Figure C.3: The deterministic automaton corresponding to ‘true.’

Figure C.4: The deterministic automaton corresponding to ‘false.’

Figure C.2: The four boolean functions on two variables.

Figure C.1: An example probabilistic automaton.

If we divide the path set for $u_i$ into $\Omega_1$ and $\overline{\Omega_1}$, defined as before, then if we let $M_\tau$ be an arbitrary canonical matrix, we can show the following

$$\begin{aligned} M(u_i) &= M(u_i, \Omega_1) + M(u_i, \overline{\Omega_1}) \\ &\leq (1 - \delta^{(a,b,c_i,d_i,x_i,\overline{x_i},e)}(M) + \frac{1}{2}\lambda^{(a,b,c_i,d_i,x_i,\overline{x_i},e)}(M, M_\tau))C[M](u) \\ &= (1 - \frac{\delta^{(a,b,c_i,d_i,x_i,\overline{x_i},e)}(M))}{2})C[M](u) \end{aligned} \tag{B.21}$$

**Proof of Lemma 5.6, part 7**

As we observed in the proof sketch, if $F$ is unsatisfiable, then for any canonical stochastic matrix $M_\tau$, we must have $M_\tau(w_5) = 0$. But we have shown that $\delta(M) \leq \frac{1}{s2^{10}}$. In other words, for some particular truth assignment $\tau$, $\delta(M, M_\tau) \leq \frac{1}{s2^{10}}$. Therefore we can apply Lemma 5.5 to each substring $u = \prod_{j=1}^{s} ab(l_{j,1}l_{j,2}l_{j,3})b$ and obtain the following.

$$M(u) \leq 6\frac{1}{s2^{10}}(1 + \sqrt{\frac{1}{s2^{10}}})^6 C(u) \tag{B.22}$$

Clearly $M(u) \leq (\frac{1}{2})C[M](u)$ holds, and thus we obtain $M(w_5) \leq (\frac{1}{2})^{k_5}C[M](w_5)$.
**End of proof of Lemma 5.6**

## C Figures

Sorry, the figures are missing. If you need them please ask for full copy from

Jean McKnight
Dep. of Computer Science
225 Applied Sciences
Santa Cruz, CA 95064

Putting (B.16) and (B.18) together, we obtain:

$$\begin{aligned} M(u) &= M(u,\Omega_1) + M(u,\overline{\Omega_1}) \\ &\leq (1 - \delta^{(a,b,f)}(M) + \frac{1}{2}\lambda^{(a,b,f)}(M,M_\tau))C[M](u) \\ &= (1 - \frac{\delta^{(a,b,f)}(M)}{2}(M))C[M](u) \end{aligned}$$

**Proof of Lemma 5.6, part 4**

This part follows from an analogous argument to the proof of Lemma 5.6, part 1, except the part that eliminates two of the four possible 'cycles,' since that part makes use of the fact that the frequency of $a$ and $b$ in $w$ is very close to one. First, if we define $\Lambda_i$ for each $i \leq n$, analogously to $\Lambda$ in the earlier proof, then we can show that $min_{i\leq n}\Lambda_i \leq \frac{1}{s2^{10}}$. We can also show that $\nu^{(c_i,d_i)}(M) \leq \frac{1}{s2^{10}}$ by the same argument. So, it follows that if we let $\Delta = min_{i\leq n} min\{\delta^{(c_i,d_i)}(M,\{c_i : 0 \to 0, d_i : 0 \to 0\}), \delta^{(c_i,d_i)}(M,\{c_i : 0 \to 1, d_i : 1 \to 0\}), \delta^{(c_i,d_i)}(M,\{c_i : 1 \to 0, d_i : 0 \to 1\}), \delta^{(c_i,d_i)}(M,\{c_i : 1 \to 1, d_i : 1 \to 1\})\}$, then we have that if $M(w) \geq \frac{C(w)}{K}$ then $\Delta \leq \frac{1}{s2^{10}}$.

**Proof of Lemma 5.6, part 5**

There are two cases. First, suppose that for some $i$, $(c_i, d_i)$ pair is *not* either $(0 \to 1, 1 \to 0)$ or $(1 \to 0, 0 \to 1)$. More precisely, suppose that one of the following holds:

$$\Delta_i = \delta^{(c_i,d_i)}(M,\{c_i : 1 \to 1, d_i : 1 \to 1\}) \leq \frac{1}{s2^{10}} \tag{B.19}$$

$$\Delta_i = \delta^{(c_i,d_i)}(M,\{c_i : 0 \to 0, d_i : 0 \to 0\}) \leq \frac{1}{s2^{10}} \tag{B.20}$$

In this case, each of the $2^7$ pathes for the string $w_{3,i} = c_i f c_i d_i f d_i$ must contain at least two *unintended* transitions, that is, two transitions other than $f : 0 \to 1$, $f : 1 \to 0$, $c_i : 1 \to 1$, $d_i : 1 \to 1$ when (B.19) holds, and two transitions other than $f : 0 \to 1$, $f : 1 \to 0$, $c_i : 0 \to 0$, $d_i : 0 \to 0$, when (B.20) holds. Since all of $\Delta_i$, $\lambda^{(f)}(M, M_\tau) \leq \delta^{(f)}(M, M_\tau)$, and $\nu^{(c,d,f)}(M) \leq max\{\Delta_i, \delta^{(f)}(M, M_\tau)\}$ are at most $\frac{1}{s2^{10}}$, we can bound $M(w_{3,i})$ from above as follows.

$$\begin{aligned} M(w_{3,i}) &\leq 2^7(max\{\Delta_i, \lambda^{(f)}(M,M_\tau)\})^2 \cdot (1 + \nu^{(c,d,f)}(M))^6 (\frac{M(*,*,(c_i,d_i))}{2})^4 (\frac{M(*,*,f)}{2})^2 \\ &\leq \frac{1}{2}C[M](w_{3,i}) \end{aligned}$$

It follows that in fact in this case $M(w_{3,i}) \leq (\frac{1}{2})^{k_3} C[M](w_{3,i})$. Next, we assume that each $(c_i, d_i)$ pair is set as intended, namely $(0 \to 1, 1 \to 0)$ or $(1 \to 0, 0 \to 1)$. Given this, we can prove the analogue of Claim B.1 in the proof of Lemma 5.6, part 3, where the string $afabfb$ is replaced by $c_i f c_i d_i f d_i$. The desired conclusion follows at once.

**Proof of Lemma 5.6, part 6**

The proof of part 6 is similar to the proof of part 3. Using the same technique of dividing the set of pathes for each substring of the form $u_i = a x_i b x_i c_i \overline{x_i} d_i \overline{x_i} b z a z a b$ we can obtain the following claim, exactly analogous to the claim in the earlier proof.

