Learning Shuffle Ideals Under Restricted Distributions

Dongqu Chen
Department of Computer Science
Yale University
dongqu.chen@yale.edu

Abstract

The class of shuffle ideals is a fundamental sub-family of regular languages. The shuffle ideal generated by a string set U is the collection of all strings containing some string $u \in U$ as a (not necessarily contiguous) subsequence. In spite of its apparent simplicity, the problem of learning a shuffle ideal from given data is known to be computationally intractable. In this paper, we study the PAC learnability of shuffle ideals and present positive results on this learning problem under element-wise independent and identical distributions and Markovian distributions in the statistical query model. A constrained generalization to learning shuffle ideals under product distributions is also provided. In the empirical direction, we propose a heuristic algorithm for learning shuffle ideals from given labeled strings under general unrestricted distributions. Experiments demonstrate the advantage for both efficiency and accuracy of our algorithm.

1 Introduction

The learnablity of regular languages is a classic topic in computational learning theory. The applications of this learning problem include natural language processing (speech recognition, morphological analysis), computational linguistics, robotics and control systems, computational biology (phylogeny, structural pattern recognition), data mining, time series and music ([7, 16, 18, 19, 20, 21, 23, 25]). Exploring the learnability of the family of formal languages is significant to both theoretical and applied realms. In the classic PAC learning model defined by Valiant [26], unfortunately, the class of regular languages, or equivalently the concept class of deterministic finite automata (DFA), is known to be inherently unpredictable ([1, 9, 22]). In a modified version of Valiant's model which allows the learner to make membership queries, Angluin [2] has shown that the concept class of regular languages is PAC learnable.

Throughout this paper we study the *PAC learnability* of a fundamental subclass of regular languages, the class of *(extended) shuffle ideals*. The shuffle ideal generated by an augmented string U is the collection of all strings containing some $u \in U$ as a (not necessarily contiguous) subsequence, where an *augmented string* is a finite concatenation of symbol sets (see Figure 1 for an illustration). The special class of shuffle ideals generated by a single string is called the *principal* shuffle ideals. In spite of its simplicity, the class of shuffle ideals plays a prominent role in formal language theory. The boolean closure of shuffle ideals is the important language family known as piecewise-testable languages ([24]). The rich structure of this language family has made it an object of intensive study in complexity theory and group theory ([12, 17]). In the applied direction, Kontorovich et al. [15] show the shuffle ideals capture some rudimentary phenomena in human language morphology.

Unfortunately, even such a simple class is not PAC learnable, unless RP=NP ([3]). However, in most application scenarios, the strings are drawn from some particular distribution we are interested in. Angluin et al. [3] prove under the uniform string distribution, principal shuffle ideals are PAC learnable. Nevertheless, the requirement of complete knowledge of the distribution, the dependence

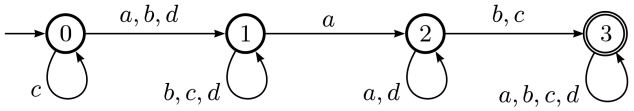

Figure 1: The DFA accepting precisely the shuffle ideal of $U = (a|b|d)a(b|c)$ over $\Sigma = \{a, b, c, d\}$.

on the symmetry of the uniform distribution and the restriction of principal shuffle ideals lead to the lack of generality of the algorithm. Our main contribution in this paper is to present positive results on learning the class of shuffle ideals under element-wise independent and identical distributions and Markovian distributions. Extensions of our main results include a constrained generalization to learning shuffle ideals under product distributions and a heuristic method for learning principal shuffle ideals under general unrestricted distributions.

After introducing the preliminaries in Section 2, we present our main result in Section 3: the extended class of shuffle ideals is PAC learnable from element-wise i.i.d. strings. That is, the distributions of the symbols in a string are identical and independent of each other. A constrained generalization to learning shuffle ideals under product distributions is also provided. In Section 4, we further show the PAC learnability of principal shuffle ideals when the example strings drawn from $\Sigma^{\leq n}$ are generated by a Markov chain with some lower bound assumptions on the transition matrix. In Section 5, we propose a greedy algorithm for learning principal shuffle ideals under general unrestricted distributions. Experiments demonstrate the advantage for both efficiency and accuracy of our heuristic algorithm.

2 Preliminaries

We consider strings over a fixed finite alphabet Σ. The empty string is λ. Let Σ^* be the Kleene star of Σ and $\Sigma^\cup$ be the collection of all subsets of Σ. As strings are concatenations of symbols, we similarly define augmented strings as concatenations of unions of symbols.

Definition 1 (Alphabet, simple string and augmented string) *Let Σ be a non-empty finite set of symbols, called the alphabet. A simple string over Σ is any finite sequence of symbols from Σ, and Σ^* is the collection of all simple strings. An augmented string over Σ is any finite concatenation of symbol sets from $\Sigma^\cup$, and $(\Sigma^\cup)^*$ is the collection of all augmented strings.*

Denote by s the cardinality of Σ. Because an augmented string only contains strings of the same length, the length of an augmented string U, denoted by $|U|$, is the length of any $u \in U$. We use exponential notation for repeated concatenation of a string with itself, that is, v^k is the concatenation of k copies of string v. Starting from index 1, we denote by v_i the i-th symbol in string v and use notation $v[i, j] = v_i \dots v_j$ for $1 \leq i \leq j \leq |v|$. Define the binary relation $\sqsubseteq$ on $\langle (\Sigma^\cup)^*, \Sigma^* \rangle$ as follows. For a simple string w, $w \sqsubseteq v$ holds if and only if there is a witness $\vec{i} = (i_1 < i_2 < \dots < i_{|w|})$ such that $v_{i_j} = w_j$ for all integers $1 \leq j \leq |w|$. For an augmented string W, $W \sqsubseteq v$ if and only if there exists some $w \in W$ such that $w \sqsubseteq v$. When there are several witnesses for $W \sqsubseteq v$, we may order them coordinate-wise, referring to the unique minimal element as the leftmost embedding. We will write $I_{W \sqsubseteq v}$ to denote the position of the last symbol of W in its leftmost embedding in v (if the latter exists; otherwise, $I_{W \sqsubseteq v} = \infty$).

Definition 2 (Extended/Principal Shuffle Ideal) *The (extended) shuffle ideal of an augmented string $U \in (\Sigma^\cup)^L$ is a regular language defined as $\text{Ш}(U) = \{v \in \Sigma^* \mid \exists u \in U, u \sqsubseteq v\} = \Sigma^* U_1 \Sigma^* U_2 \Sigma^* \dots \Sigma^* U_L \Sigma^*$. A shuffle ideal is principal if it is generated by a simple string.*

Shuffle ideal is an order ideal on monoid $\langle \Sigma^*, \cdot, \lambda \rangle$ and was originally defined for lattices. Denote by $\sqcup\!\sqcup$ the class of principal shuffle ideals and by Ш the class of extended shuffle ideals. Unless otherwise stated, in this paper shuffle ideal refers to the extended ideal. An example is given in Figure 1. The feasibility of determining whether a string is in the class $\text{Ш}(U)$ is obvious.

Lemma 1 *Evaluating relation $U \sqsubseteq x$ and meanwhile determining $I_{U \sqsubseteq x}$ is feasible in time $O(|x|)$.*

In a computational learning model, an algorithm is usually given access to an oracle providing information about the sample. In Valiant's work [26], the example oracle $EX(c, \mathcal{D})$ was defined, where c is the target concept and $\mathcal{D}$ is a distribution over the instance space. On each call, $EX(c, \mathcal{D})$ draws an input x independently at random from the instance space $\mathcal{I}$ under the distribution $\mathcal{D}$, and returns the labeled example $\langle x, c(x)\rangle$.

Definition 3 (PAC Learnability: [26]) *Let $\mathcal{C}$ be a concept class over the instance space $\mathcal{I}$. We say $\mathcal{C}$ is probably approximately correctly (PAC) learnable if there exists an algorithm $\mathcal{A}$ with the following property: for every concept $c \in \mathcal{C}$, for every distribution $\mathcal{D}$ on $\mathcal{I}$, and for all $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, if $\mathcal{A}$ is given access to $EX(c, \mathcal{D})$ on $\mathcal{I}$ and inputs ϵ and δ, then with probability at least $1 - \delta$, $\mathcal{A}$ outputs a hypothesis $h \in \mathcal{H}$ satisfying $\Pr_{x\in\mathcal{D}}[c(x) \neq h(x)] \leq \epsilon$. If $\mathcal{A}$ runs in time polynomial in $1/\epsilon$, $1/\delta$ and the representation size of c, we say that $\mathcal{C}$ is efficiently PAC learnable.*

We refer to ϵ as the error parameter and δ as the confidence parameter. If the error parameter is set to 0, the learning is exact ([6]). Kearns [11] extended Valiant's model and introduced the statistical query oracle *STAT*$(c, \mathcal{D})$. Kearns' oracle takes as input a statistical query of the form (χ, τ). Here χ is any mapping of a labeled example to $\{0, 1\}$ and $\tau \in [0, 1]$ is called the noise tolerance. *STAT*$(c, \mathcal{D})$ returns an estimate for the expectation $\mathbf{E}\chi$, that is, the probability that $\chi = 1$ when the labeled example is drawn according to $\mathcal{D}$. A statistical query can have a condition so $\mathbf{E}\chi$ can be a conditional probability. This estimate is accurate within additive error τ.

Definition 4 (Legitimacy and Feasibility: [11]) *A statistical query χ is legimate and feasible if and only if with respect to $1/\epsilon$, $1/\tau$ and representation size of c:*

1. *Query χ maps a labeled example $\langle x, c(x)\rangle$ to $\{0, 1\}$;*
2. *Query χ can be efficiently evaluated in polynomial time;*
3. *The condition of χ, if any, can be efficiently evaluated in polynomial time;*
4. *The probability of the condition of χ, if any, should be at least polynomially large.*

Throughout this paper, the learnability of shuffle ideals is studied in the statistical query model. Kearns [11] proves that oracle *STAT*$(c, \mathcal{D})$ is weaker than oracle $EX(c, \mathcal{D})$. In words, if a concept class is PAC learnable from *STAT*$(c, \mathcal{D})$, then it is PAC learnable from $EX(c, \mathcal{D})$, but not necessarily vice versa.

3 Learning shuffle ideals from element-wise i.i.d. strings

Although learning the class of shuffle ideals has been proved hard, in most scenarios the string distribution is restricted or even known. A very usual situation in practice is that we have some prior knowledge of the unknown distribution. One common example is the string distributions where each symbol in a string is generated independently and identically from an unknown distribution. It is element-wise i.i.d. because we view a string as a vector of symbols. This case is general enough to cover some popular distributions in applications such as the uniform distribution and the multinomial distribution. In this section, we present as our main result a statistical query algorithm for learning the concept class of extended shuffle ideals from element-wise i.i.d. strings and provide theoretical guarantees of its computational efficiency and accuracy in the statistical query model. The instance space is Σ^n. Denote by U the augmented pattern string that generates the target shuffle ideal and by $L = |U|$ the length of U.

3.1 Statistical query algorithm

Before presenting the algorithm, we define function $\theta_{V,a}(\cdot)$ and query $\chi_{V,a}(\cdot, \cdot)$ for any augmented string $V \in (\Sigma^{\cup})^{\leq n}$ and any symbol $a \in \Sigma$ as as follows.

$$\theta_{V,a}(x) = \begin{cases} a & \text{if } V \not\sqsubseteq x[1, n-1] \\ x_{I_{V\sqsubseteq x}+1} & \text{if } V \sqsubseteq x[1, n-1] \end{cases}$$

$$\chi_{V,a}(x, y) = \frac{1}{2}(y + 1) \quad \text{given } \theta_{V,a}(x) = a$$

where $y = c(x)$ is the label of example string x. More precisely, $y = +1$ if $x \in \text{Ш}(U)$ and $y = -1$ otherwise. Our learning algorithm uses statistical queries to recover string $U \in (\Sigma^{\cup})^L$ one element at a time. It starts with the empty string $V = \lambda$. Having recovered $V = U[1, \ell]$ where $0 \le \ell < L$, we infer $U_{\ell+1}$ as follows. For each $a \in \Sigma$, the statistical query oracle is called with the query $\chi_{V,a}$ at the error tolerance τ claimed in Theorem 1. Our key technical observation is that the value of $\mathbf{E}\chi_{V,a}$ effectively selects $U_{\ell+1}$. The query results of $\chi_{V,a}$ will form two separate clusters such that the maximum difference (variance) inside one cluster is smaller than the minimum difference (gap) between the two clusters, making them distinguishable. The set of symbols in the cluster with larger query results is proved to be $U_{\ell+1}$. Notice that this statistical query only works for $0 \le \ell < L$. To complete the algorithm, we address the trivial case $\ell = L$ with query $\Pr[y = +1 \mid V \sqsubseteq x]$ and the algorithm halts if the query answer is close to 1.

3.2 PAC learnability of ideal Ш

We show the algorithm described above learns the class of shuffle ideals from element-wise i.i.d. strings in the statistical query learning model.

Theorem 1 *Under element-wise independent and identical distributions over instance space $\mathcal{I} = \Sigma^n$, concept class Ш is approximately identifiable with $O(sn)$ conditional statistical queries from STAT$(\text{Ш}, \mathcal{D})$ at tolerance*

$$\tau = \frac{\epsilon^2}{40sn^2 + 4\epsilon}$$

or with $O(sn)$ statistical queries from STAT$(\text{Ш}, \mathcal{D})$ at tolerance

$$\bar{\tau} = \left(1 - \frac{\epsilon}{20sn^2 + 2\epsilon}\right) \frac{\epsilon^4}{16sn(10sn^2 + \epsilon)}$$

We provide the main idea of the proofs in this section and defer the details and algebra to Appendix A. The proof starts from the legitimacy and feasibility of the algorithm. Since $\chi_{V,a}$ computes a binary mapping from labeled examples to $\{0, 1\}$, the legitimacy is trivial. But $\chi_{V,a}$ is not feasible for symbols in Σ of small occurrence probabilities. We avoid the problematic cases by reducing the original learning problem to the same problem with a polynomial lower bound assumption $\Pr[x_i = a] \ge \epsilon/(2sn) - \epsilon^2/(20sn^2 + 2\epsilon)$ for any $a \in \Sigma$ and achieve feasibility.

