\lambda_{\min}$. The proof is a straightforward calculation combining Theorem 4.3.10, Corollary 4.4.3, and Corollary 4.5.5.

Corollary 4.6.1. *There exists an algorithm such that, for every $\varepsilon, \delta \in (0,1)$, on input Σ, n, δ and an i.i.d. sample S of size m from some distribution f_A with $A \in \mathcal{GPDFA}_{\mathrm{Exp}}(\Sigma, n, L, \mu, \lambda_{\min}, \lambda_{\max})$, with probability at least $1-\delta$, if*

$$m \geq N = \tilde{O}\left(\frac{|\Sigma| n^2 L^4 \tilde{\lambda}^2}{\varepsilon^3} \ln\left(\frac{1}{\delta}\right)^{3/2} \max\left\{\frac{|\Sigma|^3 L^3 \tilde{\lambda}}{\varepsilon^3}, \frac{1}{\mu^2}\right\}\right) ,$$

then the algorithms returns a GPDFA $\hat{A}$ with $\mathrm{KL}(f_A \| f_{\hat{A}}) \leq \varepsilon$.

Our second application is to the problem of learning (stochastic) transductions. That is, functions of type $\Sigma^\star \to \Delta^\star$ and (conditional) probability distributions on $\Sigma^\star \times \Delta^\star$, where Σ is an input alphabet and Δ an output alphabet. In particular, our results on learning GPDFA can be used to prove the PAC learnability of some classes of distributions over $(\Sigma \times \Delta)^\star$ which define *aligned transductions*. That is, transductions where input and output strings must have the same length. We note that our transductions are incomparable to most of the standard classes of transductions considered in the literature. For example, sequential transductions allow input and output strings of different lengths but constrain transitions inside the transducer to be deterministic in terms of both input and output [Hig10]. Our transducers, however, are only required to be deterministic with respect to the input symbols. In particular, the marginal distribution over $\Delta^\star$ defined by a GPDFA with $\mathcal{X} = \Delta$ cannot, in general, be realized by any PDFA.

Let Σ and Δ be finite alphabets and let n be a number of states. Let $\mathrm{L}_\infty^{\mathfrak{E}_n}$ denote the discrepancy in $(\Sigma' \times \Delta)$ built from either the L_∞ or L_1 distance on Δ as described in Section 4.4.1. We write $\mathcal{GPDFA}_\Delta(\Sigma, n, L, \mu)$ to denote the class of all GPDFA $A = \langle Q, \Sigma, \tau, \gamma, q_0, \xi, \mathcal{D}\rangle$ such that the following hold:

1. $|Q| \leq n$,
2. $L(q) \leq L$ for every $q \in Q$,
3. $D_{q,\sigma}$ is a distribution over Δ for every $q \in Q$ and $\sigma \in \Sigma'$,
4. $\min_{q \neq q'} \mathrm{L}_\infty^{\mathfrak{E}_n}(f_{A_q}, f_{A_{q'}}) \geq \mu$.

With this notation, we have the following result which gives a PAC learning algorithm for $\mathcal{GPDFA}_\Delta(\Sigma, n, L, \mu)$. The proof is a straightforward calculation combining Theorem 4.3.10, Corollary 4.4.3, and Corollary 4.5.3.

Corollary 4.6.2. *There exists an algorithm such that, for every $\varepsilon, \delta \in (0,1)$, on input Σ, n, δ and an i.i.d. sample S of size m from some distribution f_A with $A \in \mathcal{GPDFA}_\Delta(\Sigma, n, L, \mu)$, with probability at least $1-\delta$, if*

$$m \geq N = \tilde{O}\left(\frac{|\Sigma| n^2 L^4}{\varepsilon^3} \ln\left(\frac{1}{\delta}\right)^{3/2} \max\left\{\frac{|\Sigma|^3 L^3 |\Delta|^{3/2}}{\varepsilon^3}, \frac{1}{\mu^2}\right\}\right) ,$$

then the algorithms returns a GPDFA $\hat{A}$ with $\mathrm{KL}(f_A \| f_{\hat{A}}) \leq \varepsilon$.

4.7 Proof of Theorem 4.3.2

We will make extensive use of notation from measure theory in this section; see [Pol03] for details. Let $(\mathcal{X}, \mathfrak{A})$ and $(\mathcal{Y}, \mathfrak{B})$ be two measure spaces. We assume that all the measures we consider are sigma-finite. A family of measures $\Lambda = \{\lambda_x | x \in \mathcal{X}\}$ is a *kernel* from $(\mathcal{X}, \mathfrak{A})$ to $(\mathcal{Y}, \mathfrak{B})$ if the map $x \mapsto \lambda_x(B)$ is $\mathfrak{A}$-measurable for each B in $\mathfrak{B}$ and $\lambda_x(\mathcal{Y}) < \infty$ for each x in $\mathcal{X}$. If μ is a measure on $\mathfrak{A}$, then the functional on $\mathcal{M}^+(\mathcal{X} \times \mathcal{Y}, \mathfrak{A} \otimes \mathfrak{B})$ given by

$$f \mapsto \iint f(x,y) \lambda_x(dy) \mu(dx)$$

defines a measure on $\mathfrak{A} \otimes \mathfrak{B}$ which we denote by $\mu \otimes \Lambda$.

Let ν be a measure in $\mathfrak{A}$ such that $\mu \ll \nu$ and λ a measure in $\mathfrak{B}$ such that $\lambda_x \ll \lambda$ for each x. Then, by the Radon–Nikodym Theorem, the following densities are well-defined: $d\mu/d\nu = f$ and $d\lambda_x/d\lambda = g_x$. In this setting one can proof the following.

Lemma 4.7.1. *If $g_x(y)$ is $\mathfrak{A} \otimes \mathfrak{B}$-measurable, then $\mu \otimes \Lambda \ll \nu \otimes \lambda$. Furthermore, $\mu \otimes \Lambda$ has density $(x, y) \mapsto f(x)g_x(y)$ with respect to $\nu \otimes \lambda$.*

Proof. Let C be a $\nu \otimes \lambda$-negligible set and denote by χ_C its characteristic function, which, by assumption, is zero $\nu \otimes \lambda$-almost everywhere. We want to see that

$$(\mu \otimes \Lambda)(C) = \iint \chi_C(x, y)\lambda_x(dy)\mu(dx) = 0 \ .$$

By the measurability of $g_x(y)$, we can use Tonelli's theorem to show that

$$(\mu \otimes \Lambda)(C) = \iint \chi_C(x, y)f(x)g_x(y)\lambda(dy)\nu(dx) \ ,$$

which is equal to zero because $\chi_C(x, y)f(x)g_x(y)$ is zero $\nu \otimes \lambda$-almost everywhere. A similar computation with χ_C replaced by an arbitrary non-negative product measurable function shows that the density is $f(x)g_x(y)$. □

Finally, recall that given measures μ_1, μ_2, ν on a measure space $(\mathcal{X}, \mathfrak{A})$, with $\mu_1, \mu_2 \ll \nu$, $d\mu_1/d\nu = p_1$, and $d\mu_2/d\nu = p_2$, the *relative entropy* or *Kullback-Leibler divergence* between μ_1 and μ_2 is given by

$$\mathrm{KL}(\mu_1 \| \mu_2) = \int_{\mathcal{X}} p_1(x) \log \frac{p_1(x)}{p_2(x)} \nu(dx) \ .$$

4.7.1 Measures Defined by GPDFA

Recall that a GPDFA $A = \langle Q, \Sigma, \tau, \gamma, q_0, \xi, \mathcal{D} \rangle$ induces a probability distribution over the space $\Omega = (\Sigma \times \mathcal{X})^\star \times (\{\xi\} \times \mathcal{X})$. Now we will describe the sigma-field, the measure, and the density associated with this distribution. To compute the density we will need to assume that every distribution $D \in \mathcal{D}$ over $\mathcal{X}$ is absolutely continuous with respect to some fixed measure $\mathfrak{m}$. In particular, for any $D_{q,\sigma} \in \mathcal{D}$ we will denote its density with respect to $\mathfrak{m}$ by $dD_{q,\sigma}/d\mathfrak{m} = g_{q,\sigma}$.

We start by decomposing Ω into the disjoint union of an infinite number of subspaces, each of them having its own sigma-field and measure. Let $\Omega_0 = \{\lambda\} \times (\{\xi\} \times \mathcal{X})$, and for $n \geq 1$, $\Omega_n = (\Sigma \times \mathcal{X})^n \times (\{\xi\} \times \mathcal{X})$. All these spaces are product spaces that can be endowed with a product sigma-field generated by the sigma-fields in each component. In this case, we will take the sigma-field associated with $\mathfrak{m}$ in $\mathcal{X}$, and the power set sigma-field 2^Y for each finite set Y. We will use $\mathfrak{c}$ to denote the counting measure on any discrete set. With these notation we can equip each Ω_n with a standard measure $\nu_n = (\mathfrak{c} \otimes \mathfrak{m})^{n+1}$ for $n \geq 0$.

Now, for each Ω_n we will define a measure μ_n coming from the GPDFA. These μ_n will be defined in terms of measures and kernels from three families that will now be described. From now on, we will identify $(\Sigma \times \mathcal{X})^n$ with $\Sigma^n \times \mathcal{X}^n$ for convenience, and use the notation (x, y) to denote an element from the former set.

The family Θ^n, where Θ^0 is a kernel from Σ' to $\mathcal{X}$, and Θ^n, for $n > 0$, is a kernel from $(\Sigma \times \mathcal{X})^n \times \Sigma'$ to $\mathcal{X}$. The measures contained in these kernels are defined by their densities with respect to $\mathfrak{m}$ as follows:

$$\frac{d\Theta^0_\sigma}{d\mathfrak{m}} = g_{q_0,\sigma} \ , \qquad \frac{d\Theta^n_{(x,y),\sigma}}{d\mathfrak{m}} = g_{\tau(q_0,x),\sigma} \ .$$

The family Γ^n, where Γ^0 is a measure on Σ, and Γ^n, for $n > 0$, is a kernel from $(\Sigma \times \mathcal{X})^n$ to Σ. Their densities with respect to the counting measure are:

$$\frac{d\Gamma^0}{d\mathfrak{c}} = \gamma(q_0, \bullet) \ , \qquad \frac{d\Gamma^n_{(x,y)}}{d\mathfrak{c}} = \gamma\left(\tau(q_0, x), \bullet\right) \ .$$

The family Ξ^n, where Ξ^0 is a measure on $\{\xi\}$, and Ξ^n, for $n > 0$, is a kernel from $(\Sigma \times \mathcal{X})^n$ to $\{\xi\}$. Their densities with respect to the counting measure are:

$$\frac{d\Xi^0}{d\mathfrak{c}} = \gamma(q_0, \bullet) \ , \qquad \frac{d\Xi^n_{(x,y)}}{d\mathfrak{c}} = \gamma(\tau(q_0, x), \bullet) \ .$$

Note that the only difference between the densities from the family Γ^n and the family Ξ^n is the domain where they are defined. Furthermore, the measures in these families may not be probability measures – although they are all finite. However, measures in the family Θ^n are, indeed, probability measures.

Taking products of these measures and kernels we define the following new measures. On Ω_0 we define $\mu_0 = \Xi^0 \otimes \Theta^0$, and on Ω_n for $n > 0$ we define

$$\mu_n = \Gamma^0 \otimes \Theta^0 \otimes \cdots \otimes \Gamma^{n-1} \otimes \Theta^{n-1} \otimes \Xi^n \otimes \Theta^n = \left(\bigotimes_{i=0}^{n-1} \Gamma^i \otimes \Theta^i \right) \otimes (\Xi^n \otimes \Theta^n) \ .$$

It is easy to check that all the densities defined above for the families Θ^n, Λ^n and Ξ^n satisfy the measurability conditions required by Lemma 4.7.1. Therefore, we have $\mu_n \ll \nu_n$ for $n \geq 0$ and the respective densities can be computed. For $n = 0$ one has

$$\left(\frac{d\mu_0}{d\nu_0} \right) (\lambda, \xi, y) = \gamma(q_0, \xi) g_{q_0,\xi}(y) \ .$$

And for $n > 0$,

$$\left(\frac{d\mu_n}{d\nu_n} \right) (x, y) = \prod_{i=0}^{n} \gamma(q_i, x_i) g_{q_i,x_i}(y_i) \ ,$$

where $x = x_0 \cdots x_n$ with $x_n = \xi$, $y = (y_0, \ldots, y_n)$ and $q_i = \tau(q_0, x_0 \cdots x_{i-1})$ for $1 \leq i \leq n$.

4.7.2 Relative Entropy Between GPDFA

Suppose we are given two GPDFA, A and $\hat{A}$, over the same alphabet Σ with the same terminal symbol ξ. These automata induce measures μ and $\hat{\mu}$ over the same space Ω, both dominated by the standard measure ν on Ω. Therefore we can compute the Kullback–Leibler divergence between these two measures. By definition of the divergence and remembering that Ω is the disjoint union of the Ω_n one obtains:

$$\mathrm{KL}(\mu \| \hat{\mu}) = \int_\Omega \frac{d\mu}{d\nu} \log \frac{d\mu/d\nu}{d\hat{\mu}/d\nu} d\nu = \sum_{n \geq 0} \mathrm{KL}(\mu_n \| \hat{\mu}_n) \ .$$

Now we will proceed to evaluate the terms in the sum,

$$\mathrm{KL}(\mu_n \| \hat{\mu}_n) = \int_{\Omega_n} \frac{d\mu_n}{d\nu_n} \log \frac{d\mu_n/d\nu_n}{d\hat{\mu}_n/d\nu_n} d\nu_n \ .$$

Substituting the densities computed in previous section and, recalling that integral over finite spaces with respect to the counting measure is equivalent to summation, one obtains

$$\sum_{x_0} \int \cdots \sum_{x_n} \int \prod_{i=0}^{n} \gamma(q_i, x_i) g_{q_i,x_i}(y_i) \log \frac{\prod_i \gamma(q_i, x_i) g_{q_i,x_i}(y_i)}{\prod_i \hat{\gamma}(\hat{q}_i, x_i) \hat{g}_{\hat{q}_i,x_i}(y_i)} dy_n \cdots dy_0 \ ,$$

where the sums for x_i run over Σ for $0 \leq i \leq n-1$ and $x_n = \xi$.

Now we can permute the integral and sum signs, reorder the products, turn the logarithm of products into sums of logarithms and use the recursive definition of $\gamma : Q \times (\Sigma')^\star \to [0, 1]$. This yields

$$\sum_{x} \int \cdots \int \gamma(q_0, x) \prod_{i=0}^{n} g_i(y_i) \left[\log \frac{\gamma(q_0, x)}{\hat{\gamma}(\hat{q}_0, x)} + \sum_{i=0}^{n} \log \frac{g_i(y_i)}{\hat{g}_i(y_i)} \right] dy_n \cdots dy_0 \ ,$$

where $x = x_0 \cdots x_n$ runs over all the words in $\Sigma^n \xi$ and $g_i = g_{q_i, x_i}$.

Recall that the $g_{q,\sigma}$ are densities of probability measures over $\mathcal{X}$ and therefore $\int g_{q,\sigma}(y)dy = 1$. Hence,

$$\int \cdots \int \prod_{i=0}^{n} g_i(y_i) dy_n \cdots dy_0 = 1 \ ,$$

and, for $0 \leq i \leq n$,

$$\int \cdots \int \prod_{j=0}^{n} g_j(y_j) \log \frac{g_i(y_i)}{\hat{g}_i(y_i)} dy_n \cdots dy_0$$
$$= \int g_i(y_i) \log \frac{g_i(y_i)}{\hat{g}_i(y_i)} dy_i = \mathrm{KL}(g_i \| \hat{g}_i) \ .$$

This allows us to write

$$\mathrm{KL}(\mu_n \| \hat{\mu}_n) = \sum_x \gamma(q_0, x) \left[\log \frac{\gamma(q_0, x)}{\hat{\gamma}(q_0, x)} + \sum_{i=0}^{n} \mathrm{KL}(g_i \| \hat{g}_i) \right] \ .$$

And we finally obtain

$$\mathrm{KL}(\mu \| \hat{\mu}) = \sum_{x \in \Sigma^\star \xi} \gamma(q_0, x) \left[\log \frac{\gamma(q_0, x)}{\hat{\gamma}(q_0, x)} + \sum_{i=0}^{|x|-1} \mathrm{KL}(g_i \| \hat{g}_i) \right] \ ,$$

where $x = x_0 \cdots x_{|x|-1}$, $g_i = g_{q_i, x_i}$ for $0 \leq i \leq |x| - 1$ and $q_i = \tau(q_0, x_{0:i-1})$ for $1 \leq i \leq |x| - 1$.

Now, using the procedure described in [Car97], one can manipulate this expression to obtain a new expression for $\mathrm{KL}(\mu \| \hat{\mu})$ in terms of the divergences between the local distributions in each pair of states from A and $\hat{A}$. This yields

$$\mathrm{KL}(\mu \| \hat{\mu}) = \sum_{q \in Q} \sum_{\hat{q} \in \hat{Q}} W(q, \hat{q}) \sum_{\sigma \in \Sigma'} \gamma(q, \sigma) \left[\log \frac{\gamma(q, \sigma)}{\hat{\gamma}(\hat{q}, \sigma)} + \mathrm{KL}(g_{q,\sigma} \| \hat{g}_{\hat{q},\sigma}) \right] \ .$$

Part II

Spectral Methods

Chapter 5

A Spectral Learning Algorithm for Weighted Automata

This chapter begins the second part of the present dissertation. In it we introduce *weighted automata* and discuss their learnability in a very abstract setting. Weighted automata are a natural generalization of the classes of automata we have seen so far. In particular, they can realize probability distributions generated by HMM and PNFA, as well as other types of distributions which do not admit probabilistic parametrizations (see [DE08] for examples). The learnability results that we present in this chapter require the knowledge of the output produced by the target automaton on a particular set strings. In this respect, the results resemble those on the *query learning* framework.

The starting point for our derivations is to study *Hankel matrices* of functions from strings to real numbers and their relation to weighted automata computing such functions. Using tools from linear algebra, we uncover a duality principle between minimal weighted automata and factorization of their Hankel matrices. This duality will yield a derivation of the so-called spectral learning method, which will be the basis for all the learning algorithms presented in subsequent chapters.

Although the spectral learning algorithm is derived assuming that some Hankel matrices are known exactly, it turns out to be quite robust against approximation noise. Thus, we end the chapter by proving a series of technical results that will be used later for bounding the error of learning algorithms which apply the spectral principle to empirical Hankel matrices estimated from a sample. Here we present the analytical and algebraic machinery needed to obtain such bounds, and defer the statistical aspects of such analyses to next chapters. Some notation and linear algebra results used in our proofs can be found in Appendix A.2.

5.1 Weighted Automata and Hankel Matrices

Let $f : \Sigma^\star \to \mathbb{R}$ be a function over strings. The *Hankel matrix* of f is a bi-infinite matrix $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$ whose entries are defined as $\mathbf{H}_f(u, v) = f(uv)$ for any $u, v \in \Sigma^\star$. That is, rows are indexed by prefixes and columns by suffixes. Note that the Hankel matrix of a function f is a very redundant way to represent f. In particular, the value $f(x)$ appears $|x| + 1$ times in $\mathbf{H}_f$, and we have $f(x) = \mathbf{H}_f(x, \lambda) = \mathbf{H}_f(\lambda, x)$. An obvious observation is that a matrix $\mathbf{M} \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$ satisfying $\mathbf{M}(u_1, v_1) = \mathbf{M}(u_2, v_2)$ for any $u_1 v_1 = u_2 v_2$ is the Hankel matrix of some function $f : \Sigma^\star \to \mathbb{R}$.

We will be considering (finite) sub-blocks of a bi-infinite Hankel matrix $\mathbf{H}_f$. An easy way to define such sub-blocks is using a *basis* $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, where $\mathcal{P} \subseteq \Sigma^\star$ is a set of prefixes and $\mathcal{S} \subseteq \Sigma^\star$ a set of suffixes. We shall write $p = |\mathcal{P}|$ and $s = |\mathcal{S}|$. The sub-block of $\mathbf{H}_f$ defined by $\mathcal{B}$ is the $p \times s$ matrix $\mathbf{H}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ with $\mathbf{H}_\mathcal{B}(u, v) = \mathbf{H}_f(u, v) = f(uv)$ for any $u \in \mathcal{P}$ and $v \in \mathcal{S}$. We may just write $\mathbf{H}$ if the basis $\mathcal{B}$ is arbitrary or obvious from the context.

Not all bases will be equally useful for our purposes. In particular, we will be interested in two properties of bases: closure and completeness. Let $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ be a basis and write $\Sigma' = \Sigma \cup \{\lambda\}$ for convenience. The *p-closure*[1]of $\mathcal{B}$ is the basis $\mathcal{B}' = (\mathcal{P}', \mathcal{S})$, where $\mathcal{P}' = \mathcal{P}\Sigma'$. Equivalently, a basis

$\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is said to be *p-closed* if $\mathcal{P} = \mathcal{P}'\Sigma'$ for some $\mathcal{P}'$ called the *root* of $\mathcal{P}$. It turns out that a Hankel matrix over a p-closed basis can be partitioned into $|\Sigma| + 1$ blocks of the same size. This partition will be useful in many of our results. Let $\mathbf{H}_f$ be a Hankel matrix and $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ a basis. For any $\sigma \in \Sigma'$ we write $\mathbf{H}_\sigma$ to denote the sub-block of $\mathbf{H}_f$ over the basis $(\mathcal{P}\sigma, \mathcal{S})$. That is, the sub-block $\mathbf{H}_\sigma \in \mathbb{R}^{\mathcal{P}\sigma \times \mathcal{S}}$ of $\mathbf{H}_f$ is the $p \times s$ matrix defined by $\mathbf{H}_\sigma(u, v) = \mathbf{H}_f(u\sigma, v)$. Thus, if $\mathcal{B}' = (\mathcal{P}', \mathcal{S})$ is the p-closure of $\mathcal{B}$, then for a particular ordering of the strings in $\mathcal{P}'$, we have

$$\mathbf{H}_{\mathcal{B}'}^\top = \left[\begin{array}{c|c|c|c} \mathbf{H}_\lambda^\top & \mathbf{H}_{\sigma_1}^\top & \cdots & \mathbf{H}_{\sigma_{|\Sigma|}}^\top \end{array} \right] \ .$$

The *rank* of a function $f : \Sigma^\star \to \mathbb{R}$ is defined as the rank of its Hankel matrix: $\mathrm{rank}(f) = \mathrm{rank}(\mathbf{H}_f)$. The rank of a sub-block of $\mathbf{H}_f$ cannot exceed $\mathrm{rank}(f)$, and we will be specially interested on sub-blocks with full rank. We say that a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is *complete* for f if the sub-block $\mathbf{H}_\mathcal{B}$ has full rank: $\mathrm{rank}(\mathbf{H}_\mathcal{B}) = \mathrm{rank}(\mathbf{H}_f)$. In this case we say that $\mathbf{H}_\mathcal{B}$ is a complete sub-block of $\mathbf{H}_f$. It turns out that the rank of f is related to the number of states needed to compute f with a weighted automaton, and that the p-closure of a complete sub-block of $\mathbf{H}_f$ contains enough information to compute this automata. These two results provide the foundations for the learning algorithm presented in Section 5.3.

A widely used class of functions mapping strings to real numbers is that of functions defined by *weighted finite automata* (WFA) or in short *weighted automata* [Moh09]. These functions are also known as *rational power series* [SS78; BR88]. A WFA over Σ with n states is given by a tuple $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$, where $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$ are the *initial* and *final weight* vectors, and $\mathbf{A}_\sigma \in \mathbb{R}^{n \times n}$ the transition matrix associated to each alphabet symbol $\sigma \in \Sigma$. The function f_A realized by a WFA A is defined by

$$f_A(x) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1} \cdots \mathbf{A}_{x_t} \boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty \ ,$$

for any string $x = x_1 \cdots x_t \in \Sigma^\star$ with $t = |x|$ and $x_i \in \Sigma$ for all $1 \leq i \leq t$. We will write $|A|$ to denote the number of states of a WFA. Since $\boldsymbol{\alpha}_0$ will mostly be used as a row vector, we shall omit the transpose sign whenever there is no risk of confusion. The following characterization of the set of functions $f : \Sigma^\star \to \mathbb{R}$ realized by WFA in terms of the rank of their Hankel matrix was given by Carlyle and Paz [CP71] and Fliess [Fli74]. We also note that the construction of an equivalent WFA with the minimal number of states from a given WFA was first given by Schützenberger [Sch61].

Theorem 5.1.1 ([CP71; Fli74])**.** *A function $f : \Sigma^\star \to \mathbb{R}$ can be realized by a WFA if and only if $\mathrm{rank}(\mathbf{H}_f)$ is finite, and in that case $\mathrm{rank}(\mathbf{H}_f)$ is the minimal number of states of any WFA A such that $f = f_A$.*

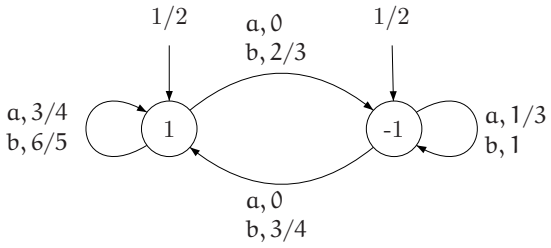
In view of this result, we will say that A is *minimal* for f if $f_A = f$ and $|A| = \mathrm{rank}(f)$. Another useful fact about WFA is their invariance under change of basis. It follows from the definition of f_A that if $\mathbf{M} \in \mathbb{R}^{n \times n}$ is an invertible matrix, then the WFA $B = \langle \boldsymbol{\alpha}_0^\top \mathbf{M}, \mathbf{M}^{-1} \boldsymbol{\alpha}_\infty, \{\mathbf{M}^{-1} \mathbf{A}_\sigma \mathbf{M}\} \rangle$ satisfies $f_B = f_A$. Sometimes B is denoted by $\mathbf{M}^{-1} A \mathbf{M}$. This fact will prove very useful when we consider the problem of learning a WFA realizing a certain function.

Weighted automata are related to other finite state computational models. In particular, WFA can also be defined more generally over an arbitrary semi-ring instead of the field of real numbers, in which case they are sometimes called *multiplicity automata* (MA) [BBBKV00]. It is well known that using weights over an arbitrary semi-ring yields more computational power. However, in this thesis we will only consider WFA with real weights. It is easy to see that several models considered so far (DFA, PDFA, PNFA) can easily be cast as special cases of WFA.

5.1.1 Example

Figure 5.2 shows an example of a weighted automaton $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$ with two states defined over the alphabet $\Sigma = \{a, b\}$, with both its algebraic representation (Figure 5.1b) in terms of vectors and matrices and the equivalent graph representation (Figure 5.1a) useful for a variety of WFA algorithms

[1] The notation *p*-closure stands for prefix-closure. A similar notion can be defined for suffixes as well.



(a) Graph representation

$$\boldsymbol{\alpha}_0^\top = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad \boldsymbol{\alpha}_\infty^\top = \begin{bmatrix} 1 & -1 \end{bmatrix}$$
$$\mathbf{A}_a = \begin{bmatrix} \frac{3}{4} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \quad \mathbf{A}_b = \begin{bmatrix} \frac{6}{5} & \frac{2}{3} \\ \frac{3}{4} & 1 \end{bmatrix}$$

(b) Algebraic representation

Figure 5.2: Example of a weighted automaton over $\Sigma = \{a, b\}$ with 2 states.

[Moh09]. Letting $\mathcal{W} = \{\epsilon, a, b\}$, then $\mathcal{B} = (\mathcal{W}\Sigma', \mathcal{W})$ is a p-closed basis. The following is the Hankel matrix of A on this basis shown with three-digit precision entries:

$$\mathbf{H}_\mathcal{B}^\top = \begin{array}{c} \\ \epsilon \\ a \\ b \end{array} \begin{array}{c} \begin{array}{ccccccc} \epsilon & a & b & aa & ab & ba & bb \end{array} \\ \begin{bmatrix} 0.00 & 0.20 & 0.14 & 0.22 & 0.15 & 0.45 & 0.31 \\ 0.20 & 0.22 & 0.45 & 0.19 & 0.29 & 0.45 & 0.85 \\ 0.14 & 0.15 & 0.31 & 0.13 & 0.20 & 0.32 & 0.58 \end{bmatrix} \end{array}$$

5.2 Duality and Minimal Weighted Automata

Let f be a real function on strings with rank r and $\mathbf{H}_f$ its Hankel matrix. In this section we consider factorizations of $\mathbf{H}_f$ and minimal WFA for f. We will show that there exists an interesting relation between these two concepts. This relation will motivate the algorithm presented on next section that factorizes a (sub-block of a) Hankel matrix in order to learn a WFA for some unknown function.

Our initial observation is that a WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$ for f with n states induces a factorization of $\mathbf{H}_f$. Let $\mathbf{P} \in \mathbb{R}^{\Sigma^\star \times n}$ be a matrix whose uth row equals $\boldsymbol{\alpha}_0^\top \mathbf{A}_u$ for any $u \in \Sigma^\star$. Furthermore, let $\mathbf{S} \in \mathbb{R}^{n \times \Sigma^\star}$ be a matrix whose columns are of the form $\mathbf{A}_v \boldsymbol{\alpha}_\infty$ for all $v \in \Sigma^\star$. It is trivial to check that one has $\mathbf{H}_f = \mathbf{PS}$. The same happens for sub-blocks: if $\mathbf{H}_\mathcal{B}$ is a sub-block of $\mathbf{H}_f$ defined over an arbitrary basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, then the corresponding sub-matrices $\mathbf{P}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times n}$ and $\mathbf{S}_\mathcal{B} \in \mathbb{R}^{n \times \mathcal{S}}$ of $\mathbf{P}$ and $\mathbf{S}$ induce the factorization $\mathbf{H}_\mathcal{B} = \mathbf{P}_\mathcal{B}\mathbf{S}_\mathcal{B}$. Furthermore, if $\mathbf{H}_\sigma$ is a sub-block of the matrix $\mathbf{H}_{\mathcal{B}'}$ corresponding to the p-closure of $\mathbf{H}_\mathcal{B}$, then we have the factorization $\mathbf{H}_\sigma = \mathbf{P}_\mathcal{B}\mathbf{A}_\sigma\mathbf{S}_\mathcal{B}$.

An interesting consequence of this construction is that if A is minimal for f, that is $n = r$, then the factorization $\mathbf{H}_f = \mathbf{PS}$ is in fact a *rank factorization*: $\text{rank}(\mathbf{P}) = \text{rank}(\mathbf{S}) = \text{rank}(\mathbf{H}_f)$. Since in general $\text{rank}(\mathbf{H}_\mathcal{B}) \leq r$, in this case the factorization $\mathbf{H}_\mathcal{B} = \mathbf{P}_\mathcal{B}\mathbf{S}_\mathcal{B}$ is a rank factorization if and only if $\mathbf{H}_\mathcal{B}$ is a complete sub-block. Thus, we see that a minimal WFA that realizes a function f induces a rank factorization on any complete sub-block of $\mathbf{H}_f$. The converse is even more interesting: give a rank factorization of a complete sub-block of $\mathbf{H}_f$, one can compute a minimal WFA for f.

Let $\mathbf{H}$ be a complete sub-block of $\mathbf{H}_f$ defined by the basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ and let $\mathbf{H}_\sigma$ denote the sub-block of the p-closure of $\mathbf{H}$ corresponding to the basis $(\mathcal{P}\sigma, \mathcal{S})$. Let $\mathbf{h}_{\mathcal{P},\lambda} \in \mathbb{R}^\mathcal{P}$ denote the p-dimensional vector with coordinates $\mathbf{h}_{\mathcal{P},\lambda}(u) = f(u)$, and $\mathbf{h}_{\lambda,\mathcal{S}} \in \mathbb{R}^\mathcal{S}$ the s-dimensional vector with coordinates $\mathbf{h}_{\lambda,\mathcal{S}}(v) = f(v)$. Now we can state our result.

Lemma 5.2.1. *If $\mathbf{H} = \mathbf{PS}$ is a rank factorization, then the WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$ with $\boldsymbol{\alpha}_0^\top = \mathbf{h}_{\lambda,\mathcal{S}}^\top \mathbf{S}^+$, $\boldsymbol{\alpha}_\infty = \mathbf{P}^+ \mathbf{h}_{\mathcal{P},\lambda}$, and $\mathbf{A}_\sigma = \mathbf{P}^+ \mathbf{H}_\sigma \mathbf{S}^+$, is minimal for f. Furthermore, A induces the factorizations $\mathbf{H} = \mathbf{PS}$ and $\mathbf{H}_\sigma = \mathbf{PA}_\sigma\mathbf{S}$.*

Proof. Let $A' = \langle \boldsymbol{\alpha}'_0, \boldsymbol{\alpha}'_\infty, \{\mathbf{A}'_\sigma\} \rangle$ be a minimal WFA for f that induces a rank factorization $\mathbf{H} = \mathbf{P}'\mathbf{S}'$. It suffices to show that there exists an invertible $\mathbf{M}$ such that $\mathbf{M}^{-1}A'\mathbf{M} = A$. Define $\mathbf{M} = \mathbf{S}'\mathbf{S}^+$ and note that $\mathbf{P}^+\mathbf{P}'\mathbf{S}'\mathbf{S}^+ = \mathbf{P}^+\mathbf{HS}^+ = \mathbf{I}$ implies that $\mathbf{M}$ is invertible with $\mathbf{M}^{-1} = \mathbf{P}^+\mathbf{P}'$. Now we check that the operators of A correspond to the operators of A' under this change of basis. First we see that $\mathbf{A}_\sigma = \mathbf{P}^+\mathbf{H}_\sigma\mathbf{S}^+ = \mathbf{P}^+\mathbf{P}'\mathbf{A}'_\sigma\mathbf{S}'\mathbf{S}^+ = \mathbf{M}^{-1}\mathbf{A}'_\sigma\mathbf{M}$. Now observe that by the construction of $\mathbf{S}'$ and $\mathbf{P}'$ we have $\boldsymbol{\alpha}_0'^\top \mathbf{S}' = \mathbf{h}_{\lambda,\mathcal{S}}$, and $\mathbf{P}'\boldsymbol{\alpha}'_\infty = \mathbf{h}_{\mathcal{P},\lambda}$. Thus, it follows that $\boldsymbol{\alpha}_0^\top = \boldsymbol{\alpha}_0'^\top \mathbf{M}$ and $\boldsymbol{\alpha}_\infty = \mathbf{M}^{-1}\boldsymbol{\alpha}'_\infty$.

Now let $\mathbf{H} = \mathbf{P}_A \mathbf{S}_A$ be the factorization induced by A. Observe that for any $u \in \mathcal{P}$ the uth row of $\mathbf{P}_A$ satisfies:

$$\boldsymbol{\alpha}_0^\top \mathbf{A}_u = {\boldsymbol{\alpha}'_0}^\top \mathbf{A}'_u \mathbf{M} = \mathbf{P}'_u \mathbf{S}' \mathbf{S}^+ = \mathbf{h}_{u,\mathcal{S}}^\top \mathbf{S}^+ = \mathbf{P}_u \mathbf{S} \mathbf{S}^+ = \mathbf{P}_u \ ,$$

where $\mathbf{P}'_u$ and $\mathbf{P}_u$ represent to the uth rows of $\mathbf{P}'$ and $\mathbf{P}$ respectively, and $\mathbf{h}_{u,\mathcal{S}}^\top$ is the uth row of $\mathbf{H}$. Thus we get $\mathbf{P}_A = \mathbf{P}$, and a symmetric argument shows $\mathbf{S}_A = \mathbf{S}$. The factorization of $\mathbf{H}_\sigma$ follows immediately. □

This result shows that there exists a duality between rank factorizations of complete sub-blocks of $\mathbf{H}_f$ and minimal WFA for f. A consequence of this duality is that all minimal WFA for a function f are related via some change of basis. In other words, modulo change of basis, there exists a unique minimal WFA for any function f of finite rank.

Corollary 5.2.2. *Let $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$ and $A' = \langle \boldsymbol{\alpha}'_0, \boldsymbol{\alpha}'_\infty, \{\mathbf{A}'_\sigma\} \rangle$ be minimal WFA for some f of rank r. Then there exists an invertible matrix $\mathbf{M} \in \mathbb{R}^{r \times r}$ such that $A = \mathbf{M}^{-1} A' \mathbf{M}$.*

Proof. Let $\mathcal{B}$ be any complete basis for f a let $\mathbf{H}_\mathcal{B} = \mathbf{P}\mathbf{S} = \mathbf{P}'\mathbf{S}'$ be the rank factorizations induced by A and A' respectively. Then, by the same arguments used in Lemma 5.2.1, the matrix $\mathbf{M} = \mathbf{S}'\mathbf{S}^+$ is invertible and satisfies the equation $A = \mathbf{M}^{-1} A' \mathbf{M}$. □

The following technical lemma is a duality-like result for models with different numbers of states. It will be useful in later chapters for proving some of our results.

Lemma 5.2.3. *Let $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$ be a WFA with n states. Suppose that $(\mathcal{P}, \mathcal{S})$ is a complete basis for f_A and write $\mathbf{H} = \mathbf{P}\mathbf{S}$ for the factorization induced by A on this Hankel sub-block. For any k and any pair of matrices $\mathbf{N} \in \mathbb{R}^{k \times n}$ and $\mathbf{M} \in \mathbb{R}^{n \times k}$ such that $\mathbf{P}\mathbf{M}\mathbf{N} = \mathbf{P}$, the WFA $B = \mathbf{N}A\mathbf{M} = \langle \boldsymbol{\alpha}_0^\top \mathbf{M}, \mathbf{N}\boldsymbol{\alpha}_\infty, \{\mathbf{N}\mathbf{A}_\sigma\mathbf{M}\} \rangle$ satisfies $f_B = f_A$.*

Proof. Write $B = \langle \boldsymbol{\beta}_0, \boldsymbol{\beta}_\infty, \{\mathbf{B}_\sigma\} \rangle$. It is enough to show that for any $x \in \Sigma^\star$ one has $\boldsymbol{\beta}_0^\top \mathbf{B}_x = \beta_x^\top = \alpha_x^\top \mathbf{M} = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \mathbf{M}$, for then $f_B(x) = \beta_x^\top \boldsymbol{\beta}_\infty = \alpha_x^\top \mathbf{M}\mathbf{N}\boldsymbol{\alpha}_\infty = f_A(x)$, where this last equality follows from $\mathbf{P}\mathbf{M}\mathbf{N} = \mathbf{P}$ and the observation that the rows of $\mathbf{P}$ span the set $\{\alpha_x^\top\}_{x \in \Sigma^\star}$. Note that $\beta_\lambda^\top = \boldsymbol{\beta}_0^\top = \boldsymbol{\alpha}_0^\top \mathbf{M}$ by definition. And thus, by induction on the length of x, we have $\beta_{x\sigma}^\top = \beta_x^\top \mathbf{B}_\sigma = \alpha_x^\top \mathbf{M}\mathbf{N}\mathbf{A}_\sigma\mathbf{M} = \alpha_{x\sigma}^\top \mathbf{M}$. □

5.3 The Spectral Method

The spectral method is basically an efficient algorithm that implements the ideas in the proof of Lemma 5.2.1 to find a rank factorization of a complete sub-block $\mathbf{H}$ of $\mathbf{H}_f$ and obtain from it a minimal WFA for f. A particularly interesting feature of the method is its robustness to noise. Since it uses a factorization based on the *singular value decomposition* (SVD) of $\mathbf{H}$, one can show that if only an approximate sub-block $\hat{\mathbf{H}}$ is available, then the resulting WFA will be close the one one would obtain using the exact matrix. The term *spectral* comes from the fact that SVD is a type of spectral decomposition. We give the basic algorithm in this section. The techniques needed to prove error bounds will be outlined in next section. Specific applications and variations of this algorithm will be the subject of subsequent chapters.

Suppose $f : \Sigma^\star \to \mathbb{R}$ is an unknown function of finite rank r and we want to compute a minimal WFA for it. Let us assume that we know that $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is a complete basis for f. Our algorithm receives as input: the basis $\mathcal{B}$ and the values of f on a set of strings $\mathcal{W}$. In particular, we assume that $\mathcal{P}\Sigma'\mathcal{S} \cup \mathcal{P} \cup \mathcal{S} \subseteq \mathcal{W}$. It is clear that using these values of f the algorithm can compute sub-blocks $\mathbf{H}_\sigma$ for $\sigma \in \Sigma'$ of $\mathbf{H}_f$. Furthermore, it can compute the vectors $\mathbf{h}_{\lambda,\mathcal{S}}$ and $\mathbf{h}_{\mathcal{P},\lambda}$. Thus, the algorithm only needs a rank factorization of $\mathbf{H}_\lambda$ to be able to apply the formulas given in Lemma 5.2.1.

Recall that the *compact SVD* of a $p \times s$ matrix $\mathbf{H}_\lambda$ of rank r is given by the expression $\mathbf{H}_\lambda = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{p \times r}$ and $\mathbf{V} \in \mathbb{R}^{s \times r}$ are orthogonal matrices, and $\boldsymbol{\Lambda} \in \mathbb{R}^{r \times r}$ is a diagonal matrix containing the singular values of $\mathbf{H}_\lambda$. The most interesting property of thin SVD for our purposes is that $\mathbf{H}_\lambda = (\mathbf{U}\boldsymbol{\Lambda})\mathbf{V}^\top$ is a rank factorization. We will use this factorization in the algorithm, but write it in a different way. Note that since $\mathbf{V}$ is orthogonal we have $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$, and in particular $\mathbf{V}^+ = \mathbf{V}^\top$.

Thus, the factorization above is equivalent to $\mathbf{H}_\lambda = (\mathbf{H}_\lambda \mathbf{V})\mathbf{V}^\top$. With this factorization, equations from Lemma 5.2.1 are written as follows:

$$\begin{aligned}\boldsymbol{\alpha}_0^\top &= \mathbf{h}_{\lambda,\mathcal{S}}^\top \mathbf{V} \ , \\ \boldsymbol{\alpha}_\infty &= (\mathbf{H}_\lambda \mathbf{V})^+ \mathbf{h}_{\mathcal{P},\lambda} \ , \\ \mathbf{A}_\sigma &= (\mathbf{H}_\lambda \mathbf{V})^+ \mathbf{H}_\sigma \mathbf{V} \ .\end{aligned}$$

These equations are in essence the spectral learning algorithm for WFA. The underlying algebraic ideas of the algorithm are not much different from the ones used in the algorithm given in [BBBKV00] for learning multiplicity automata from *membership and equivalence queries*. However, there are substantial differences between both algorithms in terms of learning frameworks and capabilities. The first is that in their case the basis is initially unknown and is constructed along the way by trial-and-error via equivalence queries. In contrast, our algorithm requires the basis to be known, but in contrast the use of SVD makes the algorithm robust to noise. This is what will permit us to learn classes of WFA for which approximations of their Hankel matrices can be computed from a *random sample*.

5.4 Recipes for Error Bounds

Being able to bound the error incurred by using an approximate Hankel matrix instead of an exact one in the spectral learning algorithm is one of its major feats. In this section we present the main technical tools needed to prove such bounds. Generally, this involves two steps. The first one analyzes the error in the individual operators, and the second one analyzes how these errors aggregate when evaluating the automaton on a particular string. We will give several variants of each of these steps that will prove useful in later chapters.

5.4.1 Pointwise Difference Between WFA

Let $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ and $A' = \langle \boldsymbol{\alpha}'_0, \boldsymbol{\alpha}'_\infty, \{\mathbf{A}'_\sigma\}\rangle$ be two WFA over Σ with the same number of states n. In this section we are interested in bounding the difference $|f_A(x) - f_{A'}(x)|$ for an arbitrary string $x \in \Sigma^\star$. The bounds we give will be in terms of the differences between the operators of A and A', and of the norms of these operators. We begin with a simple lemma.

Lemma 5.4.1. *Let $\|\cdot\|$ be an arbitrary submultiplicative matrix norm such that for any $\sigma \in \Sigma$ we have $\|\mathbf{A}_\sigma\|, \|\mathbf{A}'_\sigma\| \leq \gamma$ for some $\gamma > 0$. Then, for any $x \in \Sigma^+$ we have*

$$\|\mathbf{A}_x - \mathbf{A}'_x\| \leq \gamma^{|x|-1} \sum_{i=1}^{|x|} \|\mathbf{A}_{x_i} - \mathbf{A}'_{x_i}\| \ .$$

Proof. By induction on the length of x. For $|x| = 1$ the result is trivial. Now suppose $x = y\sigma$. Then, by the triangle inequality, submultiplicativity of the norm, and γ-boundness hypothesis, we have

$$\begin{aligned}\|\mathbf{A}_{y\sigma} - \mathbf{A}'_{y\sigma}\| &\leq \|\mathbf{A}_y - \mathbf{A}'_y\|\|\mathbf{A}_\sigma\| + \|\mathbf{A}'_y\|\|\mathbf{A}_\sigma - \mathbf{A}'_\sigma\| \\ &\leq \gamma \left(\gamma^{|y|-1} \sum_{i=1}^{|y|} \|\mathbf{A}_{y_i} - \mathbf{A}'_{y_i}\| \right) + \gamma^{|y|} \|\mathbf{A}_\sigma - \mathbf{A}'_\sigma\| \\ &= \gamma^{|x|-1} \sum_{i=1}^{|x|} \|\mathbf{A}_{x_i} - \mathbf{A}'_{x_i}\| \ .\end{aligned}$$

□

Using this lemma we will be able to prove the desired bound. The bound will be parametrized by two *Hölder conjugate* real numbers: that is, $p, q \in [1, \infty]$ such that $1/p + 1/q = 1$. We will be using

ℓ_p and ℓ_q vector norms and the corresponding induced norms for matrices. In particular, we define the following quantities:

$$\begin{aligned}\rho_0 &= \|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0\|_p \ , \\ \rho_\infty &= \|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\|_q \ , \\ \rho_\sigma &= \|\mathbf{A}_\sigma - \mathbf{A}'_\sigma\|_q \ .\end{aligned}$$

With these definitions we have the following result.

Lemma 5.4.2. *Let $\gamma > 0$ be such that $\|\boldsymbol{\alpha}_0\|_p, \|\boldsymbol{\alpha}_\infty\|_q, \|\mathbf{A}_\sigma\|_q \leq \gamma$. Then the following holds for any $x \in \Sigma^\star$:*

$$|f_A(x) - f_{A'}(x)| \leq \gamma^{|x|+1} \left(\rho_0 + \rho_\infty + \sum_{i=1}^{|x|} \rho_{x_i} \right) \ .$$

Proof. Applying the triangle inequality, Hölder's inequality, a property of induced norms, norm submultiplicativity, and the γ-boundness assumption, we have

$$\begin{aligned}|\boldsymbol{\alpha}_0 \mathbf{A}_x \boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_0 \mathbf{A}'_x \boldsymbol{\alpha}'_\infty| &\leq |\boldsymbol{\alpha}_0(\mathbf{A}_x \boldsymbol{\alpha}_\infty - \mathbf{A}'_x \boldsymbol{\alpha}'_\infty)| + |(\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0)\mathbf{A}'_x \boldsymbol{\alpha}'_\infty| \\ &\leq \|\boldsymbol{\alpha}_0\|_p \|\mathbf{A}_x \boldsymbol{\alpha}_\infty - \mathbf{A}'_x \boldsymbol{\alpha}'_\infty\|_q + \|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0\|_p \|\mathbf{A}'_x \boldsymbol{\alpha}'_\infty\|_q \\ &\leq \|\boldsymbol{\alpha}_0\|_p \|\mathbf{A}_x\|_q \|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\|_q \\ &\quad + \|\boldsymbol{\alpha}_0\|_p \|\mathbf{A}_x - \mathbf{A}'_x\|_q \|\boldsymbol{\alpha}'_\infty\|_q \\ &\quad + \|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0\|_p \|\mathbf{A}'_x\|_q \|\boldsymbol{\alpha}'_\infty\|_q \\ &\leq \gamma^{|x|+1} \|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\|_q \\ &\quad + \gamma^2 \|\mathbf{A}_x - \mathbf{A}'_x\|_q \\ &\quad + \gamma^{|x|+1} \|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0\|_p \ .\end{aligned}$$

The desired results follows now from Lemma 5.4.1. □

5.4.2 Total Variation Between WFA

If one needs to control the error between A and A' over a whole "slice" of $\Sigma^\star$ the previous bound turns out to be quite loose in many cases. Indeed, using Lemma 5.4.2 we easily get

$$\sum_{x \in \Sigma^t} |f_A(x) - f_{A'}(x)| \leq \gamma^{t+1} \cdot \left(|\Sigma|^t \rho_0 + |\Sigma|^t \rho_\infty + t|\Sigma|^{t-1} \sum_{\sigma \in \Sigma} \rho_\sigma \right) \ .$$

In this case, there are only two ways to avoid an exponential blowup with t on the error: either have the error terms $\rho_0, \rho_\infty, \rho_\sigma$ exponentially small with t, or have a bound $\gamma = O(1/|\Sigma|)$. Both of these are quite restrictive. Thus, in this section we seek a finer control on the accumulation of the error when summing over Σ^t.

