Machine Learning, 9, 205-260 (1992)


# On the Computational Complexity of Approximating Distributions by Probabilistic Automata

NAOKI ABE
*Information Basic Research Laboratory, C&C Information Technology Research Laboratories, NEC Corporation, 4-1-1 Miyazaki, Miyamae-ku, Kawasaki 216 Japan*

MANFRED K. WARMUTH
*Computer Engineering and Information Sciences, University of California, Santa Cruz, CA 95064*

**Abstract.** We introduce a rigorous performance criterion for training algorithms for probabilistic automata (PAs) and hidden Markov models (HMMs), used extensively for speech recognition, and analyze the complexity of the training problem as a computational problem. The PA training problem is the problem of approximating an arbitrary, unknown source distribution by distributions generated by a PA. We investigate the following question about this important, well-studied problem: Does there exist an *efficient* training algorithm such that the trained PAs *provably converge* to a model close to an optimum one with high confidence, after only a feasibly small set of training data? We model this problem in the framework of computational learning theory and analyze the sample as well as computational complexity. We show that the number of examples required for training PAs is moderate—except for some log factors the number of examples is linear in the number of transition probabilities to be trained and a low-degree polynomial in the example length and parameters quantifying the accuracy and confidence. Computationally, however, training PAs is quite demanding: Fixed state size PAs are trainable in time polynomial in the accuracy and confidence parameters and example length, but *not* in the alphabet size unless **RP** = **NP**. The latter result is shown via a strong non-approximability result for the single string maximum likelihood model probem for 2-state PAs, which is of independent interest.

**Keywords.** Hidden Markov models, PAC learning model, density estimation, Kullback-Leibler divergence, computational learning theory

## 1. Introduction

We address the problem of approximating an arbitrary, unknown source distribution by distributions generated by probabilistic automata. Probabilistic automata (PAs), and hidden Markov models[1] (HMMs) which are closely related to PAs, are used extensively as models for probabilistic generation of speech signals for the purpose of speech recognition (see for example Levinson, Rabiner & Sondhi, (1983)). The problem addressed in the present paper corresponds to that of training a parameterized hidden Markov model for a particular spoken word with a set of actual speech signals for that word. In particular, we are interested in the question of whether there exists an algorithm that, when given a sample generated from an arbitrary unknown target distribution, outputs a probabilistic automaton that approximates the unknown distribution 'as closely as possible,' that is, with high probability the distribution induced by the output PA is sufficiently close to an 'optimal' one among all possible probabilistic automata satisfying a certain *prescribed constraint.* Here a *constraint* is given to the algorithm in the form of a subset of the state set specifying

the *legal initial states* and a labeled directed graph specifying the set of *legal transitions.* The training problem, therefore, is the problem of finding a near optimal setting of the initial and transition probabilities on the legal initial states and transitions in the input constraint. A *class* of constraints is said to be trainable with sample complexity $q(\ldots)$ if there exists an algorithm which trains every constraint in the class, and the sample size required for a given acuracy, confidence, length of the example strings, and the size of the input constraint is bounded above by the function $q$ of these parameters. Here the size of the input constraint translates to the number of probability parameters being trained. A class of constraints is said to be *polynomially trainable* if there is a training algorithm with polynomial sample complexity whose running time is polynomial in the total sample length. Of particular interest to us is the special case of this problem in which the input constraint is null, namely all initial states and transitions are legal. This special case translates to the problem of finding a near optimal probabilistic automaton with a *given number of states.*