The correctness of the algorithm is based on the intuition that the query result $\mathbf{E}\chi_{V,a_+}$ of a symbol $a_+ \in U_{\ell+1}$ should be greater than that of a symbol $a_- \notin U_{\ell+1}$ and the difference is large enough to tolerate the noise from the oracle. To prove this, we first consider the exact learning case. Define an infinite string $U' = U[1, \ell]U[\ell + 2, L]U_{\ell+1}^{\infty}$ and let $x' = x\Sigma^{\infty}$ be the extension of x obtained by padding it on the right with an infinite string generated from the same distribution as x. Let $Q(j, i)$ be the probability that the largest g such that $U'[1, g] \sqsubseteq x'[1, i]$ is j, or formally

$$Q(j, i) = \Pr[U'[1, j] \sqsubseteq x'[1, i] \wedge U'[1, j + 1] \not\sqsubseteq x'[1, i]]$$

By taking the difference between $\mathbf{E}\chi_{V,a_+}$ and $\mathbf{E}\chi_{V,a_-}$ in terms of $Q(j, i)$, we get the query tolerance for exact learning.

Lemma 2 *Under element-wise independent and identical distributions over instance space $\mathcal{I} = \Sigma^n$, concept class Ш is exactly identifiable with $O(sn)$ conditional statistical queries from STAT$(\text{Ш}, \mathcal{D})$ at tolerance*

$$\tau' = \frac{1}{5}Q(L - 1, n - 1)$$

Lemma 2 indicates bounding the quantity $Q(L - 1, n - 1)$ is the key to the tolerance for PAC learning. Unfortunately, the distribution $\{Q(j, i)\}$ doesn't seem to have any strong properties we know of providing a polynomial lower bound. Instead we introduce new quantity

$$R(j, i) = \Pr[U'[1, j] \sqsubseteq x'[1, i] \wedge U'[1, j] \not\sqsubseteq x'[1, i - 1]]$$

being the probability that the smallest g such that $U'[1, j] \sqsubseteq x'[1, g]$ is i. An important property of distribution $\{R(j, i)\}$ is its strong unimodality as defined below.

Definition 5 (Unimodality: [8]) *A distribution $\{P(i)\}$ with all support on the lattice of integers is unimodal if and only if there exists at least one integer K such that $P(i) \geq P(i-1)$ for all $i \leq K$ and $P(i+1) \leq P(i)$ for all $i \geq K$. We say K is a mode of distribution $\{P(i)\}$.*

Throughout this paper, when referring to the mode of a distribution, we mean the one with the largest index, if the distribution has multiple modes with equal probabilities.

Definition 6 (Strong Unimodality: [10]) *A distribution $\{H(i)\}$ is strongly unimodal if and only if the convolution of $\{H(i)\}$ with any unimodal distribution $\{P(i)\}$ is unimodal.*

Since a distribution with all mass at zero is unimodal, a strongly unimodal distribution is also unimodal. In this paper, we only consider distributions with all support on the lattice of integers. So the convolution of $\{H(i)\}$ and $\{P(i)\}$ is

$$\{H * P\}(i) = \sum_{j=-\infty}^{\infty} H(j)P(i-j) = \sum_{j=-\infty}^{\infty} H(i-j)P(j)$$

We prove the strong unimodality of $\{R(j,i)\}$ with respect to i via showing it is the convolution of two log-concave distributions by induction. We do an initial statistical query to estimate $\Pr[y=+1]$ to handle two marginal cases $\Pr[y=+1] \leq \epsilon/2$ and $\Pr[y=+1] \geq 1-\epsilon/2$. After that an additional query $\Pr[y=+1 \mid V \sqsubseteq x]$ is made to tell whether $\ell = L$. If the algorithm doesn't halt, it means $\ell < L$ and both $\Pr[y=+1]$ and $\Pr[y=-1]$ are at least $\epsilon/2 - 2\tau$. By upper bounding $\Pr[y=+1]$ and $\Pr[y=-1]$ using linear sums of $R(j,i)$, the strong unimodality of $\{R(j,i)\}$ gives a lower bound for $R(L,n)$, which further implies one for $Q(L-1,n-1)$ and completes the proof.

3.3 A generalization to instance space $\Sigma^{\leq n}$

We have proved the extended class of shuffle ideals is PAC learnable from element-wise i.i.d. fixed-length strings. Nevertheless, in many real-world applications such as natural language processing and computational linguistics, it is more natural to have strings of varying lengths. Let n be the maximum length of the sample strings and as a consequence the instance space for learning is $\Sigma^{\leq n}$. Here we show how to generalize the statistical query algorithm in Section 3.1 to the more general instance space $\Sigma^{\leq n}$.

Let $\mathcal{A}_i$ be the algorithm in Section 3.1 for learning shuffle ideals from element-wise i.i.d. strings of fixed length i. Because instance space $\Sigma^{\leq n} = \bigcup_{i \leq n} \Sigma^i$, we divide the sample S into n subsets $\{S_i\}$ where $S_i = \{x \mid |x| = i\}$. An initial statistical query then is made to estimate probability $\Pr[|x| = i]$ for each $i \leq n$ at tolerance $\epsilon/(8n)$. We discard all subsets S_i with query answer $\leq 3\epsilon/(8n)$ in the learning procedure, because we know $\Pr[|x| = i] \leq \epsilon/(2n)$. As there are at most $(n-1)$ such S_i of low occurrence probabilities. The total probability that an instance comes from one of these negligible sets is at most $\epsilon/2$. Otherwise, $\Pr[|x| = i] \geq \epsilon/(4n)$ and we apply algorithm $\mathcal{A}_i$ on each S_i with query answer $\geq 3\epsilon/(8n)$ with error parameter $\epsilon/2$. Because the probability of the condition is polynomially large, the algorithm is feasible. Finally, the total error over the whole instance space will be bounded by ϵ and concept class $\amalg$ is PAC learnable from element-wise i.i.d. strings over instance space $\Sigma^{\leq n}$.

Corollary 1 *Under element-wise independent and identical distributions over instance space $\mathcal{I} = \Sigma^{\leq n}$, concept class $\amalg$ is approximately identifiable with $O(sn^2)$ conditional statistical queries from STAT$(\amalg, \mathcal{D})$ at tolerance*

$$\tau = \frac{\epsilon^2}{160sn^2 + 8\epsilon}$$

or with $O(sn^2)$ statistical queries from STAT$(\amalg, \mathcal{D})$ at tolerance

$$\bar{\tau} = \left(1 - \frac{\epsilon}{40sn^2 + 2\epsilon}\right) \frac{\epsilon^5}{512sn^2(20sn^2 + \epsilon)}$$

3.4 A constrained generalization to product distributions

A direct generalization from element-wise independent and identical distributions is product distributions. A random string, or a random vector of symbols under a product distribution has element-wise independence between its elements. That is, $\Pr[X = x] = \prod_{i=1}^{|x|} \Pr[X_i = x_i]$. Although strings under product distributions share many independence properties with element-wise i.i.d. strings, the algorithm in Section 3.1 is not directly applicable to this case as the distribution $\{R(j, i)\}$ defined above is not unimodal with respect to i in general. However, the intuition that given $I_{V \sqsubseteq x} = h$, the strings with $x_{h+1} \in U_{\ell+1}$ have higher probability of positivity than that of the strings with $x_{h+1} \notin U_{\ell+1}$ is still true under product distributions. Thus we generalize query $\chi_{V,a}$ and define for any $V \in (\Sigma^{\cup})^{\leq n}$, $a \in \Sigma$ and $h \in [0, n-1]$,

$$\tilde{\chi}_{V,a,h}(x, y) = \frac{1}{2}(y + 1) \quad \text{given } I_{V \sqsubseteq x} = h \text{ and } x_{h+1} = a$$

where $y = c(x)$ is the label of example string x. To ensure the legitimacy and feasibility of the algorithm, we have to attach a lower bound assumption that $\Pr[x_i = a] \geq t > 0$, for $\forall 1 \leq i \leq n$ and $\forall a \in \Sigma$. Appendix C provides a constrained algorithm based on this intuition. Let $P(+|a, h)$ denote $\mathbf{E}\tilde{\chi}_{V,a,h}$. If the difference $P(+|a_+, h) - P(+|a_-, h)$ is large enough for some h with nonnegligible $\Pr[I_{V \sqsubseteq x} = h]$, then we are able to learn the next element in U. Otherwise, the difference is very small and we will show that there is an interval starting from index $(h + 1)$ which we can skip with little risk. The algorithm is able to classify any string whose classification process skips $O(1)$ intervals. Details of this constrained generalization are deferred to Appendix C.

4 Learning principal shuffle ideals from Markovian strings

Markovian strings are widely studied in natural language processing and biological sequence modeling. Formally, a random string x is Markovian if the distribution of x_{i+1} only depends on the value of x_i: $\Pr[x_{i+1} \mid x_1 \ldots x_i] = \Pr[x_{i+1} \mid x_i]$ for any $i \geq 1$. If we denote by π_0 the distribution of x_1 and define $s \times s$ stochastic matrix M by $M(a_1, a_2) = \Pr[x_{i+1} = a_1 \mid x_i = a_2]$, then a random string can be viewed as a Markov chain with initial distribution π_0 and transition matrix M. We choose $\Sigma^{\leq n}$ as the instance space in this section and assume independence between the string length and the symbols in the string. We assume $\Pr[|x| = k] \geq t$ for all $1 \leq k \leq n$ and $\min\{M(\cdot, \cdot), \pi_0(\cdot)\} \geq c$ for some positive t and c. We will prove the PAC learnability of class $\sqcup\!\sqcup$ under this lower bound assumption. Denote by u be the target pattern string and let $L = |u|$.

4.1 Statistical query algorithm

Starting with empty string $v = \lambda$, the pattern string u is recovered one symbol at a time. Having recovered $v = u[1, \ell]$, we infer $u_{\ell+1}$ by $\Psi_{v,a} = \sum_{k=h+1}^{n} \mathbf{E}\chi_{v,a,k}$, where

$$\chi_{v,a,k}(x, y) = \frac{1}{2}(y + 1) \quad \text{given } I_{v \sqsubseteq x} = h,\ x_{h+1} = a \text{ and } |x| = k$$

$0 \leq \ell < L$ and h is chosen from $[0, n-1]$ such that the probability $\Pr[I_{v \sqsubseteq x} = h]$ is polynomially large. The statistical queries $\chi_{v,a,k}$ are made at tolerance τ claimed in Theorem 2 and the symbol with the largest query result of $\Psi_{v,a}$ is proved to be $u_{\ell+1}$. Again, the case where $\ell = L$ is addressed by query $\Pr[y = +1 \mid v \sqsubseteq x]$. The learning procedure is completed if the query result is close to 1.

4.2 PAC learnability of principal ideal $\sqcup\!\sqcup$

With query $\Psi_{v,a}$, we are able to recover the pattern string u approximately from *STAT*$(\sqcup\!\sqcup(u), \mathcal{D})$ at proper tolerance as stated in Theorem 2:

Theorem 2 *Under Markovian string distributions over instance space $\mathcal{I} = \Sigma^{\leq n}$, given $\Pr[|x| = k] \geq t > 0$ for $\forall 1 \leq k \leq n$ and $\min\{M(\cdot, \cdot), \pi_0(\cdot)\} \geq c > 0$, concept class $\sqcup\!\sqcup$ is approximately identifiable with $O(sn^2)$ conditional statistical queries from STAT$(\sqcup\!\sqcup, \mathcal{D})$ at tolerance*

$$\tau = \frac{\epsilon}{3n^2 + 2n + 2}$$

or with $O(sn^2)$ statistical queries from STAT$(\sqcup, \mathcal{D})$ at tolerance

$$\bar{\tau} = \frac{3ctn\epsilon^2}{(3n^2+2n+2)^2}$$

Please refer to Appendix B for a complete proof of Theorem 2. Due to the probability lower bound assumptions, the legitimacy and feasibility are obvious. To calculate the tolerance for PAC learning, we first consider the exact learning tolerance. Let x' be an infinite string generated by the Markov chain defined above. For any $0 \leq \ell \leq L-j$, we define quantity $R_\ell(j,i)$ by

$$R_\ell(j,i) = \Pr[u[\ell+1,\ell+j] \sqsubseteq x'[m+1,m+i] \wedge u[\ell+1,\ell+j] \not\sqsubseteq x'[m+1,m+i-1] \mid x'_m = u_\ell]$$

Intuitively, $R_\ell(j,i)$ is the probability that the smallest g such that $u[\ell+1,\ell+j] \sqsubseteq x'[m+1,m+g]$ is i, given $x'_m = u_\ell$. We have the following conclusion on the exact learning tolerance.

Lemma 3 *Under Markovian string distributions over instance space $\mathcal{I} = \Sigma^{\leq n}$, given $\Pr[|x| = k] \geq t > 0$ for $\forall 1 \leq k \leq n$ and $\min\{M(\cdot,\cdot), \pi_0(\cdot)\} \geq c > 0$, the concept class $\sqcup$ is exactly identifiable with $O(sn^2)$ conditional statistical queries from STAT$(\sqcup, \mathcal{D})$ at tolerance*

$$\tau' = \min_{0 \leq \ell < L} \left\{ \frac{1}{3(n-h)} \sum_{k=h+1}^{n} R_{\ell+1}(L-\ell-1, k-h-1) \right\}$$

The algorithm first deals with the marginal case where $\Pr[y=+1] \leq \epsilon$ through query $\Pr[y=+1]$. If it doesn't halt, we know $\Pr[y=+1]$ is at least $(3n^2+2n)\epsilon/(3n^2+2n+2)$. We then make a statistical query $\chi'_h(x,y) = \frac{1}{2}(y+1) \cdot \mathbb{1}_{\{I_{v \sqsubseteq x} = h\}}$ for each h from ℓ to $n-1$. It can be shown that at least one h will give an answer $\geq (3n+1)\epsilon/(3n^2+2n+2)$. This implies lower bounds for $\Pr[I_{v \sqsubseteq x} = h]$ and $\Pr[y=+1 \mid I_{v \sqsubseteq x} = h]$. The former guarantees the feasibility while the latter can serve as a lower bound for the sum in Lemma 3 after some algebra and completes the proof.