This may not be possible in all cases, but here we will concentrate only on the case $p = 1$ and $q = \infty$. Besides the notation used in previous section, here we also define $\gamma_\infty = \|\boldsymbol{\alpha}_\infty\|_\infty$, $\rho_\Sigma = \sum_{\sigma \in \Sigma} \rho_\sigma$, and $\gamma_k = \sum_{x \in \Sigma^k} \|\boldsymbol{\alpha}_0 \mathbf{A}_x\|_1$. Furthermore, if $|\mathbf{M}|$ denotes the matrix obtained by taking the absolute value in all the entries of $\mathbf{M}$, we assume there exists a $\gamma_\Sigma > 0$ such that $\sum_{\sigma \in \Sigma} |\mathbf{A}_\sigma| \mathbf{1} \leq \gamma_\Sigma \mathbf{1}$, where we mean that the inequality must be satisfied by each entry in the vectors.

The following technical lemma will be the key to prove the main result of this section.

Lemma 5.4.3. *The following inequality holds for any $t \geq 0$:*

$$\sum_{x \in \Sigma^t} \|\boldsymbol{\alpha}_0 \mathbf{A}_x - \boldsymbol{\alpha}'_0 \mathbf{A}'_x\|_1 \leq \rho_0 (\gamma_\Sigma + \rho_\Sigma)^t + \rho_\Sigma \sum_{i=0}^{t-1} (\gamma_\Sigma + \rho_\Sigma)^i \gamma_{t-1-i} \ .$$

Proof. In the first place, note that for any $t \geq 0$ the following holds:

$$\begin{aligned}\sum_{y\in\Sigma^t}\sum_{\sigma\in\Sigma}\|(\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y')\mathbf{A}_\sigma\|_1 &\leq \sum_{y\in\Sigma^t}\sum_{\sigma\in\Sigma}|\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y'||\mathbf{A}_\sigma|\mathbf{1} \\ &\leq \gamma_\Sigma \sum_{y\in\Sigma^t}|\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y'|\mathbf{1} \\ &= \gamma_\Sigma \sum_{y\in\Sigma^t}\|\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y'\|_1 \ . \end{aligned} \tag{5.1}$$

Now we proceed by induction on t. Let us define $\varphi_t = \sum_{x\in\Sigma^t}\|\boldsymbol{\alpha}_0\mathbf{A}_x - \boldsymbol{\alpha}_0'\mathbf{A}_x'\|_1$. For $t = 0$ we have $\varphi_0 = \rho_0 = \|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}_0'\|_1$ which satisfies the desired inequality. Now suppose that

$$\rho_t \leq (\rho_\Sigma + \gamma_\Sigma)^t\rho_0 + \rho_\Sigma\sum_{i=0}^{t-1}(\rho_\Sigma + \gamma_\Sigma)^i\gamma_{t-1-i} \ .$$

Recall the duality between induced norms $\|\cdot\|_1$ and $\|\cdot\|_\infty$ which implies that for any vector $\mathbf{v}$ and matrix $\mathbf{M}$ one has $\|\mathbf{v}^\top\mathbf{M}\|_1 = \|\mathbf{M}^\top\mathbf{v}\|_1 \leq \|\mathbf{M}^\top\|_1\|\mathbf{v}\|_1 = \|\mathbf{M}\|_\infty\|\mathbf{v}\|_1$. Then, using this property, the triangle inequality, submultiplicativity of the norm we have:

$$\begin{aligned}\varphi_{t+1} &= \sum_{x\in\Sigma^{t+1}}\|\boldsymbol{\alpha}_0\mathbf{A}_x - \boldsymbol{\alpha}_0'\mathbf{A}_x'\|_1 \\ &\leq \sum_{y\in\Sigma^t}\sum_{\sigma\in\Sigma}\|\boldsymbol{\alpha}_0\mathbf{A}_y(\mathbf{A}_\sigma - \mathbf{A}_\sigma')\|_1 \\ &\quad + \sum_{y\in\Sigma^t}\sum_{\sigma\in\Sigma}\|(\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y')\mathbf{A}_\sigma\|_1 \\ &\quad + \sum_{y\in\Sigma^t}\sum_{\sigma\in\Sigma}\|(\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y')(\mathbf{A}_\sigma - \mathbf{A}_\sigma')\|_1 \\ &\leq \sum_{y\in\Sigma^t}\|\boldsymbol{\alpha}_0\mathbf{A}_y\|_1\sum_{\sigma\in\Sigma}\|\mathbf{A}_\sigma - \mathbf{A}_\sigma'\|_\infty \\ &\quad + \gamma_\Sigma\sum_{y\in\Sigma^t}\|\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y'\|_1 \\ &\quad + \sum_{y\in\Sigma^t}\|\boldsymbol{\alpha}_0\mathbf{A}_y - \boldsymbol{\alpha}_0'\mathbf{A}_y'\|_1\sum_{\sigma\in\Sigma}\|\mathbf{A}_\sigma - \mathbf{A}_\sigma'\|_\infty \\ &= \rho_\Sigma\gamma_t + \rho_t(\rho_\Sigma + \gamma_\Sigma) \\ &\leq (\rho_\Sigma + \gamma_\Sigma)^{t+1}\rho_0 + \rho_\Sigma\sum_{i=0}^{t}(\rho_\Sigma + \gamma_\Sigma)^i\gamma_{t-i} \ , \end{aligned} \tag{5.2}$$

where the last bound follows from the induction hypothesis. □

The following results gives the bound we will be using when bounding the errors between two WFA A and A' over the set Σ^t.

Lemma 5.4.4. *For any $t \geq 0$ the sum $\sum_{x\in\Sigma^t}|f_A(x) - f_{A'}(x)|$ is at most:*

$$(\gamma_\infty + \rho_\infty)\left((\gamma_\Sigma + \rho_\Sigma)^t\rho_0 + \rho_\Sigma\cdot\sum_{i=0}^{t-1}(\gamma_\Sigma + \rho_\Sigma)^i\,\gamma_{t-i-1}\right) + \gamma_t\rho_\infty \ .$$

Proof. Using triangle and Hölder's inequality we obtain the following:

$$\begin{aligned}
\sum_{x\in\Sigma^t} |f_A(x) - f_{A'}(x)| &= \sum_{x\in\Sigma^t} |\boldsymbol{\alpha}_0\mathbf{A}_x\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_0\mathbf{A}'_x\boldsymbol{\alpha}'_\infty| \\
&\leq \sum_{x\in\Sigma^t} |\boldsymbol{\alpha}_0\mathbf{A}_x(\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty)| \\
&\quad + \sum_{x\in\Sigma^t} |(\boldsymbol{\alpha}_0\mathbf{A}_x - \boldsymbol{\alpha}'_0\mathbf{A}'_x)\boldsymbol{\alpha}_\infty| \\
&\quad + \sum_{x\in\Sigma^t} |(\boldsymbol{\alpha}_0\mathbf{A}_x - \boldsymbol{\alpha}'_0\mathbf{A}'_x)(\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty)| \\
&\leq \sum_{x\in\Sigma^t} \|\boldsymbol{\alpha}_0\mathbf{A}_x\|_1\|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\|_\infty \\
&\quad + \sum_{x\in\Sigma^t} \|\boldsymbol{\alpha}_0\mathbf{A}_x - \boldsymbol{\alpha}'_0\mathbf{A}'_x\|_1\|\boldsymbol{\alpha}_\infty\|_\infty \\
&\quad + \sum_{x\in\Sigma^t} \|\boldsymbol{\alpha}_0\mathbf{A}_x - \boldsymbol{\alpha}'_0\mathbf{A}'_x\|_1\|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\|_\infty \\
&= \gamma_t\rho_\infty + (\gamma_\infty + \rho_\infty) \sum_{x\in\Sigma^t} \|\boldsymbol{\alpha}_0\mathbf{A}_x - \boldsymbol{\alpha}'_0\mathbf{A}'_x\|_1 \ . \qquad (5.3)
\end{aligned}$$

Applying Lemma 5.4.3 we get the desired inequality. □

To see that this is tighter that what we obtained from Lemma 5.4.2 suppose we have $\gamma_\infty = \gamma_\Sigma = \gamma_k = 1$. Then the bound from previous lemma yields

$$\sum_{x\in\Sigma^t} |f_A(x) - f_{A'}(x)| \leq (1+\rho_0)(1+\rho_\Sigma)^t(1+\rho_\infty) - 1 \ .$$

In this case we can avoid an exponential blowup in t by just having $\rho_\Sigma = O(1/t)$, which is a clear improvement.

5.4.3 Difference in Operators

From previous sections we know that we can compare the values assigned to strings by two WFA in terms of the differences between their individual operators. Since our main tool for obtaining a WFA from a Hankel matrix will be the spectral method, in this section we study the difference between two WFA obtained from two different sets of Hankel matrices via the equations from Section 5.3. We begin by introducing some notation.

Let $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ be a fixed basis with $p = |\mathcal{P}|$ and $s = |\mathcal{S}|$. Suppose that $\mathbf{H}_\lambda, \mathbf{H}_\sigma, \mathbf{H}'_\lambda, \mathbf{H}'_\sigma \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ are arbitrary Hankel matrices over $\mathcal{B}$. Also, let us take arbitrary vectors $\mathbf{h}_0, \mathbf{h}'_0 \in \mathbb{R}^{\mathcal{P}}$ and $\mathbf{h}_\infty, \mathbf{h}'_\infty \in \mathbb{R}^{\mathcal{S}}$. Furthermore, assume that we are given two matrices $\mathbf{V}, \mathbf{V}' \in \mathbb{R}^{p\times n}$ such that $\mathbf{V}^\top\mathbf{V} = \mathbf{V}'^\top\mathbf{V}' = \mathbf{I}$. Here n is some fixed integer such that $n \leq \min\{p, s\}$.

In this section we will be using $\|\cdot\|$ to denote the euclidean norm for vectors and the operator norm for matrices, which is the corresponding induced norm. Let us define the following quantities in terms of the above vectors and matrices:

$$\begin{aligned}
\varepsilon_\lambda &= \|\mathbf{H}_\lambda - \mathbf{H}'_\lambda\| \ , \\
\varepsilon_\sigma &= \|\mathbf{H}_\sigma - \mathbf{H}'_\sigma\| \ , \\
\varepsilon_V &= \|\mathbf{V} - \mathbf{V}'\| \ , \\
\varepsilon_0 &= \|\mathbf{h}_0 - \mathbf{h}'_0\| \ , \\
\varepsilon_\infty &= \|\mathbf{h}_\infty - \mathbf{h}'_\infty\| \ .
\end{aligned}$$

We also use the notation $\mathfrak{s}_n(\mathbf{M})$ to denote the nth largest singular value of a matrix $\mathbf{M}$. With these definitions we can now state the main result of this section, which is a purely algebraic result about the following differences between vectors and matrices:

$$\begin{aligned}\Delta_\sigma &= \|(\mathbf{H}_\lambda\mathbf{V})^+\mathbf{H}_\sigma\mathbf{V} - (\mathbf{H}'_\lambda\mathbf{V}')^+\mathbf{H}'_\sigma\mathbf{V}'\| \ , \\ \Delta_0 &= \|\mathbf{V}^\top\mathbf{h}_0 - \mathbf{V}'^\top\mathbf{h}'_0\| \ , \\ \Delta_\infty &= \|(\mathbf{H}_\lambda\mathbf{V})^+\mathbf{h}_\infty - (\mathbf{H}'_\lambda\mathbf{V}')^+\mathbf{h}'_\infty\| \ .\end{aligned}$$

Lemma 5.4.5. *With the notation given above, the following three bounds hold:*

$$\begin{aligned}\Delta_\sigma &\leq \frac{\varepsilon_\sigma + \varepsilon_V\|\mathbf{H}'_\sigma\|}{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})} + \frac{1+\sqrt{5}}{2}\frac{\|\mathbf{H}'_\sigma\|(\varepsilon_\lambda + \varepsilon_V\|\mathbf{H}'_\lambda\|)}{\min\{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})^2, \mathfrak{s}_n(\mathbf{H}'_\lambda\mathbf{V}')^2\}} \ , \\ \Delta_0 &\leq \varepsilon_0 + \varepsilon_V\|\mathbf{h}_0\| \ , \\ \Delta_\infty &\leq \frac{\varepsilon_\infty}{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})} + \frac{1+\sqrt{5}}{2}\frac{\|\mathbf{h}'_\infty\|(\varepsilon_\lambda + \varepsilon_V\|\mathbf{H}'_\lambda\|)}{\min\{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})^2, \mathfrak{s}_n(\mathbf{H}'_\lambda\mathbf{V}')^2\}} \ .\end{aligned}$$

Proof. Using the triangle inequality, the submultiplicativity of the operator norm, and the properties of the pseudo-inverse, we can write

$$\begin{aligned}&\|(\mathbf{H}_\lambda\mathbf{V})^+\mathbf{H}_\sigma\mathbf{V} - (\mathbf{H}'_\lambda\mathbf{V}')^+\mathbf{H}'_\sigma\mathbf{V}'\| \\ &\quad= \|(\mathbf{H}_\lambda\mathbf{V})^+(\mathbf{H}_\sigma\mathbf{V} - \mathbf{H}'_\sigma\mathbf{V}') + ((\mathbf{H}'_\lambda\mathbf{V}')^+ - (\mathbf{H}_\lambda\mathbf{V})^+)\mathbf{H}'_\sigma\mathbf{V}'\| \\ &\quad\leq \|(\mathbf{H}_\lambda\mathbf{V})^+\|\|\mathbf{H}_\sigma\mathbf{V} - \mathbf{H}'_\sigma\mathbf{V}'\| + \|(\mathbf{H}_\lambda\mathbf{V})^+ - (\mathbf{H}'_\lambda\mathbf{V}')^+\|\|\mathbf{H}'_\sigma\mathbf{V}'\| \\ &\quad\leq \frac{\|\mathbf{H}_\sigma\mathbf{V} - \mathbf{H}'_\sigma\mathbf{V}'\|}{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})} + \|\mathbf{H}'_\sigma\|\|(\mathbf{H}_\lambda\mathbf{V})^+ - (\mathbf{H}'_\lambda\mathbf{V}')^+\| \ ,\end{aligned}$$

where we used that $\|(\mathbf{H}_\lambda\mathbf{V})^+\| = 1/\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})$ by the properties of pseudo-inverse and operator norm, and $\|\mathbf{H}'_\sigma\mathbf{V}'\| \leq \|\mathbf{H}'_\sigma\|$ by submultiplactivity and $\|\mathbf{V}'\| = 1$. Now note that we also have

$$\|\mathbf{H}_\sigma\mathbf{V} - \mathbf{H}'_\sigma\mathbf{V}'\| \leq \|\mathbf{V}\|\|\mathbf{H}_\sigma - \mathbf{H}'_\sigma\| + \|\mathbf{H}'_\sigma\|\|\mathbf{V} - \mathbf{V}'\| \leq \varepsilon_\sigma + \varepsilon_V\|\mathbf{H}'_\sigma\| \ .$$

Furthermore, using Lemma A.2.3 we obtain

$$\begin{aligned}&\|(\mathbf{H}_\lambda\mathbf{V})^+ - (\mathbf{H}'_\lambda\mathbf{V}')^+\| \\ &\quad\leq \frac{1+\sqrt{5}}{2}\|\mathbf{H}_\lambda\mathbf{V} - \mathbf{H}'_\lambda\mathbf{V}'\|\max\{\|(\mathbf{H}_\lambda\mathbf{V})^+\|^2, \|(\mathbf{H}'_\lambda\mathbf{V}')^+\|^2\} \\ &\quad\leq \frac{1+\sqrt{5}}{2}\frac{\|\mathbf{H}_\lambda - \mathbf{H}_\lambda\|\|\mathbf{V}\| + \|\mathbf{H}'_\lambda\|\|\mathbf{V} - \mathbf{V}'\|}{\min\{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})^2, \mathfrak{s}_n(\mathbf{H}'_\lambda\mathbf{V}')^2\}} \\ &\quad= \frac{1+\sqrt{5}}{2}\frac{\varepsilon_\lambda + \varepsilon_V\|\mathbf{H}'_\lambda\|}{\min\{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})^2, \mathfrak{s}_n(\mathbf{H}'_\lambda\mathbf{V}')^2\}} \ .\end{aligned}$$

Thus we get the first of the bounds. The second bound follows straightforwardly from

$$\|\mathbf{V}^\top\mathbf{h}_0 - \mathbf{V}'^\top\mathbf{h}'_0\| \leq \|\mathbf{V}^\top - \mathbf{V}'^\top\|\|\mathbf{h}_0\| + \|\mathbf{V}'^\top\|\|\mathbf{h}_0 - \mathbf{h}'_0\| = \varepsilon_0 + \varepsilon_V\|\mathbf{h}_0\| \ ,$$

which uses that $\|\mathbf{M}^\top\| = \|\mathbf{M}\|$ because the operator norm is self-dual.

Finally, the last bound follows from the following inequalities, where we use Lemma A.2.3 again:

$$\begin{aligned}&\|(\mathbf{H}_\lambda\mathbf{V})^+\mathbf{h}_\infty - (\mathbf{H}'_\lambda\mathbf{V}')^+\mathbf{h}'_\infty\| \\ &\quad\leq \|(\mathbf{H}_\lambda\mathbf{V})^+\|\|\mathbf{h}_\infty - \mathbf{h}'_\infty\| + \|\mathbf{h}'_\infty\|\|(\mathbf{H}_\lambda\mathbf{V})^+ - (\mathbf{H}'_\lambda\mathbf{V}')^+\| \\ &\quad\leq \frac{\|\mathbf{h}_\infty - \mathbf{h}'_\infty\|}{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})} + \frac{1+\sqrt{5}}{2}\frac{\|\mathbf{h}'_\infty\|\|\mathbf{H}_\lambda\mathbf{V} - \mathbf{H}'_\lambda\mathbf{V}'\|}{\min\{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})^2, \mathfrak{s}_n(\mathbf{H}'_\lambda\mathbf{V}')^2\}} \\ &\quad\leq \frac{\varepsilon_\infty}{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})} + \frac{1+\sqrt{5}}{2}\frac{\|\mathbf{h}'_\infty\|(\varepsilon_\lambda + \varepsilon_V\|\mathbf{H}'_\lambda\|)}{\min\{\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V})^2, \mathfrak{s}_n(\mathbf{H}'_\lambda\mathbf{V}')^2\}} \ .\end{aligned}$$

□

In the following chapters we will see several use cases for the bounds from Lemma 5.4.5. In some cases we will have $\mathbf{V} = \mathbf{V}'$ and $\varepsilon_V = 0$, which greatly simplifies the bounds. In some other cases $\mathbf{V}$ will contain singular vectors from $\mathbf{H}_\lambda$, in which case $\mathfrak{s}_n(\mathbf{H}_\lambda \mathbf{V}) = \mathfrak{s}_n(\mathbf{H}_\lambda)$. However, the common factor in all cases where the Hankel matrices come from some sampling process will be that $\varepsilon_\lambda, \varepsilon_\sigma, \varepsilon_V, \varepsilon_0, \varepsilon_\infty \to 0$ with growing sample size. Combining this behavior with the bounds from Sections 5.4.1 and 5.4.2 we will get point-wise or total variation convergence to the target WFA.

Chapter 6

Sample Bounds for Learning Stochastic Automata

The spectral method described in previous chapter can serve as the basis for deriving learning algorithms in any setting where the Hankel matrix of the target function $f : \Sigma^\star \to \mathbb{R}$ can be efficiently estimated. This chapter analyzes one such setting in detail: the case where f is a probability distribution over $\Sigma^\star$ and an i.i.d. sample drawn from f is given to the learning algorithm. Using similar ideas to the ones we present here, spectral learning algorithms for several classes of probabilistic FSM have been obtained recently. Examples include Hidden Markov Models (HMM) [HKZ09], rational stochastic languages [BDR09], reduced rank HMM [SBG10], Predictive State Representations (PSR) [BSG11], and stochastic Quadratic Weighted Automata (QWA) [Bai11b].

This chapter gives a result along these lines for the class of all Probabilistic Non-deterministic Finite Automata (PNFA) with strong PAC-style learning guarantees. Our result extends the powerful techniques of [HKZ09] to a wider class of distributions and provides much stronger bounds to those proved in [BDR09] for a slightly more general class. Furthermore, we show how learning algorithms based on different statistics over strings – namely, complete strings, prefixes, and substrings – are all reducible to each other.

Like in previous results, our learning algorithm assumes that a complete basis for f is given as input. In practice this is a rather strong assumption which is hard to verify. The last section of this chapter gives a randomized algorithm for finding a complete basis for f under very mild conditions. Though not strictly practical, our approach justifies some of the heuristics used in empirical studies that take the most frequent prefixes and suffixes to build a basis [LQBC12; BQC12].

6.1 Stochastic and Probabilistic Automata

We say that a WFA A is *stochastic* if the function $f = f_A$ is a probability distribution over $\Sigma^\star$. That is, if $f(x) \geq 0$ for all $x \in \Sigma^\star$ and $\sum_{x\in\Sigma^\star} f(x) = 1$. To make it clear that f represents a probability distribution we may sometimes write it as $f(x) = \mathbb{P}[x]$. An interesting fact about distributions over $\Sigma^\star$ is that given an i.i.d. sample generated from that distribution one can compute an estimation $\hat{\mathbf{H}}_f$ of its Hankel matrix, or of any finite sub-block $\hat{\mathbf{H}}_\mathcal{B}$. When the sample is large enough, these estimates will converge to the true Hankel matrices and the spectral method will yield a WFA $\hat{A}$ computing a function $\hat{f} = f_{\hat{A}} \approx f$.

When f realizes a distribution over $\Sigma^\star$, one can think of computing other probabilistic quantities besides probabilities of strings $\mathbb{P}[x]$. For example, one can define the function f_p that computes probabilities of prefixes; that is, $f_\mathrm{p}(x) = \mathbb{P}[x\Sigma^\star]$. Another probabilistic function that can be computed give a distribution over $\Sigma^\star$ is the expected number of times a particular string appears as a substring of random strings. We use f_s to denote this function, which in probabilistic notation can be written as $f_\mathrm{s}(x) = \mathbb{E}[|w|_x]$, where the expectation is with respect to w sampled from f: $\mathbb{E}[|w|_x] = \sum_{w\in\Sigma^\star} |w|_x \mathbb{P}[w]$.

In general the class of stochastic WFA may include some pathological examples with states that are not connected to any terminating state. In order to avoid such cases we introduce the following technical condition. Given a stochastic WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ let $\mathbf{A} = \sum_{\sigma\in\Sigma} \mathbf{A}_\sigma$. We say that A is *irredundant* if $\|\mathbf{A}\| < 1$ for some submultiplicative matrix norm $\|\cdot\|$. Note that a necessary condition for this to happen is that the spectral radius of $\mathbf{A}$ is less than one: $\rho(\mathbf{A}) < 1$. In particular, irredundancy implies that the sum $\sum_{k\geq 0} \mathbf{A}^k$ converges to $(\mathbf{I}-\mathbf{A})^{-1}$. An interesting property of irredundant stochastic WFA is that both f_{p} and f_{s} can also be computed by WFA as shown by the following result.

Lemma 6.1.1. *Let $\langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ be an irredundant stochastic WFA and write: $\mathbf{A} = \sum_{\sigma\in\Sigma} \mathbf{A}_\sigma$, $\tilde{\boldsymbol{\alpha}}_0^\top = \boldsymbol{\alpha}_0^\top(\mathbf{I}-\mathbf{A})^{-1}$, and $\tilde{\boldsymbol{\alpha}}_\infty = (\mathbf{I}-\mathbf{A})^{-1}\boldsymbol{\alpha}_\infty$. Suppose $f : \Sigma^\star \to \mathbb{R}$ is a probability distribution such that $f(x) = \mathbb{P}[x]$ and define functions $f_{\mathrm{p}}(x) = \mathbb{P}[x\Sigma^\star]$ and $f_{\mathrm{s}}(x) = \mathbb{E}[|w|_x]$. Then, the following are equivalent:*

1. *$A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ realizes f,*
2. *$A_{\mathrm{p}} = \langle \boldsymbol{\alpha}_0, \tilde{\boldsymbol{\alpha}}_\infty, \{\mathbf{A}_\sigma\}\rangle$ realizes f_{p},*
3. *$A_{\mathrm{s}} = \langle \tilde{\boldsymbol{\alpha}}_0, \tilde{\boldsymbol{\alpha}}_\infty, \{\mathbf{A}_\sigma\}\rangle$ realizes f_{s}.*

Proof. In the first place we note that because A is irredundant we have

$$\tilde{\boldsymbol{\alpha}}_0^\top = \boldsymbol{\alpha}_0^\top \sum_{k\geq 0} \mathbf{A}^k = \sum_{x\in\Sigma^\star} \boldsymbol{\alpha}_0^\top \mathbf{A}_x \ ,$$

where the second equality follows from a term reordering. Similarly, we have $\tilde{\boldsymbol{\alpha}}_\infty = \sum_{x\in\Sigma^\star} \mathbf{A}_x \boldsymbol{\alpha}_\infty$. The rest of the proof follows from checking several implications.

($1 \Rightarrow 2$) Using $f(x) = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty$ and the definition of $\tilde{\boldsymbol{\alpha}}_\infty$ we have:

$$\mathbb{P}[x\Sigma^\star] = \sum_{y\in\Sigma^\star} \mathbb{P}[xy] = \sum_{y\in\Sigma^\star} \boldsymbol{\alpha}_0^\top \mathbf{A}_x \mathbf{A}_y \boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \tilde{\boldsymbol{\alpha}}_\infty \ .$$

($2 \Rightarrow 1$) It follows from $\mathbb{P}[x\Sigma^+] = \sum_{\sigma\in\Sigma} \mathbb{P}[x\sigma\Sigma^\star]$ that

$$\mathbb{P}[x] = \mathbb{P}[x\Sigma^\star] - \mathbb{P}[x\Sigma^+] = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \tilde{\boldsymbol{\alpha}}_\infty - \boldsymbol{\alpha}_0^\top \mathbf{A}_x \mathbf{A} \tilde{\boldsymbol{\alpha}}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x (\mathbf{I}-\mathbf{A}) \tilde{\boldsymbol{\alpha}}_\infty \ .$$

($1 \Rightarrow 3$) Since we can write $\sum_{w\in\Sigma^\star} \mathbb{P}[w]|w|_x = \mathbb{P}[\Sigma^\star x \Sigma^\star]$, it follows that

$$\begin{aligned}\mathbb{E}[|w|_x] &= \sum_{w\in\Sigma^\star} \mathbb{P}[w]|w|_x \\ &= \sum_{u,v\in\Sigma^\star} \mathbb{P}[uxv] = \sum_{u,v\in\Sigma^\star} \boldsymbol{\alpha}_0^\top \mathbf{A}_u \mathbf{A}_x \mathbf{A}_v \boldsymbol{\alpha}_\infty = \tilde{\boldsymbol{\alpha}}_0^\top \mathbf{A}_x \tilde{\boldsymbol{\alpha}}_\infty \ .\end{aligned}$$

($3 \Rightarrow 1$) Using similar arguments as before we observe that

$$\begin{aligned}\mathbb{P}[x] &= \mathbb{P}[\Sigma^\star x \Sigma^\star] + \mathbb{P}[\Sigma^+ x \Sigma^+] - \mathbb{P}[\Sigma^+ x \Sigma^\star] - \mathbb{P}[\Sigma^\star x \Sigma^+] \\ &= \tilde{\boldsymbol{\alpha}}_0^\top \mathbf{A}_x \tilde{\boldsymbol{\alpha}}_\infty + \tilde{\boldsymbol{\alpha}}_0^\top \mathbf{A}\mathbf{A}_x \mathbf{A} \tilde{\boldsymbol{\alpha}}_\infty - \tilde{\boldsymbol{\alpha}}_0^\top \mathbf{A}\mathbf{A}_x \tilde{\boldsymbol{\alpha}}_\infty - \tilde{\boldsymbol{\alpha}}_0^\top \mathbf{A}_x \mathbf{A} \tilde{\boldsymbol{\alpha}}_\infty \\ &= \tilde{\boldsymbol{\alpha}}_0^\top (\mathbf{I}-\mathbf{A}) \mathbf{A}_x (\mathbf{I}-\mathbf{A}) \tilde{\boldsymbol{\alpha}}_\infty \ .\end{aligned}$$

□

A direct consequence of this constructive result is that given a WFA realizing a probability distribution $\mathbb{P}[x]$ we can easily compute WFA realizing the functions f_{p} and f_{s}; and the converse holds as well. Lemma 6.1.1 also implies the following result, which characterizes the rank of f_{p} and f_{s}.

Corollary 6.1.2. *Suppose $f : \Sigma^\star \to \mathbb{R}$ is stochastic and admits a minimal irredundant WFA. Then* $\operatorname{rank}(f) = \operatorname{rank}(f_{\mathrm{p}}) = \operatorname{rank}(f_{\mathrm{s}})$.

Proof. Since all the constructions of Lemma 6.1.1 preserve the number of states, the result follows from considering minimal WFA for f, f_{p}, and f_{s}. □

From the point of view of learning, Lemma 6.1.1 provides us with tools for proving two-sided reductions between the problems of learning f, f_{p}, and f_{s}. Thus, in the following sections we will give a PAC learning algorithm for stochastic WFA which, modulo translations in accuracy parameters, will yield learning algorithms for probabilities over prefixes and expected counts of substrings. We will not state the resulting theorems, but note that this turns out to be a simple exercise in view of the following lemma and the tools from Section 5.4.

Lemma 6.1.3. *Let $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ be an irredundant stochastic WFA and $A' = \langle \boldsymbol{\alpha}'_0, \boldsymbol{\alpha}'_\infty, \{\mathbf{A}'_\sigma\}\rangle$ an arbitrary WFA, both with n states. Write $\rho_0 = \|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0\|$, $\rho_\infty = \|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\|$, and $\rho_\Sigma = \|\mathbf{A} - \mathbf{A}'\|$. Suppose that $\rho_\Sigma \leq \mathfrak{s}_n(\mathbf{I} - \mathbf{A})$ and define: $\tilde{\boldsymbol{\alpha}}_0 = \boldsymbol{\alpha}_0(\mathbf{I} - \mathbf{A})^{-1}$, $\tilde{\boldsymbol{\alpha}}_\infty = (\mathbf{I} - \mathbf{A})^{-1}\boldsymbol{\alpha}_\infty$, $\tilde{\boldsymbol{\alpha}}'_0 = \boldsymbol{\alpha}'_0(\mathbf{I} - \mathbf{A}')^{-1}$, and $\tilde{\boldsymbol{\alpha}}'_\infty = (\mathbf{I} - \mathbf{A}')^{-1}\boldsymbol{\alpha}'_\infty$. Then the two following inequalities hold:*

$$\|\tilde{\boldsymbol{\alpha}}_0 - \tilde{\boldsymbol{\alpha}}'_0\| \leq \frac{\rho_\Sigma \|\boldsymbol{\alpha}_0\|}{\mathfrak{s}_n(\mathbf{I} - \mathbf{A})(\mathfrak{s}_n(\mathbf{I} - \mathbf{A}) - \rho_\Sigma)} + \frac{\rho_0}{\mathfrak{s}_n(\mathbf{I} - \mathbf{A})} \ ,$$

$$\|\tilde{\boldsymbol{\alpha}}_\infty - \tilde{\boldsymbol{\alpha}}'_\infty\| \leq \frac{\rho_\Sigma \|\boldsymbol{\alpha}_\infty\|}{\mathfrak{s}_n(\mathbf{I} - \mathbf{A})(\mathfrak{s}_n(\mathbf{I} - \mathbf{A}) - \rho_\Sigma)} + \frac{\rho_\infty}{\mathfrak{s}_n(\mathbf{I} - \mathbf{A})} \ .$$

Proof. Both bounds follow from the triangle inequality and Lemma A.2.2. □

The family of stochastic WFA contains a particular sub-family with intrinsic interest. We say that a WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ with n states is *probabilistic* if the following are satisfied:

1. all parameters are non-negative. That is, for all $\sigma \in \Sigma$ and all $i, j \in [n]$: $\mathbf{A}_\sigma(i,j) \geq 0$, $\boldsymbol{\alpha}_0(i) \geq 0$, and $\boldsymbol{\alpha}_\infty(i) \geq 0$,
2. initial weights add up to one: $\sum_{i \in [n]} \boldsymbol{\alpha}_0(i) = 1$,
3. transition and final weights from each state add up to one. That is, for all $i \in [n]$: $\boldsymbol{\alpha}_\infty(i) + \sum_{\sigma \in \Sigma} \sum_{j \in [n]} \mathbf{A}_\sigma(i,j) = 1$.

This model is also called in the literature a *probabilistic finite automata* (PFA) or a *probabilistic nondeterministic finite automata* (PNFA). It is obvious that probabilistic WFA are also stochastic, since $f_A(x)$ is the probability of generating x using the given automaton.

It turns out that when a probabilistic WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ is considered, the factorization induced on $\mathbf{H}$ has a nice probabilistic interpretation. Analyzing the spectral algorithm from this perspective yields additional insights which are useful to keep in mind.

Let $\mathbf{H}_f = \mathbf{PS}$ be the factorization induced by a probabilistic WFA with n states on the Hankel matrix of $f_A(x) = f(x) = \mathbb{P}[x]$. Then, for any prefix $u \in \Sigma^\star$, the uth row of $\mathbf{P}$ is given by the following n-dimensional vector:

$$\mathbf{p}_u(i) = \mathbb{P}[u\Sigma^\star \ , \ s_{|u|} = i] \ , \qquad i \in [n] \ ,$$

where s_t denotes the current state of the automaton after t steps. That is, the probability that the probabilistic transition system given by A generates the prefix u and ends up in state i. The coordinates of these vectors are usually called *forward probabilities*. Similarly, the column of $\mathbf{S}$ given by suffix $v \in \Sigma^\star$ is the n-dimensional vector given by:

$$\mathbf{s}_v(i) = \mathbb{P}[v \mid s = i] \ , \qquad i \in [n] \ .$$

This is the probability of generating a suffix s when A is started from state i. These are usually called *backward probabilities*.

The same interpretation applies to the factorization induced on a sub-block $\mathbf{H}_\mathcal{B} = \mathbf{P}_\mathcal{B}\mathbf{S}_\mathcal{B}$. Therefore, assuming there exists a minimal WFA for $f(x) = \mathbb{P}[x]$ which is probabilistic, Lemma 5.2.1 says that a WFA for f can be learned from information about the forward and backward probabilities over a small set of prefixes and suffixes. Teaming this basic observation with the spectral method and invariance under change of basis one can show an interesting fact: forward and backward (empirical) probabilities for a probabilistic WFA can be recovered (modulo a change of basis) by computing an SVD on (empirical) string probabilities. In other words, though state probabilities are *non-observable*, they can be recovered

(modulo a linear transformation) from *observable* quantities. In the following sections we will prove bounds for learning probability distributions generated by (possibly non-minimal) probabilistic automata. This turns out to require a bit more work than previous results in [HKZ09] for learning a sub-family of HMM.

6.2 Sample Bounds for Hankel Matrix Estimation

Let $f(x) = \mathbb{P}[x]$ be a probability distribution over $\Sigma^\star$. Let $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ be a basis and denote by $\mathbf{H}$ the Hankel sub-block over this basis of $\mathbf{H}_f$; that is, the entries in $\mathbf{H}$ correspond to probabilities of certain events. Suppose $S = (x^1, \ldots, x^m)$ is a sample of m i.i.d. strings sampled from the distribution computed by f. Let $\hat{\mathbf{H}}$ be the empirical estimation of $\mathbf{H}$ computed from S. Our goal is to give a bound for the error $\|\mathbf{H} - \hat{\mathbf{H}}\|_F$ that holds with high probability over the sampling of S.

An important observation about $\hat{\mathbf{H}}$ is that it can be written as the sum of independent random Hankel matrices. First, observe that for any $u \in \mathcal{P}$ and $v \in \mathcal{S}$ the entry $\hat{\mathbf{H}}(u,v)$ corresponds to the empirical frequency in S of the string uv: $\hat{\mathbf{H}}(u,v) = \frac{1}{m}\sum_{i=1}^m \mathbb{1}_{x^i=uv}$. Thus, letting $\mathbf{H}_{x^i} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ be a Hankel matrix with entries $\mathbf{H}_{x^i}(u,v) = \mathbb{1}_{x^i=uv}$, we get $\hat{\mathbf{H}} = \frac{1}{m}\sum_{i=1}^m \mathbf{H}_{x^i}$. Now, since for any fixed (u,v) we have $\mathbb{E}[\mathbb{1}_{x=uv}] = \mathbb{P}[uv]$, it is clear that $\mathbb{E}[\mathbf{H}_x] = \mathbf{H}$ and $\mathbb{E}[\hat{\mathbf{H}}] = \mathbf{H}$. Furthermore, because $\hat{\mathbf{H}}(u,v)$ is the sum of m i.i.d. Bernoulli random variables we have for any $(u,v) \in \mathcal{P}\times\mathcal{S}$:

$$\mathbb{V}[\hat{\mathbf{H}}(u,v)] = \mathbb{E}[(\mathbf{H}(u,v) - \hat{\mathbf{H}}(u,v))^2] = \frac{1}{m}\mathbf{H}(u,v)(1-\mathbf{H}(u,v)) \ . \tag{6.1}$$

The next result will prove the desired bound on $\|\mathbf{H} - \hat{\mathbf{H}}\|_F$ by using McDiarmid's inequality. The following quantity, called *redundancy coefficient* $c_{\mathcal{B}}$ of the basis $\mathcal{B}$, will appear in the bound: $c_{\mathcal{B}} = \max_{x\in\Sigma^\star} \|\mathbf{H}_x\|_F^2$. Note that $c_{\mathcal{B}}$ is precisely the maximum number of pairs (u,v) that can be simultaneously "hit" by any $x \in \Sigma^\star$.

Theorem 6.2.1. *With probability at least $1-\delta$, the following holds: $\|\mathbf{H}-\hat{\mathbf{H}}\|_F \leq \sqrt{c_{\mathcal{B}}/m}(1+\sqrt{\ln(1/\delta)})$.*

Proof. We apply McDiarmid's inequality (A.1) to the random variable $\|\mathbf{H} - \hat{\mathbf{H}}\|_F$. The first step is to bound the expectation of this random variable as follows:

$$\begin{aligned}
\mathbb{E}\left[\|\mathbf{H} - \hat{\mathbf{H}}\|_F\right] &= \mathbb{E}\left[\sqrt{\sum_{(u,v)\in\mathcal{P}\times\mathcal{S}} (\mathbf{H}(u,v) - \hat{\mathbf{H}}(u,v))^2}\right] \\
&\leq \sqrt{\sum_{(u,v)\in\mathcal{P}\times\mathcal{S}} \mathbb{E}[(\mathbf{H}(u,v) - \hat{\mathbf{H}}(u,v))^2]} \\
&= \frac{1}{\sqrt{m}}\sqrt{\sum_{(u,v)\in\mathcal{P}\times\mathcal{S}} \mathbf{H}(u,v)(1-\mathbf{H}(u,v))} \\
&\leq \frac{1}{\sqrt{m}}\sqrt{c_{\mathcal{B}} - \|\mathbf{H}\|_F^2} \\
&\leq \sqrt{\frac{c_{\mathcal{B}}}{m}} \ ,
\end{aligned}$$

where we used Jensen's inequality, Equation (6.1), and the bound

$$\sum_{(u,v)\in\mathcal{P}\times\mathcal{S}} \mathbf{H}(u,v) \leq \sum_{x\in\mathcal{B}} f(x)\mathbf{1}^\top\mathbf{H}_x\mathbf{1} = \sum_{x\in\mathcal{B}} f(x)\|\mathbf{H}_x\|_F \leq c_{\mathcal{B}}\sum_{x\in\mathcal{B}} f(x) \leq c_{\mathcal{B}} \ .$$

Now we bound the stability coefficients. Let S' be a sample that differs from S only on one string and denote by $\hat{\mathbf{H}}'$ the estimation of $\mathbf{H}$ computed from sample S'. Since all examples in S are identically distributed, we can assume the difference is in the last string: $x'^m \neq x^m$. Thus, we have

$$\left|\|\mathbf{H} - \hat{\mathbf{H}}\|_F - \|\mathbf{H} - \hat{\mathbf{H}}'\|_F\right| \leq \|\hat{\mathbf{H}} - \hat{\mathbf{H}}'\|_F = \frac{1}{m}\|\mathbf{H}_{x^m} - \mathbf{H}_{x'^m}\|_F \ .$$

Algorithm 5: Algorithm for Learning Stochastic WFA

Input: r, Σ, $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, $S = (x^1, \ldots, x^m)$
Output: A WFA $\hat{A} = \langle \hat{\boldsymbol{\alpha}}_0, \hat{\boldsymbol{\alpha}}_\infty, \{\hat{\mathbf{A}}_\sigma\} \rangle$

Using S, compute estimations $\hat{\mathbf{H}}_\sigma$ for all $\sigma \in \Sigma'$, $\hat{\mathbf{h}}_{\lambda,\mathcal{S}}$, and $\hat{\mathbf{h}}_{\mathcal{P},\lambda}$;
Compute the r top right singular vectors $\hat{\mathbf{V}}$ of $\mathbf{H}_\lambda$;
Let $\hat{\boldsymbol{\alpha}}_0^\top \leftarrow \hat{\mathbf{h}}_{\lambda,\mathcal{S}}^\top \hat{\mathbf{V}}$ and $\hat{\boldsymbol{\alpha}}_\infty \leftarrow (\hat{\mathbf{H}}_\lambda \hat{\mathbf{V}})^+ \hat{\mathbf{h}}_{\mathcal{P},\lambda}$;
foreach $\sigma \in \Sigma$ **do** let $\hat{\mathbf{A}}_\sigma \leftarrow (\hat{\mathbf{H}}_\lambda \hat{\mathbf{V}})^+ \hat{\mathbf{H}}_\sigma \hat{\mathbf{V}}$;

Now we observe that from the definition of $c_{\mathcal{B}}$ we get

$$\begin{aligned}
\|\mathbf{H}_{x^m} - \mathbf{H}_{x'^m}\|_F &\leq \sqrt{\sum_{(u,v)\in\mathcal{P}\times\mathcal{S}} (\mathbf{H}_{x^m}(u,v)^2 + \mathbf{H}_{x'^m}(u,v)^2)} \\
&= \sqrt{\|\mathbf{H}_{x^m}\|_F^2 + \|\mathbf{H}_{x'^m}\|_F^2} \\
&\leq \sqrt{2} \max_{x\in\Sigma^\star} \|\mathbf{H}_x\|_F \\
&= \sqrt{2c_{\mathcal{B}}} \quad .
\end{aligned}$$

Applying McDiarmid's inequality we now get

$$\mathbb{P}\left[\|\mathbf{H} - \hat{\mathbf{H}}\|_F \geq \mathbb{E}\left[\|\mathbf{H} - \hat{\mathbf{H}}\|_F\right] + t\right] \leq \exp\left(-\frac{mt^2}{c_{\mathcal{B}}}\right) \quad ,$$

which implies that with probability at least $1 - \delta$ one has

$$\begin{aligned}
\|\mathbf{H} - \hat{\mathbf{H}}\|_F &\leq \mathbb{E}\left[\|\mathbf{H} - \hat{\mathbf{H}}\|_F\right] + \sqrt{\frac{c_{\mathcal{B}}}{m} \ln \frac{1}{\delta}} \\
&\leq \sqrt{\frac{c_{\mathcal{B}}}{m}} + \sqrt{\frac{c_{\mathcal{B}}}{m} \ln \frac{1}{\delta}} \quad .
\end{aligned}$$

□

6.3 PAC Learning PNFA

In this section we present and analyze a PAC learning algorithm for stochastic WFA. We will not analyze the algorithm in general, but only for those probability distributions over $\Sigma^\star$ that can be parametrized by a PNFA. A similar analysis could be performed in the general case using techniques similar to those from [Bai11b]. However, in that case the sample bound one obtains is of type $\exp(\log(|\Sigma|)\log(1/\varepsilon)) \notin \mathsf{poly}(1/\varepsilon, |\Sigma|)$. On the other hand, by refining the techniques from [HKZ09] – in particular Lemma 5.4.4 – we are able to give here PAC-style bounds for all PNFA which are polynomial in all of their parameters. We now start by introducing some notation and describing the algorithm.

Let $f : \Sigma^\star \to \mathbb{R}$ compute a probability distribution $f(x) = \mathbb{P}[x]$ with $\mathrm{rank}(f) = r$. We also suppose that $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is a complete basis for f. Finally, given a sample size m, suppose $S = (x^1, \ldots, x^m)$ are i.i.d. strings sampled from the distribution computed by f. In this setup, given r, Σ, $\mathcal{B}$, and S, the pseudo-code from Algorithm 5 implements the spectral method described in Section 5.3 and returns a WFA $\hat{A}$ such that $\hat{f} = f_{\hat{A}}$ approximates f. Noting that the approximate Hankel matrices can be computed from S in linear time $O(m)$, that the cost of computing the truncated SVD and the pseudo-inverse is $O(|\mathcal{P}|\ |\mathcal{S}|r)$, and the cost of computing the operators is $O(|\Sigma|\ |\mathcal{P}|r^2)$, we get a total running time for this spectral algorithm of $O(m + r|\mathcal{P}||\mathcal{S}| + r^2|\mathcal{P}|\ |\Sigma|)$.

We will now proceed to make this statement formal by proving a PAC learning bound whose sample size will depend polynomially on some parameters of the target f. Let $\mathbf{V}$ denote the top r right singular

vectors of $\mathbf{H}_\lambda$. For further reference we shall define the WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ as

$$\begin{aligned}\boldsymbol{\alpha}_0^\top &= \mathbf{h}_{\lambda,\mathcal{S}}^\top \mathbf{V} \ , \\ \boldsymbol{\alpha}_\infty &= (\mathbf{H}_\lambda \mathbf{V})^+ \mathbf{h}_{\mathcal{P},\lambda} \ , \\ \mathbf{A}_\sigma &= (\mathbf{H}_\lambda \mathbf{V})^+ \mathbf{H}_\sigma \mathbf{V} \ .\end{aligned}$$

Note that A depends only on f and the basis $\mathcal{B}$ used to define $\mathbf{H}_\lambda$. To begin with, we prove that if $\hat{\mathbf{H}}_\lambda$ is close enough to the true Hankel matrix $\mathbf{H}_\lambda$, then the approximate singular vectors $\hat{\mathbf{V}}$ satisfy an interesting property. That is, that the WFA $\tilde{A} = \langle \tilde{\boldsymbol{\alpha}}_0, \tilde{\boldsymbol{\alpha}}_\infty, \{\tilde{\mathbf{A}}_\sigma\}\rangle$ given by

$$\begin{aligned}\tilde{\boldsymbol{\alpha}}_0^\top &= \mathbf{h}_{\lambda,\mathcal{S}}^\top \hat{\mathbf{V}} \ , \\ \tilde{\boldsymbol{\alpha}}_\infty &= (\mathbf{H}_\lambda \hat{\mathbf{V}})^+ \mathbf{h}_{\mathcal{P},\lambda} \ , \\ \tilde{\mathbf{A}}_\sigma &= (\mathbf{H}_\lambda \hat{\mathbf{V}})^+ \mathbf{H}_\sigma \hat{\mathbf{V}} \ ,\end{aligned}$$

satisfies $f_{\tilde{A}} = f$. This will prove useful when bounding the error between f and $\hat{f}$ because it will enable us to compare $f_{\tilde{A}}$ with $f_{\hat{A}}$ in our error bounds and thus use Lemma 5.4.5.