**Claim B.2:** *Let* $u_i = a x_i b x_i c_i \overline{x_i} d_i \overline{x_i} b z a z a b$. *Then, if* $\delta^{(a,b,c_i,d_i)}(M) \leq (\frac{1}{2})^{10}$ *then* $M(u_i) \leq (1 - \frac{\delta^{(a,b,c_i,d_i,x_i,\overline{x_i},e)}(M)}{2})C[M](u_i)$.

one of these six places as the place for the first occurrence of two consecutive unintended transitions, then the total probability of generating $u$ via these pathes is less than $C[M](u)$ by an approximate factor of $\lambda^{(a,b)}(M, M_\tau)$ times $\lambda^{(a,b,f)}(M, M_\tau)$, accounting for the presence of unintended transitions at two specific positions, disregarding any favorable skews that might be present. For example, if we let $\overline{\Omega_{1,1}}$ denote the set of pathes in which the first two consecutive transitions (for $a$ and $f$) are unintended, then we can bound $M(u, \overline{\Omega_{1,1}})$ from above as follows. First, from a generalization of Lemma 5.3 to any subset of pathes, we have:

$$\begin{aligned} M(u, \overline{\Omega_{1,1}}) &= max\{M(0,0,a), M(1,*,a)\} \cdot max_{i \in S} M(i,i,f) \cdot max_{i \in S} M(i,*,a) \cdot \\ &\quad max_{i \in S} M(i,*,b) \cdot max_{i \in S} M(i,*,f) \cdot max_{i \in S} M(i,*,b) \end{aligned} \tag{B.17}$$

Now by the definitions of leak and skew, if we let $M_\tau$ be an arbitrary canonical matrix, we have:

$$\begin{aligned} max\{M(0,0,a), M(1,*,a)\} &\leq max\{\lambda_0^{(a,b)}(M, M_\tau) \cdot M(0,*,(a,b)), \\ &\quad \lambda_1^{(a,b)}(M, M_\tau) \cdot M(1,*,(a,b))\} \text{ by (5.8)} \\ &\leq \lambda^{(a,b)}(M, M_\tau) \cdot max_{i \in S} M(i,*,(a,b)) \text{ by (5.9)} \\ &\leq \lambda^{(a,b)}(M, M_\tau) \cdot (1 + \nu^{(a,b)}(M)) \cdot \frac{M(*,*,(a,b))}{2} \text{ by (5.11)} \end{aligned}$$

Similarly,

$$max_{i \in S}\{M(i,i,f)\} \leq \lambda^{(f)}(M, M_\tau) \cdot (1 + \nu^{(f)}(M)) \cdot \frac{M(*,*,f)}{2}$$

For an arbitrary letter $z$ in an arbitrary letter group $G$,

$$max_{i \in S} M(i,*,z) \leq (1 + \nu^G(M)) \cdot \frac{M(*,*,G)}{2} \text{ by (5.11)}$$

Plugging these into (B.17), we obtain:

$$\begin{aligned} M(u, \overline{\Omega_{1,1}}) &= \lambda^{(a,b)}(M, M_\tau)(1 + \nu^{(a,b)}(M))\frac{M(*,*,(a,b))}{2} \cdot \\ &\quad \lambda^{(f)}(M, M_\tau)(1 + \nu^{(f)}(M))\frac{M(*,*,f)}{2} \cdot \\ &\quad (1 + \nu^{(a,b)}(M))^3 (\frac{M(*,*,(a,b))}{2})^3 \cdot (1 + \nu^{(f)}(M))\frac{M(*,*,f)}{2} \\ &\leq \lambda^{(a,b)}(M, M_\tau) \cdot \lambda^{(f)}(M, M_\tau) \cdot (1 + \nu^{(a,b)}(M))^4 \cdot (1 + \nu^{(f)}(M))^2 \cdot \\ &\quad (\frac{M(*,*,(a,b))}{2})^4 \cdot (\frac{M(*,*,f)}{2})^2 \\ &= \lambda^{(a,b)}(M, M_\tau) \cdot \lambda^{(a,b,f)}(M, M_\tau) \cdot (1 + \nu^{(a,b)}(M))^4 \cdot (1 + \nu^{(f)}(M))^2 \cdot C[M](u) \end{aligned}$$

Since we can derive the same inequality for each of the six possible places for the first two consecutive positions of unintended transitions, we obtain the following.

$$M(u, \overline{\Omega_1}) \leq 6\lambda^{(a,b)}(M, M_\tau)\lambda^{(a,b,f)}(M, M_\tau)(1 + \nu^{(a,b)}(M))^4(1 + \nu^{(f)}(M))^2 C[M](u)$$

Now since we have $\delta^{(a,b)}(M, M_\tau) = max\{\lambda^{(a,b)}(M, M_\tau), (\nu^{(a,b)}(M))^2\} \leq (\frac{1}{2})^{10}$, and $1 + \nu^{(f)}(M) \leq 2$ by (5.12), we obtain:

$$\begin{aligned} M(u, \overline{\Omega_1}) &\leq 6 \cdot (\frac{1}{2})^{10}(1 + (\frac{1}{2})^5)^4 2^2 \lambda^{(a,b,f)}(M, M_\tau) C[M](u) \\ &< \frac{1}{2}\lambda^{(a,b,f)}(M, M_\tau) C[M](u) \end{aligned} \tag{B.18}$$

**Proof of Claim B.1** What we wish to show is roughly as follows: Given that $a$ and $b$ go essentially as *intended*, that is $a$ mostly goes from 0 to 1, and $b$ mostly goes from 1 to 0, then $f$ must also go essentially as intended, that is $f$ must go mostly from 0 to 1, and 1 to 0. (In the mirror image of a canonical matrix, the roles of $a$ and $b$ are flipped, but here we assume without loss of generality that $a$ goes from 0 to 1, and not the other way round.) We call these four transitions *intended transitions*, and all others *unintended transitions*. Note that $\mid u \mid = 6$, and so the length of each path (state sequence) for $u$ is of length $\mid u \mid +1 = 7$. Let $\Omega$ denote the set of all pathes[7] possibly generating $u$, or $\Omega = S^7$. We then let $\Omega_1$ denote the set of those pathes in $\Omega$ containing only intended transitions (for $u$), except possibly at the two ends. For example, $1 \to 1 \to 0 \to 1 \to 0 \to 1 \to 1$ is in $\Omega_1$ because the only unintended transitions in it are the first $1 \to 1$, labeled with $a$, and the last $1 \to 1$, labeled with $b$. On the other hand, $0 \to 1 \to 1 \to 1 \to 0 \to 1 \to 0$ is *not* in $\Omega_1$ because the second transition, $1 \to 1$ labeled with $f$, and the third, $1 \to 1$ labeled with $a$ are unintended. Let $\overline{\Omega_1}$ denote $\Omega \setminus \Omega_1$. Let $M(x, \Upsilon)$, in general, denote the probability that the string $x$ is generated by one of the pathes in $\Upsilon$. Then note the following.

$$M(u) = M(u, \Omega_1) + M(u, \overline{\Omega_1}) \tag{B.15}$$

Given that $\delta^{(a,b)}(M) \leq (\frac{1}{2})^{10}$ and that $\delta^{(f)}(M) > \delta^{(a,b)}(M)$, we will bound $M(u)$ from above as in the statement of the claim, by bounding from above the two terms $M(u, \Omega_1)$ and $M(u, \overline{\Omega_1})$ in (B.15) separately. In computing $M(u, \Omega_1)$, recall from Example 5.1 that:

$$M(0,1,a)M(1,0,b) \leq (1 - \delta^{(a,b)}(M))(\frac{M(*,*,(a,b))}{2})^2$$

Similarly we can also show

$$M(0,1,f)M(1,0,f) \leq (1 - \delta^{(f)}(M))(\frac{M(*,*,f)}{2})^2$$

Using these, we can bound $M(u, \Omega_1)$ as follows:

$$\begin{aligned}
M(u,\Omega_1) &\leq M(*,1,a)M(1,0,f)M(0,1,a)M(1,0,b)M(0,1,f)M(1,*,b) \\
&\leq M(*,*,a)M(*,*,b) \cdot M(0,1,a)M(1,0,b) \cdot M(0,1,f)M(1,0,f) \\
&\leq (\frac{M(*,*,(a,b))}{2})^2 \cdot (1 - \delta^{(a,b)}(M))(\frac{M(*,*,(a,b))}{2})^2 \cdot (1 - \delta^{(f)}(M))(\frac{M(*,*,f)}{2})^2 \\
&\leq (1 - \delta^{(f)}(M))(1 - \delta^{(a,b)}(M))C[M](u) \\
&\leq (1 - \delta^{(a,b,f)}(M))C[M](u)
\end{aligned} \tag{B.16}$$