The assumption on M and π_0 can be weakened to $M(u_{\ell+1}, u_\ell) = \Pr[x_2 = u_{\ell+1} \mid x_1 = u_\ell] \geq c$ and $\pi_0(u_1) \geq c$ for all $1 \leq \ell \leq L-1$. We first make a statistical query to estimate $M(a, u_\ell)$ for $\ell \geq 1$ or $\pi_0(a)$ for $\ell = 0$ for each symbol $a \in \Sigma$ at tolerance $c/3$. If the result is $\leq 2c/3$ then $M(a,u_\ell) \leq c$ or $\pi_0(a) \leq c$ and we won't consider symbol a at this position. Otherwise, $M(a,u_\ell) \geq c/3$ or $\pi_0(a) \geq c/3$ and the queries in the algorithm are feasible.

Corollary 2 *Under Markovian string distributions over instance space $\mathcal{I} = \Sigma^{\leq n}$, given $\Pr[|x| = k] \geq t > 0$ for $\forall 1 \leq k \leq n$, $\pi_0(u_1) \geq c$ and $M(u_{\ell+1}, u_\ell) \geq c > 0$ for $\forall 1 \leq \ell \leq L-1$, concept class $\sqcup$ is approximately identifiable with $O(sn^2)$ conditional statistical queries from STAT$(\sqcup, \mathcal{D})$ at tolerance*

$$\tau = \min\left\{ \frac{\epsilon}{3n^2+2n+2}, \frac{c}{3} \right\}$$

or with $O(sn^2)$ statistical queries from STAT$(\sqcup, \mathcal{D})$ at tolerance

$$\bar{\tau} = \min\left\{ \frac{ctn\epsilon^2}{(3n^2+2n+2)^2}, \frac{tn\epsilon c^2}{3(3n^2+2n+2)} \right\}$$

5 Learning shuffle ideals under general distributions

Although the string distribution is restricted or even known in most application scenarios, one might be interested in learning shuffle ideals under general unrestricted and unknown distributions without any prior knowledge. Unfortunately, under standard complexity assumptions, the answer is negative. Angluin et al. [3] have shown that a polynomial time PAC learning algorithm for principal shuffle ideals would imply the existence of polynomial time algorithms to break the RSA cryptosystem, factor Blum integers, and test quadratic residuosity.

Theorem 3 ([3]) *For any alphabet of size at least 2, given two disjoint sets of strings $S, T \subset \Sigma^{\leq n}$, the problem of determining whether there exists a string u such that $u \sqsubseteq x$ for each $x \in S$ and $u \not\sqsubseteq x$ for each $x \in T$ is NP-complete.*

As ideal $\sqcup\!\sqcup$ is a subclass of ideal $\amalg$, we know learning ideal $\amalg$ is only harder. Is the problem easier over instance space Σ^n? The answer is again no.

Lemma 4 *Under general unrestricted string distributions, a concept class is PAC learnable over instance space $\Sigma^{\leq n}$ if and only if it is PAC learnable over instance space Σ^n.*

The proof of Lemma 4 is presented in Appendix D using the same idea as our generalization in Section 3.3. Note that Lemma 4 holds under general string distributions. It is not necessarily true when we have assumptions on the marginal distribution of string length.

Despite the infeasibility of PAC learning a shuffle ideal in theory, it is worth exploring the possibilities to do the classification problem without theoretical guarantees, since most applications care more about the empirical performance than about theoretical results. For this purpose we propose a heuristic greedy algorithm for learning principal shuffle ideals based on reward strategy as follows. Upon having recovered $v = \widehat{u}[1, \ell]$, for a symbol $a \in \Sigma$ and a string x of length n, we say a consumes k elements in x if $\min\{I_{va\sqsubseteq x}, n+1\} - I_{v\sqsubseteq x} = k$. The reward strategy depends on the ratio r_+/r_-: the algorithm receives r_- reward from each element it consumes in a negative example or r_+ penalty from each symbol it consumes in a positive string. A symbol is chosen as $\widehat{u}_{\ell+1}$ if it brings us most reward. The algorithm will halt once $\widehat{u}$ exhausts any positive example and makes a false negative error, which means we have gone too far. Finally the ideal $\sqcup\!\sqcup(\widehat{u}[1, \ell-1])$ is returned as the hypothesis. The performance of this greedy algorithm depends a great deal on the selection of parameter r_+/r_-. A clever choice is $r_+/r_- = \#(-)/\#(+)$, where $\#(+)$ is the number of positive examples x such that $\widehat{u} \sqsubseteq x$ and $\#(-)$ is the number of negative examples x such that $\widehat{u} \sqsubseteq x$. A more recommended but more complex strategy to determine the parameter r_+/r_- in practice is cross validation.

A better studied approach to learning regular languages, especially the piecewise-testable ones, in recent works is kernel machines ([13, 14]). An obvious advantage of kernel machines over our greedy method is its broad applicability to general classification learning problems. Nevertheless, the time complexity of the kernel machine is $O(N^3 + n^2N^2)$ on a training sample set of size N ([5]), while our greedy method only takes $O(snN)$ time due to its great simplicity. Because N is usually huge for the demand of accuracy, kernel machines suffer from low efficiency and long running time in practice. To make a comparison between the greedy method and kernel machines for empirical performance, we conducted a series of experiments on a real world dataset [4] with string length n as a variable. The experiment results demonstrate the empirical advantage on both efficiency and accuracy of the greedy algorithm over the kernel method, in spite of its simplicity. As this is a theoretical paper, we defer the details on the experiments to Appendix D, including the experiment setup and figures of detailed experiment results.

6 Discussion

We have shown positive results for learning shuffle ideals in the statistical query model under element-wise independent and identical distributions and Markovian distributions, as well as a constrained generalization to product distributions. It is still open to explore the possibilities of learning shuffle ideals under less restricted distributions with weaker assumptions. Also a lot more work needs to be done on approximately learning shuffle ideals in applications with pragmatic approaches. In the negative direction, even a family of regular languages as simple as the shuffle ideals is not efficiently properly PAC learnable under general unrestricted distributions unless RP=NP. Thus, the search for a nontrivial properly PAC learnable family of regular languages continues. Another theoretical question that remains is how hard the problem of learning shuffle ideals is, or whether PAC learning a shuffle ideal is as hard as PAC learning a deterministic finite automaton.

Acknowledgments

We give our sincere gratitude to Professor Dana Angluin of Yale University for valuable discussions and comments on the learning problem and the proofs. Our thanks are also due to Professor Joseph Chang of Yale University for suggesting supportive references on strong unimodality of probability distributions and to the anonymous reviewers for their helpful feedback.

Appendix A Proof of Theorem 1

We start our proof from showing the legitimacy and feasibility of the algorithm at the tolerance claimed in the theorem. We first provide a quick proof of Lemma 1.

Lemma 1 (in the main paper) *Evaluating relation $U \sqsubseteq x$ and meanwhile determining $I_{U\sqsubseteq x}$ is feasible in time $O(|x|)$.*

Proof The evalution can be done recursively. The base case is $U = \lambda$, where $U \sqsubseteq x$ holds and $I_{U\sqsubseteq x} = 0$. If $U = U_1U'$ where $U_1 \in \Sigma^{\cup}$, we search for the leftmost occurrence of U_1 in x. If there is no such occurrence, then $U \not\sqsubseteq x$ and $I_{U\sqsubseteq x} = \infty$. Otherwise, $x = yU_1x'$, where $U_1 \not\sqsubseteq y$. Then $U \sqsubseteq x$ if and only if $U' \sqsubseteq x'$ and $I_{U\sqsubseteq x} = I_{U_1\sqsubseteq x} + I_{U'\sqsubseteq x'}$. We continue recursively with U' and x'. The total running time of this procedure is $O(|x|)$. ■

Lemma 5 *Under element-wise independent and identical distributions over instance space $\mathcal{I} = \Sigma^n$, the conditional statistical query $\chi_{V,a}$ is legitimate and feasible at tolerance*

$$\tau = \frac{\epsilon^2}{40sn^2 + 4\epsilon}$$

Proof First of all, the function $\chi_{V,a}$ computes a binary mapping from labeled examples (x, y) to $\{0, 1\}$ and satisfies the definition of a statistical query. Given $\theta_{V,a}(x) = a$, that is, given $V \not\sqsubseteq x[1, n-1]$ or $x_{I_{V\sqsubseteq x}+1} = a$ if $V \sqsubseteq x[1, n-1]$, the query $\chi_{V,a}(x, y)$ returns 0 if x is a negative example $(y = -1)$ or returns 1 if x is a positive example $(y = +1)$.

From Lemma 1, evaluating the relation $V \sqsubseteq x$ and meanwhile determining $I_{V\sqsubseteq x}$ is feasible in time $O(n)$. Thus, $\theta_{V,a}(x)$ and then $\chi_{V,a}(x, y)$ can be efficiently evaluated.

For

$$\begin{aligned}\Pr[\theta_{V,a}(x) = a] = &\Pr[V \not\sqsubseteq x[1, n-1]] + \\ &\Pr[V \sqsubseteq x[1, n-1]] \cdot \Pr[x_{I_{V\sqsubseteq x}+1} = a \mid V \sqsubseteq x[1, n-1]]\end{aligned}$$

in order to prove $\Pr[\theta_{V,a}(x) = a]$ not too small, we only need to show one of the two items in the sum is at least polynomially large.

We make an initial statistical query with tolerance $\tau = \epsilon^2/(40sn^2 + 4\epsilon)$ to estimate $\Pr[y = +1]$. If the answer is $\leq \epsilon - \tau$, then $\Pr[y = +1] \leq \epsilon$ and the algorithm outputs a hypothesis that all examples are negative. Otherwise, $\Pr[y = +1]$ is at least $\epsilon - 2\tau$, and the statistical query $\chi_{V,a}$ is used. As $V \sqsubseteq x[1, n-1] = U[1, \ell] \sqsubseteq x[1, n-1]$ is a necessary condition of $y = +1$, we have

$$\Pr[V \sqsubseteq x[1, n-1]] \geq \Pr[y = +1] \geq \epsilon - \frac{\epsilon^2}{20sn^2 + 2\epsilon}$$

Since $x_{I_{V\sqsubseteq x}+1}$ and $x[1, I_{V\sqsubseteq x}]$ are independent,

$$\Pr[x_{I_{V\sqsubseteq x}+1} = a \mid V \sqsubseteq x[1, n-1]] = \Pr[x_{I_{V\sqsubseteq x}+1} = a]$$

Because we don't have any knowledge of the distribution, we can't guarantee $\Pr[x_{I_{V\sqsubseteq x}+1} = a]$ is large enough for every $a \in \Sigma$. However, we notice that there is no need to consider symbols with small probabilities of occurrence. Now we show why and how. For each $a \in \Sigma$, execute a statistical query

$$\chi'_a(x, y) = \mathbb{1}_{\{x_i = a\}} \tag{1}$$

at tolerance τ, where $\mathbb{1}_{\{\pi\}}$ represents the 0-1 truth value of the predicate π. Since the strings are element-wise i.i.d., the index i can be any integer between 1 and n. If the answer from oracle *STAT* is $\leq \epsilon/(2sn) - \tau$, then $\Pr[x_i = a] \leq \epsilon/(2sn)$. For such an a, the probability that it shows up in a string is at most $\epsilon/(2s)$. Because there are at most $s - 1$ such symbols in Σ, the probability that any of them shows up in a string is at most $\epsilon/2$. Otherwise, $\Pr[x_i = a] \geq \epsilon/(2sn) - 2\tau$. Thus we only need to consider the symbols $a \in \Sigma$ such that $\Pr[x_i = a] \geq \epsilon/(2sn) - 2\tau$ and learn the ideal with error parameter $\epsilon/2$ so that the total error will be bounded within ϵ. For algebra succinctness, we use a concise lower bound for $\Pr[x_i = a]$:

$$\Pr[x_i = a] \geq \frac{\epsilon}{2sn} - 2\tau = \frac{\epsilon}{2sn} - \frac{\epsilon^2}{20sn^2 + 2\epsilon} \geq \frac{\epsilon}{4sn} \tag{2}$$

Eventually we have

$$\begin{aligned}\Pr[\theta_{V,a}(x) = a] &\geq \Pr[V \sqsubseteq x[1, n-1]] \cdot \Pr[x_{I_{V \sqsubseteq x}+1} = a \mid V \sqsubseteq x[1, n-1]] \\ &\geq \left(1 - \frac{\epsilon}{20sn^2 + 2\epsilon}\right) \frac{\epsilon^2}{4sn} \end{aligned} \tag{3}$$

is polynomially large. Query $\chi_{V,a}$ is legitimate and feasible. ■

The correctness of the algorithm is based on the intuition that the query result $\mathbf{E}\chi_{V,a_+}$ of $a_+ \in U_{\ell+1}$ should be greater than that of $a_- \notin U_{\ell+1}$ and the difference is large enough to tolerate the noise from the oracle. To prove this, we first consider the exact learning case. Define an infinite string $U' = U[1, \ell]U[\ell + 2, L]U_{\ell+1}^{\infty}$ and let $x' = x\Sigma^{\infty}$ be the extension of x obtained by padding it on the right with an infinite string generated from the same distribution as x. Let $Q(j, i)$ be the probability that the largest g such that $U'[1, g] \sqsubseteq x'[1, i]$ is j, or formally, $Q(j, i) = \Pr[U'[1, j] \sqsubseteq x'[1, i] \wedge U'[1, j + 1] \not\sqsubseteq x'[1, i]]$.

Lemma 2 (in the main paper) *Under element-wise independent and identical distributions over instance space $\mathcal{I} = \Sigma^n$, concept class $\amalg$ is exactly identifiable with $O(sn)$ conditional statistical queries from STAT($\amalg, \mathcal{D}$) at tolerance*

$$\tau' = \frac{1}{5}Q(L - 1, n - 1)$$

Proof If the algorithm doesn't halt, U has not been completely recovered and $\ell < L$. By assumption, $V = U[1, \ell]$. If $V \not\sqsubseteq x[1, n-1]$ then x must be a negative example and $\chi_{V,a}(x, y) = 0$. Hence $\chi_{V,a}(x, y) = 1$ if and only if $V \sqsubseteq x[1, n - 1]$ and $y = +1$.