Lemma 6.3.1. *Suppose that* $\|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2 \leq \epsilon/(1+\epsilon)\mathfrak{s}_r(\mathbf{H}_\lambda)$ *for some* $\epsilon < 1$. *Then the following inequality holds:*

$$\mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}}) \geq \mathfrak{s}_r(\mathbf{H}_\lambda)\sqrt{1-\epsilon^2} \ .$$

Furthermore, this implies $f_{\tilde{A}} = f$.

Proof. Let us write $\mathbf{H}_\lambda = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top$ for the compact SVD decomposition of $\mathbf{H}_\lambda$. If we denote by $\mathbf{u}_r$ the rth left singular vector of matrix $\mathbf{V}^\top\hat{\mathbf{V}}$, then using Lemma A.2.7 we get

$$\begin{aligned}\mathfrak{s}_r(\mathbf{V}^\top\hat{\mathbf{V}}) &= \|\mathbf{u}_r^\top \mathbf{V}^\top \hat{\mathbf{V}}\|_2 \\ &\geq \|\mathbf{u}_r\|_2 \sqrt{1 - \frac{\|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2^2}{(\mathfrak{s}_r(\mathbf{H}_\lambda) - \|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2)^2}} \\ &= \sqrt{1 - \frac{\|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2^2}{(\mathfrak{s}_r(\mathbf{H}_\lambda) - \|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2)^2}} \\ &\geq \sqrt{1-\epsilon^2} \ .\end{aligned}$$

Thus, using Lemma A.2.6 we get

$$\begin{aligned}\mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}}) &= \mathfrak{s}_r(\mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top\hat{\mathbf{V}}) \\ &\geq \mathfrak{s}_r(\mathbf{U}\boldsymbol{\Lambda})\mathfrak{s}_r(\mathbf{V}^\top\hat{\mathbf{V}}) \\ &\geq \mathfrak{s}_r(\mathbf{H}_\lambda)\sqrt{1-\epsilon^2} \ .\end{aligned}$$

To prove the last claim we show that $\tilde{A} = \mathbf{M}A\mathbf{M}^{-1}$ with $\mathbf{M} = (\mathbf{H}_\lambda\hat{\mathbf{V}})^+\mathbf{H}_\lambda\mathbf{V}$ and $\mathbf{M}^{-1} = \mathbf{V}^\top\hat{\mathbf{V}}$. The first step is to check that these two matrices are actually inverses:

$$(\mathbf{H}_\lambda\hat{\mathbf{V}})^+\mathbf{H}_\lambda\mathbf{V}\mathbf{V}^\top\hat{\mathbf{V}} = (\mathbf{H}_\lambda\hat{\mathbf{V}})^+\mathbf{H}_\lambda\hat{\mathbf{V}} = \mathbf{I}_r \ ,$$

where the last equality uses that $\mathfrak{s}_r(\mathbf{H}_\lambda\hat{\mathbf{V}}) > 0$. Now, since Lemma 5.2.1 implies that A induces the factorization $\mathbf{H}_\sigma = \mathbf{H}_\lambda\mathbf{V}\mathbf{A}_\sigma\mathbf{V}^\top$, by substitution we get $\tilde{\mathbf{A}}_\sigma = \mathbf{M}\mathbf{A}_\sigma\mathbf{M}^{-1}$. Furthermore, since the basis $\mathcal{B}$ is complete and $\mathbf{V}\mathbf{V}^\top$ acts as an identity on the span of the rows of $\mathbf{H}$, we have $\boldsymbol{\alpha}_0^\top\mathbf{M}^{-1} = \mathbf{h}_{\lambda,\mathcal{S}}^\top\mathbf{V}\mathbf{V}^\top\hat{\mathbf{V}} = \mathbf{h}_{\lambda,\mathcal{S}}^\top\hat{\mathbf{V}} = \tilde{\boldsymbol{\alpha}}_0^\top$. Finally, using $\mathbf{H}_\lambda\mathbf{V} = \mathbf{U}\boldsymbol{\Lambda}$ we see that

$$\begin{aligned}\mathbf{M}\boldsymbol{\alpha}_\infty &= (\mathbf{H}_\lambda\hat{\mathbf{V}})^+\mathbf{H}_\lambda\mathbf{V}(\mathbf{H}_\lambda\mathbf{V})^+\mathbf{h}_{\mathcal{P},\lambda} = (\mathbf{H}_\lambda\hat{\mathbf{V}})^+\mathbf{U}\mathbf{U}^\top\mathbf{h}_{\mathcal{P},\lambda} \\ &= (\mathbf{H}_\lambda\hat{\mathbf{V}})^+\mathbf{h}_{\mathcal{P},\lambda} = \tilde{\boldsymbol{\alpha}}_\infty \ .\end{aligned}$$

□

Now suppose that $B = \langle \boldsymbol{\beta}_0, \boldsymbol{\beta}_\infty, \{\mathbf{B}_\sigma\}\rangle$ is a probabilistic WFA with n states such that $f = f_B$. That is, f admits a probabilistic realization. Note that in general this may not be possible (see [DE08] for examples). In addition, we note that even when such a realization exists it is not necessarily minimal: it may happen that $n \geq r$. Now, assuming the existence of such B, we can define some more bits of notation.

Let $\mathbf{H}_f = \mathbf{P}_B \mathbf{S}_B$ be the factorization induced by B. Note that if $n > r$ this is not a rank factorization, but $\mathbf{P}_B \in \mathbb{R}^{\mathcal{P}\times n}$ and $\mathbf{S}_B \in R^{n\times\mathcal{S}}$ with $\operatorname{rank}(\mathbf{P}_B) = \operatorname{rank}(\mathbf{S}_B) = r$. Let us write $\mathbf{V}_B^\top \in \mathbb{R}^{r\times n}$ for the right singular vectors of the forward matrix $\mathbf{P}_B$. Since $\mathbf{P}_B \mathbf{V}_B \mathbf{V}_B^\top = \mathbf{P}_B$, then by Lemma 5.2.3 we have $f = f_{B'}$ where $B' = \mathbf{V}_B^\top B \mathbf{V}_B$. It is immediate to note that now B' is a minimal WFA for f. Thus, under the conditions of Lemma 6.3.1 we can apply Corollary 5.2.2 to find a matrix $\hat{\mathbf{M}} \in \mathbb{R}^{r\times r}$ such that $B' = \hat{\mathbf{M}}^{-1}\tilde{A}\hat{\mathbf{M}}$. Using the change of basis $\hat{\mathbf{M}}$ and the projection $\mathbf{V}_B$ we define the following two WFA with n states:

$$\tilde{B} = \mathbf{V}_B \hat{\mathbf{M}}^{-1} \tilde{A} \hat{\mathbf{M}} \mathbf{V}_B^\top \ ,$$
$$\hat{B} = \mathbf{V}_B \hat{\mathbf{M}}^{-1} \hat{A} \hat{\mathbf{M}} \mathbf{V}_B^\top \ .$$

Since $\mathbf{V}_B^\top \mathbf{V}_B = \mathbf{I}_r$, one can immediately check that $f = f_{\tilde{B}}$ and $\hat{f} = f_{\hat{B}}$.

Note that by the definition of $\tilde{A}$ matrix $\hat{\mathbf{M}}$ depends on $\hat{\mathbf{V}}$. We can make analogous definitions which only depend on f, B and $\mathcal{B}$ by using a matrix $\mathbf{M}$ such that $B' = \mathbf{M}^{-1} A \mathbf{M}$. The next result characterizes a useful relation between such an $\mathbf{M}$ and our $\hat{\mathbf{M}}$.

Lemma 6.3.2. *Suppose that $\|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2 \leq \epsilon/(1+\epsilon)\mathfrak{s}_r(\mathbf{H}_\lambda)$ for some $\epsilon < 1$. Then for any $1 \leq i \leq r$ we have $\mathfrak{s}_i(\mathbf{M})\sqrt{1-\epsilon^2} \leq \mathfrak{s}_i(\hat{\mathbf{M}}) \leq \mathfrak{s}_i(\mathbf{M})$.*

Proof. Note that by Lemma 6.3.1 we have $f = f_A = f_{\tilde{A}}$. Thus using Corollary 5.2.2 it is easy to see that $A = (\hat{\mathbf{V}}^\top \mathbf{V})^{-1} \tilde{A} (\hat{\mathbf{V}}^\top \mathbf{V})$. A simple composition then yields $\hat{\mathbf{M}} = \hat{\mathbf{V}}^\top \mathbf{V} \mathbf{M}$. Hence, by applying Lemma A.2.6 and the same bound used in the proof of Lemma 6.3.1 we get

$$\begin{aligned}
\mathfrak{s}_i(\hat{\mathbf{M}}) &= \mathfrak{s}_i(\hat{\mathbf{V}}^\top \mathbf{V} \mathbf{M}) \\
&\geq \mathfrak{s}_i(\mathbf{M}) \mathfrak{s}_r(\hat{\mathbf{V}}^\top \mathbf{V}) \\
&\geq \mathfrak{s}_i(\mathbf{M}) \sqrt{1-\epsilon^2} \ .
\end{aligned}$$

The second inequality follows from Lemma A.2.6 as well, since:

$$\begin{aligned}
\mathfrak{s}_i(\hat{\mathbf{M}}) &= \mathfrak{s}_i(\hat{\mathbf{V}}^\top \mathbf{V} \mathbf{M}) \\
&\leq \mathfrak{s}_i(\mathbf{M}) \mathfrak{s}_1(\hat{\mathbf{V}}^\top \mathbf{V}) \\
&= \mathfrak{s}_i(\mathbf{M}) \|\hat{\mathbf{V}}^\top \mathbf{V}\|_2 \\
&\leq \mathfrak{s}_i(\mathbf{M}) \ .
\end{aligned}$$

□

Before we state the main lemma in this section, we will define yet another piece of notation which will greatly simplify our calculations. We let $\tilde{\mathcal{B}}$ be the basis obtained as the p-closure of $(\mathcal{P} \cup \{\lambda\}, \mathcal{S} \cup \{\lambda\})$. If $\mathbf{H}$ denotes the sub-block of $\mathbf{H}_f$ corresponding to $\tilde{\mathcal{B}}$, it is immediate to see that $\mathbf{H}_\lambda$, $\mathbf{H}_\sigma$ for all $\sigma \in \Sigma$, $\mathbf{h}_{\mathcal{P},\lambda}$, $\mathbf{h}_{\lambda,\mathcal{S}}$ are all contained as sub-blocks in $\mathbf{H}$. In this case we get the following bounds, which follow from a simple property of Frobenius norm:

$$\|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_F, \|\mathbf{H}_\sigma - \hat{\mathbf{H}}_\sigma\|_F, \|\mathbf{h}_{\mathcal{P},\lambda} - \hat{\mathbf{h}}_{\mathcal{P},\lambda}\|_2, \|\mathbf{h}_{\lambda,\mathcal{S}} - \hat{\mathbf{h}}_{\lambda,\mathcal{S}}\|_2 \leq \|\mathbf{H} - \hat{\mathbf{H}}\|_F \ .$$

Lemma 6.3.3. *There exists a universal constant C such that for any $t \geq 0$ and $\varepsilon > 0$,*

$$\|\mathbf{H} - \hat{\mathbf{H}}\|_F \leq C \frac{\varepsilon \mathfrak{s}_r(\mathbf{M}) \mathfrak{s}_r(\mathbf{H}_\lambda)^2}{(t+1)^2 |\Sigma| \mathfrak{s}_1(\mathbf{M}) \sqrt{n c_{\tilde{\mathbf{B}}}}} \ ,$$

implies $\sum_{x\in\Sigma^t} |f(x) - \hat{f}(x)| \leq \varepsilon/(t+1)$.

Proof. The main idea of the proof is to use the representation $f = f_{\tilde{B}}$ and $\hat{f} = f_{\hat{B}}$ and apply Lemmas 5.4.4 and 5.4.5. The first step is to see that we can take $\gamma_\Sigma = \gamma_k = \gamma_\infty = 1$ in Lemma 5.4.4.

We begin with γ_k. Note that we have by construction $\tilde{B} = \mathbf{V}_B \mathbf{V}_B^\top B \mathbf{V}_B \mathbf{V}_B^\top$. Since $\mathbf{V}_B \mathbf{V}_B^\top$ acts as an identity on the rows of $\mathbf{P}_B$, we have

$$\tilde{\boldsymbol{\beta}}_0 \tilde{\mathbf{B}}_x = \boldsymbol{\beta}_0 \mathbf{B}_x \mathbf{V}_B \mathbf{V}_B^\top = \boldsymbol{\beta}_0 \mathbf{B}_x$$

for any $x \in \Sigma^\star$. For probabilistic B it is easy to see that $\|\boldsymbol{\beta}_0 \mathbf{B}_x\|_1 = \boldsymbol{\beta}_0 \mathbf{B}_x \mathbf{1} = \mathbb{P}[x\Sigma^\star]$. Thus, for any k we have

$$\gamma_k = \sum_{x \in \Sigma^k} \|\tilde{\boldsymbol{\beta}}_0 \tilde{\mathbf{B}}_x\|_1 = \mathbb{P}[|X| \geq k] \leq 1 \ .$$

Next we turn our attention to γ_Σ. Since $\mathbf{V}_B^\top \mathbf{V}_B \mathbf{V}_B^\top = \mathbf{V}_B^\top$ by definition, and for any $\mathbf{v} \in \mathbb{R}^r$ we have $\mathbf{v}^\top \mathbf{V}_B^\top \mathbf{B}_\sigma \mathbf{V}_B \mathbf{V}_B^\top = \mathbf{v}^\top \mathbf{V}_B^\top \mathbf{B}_\sigma$ because $\mathbf{v}^\top \mathbf{V}_B^\top \mathbf{B}_\sigma$ is in the span of the rows of $\mathbf{P}_B$, we get

$$\begin{aligned}(\tilde{\boldsymbol{\beta}}_0 \tilde{\mathbf{B}}_x - \hat{\boldsymbol{\beta}}_0 \hat{\mathbf{B}}_x) \tilde{\mathbf{B}}_\sigma &= (\tilde{\boldsymbol{\alpha}}_0 \tilde{\mathbf{A}}_x \hat{\mathbf{M}} \mathbf{V}_B^\top - \hat{\boldsymbol{\alpha}}_0 \hat{\mathbf{A}}_x \hat{\mathbf{M}} \mathbf{V}_B^\top) \mathbf{V}_B \mathbf{V}_B^\top \mathbf{B}_\sigma \mathbf{V}_B \mathbf{V}_B^\top \\ &= (\tilde{\boldsymbol{\alpha}}_0 \tilde{\mathbf{A}}_x \hat{\mathbf{M}} \mathbf{V}_B^\top - \hat{\boldsymbol{\alpha}}_0 \hat{\mathbf{A}}_x \hat{\mathbf{M}} \mathbf{V}_B^\top) \mathbf{B}_\sigma \ .\end{aligned}$$

Using this in (5.1) from the proof of Lemma 5.4.3 we see that we can take γ_Σ to be such that $\sum_{\sigma \in \Sigma} |\mathbf{B}_\sigma| \mathbf{1} \leq \gamma_\Sigma \mathbf{1}$, which is satisfied by $\gamma_\Sigma = 1$ because B is probabilistic. Using similar arguments, we can see that

$$(\tilde{\boldsymbol{\beta}}_0 \tilde{\mathbf{B}}_x - \hat{\boldsymbol{\beta}}_0 \hat{\mathbf{B}}_x) \tilde{\boldsymbol{\beta}}_\infty = (\tilde{\boldsymbol{\alpha}}_0 \tilde{\mathbf{A}}_x \hat{\mathbf{M}} \mathbf{V}_B^\top - \hat{\boldsymbol{\alpha}}_0 \hat{\mathbf{A}}_x \hat{\mathbf{M}} \mathbf{V}_B^\top) \boldsymbol{\beta}_\infty \ .$$

Plugging this into (5.3) from the proof of Lemma 5.4.4, we see that we can take $\gamma_\infty = \|\boldsymbol{\beta}_\infty\|_\infty \leq 1$.

The next step will be to bound ρ_0, ρ_∞, and ρ_Σ using Lemma 5.4.5. This is done in the following series of inequalities, which follow from properties of vector norms:

$$\begin{aligned}\rho_0 &= \|\tilde{\boldsymbol{\beta}}_0 - \hat{\boldsymbol{\beta}}_0\|_1 = \|(\tilde{\boldsymbol{\alpha}}_0 - \hat{\boldsymbol{\alpha}}_0) \hat{\mathbf{M}} \mathbf{V}_B^\top\|_1 \leq \sqrt{n} \|(\tilde{\boldsymbol{\alpha}}_0 - \hat{\boldsymbol{\alpha}}_0) \hat{\mathbf{M}} \mathbf{V}_B^\top\|_2 \\ &\leq \sqrt{n} \|\hat{\mathbf{M}}\|_2 \|\tilde{\boldsymbol{\alpha}}_0 - \hat{\boldsymbol{\alpha}}_0\|_2 \leq \sqrt{n} \mathfrak{s}_1(\hat{\mathbf{M}}) \|\mathbf{h}_{\lambda,\mathcal{S}} - \hat{\mathbf{h}}_{\lambda,\mathcal{S}}\|_2 \ ,\end{aligned}$$

$$\begin{aligned}\rho_\infty &= \|\tilde{\boldsymbol{\beta}}_\infty - \hat{\boldsymbol{\beta}}_\infty\|_\infty = \|\mathbf{V}_B \hat{\mathbf{M}}^{-1} (\tilde{\boldsymbol{\alpha}}_\infty - \hat{\boldsymbol{\alpha}}_\infty)\|_\infty \\ &\leq \|\mathbf{V}_B \hat{\mathbf{M}}^{-1} (\tilde{\boldsymbol{\alpha}}_\infty - \hat{\boldsymbol{\alpha}}_\infty)\|_2 \leq \|\hat{\mathbf{M}}^{-1}\|_2 \|\tilde{\boldsymbol{\alpha}}_\infty - \hat{\boldsymbol{\alpha}}_\infty\|_2 \\ &\leq \frac{1}{\mathfrak{s}_r(\hat{\mathbf{M}})} \left(\frac{\|\mathbf{h}_{\mathcal{P},\lambda} - \hat{\mathbf{h}}_{\mathcal{P},\lambda}\|_2}{\mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}})} + \frac{1+\sqrt{5}}{2} \frac{\|\hat{\mathbf{h}}_{\mathcal{P},\lambda}\|_2 \|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2}{\min\{\mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}})^2, \mathfrak{s}_r(\hat{\mathbf{H}}_\lambda \hat{\mathbf{V}})^2\}} \right) \ ,\end{aligned}$$

$$\begin{aligned}\rho_\sigma &= \|\tilde{\mathbf{B}}_\sigma - \hat{\mathbf{B}}_\sigma\|_\infty = \|\mathbf{V}_B \hat{\mathbf{M}}^{-1} (\tilde{\mathbf{A}}_\sigma - \hat{\mathbf{A}}_\sigma) \hat{\mathbf{M}} \mathbf{V}_B^\top\|_\infty \\ &\leq \sqrt{n} \|\mathbf{V}_B \hat{\mathbf{M}}^{-1} (\tilde{\mathbf{A}}_\sigma - \hat{\mathbf{A}}_\sigma) \hat{\mathbf{M}} \mathbf{V}_B^\top\|_2 \leq \sqrt{n} \|\hat{\mathbf{M}}^{-1}\|_2 \|\hat{\mathbf{M}}\|_2 \|\tilde{\mathbf{A}}_\sigma - \hat{\mathbf{A}}_\sigma\|_2 \\ &\leq \frac{\sqrt{n} \mathfrak{s}_1(\hat{\mathbf{M}})}{\mathfrak{s}_r(\hat{\mathbf{M}})} \left(\frac{\|\mathbf{H}_\sigma - \hat{\mathbf{H}}_\sigma\|_2}{\mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}})} + \frac{1+\sqrt{5}}{2} \frac{\|\hat{\mathbf{H}}_\sigma\|_2 \|\mathbf{H}_\lambda - \hat{\mathbf{H}}_\lambda\|_2}{\min\{\mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}})^2, \mathfrak{s}_r(\hat{\mathbf{H}}_\lambda \hat{\mathbf{V}})^2\}} \right) \ .\end{aligned}$$

We now simplify this expressions as follows. First write $\rho = \|\mathbf{H} - \hat{\mathbf{H}}\|_F$ and note that by the observation in equation (6.3) and Lemmas 6.3.1 and 6.3.2 choosing $\epsilon = \sqrt{3}/2$ we have that $\rho \leq \sqrt{3}/(2+\sqrt{3}) \min\{\mathfrak{s}_r(\mathbf{H}_\lambda), \mathfrak{s}_r(\mathbf{M})\}$ implies $\mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}}) \geq \mathfrak{s}_r(\mathbf{H}_\lambda)/2$ and $\mathfrak{s}_r(\hat{\mathbf{M}}) \geq \mathfrak{s}_r(\mathbf{M})/2$. Furthermore, using Lemmas 6.3.1 and A.2.1 we get

$$\begin{aligned}\mathfrak{s}_r(\hat{\mathbf{H}}_\lambda \hat{\mathbf{V}}) &\geq \mathfrak{s}_r(\mathbf{H}_\lambda \hat{\mathbf{V}}) - \|\mathbf{H}_\lambda \hat{\mathbf{V}} - \hat{\mathbf{H}}_\lambda \hat{\mathbf{V}}\|_2 \\ &\geq \frac{\mathfrak{s}_r(\mathbf{H}_\lambda)}{2} - \rho \\ &\geq \frac{\mathfrak{s}_r(\mathbf{H}_\lambda)}{2} - \frac{\sqrt{3} \mathfrak{s}_r(\mathbf{H}_\lambda)}{2+\sqrt{3}} \\ &= \frac{(2-\sqrt{3})^2}{2} \mathfrak{s}_r(\mathbf{H}_\lambda) \ .\end{aligned}$$

Now observe that since $\hat{\mathbf{h}}_{\mathcal{P},\lambda}$ is a vector of empirical probabilities of different strings we have $\|\hat{\mathbf{h}}_{\mathcal{P},\lambda}\|_2 \leq \|\hat{\mathbf{h}}_{\mathcal{P},\lambda}\|_1 \leq 1$. In addition, we have $\|\hat{\mathbf{H}}_\sigma\|_2 \leq \|\hat{\mathbf{H}}_\sigma\|_F \leq \|\hat{\mathbf{H}}\|_F \leq \sqrt{c_{\tilde{\mathcal{B}}}}$ by the same argument used in the proof of Theorem 6.2.1. Using all these observations, we see that the following hold for some constants C_0, C_∞, and C_Σ:

$$\begin{aligned}
\rho_0 &\leq C_0 \sqrt{n} \mathfrak{s}_1(\mathbf{M}) \rho \ , \\
\rho_\infty &\leq C_\infty \frac{\rho}{\mathfrak{s}_r(\mathbf{M}) \mathfrak{s}_r(\mathbf{H}_\lambda)^2} \ , \\
\rho_\Sigma &\leq C_\Sigma \frac{\sqrt{n}\sqrt{c_{\tilde{\mathbf{B}}}} \mathfrak{s}_1(\mathbf{M}) |\Sigma| \rho}{\mathfrak{s}_r(\mathbf{M}) \mathfrak{s}_r(\mathbf{H}_\lambda)^2} \ .
\end{aligned}$$

Finally note that if for some suitable constants C_0', C_∞', and C_Σ' we have

$$\rho \leq \frac{\varepsilon}{(t+1)^2} \min \left\{ \frac{C_0'}{\sqrt{n}\mathfrak{s}_1(\mathbf{M})}, C_\infty' \mathfrak{s}_r(\mathbf{M}) \mathfrak{s}_r(\mathbf{H}_\lambda)^2, \frac{C_\Sigma' \mathfrak{s}_r(\mathbf{M}) \mathfrak{s}_r(\mathbf{H}_\lambda)^2}{\sqrt{n}\sqrt{c_{\tilde{\mathcal{B}}}} |\Sigma| \mathfrak{s}_1(\mathbf{M})} \right\} \ ,$$

then by Lemma 5.4.4 we get the following bound:

$$\begin{aligned}
\sum_{x \in \Sigma^t} |f(x) - \hat{f}(x)| &\leq (1+\rho_0)(1+\rho_\Sigma)^t (1+\rho_\infty) - 1 \\
&\leq \left(1 + \frac{\varepsilon}{2(t+1)(t+2)}\right)^{t+2} - 1 \\
&\leq \frac{\varepsilon}{t+1} \ ,
\end{aligned}$$

where we used that $(1+x/t)^t \leq 1 + 2x$ for $x \leq 1/2$ and any $t \geq 0$. The result follows by choosing an adequate C. □

Combining the above lemma with the result from Section 6.2 about the estimation error in an empirical Hankel matrix $\hat{\mathbf{H}}$, we immediately get the following result.

Theorem 6.3.4. *There exists a universal constant C such that for any $t \geq 0$, $\varepsilon > 0$, $\delta > 0$, if Algorithm 5 is given a sample of size*

$$m \geq C \frac{t^4 n c_{\tilde{\mathcal{B}}}^2 |\Sigma|^2 \mathfrak{s}_1(\mathbf{M})^2}{\varepsilon^2 \mathfrak{s}_r(\mathbf{M})^2 \mathfrak{s}_r(\mathbf{H}_\lambda)^4} \log\left(\frac{1}{\delta}\right) \ ,$$

then with probability at least $1-\delta$ the hypothesis $\hat{A}$ returned by the algorithm satisfies $\sum_{|x| \leq t} |f(x) - \hat{f}(x)| \leq \varepsilon$.

At this point, we must make three remarks about the above theorem. The first is that since f is computed by a probabilistic automaton, then there exists a constant c_f such that $\mathbb{P}[|x| \geq t] \leq \exp(-c_f t)$ for all t. Thus, in order to cover all but an η fraction of the mass of f with a good approximation $\hat{f}$, one can choose $t = (1/c_f)\log(1/\eta)$ in Theorem 6.3.4.

The second observation is that the quantity $n\mathfrak{s}_1(\mathbf{M})^2/\mathfrak{s}_r(\mathbf{M})^2$ that appears in the bound depends on our choice of a probabilistic automaton B realizing f. Since this is only used in the analysis, we can restate the bounds in terms of an intrinsic quantity that depends only on f, defined as follows:

$$\theta_f = \min_B \frac{|B| \mathfrak{s}_1(\mathbf{M}_B)^2}{\mathfrak{s}_r(\mathbf{M}_B)^2} \ ,$$

where the minimum is over all probabilistic automata B such that $f_B = f$. With this intrinsic definition, we get a sample bound of the form

$$m \geq C \frac{t^4 \theta_f c_{\tilde{\mathcal{B}}}^2 |\Sigma|^2}{\varepsilon^2 \mathfrak{s}_r(\mathbf{H}_\lambda)^4} \log\left(\frac{1}{\delta}\right) \ .$$

Note that the quantity $\theta_f c_{\mathcal{B}}^2/\mathfrak{s}_r(\mathbf{H}_\lambda)^4$ in this bound still depends on the user's choice for $\mathcal{B}$. The problem of choosing a basis given a sample will be addressed in the next section.

The last remark is that unfortunately the bound on the error given in Theorem 6.3.4 does not specify what happens in the tail of the distribution. A lack of control on the behavior of $\sum_{|x|>t} |\hat{f}(x)|$ is the main obstruction to obtaining a bound on $\sum_{x\in\Sigma^\star} |f(x) - \hat{f}(x)|$. We know that $\sum_{x\in\Sigma^\star} \hat{f}(x)$ will converge to $\hat{\boldsymbol{\alpha}}_0(\mathbf{I} - \hat{\mathbf{A}})^{-1}\hat{\boldsymbol{\alpha}}_\infty$ if $\|\hat{\mathbf{A}}\|_2 < 1$. And since $\|\mathbf{A}\|_2 < 1$ because f is a probability distribution, it is easy to see that $\hat{f}$ will be convergent if $\rho_\Sigma \leq (1 - \|\mathbf{A}\|_2)/2$. However, to use a trick like the one used in Lemma 1.4.2 we need to control the *absolute* converge of this series, which turns out to be a difficult problem. In particular, if we could guarantee that $\hat{f}(x) \geq 0$ for all x, then we would immediately have absolute convergence at an exponential rate. However, it turns out that given a WFA, deciding whether $f_A(x) \geq 0$ for all x is in general an undecidable problem [SS78; DE08], and it is only semi-decidable whether f_A is absolutely convergent [BD11]. An alternative approach relying on the notion of *joint spectral radius* [Jun09] was used in [Bai11a] to control the asymptotic convergence rate of the tail in a similar analysis; this technique, however, does not provide control over the constants involved in the bounds.

6.4 Finding a Complete Basis

So far our results have assumed that we know a complete basis for the target WFA to be learned. In practice the user needs to specify a basis to the algorithm, either based on domain knowledge, some indications in the available data, or trial-and-error heuristics. A common approach taken in some works is to use basis that contains all possible strings up to a certain length. However, unless this length equals the rank of the target one cannot guarantee that the basis will be complete in general. This brute-force approach yields basis of size $O(|\Sigma|^{\mathrm{rank}(f)})$, which may be very large. In contrast, it is not difficult to show that basis of size $\mathrm{rank}(f)$ always exist, though it may be difficult to find without exact knowledge of f.

In this section we give a procedure to find a data-dependent basis which is guaranteed to succeed under some assumptions. In particular, we show that a simple randomized strategy for choosing a basis succeeds with high probability. Furthermore, our result gives bounds on the number of examples required for finding a basis that depend polynomially on some parameters of the target function $f : \Sigma^\star \to \mathbb{R}$ and the sampling distribution D, which may in general be different.

We begin with a well-known folklore result about the existence of minimal basis. This implies that in principle all methods for learning WFA from sub-blocks of the Hankel matrix can work with a block whose size is only quadratic in the number of states of the target.

Proposition 6.4.1. *For any $f : \Sigma^\star \to \mathbb{R}$ of rank r there exists a basis $(\mathcal{P}, \mathcal{S})$ of f with $|\mathcal{P}| = |\mathcal{S}| = r$.*

Proof. Take any rank factorization of $\mathbf{H}_f = \mathbf{P}_f\mathbf{S}_f$. Since $\mathrm{rank}(\mathbf{P}_f) = r$, there exist r linearly independent rows in $\mathbf{P}_f$. Let $\mathbf{P} \in \mathbb{R}^{r\times r}$ be the sub-matrix of $\mathbf{P}_f$ containing these rows and $\mathcal{P}$ the prefixes defining them. Similarly, build a matrix $\mathbf{S} \in \mathbb{R}^{r\times r}$ with r linearly independent columns of $\mathbf{S}_f$ indexed by a set of suffixes $\mathcal{S}$. Then $(\mathcal{P}, \mathcal{S})$ form a basis since $\mathbf{H} = \mathbf{PS}$ and $\mathrm{rank}(\mathbf{H}) = \mathrm{rank}(\mathbf{S}) = r$ because $\mathbf{P}$ has full row-rank. □

A WFA $A = \langle \boldsymbol{\alpha}_0^\top, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ is called *strongly bounded* if $\|\mathbf{A}_\sigma\| \leq 1$ for all $\sigma \in \Sigma^\star$. Note that this implies the boundness of f_A since

$$|f_A(x)| = |\boldsymbol{\alpha}_0^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty| \leq \|\boldsymbol{\alpha}_0\|\|\boldsymbol{\alpha}_\infty\| \ .$$

A function over strings f of finite rank is called *strongly bounded* if there exists a strongly bounded minimal WFA for f. Note that, in particular, all models of probabilistic automata are strongly bounded.

The following result states that, under some simple hypothesis, with high probability Algorithm 6 will return a correct basis. Our proof identifies a parameter depending on f and D that quantifies the sample size required to get a complete basis. The proof relies on a result on random matrix theory that can be found in Appendix A.1. It basically gives conditions for the empirical covariance matrix of a multidimensional real random variable to be full-rank.

Algorithm 6: Random Basis

Input: strings $S = (x^1, \ldots, x^m)$
Output: basis candidate $(\mathcal{P}, \mathcal{S})$

Initialize $\mathcal{P} \leftarrow \varnothing$, $\mathcal{S} \leftarrow \varnothing$;
for $i = 1$ **to** m **do**
  Choose $0 \leq t \leq |x^i|$ uniformly at random;
  Split $x^i = u^i v^i$ with $|u^i| = t$ and $|v^i| = |x^i| - t$;
  Add u^i to $\mathcal{P}$ and v^i to $\mathcal{S}$;

Theorem 6.4.2. *Let $f : \Sigma^\star \to \mathbb{R}$ be a strongly bounded function of rank r and D a distribution over $\Sigma^\star$ with full support.*[1] *Suppose that m strings sampled i.i.d. from D are given to Algorithm 6. Then, if $m \geq C\eta \log(1/\delta)$ for some universal constant C and a parameter η that depends on f and D, the output $(\mathcal{P}, \mathcal{S})$ is a basis for f with probability at least $1 - \delta$.*

Proof. We begin with some notation. Consider the prefixes produced by Algorithm 6 on input an i.i.d. random sample $S = (x^1, \ldots, x^m)$ drawn from D. We write $\mathcal{P} = (u^1, \ldots, u^m)$ for the *tuple* of prefixes produced by the algorithm and use $\mathcal{P}'$ to denote the *set* defined by these prefixes. We define $\mathcal{S}$ and $\mathcal{S}'$ similarly. Let $p' = |\mathcal{P}'|$ and $s' = |\mathcal{S}'|$. Our goal is to show that the random sub-block $\mathbf{H}' \in \mathbb{R}^{p' \times s'}$ of $\mathbf{H}_f$ defined by the output of Algorithm 6 has rank r with high probability with respect to the choices of input sample and splitting points. Our strategy will be to show that one always has $\mathbf{H}' = \mathbf{P}'\mathbf{S}'$, where $\mathbf{P}' \in \mathbb{R}^{p' \times r}$ and $\mathbf{S}' \in \mathbb{R}^{r \times s'}$ are such that with high probability $\operatorname{rank}(\mathbf{P}') = \operatorname{rank}(\mathbf{S}') = r$. The arguments are identical for $\mathbf{P}'$ and $\mathbf{S}'$.

Fix a strongly bounded minimal WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$ for f, and let $\mathbf{H}_f = \mathbf{P}_f \mathbf{S}_f$ denote the rank factorization induced by A. We write $\mathbf{p}_u^\top$ for the uth row of $\mathbf{P}_f$. Note that since A is strongly bounded we have $\|\mathbf{p}_u^\top\| = \|\boldsymbol{\alpha}_0^\top \mathbf{A}_u\| \leq \|\boldsymbol{\alpha}_0^\top\|$. The desired $\mathbf{P}'$ will be the sub-block of $\mathbf{P}_f$ corresponding to the prefixes in $\mathcal{P}'$. In the following we bound the probability that this matrix is rank deficient.

The first step is to characterize the distribution of the elements of $\mathcal{P}$. Since the prefixes u^i are all i.i.d., we write D_p to denote the distribution from which these prefixes are drawn, and observe that for any $u \in \Sigma^\star$ and any $1 \leq i \leq m$ we have $D_p(u) = \mathbb{P}[u^i = u] = \mathbb{P}[\exists\, v : x^i = uv \wedge t = |u|]$, where x^i is drawn from D and t is uniform in $[0, |x^i|]$. Thus we see that $D_p(u) = \sum_{v \in \Sigma^\star} (1 + |uv|)^{-1} D(uv)$.

Now we overload our notation and let D_p also denote the following distribution over $\mathbb{R}^r$ supported on the set of all rows of $\mathbf{P}_f$:

$$D_p(\mathbf{q}^\top) = \sum_{u : \mathbf{p}_u^\top = \mathbf{q}^\top} D_p(u) \ .$$

It follows from this definition that the covariance matrix of D_p satisfies $\mathbf{C}_p = \mathbb{E}[\mathbf{q}\mathbf{q}^\top] = \sum_u D_p(u) \mathbf{p}_u \mathbf{p}_u^\top$. Observe that this expression can be written in matrix form as $\mathbf{C}_p = \mathbf{P}_f^\top \mathbf{D}_p \mathbf{P}_f$, where $\mathbf{D}_p$ is a bi-infinite diagonal matrix whose with entries $\mathbf{D}_p(u, u) = D_p(u)$. We say that the distribution D is *pref-adversarial* for A if $\operatorname{rank}(\mathbf{C}_p) < r$. Note that if $D(x) > 0$ for all $x \in \Sigma^\star$, then D_p has full-rank and consequently $\operatorname{rank}(\mathbf{C}_p) = r$. This shows that distributions with full support are never pref-adversarial, and thus we can assume that $\mathbf{C}_p$ has full rank.[2]

Next we use the prefixes in $\mathcal{P}$ to build a matrix $\mathbf{P} \in \mathbb{R}^{m \times r}$ whose ith row corresponds to the u^ith row $\mathbf{p}_{u^i}^\top$ of $\mathbf{P}_f$. It is immediate to see that $\mathbf{P}'$ can be obtained from $\mathbf{P}$ by possibly removing some repeated rows and reordering the remaining ones. Thus we have $\operatorname{rank}(\mathbf{P}) = \operatorname{rank}(\mathbf{P}')$. Furthermore, by construction we have that $\hat{\mathbf{C}}_p = \mathbf{P}^\top \mathbf{P}$ is the sample covariance matrix of m vectors in $\mathbb{R}^r$ drawn i.i.d. from D_p. Therefore, a straightforward application of Theorem A.1.7 shows that if $m \geq C(\kappa(\mathbf{C}_p)/\mathfrak{s}_r(\mathbf{C}_p))\|\boldsymbol{\alpha}_0^\top\|^2 \log(1/\delta)$, then $\operatorname{rank}(\mathbf{P}') = r$ with probability at least $1 - \delta$. Here C is a universal constant, $\kappa(\mathbf{C}_p)$ is the condition

[1] The theorem also holds under a weaker assumption on D, see the proof for details.

[2] This justifies the assumption in the statement of Theorem 6.4.2. Note however that our arguments also work under the weaker assumption that there exists a minimal strongly bounded WFA A for f such that D is neither pref-adversarial nor suff-adversarial for A (with the obvious definitions).

number of $\mathbf{C}_p$, and $\mathfrak{s}_r(\mathbf{C}_p)$ is the smallest singular value of $\mathbf{C}_p$, where these last two terms depend on A and D.

The result follows by symmetry from a union bound. Furthermore, we can take

$$\eta = \eta(f, D) = \inf_A \max\left\{ \frac{\kappa(\mathbf{C}_p)\|\boldsymbol{\alpha}_0^\top\|^2}{\mathfrak{s}_r(\mathbf{C}_p))}, \frac{\kappa(\mathbf{C}_s)\|\boldsymbol{\alpha}_\infty\|^2}{\mathfrak{s}_r(\mathbf{C}_s))} \right\} \ ,$$

where the infimum is taken over all minimal strongly bounded WFA for f. □

Now we show how to compute the $\eta(f, D)$ appearing in the bound in a particular case. Suppose f is stochastic and let $D = f$ so that the sample comes from the distribution we are trying to learn. Furthermore, suppose $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ is a minimal WFA for f which is also irredundant. We begin by computing the probability $D_p(u)$ that a particular prefix $u \in \Sigma^\star$ with $|u| = t$ will appear in $\mathcal{P}$.

$$\begin{aligned} D_p(u) &= \sum_{v\in\Sigma^\star} \frac{1}{1+|u|+|v|} D(uv) \\ &= \sum_{k\geq 0} \frac{1}{1+t+k} D(u\Sigma^k) \\ &= \boldsymbol{\alpha}_0^\top \mathbf{A}_u \left(\sum_{k\geq 0} \frac{1}{1+t+k} \mathbf{A}^k \right) \boldsymbol{\alpha}_\infty \\ &= \boldsymbol{\alpha}_0^\top \mathbf{A}_u {}_2F_1(1, 1+t; 2+t; \mathbf{A}) \boldsymbol{\alpha}_\infty \ , \end{aligned}$$

where ${}_2F_1(1, 1+t; 2+t; \mathbf{A})$ is a *Gaussian hypergeometric function* evaluated on $\mathbf{A}$, which is guaranteed to converge because $\|\mathbf{A}\| < 1$ for some submultiplicative norm. Then, writing $\mathbf{p}_u^\top = \boldsymbol{\alpha}_0^\top \mathbf{A}_u$ we have

$$\mathbf{C}_p(i, j) = \sum_{u\in\Sigma^\star} \mathbf{p}_u(i)\mathbf{p}_u(j) D_p(u) \ .$$

Now note that $\mathbf{p}_u(i)\mathbf{p}_u(j)$ is the (i, j) coordinate of matrix $\mathbf{C}_{p,u} = \mathbf{p}_u\mathbf{p}_u^\top = \mathbf{A}_u^\top \boldsymbol{\alpha}_0 \boldsymbol{\alpha}_0^\top \mathbf{A}_u$. Hence, we get the following identity:

$$\mathbf{C}_p = \sum_{u\in\Sigma^\star} \mathbf{C}_{p,u} \boldsymbol{\alpha}_0^\top \mathbf{A}_u {}_2F_1(1, 1+|u|; 2+|u|; \mathbf{A}) \boldsymbol{\alpha}_\infty \ .$$

Similarly for suffixes, we obtain the expression

$$\mathbf{C}_s = \sum_{v\in\Sigma^\star} \mathbf{C}_{s,v} \boldsymbol{\alpha}_0^\top {}_2F_1(1, 1+|v|; 2+|v|; \mathbf{A}) \mathbf{A}_v \boldsymbol{\alpha}_\infty \ ,$$

where $\mathbf{C}_{s,v} = \mathbf{A}_v \boldsymbol{\alpha}_\infty \boldsymbol{\alpha}_\infty^\top \mathbf{A}_v^\top$. Therefore, given a WFA A for f we can compute the following upper bound for $\eta(f, f)$:

$$\eta(f, f) \leq \max\left\{ \frac{\|\mathbf{C}_p\|\|\boldsymbol{\alpha}_0\|^2}{\mathfrak{s}_r(\mathbf{C}_p)^2}, \frac{\|\mathbf{C}_s\|\|\boldsymbol{\alpha}_\infty\|^2}{\mathfrak{s}_r(\mathbf{C}_s)^2} \right\} \ .$$

We note here that the proof of Theorem 6.4.2 shows that a sufficient condition for $\mathfrak{s}_r(\mathbf{C}_p), \mathfrak{s}_r(\mathbf{C}_s) > 0$ is that f has full support. Identifying further sufficient conditions and other bounds on η is an open problem.

Chapter 7

Learning Transductions under Benign Input Distributions

Besides the straightforward application of the spectral method to PNFA given in previous chapter, learning algorithms for more complex models over sequences can be derived using the same principles. In this chapter we give a spectral learning algorithm for probabilistic transductions defining conditional distributions over pairs of strings. The focus will be on obtaining finite-sample bounds making as little assumptions as possible on the probability distribution that generates the input strings in the training sample. We will show that a spectral technique can be used to learn a class of probabilistic finite state transducers defining such conditional distributions.

The most general PAC learning result would be in a distribution-free setting where the distribution on inputs is not constrained in any way. That would mean that transducers can be learned under any input distribution, which in general is a hard problem because it contains as a subproblem that of learning all DFA under an arbitrary input distribution. In order to overcome this difficulty we will be forced to make some assumptions on the distribution of inputs. If for example we assume that it can be realized by a stochastic WFA, then the whole transducer would become a stochastic WFA and we could apply the results from previous chapter. However, since we want to be as distribution-independent as possible, we will not take this path. However, we will identify a couple of non-parametric assumptions on the input distribution which make it feasible to learn with a spectral algorithm.

7.1 Probabilistic Finite-State Transducers

We begin by introducing a model for probabilistic transductions over strings of the same length. This will correspond to a function $f : (\Sigma \times \Delta)^\star \to \mathbb{R}$ whose values $f(x, y)$ represent the conditional probability of generating output y given x as input; that is, $f(x, y) \geq 0$ for all $(x, y) \in (\Sigma \times \Delta)^\star$, and $\sum_{y \in \Delta^{|x|}} f(x, y) = 1$ for all $x \in \Sigma^\star$. Note that by definition $f(x, y)$ is not defined when $|x| \neq |y|$. Our main working hypothesis here will be that f can be computed by a *weighted transducer*, which is nothing else than a weighted automaton where transition operators are indexed over two finite alphabets Σ and Δ.[1] This means that we can write

$$f(x, y) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1}^{y_1} \cdots \mathbf{A}_{x_t}^{y_t} \boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x^y \boldsymbol{\alpha}_\infty \ ,$$

where $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$ and $\mathbf{A}_\sigma^\delta \in \mathbb{R}^{n \times n}$, n being the number of states in the transducer. We will thus write $f = f_A$, where $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma^\delta\} \rangle$. Like we did in Section 6.3, we shall assume throughout this chapter that the target transduction f admits a probabilistic realization. From the point of view of a probabilistic model, this formulation implies that the conditional probability of the output random sequence Y given as input the random sequence X can be defined through a sequence of hidden states

[1] Sometimes in the literature the term *weighted transducer* is used to denote a more general machine where transitions between states may occur without consuming an input symbol or emitting an output; this is not the case here.

H by

$$f(x, y) = \mathbb{P}[Y = y \,|\, X = x] = \sum_{h \in [n]^t} \mathbb{P}[Y = y, H = h \,|\, X = x] \ ,$$

where we assume that for a fixed sequence of states h, the probability of output y given input x factorizes as follows:

$$\begin{aligned}\mathbb{P}[Y = y, H = h \,|\, X = x] = \mathbb{P}[H_1 = h_1] \\ \times \prod_{s=1}^{t-1} \mathbb{P}[Y_s = y_s, H_{s+1} = h_{s+1} \,|\, H_s = h_s, X_s = x_s] \\ \times \mathbb{P}[Y_t = y_t \,|\, H_t = h_t, X_t = x_t] \ .\end{aligned}$$

In particular, we can write the operators in terms of this factorization as

$$\begin{aligned}\boldsymbol{\alpha}_0(i) &= \mathbb{P}[H_1 = i] \ , \\ \boldsymbol{\alpha}_\infty &= \mathbf{1} \ , \\ \mathbf{A}^\delta_\sigma(i, j) &= \mathbb{P}[Y_s = \delta, H_{s+1} = j \,|\, H_s = i, X_s = \sigma] \ .\end{aligned}$$

We note that the definition of $\mathbf{A}^\delta_\sigma$ is independent of the timestep s. Observe that writing $\mathbf{v} = \mathbf{A}^\delta_\sigma \mathbf{1}$, it is easy to check that $\mathbf{v}(i) = \mathbb{P}[Y_s = \delta \,|\, H_s = i, X_s = \sigma]$.