Our model is a natural adaptation of the PAC-learning paradigm of Valiant (1984) and Blumer, et al. (1989) and is inspired by the model of efficient unsupervised learning of Laird (1988). It is also related to the models for learning languages from stochastic data in the limit proposed and studied by Angluin (1988). Our formulation requires the algorithm to be particularly *robust* in the sense that we do not assume anything about the target distribution—a formulation which is closely related to the 'robust' generalization of the PAC paradigm proposed by Haussler (1991). The distance measure between the distributions used in this paper to evaluate the accuracy of a hypothesis with respect to the target distribution is the well-known 'Kullback-Leibler divergence' (Kullback, 1967). Other commonly used measures of distance between probability distributions are, for example, the $\chi^2$ distance, the variation distance, the quadratic distance (Kearns & Schapire, 1990), and the Hellinger distance (Barron & Cover, 1989). The Kullback-Leibler divergence is a standard notion of distance, which enjoys many desirable properties (see Section 2). Furthermore, the Kullback-Leibler divergence is known to bound from above the Hellinger distance as well as half the square of the variation distance and of the quadratic distance. These relationships for the more general case of *conditional distributions* are surveyed by Yamanishi (1991).

Using this model, we give a number of results: We show that an arbitrary class of constraints is trainable by exhibiting a training algorithm whose sample comlexity is essentially linear in the size of the constraint being trained and a low-degree polynomial in the example length and parameters quantifying the accuracy and confidence. In addition, the running time of our training algorithm is polynomial in the total sample length if the size of the input constraint is bounded by a constant, thus showing that finite classes are polynomially trainable. In particular, an arbitrary *fixed* constraint is trainable in time polynomial in the accuracy and confidence parameters and the example length. If the alphabet size is variable, however, no polynomial time training algorithm exists for the class of 2-state null constraints[2], unless **RP** = **NP**.

To the best of our knowledge, our upper bound is the first rigorous result on the sample complexity of training PAs and HMMs, with respect to the classical measure of Kullback-Leibler divergence. Our proof is also interesting in the sense that we manage to get around the problem caused by the fact that the Kullback-Leibler divergence is unbounded. This property prohibits the direct use of certain useful techniques such as Hoeffding's inequality

for showing uniform convergence of empirical estimates of random variables to their true means. We get around this problem roughly as follows: We bound the smallest transition probabilities in the training algorithm's hypotheses from below by a decreasing function of the sample size $m$. We show uniform convergence for these successive classes of 'bounded' probabilistic automata using Hoeffding's inequality. We then show that for sufficiently large $m$, an optimum automaton in the $m$-th bounded class is close to an optimum one in the entire class with high probability. Interestingly, the trick of bounding probabilities away from zero is often used in practice in an attempt to solve what is known as the 'finite sample problem' (Levinson, Rabiner & Sondhi, 1983). Our result provides a rigorous justification for a particular way of setting those probability bounds, from the point of view of proving bounds on the sample complexity.

The sample complexity bound we obtain allows us to extend the classical equivalence between the minimization of the Kullback-Leibler divergence with respect to the *empirical* distribution and the maximization of the likelihood of the given data: We show that the polynomial time trainability of a class of constraints $C$ is equivalent to the polynomial time approximability of the 'maximum likelihood model' problem (MLM) for the same class $C$—the problem of setting the initial and transition probabilities in a given constraint in $C$ so that the probability assigned on a given finite sample is maximized. More precisely, we show that the polynomial time trainability of a class of constraints $C$ is equivalent to the approximability of the MLM problem for $C$ with a factor $1 + \epsilon$ in random time polynomial in $1/\epsilon$ and the size $t$ of the input constraint. Furthermore, we show that this latter notion of approximability of the MLM problem for $C$ is also equivalent to a seemingly much weaker notion of approximability: Approximability within factor $2^{p(n,t)m^{\alpha}}$ in random polynomial time, where $m$ is the sample size, $\alpha$ is an arbitrary constant less than 1, and $p(n,t)$ is a polynomial in the example length $n$ and the size of the input constraint $t$. We use the above equivalence between the training and MLM problems to show our hardness result: For *variable* alphabet size, the MLM problem for 2-state *null* constaints, or the problem of finding a 2-state PA assigning the maximum likelihood on the input sample, is hard to approximate (unless **P** = **NP**), and hence the class of 2-state null constraints is not polynomially trainable (unless **RP** = **NP**).