Let random variable J be the largest value for which $U'[1, J]$ is a subsequence of $x[1, n - 1]$. Consequently, $\Pr[J = j] = Q(j, n - 1)$.

If $a \in U_{\ell+1}$, then $y = +1$ if and only if $J \geq L - 1$. Thus we have

$$\mathbf{E}\chi_{V,a} = \sum_{j=L-1}^{n-1} Q(j, n - 1)$$

If $a \notin U_{\ell+1}$, then $y = +1$ if and only if $U \sqsubseteq x[1, I_{V \sqsubseteq x}]x[I_{V \sqsubseteq x} + 2, n]$. Since elements in a string are i.i.d., $\Pr[U \sqsubseteq x[1, I_{V \sqsubseteq x}]x[I_{V \sqsubseteq x} + 2, n]] = \Pr[U'[1, L] \sqsubseteq x[1, n - 1]]$, which is exactly $\Pr[J \geq L]$. Thus we have

$$\mathbf{E}\chi_{V,a} = \sum_{j=L}^{n-1} Q(j, n - 1)$$

The difference between these two values is $Q(L - 1, n - 1)$. In order to distinguish the target $U_{\ell+1}$ from other symbols, the query tolerance can be set to one fifth of the difference. The alphabet Σ will be separated into two clusters by the results of $\mathbf{E}\chi_{V,a}$: $U_{\ell+1}$ and the other symbols. The maximum difference (variance) inside a cluster is smaller than the minimum difference (gap) between the two clusters, making them distinguishable. As a consequence s statistical queries for each prefix of U suffice to learn U exactly. ■

Lemma 2 indicates bounding the quantity $Q(L - 1, n - 1)$ is the key to the tolerance for PAC learning. Unfortunately, the distribution $\{Q(j, i)\}$ doesn't seem of any strong properties we know of providing a polynomial lower bound. Instead we introduce new quantity $R(j, i) = \Pr[U'[1, j] \sqsubseteq x'[1, i] \wedge U'[1, j] \not\sqsubseteq x'[1, i - 1]]$ being the probability that the smallest g such that $U'[1, j] \sqsubseteq x'[1, g]$ is i. Now we show the strong unimodality of distribution $\{R(j, i)\}$. Denote $p_j = \Pr[x_i \in U'_j]$.

Lemma 6 *The convolution of two strongly unimodal discrete distributions is strongly unimodal.*

Proof The proof is obvious from the definition of strong unimodality and the associativity of convolution. Let $H_3 = H_2 * H_1$ be the convolution of two strongly unimodal distributions H_1 and H_2. For any unimodal distribution P_1, let $P_2 = H_1 * P_1$ be the convolution of H_1 and P_1. Because of

the strong unimodality of distribution H_1, P_2 is a unimodal distribution. Also because of the strong unimodality of distribution H_2, the convolution of H_3 and P_1, $H_3 * P_1 = H_2 * H_1 * P_1 = H_2 * P_2$ is a unimodal distribution. Since P_1 can be an arbitrary unimodal distribution, H_3 is strongly unimodal according to the definition of strong unimodality. ■

Previous work [10] provided a useful equivalent statement on strong unimodality of a distribution.

Lemma 7 ([10]) *Distribution $\{H(i)\}$ is strongly unimodal if and only if $H(i)$ is log-concave. That is,*

$$H(i)^2 \geq H(i+1) \cdot H(i-1)$$

for all i.

Since a distribution with all mass at zero is unimodal, an immediate consequence is

Corollary 3 *A strongly unimodal distribution is unimodal.*

We now prove the strong unimodality of distribution $\{R(j,i)\}$.

Lemma 8 *For any fixed j, distribution $\{R(j,i)\}$ is strongly unimodal with respect to i.*

Proof This proof can be done by induction on j as follows.

Basis: For $j = 1$, it is obvious that $\{R(1,i)\} = \{(1-p_1)^{i-1}p_1\}$ is a geometric distribution, which is strongly unimodal. According to Lemma 7, this is due to $R^2(1,i) = R(1,i-1) \cdot R(1,i+1)$ for all $i > 1$.

Inductive step: For $j > 1$, assume by induction $\{R(j-1,i)\}$ is strongly unimodal. Based on the definition of $R(j,i)$, we have

$$R(j,i) = \sum_{k=j-1}^{i-1} \left(R(j-1,k) \cdot (1-p_j)^{i-k-1}p_j\right) \tag{4}$$

Thus $R(j,i)$ is the convolution of distribution $\{R(j-1,i)\}$ and distribution $\{(1-p_j)^{i-1}p_j\}$, a geometric distribution just proved to be strongly unimodal. By assumption, $\{R(j-1,i)\}$ is strongly unimodal. From Lemma 6, distribution $\{R(j,i)\}$ is also strongly unimodal.

Conclusion: For any fixed j, distribution $\{R(j,i)\}$ is strongly unimodal with respect to i. ■

Combining Lemma 8 with Corollary 3, we have

Corollary 4 *For any fixed j, distribution $\{R(j,i)\}$ is unimodal with respect to i.*

Lemma 9 *Denote by $N(j)$ the mode of $\{R(j,i)\}$, then $N(j)$ is strictly increasing with respect to j. That is, for any $j > 1$, $N(j) > N(j-1)$.*

Proof According to Equation 4, $R(j,i)$ is the convolution of distribution $\{R(j-1,i)\}$ and distribution $\{(1-p_j)^{i-1}p_j\}$ so

$$R(j,i) = \sum_{k=j-1}^{i-1} \left(R(j-1,k) \cdot (1-p_j)^{i-k-1}p_j\right)$$

and

$$R(j,i+1) = \sum_{k=j-1}^{i} \left(R(j-1,k) \cdot (1-p_j)^{i-k}p_j\right)$$

Hence, we get

$$R(j,i+1) = p_j R(j-1,i) + (1-p_j)R(j,i) \tag{5}$$

Denote by $\Delta R(j,i)$ the difference $R(j,i) - R(j,i-1)$. From Equation 5, we have

$$\Delta R(j,i+1) = p_j \Delta R(j-1,i) + (1-p_j)\Delta R(j,i) \tag{6}$$

For any $j \geq 1$, we have $R(j,1) \geq R(j,0) = 0$ or $\Delta R(j,1) \geq 0$. From the definition of $N(j)$, $N(j)$ must be at least j and for any $i \leq N(j-1)$, the difference $\Delta R(j-1,i)$ is non-negative. Hence, if $\Delta R(j,i)$ is non-negative, then $\Delta R(j,i+1)$ is non-negative for Equation 6. So inductively, for any $i \leq N(j-1)+1$, we always have $\Delta R(j,i) \geq 0$. Recall that we define the mode of a distribution with multiple modes as the one with the largest index, thus $N(j) > N(j-1)$. ■

With the strong unimodality of distribution $\{R(j,i)\}$, we are able to present the PAC learnability of concept class III in the statistical query model.

Theorem 1 (in the main paper) *Under element-wise independent and identical distributions over instance space $\mathcal{I} = \Sigma^n$, concept class III is approximately identifiable with $O(sn)$ conditional statistical queries from STAT$(\text{III}, \mathcal{D})$ at tolerance*

$$\tau = \frac{\epsilon^2}{40sn^2 + 4\epsilon}$$

or with $O(sn)$ statistical queries from STAT$(\text{III}, \mathcal{D})$ at tolerance

$$\bar{\tau} = \left(1 - \frac{\epsilon}{20sn^2 + 2\epsilon}\right) \frac{\epsilon^4}{16sn(10sn^2 + \epsilon)}$$

Proof From Lemma 5, statistical query $\chi_{V,a}$ is legitimate and feasible at tolerance $\tau = \epsilon^2/(40sn^2 + 4\epsilon)$ and our error parameter must be set to $\epsilon/2$ in order to have Inequality 2.

We modify the statistical query algorithm to make an initial statistical query with tolerance $\tau = \epsilon^2/(40sn^2 + 4\epsilon)$ to estimate $\Pr[y = +1]$. If the answer is $\leq \epsilon/2 - \tau$, then $\Pr[y = +1] \leq \epsilon/2$ and the algorithm outputs a hypothesis that all examples are negative. If the answer is $\geq 1 - \epsilon/2 + \tau$, then $\Pr[y = +1] \geq 1 - \epsilon/2$ and the algorithm outputs a hypothesis that all examples are positive.

Otherwise, $\Pr[y = +1]$ and $\Pr[y = -1]$ are both at least $\epsilon/2 - 2\tau$. We then do another statistical query at tolerance τ to estimate $\Pr[y = +1 \mid V \sqsubseteq x]$. Since $V \sqsubseteq x$ is a necessary condition of positivity, $\Pr[V \sqsubseteq x]$ must be at least $\Pr[y = +1] \geq \epsilon/2 - 2\tau$ and this statistical query is legitimate and feasible. If the answer is $\geq 1 - \epsilon/2 + \tau$, then $\Pr[y = +1 \mid V \sqsubseteq x] \geq 1 - \epsilon/2$. The algorithm outputs a hypothesis that all strings x such that $V \sqsubseteq x$ are positive and all strings x such that $V \not\sqsubseteq x$ are negative because $\Pr[y = -1 \mid V \not\sqsubseteq x] = 1$. If $\ell = L$, $\Pr[y = +1 \mid V \sqsubseteq x]$ must be 1 and the algorithm halts. Otherwise, $\ell < L$ and the first statistical query algorithm is used. We now show that $Q(L-1, n-1) \geq 5\tau$, establishing the bound on the query tolerance.

Let random variable I be the smallest value for which $U'[1,L]$ is a subsequence of $x'[1,I]$. Based on the definition of $R(j,i)$, we have $\Pr[I = i] = R(L,i)$. String x is a positive example if and only if $U'[1,L] \sqsubseteq x'[1,n]$, which is exactly $I \leq n$. As a consequence,

$$\Pr[y = +1] = \sum_{i=L}^{n} R(L,i) \tag{7}$$

From Corollary 4, distribution $\{R(L,i)\}$ is unimodal and assume its mode is $N(L)$. If $n \leq N(L)$ then $R(L,n)$ is at least as large as every term in the sum $\Pr[y = +1] = \sum_{i=L}^{n} R(L,i)$. Hence we get

$$R(L,n) \geq \frac{\epsilon - 4\tau}{2(n - L + 1)} \geq \frac{\epsilon - 4\tau}{2n} \geq \frac{5\epsilon^2}{40sn^2 + 4\epsilon} = 5\tau$$

If $n > N(L)$, according to Lemma 9, for any $j \leq L$ we have $n > N(j)$. That is, for any $j \leq L$, we have $R(j,n) \geq R(j,n+1)$.

From Equation 5,

$$R(j,n+1) = p_j R(j-1,n) + (1 - p_j) R(j,n)$$

so

$$\begin{aligned} p_j R(j-1,n) + (1-p_j) R(j,n) \leq & R(j,n) \\ = & p_j R(j,n) + (1-p_j) R(j,n) \end{aligned}$$

We then have

$$R(j-1,n) \leq R(j,n)$$

This holds for any $j \leq L$ so $R(j,n)$ is non-decreasing with respect to j when $n > N(L)$. Inductively we get $R(L,n) \geq R(j,n)$ for any $j \leq L$.

Because $U'[1,L] \not\sqsubseteq x[1,n-1]$ is a necessary condition of $y=-1$ and

$$\Pr[U'[1,L] \not\sqsubseteq x[1,n-1]] = \sum_{j=0}^{L-1} Q(j,n-1)$$

we get

$$\sum_{j=0}^{L-1} Q(j,n-1) \geq \Pr[y=-1] \geq \frac{\epsilon - 4\tau}{2}$$

Note that $R(j,n) = p_j Q(j-1,n-1)$, then

$$\sum_{j=1}^{L} \frac{R(j,n)}{p_j} \geq \frac{\epsilon - 4\tau}{2}$$

Since

$$\Pr[y=+1] \geq \frac{\epsilon - 4\tau}{2} > 0$$

from Inequality 2, we must have $p_j \geq \epsilon/(4sn)$ for all j. Then we have

$$\frac{4sn}{\epsilon} \sum_{j=1}^{L} R(j,n) \geq \sum_{j=1}^{L} \frac{R(j,n)}{p_j} \geq \frac{\epsilon - 4\tau}{2}$$

Because $R(L,n) \geq R(j,n)$ for any $j \leq L$, we get

$$\frac{4sn}{\epsilon} L R(L,n) \geq \frac{\epsilon - 4\tau}{2}$$

and

$$R(L,n) \geq \frac{(\epsilon - 4\tau)\epsilon}{8sn^2} = \frac{5\epsilon^2}{40sn^2 + 4\epsilon} = 5\tau$$

Finally, we have

$$\begin{aligned} Q(L,n) &= (1 - p_{L+1}) Q(L,n-1) + p_L Q(L-1,n-1) \\ &\geq p_L Q(L-1,n-1) = R(L,n) \geq \frac{5\epsilon^2}{40sn^2 + 4\epsilon} \end{aligned}$$

That is, $Q(L-1,n-1) \geq 5\tau$. For Lemma 2, we have $\tau = \epsilon^2/(40sn^2 + 4\epsilon)$. Inferring $\bar{\tau}$ from τ is trivial. Define general statistical query

$$\bar{\chi}_{V,a}(x,y) = \begin{cases} (y+1)/2 & \text{if } \theta_{V,a}(x) = a \\ 0 & \text{if } \theta_{V,a}(x) \neq a \end{cases} \tag{8}$$

Then for any a, the expected query result

$$\mathbf{E}\bar{\chi}_{V,a} = \Pr[\theta_{V,a}(x) = a] \cdot \mathbf{E}\chi_{V,a} + 0$$

and the difference between $\mathbf{E}\bar{\chi}_{V,a} \mid a \in U_{\ell+1}$ and $\mathbf{E}\bar{\chi}_{V,a} \mid a \notin U_{\ell+1}$ is $5\tau \cdot \Pr[\theta_{V,a}(x) = a]$. Hence, from Inequality 3,

$$\bar{\tau} = \left(1 - \frac{\epsilon}{20sn^2 + 2\epsilon}\right) \frac{\epsilon^4}{16sn(10sn^2 + \epsilon)}$$

This completes the proof. ■

Appendix B Proof of Theorem 2

To calculate the tolerance for PAC learning, we first consider the exact learning tolerance. Let x' be an infinite string generated by the Markov chain defined in Section 4. For any $0 \leq \ell \leq L$, we define quantity $R_\ell(j,i)$ by

$$R_\ell(j,i) = \Pr[u[\ell+1,\ell+j] \sqsubseteq x'[m+1,m+i] \wedge u[\ell+1,\ell+j] \not\sqsubseteq x'[m+1,m+i-1] \mid x'_m = u_\ell]$$

Intuitively, $R_\ell(j,i)$ is the probability that the smallest g such that $u[\ell+1,\ell+j] \sqsubseteq x'[m+1,m+g]$ is i, given $x'_m = u_\ell$. We have the following conclusion on the exact learning tolerance.