In general this model may be very complicated to learn, specially when no assumptions on the distribution of input examples X are made. Thus, to simplify our analysis we will need to make further assumptions on our model. In particular, we will assume that given the current state, the current output is independent of the current input, and that given the current state and input symbol the next state is independent of the current output. In terms of the model, this means that the following factorization holds:

$$\mathbb{P}[Y_s, H_{s+1} \,|\, H_s, X_s] = \mathbb{P}[Y_s \,|\, H_s] \cdot \mathbb{P}[H_{s+1} \,|\, H_s, X_s] \ .$$

Thus, the computation performed by the model reduces $\mathbb{P}[Y = y \,|\, X = x]$ to summing the following over all $h \in [n]^t$:

$$\begin{aligned}&\mathbb{P}[H_1 = h_1] \cdot \mathbb{P}[Y_t = y_t \,|\, H_t = h_t] \\ &\times \prod_{s=1}^{t-1} \mathbb{P}[Y_s = y_s \,|\, H_s = h_s] \cdot \mathbb{P}[H_{s+1} = h_{s+1} \,|\, H_s = h_s, X_s = x_s] \ .\end{aligned}$$

Observe that this implies the output sequence depends only on the first $t - 1$ symbols of x.

In addition to simplifying the learning task, these independence assumptions have a neat interpretation in terms of the model parametrization as a weighted transducer. To see this, note first that the entries in the transition operators now satisfy

$$\mathbf{A}^\delta_\sigma(i, j) = \mathbb{P}[Y_s = \delta \,|\, H_s = i] \cdot \mathbb{P}[H_{s+1} = j \,|\, H_s = i, X_s = \sigma] \ .$$

When written in matrix form, this factorization yields $\mathbf{A}^\delta_\sigma = \mathbf{O}_\delta \mathbf{T}_\sigma$ where $\mathbf{T}_\sigma \in \mathbb{R}^{n \times n}$ and $\mathbf{O}_\delta \in \mathbb{R}^{n \times n}$ is a diagonal matrix, with entries defined by

$$\begin{aligned}\mathbf{T}_\sigma(i, j) &= \mathbb{P}[H_{s+1} = j \,|\, H_s = i, X_s = \sigma] \ , \\ \mathbf{O}_\delta(i, i) &= \mathbb{P}[Y_s = \delta \,|\, H_s = i] \ .\end{aligned}$$

This means that each operator in the model is a product of two matrices, and that operators that share either the input or the output symbols share a piece of this decomposition as well. In this case, the vector $\mathbf{A}^\delta_\sigma \mathbf{1}$ takes an interesting form. Let $\mathbf{o}_\delta \in \mathbb{R}^n$ be a vector given by $\mathbf{o}_\delta(i) = \mathbf{O}_\delta(i, i)$. Also, note that $\sum_{j=1}^n \mathbb{P}[H_{s+1} = j \,|\, H_s = i, X_s = \sigma] = 1$ implies $\mathbf{T}_\sigma \mathbf{1} = \mathbf{1}$. Therefore, we have $\mathbf{A}^\delta_\sigma \mathbf{1} = \mathbf{O}_\delta \mathbf{T}_\sigma \mathbf{1} = \mathbf{O}_\delta \mathbf{1} = \mathbf{o}_\delta$. This is just another way of seeing that under our independence assumptions the value of

$f(x, y)$ depends only on the first $t-1$ symbols of x. This fact will play a crucial role in the definition of the Hankel matrices that we give in Section 7.3. For convenience we shall define an observation matrix $\mathbf{O} \in \mathbb{R}^{n \times \Delta}$ containing all the columns $\mathbf{o}_\delta$; that is, $\mathbf{O}(i, \delta) = \mathbb{P}[Y_s = \delta \,|\, H_s = i]$.

The model of probabilistic transducer with independence between transition and emission will henceforth be called *Factorized Probabilistic Weighted Transducer* (FPWT). For these models, the following fact will prove useful.

Lemma 7.1.1. *Given $k \geq 0$, for any $x \in \Sigma^k$ one has $\sum_{y \in \Delta^k} \mathbf{A}_x^y \mathbf{1} = \mathbf{1}$.*

Proof. First note that for any $1 \leq i \leq n$ one has

$$\sum_{\delta \in \Delta} \mathbf{o}_\delta(i) = \sum_{\delta \in \Delta} \mathbb{P}[Y_s = \delta \,|\, H_s = i] = 1 \ .$$

Thus, $\sum_{\delta \in \Delta} \mathbf{A}_\sigma^\delta \mathbf{1} = \sum_{\delta \in \Delta} \mathbf{o}_\delta = \mathbf{1}$. Hence, the claim follows from a simple induction argument on k. □

7.2 A Learning Model for Conditional Distributions

If we are to learn a transducer we will need a sample of input-output example pairs. Modeling how those will be sampled and how the error incurred by the learned transducer will be assessed is the subject of this section. Note that here the standard PAC learning model for probability distributions cannot be used because the goal is to learn a conditional distribution when input instances are being generated by an unknown distribution which we do *not* want to learn; we are only interested in the conditional behavior. The standard PAC model for learning deterministic functions does not fit in here either because given an input we are interested in modelling a distribution over outputs instead of choosing a single fixed output. That said, we will be using a learning model resembling the one used in [ATW01] for learning stochastic rules, but with a different loss function.

Let D be a distribution over $(\Sigma \times \Delta)^\star$ and suppose that (x, y) is a pair of strings drawn from this distribution. We will write $D = D_X \otimes D_{Y|X}$, where D_X represents the marginal distribution over $\Sigma^\star$ and $D_{Y|X}$ the conditional distribution of y given some $x \in \Sigma^\star$. With the notation from previous section we can write $D_X(x) = \mathbb{P}[X = x]$, and $D_{Y|X}(x, y) = \mathbb{P}[Y = y \,|\, X = x]$. Thinking the conditional distribution as a kernel, for any $x \in \Sigma^\star$ we will denote the function $y \mapsto D_{Y|X}(x, y)$ by $D_{Y|x}$. It will also be useful to define the marginal distribution over $\Delta^\star$ which we denote by D_Y, and which satisfies $D_Y(y) = \mathbb{P}[Y = y] = \mathbb{E}_{x \sim D_X}[D_{Y|x}(y)]$. With this notation we are ready to introduce the error measure used to determine the accuracy of a learned transducer.

Suppose we are given m input-output example pairs (x^i, y^i) generated according to D which are used for learning a transducer $\hat{D}_{Y|X}$. We define the error between $D_{Y|X}$ and $\hat{D}_{Y|X}$ to be the L_1 distance between D and $D_X \otimes \hat{D}_{Y|X}$:

$$\begin{aligned}
\mathrm{L}_1(D_X \otimes D_{Y|X}, D_X \otimes \hat{D}_{Y|X}) &= \sum_{x \in \Sigma^\star} \sum_{y \in \Delta^\star} |D_{Y|x}(y) - \hat{D}_{Y|x}(y)| D_X(x) \\
&= \mathbb{E}_{x \sim D_X} \left[\sum_{y \in \Delta^\star} |D_{Y|x}(y) - \hat{D}_{Y|x}(y)| \right] \\
&= \mathbb{E}_{x \sim D_X} \left[\mathrm{L}_1(D_{Y|x}, \hat{D}_{Y|x}) \right] \ .
\end{aligned}$$

This deserves some comments. The first is to observe that this error measure has the dependence on D_X that one expects: it ponders errors on $\hat{D}_{Y|x}$ according to the likelihood of observing x as input. The second interesting feature is that when the conditional distributions are computed by weighted transducers, then the L_1 distance in the last expression can be bounded using the tools introduced in Section 5.4. Like we did for the bounds in Section 6.3, we will need to consider the error incurred by a hypothesis $\hat{D}_{Y|X}$ only on input strings whose length is bounded by some t. Since input and output

strings are constrained to have the same length by the model, this means that we will be bounding the following:

$$\sum_{x\in\Sigma^{\leq t}}\sum_{y\in\Delta^{|x|}}|D_{Y|x}(y)-\hat{D}_{Y|x}(y)|D_X(x)\ .$$

The most general PAC learning results are in a distribution-independent setting, in the sense that they provide algorithms that can learn under any input distribution. In our case that would amount to being able to learn weighted transducers under any possible distribution D_X, which turns out to be an extremely hard problem. In contrast, if for example D_X could be realized by a stochastic WFA, then D would itself be realized by a WFA. Then, by a simple reduction we could learn $D_{Y|X}$ by using results from Section 6.3 to learn D and D_X. Our goal in this chapter will be to move as closer as possible to the distribution-independent setting, and thus we will not be willing to assume that D_X has a particular form. We will, however, need to make some non-parametric assumptions on D_X as will be seen in the following sections.

7.3 A Spectral Learning Algorithm

In this section we will derive a spectral algorithm for learning conditional distributions which can be realized by weighted transducers satisfying certain assumptions. As in previous spectral algorithms, the key will be to define adequate Hankel matrices from whose factorizations one can recover operators for a weighted transducer realizing (an approximation to) the same conditional distribution as the original transducer. The fact that the observations are sampled according to an input distribution over which we have no control will make the derivation slightly more involved than in previous methods. This complication will be the main reason why we must make assumptions about the weighted transducers we will be learning.

Let $A=\langle\boldsymbol{\alpha}_0,\mathbf{1},\{\mathbf{A}^\delta_\sigma\}_{(\sigma,\delta)\in\Sigma\times\Delta}\rangle$ be a FPWT with n states. This means that $f_A:(\Sigma\times\Delta)^\star\to\mathbb{R}$ computes a conditional distribution: $f_A(x,y)=D_{Y|x}(y)=\mathbb{P}[Y=y\,|\,X=x]$ over pairs of aligned sequences. It also means that operators of A can be obtained from a set of probabilities given by an observation matrix $\mathbf{O}\in\mathbb{R}^{n\times\Delta}$ and a set of transition matrices $\mathbf{T}_\sigma\in\mathbb{R}^{n\times n}$. The goal is then to learn a weighted transducer that approximates f_A under the metric defined in the previous section for a given input distribution D_X over $\Sigma^\star$.

We begin by defining a set of quantities derived from D_X and A and stating some assumptions which we need in order to derive our algorithm. For any symbols $\sigma,\sigma'\in\Sigma$ we define

$$\mathbb{P}[X_1=\sigma'\,|\,X_2=\sigma]=\frac{\mathbb{P}[X\in\sigma'\sigma\Sigma^+]}{\mathbb{P}[X\in\Sigma\sigma\Sigma^+]}\ .$$

Now, for any input symbol $\sigma\in\Sigma$ we define an averaged transition matrix

$$\bar{\mathbf{T}}_\sigma=\sum_{\sigma'\in\Sigma}\mathbf{T}_{\sigma'}\mathbb{P}[X_1=\sigma'\,|\,X_2=\sigma]\ .$$

Averaging over σ we also define $\bar{\mathbf{T}}=\sum_{\sigma\in\Sigma}\bar{\mathbf{T}}_\sigma\mathbb{P}[X\in\Sigma\sigma\Sigma^+]$. We will use $\mathbf{D}_{\boldsymbol{\alpha}_0}$ to denote a diagonal $n\times n$ matrix that contains $\boldsymbol{\alpha}_0$ on its diagonal and zero in every off-diagonal entry. Finally, we will make the following assumptions.

Assumption 3. *Transducer A and distribution D_X satisfy the following:*

1. $n\leq|\Delta|$,
2. $\mathrm{rank}(\mathbf{O})=\mathrm{rank}(\mathbf{O}^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}_\sigma)=\mathrm{rank}(\mathbf{O}^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}})=n$,
3. $\mathbb{P}[|X|\leq 2]=0$,
4. $\mathbb{P}[X\in\Sigma\sigma\Sigma^+]>0$ *for all* $\sigma\in\Sigma$.

A first observation about these assumptions is that without (1) the rank assumptions in (2) would not be possible due to dimensional restrictions. Assumptions (3) and (4) are what we will require on our input distribution, which turns out not to be very restrictive. It is also worth noting that the last two rank assumptions in (2) also depend on the input distribution through the definition of $\bar{\mathbf{T}}_\sigma$ and $\bar{\mathbf{T}}$. One can interpret these two assumptions as saying that the behavior of the input distribution is such that it will not act "maliciously" to hide some of the transitions in the target transducer. Also note that (3) and $\mathbf{T}_{\sigma'}\mathbf{1} = \mathbf{1}$ imply that $\bar{\mathbf{T}}_\sigma\mathbf{1} = \mathbf{1}$ for any σ, and $\bar{\mathbf{T}}\mathbf{1} = \mathbf{1}$. We note here that these assumptions resemble those made in [HKZ09] to prove the learnability of HMM with a spectral method.

Several Hankel matrices can be defined in this setting. We now introduce the ones that we will be using in our algorithm. It is important to note that unlike in previous cases, here we will work with a fixed basis $\mathcal{P} = \mathcal{S} = \Delta$. This will work because of the assumptions we just introduced, and is necessary because we are interested on marginalizing as much as possible the effect of D_X. We begin by defining a matrix $\mathbf{H}_\sigma \in \mathbb{R}^{\Delta\times\Delta}$ for each $\sigma \in \Sigma$ with entries given by

$$\mathbf{H}_\sigma(\delta_i, \delta_j) = \mathbb{P}[Y \in \delta_i\delta_j\Delta^+ \,|\, X \in \Sigma\sigma\Sigma^+] \ .$$

Averaging these matrices we get $\mathbf{H} = \sum_{\sigma\in\Sigma} \mathbf{H}_\sigma \mathbb{P}[X \in \Sigma\sigma\Sigma^+]$, whose entries correspond to

$$\mathbf{H}(\delta_i, \delta_j) = \mathbb{P}[Y \in \delta_i\delta_j\Delta^+] \ .$$

For each pair of symbols $(\sigma, \delta) \in \Sigma \times \Delta$ we define a matrix $\mathbf{H}^\delta_\sigma \in \mathbb{R}^{\Delta\times\Delta}$ with entries

$$\mathbf{H}^\delta_\sigma(\delta_i, \delta_j) = \mathbb{P}[Y \in \delta_i\delta\delta_j\Delta^\star \,|\, X \in \Sigma\sigma\Sigma^+] \ .$$

We also define a vector $\mathbf{h} \in \mathbb{R}^\Delta$ with entries $\mathbf{h}(\delta) = \mathbb{P}[Y \in \delta\Delta^\star]$.

It is clear from these definitions that given a sample of pairs of strings one can easily compute empirical estimates of all these matrices and vectors. The learning algorithm is based on applying the following result to these estimates.

Theorem 7.3.1. *Suppose Assumption 3 holds and let $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$ be a compact SVD with $\mathbf{V} \in \mathbb{R}^{\Delta\times n}$. Define the weighted transducer $A' = \langle \boldsymbol{\alpha}'_0, \boldsymbol{\alpha}'_\infty, \{\mathbf{A}'^\delta_\sigma\}\rangle$ given by ${\boldsymbol{\alpha}'_0}^\top = \mathbf{h}^\top\mathbf{V}$, $\boldsymbol{\alpha}'_\infty = (\mathbf{H}\mathbf{V})^+\mathbf{h}$, and $\mathbf{A}'^\delta_\sigma = (\mathbf{H}_\sigma\mathbf{V})^+\mathbf{H}^\delta_\sigma\mathbf{V}$. Then we have $f_A = f_{A'}$.*

Like for other spectral methods we have presented, the key to proving this theorem lies on the existence of certain factorizations for the Hankel matrices we just defined. We now establish the existence of these decompositions.

Lemma 7.3.2. *The following decompositions hold:*

1. $\mathbf{H}_\sigma = \mathbf{O}^\top\mathbf{D}_\alpha\bar{\mathbf{T}}_\sigma\mathbf{O}$,
2. $\mathbf{H} = \mathbf{O}^\top\mathbf{D}_\alpha\bar{\mathbf{T}}\mathbf{O}$,
3. $\mathbf{H}^\delta_\sigma = \mathbf{O}^\top\mathbf{D}_\alpha\bar{\mathbf{T}}_\sigma\mathbf{A}^\delta_\sigma\mathbf{O}$,
4. $\mathbf{h} = (\boldsymbol{\alpha}_0^\top\mathbf{O})^\top = \mathbf{O}^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}\mathbf{1}$.

Proof. Applying the definitions and the fact that the conditional distribution only has positive probability for strings of the same length we get

$$\begin{aligned}
\mathbf{H}_\sigma(\delta_i, \delta_j)\mathbb{P}[X \in \Sigma\sigma\Sigma^+] &= \sum_{\sigma'\in\Sigma}\sum_{x\in\Sigma^+}\sum_{y\in\Delta^+} \mathbb{P}[Y = \delta_i\delta_j y, X = \sigma'\sigma x] \\
&= \sum_{\sigma'\in\Sigma}\sum_{x\in\Sigma^+}\sum_{y\in\Delta^+} \mathbb{P}[Y = \delta_i\delta_j y \,|\, X = \sigma'\sigma x]\mathbb{P}[X = \sigma'\sigma x] \\
&= \sum_{\sigma'\in\Sigma}\sum_{x\in\Sigma^+}\sum_{y\in\Delta^{|x|}} \boldsymbol{\alpha}_0^\top\mathbf{A}^{\delta_i}_{\sigma'}\mathbf{A}^{\delta_j}_\sigma\mathbf{A}^y_x\mathbf{1}\mathbb{P}[X = \sigma'\sigma x] \ .
\end{aligned}$$

We now sum first over y and apply Lemma 7.1.1. Then, using that $\mathbf{A}_\sigma^{\delta_j}\mathbf{1} = \mathbf{O}_{\delta_j}\mathbf{1}$ and $\sum_{\sigma'\in\Sigma}\mathbf{A}_{\sigma'}^{\delta_i}\mathbb{P}[X_1 = \sigma' \,|\, X_2 = \sigma] = \mathbf{O}_{\delta_i}\bar{\mathbf{T}}_\sigma$, we obtain

$$\begin{aligned}\mathbf{H}_\sigma(\delta_i,\delta_j) &= \sum_{\sigma'\in\Sigma}\boldsymbol{\alpha}_0^\top\mathbf{A}_{\sigma'}^{\delta_i}\mathbf{A}_\sigma^{\delta_j}\mathbf{1}\;\mathbb{P}[X_1=\sigma' \,|\, X_2=\sigma]\\ &= \boldsymbol{\alpha}_0^\top\mathbf{O}_{\delta_i}\bar{\mathbf{T}}_\sigma\mathbf{O}_{\delta_j}\mathbf{1}\\ &= \mathbf{o}_{\delta_i}^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}_\sigma\mathbf{o}_{\delta_j}\;\;,\end{aligned}$$

where the last equation follows from simple matrix algebra. Thus, since the columns of $\mathbf{O}$ are the vectors $\mathbf{o}_\delta$, we get $\mathbf{H}_\sigma = \mathbf{O}^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}_\sigma\mathbf{O}$. The decomposition of $\mathbf{H}$ follows now immediately from the definition of $\bar{\mathbf{T}}$. Now we proceed to the third decomposition, for which we have

$$\begin{aligned}&\mathbf{H}_\sigma^\delta(\delta_i,\delta_j)\,\mathbb{P}[X\in\Sigma\sigma\Sigma^+]\\ &\quad= \sum_{\sigma',\sigma''\in\Sigma}\sum_{x\in\Sigma^\star}\sum_{y\in\Delta^\star}\mathbb{P}[Y=\delta_i\delta\delta_j y, X=\sigma'\sigma\sigma'' x]\\ &\quad= \sum_{\sigma',\sigma''\in\Sigma}\sum_{x\in\Sigma^\star}\sum_{y\in\Delta^\star}\mathbb{P}[Y=\delta_i\delta\delta_j y \,|\, X=\sigma'\sigma\sigma'' x]\,\mathbb{P}[X=\sigma'\sigma\sigma'' x]\\ &\quad= \sum_{\sigma',\sigma''\in\Sigma}\sum_{x\in\Sigma^\star}\sum_{y\in\Delta^{|x|}}\boldsymbol{\alpha}_0^\top\mathbf{A}_{\sigma'}^{\delta_i}\mathbf{A}_\sigma^\delta\mathbf{A}_{\sigma''}^{\delta_j}\mathbf{A}_x^y\mathbf{1}\,\mathbb{P}[X=\sigma'\sigma\sigma'' x]\;\;.\end{aligned}$$

From here we use the same arguments as in the previous case to get

$$\begin{aligned}\mathbf{H}_\sigma^\delta(\delta_i,\delta_j) &= \sum_{\sigma',\sigma''\in\Sigma}\boldsymbol{\alpha}_0^\top\mathbf{A}_{\sigma'}^{\delta_i}\mathbf{A}_\sigma^\delta\mathbf{A}_{\sigma''}^{\delta_j}\mathbf{1}\frac{\mathbb{P}[X\in\sigma'\sigma\sigma''\Sigma^\star]}{\mathbb{P}[X\in\Sigma\sigma\Sigma^+]}\\ &= \sum_{\sigma'\in\Sigma}\boldsymbol{\alpha}_0^\top\mathbf{A}_{\sigma'}^{\delta_i}\mathbf{A}_\sigma^\delta\mathbf{O}_{\delta_j}\mathbf{1}\mathbb{P}[X_1=\sigma' \,|\, X_2=\sigma]\\ &= \boldsymbol{\alpha}_0^\top\mathbf{O}_{\delta_i}\bar{\mathbf{T}}_\sigma\mathbf{A}_\sigma^\delta\mathbf{O}_{\delta_j}\mathbf{1}\\ &= \mathbf{o}_{\delta_i}^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}_\sigma\mathbf{A}_\sigma^\delta\mathbf{o}_{\delta_j}\;\;.\end{aligned}$$

Thus, we get $\mathbf{H}_\sigma^\delta = \mathbf{O}^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}_\sigma\mathbf{A}_\sigma^\delta\mathbf{O}$. Similar arguments now yield the following derivation:

$$\begin{aligned}\mathbf{h}(\delta) &= \sum_{y\in\Delta^\star}\mathbb{P}[Y=\delta y]\\ &= \sum_{\sigma\in\Sigma}\sum_{x\in\Sigma^\star}\sum_{y\in\Delta^\star}\mathbb{P}[Y=\delta y, X=\sigma x]\\ &= \sum_{\sigma\in\Sigma}\sum_{x\in\Sigma^\star}\sum_{y\in\Delta^{|x|}}\mathbb{P}[Y=\delta y \,|\, X=\sigma x]\mathbb{P}[X=\sigma x]\\ &= \sum_{\sigma\in\Sigma}\sum_{x\in\Sigma^\star}\sum_{y\in\Delta^{|x|}}\boldsymbol{\alpha}_0^\top\mathbf{A}_\sigma^\delta\mathbf{A}_x^y\mathbf{1}\mathbb{P}[X=\sigma x]\\ &= \sum_{\sigma\in\Sigma}\boldsymbol{\alpha}_0^\top\mathbf{A}_\sigma^\delta\mathbf{1}\mathbb{P}[X=\sigma\Sigma^\star]\\ &= \boldsymbol{\alpha}_0^\top\mathbf{O}_\delta\mathbf{1}\mathbb{P}[|X|\geq 1]\\ &= \boldsymbol{\alpha}_0^\top\mathbf{o}_\delta\\ &= \mathbf{o}_\delta^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\mathbf{1}\\ &= \mathbf{o}_\delta^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}\mathbf{1}\;\;,\end{aligned}$$

where last inequality follows from $\bar{\mathbf{T}}\mathbf{1} = \mathbf{1}$. Thus, the desired decompositions for $\mathbf{h}$ follow from $\mathbf{h}(\delta) = \boldsymbol{\alpha}_0^\top\mathbf{o}_\delta$ and $\mathbf{h}(\delta) = \mathbf{o}_\delta^\top\mathbf{D}_{\boldsymbol{\alpha}_0}\bar{\mathbf{T}}\mathbf{1}$. □

Algorithm 7: Algorithm for Learning FPWT

Input: n, Σ, Δ, $S = (z^1, \ldots, z^m)$
Output: A WFT $\hat{A} = \langle \hat{\boldsymbol{\alpha}}_0, \hat{\boldsymbol{\alpha}}_\infty, \{\hat{\mathbf{A}}_\sigma^\delta\}\rangle$

Using S, compute estimations $\hat{\mathbf{h}}$, $\hat{\mathbf{H}}$, $\hat{\mathbf{H}}_\sigma$ for all $\sigma \in \Sigma'$, and $\hat{\mathbf{H}}_\sigma^\delta$ for all $(\sigma, \delta) \in \Sigma \times \Delta$;
Compute the n top right singular vectors $\hat{\mathbf{V}}$ of $\mathbf{H}$;
Let $\hat{\boldsymbol{\alpha}}_0^\top \leftarrow \hat{\mathbf{h}}^\top \hat{\mathbf{V}}$ and $\hat{\boldsymbol{\alpha}}_\infty \leftarrow (\hat{\mathbf{H}}\hat{\mathbf{V}})^+ \hat{\mathbf{h}}$;
foreach $(\sigma, \delta) \in \Sigma \times \Delta$ **do** let $\hat{\mathbf{A}}_\sigma^\delta \leftarrow (\hat{\mathbf{H}}_\sigma \hat{\mathbf{V}})^+ \hat{\mathbf{H}}_\sigma^\delta \hat{\mathbf{V}}$;

Based on these factorizations and Assumptions 3, we are ready to prove the main theorem of this section.

Proof of Theorem 7.3.1. We will show that $A' = \mathbf{M}^{-1} A \mathbf{M}$ for some invertible $\mathbf{M} \in \mathbb{R}^{n \times n}$ and the result will follow from the invariance under change of basis of the function computed by a weighted automaton.

Let $\mathbf{P} = \mathbf{O}^\top \mathbf{D}_{\boldsymbol{\alpha}_0} \bar{\mathbf{T}}$ and $\mathbf{P}_\sigma = \mathbf{O}^\top \mathbf{D}_{\boldsymbol{\alpha}_0} \bar{\mathbf{T}}_\sigma$. Note that by Assumption 3 matrices $\mathbf{P}, \mathbf{P}_\sigma \in \mathbb{R}^{\Delta \times n}$ have rank n. Since $\mathbf{O}$ also has rank n, this means that the factorizations $\mathbf{H} = \mathbf{P}\mathbf{O}$ and $\mathbf{H}_\sigma = \mathbf{P}_\sigma \mathbf{O}$ are in fact rank factorizations. Thus, since $\mathbf{H}$ has rank n, using the compact SVD we get the rank factorization $\mathbf{H} = (\mathbf{H}\mathbf{V})\mathbf{V}^\top$. Therefore, the matrix $\mathbf{M} = \mathbf{O}\mathbf{V} \in \mathbb{R}^{n \times n}$ is invertible by Corollary 5.2.2. Now we check that $\mathbf{M}$ satisfies the equation. In the first place we have, by Lemma 7.3.2 and properties of Moore–Penrose pseudo-inverse:

$$\begin{aligned} \mathbf{A'}_\sigma^\delta &= (\mathbf{H}_\sigma \mathbf{V})^+ \mathbf{H}_\sigma^\delta \mathbf{V} \\ &= (\mathbf{P}_\sigma \mathbf{M})^+ \mathbf{P}_\sigma \mathbf{A}_\sigma^\delta \mathbf{M} \\ &= \mathbf{M}^{-1} \mathbf{P}_\sigma^+ \mathbf{P}_\sigma \mathbf{A}_\sigma^\delta \mathbf{M} \\ &= \mathbf{M}^{-1} \mathbf{A}_\sigma^\delta \mathbf{M} \ . \end{aligned}$$

Using the first decomposition for $\mathbf{h}$ we get

$$\boldsymbol{\alpha}_0'^\top = \mathbf{h}^\top \mathbf{V} = \boldsymbol{\alpha}_0^\top \mathbf{M} \ .$$

Finally, by the second decomposition we get

$$\boldsymbol{\alpha}'_\infty = (\mathbf{H}\mathbf{V})^+ \mathbf{h} = (\mathbf{P}\mathbf{M})^+ \mathbf{P}\mathbf{1} = \mathbf{M}^{-1}\mathbf{1} \ . \qquad \square$$

In view of this result, we can now give a spectral learning algorithm for FPWT which follows the same principles used in Algorithm 5. This new algorithm receives as input the number of states n of the target FPWT, alphabets Σ and Δ, and a sample $S = (z^1, \ldots, z^m)$ of pairs $z^i = (x^i, y^i)$ drawn from $D_X \otimes D_{Y|X}$, where D_X is an arbitrary distribution over $\Sigma^\star$ and $D_{Y|X}$ is a conditional distribution over aligned sequences realized by a FPWT. The details can be found in Algorithm 7.

7.4 Sample Complexity Bounds

In this section we give sample complexity bounds for Algorithm 7. We will assume throughout all the analysis that the target distribution D_X and FPWT $D_{Y|X} = f$ satisfy Assumptions 3. The overall structure of the analysis will be the same as in Section 6.3.

We begin by defining a transducer $\tilde{A}$ obtained from the singular vectors $\hat{\mathbf{V}}$ of $\hat{\mathbf{H}}$ and the true Hankel matrices. We will show that $f_{\tilde{A}} = f_A$ when some conditions are met. Begin by defining $\tilde{A} = \langle \tilde{\boldsymbol{\alpha}}_0, \tilde{\boldsymbol{\alpha}}_\infty, \{\tilde{\mathbf{A}}_\sigma^\delta\}\rangle$ as follows:

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_0^\top &= \mathbf{h}^\top \hat{\mathbf{V}} \ , \\ \tilde{\boldsymbol{\alpha}}_\infty &= (\mathbf{H}\hat{\mathbf{V}})^+ \mathbf{h} \ , \\ \tilde{\mathbf{A}}_\sigma^\delta &= (\mathbf{H}_\sigma \hat{\mathbf{V}})^+ \mathbf{H}_\sigma^\delta \hat{\mathbf{V}} \ . \end{aligned}$$

The following result is analogous to Lemma 6.3.1 and gives conditions under which $f_{\tilde{A}} = f$.

Lemma 7.4.1. *Suppose that* $\|\mathbf{H} - \hat{\mathbf{H}}\|_2 \leq \epsilon/(1+\epsilon)\mathfrak{s}_n(\mathbf{H})$ *for some* $\epsilon < 1$. *Then the following inequalities hold for all* $\sigma \in \Sigma$*:*

$$\begin{aligned}\mathfrak{s}_n(\mathbf{H}\hat{\mathbf{V}}) &\geq \mathfrak{s}_n(\mathbf{H})\sqrt{1-\epsilon^2} \\ \mathfrak{s}_n(\mathbf{H}_\sigma\hat{\mathbf{V}}) &\geq \mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})\sqrt{1-\epsilon^2} \ .\end{aligned}$$

Furthermore, we have $f_{\tilde{A}} = f$.

Proof. The same argument used in the proof of Lemma 6.3.1 shows that the assumption $\|\mathbf{H} - \hat{\mathbf{H}}\|_2 \leq \epsilon/(1+\epsilon)\mathfrak{s}_n(\mathbf{H})$ implies $\mathfrak{s}_n(\mathbf{V}^\top\hat{\mathbf{V}}) \geq \sqrt{1-\epsilon^2}$. Thus, since $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$, by Lemma A.2.6 we get

$$\mathfrak{s}_n(\mathbf{H}\hat{\mathbf{V}}) \geq \mathfrak{s}_n(\mathbf{U}\mathbf{\Lambda})\mathfrak{s}_n(\mathbf{V}^\top\hat{\mathbf{V}}) \geq \mathfrak{s}_n(\mathbf{H})\sqrt{1-\epsilon^2} \ .$$

In addition, since Lemma 7.3.2 gives us the rank factorizations $\mathbf{H} = \mathbf{P}\mathbf{O}$ and $\mathbf{H}_\sigma = \mathbf{P}_\sigma\mathbf{O}$, we immediately have $\mathbf{H}_\sigma\mathbf{V}\mathbf{V}^\top = \mathbf{H}_\sigma$. Thus, the following holds:

$$\begin{aligned}\mathfrak{s}_n(\mathbf{H}_\sigma\hat{\mathbf{V}}) &= \mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V}\mathbf{V}^\top\hat{\mathbf{V}}) \\ &\geq \mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})\mathfrak{s}_n(\mathbf{V}^\top\hat{\mathbf{V}}) \\ &\geq \mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})\sqrt{1-\epsilon^2} \ .\end{aligned}$$

The claim on $f_{\tilde{A}}$ follows from the same argument used in Lemma 6.3.1. □

Note that by Theorem 7.3.1 we have $A = \mathbf{M}A'\mathbf{M}^{-1}$ with $\mathbf{M} = \mathbf{O}\mathbf{V}$. Furthermore, if the conditions of Lemma 7.4.1 are met, then we also have $A = \hat{\mathbf{M}}\tilde{A}\hat{\mathbf{M}}^{-1}$ for some $\hat{\mathbf{M}}$. In this case it is easy to see, using that $(\mathbf{H}\hat{\mathbf{V}})\hat{\mathbf{V}}^\top$ is a rank factorization and Corollary 5.2.2, that one has $\hat{\mathbf{M}} = \mathbf{O}\hat{\mathbf{V}} = \mathbf{O}\mathbf{V}\mathbf{V}^\top\hat{\mathbf{V}}$. Thus, using the same proof as in Lemma 6.3.2 we get the following.

Lemma 7.4.2. *Suppose that* $\|\mathbf{H} - \hat{\mathbf{H}}\|_2 \leq \epsilon/(1+\epsilon)\mathfrak{s}_n(\mathbf{H})$ *for some* $\epsilon < 1$. *Then for any* $1 \leq i \leq n$ *we have* $\mathfrak{s}_i(\mathbf{M})\sqrt{1-\epsilon^2} \leq \mathfrak{s}_i(\hat{\mathbf{M}}) \leq \mathfrak{s}_i(\mathbf{M})$.

We can use the change of basis $\hat{\mathbf{M}}$ to define a new weighted transducer $\hat{A}' = \hat{\mathbf{M}}\hat{A}\hat{\mathbf{M}}^{-1} = \langle \hat{\boldsymbol{\alpha}}'_0, \hat{\boldsymbol{\alpha}}'_\infty, \{\hat{\mathbf{A}}'^\delta_\sigma\}\rangle$ using the hypothesis $\hat{\mathbf{A}}$ returned by Algorithm 7. It is obvious by construction that $f_{\hat{A}'} = \hat{f}$. Using this representation for $\hat{f}$ we can prove a variant of Lemma 5.4.4 which will be used to bound the error of our algorithm. Let us define the following quantities:

$$\begin{aligned}\rho_0 &= \|\boldsymbol{\alpha}_0 - \hat{\boldsymbol{\alpha}}'_0\|_1 \ , \\ \rho_\infty &= \|\boldsymbol{\alpha}_\infty - \hat{\boldsymbol{\alpha}}'_\infty\|_\infty \ , \\ \rho_\sigma &= \sum_{\delta\in\Delta} \|\mathbf{A}^\delta_\sigma - \hat{\mathbf{A}}'^\delta_\sigma\|_\infty \ .\end{aligned}$$

Lemma 7.4.3. *The following holds for any* $t \geq 0$ *and* $x \in \Sigma^t$*:*

$$\sum_{y\in\Delta^t} |f(x,y) - \hat{f}(x,y)| \leq (1+\rho_0)(1+\rho_\infty)\prod_{s=1}^{t}(1+\rho_{x_s}) - 1 \ .$$

Proof. The proof is very similar to those of Lemmas 5.4.3 and 5.4.4. Thus, we just highlight the main differences. To begin with, note that since $f = f_A$ and A is a FPWT, then we have $\gamma_\infty = \|\boldsymbol{\alpha}_\infty\|_\infty = \|\mathbf{1}\|_\infty = 1$. Furthermore, $\gamma^\Delta_\sigma = \sum_{\delta\in\Delta} |\mathbf{A}^\delta_\sigma|\mathbf{1} = 1$ and $\gamma_x = \sum_{y\in\Delta^t} \|\boldsymbol{\alpha}_0\mathbf{A}^y_x\|_1 = 1$ by Lemma 7.1.1.

Writing $\varphi_x = \sum_{y\in\Delta^t} \|\boldsymbol{\alpha}_0\mathbf{A}^y_x - \hat{\boldsymbol{\alpha}}'_0\hat{\mathbf{A}}'^y_x\|_1$ and using the same bounding scheme from (5.2) one gets

$$\varphi_{x\sigma} \leq \rho_\sigma + \varphi_x + \varphi_x\rho_\sigma = (1+\rho_\sigma)(1+\varphi_x) - 1 \ .$$

Thus, writing $\rho_0 = \varphi_\lambda$ one can show by induction on the length of x that

$$\varphi_x \leq (1+\rho_0)\prod_{x=1}^{t}(1+\rho_{x_s}) - 1 \ .$$

Finally, the argument used in (5.3) yields

$$\sum_{y\in\Delta^t}|f(x,y)-\hat{f}(x,y)| \leq \rho_\infty + (1+\rho_\infty)\varphi_x \leq (1+\rho_0)(1+\rho_\infty)\prod_{s=1}^{t}(1+\rho_{x_s}) - 1 \ . \qquad \square$$

Before stating the next result we shall define several quantities related to the accuracy with which the Hankel matrices used in Algorithm 7 are approximated. In particular, we write $\varepsilon_0 = \|\mathbf{h}-\hat{\mathbf{h}}\|_2$, $\varepsilon_\lambda = \|\mathbf{H}-\hat{\mathbf{H}}\|_2$, $\varepsilon_\sigma = \|\mathbf{H}_\sigma - \hat{\mathbf{H}}_\sigma\|_2$ and $\varepsilon_\sigma^\delta = \|\mathbf{H}_\sigma^\delta - \hat{\mathbf{H}}_\sigma^\delta\|_2$.

Lemma 7.4.4. *For any $\varepsilon > 0$ and $t \geq 0$, if the following hold for every $\sigma \in \Sigma$:*

$$\begin{aligned}
\varepsilon_0 &\leq C_0 \frac{\varepsilon \mathfrak{s}_n(\mathbf{M})\mathfrak{s}_n(\mathbf{H})^2}{(t+1)^2\sqrt{n}\mathfrak{s}_1(\mathbf{M})} \ , \\
\varepsilon_\lambda &\leq C_\lambda \frac{\varepsilon \mathfrak{s}_n(\mathbf{H})^2}{(t+1)^2\mathfrak{s}_1(\mathbf{M})} \ , \\
\varepsilon_\sigma &\leq C_\Sigma \frac{\varepsilon \mathfrak{s}_n(\mathbf{M})\mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})^2}{(t+1)^2\sqrt{n|\Delta|}\mathfrak{s}_1(\mathbf{M})} \ , \\
\sum_{\delta\in\Delta}\varepsilon_\sigma^\delta &\leq C_\Sigma^\Delta \frac{\varepsilon \mathfrak{s}_n(\mathbf{M})\mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})^2}{(t+1)^2\sqrt{n}\mathfrak{s}_1(\mathbf{M})} \ ,
\end{aligned}$$

then for any $x \in \Sigma^t$ we have $\sum_{y\in\Delta^t}|f(x,y)-\hat{f}(x,y)| \ \leq \varepsilon/(t+1)$. Here C_0, C_λ, C_Σ, and C_Σ^Δ are universal constants.

Proof. We begin by applying Lemma 5.4.5 to obtain the following bounds:

$$\rho_0 = \|\boldsymbol{\alpha}_0 - \hat{\boldsymbol{\alpha}}_0'\|_1 \leq \sqrt{n}\|\hat{\mathbf{M}}^{-1}\|_2\|\tilde{\boldsymbol{\alpha}}_0 - \hat{\boldsymbol{\alpha}}_0\|_2 \leq \frac{\sqrt{n}}{\mathfrak{s}_n(\hat{\mathbf{M}})}\|\mathbf{h}-\hat{\mathbf{h}}\|_2 \ ,$$

$$\begin{aligned}
\rho_\infty &= \|\boldsymbol{\alpha}_\infty - \hat{\boldsymbol{\alpha}}_\infty'\|_\infty \leq \|\hat{\mathbf{M}}\|_2\|\tilde{\boldsymbol{\alpha}}_\infty - \hat{\boldsymbol{\alpha}}_\infty\|_2 \\
&\leq \mathfrak{s}_1(\hat{\mathbf{M}})\left(\frac{\|\mathbf{h}-\hat{\mathbf{h}}\|_2}{\mathfrak{s}_n(\mathbf{H}\hat{\mathbf{V}})} + \frac{1+\sqrt{5}}{2}\frac{\|\hat{\mathbf{h}}\|_2\|\mathbf{H}-\hat{\mathbf{H}}\|_2}{\min\{\mathfrak{s}_n(\mathbf{H}\hat{\mathbf{V}})^2, \mathfrak{s}_n(\hat{\mathbf{H}}\hat{\mathbf{V}})^2\}}\right) \ ,
\end{aligned}$$

$$\begin{aligned}
\rho_\sigma^\delta &= \|\mathbf{A}_\sigma^\delta - \hat{\mathbf{A}}_\sigma'^\delta\|_\infty \leq \sqrt{n}\|\hat{\mathbf{M}}\|_2\|\hat{\mathbf{M}}^{-1}\|_2\|\tilde{\mathbf{A}}_\sigma^\delta - \hat{\mathbf{A}}_\sigma^\delta\|_2 \\
&\leq \frac{\sqrt{n}\mathfrak{s}_1(\hat{\mathbf{M}})}{\mathfrak{s}_n(\hat{\mathbf{M}})}\left(\frac{\|\mathbf{H}_\sigma^\delta - \hat{\mathbf{H}}_\sigma^\delta\|_2}{\mathfrak{s}_n(\mathbf{H}_\sigma\hat{\mathbf{V}})} + \frac{1+\sqrt{5}}{2}\frac{\|\hat{\mathbf{H}}_\sigma^\delta\|_2\|\mathbf{H}_\sigma - \hat{\mathbf{H}}_\sigma\|_2}{\min\{\mathfrak{s}_n(\mathbf{H}_\sigma\hat{\mathbf{V}})^2, \mathfrak{s}_n(\hat{\mathbf{H}}_\sigma\hat{\mathbf{V}})^2\}}\right) \ .
\end{aligned}$$

Now note that we have $\|\hat{\mathbf{h}}\|_2 \leq 1$ and $\|\hat{\mathbf{H}}_\sigma^\delta\|_2 \leq 1$. Furthermore, using Lemmas 7.4.1 and 7.4.2 with $\epsilon = \sqrt{3}/2$ we can see that $\mathfrak{s}_n(\mathbf{H}\hat{\mathbf{V}}) \geq \mathfrak{s}_n(\mathbf{H})/2$, $\mathfrak{s}_n(\mathbf{H}_\sigma\hat{\mathbf{V}}) \geq \mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})/2$ and $\mathfrak{s}_n(\hat{\mathbf{M}}) \geq \mathfrak{s}_n(\mathbf{M})/2$ hold when $\|\mathbf{H}-\hat{\mathbf{H}}\|_2 \leq \mathfrak{s}_n(\mathbf{H})\sqrt{1-\epsilon^2}$. If in addition $\|\mathbf{H}_\sigma - \hat{\mathbf{H}}_\sigma\| \leq \mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})\sqrt{1-\epsilon^2}$, then combining the above with Lemma A.2.1 we get $\mathfrak{s}_n(\hat{\mathbf{H}}_\sigma\hat{\mathbf{V}}) \geq ((2-\sqrt{3})^2/2)\mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})$. Thus, for some choice of constants we have

$$\begin{aligned}
\rho_0 &\leq C_0' \frac{\sqrt{n}}{\mathfrak{s}_n(\mathbf{M})}\varepsilon_0 \ , \\
\rho_\infty &\leq C_\infty' \frac{\mathfrak{s}_1(\mathbf{M})}{\mathfrak{s}_n(\mathbf{H})^2}(\varepsilon_0 + \varepsilon_\lambda) \ , \\
\rho_\sigma^\delta &\leq C_\Sigma^{\Delta\prime} \frac{\sqrt{n}\mathfrak{s}_1(\mathbf{M})}{\mathfrak{s}_n(\mathbf{M})\mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})^2}(\varepsilon_\sigma^\delta + \varepsilon_\sigma) \ .
\end{aligned}$$

Now we can finally apply Lemma 7.4.3 to see that if for some suitable constants the following hold:

$$\begin{aligned}
\varepsilon_0 &\leq C_0'' \frac{\varepsilon}{(t+1)^2} \min\left\{ \frac{\mathfrak{s}_n(\mathbf{M})}{\sqrt{n}}, \frac{\mathfrak{s}_n(\mathbf{H})^2}{\mathfrak{s}_1(\mathbf{M})} \right\} \ , \\
\varepsilon_\lambda &\leq C_\lambda \frac{\varepsilon}{(t+1)^2} \frac{\mathfrak{s}_n(\mathbf{H})^2}{\mathfrak{s}_1(\mathbf{M})} \ , \\
\max_{\sigma \in \Sigma} \frac{\varepsilon_\sigma}{\mathfrak{s}_n(\mathbf{H}_\sigma \mathbf{V})^2} &\leq C_\Sigma \frac{\varepsilon}{(t+1)^2} \frac{\mathfrak{s}_n(\mathbf{M})}{\sqrt{n}|\Delta|\mathfrak{s}_1(\mathbf{M})} \ , \\
\max_{\sigma \in \Sigma} \sum_{\delta \in \Delta} \frac{\varepsilon_\sigma^\delta}{\mathfrak{s}_n(\mathbf{H}_\sigma \mathbf{V})^2} &\leq C_\Sigma^\Delta \frac{\varepsilon}{(t+1)^2} \frac{\mathfrak{s}_n(\mathbf{M})}{\sqrt{n}\mathfrak{s}_1(\mathbf{M})} \ ,
\end{aligned}$$

then for any $x \in \Sigma^t$ we get

$$\sum_{y \in \Delta^t} |f(x,y) - \hat{f}(x,y)| \leq \left(1 + \frac{\varepsilon}{2(t+1)(t+2)}\right)^{t+2} - 1 \leq \frac{\varepsilon}{t+1} \ . \qquad \square$$

Now is time to bound the estimation error on the approximate Hankel matrices used in Algorithm 7. In this case this turns out to be a bit more cumbersome than in the analysis of Section 6.3 because some of the probabilities are conditioned on the input distribution D_X. This will make it necessary that our bounds depend on $p_\sigma = \mathbb{P}[X \in \Sigma\sigma\Sigma^+]$ for each $\sigma \in \Sigma$.