The hardness result for the MLM problem for 2-state null constaints is shown via the following non-approximability result for the *single string* MLM problem for the same class—the special case of MLM in which the input sample consists of a single string. We show that it is hard to approximate the single string MLM problem for the 2-state null constraints within a factor of $2^{|w|^{1-\alpha}}$ for any positive constant $\alpha$, where $w$ is the input word, in time polynomial in the word length and alphabet size, unless **P** = **NP**. Note that it is a very strong non-approximability result[3], since there is a trivial training algorithm, using only 1-state probabilistic automata, that can guarantee approximation within a factor of $2^{|w|}$ of the best 2-state PA. The proof of the hardness result uses as a starting point the type of technique commonly used in the learning theory literature for showing the hardness of a 'sample consistency' or 'minimum consistent concept' problem in discrete domains such as automata and boolean formulas (Gold, 1978; Angluin, 1978; Pitt & Warmuth, 1989). In particular, our proof makes use of notions used in Angluin's proof of the NP-completeness of the sample consistency problem for 2-state DFA (Angluin, 1989). The proof given here is, however, significantly more complex than the proof of the discrete

case, since corresponding to 'consistency' we have 'probability,' which is continuous and is thus much harder to get a hold of. For example, in our reduction of the satisfiability problem to the MLM problem for the 2-state null constraints, it is already non-trivial to formalize how a truth assignment is is to be simulated by a PA. We let each truth assignment correspond *conceptually* to one of $2^n$ many *deterministic* PAs of a *particular* kind, and for all the other (infinitely many) PAs, we quantify how 'far' they are from those corresponding to truth assignments. We then show that any PA that assigns the input word $w$ a probability at least $1/2^{|w|^{1-\alpha}}$ times the probability assigned on $w$ by an optimum PA must be 'close' to a deterministic PA corresponding to a satisfying assignment.

This paper is outlined as follows. We begin in Section 2 with some preliminary definitions and give the proof of the sample size bounds for training PAs in Section 3. In Section 4 we show the equivalence between the training problem and the approximate MLM problem for any class of constraints. In Section 5 we give the hardness result for the single string MLM problem for 2-state null constraints. Parts of this lengthy proof are given in Appendices A and B. In Section 6 we discuss briefly how the results of this paper apply to HMMs. We conclude by discussing a number of open problems inspired by this research in Section 7.

## 2. Preliminaries

This paper deals with approximating a probability distribution over words over some finite alphabet, $\Sigma$. For simplicity, we assume that all words with positive probability have the same length, $n$, i.e. the distribution is over the domain $\Sigma^n$. We call an element of $\Sigma^n$ an *example*. A *sample* $\Xi$ of $\Sigma^n$ is a finite sequence of examples of $\Sigma^n$, $\Xi = \langle w_1, \ldots, w_m \rangle$, where $m$ is the sample size. We abuse notation and write $x \in \Xi$ to mean that $x$ appears in the sequence $\Xi$. We let $\#(x, \Xi)$ denote the number of occurrences of example $x$ in sample $\Xi$. Using the above notation, we define the notion of the empirical distribution of a sample.

**Definition 2.1.** *Given a sample $\Xi$ of size $m$ of $\Sigma^n$, the empirical distribution of $\Xi$ over $\Sigma^n$, written $\hat{D}_\Xi$, is defined by:*

$$\forall x \in \Sigma^n \; \hat{D}_\Xi(x) = \frac{\#(x, \Xi)}{m}$$

*Note that for any $y$ not in $\Xi$, we have $\hat{D}_\Xi(y) = 0$.*

The probabilistic automation is formalized as a stochastic matrix $M$ together with an 'initial distribution' $\pi$ over the set of states. Intuitively, the probabilistic automation is much like a non-deterministic finite state automaton except that the transitions take place with probabilities prescribed by $M$. (See Figure 1.) To start the process, the machine chooses the initial state according to the initial distribution $\pi$, and then at any given point after that, the machine is in some state $i$, and at the next time step moves to another state $j$ outputting some letter $z$, with probability specified by $M(i, j, z)$. If one stops the machine