Lemma 3 (in the main paper) *Under Markovian string distributions over instance space $\mathcal{I} = \Sigma^{\leq n}$, given $\Pr[|x| = k] \geq t > 0$ for $\forall 1 \leq k \leq n$ and $\min\{M(\cdot,\cdot), \pi_0(\cdot)\} \geq c > 0$, the concept class $\sqcup\!\sqcup$ is exactly identifiable with $O(sn^2)$ conditional statistical queries from STAT($\sqcup\!\sqcup, \mathcal{D}$) at tolerance*

$$\tau' = \min_{0 \leq \ell < L} \left\{ \frac{1}{3(n-h)} \sum_{k=h+1}^{n} R_{\ell+1}(L-\ell-1, k-h-1) \right\}$$

Proof If the algorithm doesn't halt, u has not been completely recovered and $\ell < L$. Again, we calculate the difference of $\Psi_{v,a}$ between the cases $a_+ = u_{\ell+1}$ and $a_- \neq u_{\ell+1}$.

For $a_- \neq u_{\ell+1}$, let p_j denote the probability that the first passage time from a_- to $u_{\ell+1}$ is equal to j. Notice that

$$\begin{aligned}
\mathbf{E}\chi_{v,a_-,k} &= \sum_{j=1}^{k-h-1} \left(p_j \sum_{i=0}^{k-h-1-j} R_{\ell+1}(L-\ell-1, i) \right) \\
&\leq \sum_{j=1}^{k-h-1} \left(p_j \sum_{i=0}^{k-h-2} R_{\ell+1}(L-\ell-1, i) \right)
\end{aligned}$$

We get

$$\mathbf{E}\chi_{v,a_-,k} \leq \sum_{i=0}^{k-h-2} R_{\ell+1}(L-\ell-1, i)$$

For $a_+ = u_{\ell+1}$, we have

$$\mathbf{E}\chi_{v,a_+,k} = \sum_{i=0}^{k-h-1} R_{\ell+1}(L-\ell-1, i)$$

Summing up all the items, we can get the difference

$$\begin{aligned}
\Psi_{v,a_+} - \Psi_{v,a_-} &= \sum_{k=h+1}^{n} \left(\mathbf{E}\chi_{v,a_+,k} - \mathbf{E}\chi_{v,a_-,k} \right) \\
&\geq \sum_{k=h+1}^{n} \left(\sum_{i=0}^{k-h-1} R_{\ell+1}(L-\ell-1, i) - \sum_{i=0}^{k-h-2} R_{\ell+1}(L-\ell-1, i) \right) \\
&= \sum_{k=h+1}^{n} R_{\ell+1}(L-\ell-1, k-h-1)
\end{aligned}$$

In order to distinguish the target $u_{\ell+1}$ from other symbols, the query tolerance can be set to one third of the difference so that the symbol with largest query result must be $u_{\ell+1}$. Thus the overall tolerance for $\Psi_{v,a}$ is $\sum_{k=h+1}^{n} R_{\ell+1}(L-\ell-1, k-h-1)/3$. Since $\Psi_{v,a}$ is the expectation sum of $(n-h)$ statistical queries, we can evenly distribute the overall tolerance on each $\chi_{v,a,k}$. So the final tolerance on each statistical query is

$$\tau' = \min_{0 \leq \ell < L} \left\{ \frac{1}{3(n-h)} \sum_{k=h+1}^{n} R_{\ell+1}(L-\ell-1, k-h-1) \right\}$$

Taking minimum over $0 \leq \ell < L$ is because h depends on ℓ and the tolerance needs to be independent of h. As a consequence sn statistical queries for each prefix of U suffice to learn U

exactly. ■

We then show how to choose a proper h from $[0, n-1]$.

Lemma 10 *Under Markovian string distributions over instance space $\mathcal{I} = \Sigma^{\leq n}$, given $\Pr[|x| = k] \geq t > 0$ for $\forall 1 \leq k \leq n$ and $\min\{M(\cdot,\cdot), \pi_0(\cdot)\} \geq c > 0$, the conditional statistical query $\chi_{v,a,k}$ is legitimate and feasible at tolerance*

$$\tau = \frac{\epsilon}{3n^2 + 2n + 2}$$

Proof First of all, the function $\chi_{v,a,k}$ computes a binary mapping from labeled examples (x, y) to $\{0, 1\}$ and satisfies the definition of a statistical query. Under the given conditions, $\chi_{v,a,k}$ returns 0 if x is a negative example $(y = -1)$ or returns 1 if x is a positive example $(y = +1)$.

From Lemma 1, evaluating the relation $v \sqsubseteq x$ and meanwhile determining $I_{v \sqsubseteq x}$ is feasible in time $O(n)$. Since $|x| \leq n$, determining $|x|$ also takes $O(n)$ time. Thus, $\chi_{v,a,k}(x, y)$ and then $\Psi_{v,a}$ can be efficiently evaluated.

According to the Markov property and the independence between string length and symbols in a string, we have

$$\begin{aligned}
&\Pr[I_{v \sqsubseteq x} = h,\ x_{h+1} = a \text{ and } |x| = k] \\
=&\Pr[I_{v \sqsubseteq x} = h] \cdot \Pr[x_{h+1} = a \mid I_{v \sqsubseteq x} = h] \cdot \Pr[|x| = k] \\
\geq&\Pr[I_{v \sqsubseteq x} = h] \cdot c \cdot t
\end{aligned}$$

The only problem left is to make sure $\Pr[I_{v \sqsubseteq x} = h]$ is polynomially large. Obviously this can't be guaranteed for all h between ℓ and $n-1$ so h must be chosen carefully. We now show there must be such an h.

We make an initial statistical query with tolerance $\epsilon/(3n^2 + 2n + 2)$ to estimate $\Pr[y = +1]$. If the answer is $\leq (3n^2 + 2n + 1)\epsilon/(3n^2 + 2n + 2)$, then $\Pr[y = +1] \leq \epsilon$ and the algorithm outputs a hypothesis that all examples are negative. Otherwise, $\Pr[y = +1]$ is at least $(3n^2 + 2n)\epsilon/(3n^2 + 2n + 2)$, and the statistical queries $\{\chi_{v,a,k}\}$ are used. Since

$$\Pr[y = +1] = \sum_{h=\ell}^{n-1} \Pr[y = +1 \wedge I_{v \sqsubseteq x} = h] \tag{9}$$

There must be at least one h so that

$$\begin{aligned}
\Pr[y = +1 \wedge I_{v \sqsubseteq x} = h] \geq& \frac{1}{n-h} \Pr[y = +1] \\
\geq& \frac{1}{n} \Pr[y = +1] \\
\geq& \frac{1}{n} \cdot \frac{(3n^2 + 2n)\epsilon}{3n^2 + 2n + 2} \\
=& \frac{(3n + 2)\epsilon}{3n^2 + 2n + 2}
\end{aligned}$$

As

$$\Pr[y = +1 \wedge I_{v \sqsubseteq x} = h] = \Pr[y = +1 \mid I_{v \sqsubseteq x} = h] \cdot \Pr[I_{v \sqsubseteq x} = h]$$

both $\Pr[y = +1 \mid I_{v \sqsubseteq x} = h]$ and $\Pr[I_{v \sqsubseteq x} = h]$ must be at least $(3n + 2)\epsilon/(3n^2 + 2n + 2)$. This means there must be some h making our statistical queries legitimate.

We now show how to determine a proper value of h. We can do a statistical query

$$\chi'_h(x, y) = \frac{1}{2}(y + 1) \cdot \mathbb{1}_{\{I_{v \sqsubseteq x} = h\}} \tag{10}$$

for each h from ℓ to $n-1$, where $\mathbb{1}_{\{\pi\}}$ represents the 0-1 truth value of the predicate π. It is easy to see $\mathbf{E}\chi'_h = \Pr[y = +1 \wedge I_{v \sqsubseteq x} = h]$. According to our analysis above and due to the noise of the

statistical query, there must be at least one h such that the answer is $\geq (3n+1)\epsilon/(3n^2+2n+2)$. If we choose such an h, it is guaranteed to have

$$\Pr[y = +1 \wedge I_{v \sqsubseteq x} = h] \geq \frac{3n\epsilon}{3n^2+2n+2}$$

so that

$$\Pr[I_{v \sqsubseteq x} = h] \geq \frac{3n\epsilon}{3n^2+2n+2}$$

and

$$\Pr[y = +1 \mid I_{v \sqsubseteq x} = h] \geq \frac{3n\epsilon}{3n^2+2n+2} \tag{11}$$

After at most n statistical queries $\{\chi'_h\}$, we can determine the value of h in query $\chi_{v,a,k}$. Thus statistical queries $\{\chi_{v,a,k}\}$ and $\Psi_{v,a}$ are legitimate and feasible. ■

Below is the proof of Theorem 2.

Theorem 2 (in the main paper) *Under Markovian string distributions over instance space $\mathcal{I} = \Sigma^{\leq n}$, given $\Pr[|x| = k] \geq t > 0$ for $\forall 1 \leq k \leq n$ and $\min\{M(\cdot,\cdot), \pi_0(\cdot)\} \geq c > 0$, concept class $\sqcup\!\sqcup$ is approximately identifiable with $O(sn^2)$ conditional statistical queries from STAT$(\sqcup\!\sqcup, \mathcal{D})$ at tolerance*

$$\tau = \frac{\epsilon}{3n^2+2n+2}$$

or with $O(sn^2)$ statistical queries from STAT$(\sqcup\!\sqcup, \mathcal{D})$ at tolerance

$$\bar{\tau} = \frac{3ctn\epsilon^2}{(3n^2+2n+2)^2}$$

Proof From Lemma 10, statistical queries $\{\chi_{v,a,k}\}$ and $\Psi_{v,a}$ are legitimate and feasible at tolerance $\epsilon/(3n^2+2n+2)$.

We modify the statistical query algorithm to make an initial statistical query with tolerance $\epsilon/(3n^2+2n+2)$ to estimate $\Pr[y = +1]$. If the answer is $\leq (3n^2+2n+1)\epsilon/(3n^2+2n+2)$, then $\Pr[y = +1] \leq \epsilon$ and the algorithm outputs a hypothesis that all examples are negative. Otherwise, $\Pr[y = +1]$ is at least $(3n^2+2n)\epsilon/(3n^2+2n+2)$.

We then do another statistical query with tolerance $\epsilon/(3n^2+2n+2)$ to estimate $\Pr[y = +1 \mid v \sqsubseteq x]$. Since $v \sqsubseteq x$ is a necessary condition of positivity, $\Pr[v \sqsubseteq x]$ must be at least $\Pr[y = +1] \geq (3n^2+2n)\epsilon/(3n^2+2n+2)$ and this statistical query is legitimate and feasible. If the answer is $\geq 1 - (3n^2+2n)\epsilon/(3n^2+2n+2)$, then $\Pr[y = +1 \mid v \sqsubseteq x] \geq 1 - \epsilon$. The algorithm outputs a hypothesis that all strings x such that $v \sqsubseteq x$ are positive and all strings x such that $v \not\sqsubseteq x$ are negative because $\Pr[y = -1 \mid v \not\sqsubseteq x] = 1$. If $\ell = L$, $\Pr[y = +1 \mid v \sqsubseteq x]$ must be 1 and the algorithm halts. Otherwise, $\ell < L$ and the first statistical query algorithm is used.