Lemma 7.4.5. *Suppose Algorithm 7 receives a sample of size m as input. Then, for any $\delta > 0$, the following hold simultaneously for every $\sigma \in \Sigma$ with probability at least $1 - \delta$:*

$$\begin{aligned}
\|\mathbf{h} - \hat{\mathbf{h}}\|_2 &\leq \sqrt{\frac{1}{m}} \left(1 + \sqrt{\ln\left(\frac{5}{\delta}\right)}\right) \ , \\
\|\mathbf{H} - \hat{\mathbf{H}}\|_F &\leq \sqrt{\frac{1}{m}} \left(1 + \sqrt{\ln\left(\frac{5}{\delta}\right)}\right) \ , \\
\|\mathbf{H}_\sigma - \hat{\mathbf{H}}_\sigma\|_F &\leq \sqrt{\frac{1}{mp_\sigma - \sqrt{2mp_\sigma \ln(5|\Sigma|/\delta)}}} \left(1 + \sqrt{2\ln\left(\frac{5|\Sigma|}{\delta}\right)}\right) \ , \\
\sum_{\delta \in \Delta} \|\mathbf{H}_\sigma^\delta - \hat{\mathbf{H}}_\sigma^\delta\|_F &\leq \sqrt{\frac{|\Delta|}{mp_\sigma - \sqrt{2mp_\sigma \ln(5|\Sigma|/\delta)}}} \left(1 + \sqrt{\ln\left(\frac{5|\Sigma|}{\delta}\right)}\right) \ .
\end{aligned}$$

Proof. Each of the first two bounds hold with probability at least $1 - \delta/5$ by Theorem 6.2.1, where $c_{\mathcal{B}} = 1$ by definition. For each input symbol $\sigma \in \Sigma$ we define m_σ as the number of examples $(x^i, y^i) \in S$ such that $x_2^i = \sigma$. Note that since $\mathbb{P}[|X| \leq 2] = 0$ by Assumption 3, we have $m = \sum_{\sigma \in \Sigma} m_\sigma$. Furthermore we have $\mathbb{E}[m_\sigma] = mp_\sigma$. Thus, by the Chernoff bound we have that for each $\sigma \in \Sigma$ we have

$$m_\sigma \geq mp_\sigma - \sqrt{2mp_\sigma \ln\left(\frac{5|\Sigma|}{\delta}\right)}$$

with probability at least $1 - \delta/5|\Sigma|$. Since $c_{\mathcal{B}} = 1$ for each $\mathbf{H}_\sigma$, to obtain the third bound we combine this lower bound for m_σ with the following direct consequence of Theorem 6.2.1:

$$\|\mathbf{H}_\sigma - \hat{\mathbf{H}}_\sigma\|_F \leq \sqrt{\frac{1}{m_\sigma}} \left(1 + \sqrt{\ln\left(\frac{5|\Sigma|}{\delta}\right)}\right) \ ,$$

which holds with probability at least $1 - \delta/5|\Sigma|$ for each $\sigma \in \Sigma$. Now consider the matrix $\mathbf{H}_\sigma^\Delta = [\mathbf{H}_\sigma^{\delta_1} \dots \mathbf{H}_\sigma^{\delta_{|\Delta|}}] \in \mathbb{R}^{\Delta \times \Delta \times \Delta}$ whose entries are $\mathbf{H}_\Sigma(\delta_i, \delta_j, \delta_k) = \mathbb{P}[Y \in \delta_i\delta_j\delta_k\Delta^\star \mid X \in \Sigma\sigma\Sigma^+]$. It is easy

to check that this is a Hankel matrix with $c_{\mathcal{B}} = 1$. Then, by the Cauchy–Schwarz inequality and Theorem 6.2.1 we have

$$\begin{aligned}\sum_{\delta\in\Delta} \|\mathbf{H}_\sigma^\delta - \hat{\mathbf{H}}_\sigma^\delta\|_F &\leq \sqrt{|\Delta|}\sqrt{\sum_{\delta\in\Delta} \|\mathbf{H}_\sigma^\delta - \hat{\mathbf{H}}_\sigma^\delta\|_F^2} \\ &= \sqrt{|\Delta|}\|\mathbf{H}_\sigma^\Delta - \hat{\mathbf{H}}_\sigma^\Delta\|_F \\ &\leq \sqrt{\frac{|\Delta|}{m_\sigma}}\left(1 + \sqrt{\ln\left(\frac{5|\Sigma|}{\delta}\right)}\right) \ ,\end{aligned}$$

with probability at least $1 - \delta/5|\Sigma|$ for each $\sigma \in \Sigma$. The result now follows from a union bound. □

We are finally in position to state the main result of this section that gives a sample bound for Algorithm 7. To simplify the notation of the bound we will define quantities $\pi_X = \min_{\sigma\in\Sigma} p_\sigma$ and $\mathfrak{s}_f = \min_{\sigma\in\Sigma} \mathfrak{s}_n(\mathbf{H}_\sigma\mathbf{V})$, which depend on both f and D_X. The proof of the theorem is a simple calculation in view of Lemmas 7.4.4 and 7.4.5. In particular, we only need to recall that, since the rows of $\mathbf{O}$ all add up to 1, we have

$$\mathfrak{s}_1(\mathbf{M}) = \mathfrak{s}_1(\mathbf{OV}) = \|\mathbf{OV}\|_2 \leq \|\mathbf{O}\|_2 \leq \sqrt{n}\|\mathbf{O}\|_\infty = \sqrt{n} \ .$$

Theorem 7.4.6. *Suppose Algorithm 7 receives as input a sample of size m generated from some distribution $D_X \otimes D_{Y|X}$ satisfying Assumptions 3. There exists a universal constant C such that for any $t \geq 0$, $\varepsilon > 0$, $\delta > 0$, if*

$$m \geq C\frac{(t+1)^4 n^{3/2}|\Delta|^2}{\varepsilon^2\pi_X\mathfrak{s}_f^4\mathfrak{s}_n(\mathbf{H})^4\mathfrak{s}_n(\mathbf{OV})^2}\ln\left(\frac{|\Sigma|}{\delta}\right)$$

then with probability at least $1 - \delta$ the hypothesis $\hat{A}$ returned by the algorithm satisfies

$$\sum_{x\in\Sigma^{\leq t}}\sum_{y\in\Delta^{|x|}} |f(x,y) - \hat{f}(x,y)|\mathbb{P}[X = x] \leq \varepsilon \ .$$

Chapter 8

The Spectral Method from an Optimization Viewpoint

The algebraic formulation of the spectral method is in sharp contrast with the analytic formulation of most machine learning problems using some form of optimization problem. Despite its simplicity, the algebraic formulation has one big drawback: it is not clear how to enforce any type of prior knowledge into the algorithm besides the number of states in the hypothesis. On the other hand, a very intuitive way to enforce prior knowledge in optimization-based learning algorithms is by adding regularization terms to the objective function. This and some other facts motivate our study of an optimization-based learning algorithm for WFA founded upon the same principles of the spectral method. In particular, we give a non-convex optimization algorithm that shares some statistical properties with the spectral method, and then show how to obtain a convex relaxation of this problem. The last section of the chapter discusses some issues that need to be taken into account when implementing our optimization-based algorithms in practice.

8.1 Maximum Likelihood and Moment Matching

We start by describing two well-known statistical principles for model estimation which provide inspirations for the spectral method and its optimization counterpart.

8.1.1 Maximum Likelihood Estimation

In statistics, the maximum likelihood principle is a general method for fitting a statistical model to a given set of examples. The rationale behind the method is that for given a sample and a fixed class of hypotheses, the best model in the class is the one that assigns maximum probability to the observed sample; that is, the model that makes it more likely to observe the given data. Two interesting properties of this method are that it is motivated by a simple intuition, and that it is easy to formulate for any given class of models. In general, if p is the density function of some probability distribution and $S = (x^1, \ldots, x^m)$ is a sample of i.i.d. examples drawn from some distribution p^*, the *likelihood* of S with respect to p is defined as

$$\mathbb{L}(S; p) = \prod_{i=1}^{m} p(x^i) \ .$$

Sometimes it is also convenient to work with the *log-likelihood*, which is the logarithm of $\mathbb{L}$:

$$\mathbb{LL}(S; p) = \log \mathbb{L}(S; p) = \sum_{i=1}^{m} \log p(x^i) \ .$$

With these definitions, if $\mathcal{D}$ is a family of statistical models, then the *maximum likelihood* (ML) estimate of p^* in $\mathcal{D}$ given S is

$$\hat{p}_{\mathrm{ML}} = \operatorname*{argmax}_{p \in \mathcal{D}} \mathbb{L}(S; p) = \operatorname*{argmax}_{p \in \mathcal{D}} \mathbb{LL}(S; p) \ . \tag{ML}$$

A very appealing statistical property of ML estimation is that in most cases, under mild assumptions on the hypothesis class $\mathcal{D}$, one can show that if $p^* \in \mathcal{D}$, then for $m \to \infty$ one has $\hat{p}_{\mathrm{ML}} \to p^*$ under a suitable notion of convergence. In those cases it is said that ML estimation is *consistent*.

The maximum likelihood principle is known to be consistent in the case of learning probabilistic automata with a fixed number of states. It was already shown in [BP66; Pet69] that ML estimation is consistent for HMM with discrete state and observation spaces. Furthermore, generalizations to more general state and observation spaces where given in [Ler92]. These results imply the consistency of PFA learning via ML because the class of distributions generated by HMM is the same as the one generated by PFA [DDE05].

Abe and Warmuth proved in [AW92] that something stronger than consistency holds for the ML principle applied to learning the structure and transition probabilities of probabilistic automata without stopping probabilities. In particular, they showed that given a sample S of strings of length t with size $m \geq \tilde{O}((t^2 n^2 |\Sigma|/\varepsilon^2) \log(1/\delta))$ sampled from some PFA p^* with n states over Σ, then with probability at least $1 - \delta$ the hypothesis $\hat{p}_{\mathrm{ML}}$ obtained via ML estimation with S on the class of PFA with n states over Σ satisfies $\mathrm{KL}(p^* \| \hat{p}_{\mathrm{ML}}) \leq \varepsilon$, where the models are compared over Σ^t. Note that this implies that for any fixed t one has $\mathrm{KL}(p^* \| \hat{p}_{\mathrm{ML}}) \to 0$ with $m \to \infty$, with convergence occurring at a rate of type $\tilde{O}(1/\sqrt{m})$. This is much stronger that simple consistency, whose definition does not require any particular rate of convergence and can, in general, converge much slowly.

Unfortunately, despite the intrinsic statistical attractiveness of ML estimation, solving the optimization problem (ML) is in general a difficult problem from the algorithmic point of view. Thus, obtaining efficient learning algorithms by solving ML estimation is a challenging problem. The main reason for this is that optimization (ML) is rarely a convex problem. For example, in the case of PNFA with n states we see that the problem one needs to solve is:

$$\hat{A}_{\mathrm{ML}} = \operatorname*{argmax}_{\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}} \sum_{i=1}^{m} \log(\boldsymbol{\alpha}_0 \mathbf{A}_{x^i} \boldsymbol{\alpha}_\infty) \ .$$

The objective function of this optimization depends on the parameters of the target PNFA in a highly non-convex way. In fact, it turns out that even approximating the optimal solution of this optimization problem is hard [AW92].

In spite of not being able to solve optimization (ML) exactly in a reasonable time, the ML principle is commonly used by practitioners. Most of the times, this involves the use of heuristic methods to find a good *local optimum* of (ML). Among such heuristics, the most well-know is perhaps the Expectation-Maximization (EM) method, which in the case of HMM is sometimes also referred as the Baum–Welch algorithm [BPSW70]. This is an iterative heuristic algorithm for approximating ML parameter estimation on probabilistic models with latent variables. There are at least three good reasons why an iterative method like EM with no convergence guarantees is so widely used in practice.

1. Given enough running time, this type of heuristics do find good models in most applications. This suggests that either the applications are not big enough to reach the regime where the asymptotic complexity becomes impractical, or that the worst case lower bounds do not apply to the type of models one encounters in practice.

2. Heuristics based on ML estimation are usually as flexible as the ML method itself, making them easier to adapt to newly defined models. This allows practitioners to try different models easily in applications, and quickly design learning algorithm for newly defined models.

3. Incorporating prior knowledge on optimization problems via regularization terms is a standard practice in machine learning. Since ML estimation is defined in terms of an optimization problem it is susceptible to such practices. In particular, it is usual to combine ML estimation with model selection criteria like BIC or AIC [Sch78; Aka77] to express the prior knowledge conveyed by Occam's razor; namely, that at similar levels of fitness simpler models are to be preferred.

We have already seen that the spectral method for learning stochastic WFA comes with polynomial guarantees on its running time, is consistent, and it is guaranteed to find solutions close to the optimum when given a finite sample. Thus, in terms of (1) above, the spectral method is superior to ML estimation, at least from a theoretical point of view. However, though the spectral method has been adapted to a large number of probabilistic models, there seems to be no general recipe so far for applying the spectral method to generic probabilistic models. In this respect, according to (2), the ML principle is still more appealing to practitioners. Point (3) above is the main concern of the present chapter. As we will see below, the spectral method for probabilistic models is based on the method of moments. This basically means that it is based on solving a set of equations relating moments and parameters in the target distribution. Since it is not clear how to enforce some prior knowledge in the solution of these equations, this makes ML estimation more appealing from the point of view of the flexibility provided by a wide choice of regularization methods. In the following sections we will describe optimization problems that behave like the spectral method and allow for several variations in regularization strategies.

8.1.2 The Method of Moments

Roughly speaking, the methods of moments is a statistical inference technique that estimates the parameters of a probability distribution by relating them to a set of moments defined in terms of the outcomes of the distribution. A paradigmatic example is the normal distribution, where its parameters, the mean μ and variance σ^2, are in fact moments of the distribution. In this case the method of moments yields the estimate $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$, where $\hat{\mu}$ and $\hat{\sigma}^2$ are the empirical estimations of the mean and variance from a given sample. Its conception dates back to Pearson [Pea94] and is, thus, older than the ML principle. Furthermore, the fact that it is based on estimation of moments yields immediate consistence by the law of large numbers. However, for complex distributions (e.g. mixtures) deriving equations that relate moments and methods parameters can be a painful exercise. Furthermore, such equations may not always be easy to solve algebraically. Therefore, despite being attractive from a theoretical point of view, the method of moments is not often used as the basis for deriving unsupervised learning algorithms.

Nonetheless, it has been recently observed in [AHK12a] that the spectral method for learning HMM is in fact an instance of the method of moments. The nice computational and statistical properties of the spectral method, interpreted under this light, come from the fact that – at least for all HMM where the basis (Σ, Σ) is complete – the algorithm is an instance of the moment method where only moments of order at most three are used. This contrasts with previous moment algorithms that relied on the estimation of high-order moments, which are usually hard to estimate accurately due to their large variances. Furthermore, since the equations relating parameters and moments can be solved via SVD decomposition and other linear algebra operations, the error incurred by using moment estimations in such equations can be bounded with standard perturbation tools.

Thus, we see that the spectral method for probabilistic models is an instance of a well-known method in statistics. Furthermore, since the particularities of the models to which the method is applied make it possible to learn the parameters using robust linear algebra techniques applied to low-order moments, it turns out to be a computationally and statistically appealing method. The main problem with the method, which is not shared with ML estimation, is the difficulty on generalizing it to new models and deriving variations that incorporate previous knowledge. Our goal in this chapter is to tackle this problem by taking the particular method of moments described by the spectral method and framing it as an optimization problem.

8.2 Consistent Learning via Local Loss Optimization

In this section we present a learning algorithm for WFA based on solving an optimization problem. In particular, the algorithm will minimize a certain loss function defined in terms of observed data about the target WFA. Casting learning algorithms as optimization problems is a common approach in machine learning, which has led to very successful classes of algorithms based on principles like empirical and structural risk minimization. In a sense that will soon become clear, our algorithm can be thought as a reinterpretation of the spectral method in terms of loss minimization. Though not entirely practical

due to the quadratic nature of the loss and constraints present in the optimization problem, this point of view on the spectral method yields interesting insights which are valuable for designing new classes of algorithms. This will be the subject of the following sections.

In spirit, our algorithm is similar to the spectral method in the sense that in order to learn a function $f : \Sigma^\star \to \mathbb{R}$ of finite rank, the algorithm infers a WFA using (approximate) information from a sub-block of $\mathbf{H}_f$. The sub-block used by the algorithm is defined in terms of a set of prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$. Throughout this section we assume that f is fixed and has rank r, and that a basis $(\mathcal{P}, \mathcal{S})$ of f is given.

We will describe our algorithm under the hypothesis that sub-blocks $\mathbf{H}$ and $\{\mathbf{H}_\sigma\}_{\sigma\in\Sigma}$ of $\mathbf{H}_f$ are known exactly. It is trivial to modify the algorithm to work in the case when only *approximations* $\hat{\mathbf{H}}$ and $\{\hat{\mathbf{H}}_\sigma\}_{\sigma\in\Sigma}$ of the Hankel sub-blocks are known.

Given $1 \leq n \leq s = |\mathcal{S}|$ we define the *local loss* function $\ell_n(\mathbf{X}, \boldsymbol{\beta}_\infty, \{\mathbf{B}_\sigma\})$ on variables $\mathbf{X} \in \mathbb{R}^{s\times n}$, $\boldsymbol{\beta}_\infty \in \mathbb{R}^n$ and $\mathbf{B}_\sigma \in \mathbb{R}^{n\times n}$ for $\sigma \in \Sigma$ as:

$$\ell_n = \|\mathbf{H}\mathbf{X}\boldsymbol{\beta}_\infty - \mathbf{h}_{\mathcal{P},\lambda}\|_2^2 + \sum_{\sigma\in\Sigma} \|\mathbf{H}\mathbf{X}\mathbf{B}_\sigma - \mathbf{H}_\sigma\mathbf{X}\|_F^2 \tag{8.1}$$

Our learning algorithm is a constrained minimization of this local loss, which we call *spectral optimization* (SO):

$$\min_{\mathbf{X},\boldsymbol{\beta}_\infty,\{\mathbf{B}_\sigma\}} \ell_n(\mathbf{X}, \boldsymbol{\beta}_\infty, \{\mathbf{B}_\sigma\}) \text{ s.t. } \mathbf{X}^\top\mathbf{X} = \mathbf{I} \tag{SO}$$

Intuitively, this optimization tries to *jointly* solve the optimizations solved by SVD and pseudo-inverse in the spectral method based on Lemma 5.2.1. In particular, likewise for the SVD-based method, it can be shown that (SO) is consistent whenever $n \geq r$ and $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is complete. This shows that the algorithms are in some sense equivalent, and thus provides a novel interpretation of spectral learning algorithms as minimizing a loss function on *local* data – when contrasted to any algorithm based on maximum likelihood – in the sense that only examples contained in the basis are considered in the loss function.

Theorem 8.2.1. *Suppose $n \geq r$ and $\mathcal{B}$ is a complete basis. Then, for any solution $(\mathbf{X}^*, \boldsymbol{\beta}^*_\infty, \{\mathbf{B}^*_\sigma\})$ to optimization problem* (SO)*, the WFA $B^* = \langle \mathbf{h}_{\lambda,\mathcal{S}}^\top \mathbf{X}^*, \boldsymbol{\beta}^*_\infty, \{\mathbf{B}^*_\sigma\}\rangle$ satisfies $f = f_{B^*}$*

The proof of this theorem is given in Section 8.2.1. Though the proof is relatively simple in the case $n = r$, it turns out that the case $n > r$ is much more delicate – unlike in the SVD-based method, where the same proof applies to all $n \geq r$.

Of course, if $\mathbf{H}$ and $\{\mathbf{H}_\sigma\}$ are not fully known, but approximations $\hat{\mathbf{H}}$ and $\{\hat{\mathbf{H}}_\sigma\}$ are given to the algorithm, we can still minimize the *empirical local loss* $\hat{\ell}_n$ and build a WFA from the solution using the same method of Theorem 8.2.1.

Despite its consistency, in general the optimization (SO) is not algorithmically tractable because its objective function is quadratic non-positive semidefinite and the constraint on $\mathbf{X}$ is not convex. Nonetheless, the proof of Theorem 8.2.1 shows that when $\mathbf{H}$ and $\{\mathbf{H}_\sigma\}$ are known exactly, the SVD method can be used to efficiently compute an optimal solution of (SO). Furthermore, the SVD method can be regarded as an approximate solver for (SO) with an empirical loss function $\hat{\ell}_n$ as follows. Find first an $\hat{\mathbf{X}}$ satisfying the constraints using the SVD of $\hat{\mathbf{H}}$, and then compute $\hat{\boldsymbol{\beta}}_\infty$ and $\{\hat{\mathbf{B}}_\sigma\}$ by minimizing the loss (8.1) with fixed $\hat{\mathbf{X}}$ – note that in this case, the optimization turns out to be convex. This is just one iteration of a general heuristic method known as *alternate minimization* that can be applied to quadratic objective functions on two variables where the objective is convex when one of the two variables is considered fixed. In this case the SVD provides a clever, though costly, initialization to the variable that will be fixed during the first iteration.

From the perspective of an optimization algorithm, the bounds for the distance between operators recovered with full and approximate data given in Section 5.4 for the spectral method, can be restated as a sensitivity analysis of the optimization solved by the algorithm given here. In fact, a similar analysis can be carried out for (SO), though we shall not pursue this direction here.

8.2.1 Proof of Theorem 8.2.1

The following technical result will be used in the proof.

Lemma 8.2.2. *Let $f : \Sigma^\star \to \mathbb{R}$ be a function of finite rank r and suppose that $(\mathcal{P}, \mathcal{S})$ is a complete basis for f. Then the matrix $\mathbf{H}_\Sigma = [\mathbf{H}_{\sigma_1} \dots \mathbf{H}_{\sigma_{|\Sigma|}}]$ has rank r.*

Proof. Let $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ be a minimal automaton for f and denote by $\mathbf{H} = \mathbf{PS}$ the rank factorization induced by A. Since by construction we have $\mathbf{H}_\Sigma = \mathbf{P}\,[\mathbf{A}_{\sigma_1}\mathbf{S}, \dots, \mathbf{A}_{\sigma_{|\Sigma|}}\mathbf{S}]$, suffices to show that $\operatorname{rank}([\mathbf{A}_{\sigma_1}\mathbf{S} \dots \mathbf{A}_{\sigma_{|\Sigma|}}\mathbf{S}]) = r$. Since each column of $\mathbf{S}$ is of the form $\mathbf{s}_v = \mathbf{A}_v\boldsymbol{\alpha}_\infty$ for some suffix $v \in \mathcal{S}$, and $\operatorname{rank}(\mathbf{S}) = r$, we have $\mathbb{R}^r = \operatorname{span}(\{\mathbf{s}_v\}_{v\in\mathcal{S}}) \subseteq \cup_{\sigma\in\Sigma} \operatorname{range}(\mathbf{A}_\sigma)$. Now, since for all $\sigma \in \Sigma$ one has $\operatorname{span}(\{\mathbf{A}_\sigma\mathbf{s}_v\}_{v\in\mathcal{S}}) = \operatorname{range}(\mathbf{A}_\sigma)$, we see that the columns of $[\mathbf{A}_{\sigma_1}\mathbf{S}, \dots, \mathbf{A}_{\sigma_{|\Sigma|}}\mathbf{S}]$ span $\cup_{\sigma\in\Sigma} \operatorname{range}(\mathbf{A}_\sigma) = \mathbb{R}^r$. From which it follows that $\operatorname{rank}(\mathbf{H}_\Sigma) = r$. □

The proof of Theorem 8.2.1 is divided into the following four claims.

Claim 8.1. The optimal value of problem (SO) is zero.

Let $\mathbf{H} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top$ be a full SVD of $\mathbf{H}$ and write $\mathbf{V}_n \in \mathbb{R}^{s\times n}$ for the n right singular vectors corresponding to the n largest singular values (some of which might be zero since $n \geq r$). Note that we have $\mathbf{V}_n^\top\mathbf{V}_n = \mathbf{I}$ and $\mathbf{H}\mathbf{V}_n\mathbf{V}_n^\top = \mathbf{H}$. We will see that using these matrices we can construct a solution to (SO) with zero loss:

$$\ell_n(\mathbf{V}_n, (\mathbf{H}\mathbf{V}_n)^+\mathbf{h}_{\mathcal{P},\lambda}, \{(\mathbf{H}\mathbf{V}_n)^+\mathbf{H}_\sigma\mathbf{V}_n\}) = 0 \ .$$

Recall that $\mathbf{H}\mathbf{V}_n(\mathbf{H}\mathbf{V}_n)^+$ acts as an identity in the span of the columns of $\mathbf{H}\mathbf{V}_n$. Thus, writing $\mathbf{h}_{\mathcal{P},\lambda} = \mathbf{H}\mathbf{V}_n\mathbf{V}_n^\top\mathbf{e}_\lambda$ w can see that

$$(\mathbf{H}\mathbf{V}_n)(\mathbf{H}\mathbf{V}_n)^+\mathbf{h}_{\mathcal{P},\lambda} = \mathbf{h}_{\mathcal{P},\lambda} \ .$$

Furthermore, since $\operatorname{rank}([\mathbf{H}, \mathbf{H}_\sigma]) = \operatorname{rank}(\mathbf{H})$, we have

$$\mathbf{H}\mathbf{V}_n(\mathbf{H}\mathbf{V}_n)^+\mathbf{H}_\sigma\mathbf{V}_n = \mathbf{H}_\sigma\mathbf{V}_n \ .$$

This verifies the claim.

Claim 8.2. For any $n \geq r$, $A_n = \langle \mathbf{h}_{\lambda,\mathcal{S}}^\top\mathbf{V}_n, (\mathbf{H}\mathbf{V}_n)^+\mathbf{h}_{\mathcal{P},\lambda}, \{(\mathbf{H}\mathbf{V}_n)^+\mathbf{H}_\sigma\mathbf{V}_n\}\rangle$ satisfies $f_{A_n} = f$.

We will show that for any $n > r$ one has $f_{A_n} = f_{A_r}$. Then the claim will follow from Lemma 5.2.1, since A_r is the WFA corresponding to the rank factorization $\mathbf{H} = (\mathbf{H}\mathbf{V}_r)(\mathbf{V}_r^\top)$. Write $\boldsymbol{\Pi}_n = [\mathbf{I}_r, \mathbf{0}] \in \mathbb{R}^{r\times n}$. Since any singular vector in $\mathbf{V}_n$ which is not in $\mathbf{V}_r$ is orthogonal to the rowspace of $\mathbf{H}$ and $\mathbf{H}_\sigma$, by construction we have $\boldsymbol{\Pi}_n A_n \boldsymbol{\Pi}_n^\top = A_r$. Now we consider the factorization $\mathbf{H} = \mathbf{P}_n\mathbf{S}_n$ induced by A_n. If we show that the rows of $\mathbf{P}_n$ lie in the span of the rows of $\mathbf{H}\mathbf{V}_n$, then $\mathbf{P}_n\boldsymbol{\Pi}_n^\top\boldsymbol{\Pi}_n = \mathbf{P}_n$ and Lemma 5.2.3 tells us that $f_{A_n} = f_{A_r}$. Write $A_n = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$. We prove by induction on $|u|$ that $\alpha_u^\top = \boldsymbol{\alpha}_0^\top\mathbf{A}_u$ lie in the span of the rows of $\mathbf{H}\mathbf{V}_n$, which we denote by $\mathcal{H}$. This will imply the claim about the rows of $\mathbf{P}_n$. For $|u| = 0$ we trivially have $\boldsymbol{\alpha}_0^\top = \mathbf{h}_{\lambda,\mathcal{S}}^\top\mathbf{V}_n \in \mathcal{H}$. Furthermore, by the induction hypothesis we have $\alpha_u^\top = \gamma_u^\top\mathbf{H}\mathbf{V}_n$ for some $\gamma_u \in \mathbb{R}^p$. Thus we get

$$\alpha_{u\sigma}^\top = \alpha_u^\top\mathbf{A}_\sigma = \gamma_u^\top\mathbf{H}\mathbf{V}_n(\mathbf{H}\mathbf{V}_n)^+\mathbf{H}_\sigma\mathbf{V}_n = \gamma_u^\top\mathbf{H}_\sigma\mathbf{V}_n \in \mathcal{H} \ .$$

Claim 8.3. Let $(\mathbf{X}^*, \boldsymbol{\beta}_\infty^*, \{\mathbf{B}_\sigma^*\})$ be an optimal solution to (SO) and define $B' = \langle \mathbf{h}_{\lambda,\mathcal{S}}^\top\mathbf{X}^*, \boldsymbol{\beta}'_\infty, \{\mathbf{B}'_\sigma\}\rangle$, where $\mathbf{B}'_\sigma = (\mathbf{H}\mathbf{X}^*)^+\mathbf{H}_\sigma\mathbf{X}^*$ and $\boldsymbol{\beta}'_\infty = (\mathbf{H}\mathbf{X}^*)^+\mathbf{h}_{\mathcal{P},\lambda}$. Then $f_{B^*} = f_{B'}$.

Since the optimal value of the objective is zero, we must have $\mathbf{H}\mathbf{X}^*\mathbf{B}_\sigma^* = \mathbf{H}_\sigma\mathbf{X}^*$ and $\mathbf{H}\mathbf{X}^*\boldsymbol{\beta}_\infty^* = \mathbf{h}_{\mathcal{P},\lambda}$. Thus, by properties of the Moore–Penrose pseudo-inverse we must have that $\mathbf{B}_\sigma^* = \mathbf{B}'_\sigma + \mathbf{C}'_\sigma$ and $\boldsymbol{\beta}_\infty^* = \boldsymbol{\beta}'_\infty + \gamma'$, where

$$\mathbf{C}'_\sigma = (\mathbf{I} - (\mathbf{H}\mathbf{X}^*)^+\mathbf{H}\mathbf{X}^*)\mathbf{C}_\sigma \ ,$$
$$\gamma' = (\mathbf{I} - (\mathbf{H}\mathbf{X}^*)^+\mathbf{H}\mathbf{X}^*)\gamma \ ,$$

for some arbitrary $\mathbf{C}_\sigma \in \mathbb{R}^{n\times n}$ and $\gamma \in \mathbb{R}^n$. Now the claim follows from similar arguments as the ones used above, showing by induction on the length of u that $(\beta_u^*)^\top = \mathbf{h}_{\lambda,\mathcal{S}}^\top\mathbf{X}^*\mathbf{B}_u^*$ lies in the span of the rows of $\mathbf{H}\mathbf{X}^*$.

Claim 8.4. With notation from previous claims, we have $f_{B'} = f_{A_n}$

In the first place, note that Lemma 8.2.2 and the equation

$$\mathbf{H}\mathbf{X}^* \left[\mathbf{B}'_{\sigma_1}, \ldots, \mathbf{B}'_{\sigma_{|\Sigma|}}\right] = [\mathbf{H}_{\sigma_1}\mathbf{X}^*, \ldots, \mathbf{H}_{\sigma_m}\mathbf{X}^*]$$

necessarily imply $\mathbf{H}\mathbf{X}^*(\mathbf{X}^*)^\top = \mathbf{H}$. Using this property, we can show that $B' = \mathbf{N}A_n\mathbf{M}$ with $\mathbf{N} = (\mathbf{X}^*)^\top\mathbf{V}_n$ and $\mathbf{M} = \mathbf{V}_n^\top\mathbf{X}^*$. Thus, the claim follows from Lemma 5.2.3, where the condition $\mathbf{P}_n\mathbf{M}\mathbf{N} = \mathbf{P}_n$ can be verified via an induction argument on the length of prefixes.

Summarizing, the consistency of the algorithm given by (SO) follows from the chain of equalities $f_{B^*} = f_{B'} = f_{A_n} = f_{A_r} = f$ proved in the previous claims.

8.3 A Convex Relaxation of the Local Loss

We shall now present a convex relaxation of the optimization on (SO). Besides the obvious advantage represented by having a convex problem, our relaxation addresses another practical issue: that the only parameter a user can adjust in (SO) in order to trade accuracy and model complexity is the number of states n. Though the discreteness of this parameter allows for a fast model selection scheme through a full exploration of the parameter space, in some applications one may be willing to invest some time in exploring a larger, more fine-grained space of parameters, with the hope of reaching a better trade-off between accuracy and model complexity. The algorithm presented here does this by incorporating a continuous regularization parameter.

The main idea in order to obtain a convex optimization problem similar to (SO) will be to remove the projection $\mathbf{X}$, since we have already seen that it is the only source of non-convexity in the optimization. However, the new convex objective will need to incorporate a term that enforces the optimization to behave in a similar way as (SO). In particular we need to enforce that the operators of the model we learn have their dynamics inside the space spanned by the rows of $\mathbf{H}$ and $\mathbf{H}_\sigma$.

First note that the choice of n effectively restricts the maximum rank of the operators $\mathbf{B}_\sigma$. Once this maximal rank is set, $\mathbf{X}$ can be interpreted as enforcing a common "semantic space" between the different operators $\mathbf{B}_\sigma$ by making sure each of them works on a state space defined by the same projection of $\mathbf{H}$. Furthermore, the constraint on $\mathbf{X}$ tightly controls its norm and thus ensures that the operators $\mathbf{B}_\sigma$ will also have its norm tightly controlled to be in the order of $\|\mathbf{H}_a\|/\|\mathbf{H}\|$ – at least when $n = r$, see the proof of Theorem 8.2.1.

Thus, in order to obtain a convex optimization similar to (SO) we do the following. First, take $n = s$ and fix $\mathbf{X} = \mathbf{I}_s$, thus unrestricting the model class and removing the source of non-convexity. Then penalize the resulting objective with a convex relaxation of the term $\operatorname{rank}([\mathbf{B}_{\sigma_1}, \ldots, \mathbf{B}_{\sigma_{|\Sigma|}}])$, which makes sure the operators have low rank individually, and enforces them to work on a common low-dimensional state space. As a relaxation of $\operatorname{rank}(\mathbf{M})$ we us the *nuclear norm* $\|\mathbf{M}\|_\star$ (also known as *trace norm*), which is known to provide a good convex proxy for the rank function in rank minimization problems [Faz02]. The intuition behind this fact in the expression $\|\mathbf{M}\| = \sum_i \mathfrak{s}_i(\mathbf{M})$ showing the nuclear norm is in fact a Schatten norm which corresponds to the L_1 norm of the singular values of $\mathbf{M}$.

More formally, for any regularization parameter $\tau > 0$, the *relaxed local loss* $\tilde{\ell}_\tau(\mathbf{B}_\Sigma)$ on a matrix variable $\mathbf{B}_\Sigma \in \mathbb{R}^{s\times|\Sigma|s}$ is defined as:

$$\tilde{\ell}_\tau = \|\mathbf{B}_\Sigma\|_\star + \tau\|\mathbf{H}\mathbf{B}_\Sigma - \mathbf{H}_\Sigma\|_F^2 \ ,$$

where $\mathbf{B}_\Sigma = [\mathbf{B}_{\sigma_1}, \ldots, \mathbf{B}_{\sigma_{|\Sigma|}}]$ is a block-concatenation of the operators, and $\mathbf{H}_\Sigma = [\mathbf{H}_{\sigma_1}, \ldots, \mathbf{H}_{\sigma_{|\Sigma|}}]$. Since $\tilde{\ell}$ is clearly convex on $\mathbf{B}_\Sigma$, we can learn a set of operators by solving the convex optimization problem

$$\min_{\mathbf{B}_\Sigma} \tilde{\ell}(\mathbf{B}_\Sigma) \ . \tag{CO}$$

Given an optimal solution $\mathbf{B}^*_\Sigma$ of (CO), we define a WFA $B^* = \langle \mathbf{h}_{\lambda,\mathcal{S}}^\top, \mathbf{e}_\lambda, \mathbf{B}^*_\Sigma\rangle$, where $\mathbf{e}_\lambda \in \mathbb{R}^s$ is the coordinate vector with $\mathbf{e}_\lambda(\lambda) = 1$. Some useful facts about this optimization are collected in the following proposition.

Proposition 8.3.1. *The following hold:*

1. *If* $\mathbf{H}$ *has full column rank, then* (CO) *has a unique solution*

2. *For* $n = s$ *and* $\tau \geq 1$*, the optimum value* ℓ_s^* *of* (SO) *and the optimum value* $\tilde{\ell}_\tau^*$ *of* (CO) *satisfy* $\ell_s^* \leq \tilde{\ell}_\tau^*$

3. *Suppose* $\operatorname{rank}(\mathbf{H}) = \operatorname{rank}([\mathbf{H}_\Sigma, \mathbf{H}])$ *and let* $[\mathbf{H}_\Sigma, \mathbf{H}] = \mathbf{U}\Lambda[\mathbf{V}_\Sigma^\top \mathbf{V}^\top]$ *be a compact SVD. Then,* $\mathbf{B}_\Sigma = (\mathbf{V}^\top)^+ \mathbf{V}_\Sigma^\top$ *is a closed form solution for* (CO) *when* $\tau \to \infty$

Proof. Fact (1) follows from the observation that when $\mathbf{H}$ has full rank the loss $\tilde{\ell}_\tau$ is strictly convex. To see this note that $\|\mathbf{B}_\Sigma\|_\star$ is strictly convex by definition and let $g(\mathbf{B}) = \|\mathbf{HB} - \mathbf{H}'\|_F^2$ for some fixed $\mathbf{H}$ and $\mathbf{H}'$. Then, for any $\mathbf{B}$ and $\mathbf{B}'$, and any $0 \leq t \leq 1$ we have

$$
\begin{aligned}
&g(t\mathbf{B} + (t-1)\mathbf{B}') \\
&\quad = \|t\mathbf{HB} + (1-t)\mathbf{HB}' - \mathbf{H}'\|_F^2 \\
&\quad = \|t(\mathbf{HB} - \mathbf{H}') + (1-t)(\mathbf{HB}' - \mathbf{H}')\|_F^2 \\
&\quad = t^2\|\mathbf{HB} - \mathbf{H}'\|_F^2 + (1-t)^2\|\mathbf{HB}' - \mathbf{H}'\|_F^2 \\
&\qquad + 2t(1-t)\langle \mathbf{HB} - \mathbf{H}', \mathbf{HB}' - \mathbf{H}' \rangle \\
&\quad \leq t^2 g(\mathbf{B}) + (1-t)^2 g(\mathbf{B}') + 2t(1-t)\sqrt{g(\mathbf{B})}\sqrt{g(\mathbf{B}')} \qquad (8.2) \\
&\quad \leq t g(\mathbf{B}) + (1-t) g(\mathbf{B}') \ , \qquad (8.3)
\end{aligned}
$$

where we have used the Cauchy–Schwarz inequality and that for any $x, y \geq 0$ the following holds

$$
0 \geq t(t-1)(x-y)^2 = t^2x^2 + (1-t)^2y^2 + 2t(1-t)xy - tx^2 - (1-t)y^2 \ .
$$

Note that (8.2) is an equality if and only if either $\mathbf{HB} - \mathbf{H}' = \alpha(\mathbf{HB}' - \mathbf{H}')$ for some $\alpha > 0$, or one of $\mathbf{HB} - \mathbf{H}' = \mathbf{0}$ or $\mathbf{HB}' - \mathbf{H}' = \mathbf{0}$ holds. Furthermore (8.3) is an equality if and only if $g(\mathbf{B}) = g(\mathbf{B}')$. Thus, we see that we have an overall equality if and only if $\mathbf{HB} = \mathbf{HB}'$, which implies $\mathbf{B} = \mathbf{B}'$ when $\mathbf{H}$ has full column rank. For fact (2), suppose $\mathbf{B}_\Sigma^*$ achieves the optimal value in (CO) and check that $\ell_s^* \leq \ell_s(\mathbf{I}, \mathbf{e}_\lambda, \mathbf{B}_\Sigma^*) = \|\mathbf{HB}_\Sigma^* - \mathbf{H}_\Sigma\|_F^2 \leq \tilde{\ell}_\tau^*$. Fact (3) follows from Theorem 2.1 in [LSY10] and the observation that when $\tau \to \infty$ optimization (CO) is equivalent to $\min_{\mathbf{B}_\Sigma} \|\mathbf{B}_\Sigma\|_\star$ s.t. $\mathbf{HB}_\Sigma = \mathbf{H}_\Sigma$. □

Note that in general approximations $\hat{\mathbf{H}}$ of $\mathbf{H}$ computed from samples will have full rank with high probability. Thus, fact (1) tells us that either in this case, or when $p = n$, optimization (CO) has a unique optimum. Furthermore, by fact (2) we see that minimizing the convex loss is also, in a relaxed sense, minimizing the non-convex loss which is known to be consistent. In addition, fact (3) implies that when $\mathbf{H}$ has full rank and τ is very large, we recover the spectral method with $n = s$.

8.4 Practical Considerations

Convex optimization problems can in general be (approximately) solved in polynomial time by generic algorithms, like the ellipsoid method. However, these general approach rarely yields practically efficient algorithms. For problems with specific forms, different optimizations algorithms that explicitly exploit the structure in the problem have been designed. In this section we provide two approaches for solving optimization (CO) in practice.

Framing the problem of WFA learning as a convex optimization problem has a very interesting consequence: now we can enforce different forms of prior knowledge on the hypothesis of our learning algorithm by adding convex constraints and regularizers to (CO). We will mention two particular examples here, but note that there are many other possibilities.

8.4.1 A Semidefinite Programming Approach

Optimization (CO) can be restated in several ways. In particular, by standard techniques, it can be shown that it is equivalent to a Conic Program on the intersection of a semi-definite cone (given by the nuclear norm), and a quadratic cone (given by the Frobenius norm). Similarly, the problem can also be fully expressed as a semi-definite program. Though in general this conversion is believed to be inefficient for algorithmic purposes, it has the advantage of giving the problem in a form that can be solved by many general-purpose optimization packages for semidefinite programming.

In this section we show how to obtain a semidefinite program for the following variant of (CO) where we have removed the square on the Frobenius norm:

$$\min_{\mathbf{B}_\Sigma} \|\mathbf{B}_\Sigma\|_\star + \tau\|\mathbf{H}\mathbf{B}_\Sigma - \mathbf{H}_\Sigma\|_F \ . \tag{CO'}$$

We begin by introducing some notation. For any matrix $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ we use $\mathrm{tr}(\mathbf{M})$ to denote its trace and $\mathrm{vec}(\mathbf{M})$ to denote a column vector in $\mathbb{R}^{d_1 d_2}$ containing all the elements of $\mathbf{M}$. Recall that a symmetric matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ is positive semidefinite if all its eigenvalues are non-negative, a fact we denote by $\mathbf{M} \succeq \mathbf{0}$. Now we state without proof two well-known facts about semidefinite programming that provide representations for the Frobenius and nuclear norm.

Lemma 8.4.1 ([BV04])**.** *For any $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ and $t \geq 0$ one has $\|\mathbf{M}\|_F \leq t$ if and only if*

$$\begin{bmatrix} t \cdot \mathbf{I} & \mathrm{vec}(\mathbf{M}) \\ \mathrm{vec}(\mathbf{M})^\top & t \end{bmatrix} \succeq \mathbf{0} \ .$$

Lemma 8.4.2 ([Faz02])**.** *For any $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ and $t \geq 0$ one has $\|\mathbf{M}\|_\star \leq t$ if and only if there exist symmetric matrices $\mathbf{X}_1 \in \mathbb{R}^{d_1 \times d_1}$ and $\mathbf{X}_2 \in \mathbb{R}^{d_2 \times d_2}$ such that $\mathrm{tr}(\mathbf{X}_1) + \mathrm{tr}(\mathbf{X}_2) \leq 2t$ and*

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{M} \\ \mathbf{M}^\top & \mathbf{X}_2 \end{bmatrix} \succeq \mathbf{0} \ .$$

Combining these two results with a standard reduction one can prove the following.

Theorem 8.4.3. *Optimization problem* (CO') *is equivalent to the following semidefinite program:*

$$\begin{aligned} &\min_{\mathbf{B}_\Sigma, t, \mathbf{X}_1, \mathbf{X}_2} \frac{1}{2}(\mathrm{tr}(\mathbf{X}_1) + \mathrm{tr}(\mathbf{X}_2)) + \tau t \\ &\text{subject to} \quad \begin{bmatrix} \mathbf{X}_1 & \mathbf{B}_\Sigma & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_\Sigma^\top & \mathbf{X}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & t \cdot \mathbf{I} & \mathrm{vec}(\mathbf{H}\mathbf{B}_\Sigma - \mathbf{H}_\Sigma) \\ \mathbf{0} & \mathbf{0} & \mathrm{vec}(\mathbf{H}\mathbf{B}_\Sigma - \mathbf{H}_\Sigma)^\top & t \end{bmatrix} \succeq \mathbf{0} \ . \end{aligned}$$

The above problem has $O(ps|\Sigma| + p^2 + s^2|\Sigma|^2)$ variables and a constraint space of dimension $(p + s|\Sigma| + ps|\Sigma| + 1)^2$. Though general solvers may not be able to deal with this problem for large basis and alphabets, perhaps a special-purpose interior-point method could be obtained to solve this problem efficiently (e.g. see [ADLV]).

8.4.2 A Proximal Gradient Optimization Algorithm

Besides reducing an optimization problem to a standard form like in previous section, another way to obtain efficient algorithms for solving this type of problems is to exploit their particular structure. In our case it turns out that the objective function in (CO) falls into the category of so-called *separable* convex objectives. These can be basically decomposed as the sum of a convex smooth term – the Frobenius norm in our case – and a convex non-smooth term – the nuclear norm. In this section we briefly discuss how to fit (CO) into the setting of *proximal gradient* optimization; for more details, convergence proofs, and stopping conditions we direct the reader to [Nes07; BT09].

Let us assume that one needs to solve the following optimization problem of the form

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) + g(\mathbf{x}) \ , \tag{8.4}$$

where the loss function $\ell(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$ is expressed as the sum of a smooth convex function $f(\mathbf{x})$ and a convex and possibly non-smooth $g(\mathbf{x})$. We also assume that the first derivative of f is Lipschitz continuous; that is, there exists a constant $L > 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ we have

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \ .$$

The *proximal operator* for g parametrized by a positive number η is the non-linear map $P_{g,\eta} : \mathbb{R}^d \to \mathbb{R}^d$ defined as

$$P_{g,\eta}(\mathbf{y}) = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^d} \left(g(\mathbf{x}) + \frac{1}{2\eta}\|\mathbf{x} - \mathbf{y}\|^2 \right) \ .$$

Assuming the gradient ∇f and the proximal operator $P_{g,\eta}$ can be computed exactly, and (a bound for) L is known, the following proximal-gradient iterative descent scheme provides a fast converging algorithm for solving (8.4). Choose $\mathbf{x}_0 \in \mathbb{R}^d$, let $\mathbf{y}_0 = \mathbf{x}_0$, and take $\gamma_0 = 1$. Then for $k \geq 0$ iterate the following:

$$\begin{aligned} \mathbf{x}_{k+1} &= P_{g,1/L}\left(\mathbf{y}_k - \frac{\nabla f(\mathbf{y}_k)}{L} \right) \ , \\ \gamma_{k+1} &= \frac{1 + \sqrt{1 + 4\gamma_k^2}}{2} \ , \\ \mathbf{y}_{k+1} &= \mathbf{x}_{k+1} + \frac{\gamma_k - 1}{\gamma_{k+1}}(\mathbf{x}_{k+1} - \mathbf{x}_k) \ . \end{aligned} \tag{PGD}$$

This particular scheme is called FISTA and was introduced in [BT09]. It was shown there that $\mathbf{x}_k \to \mathbf{x}^*$ for some optimal $\mathbf{x}^*$ point of (8.4), and that this convergence occurs at a rate such that $|\ell(\mathbf{x}_k) - \ell(\mathbf{x}^*)| \leq O(1/k^2)$.

Note that in the particular case of (CO) we have $f(\mathbf{B}_\Sigma) = \tau\|\mathbf{H}\mathbf{B}_\Sigma - \mathbf{H}_\Sigma\|_F^2$ and $g(\mathbf{B}_\Sigma) = \|\mathbf{B}_\Sigma\|_\star$. In order to apply the proximal-gradient optimization to problem (CO) we first observe that

$$\nabla f(\mathbf{B}_\Sigma) = 2\tau\mathbf{H}^\top(\mathbf{H}\mathbf{B}_\Sigma - \mathbf{H}_\Sigma) \ ,$$

which is Lipschitz continuous with $L = 2\tau\|\mathbf{H}^\top\mathbf{H}\|_F$. Now, the proximal operator associated to g has a closed-form expression in terms of a *singular value shrinkage* operator. Let $\mathbf{M} \in \mathbb{R}^{d_1\times d_2}$ be a matrix of rank r with SVD given by $\mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$ where $\mathbf{\Lambda} \in \mathbb{R}^{r\times r}$ is a diagonal matrix with entries $(\mathfrak{s}_1, \ldots, \mathfrak{s}_r)$. The shrinkage operator with parameter $\eta > 0$ applied to matrix $\mathbf{M}$ is defined as

$$\operatorname{shr}_\eta(\mathbf{M}) = \mathbf{U}\operatorname{shr}_\eta(\mathbf{\Lambda})\mathbf{V}^\top \ ,$$

where $\operatorname{shr}_\eta(\mathbf{\Lambda}) \in \mathbb{R}^{r\times r}$ is a diagonal matrix with entries $\operatorname{shr}_\eta(\mathbf{\Lambda})(i,i) = \max\{\mathfrak{s}_i - \eta, 0\}$. With these definitions, it was shown in [CCS10] that for any $\mathbf{M}$ and η on has

$$P_{\|\cdot\|_\star,\eta}(\mathbf{M}) = \operatorname{shr}_\eta(\mathbf{M}) \ .$$

Thus we can easily implement (PGD) to solve (CO) using these expressions for ∇f and $P_{g,\eta}$.