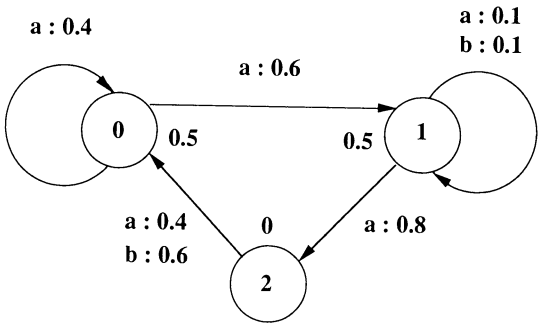


*Figure 1.* An example probabilistic automaton.

at time step $n$ the machine ends in some state having generated a string of length $n$. In this way, a probabilistic automaton naturally defines a probability distribution over the set of strings of length $n$, for any particular $n$.

**Defintion 2.2 (Probabilistic Automata (PA)).** *A probabilistic automaton $P$ is a quadruple $\langle S_P, \Sigma_P, \pi_P, M_P \rangle$ where $S_P$ is a finite set of states, $\Sigma_P$ is a finite alphabet, $\pi_P : S_P \to [0, 1]$ is a probability distribution over $S_P$, and $M_P : S_P \times S_P \times \Sigma_P \to [0, 1]$ is a stochastic matrix*[4]*, i.e.*

$$\sum_{i \in S_P} \pi_P(i) = 1 \text{ and } \forall i \in S_P \sum_{j \in S_P, z \in \Sigma_P} M_P(i, j, z) = 1 \tag{2.1}$$

*Each $\pi_P(i)$ is called an initial probability, and each $M_P(i, j, z)$ is called a transition probability. For any string $w = w_1 \ldots w_n \in \Sigma_P^*$, the generation probability assigned on it by $P = \langle S_P, \Sigma_P, \pi_P, M_P \rangle$ is computed as follows.*

$$P(w_1 \ldots w_n) = \sum_{\langle i_0, \ldots, i_n \rangle \in S_P^{n+1}} \pi_P(i_0) \cdot \prod_{j=0}^{n-1} M_P(i_j, i_{j+1}, w_{j+1}) \tag{2.2}$$

*Thus, for any given example length $n$, $P$ defines a probability distribution over $\Sigma_P^n$.*

For example, the probability assigned by the probabilistic automaton $P$ shown in Figure 1 on the string $w = aab$ is calculated as follows:

$$\begin{aligned} P(w) = {} & \pi_P(0) \cdot M_P(0, 0, a) \cdot M_P(0, 1, a) \cdot M_P(1, 1, b) \\ & + \pi_P(0) \cdot M_P(0, 1, a) \cdot M_P(1, 1, a) \cdot M_P(1, 1, b) \\ & + \pi_P(0) \cdot M_P(0, 1, a) \cdot M_P(1, 2, a) \cdot M_P(2, 0, b) \\ & + \pi_P(1) \cdot M_P(1, 1, a) \cdot M_P(1, 1, a) \cdot M_P(1, 1, b) \\ & + \pi_P(1) \cdot M_P(1, 1, a) \cdot M_P(1, 2, a) \cdot M_P(2, 0, b) \end{aligned}$$

$$= 0.5 \cdot 0.4 \cdot 0.6 \cdot 0.1 + 0.5 \cdot 0.6 \cdot 0.1 \cdot 0.1 + 0.5 \cdot 0.6 \cdot 0.8 \cdot 0.6 + 0.5 \cdot 0.1 \cdot 0.1 \cdot 0.1 + 0.5 \cdot 0.1 \cdot 0.8 \cdot 0.6$$
$$= 0.012 + 0.003 + 0.144 + 0.0005 + 0.024 = 0.1835$$