Now we bound from above the rest of $M(u)$, i.e. $M(u, \overline{\Omega_1})$. First note that by definition every path, say $\pi$, in $\overline{\Omega_1}$ contains at least one unintended transition which is at neither end of $u$. The crucial observation is that because such an unintended transition has a transition before and after it, there must be at least *two consecutive unintended* transitions in $\pi$. Now since $u = afabfb$, this implies that there must be at least one unintended $a,b$-transition, and another unintended transition in $\pi$. Now note that there are at most 6 possible places for two consecutive unintended transitions in a path in $\overline{\Omega_1}$. If one fixes

[7] Each path *for* the string $u$ can be formalized as a length 6 sequence from $S \times S \times \Sigma_n$, where each $i$-th member has the $i$-th symbol in $u$ as its third component (its label) and consecutive transitions end and start in the same state. Here we simplify our notation by viewing a path simply as a sequence of states, leaving the labelings by $u$ implicit.

$$
\begin{aligned}
M(w) &\leq (M(i_a,*,a)M(i_b,*,b))^{\frac{\sharp((a,b),w)}{2}} \prod_{z\in\Sigma_n\setminus\{a,b\}} M(*,*,z)^{\sharp(z,w)}, \text{ by Lemma 5.3, part 1} \\
&\leq (\frac{1}{2})^{\frac{\sharp((a,b),w)}{2}}(\frac{M_\tau(*,*,(a,b))}{2})^{\sharp((a,b),w)} \cdot (2h)^{\sharp(\overline{((a,b)},w)} \cdot \prod_{G\in\Gamma_n\setminus\{a,b\}} (\frac{M_\tau(*,*,G)}{2})^{\sharp(G,w)} \\
&\quad \text{by (B.10) and (B.13)} \\
&= (\frac{1}{2})^{\frac{\sharp((a,b),w)}{2}}(2h)^{\sharp(\overline{((a,b)},w)}C(w), \text{ by definition of } C(w) \\
&\leq ((\frac{1}{2})^{\frac{h-1}{2}}2h)^{\sharp(\overline{((a,b)},w)}C(w), \text{ since } \sharp((a,b),w) = (h-1)\sharp(\overline{(a,b)},w) \\
&\leq ((\frac{1}{2})^{\frac{h-3-\log h}{2}})^{\sharp(\overline{((a,b)},w)}C(w) \\
&\leq (\frac{1}{2})^{\sharp(\overline{((a,b)},w)}C(w), \\
&\quad \text{since } (\tfrac{1}{2})^{\frac{h-3-\log h}{2}} < \tfrac{1}{2} \text{ because clearly } \tfrac{h-3-\log h}{2} > 1 \text{ given } h > s2^{11} \\
&< \frac{1}{K}C(w), \quad \text{since } \sharp(\overline{(a,b)},w) > \log K
\end{aligned}
\tag{B.14}
$$

Thus, we have verified that $\sigma_a$ and $\sigma_b$ are either $0 \to 1$ and $1 \to 0$, or $1 \to 0$ and $0 \to 1$. By (B.6) and (B.7), and the definition of $\lambda^{(a,b)}(M, \{a : 0 \to 1, b : 1 \to 0\})$ and $\lambda^{(a,b)}(M, \{a : 1 \to 0, b : 0 \to 1\})$, we conclude that one of the following must hold:

$$\lambda^{(a,b)}(M, \{a : 0 \to 1, b : 1 \to 0\}) \leq \frac{1}{s2^{10}}, \text{ or}$$

$$\lambda^{(a,b)}(M, \{a : 1 \to 0, b : 0 \to 1\}) \leq \frac{1}{s2^{10}}$$

Assume without loss of generality that the first of these two holds. Note that most of $a$'s share is out of 0 and most of $b$'s share is out of 1. Then, again by Lemma 5.3, part 1, we must have:

$$
\begin{aligned}
M(w_0) &\leq (M(0,*,a)M(1,*,b))^{k_0} \\
&\leq (1-\nu^{(a,b)}(M)^2)^{k_0}(\frac{M(*,*,(a,b))}{2})^{2k_0}, \text{ as shown in Example 5.1}
\end{aligned}
$$

By an argument which is by now familiar, this implies that $\nu^{(a,b)}(M)^2 \leq \frac{1}{s2^{11}}$. Hence, together with the earlier assumption that $\lambda^{(a,b)}(M, \{a : 0 \to 1, b : 1 \to 0\}) \leq \frac{1}{s2^{10}}$, this implies that

$$
\begin{aligned}
\delta^{(a,b)}(M) &= min_{\tau\in T_n} max\{\lambda^{(a,b)}(M, M_\tau), \nu^{(a,b)}(M)^2\} \\
&\leq max\{\lambda^{(a,b)}(M, \{a : 0 \to 1, b : 1 \to 0\}), \nu^{(a,b)}(M)^2\} \\
&\leq \frac{1}{s2^{10}}.
\end{aligned}
$$

**Proof of Lemma 5.6, part 2**

This part follows directly from Lemma 5.3.

**Proof of Lemma 5.6, part 3**

This part follows at once if we establish the following claim.

**Claim B.1:** *Let $u = afabfb$. Then, if $\delta^{(a,b)}(M) \leq (\frac{1}{2})^{10}$ then $M(u) \leq (1-\frac{\delta^{(a,b,f)}(M)}{2})C[M](u)$.*

Now let $\sharp(\overline{(a,b)},w)$ be the number of all letters other than $a$ and $b$ in $w$ and let $h$ denote the inverse of the frequency of these letters in $w$, that is, $h = \frac{|w|}{\sharp(\overline{(a,b)},w)}$. Note that $h > \frac{|w|}{|w_1 w_2 w_3 w_4 w_5|} > s2^{11}$ holds by the way $w$ is defined. Also note that $(\frac{M_\tau(*,*,(a,b))}{2}) = \frac{\sharp((a,b),w)}{|w|} = 1 - \frac{1}{h}$. Using (B.9), we can derive:

$$\begin{aligned}
M(i_a,*,a)M(i_b,*,b) &\leq (\frac{M(i_a,*,a)+M(i_b,*,b)}{2})^2, \text{ since } xy \leq (\tfrac{(x+y)}{2})^2 \\
&\leq (\frac{1}{2})^2, \text{ by (B.9)} \\
&\leq (\frac{1}{2}\frac{1}{1-\frac{1}{h}}\frac{M_\tau(*,*,(a,b))}{2})^2, \text{ because } (\tfrac{M_\tau(*,*,(a,b))}{2})) = 1-\tfrac{1}{h} \\
&= \frac{1}{4}(1+\frac{1}{h-1})^2 \cdot (\frac{M_\tau(*,*,(a,b))}{2})^2 \\
&\leq \frac{1}{2}\cdot(\frac{M_\tau(*,*,(a,b))}{2})^2 \text{ noting that } h > s2^{11}. \qquad \text{(B.10)}
\end{aligned}$$