Note that the naive implementation of this scheme requires an SVD computation at each iteration. Clearly, this yields a method which is computationally worse than the standard spectral method which only requires the computation of a single SVD. However, schemes like (PGD) are known to converge even if only an approximate computation of $P_{g,\eta}$ is available [SLRB11]. Thus, using fast approximations of SVD to compute the proximal operator of the nuclear norm in a proximal-gradient descent method can yield efficient alternatives to the spectral method with the advantages of being stated as an optimization problem; a good survey paper on approximate matrix factorization is [HMT11].

8.4.3 Convex Constraints for Learning Probabilistic Automaton

We already mentioned in Chapter 6 that in general the output of spectral algorithms for learning stochastic WFA is a WFA that does not necessarily realize a probability distribution. We know it will approximate a probability distribution by our PAC results, but in general it can assign negative "probabilities" to some strings. This is not only a theoretical problem, but it has been reported in several experimental studies [JZKOPK05; BQC12; CSCFU13]. Sometimes this makes it impossible to directly evaluate models learned with spectral methods using entropy-based measures like the perplexity. Besides the use of several heuristics in the experimental studies we just mentioned, some alternative spectral methods have been derived to address this problem [ZJ10; Bai11b]. These solutions are based on restricting the class of models used as a hypothesis class by imposing a particular structure on the possible hypothesis which ensures that they assign non-negative outputs to every string. Here we present another solution along these lines based on our convex optimization problem (CO). The hypothesis class of our algorithm is that of all PNFA. Thus, we believe that our solution to this problem is much more general than all previous ones.

The main idea behind our approach is to take optimization (CO) and impose a set of convex constraints on the operator matrix $\mathbf{A}_\Sigma$ that ensures the output represents a probabilistic automaton. In order to do that, it is convenient to re-derive our optimization algorithm starting from a different variant of the spectral method than the one we gave in Section 5.3. In particular, we note that taking the rank factorization $\mathbf{H}_\lambda = \mathbf{U}(\mathbf{\Lambda}\mathbf{V}^\top)$ of a Hankel matrix of rank r given by a compact SVD decomposition, Lemma 5.2.1 yields the following spectral method:

$$\begin{aligned}\boldsymbol{\alpha}_0^\top &= \mathbf{h}_{\lambda,\mathcal{S}}^\top (\mathbf{U}^\top \mathbf{H}_\lambda)^+ \ , \\ \boldsymbol{\alpha}_\infty &= \mathbf{U}^\top \mathbf{h}_{\mathcal{P},\lambda} \ , \\ \mathbf{A}_\sigma &= \mathbf{U}^\top \mathbf{H}_\sigma (\mathbf{U}^\top \mathbf{H}_\lambda)^+ \ .\end{aligned}$$

Now, if one goes through all the derivations in Sections 8.2 and 8.3 using this spectral method as a starting point, one ends up with the following optimization problem:

$$\min_{\mathbf{B}_\Sigma} \|\mathbf{B}_\Sigma\|_\star + \tau \|\mathbf{B}_\Sigma \mathbf{H} - \mathbf{H}_\Sigma\|_F^2 \ , \tag{CO-U}$$

where now $\mathbf{B}_\Sigma \in \mathbb{R}^{p|\Sigma| \times p}$ is the matrix of operators $\mathbf{B}_\Sigma^\top = [\mathbf{B}_{\sigma_1}^\top, \ldots, \mathbf{B}_{\sigma_{|\Sigma|}}^\top]$, and $\mathbf{H}_\Sigma^\top = [\mathbf{H}_{\sigma_1}^\top, \ldots, \mathbf{H}_{\sigma_{|\Sigma|}}^\top]$. We note that the term inside the Frobenius norm is *not* the transpose of the term inside the Frobenius norm in (CO). In this case, if $\mathbf{B}_\Sigma^*$ is a solution to (CO-U), then the WFA computed by the method is given by $B^* = \langle \mathbf{e}_\lambda^\top, \mathbf{h}_{\mathcal{P},\lambda}, \mathbf{B}_\Sigma^* \rangle$. The interesting point here is that now $\boldsymbol{\beta}_0 = \mathbf{e}_\lambda$ can be regarded – without the need to perform any normalization – as a probability distribution over initial states that assigns probability 1 to a single state. In addition, $\boldsymbol{\beta}_\infty = \mathbf{h}_{\mathcal{P},\lambda}$ can be regarded as a vector of stopping probabilities. That is, the method automatically yields initial and final weights that meet the requirements of a probabilistic automaton; see Section 6.1.

Now we can modify optimization problem (CO-U) to impose that the operators $\mathbf{B}_\Sigma$ also satisfy the restrictions of a probabilistic automaton. In particular, recall that this entails the following:

1. $\mathbf{B}_\sigma(i,j) \geq 0$ for all $\sigma \in \Sigma$ and $i, j \in [p]$,
2. $\boldsymbol{\beta}_\infty(i) + \sum_{\sigma \in \Sigma} \sum_{j \in [p]} \mathbf{B}_\sigma(i,j) = 1$ for all $i \in [p]$.

It is immediate to see that these two conditions define a set of convex constraints on the operator matrix $\mathbf{B}_\Sigma$. So if $f : \Sigma^\star \to \mathbb{R}$ is a probability distribution and $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is a basis, given Hankel sub-blocks $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ and $\mathbf{H}_\Sigma \in \mathbb{R}^{\mathcal{P}\Sigma \times \mathcal{S}}$ from $\mathbf{H}_f$, we can try to find a probabilistic automaton for f by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{B}_\Sigma} \ & \|\mathbf{B}_\Sigma\|_\star + \tau \|\mathbf{B}_\Sigma \mathbf{H} - \mathbf{H}_\Sigma\|_F^2 \\ \text{subject to} \ & \mathbf{B}_\Sigma \geq 0 \ , \\ & \mathbf{h}_{\mathcal{P},\lambda}(u) + \sum_{\sigma \in \Sigma, v \in \mathcal{S}} \mathbf{B}_\Sigma(u\sigma, v) = 1 \quad \forall u \in \mathcal{P} \ . \end{aligned} \tag{CO-P}$$

We have already seen that trying to use generic convex optimization methods for solving large scale problems is not always a good idea. Thus, in the rest of this section we will give a specialization of the *alternating direction method of multipliers* (ADMM) for solving optimization (CO-P). This method is described at large in [BPCPE11], which also gives indications on how to parallelize it for large scale computations. For a given convex set $\mathcal{C}$ let $I_{\mathcal{C}}$ denote the convex indicator function for which $I_{\mathcal{C}}(x) = 0$ when $x \in \mathcal{C}$ and $I_{\mathcal{C}}(x) = \infty$ when $x \notin \mathcal{C}$. The key observation is that we can write problem (CO-P) as

$$\min_{\mathbf{B}_\Sigma} f_1(\mathbf{B}_\Sigma) + f_2(\mathbf{B}_\Sigma) + I_{\mathcal{C}_1}(\mathbf{B}_\Sigma) + I_{\mathcal{C}_2}(\mathbf{B}_\Sigma) \ ,$$

where $f_1(\mathbf{B}_\Sigma) = \|\mathbf{B}_\Sigma\|_\star$ and $f_2(\mathbf{B}_\Sigma) = \tau\|\mathbf{B}_\Sigma\mathbf{H} - \mathbf{H}_\Sigma\|_F^2$ are convex functions, and

$$\mathcal{C}_1 = \left\{ \mathbf{M} \in \mathbb{R}^{p|\Sigma| \times p} : \mathbf{M} \geq 0 \right\} \ ,$$

$$\mathcal{C}_2 = \left\{ \mathbf{M} \in \mathbb{R}^{p|\Sigma| \times p} : \forall u \in \mathcal{P} \ \ \mathbf{h}_{\mathcal{P},\lambda}(u) + \sum_{\sigma \in \Sigma, v \in \mathcal{S}} \mathbf{M}(u\sigma, v) = 1 \right\} \ ,$$

are convex sets. Using this formulation, ADMM's insight is that one can separate this problem into four parts and then impose a consensus on the solutions of each part. Writing $f_3 = I_{\mathcal{C}_1}$ and $f_4 = I_{\mathcal{C}_2}$, this translates into the following:

$$\begin{aligned} &\min_{\mathbf{B}_i, \mathbf{Z}} \ f_1(\mathbf{B}_1) + f_2(\mathbf{B}_2) + f_3(\mathbf{B}_3) + f_4(\mathbf{B}_4) \ , \\ &\text{subject to} \qquad \mathbf{B}_1 - \mathbf{Z} = \mathbf{0},\ \mathbf{B}_2 - \mathbf{Z} = \mathbf{0},\ \mathbf{B}_3 - \mathbf{Z} = \mathbf{0},\ \mathbf{B}_4 - \mathbf{Z} = \mathbf{0} \ . \end{aligned}$$

Introducing dual variables $\mathbf{Y}_i$ for $i \in \{1, 2, 3, 4\}$, this problem can be solved by an iterative scheme as follows. First, choose starting points $\mathbf{B}_i^0$, $\mathbf{Y}_i^0$, $\mathbf{Z}^0$ for $i \in \{1, 2, 3, 4\}$ and a rate parameter $\rho > 0$. Then, for $k \geq 0$ iterate the following:

$$\begin{aligned} \mathbf{B}_i^{k+1} &= \operatorname*{argmin}_{\mathbf{B}_i} \left(f_i(\mathbf{B}_i) + \frac{\rho}{2}\|\mathbf{B}_i - \mathbf{Z}^k + \mathbf{Y}_i^k\|_F^2 \right) \ , \\ \mathbf{Z}^{k+1} &= \frac{1}{4} \sum_{i=1}^{4} \mathbf{B}_i^{k+1} + \mathbf{Y}_i^k \ , \\ \mathbf{Y}_i^{k+1} &= \mathbf{Y}_i^k + \mathbf{B}_i^{k+1} - \mathbf{Z}^{k+1} \ . \end{aligned} \tag{ADMM}$$

We note that the $\mathbf{B}$-updates in this scheme are in fact proximal operators. Indeed, we have

$$\operatorname*{argmin}_{\mathbf{B}_i} \left(f_i(\mathbf{B}_i) + \frac{\rho}{2}\|\mathbf{B}_i - \mathbf{Z}^k + \mathbf{Y}_i^k\|_F^2 \right) = P_{f_i, 1/\rho}(\mathbf{Z}^k - \mathbf{Y}_i^k) \ .$$

We have already seen that for $i = 1$ this operator can be computed via an SVD decomposition. In addition, since f_2 is smooth, operator $P_{f_2,1/\rho}$ can be computed analytically. Furthermore, since f_3 and f_4 are indicator functions, in this case the proximal operator reduces to an orthogonal projection onto the corresponding convex set. Hence, we get an easy to implement iterative method for solving (CO-P) and obtaining a proper PNFA via a spectral-like method. Convergence results for this optimization scheme and guidelines on how to choose ρ can be found in [BPCPE11].

Chapter 9

Learning Weighted Automata via Matrix Completion

In this chapter we use the spectral method to tackle the problem of learning general WFA from labeled examples. This means that we are given a sample of string-label pairs $z = (x, y)$, where $x \in \Sigma^\star$ is a string and $y \in \mathbb{R}$ is a real label, and are asked to produce a WFA A that is a good model of this sample, in the sense that $f_A(x) \approx y$ for all the examples in the sample. This is a typical problem in supervised learning known as *regression*. The special difficulty here is that strings are a type of data for which it is not obvious how to find a finite-dimensional vector representation. Thus, typical regression methods for real vectors spaces cannot be applied directly. Though a *kernel method* could be used in this case, we choose to take an alternative approach by giving an algorithm for finding a hypothesis in the class of WFA directly. Our algorithm is based on combining the spectral method with a matrix completion algorithm applied to Hankel matrices.

Since the assumptions in the learning framework used in this chapter are different from previous PAC analyses, we begin in Section 9.1 by describing in detail the learning framework we will be using. We also give an idea of the type of results we will be able to prove for our algorithm. Then, in Section 9.2 we describe a family of algorithms for learning WFA from labeled samples. We analyze one particular algorithm from this family in Section 9.3, and give a generalization bound based on algorithmic stability.

9.1 Agnostic Learning of Weighted Automata

The main assumption in all PAC learning results we have given so far is that the data given to the learning algorithm was generated by a model from some class which is known a priori. We then asked the algorithm to choose a hypothesis from a class containing this unknown model. Though very convenient from a theoretical point of view, this learning framework might sound a bit unrealistic from the perspective of a practitioner who needs an accurate model for her data but is not willing to assume that such data was generated from some particular model. In this chapter we shall take a shift of paradigm by moving away from the realizability assumption of PAC learning and into the agnostic setting that characterizes most of the literature in *statistical learning*. We now introduce the learning framework that we will be using in this chapter. Then we will discuss what kind of guarantees one might ask of a learning algorithm in this framework.

Let D be a probability distribution over $\Sigma^\star \times \mathbb{R}$. That is, if we draw an example from D we get a pair $z = (x, y)$, where x is a finite string over $\Sigma^\star$ and y is a real label assigned to x. The case where x is generated from some distribution D' over $\Sigma^\star$ and $y = f(x)$ for some real function $f : \Sigma^\star \to \mathbb{R}$ is included in this setting as a particular case. Note however that the above setting also includes more general distributions. For example, the agnostic framework can also model the noisy labeling setting where $y = f(x) + N$ for some real random variable N that models the presence of random noise in the labelling function.

Learning a WFA in this setting means the following. Suppose we are given a sample $S = (z^1, \ldots, z^m)$ of i.i.d. examples drawn from some distribution D over $\Sigma^\star \times \mathbb{R}$. Then, given some number of states n, we want to take S as input and produce a WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ with at most n states such that f_A is, hopefully, a "good model for D." The *generalization error* (or *risk*) of f_A is the usual way of assessing how good the model A is. In particular, let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ be a *loss function* that quantifies the error $\ell(y, y')$ of predicting label y when y' is the correct one. Then, the generalization error of a function $f : \Sigma^\star \to \mathbb{R}$ with respect to D is defined as

$$R_D(f) = \mathbb{E}_{(x,y)\sim D}[\ell(f(x), y)] \ .$$

That is, $R_D(f)$ is the expected error, measured by ℓ, of using f to predicting the labels generated by D. When D is clear from the context we might just write $R(f)$. Another tightly related quantity is the *empirical error* (or *empirical risk*) incurred by f on the sample S, defined as:

$$\hat{R}_S(f) = \hat{\mathbb{E}}_S[\ell(f(x), y)] = \frac{1}{m}\sum_{i=1}^{m} \ell(f(x^i), y^i) \ .$$

That is, the empirical estimate of $R(f)$ based on the sample S from D.

The most natural thing to ask from an algorithm in this setting is, perhaps, that it returns a hypothesis that performs almost as well as the best one in a given class. In particular, let $\mathcal{WFA}_n$ be the class of all WFA our algorithm can produce as output, e.g. WFA with n or less states. Define A_{OPT} to be the best hypothesis in $\mathcal{WFA}_n$ in terms of generalization error:

$$A_{\mathrm{OPT}} = \operatorname*{argmin}_{A \in \mathcal{WFA}_n} R(f_A) \ .$$

Now, denote by A_S the output of a learning algorithm on input a sample S of size m from D. Suppose there exists some $\varepsilon = \varepsilon(m) = o(1)$ such that with high probability over the choice of S it holds that

$$R(f_{A_S}) \leq R(f_{A_{\mathrm{OPT}}}) + \varepsilon \ .$$

This means that as the number of examples given to the algorithm grows, its output converges to the best hypothesis in the class. In the particular case where $\varepsilon = O(1/m^c)$ for some $c > 0$ and A_S can be computed in time $\mathsf{poly}(m)$, one says that the algorithm is an *efficient agnostic PAC learner* for D [Hau92; KSS94]. It is well known that the statistical requirements of agnostic PAC learning can be achieved for many hypothesis classes via *empirical risk minimization* [Vap99]. However, in general this involves non-tractable optimization problems, and there are very few classes of functions known to be efficiently agnostic PAC learnable; see [Maa94; LBW96; GKS01; KKMS05; KMV08] and references therein. In the particular case of WFA, the hardness results in [PW93] stating that it is NP-hard to approximate the minimal DFA consistent with a given sample stand in the way of any efficient algorithm for performing empirical risk minimization for this class of hypothesis.

Nonetheless, there is a simpler and more basic property that one can ask for in algorithms in the agnostic setting. This is that the hypothesis it produces has good generalization performance, or in other words, that the algorithm does not suffer from overfitting. In cases where the hypothesis class is very large, overfitting means that an algorithm produces a hypothesis that models the available data very accurately but fails to capture the behavior of future examples drawn from the same distribution as the training sample. In particular, if f is a hypothesis learned from some sample S, its *generalization error* with respect to D is defined as the difference $R(f) - \hat{R}_S(f)$. Note that this quantity is large whenever f has small empirical error on S but large expected error on examples freshly drawn from D. Thus, we can say that an algorithm avoids overfitting if there exists some $\varepsilon = \varepsilon(m) = o(1)$ such that with high probability over the choice of S one has

$$R(f_S) \leq \hat{R}_S(f_S) + \varepsilon \ ,$$

where f_S is the hypothesis produced by the algorithm on input S. A bound of this form is known as a *generalization bound*, and it states that, for large samples, the predictive behavior of f_S on S is a good approximation to the predictive behavior of f_S on future examples.

Some standard ways of proving generalization bounds rely on measuring the complexity of the hypothesis class used by the learning algorithm. Such measures include the Vapnik–Chervonenkis dimension, or the Rademacher complexity [VC71; Kol01]. Interestingly, such methods yield generalization bounds that hold uniformly over the whole class of hypothesis and hold independently of the particular learning algorithm being used. Unfortunately, there are hypothesis classes for which these bounds become vacuous, or for which we do not know how to compute the complexity of the hypothesis class. In these cases – WFA being one of them – generalization bounds can still be obtained by alternative methods. In particular, we will use here the *stability method* [KR99], which can be used to analyze the generalization error of hypotheses produced by a fixed learning algorithm.

In summary, in the following sections we will present and analyze an algorithm for learning general WFA from labeled examples in the agnostic setting. The analysis that we give proves a generalization bound for the hypotheses produced by our algorithm. This means the algorithm does *not* overfit the training data.

9.2 Completing Hankel Matrices via Convex Optimization

The main obstruction in leveraging the spectral method for learning general weighted automata from labeled examples is that of the missing entries in the Hankel matrix. When learning stochastic WFA, the statistics used by the spectral method are essentially the probabilities assigned by the target distribution to each string in $\mathcal{P}\mathcal{S}$. As we have already seen, increasingly large samples yield uniformly convergent estimates for these probabilities. Thus, it can be safely assumed that the probability of any string from $\mathcal{P}\mathcal{S}$ not present in the sample is zero. When learning arbitrary WFA, however, the value assigned by the target to an unseen string is unknown. Furthermore, one cannot expect that a sample will contain the values of the target function for all the strings in $\mathcal{P}\mathcal{S}$. This simple observation raises the question of whether it is possible at all to apply the spectral method in a setting with missing data, or, alternatively, whether there is a principled way to "estimate" this missing information and then apply the spectral method.

It turns out that the second of these approaches can be naturally formulated as a *constrained matrix completion* problem. When applying the spectral method, the (approximate) values of the target on $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ are arranged in the matrix $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$. Thus, the main difference between the stochastic and non-stochastic setting can be restated as follows: when learning stochastic WFA, unknown entries of $\mathbf{H}$ can be filled in with zeros, while in the general setting there is a priori no straightforward method to fill in the missing values. In this section we describe a matrix completion algorithm for solving this last problem. In particular, since $\mathbf{H}$ is a Hankel matrix whose entries must satisfy some equality constraints, it turns out that the problem of learning general WFA from labeled examples leads to what we call the *Hankel matrix completion* problem. This is essentially a matrix completion problem where entries of valid hypotheses need to satisfy the set of equalities imposed by the basis $\mathcal{B}$. We will show how the problem of Hankel matrix completion can be solved with a convex optimization algorithm. In fact, many existing approaches to matrix completion are also based on convex optimization algorithms [CT10; CP10; Rec11; FSSS11]. Furthermore, since the set of valid hypotheses for our constrained matrix completion problem is convex, many of these algorithms could also be modified to solve the Hankel matrix completion problem. We choose to propose our own algorithm here because it is the one we will use on the analysis given in Section 9.3.

We begin by introducing some notation. Throughout this chapter we shall assume that $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is a p-closed basis with $\lambda \in \mathcal{P} \cup \mathcal{S}$. For any basis $\mathcal{B}$, we denote by $\mathbb{H}_{\mathcal{B}}$ the vector space of functions $\mathbb{R}^{\mathcal{P}\mathcal{S}}$ whose dimension is the dimension of $\mathcal{B}$, defined as $|\mathcal{P}\mathcal{S}|$. We will simply write $\mathbb{H}$ instead of $\mathbb{H}_{\mathcal{B}}$ when the basis $\mathcal{B}$ is clear from the context. The Hankel matrix $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ associated to a function $h \in \mathbb{H}$ is the matrix whose entries are defined by $\mathbf{H}(u, v) = h(uv)$ for all $u \in \mathcal{P}$ and $v \in \mathcal{S}$. Note that the mapping $h \mapsto \mathbf{H}$ is linear. In fact, $\mathbb{H}$ is isomorphic to the vector space formed by all $|\mathcal{P}| \times |\mathcal{S}|$ real *Hankel* matrices and we can thus write by identification that $\mathbb{H}$ contains all $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ such that

$$\forall u_1, u_2 \in \mathcal{P},\ \forall v_1, v_2 \in \mathcal{S}, \quad u_1 v_1 = u_2 v_2 \ \Rightarrow\ \mathbf{H}(u_1, v_1) = \mathbf{H}(u_2, v_2) \ .$$

It is clear from this characterization that $\mathbb{H}$ is a *convex* set because it is a subset of a convex space defined

by equality constraints. In particular, a matrix in $\mathbb{H}$ contains $|\mathcal{P}||\mathcal{S}|$ coefficients with $|\mathcal{P}\mathcal{S}|$ degrees of freedom, and the dependencies can be specified as a set of equalities of the form $\mathbf{H}(u_1, v_1) = \mathbf{H}(u_2, v_2)$ when $u_1 v_1 = u_2 v_2$. We will use both characterizations of $\mathbb{H}$ indistinctly from now on. Also, note that different orderings of $\mathcal{P}$ and $\mathcal{S}$ may result in different sets of matrices. For convenience, we will assume for all that follows an arbitrary fixed ordering, since the choice of that order has no effect on our results.

Matrix norms extend naturally to norms in $\mathbb{H}$. For any $1 \leq p \leq \infty$, the *Hankel–Schatten p-norm* on $\mathbb{H}$ is defined as $\|h\|_p = \|\mathbf{H}\|_p$. It is straightforward to verify that $\|h\|_p$ is a norm by the linearity of $h \mapsto \mathbf{H}$. In particular, this implies that the function $\|\cdot\|_p \colon \mathbb{H} \to \mathbb{R}$ is convex. In the case $p = 2$, it can be seen that $\|h\|_2^2 = \langle h, h \rangle_{\mathbb{H}}$, with the inner product on $\mathbb{H}$ defined by

$$\langle h, h' \rangle_{\mathbb{H}} = \sum_{x \in \mathcal{P}\mathcal{S}} c_x h(x) h'(x) \ ,$$

where $c_x = |\{(u, v) \in \mathcal{P} \times \mathcal{S} \colon x = uv\}|$ is the number of possible decompositions of x into a prefix in $\mathcal{P}$ and a suffix in $\mathcal{S}$.

We now outline our algorithm `HMC+SM` for learning general WFA from labeled examples based on combining matrix completion with the spectral method. As input, the algorithm takes a sample $S = (z^1, \ldots, z^m)$ containing m examples $z^i = (x^i, y^i) \in \Sigma^\star \times \mathbb{R}$, $1 \leq i \leq m$, drawn i.i.d. from some distribution D over $\Sigma^\star \times \mathbb{R}$. There are three parameters a user can specify to control the behavior of the algorithm: a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ of $\Sigma^\star$, a regularization parameter $\tau > 0$, and the desired number of states n in the hypothesis. The output returned by `HMC+SM` is a WFA A_S with n states that computes a function $f_{A_S} \colon \Sigma^\star \to \mathbb{R}$. The algorithm works in two stages. In the first stage, a constrained matrix completion algorithm with input S and regularization parameter τ is used to compute a Hankel matrix $\mathbf{H}_S \in \mathbb{H}_{\mathcal{B}}$. In the second stage, the spectral method is applied to $\mathbf{H}_S$ to compute a WFA A_S with n states. The spectral method is just a direct application of the equations given in Section 5.3 by observing that we can write $\mathbf{H}_S^\top = [\mathbf{H}_\lambda^\top, \mathbf{H}_{\sigma_1}^\top, \ldots, \mathbf{H}_{\sigma_{|\Sigma|}}^\top]$, with $\mathbf{h}_{\lambda,\mathcal{S}}$ and $\mathbf{h}_{\mathcal{P},\lambda}$ contained in $\mathbf{H}_\lambda$ because we assumed $\lambda \in \mathcal{P} \cup \mathcal{S}$. In the rest of this section we describe the Hankel matrix completion algorithm in detail.

It is easy to see that in fact `HMC+SM` defines a whole family of algorithms. In particular, by combining the spectral method with any algorithm for solving the Hankel matrix completion problem, one can derive a new algorithm for learning WFA. We shall now describe a particular family of Hankel matrix completion algorithms. This family will be parametrized by a number $1 \leq p \leq \infty$ and a convex loss $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$.

Given a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ of $\Sigma^\star$ and a sample S over $\Sigma^\star \times \mathbb{R}$, our algorithm solves a convex optimization problem and returns a matrix $\mathbf{H}_S \in \mathbb{H}_{\mathcal{B}}$. We give two equivalent descriptions of this optimization, one in terms of functions $h \colon \mathcal{P}\mathcal{S} \to \mathbb{R}$, and another in terms of Hankel matrices $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$. While the former is perhaps conceptually simpler, the latter is easier to implement within the existing frameworks of convex optimization.

We will denote by $\tilde{S}$ the subsample of S containing all the examples $z = (x, y) \in S$ such that $x \in \mathcal{P}\mathcal{S}$. We write $\tilde{m}$ to denote the size of $\tilde{S}$. For any $1 \leq p \leq \infty$ and any convex loss function $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$, we define the objective function F_S over $\mathbb{H}$ as follows:

$$F_S(h) = \tau N(h) + \hat{R}_{\tilde{S}}(h) = \tau \|h\|_p^2 + \frac{1}{\tilde{m}} \sum_{(x,y) \in \tilde{S}} \ell(h(x), y) \ ,$$

where $\tau > 0$ is a regularization parameter. F_S is clearly a convex function by the convexity of $\|\cdot\|_p$ and ℓ. Our algorithm seeks to minimize this loss function over the finite-dimensional vector space $\mathbb{H}$ and returns a function h_S satisfying

$$h_S \in \operatorname*{argmin}_{h \in \mathbb{H}} F_S(h) \ . \tag{HMC-h}$$

To define an equivalent optimization over the matrix version of $\mathbb{H}$, we introduce the following notation. For each string $x \in \mathcal{P}\mathcal{S}$, fix a pair of coordinate vectors $(\mathbf{u}_x, \mathbf{v}_x) \in \mathbb{R}^{\mathcal{P}} \times \mathbb{R}^{\mathcal{S}}$ such that $\mathbf{u}_x^\top \mathbf{H} \mathbf{v}_x = \mathbf{H}(x)$ for any $\mathbf{H} \in \mathbb{H}$. That is, $\mathbf{u}_x$ and $\mathbf{v}_x$ are coordinate vectors corresponding respectively to a prefix $u \in \mathcal{P}$ and a suffix $v \in \mathcal{S}$, and such that $uv = x$. Now, abusing our previous notation, we define the following

loss function over matrices:

$$F_S(\mathbf{H}) = \tau N(\mathbf{H}) + \hat{R}_{\tilde{S}}(\mathbf{H}) = \tau \|\mathbf{H}\|_p^2 + \frac{1}{\tilde{m}} \sum_{(x,y)\in\tilde{S}} \ell(\mathbf{u}_x^\top \mathbf{H} \mathbf{v}_x, y) \ .$$

This is a convex function defined over the space of all $|\mathcal{P}| \times |\mathcal{S}|$ matrices. Optimizing F_S over the convex set of Hankel matrices $\mathbb{H}$ leads to an algorithm equivalent to (HMC-h):

$$\mathbf{H}_S \in \operatorname*{argmin}_{\mathbf{H}\in\mathbb{H}} F_S(H) \ . \tag{HMC-H}$$

We note here that our approach shares some common aspects with some previous work in matrix completion. The fact that there may not be a true underlying Hankel matrix makes it somewhat close to the agnostic setting in [FSSS11], where matrix completion is also applied under arbitrary distributions. Nonetheless, it is also possible to consider other learning frameworks for WFA where algorithms for exact matrix completion [CT10; Rec11] or noisy matrix completion [CP10] may be useful. Furthermore, since most algorithms in the literature of matrix completion are based on convex optimization problems, it is likely that most of them can be adapted to solve constrained matrix completions problems such as the one we discuss here.

9.3 Generalization Bound

In this section, we study the generalization properties of $\texttt{HMC}_{p,\ell}\texttt{+SM}$. We give a stability analysis for a special instance of this family of algorithms and use it to derive a generalization bound. We study the specific case where $p = 2$ and $\ell(y, y') = |y - y'|$ for all (y, y'). We note that in this case the term $N(h)$ in the loss function $F_S(h)$ is differentiable and can be expressed as an inner product. Though this property is important in an early step of our analysis, we believe that similar approaches can be used to prove generalization bounds for other algorithms in this class.

We first introduce some notation needed for the presentation of our main result. For any $\nu > 0$, let t_ν be the function defined by $t_\nu(x) = x$ for $|x| \leq \nu$ and $t_\nu(x) = \nu \operatorname{sign}(x)$ for $|x| > \nu$. For any distribution D over $\Sigma^\star \times \mathbb{R}$, we denote by D_Σ its marginal distribution over $\Sigma^\star$. The probability that a string $x \sim D_\Sigma$ belongs to $\mathcal{PS}$ is denoted by $\pi = D_\Sigma(\mathcal{PS})$.

We assume that the parameters $\mathcal{B}$, n, and τ are fixed. Two parameters that depend on D will appear in our bound. In order to define these parameters, we need to consider the output $\mathbf{H}_S$ of (HMC-H) as a random variable that depends on the sample S. Writing $\mathbf{H}_S^\top = [\mathbf{H}_\lambda^\top, \mathbf{H}_{\sigma_1}^\top, \ldots, \mathbf{H}_{\sigma_{|\Sigma|}}^\top]$ we define:

$$\mathfrak{s} = \mathbb{E}_{S\sim D^m}\left[\mathfrak{s}_n(\mathbf{H}_\lambda)\right] \ ,$$
$$\rho = \mathbb{E}_{S\sim D^m}\left[\mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2\right] \ ,$$

where $\mathfrak{s}_n(\mathbf{M})$ denotes the nth singular value of matrix $\mathbf{M}$. Note that these parameters may vary with m, n, τ and $\mathcal{B}$.

In contrast to previous learning results based on the spectral method, our bound holds in an agnostic setting. That is, we do not require that the data was generated from some (probabilistic) unknown WFA. However, in order to prove our results we do need to make two assumptions about the tails of the distribution. First, we need to assume that there exists a bound on the magnitude of the labels generated by the distribution.

Assumption 4. *There exists a constant $\nu > 0$ such that if $(x, y) \sim D$, then $|y| \leq \nu$ almost surely.*

Second, we assume that the strings generated by the distribution will not be too long. In particular, that the length of the strings generated by D_Σ follows a distribution whose tail is slightly lighter than sub-exponential.

Assumption 5. *There exist constants $c, \eta > 0$ such that the following holds for all $t \geq 0$: $\mathbb{P}_{x\sim D_\Sigma}[|x| \geq t] \leq \exp(-ct^{1+\eta})$.*

We note that in the present context both assumptions are quite reasonable. Assumption 4 is equivalent to assumptions made in other contexts where a stability analysis is pursued, e.g., in the analysis of support vector regression in [BE02]. Furthermore, in our context, this assumption can be relaxed to require only that the distribution over labels be sub-Gaussian, at the expense of a more complex proof.

Assumption 5 is required by the fact already pointed out in [HKZ09] that errors in the estimation of operator models accumulate exponentially with the length of the string; this can be readily observed in the bounds given in Section 5.4. Moreover, it is well known that the tail of any probability distribution generated by a WFA is sub-exponential, see [DE08]. Thus, though we do not require D_Σ to be generated by a WFA, we do need its distribution over lengths to have a tail behavior similar to that of a distribution generated by a WFA.

We can now state our main result, which is a bound on the risk $R(f) = \mathbb{E}_{z\sim D}[\ell(f(x), y)]$ in terms of the empirical risk $\hat{R}_S(f) = \sum_{z\in S} \ell(f(x), y)/m$. Let us define the quantities $\tilde{\nu} = \max\{\nu^2, \nu^{-1}\}$, $\tilde{\tau} = \min\{\tau^2, \tau^{-1}\}$, $\tilde{\rho} = \min\{1, \rho^2\}$, and $\tilde{\mathfrak{s}} = \min\{\mathfrak{s}^2, \mathfrak{s}^{-2}\}$.

Theorem 9.3.1. *Let S be a sample of m i.i.d. examples drawn from some distribution D satisfying Assumptions 4 and 5. Let A_S be the WFA returned by algorithm $\texttt{HMC}_{p,\ell}\texttt{+SM}$ with $p = 2$ and loss function $\ell(y, y') = |y - y'|$. There exist universal constants C, C' such that for any $\delta > 0$, if*

$$m \geq C \max\left\{ \frac{\tilde{\nu}|\mathcal{P}||\mathcal{S}|}{\tilde{\tau}\tilde{\rho}\tilde{\mathfrak{s}}\pi^2}, \exp\left(\left(\frac{6c}{5}\right)^{1/\eta}\left(\ln\frac{2\nu\sqrt{|\mathcal{P}||\mathcal{S}|}}{\mathfrak{s}}\right)^{1+1/\eta}\right)\right\} \ ,$$

then the following holds with probability at least $1 - \delta$ for $f_S = t_\nu \circ f_{A_S}$:

$$R(f_S) \leq \hat{R}_S(f_S) + C' \frac{\nu^4|\mathcal{P}|^2|\mathcal{S}|^{3/2}}{\tau\rho\mathfrak{s}^3\pi} \frac{\ln m}{m^{1/3}} \sqrt{\ln\frac{1}{\delta}} \ .$$

The proof of this theorem is based on an algorithmic stability analysis. Thus, we will consider two samples of size m: $S \sim D^m$ consisting of i.i.d. examples drawn from D, and S' differing from S by just one point. Thus, we shall write $S = (z^1, \ldots, z^m)$ and $S' = (z^1, \ldots, z^{m-1}, z'^m)$, where the new example z'^m is an arbitrary point the support of D. Throughout the analysis we use the shorter notation $\mathbf{H} = \mathbf{H}_S$ and $\mathbf{H}' = \mathbf{H}_{S'}$ for the Hankel matrices obtained from (HMC-H) based on samples S and S' respectively.

The first step in the analysis is to bound the stability of the matrix completion algorithm. This is done in the following lemmas, that give a sample-dependent and a sample-independent bound for the stability of $\mathbf{H}$. We begin with a technical lemma that is useful for studying the algorithmic stability of Hankel matrix completion. Symbols $h = h_S$ and $h' = h_{S'}$ will denote the functions in $\mathbb{H}$ obtained by solving (HMC-h) with training samples S and S' respectively. The proof uses an argument similar to the one in [MRT12] for bounding the stability of kernel ridge regression. See Appendix A.4 for a brief review of Bregman divergences.

Lemma 9.3.2. *The following inequality holds for all samples S and S' differing by only one point:*

$$2\tau\|h - h'\|_2^2 \leq \hat{R}_{\tilde{S}}(h') - \hat{R}_{\tilde{S}}(h) + \hat{R}_{\tilde{S}'}(h) - \hat{R}_{\tilde{S}'}(h') \ .$$

Proof. Recall that the Bregman divergence is non-negative, and that by an adequate choice of subgradients for R_S and $R_{S'}$ we have $B_{F_S} = \tau B_N + B_{\hat{R}_{\tilde{S}}}$ and $B_{F_{S'}} = \tau B_N + B_{\hat{R}_{\tilde{S}'}}$. Thus, we have the following inequality:

$$\tau B_N(h'\|h) + \tau B_N(h\|h') \leq B_{F_S}(h'\|h) + B_{F_{S'}}(h\|h') \ .$$

Furthermore, note that by the optimality of h and h' with respect to F_S and $F_{S'}$ we can choose subgradients in B_{F_S} and $B_{F_{S'}}$ such that $\delta F_S(h) = \delta F_{S'}(h') = 0$. Hence, from the definition of Bregman divergence with these choices we get

$$\begin{aligned} B_{F_S}(h'\|h) + B_{F_{S'}}(h\|h') &= F_S(h') - F_S(h) + F_{S'}(h) - F_{S'}(h') \\ &= \hat{R}_{\tilde{S}}(h') - \hat{R}_{\tilde{S}}(h) + \hat{R}_{\tilde{S}'}(h) - \hat{R}_{\tilde{S}'}(h') \ . \end{aligned}$$

Now, if we write h_c for the function $x \mapsto c_x h(x)$, then it is immediate to see that $\nabla N(h) = \nabla \langle h, h \rangle_{\mathbb{H}} = 2h_c$. Thus, we have the following:

$$B_N(h'\|h) + B_N(h\|h') = -\langle h' - h, 2h_c \rangle_{\mathbb{H}} - \langle h - h', 2h'_c \rangle_{\mathbb{H}} = 2\langle h - h', h_c - h'_c \rangle_{\mathbb{H}} \ .$$

Finally, since $c_x \geq 1$ for all $x \in \mathcal{PS}$, we have that

$$\begin{aligned}
\|h - h'\|_2^2 &= \langle h - h', h - h' \rangle_{\mathbb{H}} \\
&= \sum_{x \in \mathcal{PS}} c_x (h(x) - h'(x))^2 \\
&\leq \sum_{x \in \mathcal{PS}} c_x^2 (h(x) - h'(x))^2 \\
&= \sum_{x \in \mathcal{PS}} c_x (h(x) - h'(x))(h_c(x) - h'_c(x)) \\
&= \langle h - h', h_c - h'_c \rangle_{\mathbb{H}} \ .
\end{aligned}$$

□

Now we bound the stability of the optimization algorithm (HMC-H) using Lemma 9.3.2.

Lemma 9.3.3. *Assume that D satisfies Assumption 4. Then, the following holds:*

$$\|\mathbf{H} - \mathbf{H}'\|_F \leq \min \left\{ 2\nu\sqrt{|\mathcal{P}||\mathcal{S}|}, \frac{1}{\tau \min\{\tilde{m}, \tilde{m}'\}} \right\} \ .$$

Proof. Note that by Assumption 4, for all (x, y) in $\tilde{S}$, or $\tilde{S}'$, we have $|y| \leq \nu$. Therefore, we must have $|\mathbf{H}(u, v)| \leq \nu$ for all $u \in \mathcal{P}$ and $v \in \mathcal{S}$, otherwise the value of $F_S(\mathbf{H})$ is not minimal because decreasing the absolute value of an entry $|\mathbf{H}(u, v)| > \nu$ decreases the value of $F_S(\mathbf{H})$. The same holds for $\mathbf{H}'$. Thus, the first bound follows from $\|\mathbf{H} - \mathbf{H}'\|_F \leq \|\mathbf{H}\|_F + \|\mathbf{H}'\|_F \leq 2\nu\sqrt{|\mathcal{P}||\mathcal{S}|}$. Now we proceed to show the second bound. Since by definition we have $\|\mathbf{H} - \mathbf{H}'\|_F = \|h - h'\|_2$, it is sufficient to bound this second quantity. By Lemma 9.3.2, we have

$$2\tau\|h - h'\|_2^2 \leq \hat{R}_{\tilde{S}}(h') - \hat{R}_{\tilde{S}}(h) + \hat{R}_{\tilde{S}'}(h) - \hat{R}_{\tilde{S}'}(h') \ . \tag{9.1}$$

We can consider four different situations for the right-hand side of this expression, depending on the membership of x^m and x'^m in the set $\mathcal{PS}$. If $x^m, x'^m \notin \mathcal{PS}$, then $\tilde{S} = \tilde{S}'$. Therefore, $\hat{R}_{\tilde{S}}(h) = \hat{R}_{\tilde{S}'}(h)$, $\hat{R}_{\tilde{S}}(h') = \hat{R}_{\tilde{S}'}(h')$, and $\|h - h'\|_2 = 0$. If $x^m, x'^m \in \mathcal{PS}$, then $\tilde{m} = \tilde{m}'$, and the following equalities hold:

$$\begin{aligned}
\hat{R}_{\tilde{S}'}(h) - \hat{R}_{\tilde{S}}(h) &= \frac{|h(x'^m) - y'^m| - |h(x^m) - y^m|}{\tilde{m}} \ , \\
\hat{R}_{\tilde{S}}(h') - \hat{R}_{\tilde{S}'}(h') &= \frac{|h'(x^m) - y^m| - |h'(x'^m) - y'^m|}{\tilde{m}} \ .
\end{aligned}$$

Thus, in view of (9.1), we can write

$$2\tau\|h - h'\|_2^2 \leq \frac{|h(x^m) - h'(x^m)| + |h(x'^m) - h'(x'^m)|}{\tilde{m}} \leq \frac{2}{\tilde{m}}\|h - h'\|_2 \ ,$$

where we used the inequalities

$$|\,|h(x) - y| - |h'(x) - y|\,| \leq |h(x) - h'(x)| \leq \|h - h'\|_2 \ .$$

If $x^m \in \mathcal{PS}$ and $x'^m \notin \mathcal{PS}$, the right-hand side of (9.1) equals

$$\begin{aligned}
&\sum_{z \in \tilde{S}'} \left(\frac{|h'(x) - y|}{\tilde{m}} - \frac{|h'(x) - y|}{\tilde{m}'} + \frac{|h(x) - y|}{\tilde{m}'} - \frac{|h(x) - y|}{\tilde{m}} \right) \\
&\quad + \frac{|h'(x^m) - y^m|}{\tilde{m}} - \frac{|h(x^m) - y^m|}{\tilde{m}} \ .
\end{aligned}$$

Now, since $\tilde{m} = \tilde{m}' + 1$ we can write

$$2\tau\|h - h'\|_2^2 \leq \sum_{z \in \tilde{S}'} \frac{|h(x) - h'(x)|}{\tilde{m}\,\tilde{m}'} + \frac{|h(x^m) - h'(x^m)|}{\tilde{m}} \leq \frac{2}{\tilde{m}}\|h - h'\|_2 \ .$$

By symmetry, a similar bound holds in the case where $x^m \notin \mathcal{PS}$ and $x'^m \in \mathcal{PS}$. Combining these four bounds yields the desired inequality. □

The standard method for deriving generalization bounds from algorithmic stability results could be applied here to obtain a generalization bound for our Hankel matrix completion algorithm. However, our goal is to give a generalization bound for the full `HMC+SM` algorithm. To bound the stability of `HMC+SM` we will need to study the point-wise stability $|f_{A_S}(x) - f_{A_{S'}}(x)|$ in our agnostic setting. The next lemmas give the main technical tools we will need to bound this difference by using results from Section 5.4.1.

Lemma 9.3.4. *Let $\gamma = \nu\sqrt{|\mathcal{P}||\mathcal{S}|}/\mathfrak{s}_n(\mathbf{H}_\lambda)$. The weighted automaton A_S satisfies: $\|\boldsymbol{\alpha}_0\|, \|\boldsymbol{\alpha}_\infty\|, \|\mathbf{A}_\sigma\| \leq \gamma$.*

Proof. Since $\|\mathbf{H}_\sigma\| \leq \nu\sqrt{|\mathcal{P}|\ |\mathcal{S}|}$, simple calculations show $\|\boldsymbol{\alpha}_0^\top\| \leq \nu\sqrt{|\mathcal{S}|}$, $\|\boldsymbol{\alpha}_\infty\| \leq \nu\sqrt{|\mathcal{P}|}/\mathfrak{s}_n(\mathbf{H}_\lambda)$, and $\|\mathbf{A}_\sigma\| \leq \nu\sqrt{|\mathcal{P}||\mathcal{S}|}/\mathfrak{s}_n(\mathbf{H}_\lambda)$. □

Now we give an application of Lemma 5.4.5 to our stability problem. Using the bound on the Frobenius norm $\|\mathbf{H} - \mathbf{H}'\|_F$, we can analyze the stability of $\mathfrak{s}_n(\mathbf{H}_\lambda)$, $\mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2$, and $\mathbb{V}_n$ using well-known results on the stability of singular values and singular vectors. These results are used to bound the difference between the operators of WFA A_S and $A_{S'}$ in terms of the following quantities:

$$\begin{aligned}
\varepsilon &= \|\mathbf{H} - \mathbf{H}'\|_F \ , \\
\hat{\mathfrak{s}} &= \min\{\mathfrak{s}_n(\mathbf{H}_\lambda), \mathfrak{s}_n(\mathbf{H}'_\lambda)\} \ , \\
\hat{\rho} &= \mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2 \ .
\end{aligned}$$

Lemma 9.3.5. *Suppose $\varepsilon \leq \sqrt{\hat{\rho}}/4$. There exists a universal constant $c_1 > 0$ such that the following inequalities hold for all $\sigma \in \Sigma$:*

$$\begin{aligned}
\|\mathbf{A}_\sigma - \mathbf{A}'_\sigma\| &\leq c_1 \frac{\varepsilon\nu^3|\mathcal{P}|^{3/2}|\mathcal{S}|^{1/2}}{\hat{\rho}\hat{\mathfrak{s}}^2} \ , \\
\|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0\| &\leq c_1 \frac{\varepsilon\nu^2|\mathcal{P}|^{1/2}|\mathcal{S}|}{\hat{\rho}} \ , \\
\|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\| &\leq c_1 \frac{\varepsilon\nu^3|\mathcal{P}|^{3/2}|\mathcal{S}|^{1/2}}{\hat{\rho}\hat{\mathfrak{s}}^2} \ .
\end{aligned}$$

Proof. We begin with a few observations that will help us apply Lemma 5.4.5. First note that by definition of ε the following hold for every $\sigma \in \Sigma'$:

$$\begin{aligned}
\|\mathbf{H}_\sigma - \mathbf{H}'_\sigma\| &\leq \varepsilon \ , \\
\|\mathbf{h}_{\mathcal{P},\lambda} - \mathbf{h}'_{\mathcal{P},\lambda}\| &\leq \varepsilon \ , \\
\|\mathbf{h}_{\lambda,\mathcal{S}} - \mathbf{h}'_{\lambda,\mathcal{S}}\| &\leq \varepsilon \ .
\end{aligned}$$

Furthermore, $\|\mathbf{H}_\sigma\| \leq \|\mathbf{H}_\sigma\|_F \leq \nu\sqrt{|\mathcal{P}||\mathcal{S}|}$ and $\|\mathbf{H}'_\sigma\| \leq \nu\sqrt{|\mathcal{P}||\mathcal{S}|}$ for all $\sigma \in \Sigma'$. In addition, we have $\|\mathbf{h}_{\lambda,\mathcal{S}}\| \leq \nu\sqrt{|\mathcal{S}|}$ and $\|\mathbf{h}'_{\mathcal{P},\lambda}\| \leq \nu\sqrt{|\mathcal{P}|}$. Finally, by construction we also have $\mathfrak{s}_n(\mathbf{H}_\lambda\mathbf{V}) = \mathfrak{s}_n(\mathbf{H}_\lambda)$ and $\mathfrak{s}_n(\mathbf{H}'_\lambda\mathbf{V}') = \mathfrak{s}_n(\mathbf{H}'_\lambda)$. Therefore, it only remains to bound $\|\mathbf{V} - \mathbf{V}'\|$, which by Corollary A.2.5 is

$$\|\mathbf{V} - \mathbf{V}'\| \leq \frac{4\varepsilon}{\hat{\rho}}\left(2\nu\sqrt{|\mathcal{P}||\mathcal{S}|} + \varepsilon\right) \leq \frac{16\varepsilon\nu\sqrt{|\mathcal{P}||\mathcal{S}|}}{\hat{\rho}} \ ,$$

where the last inequality follows from Lemma 9.3.3. Plugging all the bounds above in Lemma 5.4.5 yields the following inequalities:

$$\|\mathbf{A}_\sigma - \mathbf{A}'_\sigma\| \leq \frac{\varepsilon}{\hat{\mathfrak{s}}}\left(1 + \frac{16\nu|\mathcal{P}|^{1/2}|\mathcal{S}|^{1/2}}{\hat{\rho}}\right) + \frac{1+\sqrt{5}}{2}\frac{\varepsilon\nu|\mathcal{P}|^{1/2}|\mathcal{S}|^{1/2}}{\hat{\mathfrak{s}}^2}\left(1 + \frac{16\nu^2|\mathcal{P}||\mathcal{S}|}{\hat{\rho}}\right) \ ,$$

$$\|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}'_0\| \leq \varepsilon\left(1 + \frac{16\nu^2|\mathcal{P}|^{1/2}|\mathcal{S}|}{\hat{\rho}}\right) \ ,$$
$$\|\boldsymbol{\alpha}_\infty - \boldsymbol{\alpha}'_\infty\| \leq \frac{\varepsilon}{\hat{\mathfrak{s}}} + \frac{1+\sqrt{5}}{2}\frac{\varepsilon\nu|\mathcal{P}|^{1/2}}{\hat{\mathfrak{s}}^2}\left(1 + \frac{16\nu^2|\mathcal{P}|\ |\mathcal{S}|}{\hat{\rho}}\right) \ .$$

The result now follows from an adequate choice of c_1. □

The other half of the proof results from combining Lemmas 9.3.3 and 9.3.5 with Lemma 5.4.2 to obtain a bound for $|f_S(x) - f_{S'}(x)|$. This is a delicate step, because some of the bounds given above involve quantities that are defined in terms of S. Therefore, all these parameters need to be controlled in order to ensure that the bounds do not grow too large. This will be done by using a variant of McDiarmid's inequality that accounts for the occurrence of rare bad events. In particular, we define the properties that make S a good sample and show that for large enough m they are satisfied with high probability.