Note that for a probabilistic automaton $P$ we use the same letter $P$ to denote the probability distribution defined by $P$ on $\Sigma_P^n$, where $n$ will be clear from the context. A *PA constraint* is a quadruple $C = \langle \Sigma, S, I, G \rangle$ where *$I$ is the inital state set* and $G$ is the transition graph of $C$. $I$ is a subset of the set $S$ of all states. $G$ is a subset of the set $S \times S \times \Sigma$ of all *transitions.* Note that a transition graph is a labeled directed graph with the state set $S$ as the vertices, and the alphabet $\Sigma$ as the set of labels. We write $|I|$ for the number of states in $I$, $|G|$ for the number of transitions in $G$. We then define the *size* of a constraint $C$, written $|C|$, as $|C| = |I| + |G|$. Note that the size of $C$ corresponds to the number of probability parameters included in $C$.

We say that there is a path in $C$ for a string $w_1 \ldots w_n \in \Sigma^n$, if there is a sequence of states $\langle i_0, \ldots, i_n \rangle$ from $S^{n+1}$ such that $i_0 \in I$ and for all $j$, $1 \leq j \leq n$, $(i_{j-1}, i_j, w_j) \in G$. We say that a probabilistic automaton $P$ *satisfies* a constraint $C = \langle \Sigma, S, I, G \rangle$, if and only if $S_P = S$ and $\Sigma_P = \Sigma$ and

$$\forall i \notin I,\ \pi_P(i) = 0 \text{ and}$$

$$\forall (i, j, z) \notin G,\ M_P(i, j, z) = 0 \tag{2.3}$$

Note that if a probabilistic automaton $p$ satisfies a constraint $C$ then for every word on which $P$ assigns a positive probability, there is a path for it in $C$. If $P$ satisfies $C$, one can think of $\pi_P$ as a function from $I$ into [0, 1], and $M_P$ as a function from $G$ into [0, 1]. We let $\mathcal{PA}(C)$ denote the class of probabilistic automata satisfying the constraint $C$. Note that $\mathcal{PA}(C) \subset [0, 1]^I \times [0, 1]^G$, where we let $[0, 1]^I$ denote the class of all functions from $I$ into [0, 1], and $[0, 1]^G$ from $G$ into [0, 1]. We also let $\mathcal{M}(G)$ denote the class of stochastic matrices $M_P$ satisfying (2.3).

The notion of 'distance' among distributions we employ in this paper is a well-known measure in information theory called 'Kullback-Leibler divergence,' also known as the 'relative entropy.'

**Definition 2.3 (Kullback-Leibler Divergence).** *Let $D$ and $Q$ be probability distributions over countable domain $X$. The 'Kullback-Leibler divergence' of $Q$ with respect to $D$, $d_{KL}(D, Q)$ is defined as follows.*

$$d_{KL}(D, Q) = \sum_{x \in X} D(x) \log \frac{D(x)}{Q(x)}$$

*(Normally we think of $D$ as the actual distribution against which a 'candidate' distribution $Q$ is being compared. By convention, we let* $0 \log 0 = 0$, *and* $0/0 = 1$.)