From the proof for Lemma 10, we then use $O(n)$ statistical queries

$$\chi'_h(x, y) = \frac{1}{2}(y+1) \cdot \mathbb{1}_{\{I_{v \sqsubseteq x} = h\}}$$

to find an h such that Inequality 11 holds:

$$\Pr[y = +1 \mid I_{v \sqsubseteq x} = h] \geq \frac{3n\epsilon}{3n^2+2n+2}$$

Similarly, let q_j denote the probability that the first passage time from u_ℓ to $u_{\ell+1}$ is equal to j. Notice that

$$\Pr[y = +1 \mid I_{v \sqsubseteq x} = h] \leq \sum_{j=1}^{n-h} \left(q_j \sum_{i=0}^{n-h-j} R_{\ell+1}(L-\ell-1, i) \right)$$

We have

$$\begin{aligned}\frac{3n\epsilon}{3n^2+2n+2} &\le \Pr[y=+1 \mid I_{v\sqsubseteq x}=h] \\ &\le \sum_{j=1}^{n-h}\left(q_j \sum_{i=0}^{n-h-j} R_{\ell+1}(L-\ell-1,i)\right) \\ &\le \sum_{j=1}^{n-h}\left(q_j \sum_{i=0}^{n-h-1} R_{\ell+1}(L-\ell-1,i)\right) \\ &\le \sum_{i=0}^{n-h-1} R_{\ell+1}(L-\ell-1,i) \\ &= \sum_{k=h+1}^{n} R_{\ell+1}(L-\ell-1,k-h-1)\end{aligned}$$

From Lemma 3, the conditional tolerance is

$$\tau = \min_{0\le\ell<L}\left\{\frac{1}{3(n-h)}\sum_{k=h+1}^{n} R_{\ell+1}(L-\ell-1,k-h-1)\right\} \ge \frac{\epsilon}{3n^2+2n+2}$$

Similar to the proof of Theorem 1, define general statistical query

$$\bar{\chi}_{v,a,k}(x,y) = \begin{cases} (y+1)/2 & \text{if } I_{v\sqsubseteq x}=h,\ x_{h+1}=a \text{ and } |x|=k \\ 0 & \text{otherwise}\end{cases} \tag{12}$$

and

$$\bar{\Psi}_{v,a} = \sum_{k=h+1}^{n} \mathbf{E}\bar{\chi}_{v,a,k} \tag{13}$$

Then the general tolerance $\bar{\tau}$ can be easily inferred from the conditional tolerance τ:

$$\bar{\tau} = \frac{3ctn\epsilon^2}{(3n^2+2n+2)^2}$$

Considering we have used n statistical queries to determine h, $(s+1)n$ statistical queries for each prefix of u suffice to PAC learn u. This completes the proof. ■

Appendix C A constrained generalization to learning shuffle ideals under product distributions

In this section we generalize the idea in Section 3 to learning the extended class of shuffle ideals when example strings are drawn from a product distribution. For any augmented string $V \in (\Sigma^{\cup})^{\le n}$, any symbol $a \in \Sigma$ and any integer $h \in [0, n-1]$, define

$$\tilde{\chi}_{V,a,h}(x,y) = \frac{1}{2}(y+1) \quad \text{given } I_{V\sqsubseteq x}=h \text{ and } x_{h+1}=a$$

where $y = c(x)$ is the label of x. Again the algorithm uses query $\Pr[y=+1 \mid V \sqsubseteq x]$ to tell whether it is time to halt. As before, let V be the partial pattern we have learned and the algorithm starts with $V=\lambda$. For $1 \le i \le n$ and $1 \le j \le L$, define probability $\tilde{Q}(j,i)$ as below.

$$\tilde{Q}(j,i) = \begin{cases} \Pr[U[L-j+1,L] \sqsubseteq x[n-i+1,n] \wedge U[L-j,L] \not\sqsubseteq x[n-i+1,n]] & \text{if } 1 \le j < L \\ \Pr[U \sqsubseteq x[n-i+1,n]] & \text{if } j = L\end{cases}$$

Lemma 11 *Under product distributions over instance space $\mathcal{I} = \Sigma^n$, given $\Pr[x_i = a] \geq t > 0$ for $\forall 1 \leq i \leq n$ and $\forall a \in \Sigma$, concept class Ш is exactly identifiable with $O(sn)$ conditional statistical queries from STAT$(\text{Ш}, \mathcal{D})$ at tolerance*

$$\tau' = \frac{1}{5} \min \left\{ \tilde{Q}(L-1, n-1), \min_{1 \leq \ell \leq L} \max_{\ell \leq h \leq n-1} \tilde{Q}(L-\ell-1, n-h-1) \right\}$$

Proof If the algorithm doesn't halt, U has not been completely recovered and $\ell < L$. As before, we calculate the difference of $\mathbf{E}\tilde{\chi}_{V,a,h}$ between the cases $a_+ \in U_{\ell+1}$ and $a_- \notin U_{\ell+1}$.

When $\ell = 0$ and $V = \lambda$, the value of $I_{V \sqsubseteq x}$ must be 0 so h is fixed to be 0 in the query. For symbol $a_+ \in U_1$, we have

$$\mathbf{E}\tilde{\chi}_{\lambda,a_+,0} = \tilde{Q}(L-1, n-1) + \tilde{Q}(L, n-1)$$

and for symbol $a_- \notin U_1$,

$$\mathbf{E}\tilde{\chi}_{\lambda,a_-,0} = \tilde{Q}(L, n-1)$$

Taking one fifth of the difference gives the tolerance $\tilde{Q}(L-1, n-1)/5$ for $\ell = 0$.

When $1 \leq \ell < L$ and $V = U[1, \ell]$, we have for symbol $a_+ \in U_{\ell+1}$,

$$\mathbf{E}\tilde{\chi}_{V,a_+,h} = \sum_{j=L-\ell-1}^{L} \tilde{Q}(j, n-h-1)$$

and for symbol $a_- \notin U_{\ell+1}$,

$$\mathbf{E}\tilde{\chi}_{V,a_-,h} = \sum_{j=L-\ell}^{L} \tilde{Q}(j, n-h-1)$$

Again taking one fifth of the difference gives the tolerance $\tilde{Q}(L-\ell-1, n-h-1)/5$. For a fixed $1 \leq \ell < L$, tolerance $\max_{\ell \leq h \leq n-1} \tilde{Q}(L-\ell-1, n-h-1)/5$ is enough to learn $U_{\ell+1}$ exactly. Taking the minimum tolerance among all $0 \leq \ell < L$ gives the overall tolerance in the statement. As a consequence s statistical queries for each prefix of U suffice to learn U exactly. ■

A more complicated algorithm is needed to PAC learn shuffle ideals under product distributions. We first define two additional simple queries:

$$\chi'_{V,a,h,i}(x, y) = \mathbb{1}_{\{x_{h+i}=a\}} \quad \text{given } I_{V \sqsubseteq x} = h$$
$$\chi^+_{V,a,h,i}(x, y) = \mathbb{1}_{\{x_{h+i}=a\}} \quad \text{given } I_{V \sqsubseteq x} = h \text{ and } y = +1$$

whose expectations serve as empirical estimators for the distributions of the symbol at the next i-th position over all strings ($\chi'_{V,a,i}$) and over all positive strings ($\chi^+_{V,a,i}$), both conditioned on $I_{V \sqsubseteq x} = h$. Below is how the algorithm works, with $\bar{\epsilon}_{g+1}$ and ϵ' to be decided later in the proof.

First an initial query to estimate probability $\Pr[y = +1 \mid V \sqsubseteq x]$ is made. The algorithm will classify all strings such that $V \sqsubseteq x$ negative if the answer is close to 0, or positive if the answer is close to 1. To ensure the legitimacy and feasibility of the algorithm, we make another initial query to estimate the probability $\Pr[I_{V \sqsubseteq x} = h]$ for each h. The algorithm then excludes the low-probability cases such that any of the excluded ones happens with probability lower than $\epsilon/2$. Thus we only need to consider the cases with polynomially large $\Pr[I_{V \sqsubseteq x} = h]$ and learn the target ideal within error $\epsilon/2$. Otherwise, let $P(+|a, h)$ denote $\mathbf{E}\tilde{\chi}_{V,a,h}$ and we make a statistical query to estimate $P(+|a, h)$ for each $a \in \Sigma$. If the difference $P(+|a_+, h) - P(+|a_-, h)$, where a_+ is in the next element of U and a_- is not, is large enough for some h, then the results of queries for $P(+|a, h)$ will form two distinguishable clusters, where the maximum difference inside one cluster is smaller than the minimum gap between them, so that we are able to learn the next element in U.

Otherwise, for all h with nonnegligible $\Pr[I_{V \sqsubseteq x} = h]$, the difference $P(+|a_+, h) - P(+|a_-, h)$ is very small and we will show that there is an interval starting from index $h+1$ which we can skip with little risk for each case when $I_{V \sqsubseteq x} = h$. Problematic cases leading to misclassification will happen with very small probability within this interval. We are safe to skip the whole interval and move on. The remaining problem is to identify the length of this interval, that is, to estimate the probability

1. Estimate probability $\Pr[y = +1 \mid V \sqsubseteq x]$ at tolerance $\epsilon'/3$. If the answer is $\leq 2\epsilon'/3$, classify all strings x such that $V \sqsubseteq x$ as negative and backtrack on the classification tree. If the answer is $\geq 1 - 2\epsilon'/3$, classify all strings x such that $V \sqsubseteq x$ as positive and backtrack. If the number of intervals skipped on the current path exceeds C, classify all strings x such that $V \sqsubseteq x$ as positive and backtrack. Otherwise go to Step 2
2. For each h with nonnegligible $\Pr[I_{V \sqsubseteq x} = h]$, estimate $\mathbf{E}\chi_{V,a,h}$ at tolerance $\tau_1 = \bar{\epsilon}_{g+1}^2/384$ for each $a \in \Sigma$. Go to Step 3.
3. If the results for some h produce two distinguishable clusters, where the maximum difference inside one cluster is $\leq 4\tau_1$ while the minimum gap between two clusters is $> 4\tau_1$, then the set of all the symbols that belong to the cluster with larger query results is the next element in U. Update V and go to Step 1. Otherwise, branch the classification tree. For each h, let $k \leftarrow 1$ and $T \leftarrow 1$. Go to Step 4.
4. For each $a \in \Sigma$, estimate $\mathbf{E}\chi'_{V,a,h,k}$ and $\mathbf{E}\chi^+_{V,a,h,k}$ at tolerance $\tau_2 = \bar{\epsilon}_{g+1}/(8sn)$ so that we will have estimators $\widehat{\mathcal{D}}_k(h)$ and $\widehat{\mathcal{D}}^+_k(h)$. Go to Step 5.
5. $T \leftarrow (1 - \|\widehat{\mathcal{D}}_k(h) - \widehat{\mathcal{D}}^+_k(h)\|_{TV}) \cdot T$. If $1 - T \leq 3\bar{\epsilon}_{g+1}/4$, $k \leftarrow k + 1$ and go to Step 4. Otherwise, skip the interval from x_{h+1} to x_{h+k-1}. Update V and go to Step 1.

Figure 2: Approximately learning III under product distributions

that an error happens if we skip an interval. Let $\mathcal{D}_{1:k}(h)$ be the distribution of $x[h+1, h+k]$ over all strings given $I_{V \sqsubseteq x} = h$ and $\mathcal{D}^+_{1:k}(h)$ be the corresponding distribution over all positive strings given $I_{V \sqsubseteq x} = h$. The probability that an error happens due to skipping the next k elements is the total variation distance between $\mathcal{D}_{1:k}(h)$ and $\mathcal{D}^+_{1:k}(h)$. Thanks to the independence between the elements in a string, it can be proved that $\|\mathcal{D}_{1:k}(h) - \mathcal{D}^+_{1:k}(h)\|_{TV}$ can be estimated within polynomially bounded error.

Because the lengths of skipped intervals in cases with different $I_{V \sqsubseteq x}$ could be different, the algorithm branches the classification tree to determine the skipped interval according to the value of $I_{V \sqsubseteq x}$. The algorithm runs the procedure above recursively on each branch. Figure 2 demonstrates this skipping strategy of the algorithm, where parameter C is the maximum allowed number of skipped intervals on each path. Notice that the algorithm might not recover the complete pattern string U. Instead the hypothesis pattern string returned by the algorithm for one classification path is a subsequence of U with skipped intervals. We provide a toy example to explain the skipping logic. Let $n = 4$, $\Sigma =$ {a, b, c} and $U =$ 'ab'. Strings are drawn from a product distribution such that x_1, x_2 and x_4 are uniformly distributed over Σ but x_2 is almost surely 'a'. The algorithm first estimates $\Pr[y = +1 \mid x_1 = a]$ for each $a \in \Sigma$ and finds the value of x_1 matters little to the positivity. It then estimates the distance between the distribution of x_1x_2 over all positive strings and that over all strings and finds the two distributions are close. However, when it moves on to estimate the distance between the distribution of $x_1x_2x_3$ over all positive strings and that over all strings, it gets a nonnegligible total variation distance. Therefore, the skipped interval is x_1x_2. The algorithm finally outputs the hypothesis pattern string 'ΣΣb' which means skipping the first two symbols and matching symbol 'b' in the rest of the string.

Theorem 4 *Under product distributions over instance space $\mathcal{I} = \Sigma^n$, given $\Pr[x_i = a] \geq t > 0$ for $\forall 1 \leq i \leq n$ and $\forall a \in \Sigma$, the algorithm PAC classifies any string that skips $C = O(1)$ intervals during the classification procedure with $O(sn^{C+2})$ conditional statistical queries from STAT$(\mathrm{III}, \mathcal{D})$ at tolerance*

$$\tau = \min\left\{\frac{\bar{\epsilon}_1^2}{384}, \frac{\bar{\epsilon}_1}{8sn}\right\}$$

or with $O(sn^{C+2})$ statistical queries from STAT$(\mathrm{III}, \mathcal{D})$ at tolerance

$$\bar{\tau} = (\epsilon' - 2\tau) \cdot \min\left\{\frac{t\bar{\epsilon}_1^2}{384}, \frac{\bar{\epsilon}_1}{8sn}\right\}$$

where $\bar{\epsilon}_1 = (\epsilon'/3^{C+2})^{2^C}$ and $\epsilon' = \epsilon/(2n^C)$.

Proof For the sake of the legitimacy and feasibility of the algorithm, we make an initial query to estimate the probability $\Pr[I_{V\sqsubseteq x} = h]$ for each h at tolerance τ. Denote $\epsilon' = \epsilon/(2n^C)$. If the answer is $\leq \epsilon' - \tau$, then $\Pr[I_{V\sqsubseteq x} = h] \leq \epsilon'$ is negligible and we won't consider such cases because any of them happens with probability $\leq \epsilon/2$. Otherwise we have $\Pr[I_{V\sqsubseteq x} = h] \geq \epsilon' - 2\tau$. With the lower bound assumption that $\Pr[x_i = a] \geq t > 0$ for $\forall 1 \leq i \leq n$ and $\forall a \in \Sigma$, the legitimacy and feasibility are assured. Thus bounding the classification error in the nonnegligible cases within $\epsilon/2$ establishes a total error bound ϵ. Because there are at most n^C nonnegligible cases, the problem reduces to bounding the classification error for each within ϵ'.

In the learning procedure, the algorithm skips an interval $x[i_1, i_2]$ given $I_{V\sqsubseteq x} = h$ based on the assumption that the interval $x[i_1, i_2]$ matches some segment next to V in the pattern string U. Let ι_g be the indicator for the event that the assumption is false in the first g skipped intervals and denote probability $\epsilon_g = \mathbf{E}\iota_g$. Let $\epsilon_0 = 0$. Note that ϵ_g serves as an upper bound for the probability of misclassification due to skipping the first g intervals, because there are some lucky cases where the assumption doesn't hold but the algorithm still makes correct classifications. To ensure the accuracy of the algorithm, it suffices to prove ϵ_g is small. Let $\bar{\epsilon}_{g+1} = 8\sqrt{3\epsilon_g}$ for $g \geq 1$ and $\bar{\epsilon}_1$ as defined in the theorem. We will prove $\epsilon_{g+1} \leq \bar{\epsilon}_{g+1}$ so that by induction and taking the minimum tolerance among all $g \leq C$ we then have the overall tolerances τ and $\bar{\tau}$ as claimed in the statement.