Note that the above quantifies how much $M$ loses on the letters $a$ and $b$ in $w$, as compared to a canonical matrix $M_\tau$: $M$ loses by at least a factor of $\frac{1}{2}$ per each pair of $a$ and $b$, if we have option 2. Next, we will bound how much $M$ could possibly *gain* on the remaining letters as compared to a canonical matrix, by the fact that option 2 gives $a$ and $b$ *less* share. This will again be quantified in terms of $h$. From Lemma 5.3, part 1, we know that the total generation probability assigned on these letters is at most $\prod_{z\in\Sigma_n\setminus\{a,b\}} M(*,*,z)^{\sharp(z,w)}$ and this quantity is maximized when $M(*,*,z)$'s are set proportionally to their frequencies in $w$ within their total share, i.e. subject to $\prod_{z\in\Sigma_n\setminus\{a,b\}} M(*,*,z) = 2 - \sum_{z\in\{a,b\}} M(*,*,z) \leq 2$. Hence,

$$\prod_{z\in\Sigma_n\setminus\{a,b\}} M(*,*,z)^{\sharp(z,w)} \leq \prod_{z\in\Sigma_n\setminus\{a,b\}} (\frac{2\sharp(z,w)}{\sharp(\overline{(a,b)},w)})^{\sharp(z,w)} \qquad \text{(B.11)}$$

But, by the definition of $M_\tau$, we have that :

$$\prod_{z\in\Sigma_n\setminus\{a,b\}} M_\tau(*,*,z)^{\sharp(z,w)} = \prod_{z\in\Sigma_n\setminus\{a,b\}} (\frac{2\sharp(z,w)}{\mid w \mid})^{\sharp(z,w)} \qquad \text{(B.12)}$$

from (B.11) and (B.12), it follows that :

$$\begin{aligned}
\prod_{z\in\Sigma_n\setminus\{a,b\}} M(*,*,z)^{\sharp(z,w)} &\leq \prod_{z\in\Sigma_n\setminus\{a,b\}} (\frac{\mid w \mid}{\sharp(\overline{(a,b)},w)}\cdot\frac{2\sharp(z,w)}{\mid w \mid})^{\sharp(z,w)} \\
&< \prod_{z\in\Sigma_n\setminus\{a,b\}} (\frac{\mid w \mid}{\sharp(\overline{(a,b)},w)})^{\sharp(z,w)} \cdot \prod_{G\in\Gamma_n\setminus\{(a,b)\}} M_\tau(*,*,G)^{\sharp(G,w)} \\
&\leq (2h)^{\sharp(\overline{(a,b)},w)} \cdot \prod_{G\in\Gamma_n\setminus\{(a,b)\}} (\frac{M_\tau(*,*,G)}{2})^{\sharp(G,w)} \qquad \text{(B.13)}
\end{aligned}$$

Here, the second to last inequality follows from the observation that by (5.3) for any partial letter $z$ in letter group $G$, $M_\tau(*,*,G) = 4^{\frac{\sharp(z,w)}{|w|}}$ and by (5.4) for any total letter $z$, $M_\tau(*,*,\{z\}) = 2^{\frac{\sharp(z,w)}{|w|}}$. Now recalling that $C(w) = \prod_{G\in\Gamma_n}(\frac{M_\tau(*,*,G)}{2})^{\sharp(G,w)}$, we can use (B.10) and (B.13) to bound $M(w)$ from above by $\frac{1}{K}C(w)$, again contradicting our assumption.

This contradicts our assumption, so we must conclude that $\Lambda \leq \frac{1}{s2^{11}}$. Intuitively, we have shown that both the out-share and in-share of each of $a$ and $b$ must be highly concentrated on one of the two states, and hence that for each of $a$ and $b$, there is exactly one transition in $M$ which has almost all of the letter's share. Let $\sigma_a = (i_a, j_a)$ and $\sigma_b = (i_b, j_b)$ these two *dominating* transitions for $a$ and $b$, respectively. Formally we can derive the following, recalling that $\Lambda_{out} = \frac{\sum_{i \neq i_a} M(i,*,a) + \sum_{i \neq i_b} M(i,*,b)}{M(*,*,(a,b))}$, and $\Lambda_{in} = \frac{\sum_{j \neq j_a} M(*,j,a) + \sum_{j \neq j_b} M(*,j,b)}{M(*,*,(a,b))}$

$$\begin{aligned}
\sum_{(i,j) \neq (i_a, j_a)} M(i,j,a) &\leq \sum_{i \neq i_a} M(i,*,a) + \sum_{j \neq j_a} M(*,j,a) \\
&\leq \Lambda_{out} M(*,*,(a,b)) + \Lambda_{in} M(*,*,(a,b)) \\
&\leq 2\Lambda M(*,*,(a,b)) \\
&\leq \frac{1}{s2^{10}} M(*,*,(a,b)) \qquad \text{(B.6)}
\end{aligned}$$

Similarly, for the letter $b$, we can show:

$$\sum_{(i,j) \neq (i_b, j_b)} M(i,j,b) \leq \frac{1}{s2^{10}} M(*,*,(a,b)) \qquad \text{(B.7)}$$

Next we will show that $\sigma_a$ and $\sigma_b$ must form a *cycle*, that is, $j_a = i_b$ and $j_b = i_a$. For suppose otherwise, then each of the $2^5$ possible pathes for the substring *abab* contains at least one transition other than $\sigma_a$ and $\sigma_b$ which by (B.6) and (B.7) has probability at most $\frac{1}{s2^{10}} M(*,*,(a,b))$. Thus, each of the $2^5$ pathes has probability at most $\frac{1}{s2^{10}} M(*,*,(a,b)) \cdot max\{M(*,*,a), M(*,*,b)\}^3$ and

$$\begin{aligned}
M(abab) &\leq 2^5 \cdot \frac{1}{s2^{10}} M(*,*,(a,b)) \cdot max\{M(*,*,a), M(*,*,b)\}^3 \\
&\leq \frac{1}{s2^5} M(*,*,(a,b))^4 \\
&\leq \frac{1}{2} \left(\frac{M(*,*,(a,b))}{2}\right)^4 = \frac{1}{2} C[M](abab) \qquad \text{(B.8)}
\end{aligned}$$

Recalling that $w_0 = (ab)^{k_0}$, the above implies that $M(w_0) \leq (\frac{1}{2})^{k_0} C[M](w_0)$. As before, by the choice of $k_0$, this implies that $M(w) < \frac{1}{K} C(w)$. We have thus shown that $\sigma_a$ and $\sigma_b$ must form a cycle. More precisely, we must have one of the following four possibilities.

1. $(\sigma_a, \sigma_b) = (0 \to 1, 1 \to 0)$.
2. $(\sigma_a, \sigma_b) = (0 \to 0, 0 \to 0)$.
3. $(\sigma_a, \sigma_b) = (1 \to 1, 1 \to 1)$.
4. $(\sigma_a, \sigma_b) = (1 \to 0, 0 \to 1)$.

Next, we will show that in fact only options 1 and 4 are possible, given the assumption that $M(w) \geq \frac{1}{K} C(w)$: if we had either option 2 or 3, then we would have $M(w) < \frac{1}{K} C(w)$. Assume without loss of generality that we have option 2, that is, $(\sigma_a, \sigma_b) = (0 \to 0, 0 \to 0)$. Then, note that $i_a = 0$ and $i_b = 0$. Hence we conclude:

$$M(i_a, *, a) + M(i_b, *, b) = M(0, *, a) + M(0, *, b) \leq 1 \qquad \text{(B.9)}$$

## B Proof of Lemma 5.6

In this appendix we prove Lemma 5.6, the key lemma used in the proof of the Lemma 5.1, part 2. The proof uses the technical lemmas 5.2 through 5.5.