Definition 9.3.6. *We say that a sample S of m i.i.d. examples from D is* good *if the following conditions are satisfied for any $z'^m = (x'^m, y'^m) \in \mathrm{supp}(D)$:*

1. $|x^i| \leq ((1/c)\ln(4m^4))^{1/(1+\eta)}$ *for all* $1 \leq i \leq m$,
2. $\|\mathbf{H} - \mathbf{H}'\|_F \leq 4/(\tau\pi m)$,
3. $\min\{\mathfrak{s}_n(\mathbf{H}_\lambda), \mathfrak{s}_n(\mathbf{H}'_\lambda)\} \geq \mathfrak{s}/2$,
4. $\mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2 \geq \rho/2$.

Lemma 9.3.7. *Suppose D satisfies Assumptions 4 and 5. There exists a quantity $M = \mathsf{poly}(\nu, \pi, \mathfrak{s}, \rho, \tau, |\mathcal{P}|, |\mathcal{S}|)$ such that if $m \geq M$, then S is good with probability at least $1 - 1/m^3$.*

Proof. First note that by Assumption 5, writing $L = ((1/c)\ln(4m^4))^{1/(1+\eta)}$ a union bound yields

$$\mathbb{P}\left[\bigvee_{i=1}^m |x^i| > L\right] \leq m\exp(-cL^{1+\eta}) = \frac{1}{4m^3} \ .$$

Now let $\bar{m}$ denote the length of $(x^1, \ldots, x^{m-1}) \cap \mathcal{PS}$. Note that we have $\min\{\tilde{m}, \tilde{m}'\} \geq \bar{m}$ and $\mathbb{E}_S[\bar{m}] = \pi(m-1)$. Thus, for any $\Delta \in (0,1)$ the Chernoff bound gives

$$\mathbb{P}[\bar{m} < \pi(m-1)(1-\Delta)] \leq \exp\left(-\frac{(m-1)\pi\Delta^2}{2}\right) \leq \exp\left(-\frac{m\pi\Delta^2}{4}\right) \ ,$$

where we have used that $(m-1)/m \geq 1/2$ for $m \geq 2$. Therfore, taking $\Delta = \sqrt{(4/m\pi)\ln(4m^3)}$ we see that with probability at least $1 - 1/(4m^3)$ we have

$$\min\{\tilde{m}, \tilde{m}'\} \geq (m-1)\pi(1-\Delta) \geq \frac{m\pi(1-\Delta)}{2} \ .$$

Now note that $m \geq (16/\pi)\ln(4m^3)$ implies $\Delta \leq 1/2$. Therefore, applying Lemma 9.3.3 we can see that $\|\mathbf{H} - \mathbf{H}'\|_F \leq 4/(\tau\pi m)$ holds with probability at least $1 - 1/(4m^3)$ whenever

$$m \geq \max\left\{2, \frac{16}{\pi}\ln(4m^3), \frac{2}{\tau\pi\nu\sqrt{|\mathcal{P}||\mathcal{S}|}}\right\} \ .$$

For the third claim note that by Lemma A.2.1 we have

$$|\mathfrak{s}_n(\mathbf{H}_\lambda) - \mathfrak{s}_n(\mathbf{H}'_\lambda)| \leq \|\mathbf{H}_\lambda - \mathbf{H}'_\lambda\|_F \leq \|\mathbf{H} - \mathbf{H}'\|_F \ .$$

Thus, from the argument we just used in the previous bound we can see that when $m \geq 2$ the function $\Phi(S) = \mathfrak{s}_n(\mathbf{H}_\lambda)$ is strongly difference-bounded by $(b_\mathfrak{s}, \theta_\mathfrak{s}/m, \exp(-K_\mathfrak{s} m))$ with $b_\mathfrak{s} = 2\nu\sqrt{|\mathcal{P}||\mathcal{S}|}$, $\theta_\mathfrak{s} = 2/(\tau\pi(1-\Delta))$, and $K_\mathfrak{s} = \pi\Delta^2/4$ for any $\Delta \in (0,1)$. Now note that by Lemma A.2.1 and the previous goodness condition on $\|\mathbf{H} - \mathbf{H}'\|_F$ we have

$$\min\{\mathfrak{s}_n(\mathbf{H}_\lambda), \mathfrak{s}_n(\mathbf{H}'_\lambda)\} \geq \mathfrak{s}_n(\mathbf{H}_\lambda) - \|\mathbf{H} - \mathbf{H}'\|_F \geq \mathfrak{s}_n(\mathbf{H}_\lambda) - \frac{4}{\nu\pi m} \ .$$

Furthermore, taking $\Delta = 1/2$ and assuming that

$$m \geq \max\left\{\frac{\nu\tau\pi\sqrt{|\mathcal{P}||\mathcal{S}|}}{2}, \left(9 + \frac{288}{\pi}\right)\ln\left(3 + \frac{96}{\pi}\right), \frac{32}{\pi}\ln(8m^3)\right\} \ ,$$

we can apply Corollary A.1.4 with $\delta = 1/(4m^3)$ to see that

$$\mathfrak{s}_n(\mathbf{H}_\lambda) - \frac{4}{\nu\pi m} \geq \mathfrak{s} - \sqrt{\frac{128}{\tau^2\pi^2 m}\ln(8m^3)} - \frac{4}{\nu\pi m}$$

holds with probability at least $1 - 1/(4m^3)$. Hence, we get

$$\min\{\mathfrak{s}_n(\mathbf{H}_\lambda), \mathfrak{s}_n(\mathbf{H}'_\lambda)\} \geq \mathfrak{s} - \sqrt{\frac{128}{\tau^2\pi^2 m}\ln(8m^3)} - \frac{4}{\nu\pi m} \geq \mathfrak{s} - \frac{\mathfrak{s}}{4} - \frac{\mathfrak{s}}{4} = \frac{\mathfrak{s}}{2}$$

for any sample size such that $m \geq \max\{16/(\nu\pi\mathfrak{s}), (2048/\tau^2\pi^2\mathfrak{s}^2)\ln(8m^3)\}$. To prove the fourth bound we shall study the stability of the function $\Phi(S) = \mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2$. We begin with the following chain of inequalities, which follows from Lemma A.2.1 and $\mathfrak{s}_n(\mathbf{H}_\lambda) \geq \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)$:

$$\begin{aligned}
|\Phi(S) - \Phi(S')| &= \left|(\mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2) - (\mathfrak{s}_n(\mathbf{H}'_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}'_\lambda)^2)\right| \\
&\leq |\mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_n(\mathbf{H}'_\lambda)^2| + |\mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}'_\lambda)^2| \\
&= |\mathfrak{s}_n(\mathbf{H}_\lambda) + \mathfrak{s}_n(\mathbf{H}'_\lambda)|\,|\mathfrak{s}_n(\mathbf{H}_\lambda) - \mathfrak{s}_n(\mathbf{H}'_\lambda)| \\
&\quad + |\mathfrak{s}_{n+1}(\mathbf{H}_\lambda) + \mathfrak{s}_{n+1}(\mathbf{H}'_\lambda)|\,|\mathfrak{s}_{n+1}(\mathbf{H}_\lambda) - \mathfrak{s}_{n+1}(\mathbf{H}'_\lambda)| \\
&\leq (2\mathfrak{s}_n(\mathbf{H}_\lambda) + \|\mathbf{H}_\lambda - \mathbf{H}'_\lambda\|)\,\|\mathbf{H}_\lambda - \mathbf{H}'_\lambda\| \\
&\quad + (2\mathfrak{s}_{n+1}(\mathbf{H}_\lambda) + \|\mathbf{H}_\lambda - \mathbf{H}'_\lambda\|)\,\|\mathbf{H}_\lambda - \mathbf{H}'_\lambda\| \\
&\leq 4\mathfrak{s}_n(\mathbf{H}_\lambda)\|\mathbf{H} - \mathbf{H}'\|_F + 2\|\mathbf{H} - \mathbf{H}'\|_F^2 \ .
\end{aligned}$$

Now we can use this last bound to show that $\Phi(S)$ is strongly difference-bounded by $(b_\rho, \theta_\rho/m, \exp(-K_\rho m))$ with the definitions: $b_\rho = 16\nu^2|\mathcal{P}||\mathcal{S}|$, $\theta_\rho = 64\mathfrak{s}/(\tau\pi)$ and $K_\rho = \min\{\mathfrak{s}^2\tau^2\pi^2/256, \pi/64\}$. For b_ρ just observe that from Lemma 9.3.3 and $\mathfrak{s}_n(\mathbf{H}_\mathfrak{s}) \leq \|\mathbf{H}_\mathfrak{s}\|_F \leq \nu\sqrt{|\mathcal{P}||\mathcal{S}|}$ we get

$$4\mathfrak{s}_n(\mathbf{H}_\lambda)\|\mathbf{H} - \mathbf{H}'\|_F + 2\|\mathbf{H} - \mathbf{H}'\|_F^2 \leq 16\nu^2|\mathcal{P}||\mathcal{S}| \ .$$

By the same arguments used above, if m is large enough we have $\|\mathbf{H} - \mathbf{H}'\|_F \leq 4/(\tau\pi m)$ with probability at least $1 - \exp(-m\pi/16)$. Furthermore, by taking $\Delta = 1/2$ in the stability argument given above for $\mathfrak{s}_n(\mathbf{H}_\lambda)$, and invoking Corollary A.1.4 with $\delta = 2\exp(-Km)$ for some $0 < K \leq K_\mathfrak{s}/2 = \pi/32$, we get

$$\mathfrak{s}_n(\mathbf{H}_\lambda) \leq \mathfrak{s} + \sqrt{\frac{128K}{\tau^2\pi^2}} \ ,$$

with probability at least $1 - 2\exp(-Km)$. Thus, for $K = \min\{\pi/32, \mathfrak{s}^2\tau^2\pi^2/128\}$ we get $\mathfrak{s}_n(\mathbf{H}_\lambda) \leq 2\mathfrak{s}$. If we now combine the bounds for $\|\mathbf{H} - \mathbf{H}'\|_F$ and $\mathfrak{s}_n(\mathbf{H}_\lambda)$, we get

$$4\mathfrak{s}_n(\mathbf{H}_\lambda)\|\mathbf{H} - \mathbf{H}'\|_F + 2\|\mathbf{H} - \mathbf{H}'\|_F^2 \leq \frac{32\mathfrak{s}}{\tau\pi m} + \frac{32}{\tau^2\pi^2 m^2} \leq \frac{64\mathfrak{s}}{\tau\pi m} = \frac{\theta_\rho}{m} \ ,$$

where have assumed that $m \geq 1/(\tau\pi\mathfrak{s})$. To get K_ρ note that the above bound holds with probability at least

$$1 - e^{-m\pi/16} - 2e^{-Km} \geq 1 - 3e^{-Km} \geq 1 - e^{-Km/2} = 1 - e^{-K_\rho m} \ ,$$

where we have used that $K \leq \pi/16$ and assumed that $m \geq 2\ln(3)/K$. Finally, applying Corollary A.1.4 to $\Phi(S)$ we see that with probability at least $1 - 1/(4m^3)$ one has

$$\mathfrak{s}_n(\mathbf{H}_\lambda)^2 - \mathfrak{s}_{n+1}(\mathbf{H}_\lambda)^2 \geq \rho - \sqrt{\frac{2^{15}\mathfrak{s}^2}{\tau^2\pi^2 m}\ln(8m^3)} \geq \frac{\rho}{2} \ ,$$

whenever the sample size m is at least

$$\max\left\{\frac{2^{17}\mathfrak{s}^2}{\tau^2\pi^2\rho^2}\ln(8m^3), \frac{\nu^2\tau\pi|\mathcal{P}||\mathcal{S}|}{4\mathfrak{s}}, \left(9 + \frac{18}{K_\rho}\right)\ln\left(3 + \frac{6}{K_\rho}\right), \frac{2}{K_\rho}\ln(8m^3)\right\} \ .$$

□

Now we can give two upper bounds for $|f_S(x) - f_{S'}(x)|$: a tighter bound that holds for "good" samples S and S', and a another one that holds for all samples. These bounds can be combined by using the variant of McDiarmid's inequality for dealing with functions that do not satisfy the bounded differences assumption almost surely given in [Kut02]. The rest of the proof then follows the same scheme as the standard one for deriving generalization bounds for stable algorithms [BE02; MRT12].

Lemma 9.3.8. *Let $\gamma_1 = 64\nu^4|\mathcal{P}|^2|\mathcal{S}|^{3/2}/(\tau\mathfrak{s}^3\rho\pi)$ and $\gamma_2 = 2\nu|\mathcal{P}|^{1/2}|\mathcal{S}|^{1/2}/\mathfrak{s}$. There exists a constant $c_2 > 0$ such that the function the function $\Phi(S) = R(f_S) - \hat{R}_S(f_S)$ is strongly difference-bounded by $(4\nu + 2\nu/m, c_2\gamma_1 m^{-5/6}\ln m, 1/m^3)$ for every sample size*

$$m \geq \max\left\{M, \frac{16\sqrt{2}}{\tau\pi\sqrt{\rho}}, \exp\left(6\ln\gamma_2(1.2c\ln\gamma_2)^{1/\eta}\right)\right\} \ .$$

Proof. We will write for short $f = f_S$ and $f' = f_{S'}$. We start by defining

$$\beta_1 = \mathbb{E}_{x\sim D_\Sigma}[|f(x) - f'(x)|] \ ,$$
$$\beta_2 = \max_{1\leq i\leq m-1}|f(x^i) - f'(x^i)| \ .$$

We first show that $|\Phi(S) - \Phi(S')| \leq \beta_1 + \beta_2 + 2\nu/m$. By definition of Φ we can write

$$|\Phi(S) - \Phi(S')| \leq |R(f) - R(f')| + |\hat{R}_S(f) - \hat{R}_{S'}(f')| \ .$$

By Jensen's inequality, the first term can be upper bounded as

$$|R(f) - R(f')| \leq \mathbb{E}_{(x,y)\sim D}\left[|\,|f(x) - y| - |f'(x) - y|\,|\right] \leq \beta_1 \ .$$

Now, using the triangle inequality and $|f(x^m) - y^m|, |f'(x'^m) - y'^m| \leq 2\nu$, the second term can be bounded as follows:

$$|\hat{R}_S(f) - \hat{R}_{S'}(f')| \leq \frac{2\nu}{m} + \frac{1}{m}\sum_{i=1}^{m-1}|f(x^i) - f'(x^i)| \leq \frac{2\nu}{m} + \beta_2\frac{m-1}{m} \ .$$

Observe that for any samples S and S' we have $\beta_1, \beta_2 \leq 2\nu$. This provides an almost-sure upper bound needed in the definition of strongly difference-boundedness. We use this bound when the sample S is

not good. When m is large enough, Lemma 9.3.7 guarantees that this event will occur with probability at most $1/m^3$. It remains to bound β_1 and β_2 assuming that S is good. In the first place note that by Lemma 9.3.7, $m \geq \max\{M, 16\sqrt{2}/(\tau\pi\sqrt{\rho})\}$ implies $\|\mathbf{H} - \mathbf{H}'\|_F \leq \sqrt{\hat{\rho}}/4$. Thus, by combining Lemmas 5.4.2, 9.3.4, 9.3.5, and 9.3.7, we see that the following holds for any $x \in \Sigma^\star$:

$$\begin{aligned}|f(x) - f'(x)| &\leq \left(\frac{2\nu|\mathcal{P}|^{1/2}|\mathcal{S}|^{1/2}}{\mathfrak{s}}\right)^{|x|+1} \frac{32c_1(|x|+2)\nu^3|\mathcal{P}|^{3/2}|\mathcal{S}|}{m\tau\pi\mathfrak{s}^2\rho} \\ &= \frac{c_1\gamma_1}{m}\exp(|x|\ln\gamma_2 + \ln(|x|+2)) \ .\end{aligned}$$

Now let $L = ((1/c)\ln(4m^4))^{1/(1+\eta)}$. The bound we just showed guarantees that there exists a constant C such that when $m \geq \exp(6\ln\gamma_2(1.2c\ln\gamma_2)^{1/\eta})$, we get $|f(x) - f'(x)| \leq C\gamma_1 m^{-5/6}\ln m$ for every $|x| \leq L$. Thus, we can write

$$\begin{aligned}\beta_1 &\leq \mathbb{E}_{x\sim D_\Sigma}\left[\,|f(x) - f'(x)| \mid |x| \leq L\,\right] + 2\nu\,\mathbb{P}_{x\sim D_\Sigma}\left[\,|x| \geq L\,\right] \\ &\leq C\gamma_1 m^{-5/6}\ln m + \nu/2m^3 \ ,\end{aligned}$$

and $\beta_2 \leq C\gamma_1 m^{-5/6}\ln m$, where the last bound follows from the goodness of S. Combining these bounds yields the desired result. □

The following is the proof of our main result.

Proof of Theorem 9.3.1. The result follows from an application of Theorem A.1.2 to $\Phi(S)$, defined as in Lemma 9.3.8. In particular, for large enough m, the following holds with probability at least $1 - \delta$:

$$R(f_S) \leq \hat{R}_S(f_S) + \mathbb{E}_{S\sim D^m}[\Phi(S)] + \sqrt{C\gamma_1^2\frac{\ln^2 m}{m^{2/3}}\ln\left(\frac{1}{\delta - \frac{6\nu}{C'\gamma_1}\frac{1}{m^{7/6}\ln m}}\right)} \ ,$$

for some constants C, C' and $\gamma_1 = \nu^4|\mathcal{P}|^2|\mathcal{S}|^{3/2}/\tau\mathfrak{s}^3\rho\pi$. Thus, it remains to bound $\mathbb{E}_{S\sim D^m}[\Phi(S)]$.

First note that we have $\mathbb{E}_{S\sim D^m}[R(f_S)] = \mathbb{E}_{S,z\sim D^{m+1}}[|f_S(x) - y|]$. On the other hand, we can also write $\mathbb{E}_{S\sim D^m}[\hat{R}_S(f_S)] = \mathbb{E}_{S,z\sim D^{m+1}}[|f_{S'}(x) - y|]$, where S' is a sample of size m containing z and $m-1$ other points in S chosen at random. Thus, by Jensen's inequality we can write

$$|\mathbb{E}_{S\sim D^m}[\Phi(S)]| \leq \mathbb{E}_{S,z\sim D^{m+1}}[|f_S(x) - f_{S'}(x)|] \ .$$

Now an argument similar to the one used in Lemma 9.3.8 for bounding β_1 can be used to show that, for large enough m, the following inequality holds:

$$\left|\mathbb{E}_{S\sim D^m}[\Phi(S)]\right| \leq C\gamma_1\frac{\ln m}{m^{5/6}} + \frac{2\nu}{m^3} \ ,$$

which completes the proof. □

Conclusion and Open Problems

The previous chapters have covered a large part of what was known about learning automata with state-merging and spectral methods. They also provide new results and describe novel points of view on results that were already known. Now it is time to dwell a little bit into the unknown. To conclude this thesis we look back into the material described in previous chapters to point out possible extensions and pose interesting open questions that can give rise to further research. What follows is an account that mixes concrete open problems with broad speculations, all centered around the areas covered by this thesis. For the sake of organization, the discussion has been split into three parts. We would like to note that this account is by no means exhaustive; it is just a sample biased by the personal interests of the author.

Generalized and Alternative Algorithms

We begin by discussing several more or less obvious extensions and generalizations of the learning algorithms given in this thesis.

We have seen in Chapter 2 that statistical tests based on Efron's bootstrap provide a very powerful method for similarity testing between distributions over $\Sigma^\star$. In practice these tests behave much better than tests based on VC bounds. However, lower confidence intervals based on bootstrap estimates cannot be used to certify dissimilarity between probability distributions. Thus, we used VC bounds to implement this side of the test. Trying to derive dissimilarity tests based on bootstrap estimates with similarly good behavior would be very interesting. In particular, they can yield practical state-merging algorithms that can learn confidently from smaller samples. One possible direction is to observe that the *histograms* of the bootstrap estimates $\hat{\mu}_i$ tend to be symmetric when $\mu_\star > 0$ and asymmetric when $\mu_\star = 0$. Designing, implementing, and analyzing tests based on this idea is a promising line of work.

A useful modification to the adaptive on-line state-merging algorithm given in Chapter 3 would be to be able to recycle parts of the current hypothesis when a change in the target distribution is detected. That entails keeping track of the merge history, being able to detect whether the change affects a particular state or transition, and backtrace the state-merging process exactly to the point where this transition or state was added to the hypothesis. In practice this would yield a great advantage when trying to track targets that change often but very locally. There are two main challenges here: how to implement the backtracking efficiently and how to decide when making local changes to the current hypothesis will be enough to learn the new target.

An interesting follow-up on the results from Chapter 4 would be to implement those methods and see how they behave in practice. In particular, we would like to compare our algorithm for learning sequential transductions with other state-merging algorithms that have been proposed for this problem. Of course, it would also be interesting to identify more FSM models that can be represented using GPDFA and give PAC learning results for them using these techniques.

We mentioned in Chapter 8 that one of the reasons for introducing an optimization-based method for learning WFA is that it makes it possible to add different regularizing penalties to the spectral learning algorithm. We showed how this approach can be used to learn proper PNFA by imposing that the output must be a probabilistic automaton. Another interesting regularization term that one may want to add is a *group matrix norm* that promotes sparsity in the transition structure of hypothesis automata. This is interesting because in practice one observes that good solutions found via Expectation-Maximization tend to have sparse transition structures. We also note that many other regularization techniques for

learning with matrices exist – some of them may also interesting interpretations when applied to the operators of a WFA; see [KSST12] for details.

In Chapter 9 we were able to prove a generalization bound for an algorithm combining matrix completion with spectral learning. The particular matrix completion algorithm we analyzed was based on a Frobenius norm regularization. However, it is well known in matrix completion theory that, if one intends to minimize the rank of the completed matrix, it is best to use a regularizer based on the nuclear norm [CT10; CP10; Rec11; FSSS11]. In fact, in some preliminary experiments we observed that in practice the algorithm does behave better when nuclear norm regularization is used. Thus, an open problem is to see whether our techniques can be extended to this other algorithm. However, we note that this may not be possible to achieve using stability analyses. It was recently proved in [XCM12] that some sparsity-inducing algorithms can never be stable. Since the nuclear norm is tries to induce sparsity on the spectrum of hypothesis matrices, we may regard matrix completion with nuclear norm regularization as a sparsity-inducing algorithm. An alternative approach would to be compute some measure of complexity of WFA like VC dimension or Rademacher complexity.

Choice or Reduction of Input Parameters

All learning algorithms discussed in this thesis take some parameters as input: number of states n, distinguishability μ, basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, etc. How to choose these parameters in practice is usually a matter of trial-and-error mixed with cross-validation and domain knowledge. This is not very satisfactory, and although some fine-tuning will always be necessary, one would like to have some theory to guide the choice of parameters. Or even better, one could hope to design learning algorithms that do not need any input parameters at all.

In the case of state-merging algorithms, the data stream implementation given in Chapter 3 includes a search strategy to find n and μ when they are not known. On the other hand, the batch implementation from Chapter 4 for learning GPDFA – and its immediate predecessor [CG08] – do not need μ as an input parameter. Instead, the algorithm makes confident merges in view of the available sample, and guarantees that these will be correct whenever the sample is large enough, were the bound does depend on the true distinguishability. This is not possible in the data stream setting because a potentially infinite sample is available there; thus we need a μ to bound the time the algorithm is allowed to wait until it makes a decision about a candidate state. However, requiring the number of states in these two algorithms does feel a little artificial. Indeed, if the sample is large enough – in the batch case – or the input μ is correct – in the on-line setting – then with high probability all tests will be correct and a DFA with finitely many states will be identified. But what happens if some incorrect merge decision is made by the algorithm? Then, after that point we cannot guarantee that distributions of examples in the remaining candidate states correspond to distributions generated by states in the target. Thus, the rest of the process could go wrong, and in principle the number of states in the hypothesis could grow like the number of prefixes in the sample, yielding a very large automaton. Using the input n as a stopping condition for the algorithm precludes this from happening. Analyzing whether this happens in practice, or designing a different split/merge policy for which this does not happen is necessary in order to remove n from the input in state-merging algorithms.

In the case of spectral methods, the number of states seems less critical. For one thing, using concentration properties about the singular values of empirical Hankel matrices $\hat{\mathbf{H}}$ one can easily compute a lower bound on the number of states for any given sample size. Furthermore, with large enough samples this bound converges to the true value; see [Bai11a] for details. A more challenging problem in this case is how to choose a good basis for a given sample. We showed in Chapter 6 that a random selection strategy will succeed with high probability. A heuristic based on this approach is to choose the most frequent prefixes and suffixes in a sample because they are more likely to appear in a randomly selected basis. However, one could arguably do a better job trying to find a basis by looking at the spectrum of the resulting Hankel matrix. For example, a greedy algorithm that adds prefixes (and suffixes) one at a time while trying to maximize the smallest singular value of the corresponding Hankel matrix is a good candidate for finding a basis which captures all states and minimizes the noise in the tail of the singular values. How to analyze and efficiently implement this type of approach is an interesting open problem

that could potentially improve the performance of spectral methods in practice.

Another similar question is to study the effect of parameter mismatch in algorithms for learning FSM. For example, what happens if we give a number of states too small to the spectral method: can we quantify the error in terms of, say, the singular values that will be missing from our hypothesis? Note that this is close to analyzing how state-merging and spectral algorithms behave in non-realizable settings. This is the subject of the next section.

Learning Distributions in Non-Realizable Settings

A corollary from the results in Chapter 1 states that state-merging algorithms can accurately learn distributions on $\Sigma^\star$ which are close to being realized by PDFA. We saw that this is a consequence of formulating state-merging algorithms using statistical queries. In fact, a similar conclusion can be drawn for the spectral algorithm that learns PNFA. Indeed, noting that any finite sub-block of the Hankel matrix $\mathbf{H}_D$ of a probability distribution D over $\Sigma^\star$ can be estimated with statistical queries, the reasoning used in Chapter 1 can be applied here as well; such a possibility was already sketched in [HKZ09]. Thus, we see that both state-merging and spectral algorithms for learning distributions over $\Sigma^\star$ are resilient to outliers and can learn distributions "close" to their original target class.

These observations yield timid – yet encouraging – indications that these methods can learn some distributions beyond the classes they use for representing hypotheses. That is, we get mild learning results in some non-realizable settings. This raises the natural question of whether these hypothesis classes and algorithmic paradigms can be used for learning good models under more relaxed conditions. Such results would be interesting for two reasons. First, because these algorithms can already be run on any sample from $\Sigma^\star$. Hence, we could use the same algorithms that learn in the realizable case for learning these wider classes of distributions. And second, because these algorithms find hypotheses from classes which are well understood (PDFA and WFA) in terms of how they are used for inference and prediction. So, what kind of results can one hope to prove along these lines?

One possibility is to seek structural results stating that distributions over $\Sigma^\star$ that satisfy this and that condition can be well approximated by, say, PDFA or stochastic WFA. We are not aware of many results of this type. One example is the proof in [GKPP06] that every PNFA can be approximated by a large enough PDFA; the bound however, seems too large for any practical purpose. Thus, besides their immediate applications to learning theory, these results would have intrinsic mathematical value in themselves; i.e. they would shed some light into what classes of distributions can be approximated by probabilistic automata. Speculatively speaking, it seems reasonable to try to generalize parameters that measure the complexity of learning PDFA and WFA – the distinguishability μ and the smallest singular value $\mathfrak{s}_r(\mathbf{H})$ – to distributions not realizable by FSM, and then see what is the effect of these parameters on the approximability of general distributions by probabilistic FSM.

An alternative path would be, instead of analyzing how known algorithms perform in non-realizable settings, to use these algorithms as building blocks for deriving new learning algorithms for more general problems using the same hypothesis classes. A promising idea in this direction is to use the output of state-merging or spectral algorithms as the initial point for some type of *local maximum likelihood* optimization. For example, these can be based on heuristics like Expectation-Maximization or standard gradient ascent on the log-likelihood. Promising experimental results with this type of methods have been recently reported in [Bai11b; KNW13; CL13]. These experiments suggest that an algorithm based on this principle is in general faster than standard EM and statistically better than spectral methods by themselves, both in realizable and non-realizable settings. Developing new theory to explain this findings is an interesting open problem. In particular, combined with other recent spectral learning methods for many classes of probabilistic models, it could lead to truly practical learning algorithms with formal guarantees for these classes.

Of course, it could be that probabilistic FSM can only provide very coarse approximations for general classes of distributions over $\Sigma^\star$. In this case, one could try to design a boosting-like approach [SF12] that uses a state-merging or spectral algorithm as a weak learner to obtain a hypothesis which is a mixture of automata. Though it is less clear how to apply the boosting paradigm to a learning problem on probability distributions, some attempts have been made to derive algorithms of this type [Pav04].

In fact, we tried to apply these ideas directly to boost the spectral method and entered the Pautomac competition on learning PFA and HMM [VEH12]. However, we found out that our algorithm always ended up getting stuck on trying to model the longest strings in the sample. Clearly, more work is needed on this approach, specially on designing a reweighting scheme for the boosting iterations that avoids putting too much weight on long strings. It would also be interesting to characterize for which distributions over $\Sigma^\star$ can state-merging and spectral methods be regarded as provable weak learners.

A question related to both, non-realizable learning and input parameters, is how one should choose the parameters given to state-merging or spectral algorithms when an arbitrary sample from a non-realizable target is available. That is: how can one make the most of these algorithms in a practical problem without making any assumption on the target distribution? This is basically a statistical model selection problem. In particular, given a large enough number of states, these algorithms will learn the empirical distribution defined by the sample, thus yielding a grossly overfitted model. On the other hand, if we choose a model with too few states we may be losing generalization power. In classification problems this can be solved by providing generalization bounds on the predictive performance of a classifier and then minimizing those; sometimes this is referred as *structural risk minimization* [Vap99]. When the problem is to learn a probability distribution, one approach is to give generalization bounds for the relative entropy between the target and a hypothesis in terms of the empirical relative entropy – which is directly related to the sample likelihood – and a penalty term that depends on the complexity of the model. For example, bounds of this sort exist for density estimation with *maximum entropy models* [DPS04; MMMSW09]. This makes it possible to select a model that minimizes the sum of the empirical relative entropy plus a penalty term that, in the case of stochastic FSM, would depend on the number of states and possibly other quantities. To prove this type of bounds one can resort to stability arguments like the ones in Chapter 9, or to VC-like quantities. Computing the latter in the case of PDFA and WFA is an open problem that could also yield improvements on the analysis of other learning algorithms based on these hypothesis classes, as we have already mentioned above.

Bibliography

[ATW01] N. Abe, J. Takeuchi, and M. K. Warmuth. "Polynomial learnability of stochastic rules with respect to the KL-divergence and quadratic distance". In: *IEICE Transactions on Information and Systems* (2001).

[AW92] N. Abe and M. K. Warmuth. "On the computational complexity of approximating distributions by probabilistic automata". In: *Machine Learning* (1992).

[Agg07] C. Aggarwal, ed. *Data streams: Models and algorithms*. Springer, 2007.

[Aka77] H. Akaike. "On entropy maximization principle". In: *Applications of Statistics*. North-Holland, 1977.

[AD94] R. Alur and D. L. Dill. "A theory of timed automata". In: *Theoretical computer science* (1994).

[ACHKSZ11] A. Anandkumar, K. Chaudhuri, D. Hsu, S. M. Kakade, L. Song, and T. Zhang. "Spectral methods for learning multivariate latent tree structure". In: *Neural Information Processing Systems Conference (NIPS)* (2011).

[AFHKL12a] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y-K. Liu. "A spectral algorithm for latent Dirichlet allocation". In: *Neural Information Processing Systems Conference (NIPS)* (2012).

[AFHKL12b] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y-K. Liu. "Two SVDs suffice: Spectral decompositions for probabilistic topic modeling and latent Dirichlet allocation". In: *CoRR* abs/1204.6703 (2012).

[AGHKT12] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. "Tensor decompositions for learning latent variable models". In: *CoRR* abs/1210.7559 (2012).

[AHHK12] A. Anandkumar, D. Hsu, F. Huang, and S. M. Kakade. "Learning mixtures of tree graphical models". In: *Neural Information Processing Systems Conference (NIPS)* (2012).

[AHK12a] A. Anandkumar, D. Hsu, and S. M. Kakade. "A method of moments for mixture models and hidden Markov models". In: *Conference on Learning Theory (COLT)* (2012).

[AHK12b] A. Anandkumar, D. Hsu, and S. M. Kakade. "Learning high-dimensional mixtures of graphical models". In: *CoRR* abs/1203.0697 (2012).

[AB12] E. Anceaume and Y. Busnel. "Sketch $\star$-metric: Comparing data streams via sketching". In: *HAL* hal-00764772 (2012).

[ADLV] M. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe. "Interior-point methods for large-scale cone programming". In: *Optimization for Machine Learning*. The MIT Press,

[Ang82] D. Angluin. "Inference of reversible languages". In: *Journal of the ACM* (1982).

[Ang87] D. Angluin. "Queries and concept learning". In: *Machine Learning* (1987).

[AD98] J. A. Aslam and S. E. Decatur. "Specification and simulation of statistical query algorithms for efficiency and noise tolerance". In: *Journal of Computing Systems Science* (1998).

[BHHK03] C. Baier, B. Haverkort, H. Hermanns, and J-P. Katoen. “Model-checking algorithms for continuous-time Markov chains”. In: *IEEE Transactions on Software Engineering* (2003).

[Bai11a] R. Bailly. “Méthodes spectrales pour l’inférence grammaticale probabiliste de langages stochastiques rationnels”. PhD thesis. Aix-Marseille Université, 2011.

[Bai11b] R. Bailly. “Quadratic weighted automata: Spectral algorithm and likelihood maximization”. In: *Asian Conference on Machine Learning (ACML)* (2011).

[BD11] R. Bailly and F. Denis. “Absolute convergence of rational series is semi-decidable”. In: *Information and Computation* (2011).

[BDR09] R. Bailly, F. Denis, and L. Ralaivola. “Grammatical inference as a principal component analysis problem”. In: *International Conference on Machine Learning (ICML)* (2009).

[BHD10] R. Bailly, A. Habrard, and F. Denis. “A spectral approach for probabilistic grammatical inference on trees”. In: *Conference on Algorithmic Learning Theory (ALT)* (2010).

[BCLQ13] B. Balle, X. Carreras, F.M. Luque, and A. Quattoni. “Spectral learning of weighted automata: A forward-backward perspective”. Submitted. 2013.

[BCG10a] B. Balle, J. Castro, and R. Gavaldà. “A lower bound for learning distributions generated by probabilistic automata”. In: *Conference on Algorithmic Learning Theory (ALT)* (2010).

[BCG10b] B. Balle, J. Castro, and R. Gavaldà. “Learning PDFA with asynchronous transitions”. In: *International Colloquium on Grammatical Inference (ICGI)* (2010).

[BCG12a] B. Balle, J. Castro, and R. Gavaldà. “Adaptive learning of probabilistic deterministic automata from data streams”. Submitted. 2012.

[BCG12b] B. Balle, J. Castro, and R. Gavaldà. “Bootstrapping and learning PDFA in data streams”. In: *International Colloquium on Grammatical Inference (ICGI)* (2012).

[BCG13] B. Balle, J. Castro, and R. Gavaldà. “Learning probabilistic automata: A study in state distinguishability”. In: *Theoretical Computer Science* (2013).

[BM12] B. Balle and M. Mohri. “Spectral learning of general weighted automata via constrained matrix completion”. In: *Neural Information Processing Systems Conference (NIPS)* (2012).

[BQC11] B. Balle, A. Quattoni, and X. Carreras. “A spectral learning algorithm for finite state transducers”. In: *European Conference on Machine Learning (ECML)* (2011).

[BQC12] B. Balle, A. Quattoni, and X. Carreras. “Local loss optimization in operator models: A new insight into spectral learning”. In: *International Conference on Machine Learning (ICML)* (2012).

[BFRSW13] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. “Testing closeness of discrete distributions”. In: *Journal of the ACM* (2013).

[BP66] L. E. Baum and T. Petrie. “Statistical inference for probabilistic functions of finite state Markov chains”. In: *The Annals of Mathematical Statistics* (1966).

[BPSW70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains”. In: *The Annals of Mathematical Statistics* (1970).

[BT09] A. Beck and M. Teboulle. “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”. In: *SIAM Journal on Imaging Sciences* (2009).

[BBBKV00] A. Beimel, F. Bergadano, N.H. Bshouty, E. Kushilevitz, and S. Varricchio. “Learning functions represented as multiplicity automata”. In: *Journal of the ACM* (2000).

[Ber79] J. Berstel. *Transductions and context-free languages*. Teubner, 1979.

[BR88] J. Berstel and C. Reutenauer. *Rational series and their languages*. Springer, 1988.

[Bif10] A. Bifet. *Adaptive stream mining: Pattern learning and mining from evolving data streams*. IOS Press, 2010.

[BSG11] B. Boots, S. Siddiqi, and G. Gordon. "Closing the learning planning loop with predictive state representations". In: *International Journal of Robotic Research* (2011).

[BLB04] S. Boucheron, G. Lugosi, and O. Bousquet. "Concentration inequalities". In: *Advanced Lectures on Machine Learning* (2004).

[BLM13] S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press, 2013.

[BBL04] O. Bousquet, S. Boucheron, and G. Lugosi. "Introduction to statistical learning theory". In: *Advanced Lectures on Machine Learning* (2004).

[BE02] O. Bousquet and A. Elisseeff. "Stability and generalization". In: *Journal of Machine Learning Research* (2002).

[BPCPE11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends in Machine Learning* (2011).

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[Bre96] L. Breiman. "Bagging predictors". In: *Machine Learning* (1996).

[CCS10] J-F. Cai, E. J. Candès, and Z. Shen. "A singular value thresholding algorithm for matrix completion". In: *SIAM Journal on Optimization* (2010).

[CP10] E. J. Candes and Y. Plan. "Matrix completion with noise". In: *Proceedings of the IEEE* (2010).

[CT10] E. J. Candes and T. Tao. "The power of convex relaxation: Near-optimal matrix completion". In: *IEEE Transactions on Information Theory* (2010).

[CP71] J. W. Carlyle and A. Paz. "Realizations by stochastic finite automata". In: *Journal of Computer Systems Science* (1971).

[Car97] R. C. Carrasco. "Accurate computation of the relative entropy between stochastic regular grammars". In: *RAIRO (Theoretical Informatics and Applications)* (1997).

[CO94] R. C. Carrasco and J. Oncina. "Learning stochastic regular grammars by means of a state merging method". In: *International Colloquium on Grammatical Inference (ICGI)* (1994).

[CG08] J. Castro and R. Gavaldà. "Towards feasible PAC-learning of probabilistic deterministic finite automata". In: *International Colloquium on Grammatical Inference (ICGI)* (2008).

[CBL06] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[CL13] A. Chaganty and P. Liang. "Spectral experts for estimating mixtures of linear regressions". In: *International Conference on Machine Learning (ICML)* (2013).

[Cha96] J. T. Chang. "Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency". In: *Mathematical Biosciences* (1996).

[Cla09] A. Clark. Private Communication. 2009.

[CT04a] A. Clark and F. Thollard. "PAC-learnability of probabilistic deterministic finite state automata". In: *Journal of Machine Learning Research* (2004).

[CT04b] A. Clark and F. Thollard. "Partially distribution-free learning of regular languages from positive samples". In: *Conference on Computational Linguistics (COLING)* (2004).

[CC12] S. B. Cohen and M. Collins. "Tensor decomposition for fast parsing with latent-variable PCFGs". In: *Neural Information Processing Systems Conference (NIPS)* (2012).

[CSCFU12] S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. "Spectral learning of latent-variable PCFGs". In: *Annual Meeting of the Association for Computational Linguistics (ACL)* (2012).

[CSCFU13] S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. "Experiments with spectral learning of latent-variable PCFGs". In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HTL-NAACL)* (2013).

[DE08] F. Denis and Y. Esposito. "On rational stochastic languages". In: *Fundamenta Informaticae* (2008).

[DL01] L. Devroye and G. Lugosi. *Combinatorial methods in density estimation*. Springer, 2001.

[DRCFU12] P. S. Dhillon, J. Rodu, M. Collins, D. P. Foster, and L. H. Ungar. "Spectral dependency parsing with latent variables". In: *EMNLP-CoNLL* (2012).

[DRFU12] P. S. Dhillon, J. Rodu, D. P. Foster, and L. H. Ungar. "Using CCA to improve CCA: A new spectral method for estimating vector models of words". In: *International Conference on Machine Learning (ICML)* (2012).

[DP12] D.P. Dubhashi and A. Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2012.

[DPS04] M. Dudik, S. Phillips, and R. Schapire. "Performance guarantees for regularized maximum entropy density estimation". In: *Conference on Learning Theory (COLT)* (2004).

[DDE05] P. Dupont, F. Denis, and Y. Esposito. "Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms". In: *Pattern Recognition* (2005).

[Efr79] B. Efron. "Bootstrap methods: Another look at the jackknife". In: *The Annals of Statistics* (1979).

[EG02] L. El Ghaoui. "Inversion error, condition number, and approximate inverses of uncertain matrices". In: *Linear algebra and its applications* (2002).

[Faz02] M. Fazel. "Matrix rank minimization with applications". PhD thesis. Stanford University, 2002.

[Fli74] M. Fliess. "Matrices de Hankel". In: *Journal de Mathématiques Pures et Appliquées* (1974).

[FRU12] D. P. Foster, J. Rodu, and L. H. Ungar. "Spectral dimensionality reduction for HMMs". In: *CoRR* abs/1203.6130 (2012).

[FSSS11] R. Foygel, R. Salakhutdinov, O. Shamir, and N. Srebro. "Learning with the weighted trace-norm under arbitrary sampling distributions". In: *Neural Information Processing Systems Conference (NIPS)* (2011).

[Fro07] M. Fromont. "Model selection by bootstrap penalization for classification". In: *Machine Learning* (2007).

[FLLRB12] M. Fromont, B. Laurent, M. Lerasle, and P. Reynaud-Bouret. "Kernels based tests with non-asymptotic bootstrap approaches for two-sample problems". In: *Conference on Learning Theory (COLT)* (2012).

[Gam10] J. Gama. *Knowledge discovery from data streams*. Taylor and Francis, 2010.

[GV90] P. Garcia and E. Vidal. "Inference of k-testable languages in the strict sense and application to syntactic pattern recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1990).

[GKPP06] R. Gavaldà, P. W. Keller, J. Pineau, and D. Precup. "PAC-learning of Markov models with hidden state". In: *European Conference on Machine Learning (ECML)* (2006).

[Gol67] E.M. Gold. "Language identification in the limit". In: *Information and Control* (1967).

[GKS01] S. A. Goldman, S. S. Kwek, and S. D. Scott. "Agnostic learning of geometric patterns". In: *Journal of Computer and System Sciences* (2001).

[Gol06] O. Goldreich. "On promise problems: A survey". In: *Essays in Memory of Shimon Even* (2006).

[GBRSS12] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola. "A kernel two-sample test". In: *Journal of Machine Learning Research* (2012).

[GH98] D. Gross and C. M. Harris. *Fundamentals of queuing theory*. Wiley, 1998.

[GVW05] O. Guttman, S. V. N. Vishwanathan, and R. C. Williamson. "Learnability of probabilistic automata via oracles". In: *Conference on Algorithmic Learning Theory (ALT)* (2005).

[HMT11] N. Halko, P-G. Martinsson, and J. A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". In: *SIAM Review* (2011).

[Hal92] P. Hall. *The bootstrap and Edgeworth expansion*. Springer, 1992.

[Hau92] D. Haussler. "Decision theoretic generalizations of the PAC model for neural net and other learning applications". In: *Information and computation* (1992).

[Hig10] C. de la Higuera. *Grammatical inference: Learning automata and grammars*. Cambridge University Press, 2010.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. "Expander graphs and their applications". In: *Bulletin of the AMS* (2006).

[HKZ09] D. Hsu, S. M. Kakade, and T. Zhang. "A spectral algorithm for learning hidden Markov models". In: *Conference on Learning Theory (COLT)* (2009).

[Jae00] H. Jaeger. "Observable operator models for discrete stochastic time series". In: *Neural Computation* (2000).

[JZKOPK05] H. Jaeger, M. Zhao, K. Kretzschmar, T. Oberstein, D. Popovici, and A. Kolling. "Learning observable operator models via the ES algorithm". In: *New Directions in Statistical Signal Processing: From Systems to Brains*. The MIT Press, 2005.

[Jun09] R. Jungers. *The joint spectral radius: Theory and applications*. Springer, 2009.

[KSST12] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari. "Regularization techniques for learning with matrices". In: *Journal of Machine Learning Research* (2012).

[KKMS05] A. T. Kalai, A. R. Klivans, Y. Mansour, and R. A. Servedio. "Agnostically learning halfspaces". In: *IEEE Symposium on Foundations of Computer Science (FOCS)* (2005).

[KMV08] A. T. Kalai, Y. Mansour, and E. Verbin. "On agnostic boosting and parity learning". In: *Symposium on Theory of Computation (STOC)* (2008).

[KR99] M. Kearns and D. Ron. "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation". In: *Neural Computation* (1999).

[Kea98] M. J. Kearns. "Efficient noise-tolerant learning from statistical queries". In: *Journal of the ACM* (1998).

[KMRRSS94] M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. "On the learnability of discrete distributions". In: *Symposium on Theory of Computation (STOC)* (1994).

[KSS94] M. J. Kearns, R. E. Schapire, and L. M. Sellie. "Toward efficient agnostic learning". In: *Machine Learning* (1994).