Note in the above definition that the base of logarithm is 2, following the conventions in coding theory. (Throughout this paper we let ln denote the natural logarithm and log the logarithm base 2.) The Kullback-Leibler divergence enjoys many natural properties: We can rewrite $d_{KL}(D, Q)$ as $E_D(\log 1/Q(\cdot)) - H(D)$ where $E_D(\log 1/Q(\cdot))$ is the expectation according to $D$ of the random variable $\log 1/Q(\cdot)$, and $H(D)$ is the *entropy*[5] of $D$, defined as $\Sigma_{x \in X} D(x) \log 1/D(x)$. Recall that $\log 1/D(x)$ is the code length for $x$ with respect to the "ideal code"[6] for $D$, and $H(D)$ is the expected code length of that code for the source distribution $D$ (see Hamming (1986)). In other words, for the source distribution $D$, the divergence $d_{KL}(D, Q)$ measures the expected additional code length required when using the ideal code for $Q$ instead of the ideal code for $D$. Thus, the ideal code of the distribution which minimizes the Kullback-Leibler divergence with respect to the source distribution also minimizes the expected code length of the future data. It is also well-known that minimizing the Kullback-Leibler divergence with respect to the *empirical* distribution $\hat{D}_{\Xi}$ observed in a sample $\Xi = \langle w_1, \ldots, w_m \rangle$ (Definition 2.1) corresponds to maximizing the likelihood of the sample, as demonstrated below. Minimizing $d_{KL}(\hat{D}_{\Xi}; Q)$ corresponds to minimizing $E_{\hat{D}_{\Xi}}(\log 1/Q(\cdot))$, and the following always holds:

$$E_{\hat{D}_{\Xi}}\left[\log \frac{1}{Q(\cdot)}\right] = \frac{1}{m} \cdot \sum_{i=1}^{m} \log \frac{1}{Q(w_i)} = \frac{1}{m} \log \prod_{i=1}^{m} \frac{1}{Q(w_i)} \tag{2.4}$$

Thus, $d_{KL}(\hat{D}_{\Xi}; Q)$ is minimized when $\Pi_{i=1}^{m} Q(w_i)$ is maximized, i.e. when $Q$ maximizes the probability of having generated the sample. We summarize this as a lemma.

**Lemma 2.1.** *Let $\mathcal{P}$ be an arbitrary class of distributions over $\Sigma^n$, $\Xi = \langle w_1, \ldots, w_m \rangle$ an arbitrary sample of $\Sigma^n$, and $\hat{D}_{\Xi}$ the empirical distribution of $\Xi$. Then,*

$$E_{\hat{D}_{\Xi}}\left[\log \frac{1}{Q(\cdot)}\right] = \inf\{E_{\hat{D}_{\Xi}}\left[\log \frac{1}{P(\cdot)}\right] : P \in \mathcal{P}\}$$

*if and only if*

$$\prod_{i=1}^{m} Q(w_i) = \sup\{\prod_{i=1}^{m} P(w_i) : P \in \mathcal{P}\}$$

Below we give the definition of a training algorithm which is the central definition in this paper. Here we assume that a *randomized algorithm* has access to a fair coin and can flip it in a single time unit.

**Definition 2.4 (Training PA Constraints).** *A training algorithm takes as input a constraint $C$, a string length $n$, and a finite sample $\Xi$ of $\Sigma^n$ for some alphabet $\Sigma$, and outputs a probabilistic automaton which satisfies $C$. We say that a (possibly randomized) training algorithm $A$ trains a class of constraints $\mathcal{C}$ with sample size $q(1/\epsilon, 1/\delta, n, t)$, if $A$, when given as*

*input an arbitrary constraint* $C \in \mathcal{C}$ *of size* $t$, *a string length* $n$, *and a finite sample* $\Xi$ *drawn independently at random from an arbitrary unknown distribution* $D$ *over* $\Sigma^n$, *is such that whenever the sample size* $m$ *exceeds* $q(1/\epsilon, 1/\delta, n, t)$, *then provided that* $inf\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$ *is finite, A's output* $Q$ *satisfies the following with probability at least* $1 - \delta$:

$$d_{KL}(D, Q) - d_{KL}(D, Opt) \leq \epsilon$$

*where Opt is a member of* $\mathcal{PA}(C)$ *satisfying:*

$$d_{KL}(D, Opt) = min\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$$

*Here the probability is taken over the product distribution of D producing the sample and the random coin flips of A, if A is randomized. If there exists such a training algorithm then we say that* $\mathcal{C}$ *is trainable with sample complexity* $q(1/\epsilon, 1/\delta, n, t)$. *If there exists a training algorithm with a* polynomial *sample complexity which also runs in time* polynomial *in the total sample length, then we say that* $\mathcal{C}$ *is polynomially trainable.*