**Proof of Lemma 5.6, part 1**

Assume that $M(w) \geq \frac{1}{K}C(w)$. For an arbitrary letter $z$, let $i_z$ be the state $i \in S$ with the maximum out-share for $z$ (Let $i_z = 0$, if $M(0,*,z) = M(1,*,z)$.) Similarly let $j_z$ be the state with the maximum in-share for $z$. Then by Lemma 5.3, part 1, we have:

$$\begin{aligned} M(w_0) &\leq M(i_a,*,a)^{k_0} \cdot M(i_b,*,b)^{k_0} \\ &\leq (\frac{M(i_a,*,a)+M(i_b,*,b)}{2})^{2k_0}, \text{ because } xy \leq (\tfrac{x+y}{2})^2 \ \forall x,y \in \mathbf{R} \end{aligned} \quad \text{(B.1)}$$

Let $\Lambda_{out}$ denote the fraction of the total $(a,b)$-share *not* given to the states with maximum out-shares for $a$ and $b$, that is, $\Lambda_{out} = \frac{\sum_{i\neq i_a} M(i,*,a)+\sum_{i\neq i_b} M(i,*,b)}{M(*,*,(a,b))}$. Alternatively, we can write $\Lambda_{out}$ as $1 - \frac{M(i_a,*,a)+M(i_b,*,b)}{M(*,*,(a,b))}$. Noting that $M(i_a,*,a) + M(i_b,*,b) = (1 - \Lambda_{out})M(*,*,(a,b))$, we obtain from (B.1):

$$\begin{aligned} M(w_0) &\leq (1-\Lambda_{out})^{2k_0}(\frac{M(*,*,(a,b))}{2})^{2k_0} \\ &= (1-\Lambda_{out})^{2k_0}C[M](w_0), \text{ by definition of } C[M] \text{ and noting } w_0 = (ab)^{k_0} \end{aligned}$$

Similarly, if we define $\Lambda_{in}$ to be $\frac{\sum_{j\neq j_a} M(*,j,a)+\sum_{j\neq j_b} M(*,j,b)}{M(*,*,(a,b))} = 1 - \frac{M(*,j_a,a)+M(*,j_b,b)}{M(*,*,(a,b))}$, we obtain the following from Lemma 5.3, part 2.

$$M(w_0) \leq (1-\Lambda_{in})^{2k_0}C[M](w_0) \quad \text{(B.2)}$$

So if we now let $\Lambda = max\{\Lambda_{out}, \Lambda_{in}\}$, then we have:

$$M(w_0) \leq (1-\Lambda)^{2k_0}C[M](w_0) \quad \text{(B.3)}$$

Suppose for contradiction that $\Lambda > \frac{1}{s2^{11}}$. Then by the choice of $k_0 = s2^{11}$ $(\log K + |\ w_1w_2w_3w_4w_5\ |)$, we must have

$$\begin{aligned} M(w_0) &< (1-\frac{1}{s2^{11}})^{k_0}C[M](w_0) \quad \text{(B.4)} \\ &< e^{-\frac{1}{s2^{11}}s2^{11}(\log K+|w_1w_2w_3w_4w_5|)}C[M](w_0) \\ &< \frac{1}{K}\cdot(\frac{1}{2})^{|w_1w_2w_3w_4w_5|}C[M](w_0) \quad \text{(B.5)} \end{aligned}$$

Thus an argument of the style that was used frequently in the proof of Lemma 5.6 shows:

$$\begin{aligned} M(w) &\leq M(w_0)M(w_1w_2w_3w_4w_5) \\ &< \frac{1}{K}(\frac{1}{2})^{|w_1w_2w_3w_4w_5|}C[M](w_0)\cdot 2^{|w_1w_2w_3w_4w_5|}C[M](w_1w_2w_3w_4w_5), \text{ by Lemma 5.4} \\ &\leq \frac{1}{K}C[M](w_0)C[M](w_1w_2w_3w_4w_5) \\ &= \frac{1}{K}C[M](w) \\ &\leq \frac{1}{K}C(w), \text{ by Lemma 5.2} \end{aligned}$$

But by Lemma 5.3, part 1, we have:

$$M(x) \leq \prod_{G \in \Gamma_n} (M(*,*,G))^{\sharp(G,x)}$$

So it follows:

$$M(x) \leq 2^{|x|} \cdot C[M](x)$$

**End of proof of Lemma 5.4**

**Proof of Lemma 5.5**

Since $M_\tau(x) = 0$, in every path for $x$ there must be at least one transition assigned zero probability by $M_\tau$. Now since $\delta(M, M_\tau) = \Delta$, we have:

$$\lambda(M, M_\tau) = max_{G \in \Gamma_n, i \in S} \sum_{z \in G, j \in S, M_\tau(i,j,z)=0} \frac{M(i,j,z)}{M(i,*,G)} \leq \delta(M, M_\tau) = \Delta$$

Hence, in every path for $x$, there must be at least one transition, say the $k$-th one, with probability not exceeding $\Delta \cdot max_{i \in S} M(i,*,G(x_k))$, where we wrote $G(x_k)$ for the letter group to which $x_k$ belongs. But since $\nu(M, M_\tau) = max_{G \in \Gamma_n, i \in S}\ 2 \mid \frac{M(i,*,G)}{M(*,*,G)} - \frac{1}{2} \mid \leq \sqrt{\delta(M, M_\tau)} = \sqrt{\Delta}$,

$$\Delta \cdot max_{i \in S}\{M(i,*,G(x_k))\} \leq \Delta(1+\sqrt{\Delta})\frac{M(*,*,G(x_k))}{2}.$$

Let $\Gamma = S^{l+1}$ be the set of all pathes of length $l = |x|$. Partition $\Gamma$ into $\Gamma_1, \Gamma_2, ..., \Gamma_l$ according to the first occurrence $k$ of a transition probability not exceeding $\Delta(1+\sqrt{\Delta})\frac{M(*,*,G(x_k))}{2}$. Notice that $\Gamma = \cup_{k=1}^{l} \Gamma_k$ and $\Gamma_k$'s are mutually disjoint. Therefore if we define $M(x, \Gamma_k) = \sum_{\sigma \in \Gamma_k} \prod_{k=1}^{l} M(\sigma_{k-1}, \sigma_k, x_k)$, then we have $M(x) = \sum_{k=1}^{l} (\sum_{\sigma \in \Gamma_k} M(x, \sigma))$. Applying a similar argument as in the proof of Lemma 5.3, part 1, on each $\Gamma_k$, we obtain:

$$\begin{aligned} M(x, \Gamma_k) &\leq (\prod_{j \neq k} max\{M(0,*,x_j), M(1,*,x_j)\}) \cdot \Delta(1+\sqrt{\Delta})\frac{M(*,*,G(x_k))}{2} \\ &\leq (\prod_{j \neq k} (1+\sqrt{\Delta})\frac{M(*,*,G(x_j))}{2}) \cdot \Delta(1+\sqrt{\Delta})\frac{M(*,*,G(x_k))}{2} \\ &\leq \Delta(1+\sqrt{\Delta})^l \cdot C[M](x) \end{aligned}$$

Hence we have:

$$M(x) \leq l\Delta(1+\sqrt{\Delta})^l \cdot C[M](x) \tag{A.3}$$

**End of proof of Lemma 5.5**

# A Proofs of Technical Lemmas

In this appendix we prove the technical lemmas 5.2 through 5.5.