Let a_+, a'_+ be two (not necessarily distinct) symbols in the next element of U and a_-, a'_- be two (not necessarily distinct) symbols not in the next element of U. We have $|P(+|a_+, h) - P(+|a'_+, h)| \leq \epsilon_g$ and likewise $|P(+|a_-, h) - P(+|a'_-, h)| \leq \epsilon_g$. Let $P_i(+|a, h) = P(+|a, h, \iota_g = i)$ and denote $\Delta = P(+|a_+, h) - P(+|a_-, h)$ and $\Delta_i = P_i(+|a_+, h) - P_i(+|a_-, h)$ for $i \in \{0, 1\}$. As a consequence, $\Delta = \epsilon_g\Delta_1 + (1 - \epsilon_g)\Delta_0$ and $\Delta_0 = \frac{\Delta - \epsilon_g\Delta_1}{1-\epsilon_g} \geq \frac{\Delta-\epsilon_g}{1-\epsilon_g}$. Therefore, $\Delta > \epsilon_g$ implies $\Delta_0 > 0$. In the other direction, $\Delta_0 = \frac{\Delta-\epsilon_g\Delta_1}{1-\epsilon_g} \leq 2(\Delta + \epsilon_g)$.

For each h we make a statistical query to estimate $P(+|a, h)$ for each $a \in \Sigma$ at tolerance $\tau_1 = \bar{\epsilon}^2_{g+1}/384$. If the minimum Δ among all pairs of (a_+, a_-), denoted by $\Delta_{\min}$, is $> 6\tau_1$, the results of queries for $P(+|a, h)$ must form two distinguishable clusters, where the maximum difference inside one cluster is $\leq 4\tau_1$ while the minimum gap between two clusters is $> 4\tau_1$. According to Lemma 11, the set of symbols with larger query answers is the next element in U because $\Delta > \epsilon_g$ holds for all pairs of (a_+, a_-).

Otherwise, the difference $\Delta_0 \leq 2(\Delta_{\min} + 2\epsilon_g + \epsilon_g) \leq \bar{\epsilon}^2_{g+1}/16$ for all h. Let $x' = xz$ where z is an infinite string under the uniform distribution. Let $E_h(1, i)$ be the event that matching the next element in U consumes exactly i symbols in string x' given $I_{V\sqsubseteq x'} = h$ and $\iota_g = 0$. Define probability $R_h(1, i) = \Pr[E_h(1, i)]$. Let conditional probability $P_0(+|E_h(1, i))$ be the probability of positivity conditioned on event $E_h(1, i)$. For example, $P_0(+|a_+, h)$ is indeed $P_0(+|E_h(1, 1))$.

Denote by $P_0(+|h) = \Pr[y = +1 \mid I_{V\sqsubseteq x} = h \wedge \iota_g = 0]$. Because $P_0(+|h) \geq P_0(+|a_-, h)$, we have

$$P_0(+|a_+, h) - P_0(+|h) \leq P_0(+|a_+, h) - P_0(+|a_-, h) < \frac{\bar{\epsilon}^2_{g+1}}{16}$$

while

$$P_0(+|a_+, h) - P_0(+|h) = \sum_{i=1}^{+\infty} R_h(1, i) \cdot (P_0(+|E_h(1, 1)) - P_0(+|E_h(1, i)))$$

Notice that probability $P_0(+|E_h(1, i))$ is monotonically non-increasing with respect to i. Then there must exist an integer $k \in [1, +\infty]$ such that $P_0(+|E_h(1, 1)) - P_0(+|E_h(1, i)) \leq \bar{\epsilon}_{g+1}/4$ for $\forall i \leq k$ and $P_0(+|E_h(1, 1)) - P_0(+|E_h(1, i)) \geq \bar{\epsilon}_{g+1}/4$ for $\forall i > k$. This implies

$$\sum_{i\leq k} R_h(1, i) \left(P_0(+|E_h(1, 1)) - P_0(+|E_h(1, i))\right) + \sum_{i>k} R_h(1, i) \left(P_0(+|E_h(1, 1)) - P_0(+|E_h(1, i))\right) < \frac{\bar{\epsilon}^2_{g+1}}{16}$$

and

$$\frac{\bar{\epsilon}_{g+1}}{4} \sum_{i>k} R_h(1, i) < \frac{\bar{\epsilon}^2_{g+1}}{16}$$

Then we have $\sum_{i>k} R_h(1,i) < \bar{\epsilon}_{g+1}/4$. This means the next element in U almost surely shows up in this k-length interval. In addition, the difference $P_0(+|E_h(1,1)) - P_0(+|E_h(1,i)) \le \bar{\epsilon}_{g+1}/4$ for $\forall i \le k$ means whether the next element in U first shows up at x_{h+1} or x_{h+k} has little effect on the probability of positivity. There are two cases where an error happens due to skipping the interval. The first case is that the next element in U doesn't occur within the interval, whose probability is $\sum_{i>k} R_h(1,i)$. The second case is that after matching the next element in U at x_{h+i} for some $1 \le i < k$, the value of $x[h+i+1, h+k]$ flips the class of the string. This happens with probability $\le P_0(+|E_h(1,1)) - P_0(+|E_h(1,k))$. By union bound, the probability of the errors because of skipping the interval $x[h+1, h+k]$ is at most $\bar{\epsilon}_{g+1}/2$.

It is worth pointing out that k is an integer from 1 to $+\infty$ because when $i = 1$ the difference $P_0(+|E_h(1,1)) - P_0(+|E_h(1,i))$ is $0 \le \bar{\epsilon}_{g+1}/4$ and surely $k \ge 1$. This means this interval is not empty and ensures the existence of the interval we want. On the other hand, the value k can be positive infinity but this makes no difference because the algorithm will skip everything until the end of a string.

After showing the existence of such an interval, we need to determine k and locate the interval. Let $\mathcal{D}_k(h)$ be the distribution of x_{h+k} and $\mathcal{D}_{1:k}(h)$ be the distribution of the $x[h+1, h+k]$ over all strings, both conditioned on $I_{V \sqsubseteq x} = h$. Also, let $\mathcal{D}^+_k(h)$ and $\mathcal{D}^+_{1:k}(h)$ be the corresponding distributions over all positive strings. We use $\widehat{\cdot}$ as estimators for probabilities or distributions. The probability that an error happens due to skipping the next k letters is the total variation distance between $\mathcal{D}_{1:k}(h)$ and $\mathcal{D}^+_{1:k}(h)$. Recall that the total variation distance between two distributions μ_1 and μ_2 is

$$\|\mu_1 - \mu_2\|_{TV} = \frac{1}{2}\|\mu_1 - \mu_2\|_1 = \min_{(Y,Z)} \Pr[Y \neq Z]$$

where $Y \sim \mu_1$ and $Z \sim \mu_2$ are random variables over μ_1 and μ_2 respectively. The minimum is taken over all joint distributions (Y, Z) such that the marginal distributions are still μ_1 and μ_2, i.e., $Y \sim \mu_1$ and $Z \sim \mu_2$.

Now let $Y \sim \mathcal{D}_{1:k}(h)$ and $Z \sim \mathcal{D}^+_{1:k}(h)$ be random strings over $\mathcal{D}_{1:k}(h)$ and $\mathcal{D}^+_{1:k}(h)$ respectively. Then

$$\begin{aligned}
\|\mathcal{D}_{1:k}(h) - \mathcal{D}^+_{1:k}(h)\|_{TV} &= \min_{(Y,Z)} \Pr[Y \neq Z] \\
&= 1 - \max_{(Y,Z)} \Pr[Y = Z] \\
&= 1 - \max_{(Y,Z)} \prod_{i=1}^{k} \Pr[Y_i = Z_i] \\
&= 1 - \prod_{i=1}^{k} \max_{(Y,Z)} \Pr[Y_i = Z_i] \\
&= 1 - \prod_{i=1}^{k} \left(1 - \min_{(Y,Z)} \Pr[Y_i \neq Z_i]\right) \\
&= 1 - \prod_{i=1}^{k} \left(1 - \|\mathcal{D}_i(h) - \mathcal{D}^+_i(h)\|_{TV}\right)
\end{aligned}$$

because of the independence between the symbols in a string and the fact that all minimums and maximums are taken over all joint distributions (Y, Z) such that the marginal distributions are still product distributions.

Thus we could estimate the global total variation distance $\|\mathcal{D}_{1:k}(h) - \mathcal{D}^+_{1:k}(h)\|_{TV}$ through estimating the local variation distance $\|\mathcal{D}_i(h) - \mathcal{D}^+_i(h)\|_{TV}$ for each $1 \le i \le k$. Assume $\widehat{p}_1$ and $\widehat{p}_2$ are estimates of two probabilities p_1 and p_2 from a statistical query at some tolerance τ_0. We have

$$\begin{aligned}
|p_1 p_2 - \widehat{p}_1 \widehat{p}_2| &= |p_1 p_2 - p_1 \widehat{p}_2 + p_1 \widehat{p}_2 - \widehat{p}_1 \widehat{p}_2| \\
&= |p_1(p_2 - \widehat{p}_2) + (p_1 - \widehat{p}_1)\widehat{p}_2| \\
&\le p_1 |p_2 - \widehat{p}_2| + |p_1 - \widehat{p}_1| \widehat{p}_2 \\
&\le (p_1 + \widehat{p}_2)\tau_0 \le 2\tau_0
\end{aligned}$$

By induction it can be proved that $\left|\prod_{i=1}^{k} p_i - \prod_{i=1}^{k} \widehat{p}_i\right| \leq k\tau_0$, which is a polynomial bound. For a probability q, let q_i be the corresponding probability conditioned on $\iota_g = i$ for $i \in \{0, 1\}$. We have $q = \epsilon_g q_1 + (1 - \epsilon_g)q_0$ and

$$q_0 = \frac{q - \epsilon_g q_1}{1 - \epsilon_g} \geq q - \epsilon_g q_1 \geq q - \epsilon_g$$

In the other direction,

$$\begin{aligned} q_0 &= \frac{q - \epsilon_g q_1}{1 - \epsilon_g} = \frac{q + \epsilon_g - \epsilon_g^2 - \epsilon_g q - \epsilon_g + \epsilon_g^2 + \epsilon_g q - \epsilon_g q_1}{1 - \epsilon_g} \\ &= \frac{(q + \epsilon_g)(1 - \epsilon_g) - \epsilon_g(1 + q_1 - \epsilon_g - q)}{1 - \epsilon_g} \leq q + \epsilon_g \end{aligned}$$

Note that here without loss of generality, we assume $\epsilon \leq \min\{(n-1)t, 24/(sn)\}$ so that $1 + q_1 - \epsilon_g - q \geq (n-1)t - \epsilon_g + q_1 > 0$ and $\epsilon_g \leq \bar{\epsilon}_{g+1}^2/192 < \bar{\epsilon}_{g+1}/(8sn)$. In PAC learning model a polynomial upper bound for error parameter ϵ is trivial. Because if a learning algorithm works with a small error bound, it automatically guarantees larger error bounds. As a consequence, $|q - q_0| \leq \epsilon_g$. In addition, using the definition of $\|\cdot\|_{TV}$,

$$\begin{aligned} & \mid \|\mathcal{D}_i(h) - \mathcal{D}_i^+(h)\|_{TV} - \|\widehat{\mathcal{D}}_i(h) - \widehat{\mathcal{D}}_i^+(h)\|_{TV} \mid \\ =& \frac{1}{2} \mid \|\mathcal{D}_i(h) - \mathcal{D}_i^+(h)\|_1 - \|\widehat{\mathcal{D}}_i(h) - \widehat{\mathcal{D}}_i^+(h)\|_1 \mid \\ \leq& \frac{1}{2} \mid \|\mathcal{D}_i(h) - \mathcal{D}_i^+(h) - \widehat{\mathcal{D}}_i(h) + \widehat{\mathcal{D}}_i^+(h)\|_1 \mid \\ \leq& \frac{1}{2} \left(\|\mathcal{D}_i(h) - \widehat{\mathcal{D}}_i(h)\|_1 + \|\mathcal{D}_i^+(h) - \widehat{\mathcal{D}}_i^+(h)\|_1\right) \\ \leq& \frac{s}{2} \left(\|\mathcal{D}_i(h) - \widehat{\mathcal{D}}_i(h)\|_\infty + \|\mathcal{D}_i^+(h) - \widehat{\mathcal{D}}_i^+(h)\|_\infty\right) \end{aligned}$$

Hence, if we make statistical queries $\chi'_{V,a,h,i}$ and $\chi^+_{V,a,h,i}$ at tolerance $\tau_2 = \bar{\epsilon}_{g+1}/(8sn)$ and because $\bar{\epsilon}_{g+1}/(8sn) + \epsilon_g < \bar{\epsilon}_{g+1}/(4sn)$, the noise on $\|\mathcal{D}_i(h) - \mathcal{D}_i^+(h)\|_{TV}$ will be at most $\bar{\epsilon}_{g+1}/(4n)$ and we will be able to estimate $\|\mathcal{D}_{1:k}(h) - \mathcal{D}_{1:k}^+(h)\|_{TV}$ within error $k\bar{\epsilon}_{g+1}/(4n) \leq \bar{\epsilon}_{g+1}/4$. If $\|\widehat{\mathcal{D}}_{1:k}(h) - \widehat{\mathcal{D}}_{1:k}^+(h)\|_{TV} \geq 3\bar{\epsilon}_{g+1}/4$, then $\|\mathcal{D}_{1:k}(h) - \mathcal{D}_{1:k}^+(h)\|_{TV} \geq \bar{\epsilon}_{g+1}/2$. Otherwise, $\|\mathcal{D}_{1:k}(h) - \mathcal{D}_{1:k}^+(h)\|_{TV} < \bar{\epsilon}_{g+1}$ and we are still safe to increase k.