Note that $inf\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$ is infinite if and only if there is a word $x \in \Sigma^n$ such that $D(x) > 0$ and there is no path in $C$ for $x$. In this case, $d_{KL}(D, P) = \infty$ for any $P$ in $\mathcal{PA}(C)$. We still need to verify that *Opt* in the above definition is well-defined, when $inf\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$ is finite. Define a function $\xi_D : \mathcal{PA}(C) \to [0, 1]$ for an arbitrary target distribution $D$ by:

$$\xi_D(P) = \prod_{x \in \Sigma^n} P(x)^{D(x)}$$

Then since $\xi_D$ is a continuous function, for an arbitrary $D$, mapping a compact subset of the parameter space $[0, 1]^I \times [0, 1]^G$, it attains a maximum at a particular PA, say *Opt'*:

$$\xi_D(Opt') = max\{\xi_D(P): P \in \mathcal{PA}(C)\}$$

Hence,

$$\begin{aligned} E_D\left[\log \frac{1}{Opt'(\cdot)}\right] &= \sum_{x \in \Sigma^n} D(x) \log \frac{1}{Opt'(x)} \\ &= \log \frac{1}{\xi_D(Opt')} \\ &= \log \frac{1}{max\{\xi_D(P) : P \in \mathcal{PA}(C)\}} \\ &= min\{\log \frac{1}{\xi_D(P)} : P \in \mathcal{PA}(C)\} \\ &= min\{E_D\left[\log \frac{1}{P(\cdot)}\right] : P \in \mathcal{PA}(C)\} \end{aligned} \tag{2.5}$$

Noting that $d_{KL}(D, P) = E_D(\log 1/P(\cdot)) - H(D)$, the equality (2.5) implies that $d_{KL}(D, Opt')$ equals $min\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$. Now let $Opt = Opt'$ and we see that $Opt$ is well-defined.

We define two versions of the maximum likelihood model problem for PAs considered in this paper.

**Definition 2.5 Sample MLM Problem for** $C$

*Input: A constraint $C \in \mathcal{C}$, a string length $n$ and a finite sample $\Xi = \langle w_1, \ldots, w_m \rangle$ of strings from $\Sigma^n$.*

*Output: A probabilistic automaton $Q$ satisfying $C$ which asigns the maximum genration probability (or the maximum likelihood) on $\Xi$ among all such probabilistic automata, i.e.*

$$\prod_{i=1}^{m} Q(w_i) = max\left\{\prod_{i=1}^{m} P(w_i) : p \in \mathcal{PA}(C)\right\}$$

Again note that $max\{\Pi_{i=1}^{m} P(w_i) : P \in \mathcal{PA}(C)\}$ is well-defined because $\zeta : \mathcal{PA}(C) \to [0, 1]$ defined by

$$\zeta(P) = \prod_{i=1}^{m} P(w_i)$$

is a continuous function mapping a compact domain $\mathcal{PA}(C)$ into the range $[0, 1]$.

The following definition is a special case of the sample maximum likelihood model problem in which the input sample consists of a single string. Note that for a single string, the initial probability distribution plays no significant role, because among probabilistic automata assigning the maximum probability on a given string there is always a probabilistic automaton in which exactly one state has initial probability one and other states have probability zero.

**Definition 2.6** *For a stochastic matrix $M$ and a word $w$, let $M(w)$ be the maximum generation probability assignable on $w$ by $M$ with the* best *initial state.*

Note that with this definition $M(uv) \leq M(u) \cdot M(v)$, in general. The single string MLM problem is defined as the problem of finding a *stochastic matrix $M$* satisfying the input constraint, which maximizes $M(w)$ on the input string $w$.