**Proof of Lemma 5.2**

Recall that $C[M](w) = \sum_{G \in \Gamma_n} (\frac{M(*,*,G)}{2})^{\sharp(G,w)}$ with the constraint that $\sum_{G \in \Gamma_n} \frac{M(*,*,G)}{2} = 1$. By Lemma 4.2, the function of the form $f(x_1, ..., x_n) = x_1^{a_1} \cdot x_2^{a_2} \cdot ... \cdot x_n^{a_n}$ subject to the constraint $x_1 + x_2 + ... + x_n = c$ attains its maximum when $x_i = (\frac{a_i}{\sum_{i=1}^{n} a_i}) \cdot c$. Thus the maximum of $C[M](w)$ is obtained when the share $\frac{M(*,*,G)}{2}$ of each letter group $G$ is set to its frequency in $w$, as it is done in any canonical stochastic matrix. That is, for any $M_\tau$, $M_\tau(*,*,G) = 2\frac{\sharp(G,w)}{|w|}$ for all letter groups $G$.

**End of proof of Lemma 5.2**

**Proof of Lemma 5.3**

Let $M \in \mathcal{M}_n$ be an arbitrary stochastic matrix and $x = x_1 x_2 ... x_l \in \Sigma_n^l$ be an arbitrary string, as in the statement of lemma. For any $k$, $1 \leq k \leq l$, let $p_{k,j}$ denote the *conditional* probability that the machine $M$ is in state $j$, *given* that it has just generated $x_1 x_2 ... x_{k-1}$. Note that $p_{k,0} + p_{k,1} = 1$ for any $k$. We can write $M(x_1 ... x_k)$ as follows:

$$\begin{aligned} M(x_1...x_k) &\leq M(x_1...x_{k-1}) \cdot p_{k,0} \cdot M(0,*,x_k) + M(x_1...x_{k-1}) \cdot p_{k,1} \cdot M(1,*,x_k) \\ &= M(x_1...x_{k-1})(p_{k,0} \cdot M(0,*,x_k) + p_{k,1} \cdot M(1,*,x_k)) \\ &\leq M(x_1...x_{k-1}) max\{M(0,*,x_k), M(1,*,x_k)\}, \text{ since } p_{k,0} + p_{k,1} = 1. \end{aligned}$$

Hence it follows that:

$$M(x) \leq \prod_{i=1}^{|x|} max\{M(0,*,x_i), M(1,*,x_i)\} \tag{A.1}$$

Part 2 of the lemma follows from a similar argument. First, analogously to $p_{k,j}$, let $q_{k,j}$ denote the *conditional* probability that the machine $M$ was in state $j$ at time step $k$, *given* that it generated $x_{k+1} x_2 ... x_l$ after step $k$. Note that $q_{k,0} + q_{k,1} = 1$ for any $k$. We can write $M(x_k ... x_l)$ as follows:

$$\begin{aligned} M(x_k...x_l) &\leq M(*,0,x_k) \cdot q_{k,0} \cdot M(x_{k+1}...x_l) + M(*,1,x_k) \cdot q_{k,1} \cdot M(x_{k+1}...x_l) \\ &= (q_{k,0} \cdot M(*,0,x_k) + q_{k,1} \cdot M(*,1,x_k)) M(x_{k+1}...x_l) \\ &\leq max\{M(*,0,x_k), M(*,1,x_k)\} M(x_{k+1}...x_l), \text{ since } q_{k,0} + q_{k,1} = 1. \end{aligned}$$

Hence it follows that:

$$M(x) \leq \prod_{i=1}^{|x|} max\{M(*,0,x_i), M(*,1,x_i)\} \tag{A.2}$$

**End of proof of Lemma 5.3**

**Proof of Lemma 5.4**

This lemma follows straightforwardly from Lemma 5.3. By the definition of $C[M](x)$,

$$C[M](x) \leq \prod_{G \in \Gamma_n} (\frac{M(*,*,G)}{2})^{\sharp(G,x)}$$

## References


[Ang78] Dana Angluin. On the complexity of minimal inference of regular sets. *Information and Control*, 39:337–350, 1978.

[Ang88] Dana Angluin. Identifying languages from stochastic examples. Technical Report YALEU/DCS/RR-614, Yale University, March 1988.

[Ang89] Dana Angluin. Minimum consistent 2-state DFA problem is NP-complete. Unpublished manuscript, 1989.

[Bau72] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a markov process. *Inequalities*, 3:1 – 8, 1972.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929 – 965, October 1989.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6(4):675 – 695, 1977.

[Gol78] E. Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[Ham86] Richard W. Hamming. *Coding and Information Theory, Second Edition.* Prentice-Hall, 1986.

[Hau89] David Haussler. Generalizing the PAC model for neural net and other learning applications. Technical Report UCSC CRL-89-30, University of California at Santa Cruz, September 1989. Extended abstract appeared in the Proceedings of FOCS '89.

[KS90] Michael Kearns and Robert Schapire. Efficient distribution-free learning of probabilistic concepts. Proceedings of IEEE Symposium on Foundations of Computer Science, October, 1990.

[Lai88] Philip D. Laird. Efficient unsupervised learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, 1988.

[LRS83] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *The Bell System Technical Journal*, 62(4), April 1983.

[Pol84] David Pollard. *Convergence of Stochastic Processes.* Springer-Verlag, 1984.

[PW89] Leonard Pitt and Manfred Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proc. 19th ACM Symp. on Theory of Computation*, 1989. To appear in JACM.

[Tze89] Wen-Guey Tzeng. The equivalence and learning of probabilistic automata. In *Proceedings of the 30th Annual Symposium on the Foundations of Computer Science*, October 1989.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of A.C.M.*, 27:1134–1142, 1984.

[Yam91] Kenji Yamanishi. A learning criterion for stochastic rules. *Machine Learning*, the same issue.

Third, we would like to know whether the single string MLM problem for the complete 2-state HMM constraints (with variable alphabet size) is approximable in polynomial time or whether there are similar hardness results for this problem as the one proven in Theorem 5.1 for the single string MLM problem for 2-state PAs with variable alphabet size?

Fourth, can the latter hardness result on the single string MLM problem be strengthened from a factor of $2^{|w|^{1-\alpha}}$ from the optimum (for any $\alpha > 0$) to a factor of $2^{(1-\alpha)\cdot|w|}$ from the optimum (for any $\alpha > 0$)?

Fifth, recall that we have shown in Theorem 4.1 that for an arbitrary class of constraints the approximability of the *sample* MLM problem for it within a factor $2^{m^{1-\alpha}}$ (for any $\alpha > 0$) where $m$ is the sample size, would imply polynomial trainability of the same class. Could this be strengthened so that polynomial approximability within a factor $2^{(1-\alpha)\cdot m}$ (for some $\alpha > 0$) would already imply polynomial trainability of the class in question? Could we perhaps show that polynomial approximability of the *single string* MLM problem within a factor of $2^{|w|^{1-\alpha}}$ or $2^{(1-\alpha)\cdot|w|}$ (for some $\alpha > 0$) would imply polynomial trainability?