The algorithm does $O(sn^{C+2})$ queries $\chi_{V,a,h}$ at tolerance $\tau_1 = \bar{\epsilon}_{g+1}^2/384$, plus $O(sn^{C+2})$ queries $\chi'_{V,a,h,i}$ and $\chi^+_{V,a,h,i}$ at tolerance $\tau_2 = \bar{\epsilon}_{g+1}/(8sn)$. Thus by induction and taking the minimum tolerance among all $g \leq C$ we have the overall tolerances τ and $\bar{\tau}$ as claimed in the statement. ■

Appendix D Proof and details from Section 5

D.1 Proof of Lemma 4

Here we provide omitted proof and discussion of Lemma 4.

Lemma 4 (in the main paper) *Under general unrestricted string distributions, a concept class is PAC learnable over instance space $\Sigma^{\leq n}$ if and only if it is PAC learnable over instance space Σ^n.*

Proof *If direction.* Assume concept class $\mathcal{C}$ is PAC learnable from fixed-length strings with algorithm $\mathcal{A}$ under unrestricted general distributions. Because instance space $\Sigma^{\leq n} = \bigcup_{i \leq n} \Sigma^i$, we divide the sample S into n subsets $\{S_i\}$ where $S_i = \{x \mid |x| = i\}$. We make an initial statistical query to estimate probability $\Pr[|x| = i]$ for each $i \leq n$ at tolerance $\epsilon/(8n)$. We discard all S_i with query answer $\leq 3\epsilon/(8n)$, because we know $\Pr[|x| = i] \leq \epsilon/(2n)$. There are at most $(n-1)$ such S_i of low occurrence probabilities. The total probability that an instance comes from one of these ignored sets is at most $\epsilon/2$. Otherwise, $\Pr[|x| = i] \geq \epsilon/(4n)$ and we apply algorithm $\mathcal{A}$ on each S_i with query answer $\geq 3\epsilon/(8n)$ with error parameter $\epsilon/2$. Because the probability of the condition is

Input: N labeled strings $\langle x^i, y^i \rangle$, string length n, alphabet Σ
Output: pattern string $\widehat{u}$

$\widehat{u} \leftarrow \lambda$
for $\ell \leftarrow 0$ **to** n
  $reward \leftarrow 1 \times |\Sigma|$ all 0 vector
  for each $a \in \Sigma$
    for $i \leftarrow 1$ **to** N
      if $\widehat{u} \sqsubseteq x^i$
        if $y^i = +1$
          $reward[a] \leftarrow reward[a] +$
            $(I_{\widehat{u} \sqsubseteq x^i} - \min\{I_{\widehat{ua} \sqsubseteq x^i}, n+1\}) r_+$
        else
          $reward[a] \leftarrow reward[a] +$
            $(\min\{I_{\widehat{ua} \sqsubseteq x^i}, n+1\} - I_{\widehat{u} \sqsubseteq x^i}) r_-$
        endif
      else
        if $y^i = +1$
          return $\widehat{u}[1, \ell - 1]$
        endif
      endif
    endfor
  endforeach
  $\widehat{u}_{\ell+1} \leftarrow \mathrm{argmax}_{a \in \Sigma}\{reward[a]\}$
  $\widehat{u} \leftarrow \widehat{u}\widehat{u}_{\ell+1}$
endfor
return $\widehat{u}$

Figure 3: A greedy algorithm for learning ideal $\sqcup\!\sqcup$ from example oracle $EX(\sqcup\!\sqcup, \mathcal{D})$

polynomially large, the algorithm is feasible. Finally, the error over the whole instance space will be bounded by ϵ and concept class $\mathcal{C}$ is PAC learnable over instance space $\Sigma^{\leq n}$.

Only-if direction. This is an immediate consequence of the fact $\Sigma^n \subseteq \Sigma^{\leq n}$. ■

Notice that Lemma 4 requires algorithm $\mathcal{A}$ to be applicable to any $S_i \mid i \leq n$. But this requirement can be weakened. There might not exist such a general algorithm $\mathcal{A}$. Instead we could have an algorithm $\mathcal{A}_i$ applicable to each subspace S_i with non-negligible occurrence probability $\Pr[|x| = i] \geq \epsilon/(4n)$, then it is easy to see that Lemma 4 still holds in this case. Moreover, Lemma 4 makes no assumption on the string distribution. In the cases under restricted string distributions, here are two conditions that suffice to keep Lemma 4 hold: First, there is no assumption on the string length distribution; Second, we have an algorithm $\mathcal{A}_i$ applicable to instance space S_i over marginal distribution $\mathcal{D}_{|x|=i}$ for each $1 \leq i \leq n$ such that $\Pr[|x| = i]$ is polynomially large.

D.2 A heuristic greedy method

Figure 3 provides detailed pseudocode of the greedy method discussed in Section 5.

D.3 Experiment settings and results

To make a comparison between the greedy method and kernel machines for empirical performance, we conducted a series of experiments in MATLAB on a workstation built with Intel i5-2500 3.30GHz CPU and 8GB memory. As discussed in Section 5, the running time of the kernel machine will be intolerable in practice when the sample size N and the string length n are large. Also, a pattern string u of improper length will lead to a degenerate sample set which contains only positive or only negative example strings. To prevent this less interesting case from happening, we set $|u| = \lceil ns^{-1} \rceil$. Intuitively, the sample set will be evenly partitioned into two classes in expectation

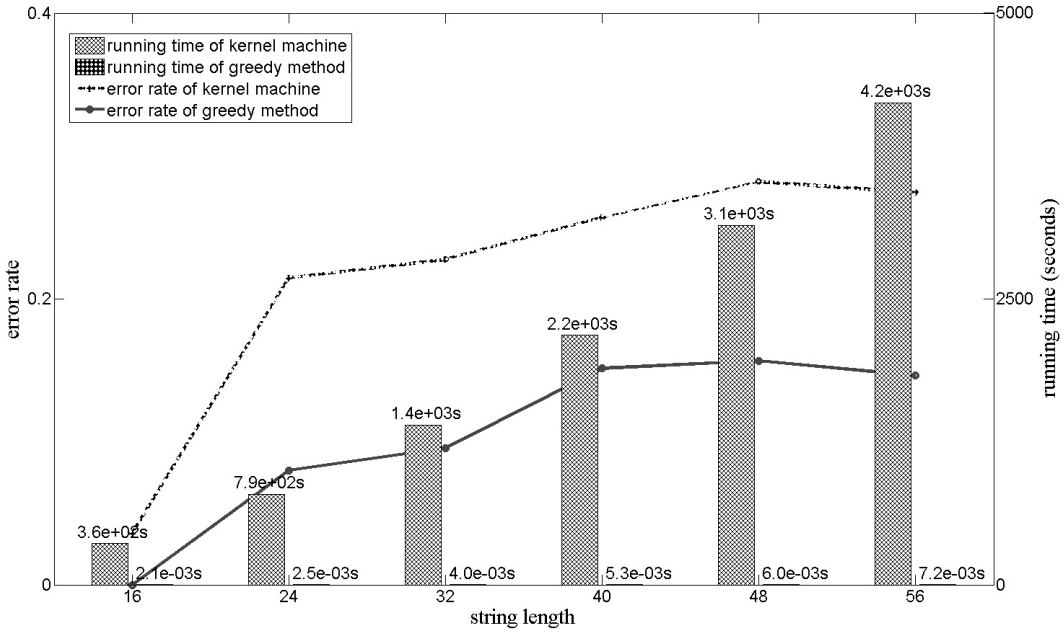

Figure 4: Experiment results with NSF abstracts data set (training 1993; testing 1992)

under the uniform distribution. However, in this case n not being large demands the alphabet size s not being large either.

Combining all these constraints together, the experiment settings are: alphabet size $s = 8$, size of training set = size of testing set = 1024. We vary the string length n from 16 to 56 and let $|u| = \lceil ns^{-1} \rceil$. The pattern string u is generated uniformly at random from $\Sigma^{|u|}$. Our tests are run on the NSF Research Award Abstracts data set [4]. We use the abstracts of year 1993 as the training set and those of year 1992 as the testing set. The tests are case-insensitive and all the characters except the subset from 'a(A)' to 'h(H)' are removed from the texts. The result texts are then partitioned into a set of strings of length n, which serve as the example strings. To be more robust against fluctuation from randomness, each test with a particular value of n is run for 10 times and the medians of error rates and running times are taken as the final performance scores. Both lines climb as n increases.

The experiment results are shown in Figure 4, with accuracy presented as line plot and efficiency demonstrated as bar chart. The overwhelming advantage of the greedy algorithm on efficiency is obvious. The kernel machine ran for hours in high dimensional cases, while the greedy method achieved even better accuracy within only milliseconds. The error rate of the greedy algorithm is always lower than that of the kernel machine as well.

It is worth noting that MATLAB started reporting no-convergence error of the kernel method when the string length n reaches 56. Only successful runs of the kernel method were taken into account. Therefore, the performance of the kernel method when $n = 56$ is very unstable over some datasets. Figure 5 is an example where kernel method became unpredictable when no-convergence error happened. In this plot when $n = 56$ the kernel machine seems to have better accuracy than the greedy method, but considering that all the failed runs of the kernel machine were ruled out and only successful ones were taken into account, the apparent accuracy of the kernel method is shaky.

References

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

[2] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, Nov. 1987.

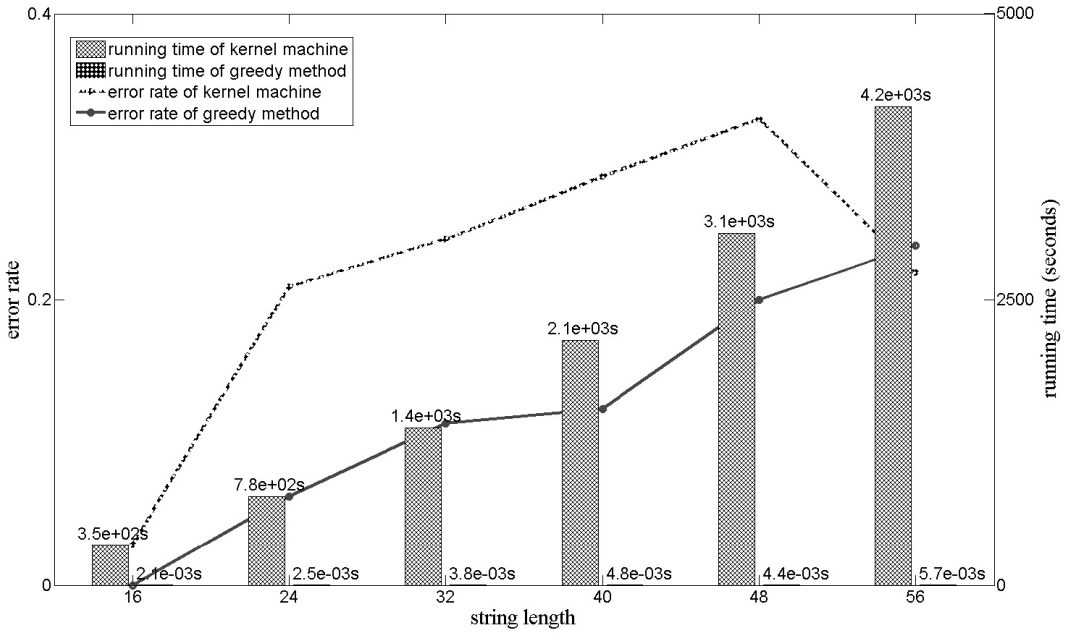

Figure 5: Experiment results with NSF abstracts data set (training 1999; testing 1998)

[3] D. Angluin, J. Aspnes, S. Eisenstat, and A. Kontorovich. On the learnability of shuffle ideals. *Journal of Machine Learning Research*, 14:1513–1531, 2013.

[4] K. Bache and M. Lichman. NSF research award abstracts 1990-2003 data set. *UCI Machine Learning Repository*, 2013.

[5] L. Bottou and C.-J. Lin. Support vector machine solvers. *Large scale kernel machines*, pages 301–320, 2007.

[6] N. H. Bshouty. Exact learning of formulas in parallel. *Machine Learning*, 26(1):25–41, Jan. 1997.

[7] C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332–1348, Sept. 2005.

[8] B. Gnedenko and A. N. Kolmogorov. Limit distributions for sums of independent random variables. *Addison-Wesley series in statistics*, 1949.

[9] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302 – 320, 1978.

[10] I. Ibragimov. On the composition of unimodal distributions. *Theory of Probability and Its Applications*, 1(2):255–260, 1956.

[11] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, Nov. 1998.

[12] O. Klíma and L. Polák. Hierarchies of piecewise testable languages. *Proceedings of the 12th International Conference on Developments in Language Theory*, pages 479–490, 2008.

[13] L. A. Kontorovich, C. Cortes, and M. Mohri. Kernel methods for learning languages. *Theoretical Computer Science*, 405(3):223–236, Oct. 2008.

[14] L. A. Kontorovich and B. Nadler. Universal kernel-based learning with applications to regular languages. *The Journal of Machine Learning Research*, 10:1095–1129, June 2009.

[15] L. A. Kontorovich, D. Ron, and Y. Singer. A markov model for the acquisition of morphological structure. *Technical Report CMU-CS-03-147*, 10, June 2003.

[16] K. Koskenniemi. Two-level model for morphological analysis. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, pages 683–685, 1983.

[17] M. Lothaire. *Combinatorics on Words (Encyclopedia of Mathematics and Its Applications - Vol 17)*. Addison-Wesley, 1983.

[18] M. Mohri. On some applications of finite-state automata theory to natural language processing. *Journal of Natural Language Engineering*, 2(1):61–80, Mar. 1996.

[19] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, June 1997.

[20] M. Mohri, P. J. Moreno, and E. Weinstein. Efficient and robust music identification with weighted finite-state transducers. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):197–207, Jan. 2010.

[21] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69 – 88, 2002.

[22] L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM (JACM)*, 40(1):95–142, Jan. 1993.

[23] O. Rambow, S. Bangalore, T. Butt, A. Nasr, and R. Sproat. Creating a finite-state parser with application semantics. *Proceedings of the 19th International Conference on Computational Linguistics - Volume 2*, pages 1–5, 2002.

[24] I. Simon. Piecewise testable events. *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, pages 214–222, 1975.

[25] R. Sproat, W. Gale, C. Shih, and N. Chang. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404, Sept. 1996.

[26] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, Nov. 1984.