**Definition 2.7 Single-String MLM Problem for** $C$

*Input: A constraint $C = \langle \Sigma, S, I, G \rangle \in \mathcal{C}$ and a string $w$ in $\Sigma^*$.*

*Output: A stochastic matrix $M^*$ satisfying $G$ which assigns the maximum generation probability on $w$ among all such stochastic matrices, i.e.*

$$M^*(w) = max\{M(w) : M \in \mathcal{M}(G)\}$$

As usual let **P** denote the class of decision problems decidable in polynomial time and **NP** the class of decision problems acceptable in non-deterministic polynomial time. **RP** denotes the class of decision problems that are acceptable in random polynomial time (Gill, 1977): A decision problem $L$ is said to be accepted in random polynomial time if and only if there exists a randomized algorithm $A$, that is, $A$ has access to a fair coin, such that $A$ halts in polynomial time on all inputs, and $A$ always outputs 'no' on a negative instance and outputs 'yes' with probability at least a half on a positive instance. It is widely conjectured that **P** is strictly contained in **NP**, and also that **RP** is strictly contained in **NP**. All hardness results of this paper only hold modulo one of the above conjectures.

## 3. Sample complexity bounds for training PAs

Our main positive result on the training problem is the following bound on the sample complexity of the PA training problem.

**Theorem 3.1.** *An arbitrary class of PA constraints $C$ is trainable with sample complexity $O((n/\epsilon)^2 t \cdot \log^3 nt/\epsilon \cdot \log 1/\delta \cdot \log^2 \log 1/\delta)$, where $t$ is the size of the input constraint.*

Note that the above sample complexity bound is essentially linear in the size of the input constraint $t$, and a low-order polynomial in $n$, $1/\epsilon$, and $\log 1/\delta$. As an easy corollary, the following bound on the sample complexity of the training problem for the null constraints follows.

**Corollary 3.1.** *The class of null PA constraints is trainable with sample size: $O((n/\epsilon))^2 s^2 a \cdot \log^3 nsa/\epsilon \cdot \log 1/\delta \cdot \log^2 \log 1/\delta)$, where $s$ is the number of states and $a$ is the alphabet size of the null constraint to be trained.*

### Outline of the proof of Theorem 3.1

Let an input constraint $C = \langle \Sigma, S, I, G \rangle \in C$ be given, and let $t$ be its size, i.e. $t = |I| + |G|$. Here $I$ is a subset of the set $S$ of all states, and $G \subseteq S \times S \times \Sigma$. Assume that $min\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$ is finite with respect to the target probability distribution $D$, since if the minimum is infinite then by the definition of trainability any sample complexity suffices. Our objective is to show that there exists a training algorithm such that for any sufficiently large sample $\Xi$, its output $Q \in \mathcal{PA}(C)$ is likely to approximately minimize $d_{KL}(D, Q)$, where $D$ is the source distribution. Recall that $d_{KL}(D, Q) = E_D(\log 1/Q(\cdot)) - H(D)$. Since the second term (the entropy of $D$) is independent of $Q$, in order to find a $Q$ that minimizes $d_{KL}(D, Q)$, it suffices to find a $Q$ that minimizes $E_D(\log 1/Q(\cdot))$. Thus a natural attempt would be to find a $Q$ that minimizes $E_{\hat{D}_\Xi}(\log 1/Q(\cdot))$ and show that the empircial estimates converge to their true means *uniformly* for the class of random variables $\mathcal{F}(\mathcal{PA}(C)) = \{\log 1/P(\cdot) : P \in \mathcal{PA}(C)\}$ for moderate sample size. The difficulty here is the fact that $\mathcal{F}(\mathcal{PA}(C))$ is unbounded in the sense that $\log 1/P(x)$ diverges to infinity with $P(x)$ goes to zero. This fact prohibits the direct application of certain lemmas on the convergence of bounded random variables, such as Hoeffding's inequality.