Finally, we emphasize that even though the hardness results may be disappointing they can serve as guidance in the search for constructive results. Perhaps the most significant open problem inspired by the results of the present paper is to determine practically relevant classes of PAs and HMMs that are provably polynomially trainable. For example, the class of HMM constraints used in speech recognition [LRS83] consist of chains of states in which only transitions that go forward in the chain or stay stationary in the chain are legal. What is the lowest sample complexity required for training this important class of HMM constraints ? Is this class polynomially trainable or is training this class hard modulo some weak assumption such as $\mathbf{RP} \neq \mathbf{NP}$ ?

## Acknowledgements

We thank David Haussler for teaching us many of the tools applied in this paper. In particular, he suggested the idea of bounding transition probabilities from below for showing sample complexity bounds. We thank Dana Angluin for showing us her hardness proof for the 2-state DFA consistency problem [Ang89] which served as a starting point for the non-approximability result for the single string MLM problem for the 2-state PA constraints. Thanks to Ron Rivest for bringing the single string MLM problem to our attention. Thanks to Nicolo Cesa-Bianchi, Yoav Freund, Phil Long, Aleksandar Milosavljevic, Dirk Van Compernolle, and Osamu Watanabe for fruitful discussions.

For an arbitrary HMM constraint $C = \langle I, G, L \rangle$, let $\mathcal{HMM}(C)$ denote the class of HMMs satisfying $C$. We can think of $\mathcal{HMM}(C)$ as the subset of $[0,1]^I \times [0,1]^G \times [0,1]^L$ satisfying the stochastic condition (6.1). The training problem for a class $\mathcal{C}$ of HMM constraints is defined analogously to the training problem for a class of PA constraints (see Definition 2.4). We can show the same sample complexity bound (up to a constant factor) for training a class of HMM constraints as we did for PAs in Theorem 3.1. We only sketch how this is shown, since the proof is almost identical to the proof of Theorem 3.1. We define $\mathcal{BHMM}(C)_m$ by bounding the initial, transition and letter generation probabilities from below by $\frac{1}{tm}$, where $t$ is the size of the input constraint. We define $\mathcal{FHMM}(C)_m$ by quantizing all the probabilities in the same way as before. The rest of the proof is essentially the same. The analogues of inequalities (3.11) and (3.12) for $\mathcal{HMM}(C)$, $\mathcal{BHMM}(C)_m$ and $\mathcal{FHMM}(C)_m$ can be shown the same way, noting the fact that the probability assigned on a string of length $n$ by an HMM is a product of $2n+1$ probabilities, as opposed to $n+1$ for PAs.

**Corollary 6.1:** *An arbitrary class of HMM constraints $\mathcal{C}$ can be trained with sample complexity $O((\frac{n}{\epsilon})^2 t \cdot \log^3 \frac{nt}{\epsilon} \cdot \log \frac{1}{\delta} \cdot \log^2 \log \frac{1}{\delta})$, where $t$ is the size of the input constraint.*

**Corollary 6.2:** *Any finite class of HMM constraints is polynomially trainable.*

## 7 Concluding Remarks

We were able to show that training an arbitrary class of PAs and HMMs can be done with polynomial sample complexity when computational efficiency is ignored. The sample complexity bounds given in Theorem 3.1 for training a class of PAs and in Corollary 6.1 for training a class of HMMs may perhaps be improved significantly. Lower bounds for these training problems should be investigated.

It would be interesting to establish sample complexity bounds for the training problems discussed in this paper with respect to the other measures of distance mentioned in the introduction. All our sample complexity bounds with respect to the Kullback-Leibler divergence rely on Hoeffding's inequality and thus grow with $\frac{1}{\epsilon^2}$. Is the $\frac{1}{\epsilon^2}$ growth in the sample complexity really necessary? Training with respect to other notions of distance may require only $\frac{1}{\epsilon}$ growth in the sample complexity.

We showed in Section 5 that $s$-state HMMs can easily be simulated by $s$-state PAs. How can HMMs be used to simulate PAs?

There are many open problems related to the hardness results of Section 5. First, we would like to know whether the following decision problem is in **NP**.

**Input**: two numbers $s$ and $a$ encoded in unary, a string $w \in \Sigma^*$ where $\mid \Sigma \mid = a$, a probability $q \in [0,1]$ encoded in binary.
**Question**: Does there exist an $s$-state PA $P$ with alphabet size $a$ such that $P(w) \geq q$?

It would also be interesting to determine the precise computational comlexity of various formulations of the approximate MLM problem as a kind of decision problem.

Second, Osamu Watanabe has brought to our attention that the reduction we exhibit in the proof of Theorem 5.1 can probably be modified so as to strengthen our result and show that the approximate single string MLM problem, perhaps with a more strict requirement of approximation, is in fact $\Delta_2^p$-complete.

$$\sum_{i \in S_P} \pi_P(i) = 1 \text{ and } \forall i \in S_P \sum_{j \in S_P} M_P(i,j) = 1 \text{ and } \forall i \in S_P \sum_{a \in \Sigma_P} L_P(i,a) = 1 \qquad (6.1)$$

*For any string* $w = w_1 ... w_n \in \Sigma_P^*$, *the generation probability assigned on it by* $P = \langle S_P, \Sigma_P, \pi_P, M_P, L_P \rangle$ *is computed as follows.*

$$P(w_1 ... w_n) = \sum_{\langle i_0, ..., i_n \rangle \in S_P^{n+1}} \pi_P(i_0) \cdot \prod_{j=0}^{n-1} L_P(i_j, w_{j+1}) M_P(i_j, i_{j+1}) \qquad (6.2)$$

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

Any hidden Markov model can be simulated by a probabilistic automaton of the same number of states. For an arbitrary hidden Markov model $P = \langle S_P, \Sigma_P, \pi_P, M_P, L_P \rangle$, define a probabilistic automaton $Q = \langle S_Q, \Sigma_Q, \pi_Q, M_Q \rangle$ by letting $S_Q = S_P$, $\Sigma_Q = \Sigma_P$, $\pi_Q = \pi_P$, and

$$\forall i, j \in S_Q \; \forall z \in \Sigma_Q \;\; M_Q(i,j,z) = L_P(i,z) \cdot M_P(i,j) \qquad (6.3)$$

Then, $P$ and $Q$ define the same distribution over $\Sigma_P^n$ for any $n$, as demonstrated below.

$$\begin{aligned} Q(w_1 ... w_n) &= \sum_{\langle i_0, ..., i_n \rangle \in S_Q^{n+1}} \pi_Q(i_0) \cdot \prod_{j=0}^{n-1} M_Q(i_j, i_{j+1}, w_{j+1}) \text{ by (2.2)} \\ &= \sum_{\langle i_0, ..., i_n \rangle \in S_P^{n+1}} \pi_P(i_0) \cdot \prod_{j=0}^{n-1} L_P(i_j, w_{j+1}) M_P(i_j, i_{j+1}) \text{ by (6.3)} \\ &= P(w_1 ... w_n) \text{ by (6.2)} \end{aligned}$$

Note that the ratio $\frac{M_Q(i,j,z)}{M_Q(i,j,z')}$ equals $\frac{L_P(i,z)}{L_P(i,z')}$ and is independent of $j$. In other words, the class of $s$ state hidden Markov models is equivalent to the class of $s$ state probabilistic automata satisfying the following condition:

$$\forall i, j, k \in S_P \; \forall z, z' \in \Sigma_P \; \frac{M_Q(i,j,z)}{M_Q(i,j,z')} = \frac{M_Q(i,k,z)}{M_Q(i,k,z')}$$