[KV94] M. J. Kearns and L. G. Valiant. "Cryptographic limitations on learning boolean formulae and finite automata". In: *Journal of the ACM* (1994).

[KTSJ12] A. Kleiner, A. Talwalkar, P. Sarkar, and M.I. Jordan. "The big data bootstrap". In: *International Conference on Machine Learning (ICML)* (2012).

[Kol01] V. Koltchinskii. "Rademacher penalties and structural risk minimization". In: *IEEE Transactions on Information Theory* (2001).

[KNW13] A. Kontorovich, B. Nadler, and R. Weiss. "On learning parametric-output HMMs". In: *International Conference on Machine Learning (ICML)* (2013).

[KM93] E. Kushilevitz and Y. Mansour. "Learning decision trees using the Fourier spectrum". In: *SIAM Journal on Computing* (1993).

[Kut02] S. Kutin. *Extensions to McDiarmid's inequality when differences are bounded with high probability*. Tech. rep. TR-2002-04, University of Chicago, 2002.

[LPP98] K. Lang, B. Pearlmutter, and R. Price. "Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm". In: *International Colloquium on Grammatical Inference (ICGI)* (1998).

[LBW96] W. S. Lee, P. L. Bartlett, and R. C. Williamson. "Efficient agnostic learning of neural networks with bounded fan-in". In: *IEEE Transactions on Information Theory* (1996).

[Ler92] B. G. Leroux. "Maximum-likelihood estimation for hidden Markov models". In: *Stochastic Processes and Their Applications* (1992).

[LM99] C.K. Li and R. Mathias. "The Lidskii-Mirsky-Wielandt theorem: Additive and multiplicative versions". In: *Numerische Mathematik* (1999).

[LZ08] X. Lin and Y. Zhang. "Aggregate computation over data streams". In: *Asian-Pacific Web Conference (APWeb)* (2008).

[LSS01] M. L. Littman, R. S. Sutton, and S. Singh. "Predictive representations of state". In: *Neural Information Processing Systems Conference (NIPS)* (2001).

[LSY10] G. Liu, J. Sun, and S. Yan. "Closed-form solutions to a category of nuclear norm minimization problems". In: *NIPS Workshop on Low-Rank Methods for Large-Scale Machine Learning* (2010).

[LQBC12] F.M. Luque, A. Quattoni, B. Balle, and X. Carreras. "Spectral learning in non-deterministic dependency parsing". In: *Conference of the European Chapter of the Association for Computational Linguistics (EACL)* (2012).

[Maa94] W. Maass. "Efficient agnostic pac-learning with simple hypothesis". In: *Conference on Learning Theory (COLT)* (1994).

[MMMSW09] G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. "Efficient large-scale distributed training of conditional maximum entropy models". In: *Neural Information Processing Systems Conference (NIPS)* (2009).

[MAA05] A. Metwally, D. Agrawal, and A. Abbadi. "Efficient computation of frequent and top-k elements in data streams". In: *International Conference on Database Theory (ICDT)* (2005).

[Moh09] M. Mohri. "Weighted automata algorithms". In: *Handbook of Weighted Automata*. Springer, 2009.

[MRT12] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. The MIT Press, 2012.

[MR05] E. Mossel and S. Roch. "Learning nonsingular phylogenies and hidden Markov models". In: *Symposium on Theory of Computation (STOC)* (2005).

[Mut05] S. Muthukrishnan. "Data streams: algorithms and applications". In: *Foundations and Trends in Theoretical Computer Science* (2005).

[Nes07] Y. Nesterov. *Gradient methods for minimizing composite objective function*. Tech. rep. Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2007.

[Nor09] J. P. Norton. *An introduction to identification*. Dover, 2009.

[OGV93] J. Oncina, P. García, and E. Vidal. "Learning subsequential transducers for pattern recognition interpretation tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1993).

[OG92] J. Oncina and P. García. "Identifying regular languages in polynomial time". In: *Advances in Structural and Syntactic Pattern Recognition* (1992).

[PG07] N. Palmer and P. W. Goldberg. "PAC-learnability of probabilistic deterministic finite state automata in terms of variation distance". In: *Theoretical Computer Science* (2007).

[Pal08] N. J. Palmer. "Pattern classification via unsupervised learners". PhD thesis. University of Warwick, 2008.

[PSX11] A. P. Parikh, L. Song, and E.P. Xing. "A spectral algorithm for latent tree graphical models". In: *International Conference on Machine Learning (ICML)* (2011).

[Par87] E. Parzen. *Stochastic processes*. Society for Industrial Mathematics, 1987.

[Pav04] V. Pavlovic. "Model-based motion clustering using boosted mixture modeling". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2004).

[Pea94] K. Pearson. "Contributions to the mathematical theory of evolution". In: *Philosophical Transactions of the Royal Society of London* (1894).

[Pet69] T. Petrie. "Probabilistic functions of finite state Markov chains". In: *The Annals of Mathematical Statistics* (1969).

[PW93] L. Pitt and M. K. Warmuth. "The minimum consistent DFA problem cannot be approximated within any polynomial". In: *Journal of the ACM* (1993).

[Pol03] D. Pollard. *A user's guide to measure theoretic probability*. Cambridge University Press, 2003.

[Rec11] B. Recht. "A simpler approach to matrix completion". In: *Journal of Machine Learning Research* (2011).

[RST98] D. Ron, Y. Singer, and N. Tishby. "On the learnability and usage of acyclic probabilistic finite automata". In: *Journal of Computing Systems Science* (1998).

[SS78] A. Salomaa and M. Soittola. *Automata-theoretic aspects of formal power series*. Springer, 1978.

[SF12] R. E. Schapire and Y. Freund. *Boosting: Foundations and algorithms*. The MIT Press, 2012.

[SLRB11] M. Schmidt, N. Le Roux, and F. Bach. "Convergence rates of inexact proximal-gradient methods for convex optimization". In: *Neural Information Processing Systems Conference (NIPS)* (2011).

[Sch61] M. P. Schützenberger. "On the definition of a family of automata". In: *Information and Control* (1961).

[Sch78] G. Schwarz. "Estimating the dimension of a model". In: *The Annals of Statistics* (1978).

[SBG10] S.M. Siddiqi, B. Boots, and G.J. Gordon. "Reduced-rank hidden Markov models". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2010).

[SBSGS10] L. Song, B. Boots, S. Siddiqi, G. Gordon, and A. Smola. "Hilbert space embeddings of hidden Markov models". In: *International Conference on Machine Learning (ICML)* (2010).

[SS90] G. W. Stewart and J. Sun. *Matrix perturbation theory*. Academic Press, 1990.

[TB73] B. A. Trakhtenbrot and YA. M. Barzdin. *Finite automata : behavior and synthesis*. North-Holland Pub. Co, 1973.

[Val84] L. G. Valiant. "A theory of the learnable". In: *Communications of the ACM* (1984).

[VODM94] P. Van Overschee and B. De Moor. "N4SID: subspace algorithms for the identification of combined deterministic-stochastic systems". In: *Automatica* (1994).

[Vap99] V. Vapnik. *The nature of statistical learning theory*. Springer, 1999.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. "On the uniform convergence of relative frequencies of events to their probabilities". In: *Theory of Probability and Its Applications* (1971).

[Ver12] R. Vershynin. "Introduction to the non-asymptotic analysis of random matrices". In: *Compressed Sensing, Theory and Applications*. Cambridge University Press, 2012.

[VEH12] S. Verwer, R. Eyraud, and C. de la Higuera. "Pautomac: a PFA/HMM learning competition". In: *International Colloquium on Grammatical Inference (ICGI)* (2012).

[Ver10] S.E. Verwer. "Efficient identification of timed automata: theory and practice". PhD thesis. Delft University of Technology, 2010.

[VTHCC05a] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. "Probabilistic finite-state machines – Part I". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005).

[VTHCC05b] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. "Probabilistic finite-state machines – Part II". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005).

[XCM12] H. Xu, C. Caramanis, and S. Mannor. "Sparse algorithms are not stable: A no-free-lunch theorem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012).

[ZJ10] M-J. Zhao and H. Jaeger. "Norm-observable operator models". In: *Neural computation* (2010).

[ZJT09] M-J. Zhao, H. Jaeger, and M. Thon. "A bound on modeling error in observable operator models and an associated learning algorithm". In: *Neural Computation* (2009).

[ZB06] L. Zwald and G. Blanchard. "On the convergence of eigenspaces in kernel principal component analysis". In: *Neural Information Processing Systems Conference (NIPS)* (2006).

Appendix A

Mathematical Preliminaries

A.1 Probability

Concentration of measure is a well-known phenomena in high-dimensional probability spaces that has many applications in learning theory and the analysis of randomized algorithms. In this section we recall several concentration inequalities that are extensively used in this dissertation. Most of these results can be found on the excellent books [DP12; BLM13].

The most basic concentration bound is *Markov's inequality* which states that given a random variable X and any $t > 0$, then

$$\mathbb{P}[|X| \geq t] \leq \frac{\mathbb{E}[|X|]}{t} \ .$$

If information about the variance $\mathbb{V}[X]$ is available, one can use the so-called *Chebyshev's inequality*: for any $t > 0$

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathbb{V}[X]}{t^2} \ .$$

Sometimes the following one-sided version known as *Chebyshev–Cantelli inequality* is also useful: for any $t > 0$

$$\mathbb{P}[X - \mathbb{E}[X] \geq t] \leq \frac{\mathbb{V}[X]}{\mathbb{V}[X] + t^2} \ .$$

When the variable of interest X can be expressed as the sum of independent variables, then one can show that concentration around the mean happens exponentially fast. For $i \in [m]$ let X_i be independent random variables with $0 \leq X_i \leq 1$. Define $X = \sum_{i \in [m]} X_i$. Then, for any $0 < \gamma < 1$ the following *Chernoff bounds* hold:

$$\mathbb{P}\left[X > (1+\gamma)\mathbb{E}[X]\right] \leq \exp\left(-\frac{\gamma^2 \mathbb{E}[X]}{3}\right) \ ,$$
$$\mathbb{P}\left[X < (1-\gamma)\mathbb{E}[X]\right] \leq \exp\left(-\frac{\gamma^2 \mathbb{E}[X]}{2}\right) \ .$$

Suppose now that for each $i \in [m]$ we have $a_i \leq X_i \leq b_i$. Then we can use the *Hoeffding bounds* for any $t > 0$:

$$\mathbb{P}\left[X \geq \mathbb{E}[X] + t\right], \ \ \mathbb{P}\left[X \leq \mathbb{E}[X] - t\right] \leq \exp\left(-\frac{2t^2}{\sum_{i \in [m]} (b_i - a_i)^2}\right) \ .$$

The following is another useful form of these bounds:

$$\mathbb{P}\left[|X - \mathbb{E}[X]| \geq t\right] \leq 2\exp\left(-\frac{2t^2}{\sum_{i \in [m]} (b_i - a_i)^2}\right) \ .$$

See [DP12] for proofs and further results along these lines.

Chernoff and Hoeffding bounds are useful to bound the deviation of a *sum of scalar* independent random variables from its expectation. Often we find ourselves with the need to deal with more general functions of more general random variables. In particular, let X_i be Ω-valued independent random variables for $1 \leq i \leq m$ and write $Y = (X_1, \ldots, X_m) \in \Omega^m$. Suppose that $\Phi : \Omega^m \to \mathbb{R}$ is an arbitrary measurable function. Then we want to see how the random variable $\Phi(Y)$ concentrates around its expectation $\mathbb{E}[\Phi(Y)]$.

One way is to use the *Efron–Stein inequality* to bound the variance $\mathbb{V}[\Phi(Y)]$ as follows, and then apply Chebyshev's inequality. For each $i \in [m]$ let X_i' be an independent copy of X_i. Let us write $Y_i = (X_1, \ldots, X_{i-1}, X_i', X_{i+1}, \ldots, X_m)$ for the random vector in Ω^m obtained by replacing X_i with its copy X_i'. Then we have the following:

$$\mathbb{V}[\Phi(Y)] \leq \frac{1}{2} \sum_{i \in m} \mathbb{E}\left[(\Phi(Y) - \Phi(Y_i))^2\right] \quad .$$

If the change on the value of Φ can be controlled for any arbitrary change in one of its coordinates, then exponential concentration can be proved. So suppose that there exist constants c_i such that for each $i \in [m]$,

$$\sup_{x_1, \ldots, x_m, x_i' \in \Omega} |\Phi(x_1, \ldots, x_i, \ldots, x_m) - \Phi(x_1, \ldots, x_i', \ldots, x_m)| \leq c_i \quad .$$

Such a function Φ is said to satisfy the *bounded differences assumption*. In this case, *McDiarmid's inequality* says that for any $t > 0$ the following hold:

$$\begin{aligned} \mathbb{P}\left[\Phi(Y) \geq \mathbb{E}[\Phi(Y)] + t\right] &\leq \exp\left(-\frac{2t^2}{\sum_{i \in [m]} c_i^2}\right) \quad , \\ \mathbb{P}\left[\Phi(Y) \leq \mathbb{E}[\Phi(Y)] - t\right] &\leq \exp\left(-\frac{2t^2}{\sum_{i \in [m]} c_i^2}\right) \quad . \end{aligned} \tag{A.1}$$

The next results give useful variations on McDiarmid's inequality for dealing with functions that do not satisfy the bounded differences assumption almost surely. These and other similar results can be found in [Kut02]. This kind of inequalities have proved useful in many stability analyses.

Definition A.1.1. *Let $Y = (X_1, \ldots, X_m)$ be a random variable on a probability space Ω^m. We say that a function $\Phi : \Omega^m \to \mathbb{R}$ is* strongly difference-bounded by (b, c, δ) *if the following holds:*

1. *there exists a subset $E \subseteq \Omega^m$ with $\mathbb{P}[E] \leq \delta$, such that*
2. *if Y, Y' differ only on one coordinate and $Y \notin E$, then $|\Phi(Y) - \Phi(Y')| \leq c$, and*
3. *for all Y, Y' that differ only on one coordinate $|\Phi(Y) - \Phi(Y')| \leq b$.*

Theorem A.1.2. *Let Φ be function over a probability space Ω^m that is strongly difference-bounded by (b, c, δ) with $b \geq c > 0$. Then, for any $t > 0$,*

$$\mathbb{P}\left[\Phi(Y) - \mathbb{E}[\Phi(Y)] \geq t\right] \leq \exp\left(\frac{-t^2}{8mc^2}\right) + \frac{mb\delta}{c} \quad .$$

Furthermore, the same bound holds for $\mathbb{P}[\mathbb{E}[\Phi(Y)] - \Phi(Y) \geq t]$.

Corollary A.1.3. *Let Φ be function over a probability space Ω^m that is strongly difference-bounded by $(b, \theta/m, \exp(-Km))$. Then, for any $0 < t \leq 2\theta\sqrt{K}$ and $m \geq \max\{b/\theta, (9 + 18/K)\ln(3 + 6/K)\}$,*

$$\mathbb{P}\left[\Phi(Y) - \mathbb{E}[\Phi(Y)] \geq t\right] \leq 2\exp\left(\frac{-t^2 m}{8\theta^2}\right) \quad .$$

Furthermore, the same bound holds for $\mathbb{P}[\mathbb{E}[\Phi(Y)] - \Phi(Y) \geq t]$.

The following is another useful form of the previous corollary, in the form of deviation bounds instead of bounds on tail probabilities.

Corollary A.1.4. *Let Φ be a function over a probability space Ω^m that is strongly difference-bounded by $(b, \theta/m, \exp(-Km))$. Then, for any $\delta > 0$ and any sample size*

$$m \geq \max\left\{\frac{b}{\theta}, \left(9 + \frac{18}{K}\right)\ln\left(3 + \frac{6}{K}\right), \frac{2}{K}\ln\left(\frac{2}{\delta}\right)\right\} \ ,$$

each of the following holds with probability at least $1 - \delta$:

$$\Phi(Y) \geq \mathbb{E}[\Phi(Y)] - \sqrt{\frac{8\theta^2}{m}\ln\left(\frac{2}{\delta}\right)} \ ,$$
$$\Phi(Y) \leq \mathbb{E}[\Phi(Y)] + \sqrt{\frac{8\theta^2}{m}\ln\left(\frac{2}{\delta}\right)} \ .$$

A real random variable X is *sub-exponential* if there exists a constant $c > 0$ such that $\mathbb{P}[|X| \geq t] \leq \exp(-ct)$ holds for all $t \geq 0$. The following theorem gives an exponential concentration bound for the sum of sub-exponential random variables.

Theorem A.1.5 ([Ver12])**.** *Let $X_1, \ldots, X_m$ be i.i.d. sub-exponential random variables and write $Z = \sum_{i=1}^m X_i$. Then, for every $t \geq 0$, we have*

$$\mathbb{P}[Z - \mathbb{E}[Z] \geq mt] \leq \exp\left(-\frac{m}{8e^2}\min\left\{\frac{t^2}{4c^2}, \frac{t}{2c}\right\}\right) \ ,$$

where c is the sub-exponential constant of variables X_i.

A centered real random variable X is *sub-gamma* if there exist constants ν and c such that the following inequality holds for every $0 < s < 1/c$:

$$\log\max\left\{\mathbb{E}[e^{sX}], \mathbb{E}[e^{-sX}]\right\} \leq \frac{s^2\nu}{2(1-cs)} \ .$$

Then ν is called the *variance factor* of X and c is called the *scale parameter* of X. A usual example is that of *gamma random variables*. In particular, if given a random variable $Y \sim \text{Gamma}(k, \theta)$, then $X = Y - \mathbb{E}[Y]$ is sub-gamma with $\nu = k\theta^2$ and $c = \theta$. The following result gives a two-sided concentration bounds for sub-gamma random variables. See [BLM13] for further details.

Theorem A.1.6 ([BLM13])**.** *Let X be a centered sub-gamma random variable with variance factor ν and scale parameter c. Then the two following inequalities hold for every $t > 0$:*

$$\mathbb{P}[X > \sqrt{2\nu t} + ct] \leq \exp(-t) \ ,$$
$$\mathbb{P}[-X > \sqrt{2\nu t} + ct] \leq \exp(-t) \ .$$

Concentration bounds can also be obtained for sums of i.i.d. random matrices. In particular, the following result on the rank of empirical covariance matrices will prove useful.

Theorem A.1.7 ([Ver12])**.** *Consider a probability distribution in $\mathbb{R}^d$ with full-rank covariance matrix $\mathbf{C}$ and supported in a centered Euclidean ball of radius R. Take m i.i.d. examples from the distribution and let $\hat{\mathbf{C}}$ denote its sample covariance matrix. Then, if $m \geq C(\mathfrak{s}_1(\mathbf{C})/\mathfrak{s}_d(\mathbf{C})^2)R^2\log(1/\delta)$ the matrix $\hat{\mathbf{C}}$ has full rank with probability at least $1 - \delta$. Here C is a universal constant.*

A well-known application of concentration bounds is to prove uniform convergence bounds for large classes of functions [BBL04]. This is basically a "trick" to obtain non-vacuous union bounds over infinite classes of events by exploiting certain combinatorial dependences between the possible events. This is

also related to research in probability theory about the supremum of empirical processes. To formalize these bounds, we need the following definitions.

Let $\mathfrak{F}$ be a class of zero-one valued functions over some domain Ω. That is, $f \in \mathfrak{F}$ is a function $f : \Omega \to \{0, 1\}$ that can be interpreted as the indicator function of some event over Ω. Given a vector $z = (z_1, \ldots, z_m) \in \Omega^m$, for any $f \in \mathfrak{F}$ we define the vector $f(z) = (f(z_1), \ldots, f(z_m)) \in \{0, 1\}^m$. Then, the set

$$\mathfrak{F}_z = \{f(z) \,|\, f \in \mathfrak{F}\}$$

represents all the different possible ways in which the points of z can be classified by the functions in $\mathfrak{F}$. In particular, it is immediate to see that $|\mathfrak{F}_z| \leq 2^m$, and that if for example $z_i = z_j$, then the inequality is strict because we will always get $f(z_i) = f(z_j)$. The *growth function* of $\mathfrak{F}$ is defined as $\Pi_{\mathfrak{F}}(m) = \sup_{z \in \Omega^m} |\mathfrak{F}_z|$. Sometimes the values $\Pi_{\mathfrak{F}}(m)$ are also called the *shattering coefficients* of $\mathfrak{F}$. The idea is that if $\mathfrak{F}$ is not too complicated, then for some m large enough the function $\Pi_{\mathfrak{F}}(m)$ will grow slower that exponentially. This is formalized by Sauer's lemma, see [BBL04] for example.

The following results collects some useful properties of shattering coefficients. Recall that if $\mathfrak{E}$ and $\mathfrak{F}$ are two collections of events, their tensor product is defined as

$$\mathfrak{E} \otimes \mathfrak{F} = \{E \times F : E \in \mathfrak{E}, F \in \mathfrak{F}\} \ .$$

Theorem A.1.8. *Let $\mathfrak{E}$ and $\mathfrak{F}$ be two collections of events. The following hold for all $m \geq 1$:*

1. $\Pi_{\mathfrak{E} \cup \mathfrak{F}}(m) \leq \Pi_{\mathfrak{E}}(m) + \Pi_{\mathfrak{F}}(m)$,
2. $\Pi_{\mathfrak{E} \otimes \mathfrak{F}}(m) \leq \Pi_{\mathfrak{E}}(m) \cdot \Pi_{\mathfrak{F}}(m)$,
3. *if $\mathfrak{E} \subseteq \mathfrak{F}$, then* $\Pi_{\mathfrak{E}}(m) \leq \Pi_{\mathfrak{F}}(m)$.

Using these definitions we can now state the Vapnik–Chervonenkis (VC) inequality. Let D be a probability distribution over Ω. Given $f \in \mathfrak{F}$ we write $\mathbb{E}_D[f] = \mathbb{E}_{x \sim D}[f(x)]$. Furthermore, given a sample S of m i.i.d. examples from D, $S \sim D^m$, we write $\hat{\mathbb{E}}_S[f] = \sum_{x \in S} f(x)/m$ for the empirical estimation of $\mathbb{E}_D[f]$ based on S. Then the following holds for any $t > 0$:

$$\mathbb{P}_{S \sim D^m}\left[\sup_{f \in \mathfrak{F}} |\hat{\mathbb{E}}_S[f] - \mathbb{E}_D[f]| > t\right] \leq 4\Pi_{\mathcal{F}}(2m) \exp(-mt^2/8) \ . \tag{A.2}$$

Note that if $\mathfrak{F}$ is finite, a similar bound can be obtained by combining Hoeffding inequality with the union bound. However, the VC inequality works for infinite classes $\mathfrak{F}$ as well.

A.2 Linear Algebra

This appendix collects a set of perturbation results on linear algebra that are useful in the analysis of spectral learning algorithms. Bold letters will be used for vectors $\mathbf{v}$ and matrices $\mathbf{M}$. For vectors, $\|\mathbf{v}\|$ denotes the standard euclidean norm. For matrices, $\|\mathbf{M}\|$ denotes the operator norm. For $p \in [1, +\infty]$, $\|\mathbf{M}\|_p$ denotes the Schatten p-norm: $\|\mathbf{M}\|_p = (\sum_{n \geq 1} \mathfrak{s}_n^p(\mathbf{M}))^{1/p}$, where $\mathfrak{s}_n(\mathbf{M})$ is the nth singular value of $\mathbf{M}$. The special case $p = 2$ coincides with the Frobenius norm which will be sometimes also written as $\|\mathbf{M}\|_F$. The Moore–Penrose pseudo-inverse of a matrix $\mathbf{M}$ is denoted by $\mathbf{M}^+$.

Lemma A.2.1 ([SS90])**.** *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d_1 \times d_2}$. Then, for any $1 \leq n \leq \min\{d_1, d_2\}$, one has $|\mathfrak{s}_n(\mathbf{A}) - \mathfrak{s}_n(\mathbf{B})| \leq \|\mathbf{A} - \mathbf{B}\|$.*

Lemma A.2.2 ([EG02])**.** *Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ be an invertible matrix and suppose $\mathbf{B} = \mathbf{A} + \mathbf{E}$ with $\|\mathbf{E}\| \leq \mathfrak{s}_d(\mathbf{A})$. Then $\mathbf{B}$ is invertible and one has*

$$\|\mathbf{A}^{-1} - \mathbf{B}^{-1}\| \leq \frac{\|\mathbf{E}\|}{\mathfrak{s}_d(\mathbf{A})(\mathfrak{s}_d(\mathbf{A}) - \|\mathbf{E}\|)} \ .$$

Lemma A.2.3 ([SS90])**.** *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d_1 \times d_2}$. The following holds:*

$$\|\mathbf{A}^+ - \mathbf{B}^+\| \leq \frac{1+\sqrt{5}}{2} \cdot \max\left\{\|\mathbf{A}^+\|^2, \|\mathbf{B}^+\|^2\right\} \|\mathbf{A} - \mathbf{B}\| \ .$$

Lemma A.2.4 ([ZB06])**.** *Let $\mathbf{A} \in \mathbb{R}^{d\times d}$ be symmetric positive semidefinite matrix and $\mathbf{E} \in \mathbb{R}^{d\times d}$ a symmetric matrix such that $\mathbf{B} = \mathbf{A} + \mathbf{E}$ is positive semidefinite. Fix $n \leq \operatorname{rank}(\mathbf{A})$ and suppose that $\|\mathbf{E}\|_F \leq (\lambda_n(\mathbf{A}) - \lambda_{n+1}(\mathbf{A}))/4$. Then, writing $\mathbf{V}_n$ for the top n eigenvectors of $\mathbf{A}$ and $\mathbf{W}_n$ for the top n eigenvectors of $\mathbf{B}$, we have*

$$\|\mathbf{V}_n - \mathbf{W}_n\|_F \leq \frac{4\|\mathbf{E}\|_F}{\lambda_n(\mathbf{A}) - \lambda_{n+1}(\mathbf{A})} \ .$$

Corollary A.2.5. *Let $\mathbf{A}, \mathbf{E} \in \mathbb{R}^{d_1 \times d_2}$ and write $\mathbf{B} = \mathbf{A} + \mathbf{E}$. Suppose $n \leq \operatorname{rank}\mathbf{A}$ and $\|\mathbf{E}\|_F \leq \sqrt{\mathfrak{s}_n(\mathbf{A})^2 - \mathfrak{s}_{n+1}(\mathbf{A})^2}/4$. If $\mathbf{V}_n, \mathbf{W}_n$ contain the first n right singular vectors of $\mathbf{A}$ and $\mathbf{B}$ respectively, then*

$$\|\mathbf{V}_n - \mathbf{W}_n\|_F \leq \frac{8\|\mathbf{A}\|_F\|\mathbf{E}\|_F + 4\|\mathbf{E}\|_F^2}{\mathfrak{s}_n(\mathbf{A})^2 - \mathfrak{s}_{n+1}(\mathbf{A})^2} \ .$$

Proof. Using that $\|\mathbf{A}^\top\mathbf{A} - \mathbf{B}^\top\mathbf{B}\|_F \leq 2\|\mathbf{A}\|_F\|\mathbf{E}\|_F + \|\mathbf{E}\|_F^2$ and $\lambda_n(\mathbf{A}^\top\mathbf{A}) = \mathfrak{s}_n(\mathbf{A})^2$, we can apply Lemma A.2.4 to get the bound on $\|\mathbf{V}_n - \mathbf{W}_n\|_F$ under the condition that $\|\mathbf{A}^\top\mathbf{A} - \mathbf{B}^\top\mathbf{B}\|_F \leq (\mathfrak{s}_n(\mathbf{A})^2 - \mathfrak{s}_{n+1}(\mathbf{A})^2)/4$. To see that this last condition is satisfied, observe that using Lemma A.3.4 we have

$$\begin{aligned}
\|\mathbf{A} - \mathbf{B}\|_F &\leq \frac{\sqrt{\mathfrak{s}_n(\mathbf{A})^2 - \mathfrak{s}_{n+1}(\mathbf{A})^2}}{4} \\
&\leq \frac{\sqrt{\mathfrak{s}_n(\mathbf{A})^2 - \mathfrak{s}_{n+1}(\mathbf{A})^2} + \sqrt{4\|\mathbf{A}\|_F^2} - 2\|\mathbf{A}\|_F}{2\sqrt{1+\sqrt{2}}} \\
&\leq \frac{\sqrt{4\|\mathbf{A}\|_F^2 + \mathfrak{s}_n(\mathbf{A})^2 - \mathfrak{s}_{n+1}(\mathbf{A})^2} - 2\|\mathbf{A}\|_F}{2} \ ,
\end{aligned}$$

and this last inequality implies $2\|\mathbf{A}\|_F\|\mathbf{E}\|_F + \|\mathbf{E}\|_F^2 \leq (\mathfrak{s}_n(\mathbf{A})^2 - \mathfrak{s}_{n+1}(\mathbf{A})^2)/4$. □

Lemma A.2.6 (Corollary 2.4 in [LM99])**.** *Let $\mathbf{A} \in \mathbb{R}^{d_1\times d_2}$, $\mathbf{N} \in \mathbb{R}^{d_1\times d_1}$, and $\mathbf{M} \in \mathbb{R}^{d_2\times d_2}$. Then, for any $1 \leq i \leq \operatorname{rank}(\mathbf{A})$ one has*

$$\mathfrak{s}_{d_1}(\mathbf{N})\mathfrak{s}_{d_2}(\mathbf{M}) \leq \frac{\mathfrak{s}_i(\mathbf{NAM})}{\mathfrak{s}_i(\mathbf{A})} \leq \mathfrak{s}_1(\mathbf{N})\mathfrak{s}_1(\mathbf{M}) \ .$$

The following lemma was implicitly used in [HKZ09]. A full proof can be found in [SBG10].

Lemma A.2.7. *Let $\mathbf{A} \in \mathbb{R}^{d_1\times d_2}$ be a rank r matrix and write $\mathbf{V} \in \mathbb{R}^{d_2\times r}$ for its top r right singular vectors. Let $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{E}$ with $\|\mathbf{E}\| \leq \mathfrak{s}_r(\mathbf{A})$, and write $\tilde{\mathbf{V}}$ for the top r right singular vectors of $\tilde{\mathbf{A}}$. Then, for any $\mathbf{x} \in \mathbb{R}^r$ we have*

$$\|\mathbf{x}^\top\mathbf{V}^\top\tilde{\mathbf{V}}\| \geq \|\mathbf{x}\|\sqrt{1 - \frac{\|\mathbf{E}\|^2}{(\mathfrak{s}_r(\mathbf{A}) - \|\mathbf{E}\|)^2}} \ .$$

A.3 Calculus

This appendix contains a few technical lemmas based on basic calculus. We begin by stating a simple lemma which is commonly used when approximating conditional probabilities using statistical queries (see e.g. [AD98]).

Lemma A.3.1. *Let $0 \leq a, b, \gamma \leq 1$. If $|\hat{a} - a|, |\hat{b} - b| \leq b\gamma/3$, then $|\hat{a}/\hat{b} - a/b| \leq \gamma$.*

Note that, in order to obtain an absolute approximation of a quotient, a *relative* approximation of the denominator is needed. Though the lemma will be used when estimating fractions where the denominator is unknown, in general a lower bound on b is enough to obtain the desired approximation.

Lemma A.3.2. *For any $x, y, z > 0$, if $x \geq 3y\ln(yz)$ and $yz \geq 2.48$ then $x \geq y\ln(xz)$.*

Proof. By the concavity of logarithm it is enough to show that $x \geq y\ln(xz)$ holds for $x = x_0 = 3y\ln(yz)$; that is: $3y\ln(yz) \geq y\ln(3yz\ln(yz))$. This is equivalent to $2\ln(yz) \geq \ln(3\ln(yz))$. Since we have $yz \geq 2.48 > e^{e/3}$, the result follows by using again the concavity of logarithm. □

Lemma A.3.3. *For any $0 < \varepsilon \leq c < 1$ and $x > 0$, if $|\ln(x)| \leq c^{-1}\ln(1+c)\varepsilon$ then $|1-x| \leq \varepsilon$.*

Proof. First note that for any $0 < \varepsilon \leq c < 1$ we have

$$\min\left\{\ln(1+\varepsilon), \ln\left(\frac{1}{1-\varepsilon}\right)\right\} = \ln(1+\varepsilon) \geq \frac{\ln(1+c)}{c}\varepsilon$$

by the concavity of the logarithm. Now suppose $x \geq 1$ and note that by hypothesis we have $|\ln(x)| = \ln(x) \leq \ln(1+\varepsilon)$ which implies

$$|1-x| = x - 1 = \exp(\ln(x)) - 1 \leq \exp(\ln(1+\varepsilon)) - 1 = \varepsilon \ .$$

On the other hand, for $x < 1$ we have $-\ln(x) = |\ln(x)| \leq \ln(1/(1-\varepsilon)) = -\ln(1-\varepsilon)$, i.e. $\ln(x) \geq \ln(1-\varepsilon)$. Therefore we get

$$|1-x| = 1 - x = 1 - \exp(\ln(x)) \leq 1 - \exp(\ln(1-\varepsilon)) = \varepsilon \ . \qquad \square$$

Lemma A.3.4. *For any $x, y \geq 0$ we have $\sqrt{1+\sqrt{2}}\sqrt{x+y} \geq \sqrt{x} + \sqrt{y}$.*

Proof. By taking squares on each side of the inequality twice we see that the inequality holds if and only if $0 \leq 2(x^2 + y^2)$. □

A.4 Convex Analysis

Recall that a set $\Omega \subseteq X$ of some vector space X is *convex* if for any $x, y \in \Omega$ and any $0 \leq t \leq 1$ one has $tx + (t-1)y \in \Omega$. That is, every point in the segment between x and y also lies in Ω. A function $f : \Omega \to \mathbb{R}$ defined over a convex set is *convex* if for any $x, y \in \Omega$ and a any $0 \leq t \leq 1$ one has $f(tx + (t-1)y) \leq tf(x) + (t-1)f(y)$. That is, if the set of all points that lie above the graph of f is convex. The function f is said to be *strictly convex* if the inequality is always strict for any $0 < t < 1$ and $x \neq y$. From the point of view of optimization, the most appealing property of convex functions is that a local minimum for f is always a global minimum. Furthermore, if f is strictly convex, then it has at most one global minimum.

If a convex function f is smooth, then the regular characterization of local minima based on the gradient of f works: x is a local minima if and only if $\nabla f(x) = 0$. When f is not smooth, one needs to resort the subdifferential of f. If X is equipped with an inner product, then $v \in X$ is a *subgradient* of f at $x \in \Omega$ if for any $y \in \Omega$ one has $f(y) - f(x) \geq \langle v, y - x\rangle$. An arbitrary subgradient of f at x is sometimes denoted by $\delta f(x)$. The set of all such subgradients is the *subdifferential* $\partial f(x)$ of f at x. With these definitions we can see that x is a local minima of f if and only if $0 \in \partial f(x)$.

Let $\mathbb{H}$ be a vector of real functions defined over some convex domain Ω, and let $\langle \cdot, \cdots \rangle$ be an inner product in $\mathbb{H}$. Suppose that $F : \mathbb{H} \to \mathbb{R}$ is a convex functional over $\mathbb{H}$. The *Bregman divergence* B_F of F on $f, g \in \mathbb{H}$ is defined as

$$B_F(f\|g) = F(f) - F(g) - \langle f - g, \delta F(g)\rangle \ ,$$

for some arbitrary subgradient $\delta F(g)$. Note that $B_F(f\|g)$ is basically the difference between $F(f)$ and the first-order Taylor expansion of F on g evaluated at f. A useful property of the Bregman divergence is that if $F = F_1 + \tau F_2$ is the sum of two convex functionals with $\tau > 0$, then by the "linearity" of subdifferentials one can always choose subgradients such that $B_F = B_{F_1} + \tau B_{F_2}$.

A.5 Properties of L_∞ and $\mathrm{L}_\infty^{\mathrm{p}}$

This appendix proves technical results about the L_∞ and $\mathrm{L}_\infty^{\mathrm{p}}$ distances used for testing distinguishability between states in a PDFA. If D and D' are probability distributions over a free monoid $\Sigma^\star$, these distances are defined as follows:

$$\begin{aligned}\mathrm{L}_\infty(D, D') &= \sup_{x \in \Sigma^\star} |D(x) - D'(x)| \quad , \\ \mathrm{L}_\infty^{\mathrm{p}}(D, D') &= \sup_{x \in \Sigma^\star} |D(x\Sigma^\star) - D'(x\Sigma^\star)| \ .\end{aligned}$$

In what follows we derive bounds between L_∞ and $\mathrm{L}_\infty^{\mathrm{p}}$ that justify our preference for $\mathrm{L}_\infty^{\mathrm{p}}$ as a distinguishability measure between states in PDFA. Furthermore, we show how to compute the shattering coefficients needed to bound the accuracy of empirical estimates of L_∞ and $\mathrm{L}_\infty^{\mathrm{p}}$ distances using VC bounds.

A.5.1 Bounds

It is obvious from the definition that $\mathrm{L}_\infty^{\mathrm{p}}$ is non-negative, symmetric, and satisfies the triangle inequality. In addition, $D = D'$ clearly implies $\mathrm{L}_\infty^{\mathrm{p}}(D, D') = 0$. To verify that $\mathrm{L}_\infty^{\mathrm{p}}$ is indeed a metric one needs to establish the converse to this last fact; we do so using the following result which gives a lower bound for $\mathrm{L}_\infty^{\mathrm{p}}(D, D')$ in terms of $\mathrm{L}_\infty(D, D')$. Since L_∞ is a metric, the bound implies that $\mathrm{L}_\infty^{\mathrm{p}}(D, D')$ is a metric too.

Proposition A.5.1. *For any two distributions D and D' over $\Sigma^\star$ we have*

$$\mathrm{L}_\infty(D, D') \leq (2|\Sigma| + 1)\, \mathrm{L}_\infty^{\mathrm{p}}(D, D') \ .$$

Proof. The inequality is obvious if $D = D'$. Thus, suppose $D \neq D'$ and let $x \in \Sigma^\star$ be such that $0 < \mathrm{L}_\infty(D, D') = |D(x) - D'(x)|$. Note that for any partition $x = u \cdot v$ we can write $D(x) = D^u(v) D(u\Sigma^\star)$. In particular, taking $v = \lambda$, the triangle inequality and $D'^x(\lambda) \leq 1$ yield the following:

$$\mathrm{L}_\infty(D, D') = |D(x) - D'(x)| \leq |D(x\Sigma^\star) - D'(x\Sigma^\star)| + D(x\Sigma^\star) |D^x(\lambda) - D'^x(\lambda)| \ .$$

Note that $|D(x\Sigma^\star) - D'(x\Sigma^\star)| \ \leq \mathrm{L}_\infty^{\mathrm{p}}(D, D')$. It remains to show that we also have $D(x\Sigma^\star)|D^x(\lambda) - D'^x(\lambda)| \leq 2|\Sigma|\, \mathrm{L}_\infty^{\mathrm{p}}(D, D')$. Observe the following, which is just a consequence of $D(\lambda) + \sum_\sigma D(\sigma\Sigma^\star) = 1$ for any distribution over $\Sigma^\star$:

$$D^x(\lambda) + \sum_\sigma D^x(\sigma\Sigma^\star) = D'^x(\lambda) + \sum_\sigma D'^x(\sigma\Sigma^\star) = 1 \ .$$

From these equations it is easy to show that there must exist a $\sigma \in \Sigma$ such that

$$|D^x(\lambda) - D'^x(\lambda)| \leq |\Sigma| |D^x(\sigma\Sigma^\star) - D'^x(\sigma\Sigma^\star)| \ .$$

Therefore, using $D(x\Sigma^\star) D^x(\sigma\Sigma^\star) = D(x\sigma\Sigma^\star)$ we obtain

$$\begin{aligned} D(x\Sigma^\star)|D^x(\lambda) - D'^x(\lambda)| &\leq |\Sigma| D(x\Sigma^\star)|D^x(\sigma\Sigma^\star) - D'^x(\sigma\Sigma^\star)| \\ &\leq |\Sigma| |D(x\sigma\Sigma^\star) - D'(x\sigma\Sigma^\star)| \\ &\quad + |\Sigma| D'^x(\sigma\Sigma^\star) |D(x\Sigma^\star) - D'(x\Sigma^\star)| \\ &\leq 2|\Sigma| \ \mathrm{L}_\infty^{\mathrm{p}}(D, D') \ . \end{aligned}$$

□

Recall that given a PDFA A we denote by f_A the probability distribution it defines. Furthermore, if $q \in Q$ is a state of A, then A_q represents the PDFA A where now q is the new starting state. Then the L_∞-distinguishability of A is given by

$$\mu_{\mathrm{L}_\infty} = \min_{q, q' \in Q} \mathrm{L}_\infty(f_{A_q}, f_{A_{q'}}) \ ,$$

where the minimum is taken over all pairs of distinct states in Q. A $\mathrm{L}_\infty^\mathrm{p}$-distinguishability $\mu_{\mathrm{L}_\infty^\mathrm{p}}$ is defined similarly. The bound given above implies that in general one has

$$\frac{\mu_{\mathrm{L}_\infty}}{2|\Sigma| + 1} \leq \mu_{\mathrm{L}_\infty^\mathrm{p}} \ .$$

Since the distinguishability of a PDFA quantifies how hard it is to learn its distributions by a state-merging method – with smaller distinguishability meaning harder to learn – this implies that using $\mathrm{L}_\infty^\mathrm{p}$ as a distinguishability measure one may in some cases slightly increase the hardness of learning a particular PDFA. However, this is not the case in general. And it turns out that $\mathrm{L}_\infty^\mathrm{p}$-distinguishabilities yield exponential advantages in some cases like the following example.

Recall the class $\mathcal{D}_n$ of PDFA that define probability distributions with support defined by parities of n bits as defined in Section 1.1. Let $T \in \mathcal{D}_{n/2}$ be an acyclic PDFA of $n+2$ states $Q = \{q_0, \ldots, q_{n+1}\}$ over $\Sigma = \{0, 1\}$ with a single stopping state q_{n+1}. Note that for $0 \leq i \leq n$ we have $\gamma_T(q_i, 0) = \gamma_T(q_i, 1) = 1/2$. Now let T' be the following perturbation of T: for $0 \leq i \leq n$ let $\gamma'(q'_i, 0) = 1/2 - i/(2n)$ and $\gamma'(q'_i, 1) = 1/2 + i/(2n)$. Then it is easy to check that T' has $\mu_{\mathrm{L}_\infty} = (1/2)^{\Theta(n)}$ and $\mu_{\mathrm{L}_\infty^\mathrm{p}} = \Theta(1/n)$.

A.5.2 Shattering Coefficients

Since we will be using uniform convergence bounds to estimate L_∞ and $\mathrm{L}_\infty^\mathrm{p}$ distances from samples, we will need to compute the corresponding growth functions. Suppose $\Omega = \Sigma^\star$ and let $\mathfrak{S} = \{\mathbb{1}_x \,|\, x \in \Sigma^\star\}$. Then it is obvious by construction that one has $\Pi_\mathfrak{S}(m) = m + 1$, which is the growth function needed for applying VC bounds to L_∞ estimation. The case of $\mathrm{L}_\infty^\mathrm{p}$ is slightly more complicated as can be seen in the following lemma.

Lemma A.5.2. *Let* $\mathfrak{P} = \{\mathbb{1}_{x\Sigma^\star} \,|\, x \in \Sigma^\star\}$. *Then* $\Pi_\mathfrak{P}(m) = 2m$.

Proof. In the first place note that for any $z \in (\Sigma^\star)^m$ the quantity $|\mathfrak{P}_z|$ is invariant under a permutation of the strings in z. Thus, without loss of generality we assume that the strings $z = (z^1, \ldots, z^m)$ are all different and are given in lexicographical order: $z^1 < \ldots < z^m$. Now let us define the prefixes $w^i = \mathrm{lcp}(z^1, z^i)$ for $2 \leq i \leq m$. By the ordering assumption on the z^i it is easy to see that $w^j \sqsubseteq_0 w^i$ for any $i < j$. Furthermore, we have $w^j \sqsubseteq_0 z^i$ for any $i \leq j$. Furthermore, let w be any string such that $\mathbb{1}_w(z^i) = 0$ for all $i \in [m]$. We define the set of string $Z = \{z^1, \ldots, z^m, w^2, \ldots, w^m, w\}$. We assume without loss of generality that $|Z| = 2m$. We also define the set of functions $\mathfrak{Z} = \{\mathbb{1}_x \,|\, x \in Z\}$. Now we will show that $\mathfrak{P}_z = \mathfrak{Z}_z$ and $|\mathfrak{Z}_z| = 2m$, from which it will follow that $\Pi_\mathfrak{P}(m) = 2m$.

For any $u, v \in \Sigma^\star$ such that $u \sqsubseteq_0 v$ we define the following set of strings

$$(u, v] = \{u' \,|\, u' \sqsubseteq_0 v \wedge |u| < |u'|\}$$

containing all the prefixes of v longer that u. The closed analog $[u, v]$ is defined likewise. Now we consider the following partition of $\Sigma^\star$:

$$\Sigma^\star = \bigsqcup_{i=2}^{m-1} (w^{i+1}, w^i] \sqcup [\lambda, w^m] \sqcup \bigsqcup_{i=2}^{m} (w^i, z^i] \sqcup (w^2, z^1] \sqcup W \ ,$$

where W is defined to be the complement of the rest. See Figure A.1 for an example of how this partition covers the subtree of $\Sigma^\star$ containing Z. To see that $\mathfrak{P}_z = \mathfrak{Z}_z$ holds, one just needs to verify that Z contains exactly one string in each set in the partition, and that:

$$\begin{aligned}
\forall x \in (w^{i+1}, w^i] \quad & \mathbb{1}_{x\Sigma^\star}^{-1}(1) \cap z = \{z^1, \ldots, z^i\} \\
\forall x \in [\lambda, w^m] \quad & \mathbb{1}_{x\Sigma^\star}^{-1}(1) \cap z = \{z^1, \ldots, z^m\} \\
\forall x \in (w^i, z^i] \quad & \mathbb{1}_{x\Sigma^\star}^{-1}(1) \cap z = \{z^i\} \\
\forall x \in (w^2, z^1] \quad & \mathbb{1}_{x\Sigma^\star}^{-1}(1) \cap z = \{z^1\} \\
\forall x \in W \quad & \mathbb{1}_{x\Sigma^\star}^{-1}(1) \cap z = \varnothing \ .
\end{aligned}$$

From these assertions also follows that $|\mathfrak{Z}_z| = 2m$. □

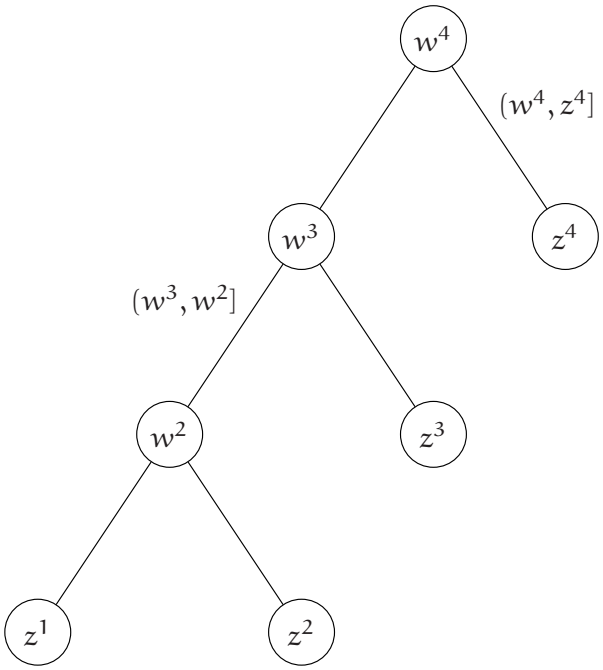


Figure A.1: Setup in the proof of Lemma A.5.2 with $m = 4$. Subtree of $\Sigma^\star$ containing $\{z^1, \ldots, z^4, w^2, \ldots, w^4\}$