We define the training problem for hidden Markov models exactly analogously to the training problem for probabilistic automata (c.f. Definition 2.4). The only difference is that the input constraint that an HMM training algorithm receives is an *HMM constraint*, in place of a PA constraint. Here, an HMM constraint is a triple $C = \langle I, G, L \rangle$, specifying respectively the legal initial states, legal transitions and legal letter generations. Formally, $I \subseteq S$, $G \subseteq S \times S$ and $L \subseteq S \times \Sigma$, where $\Sigma$ is a finite alphabet, and $S$ is a finite set of states. As before, we measure the size of the input constraint by the total number of probability parameters in it, $| I | + | G | + | L |$, and denote this by $t$. We say that an HMM $P$ satisfies $C = \langle I, G, L \rangle$ if and only if $S_P = S$ and $\Sigma_P = \Sigma$ and

$$\begin{aligned} \forall i \notin I, \; \pi_P(i) &= 0 \\ \forall (i,j) \notin G, \; M_P(i,j) &= 0 \\ \forall (i,z) \notin L, \; L_P(i,z) &= 0 \end{aligned} \qquad (6.4)$$

Now again using Lemma 5.6, parts 2 through 6, and the fact that $k_5 = \log K$, we can show that $M(w) < \frac{1}{K}C(w)$, contradicting our assumption. Hence we conclude that $F$ must be satisfiable.

**End of proof of Lemma 5.1 and proof of Theorem 5.1**

**Proof of Theorem 5.2**

First note that 1-state PAs are *deterministic*. Thus, by Corollary 4.2, it is clear that the MLM problem for the 1-state constraints is solvable in polynomial time. We then show that if we use the optimal 1-state PA for the MLM problem for $s$-state complete constraints, then it achieves the guaranteed approximation factor of $s^{|w|}$. The optimal 1-state PA sets the transition probability of each letter proportionally to the frequency of that letter in the input string $w$. More precisely, the optimal matrix, denoted $M^*$, is defined as follows: For each letter $z \in \Sigma$, we set

$$M^*(0,0,z) = \frac{\sharp(z,w)}{\mid w \mid}$$

where we let 0 be the unique state in the 1-state PA. Hence, the probability $M^*$ assigns on $w$ is easily computed as follows:

$$M^*(w) = \prod_{z\in\Sigma} (\frac{\sharp(z,w)}{\mid w \mid})^{\sharp(z,w)}$$

Now, using Lemma 5.3, we can compute the following upper bound on $M(w)$ for any $s$-state stochastic matrix $M$.

$$M(w) \leq \prod_{z\in\Sigma} (max_{i\in S} M(i,*,z))^{\sharp(z,w)} \tag{5.20}$$

Here we have the constraint that $\sum_{z\in\Sigma} max_{i\in S} M(i,*,z) \leq s$. Hence, by Lemma 4.2, we have:

$$\begin{aligned} M(w) &\leq \prod_{z\in\Sigma} (s\frac{\sharp(z,w)}{\mid w \mid})^{\sharp(z,w)} \\ &= s^{|w|} \prod_{z\in\Sigma} (\frac{\sharp(z,w)}{\mid w \mid})^{\sharp(z,w)} \\ &= s^{|w|} M^*(w) \end{aligned}$$

**End of proof of Theorem 5.2**

## 6 Application to Training Hidden Markov Models

Hidden Markov models are used extensively as probabilistic models for generation of speech signals for the purpose of speech recognition. See [LRS83] for an excellent tutorial on this material. Hidden Markov models are defined similarly to probabilistic automata, except that the generation of letters is associated with the *states* rather than with the *transitions*. We briefly review the definition below.

**Definition 6.1 (Hidden Markov Models(HMM)):** *A hidden Markov model $P$ is a quintuple $\langle S_P, \Sigma_P, \pi_P, M_P, L_P\rangle$ where $S_P$ is a finite set of states, $\Sigma_P$ is a finite alphabet, $\pi_P : S_P \to [0,1]$ is the initial probability distribution over $S_P$, $M_P : S_P \times S_P \to [0,1]$ is a stochastic matrix, and $L_P : S_P \times \Sigma_P \to [0,1]$ specifies the letter generation distribution at each state, i.e.*

Substituting $k_1 = s2^{11}(\log K + \mid w_2w_3w_4w_5 \mid)$ into the above gives us:

$$\begin{aligned} M(w_1) &\leq (1 - \frac{\delta^{(f)}(M)}{2})^{s2^{11}(\log K + |w_2w_3w_4w_5|)} \cdot C[M](w_1) \\ &\leq e^{-\frac{\delta^{(f)}(M)}{2} \cdot s2^{11}(\log K + |w_2w_3w_4w_5|)} \cdot C[M](w_1) \\ &< \frac{1}{K}(\frac{1}{2})^{|w_2w_3w_4w_5|} \cdot C[M](w_1) \end{aligned} \tag{5.15}$$

Lemma 5.4 provides us with an upper bound on how large $M(w_2w_3w_4w_5)$ could possibly be in comparison to $C[M](w_2w_3w_4w_5)$, that is,

$$M(w_1w_2w_3w_4w_5) \leq 2^{|w_1w_2w_3w_4w_5|}C[M](w_1w_2w_3w_4w_5)$$

We can use this bound and (5.15) to bound $M(w)$ above by $\frac{1}{K}C(w)$ as follows:

$$\begin{aligned} M(w) &\leq M(w_0)M(w_1)M(w_2w_3w_4w_5) \\ &\leq C[M](w_0)M(w_1)M(w_2w_3w_4w_5), \text{ by Lemma 5.6, part 2} \\ &< C[M](w_0) \cdot \frac{1}{K}(\frac{1}{2})^{|w_2w_3w_4w_5|}C[M](w_1) \cdot M(w_2w_3w_4w_5), \text{ by (5.15)} \\ &< \frac{1}{K} \cdot C[M](w_0) \cdot C[M](w_1) \cdot (\frac{1}{2})^{|w_2w_3w_4w_5|}M(w_2w_3w_4w_5), \text{ by rearranging terms} \\ &< \frac{1}{K} \cdot C[M](w_0) \cdot C[M](w_1) \cdot C[M](w_2w_3w_4w_5), \text{ by Lemma 5.4} \\ &< \frac{1}{K} \cdot C[M](w), \text{ by definition of } C[M](w) \\ &\leq \frac{1}{K} \cdot C(w), \text{ by Lemma 5.2} \end{aligned} \tag{5.16}$$

This contradicts our assumption, and hence together with (5.14), we conclude:

$$\delta^{(a,b,f)}(M) = max\{\delta^{(a,b)}(M), \delta^{(f)}(M)\} \leq \frac{1}{s2^{10}}. \tag{5.17}$$

Since by assumption, $M(w) \geq \frac{C(w)}{K}$, $\Delta$ as defined in the statement of Lemma 5.6, part 4, does not exceed $\frac{1}{s2^{10}}$. Suppose now for contradiction that $\delta^{(c,d)}(M) > \frac{1}{s2^{10}}$. Then, since $\delta^{(a,b)}(M)$, $\Delta$ and $\delta^{(f)}(M)$ are now known not to exceed $\frac{1}{s2^{10}}$, it follows from Lemma 5.6, parts 2 through 5, and a similar argument as before, that this would imply $M(w) < \frac{1}{K}C(w)$. Thus, we conclude:

$$\delta^{(a,b,c,d,f)}(M) \leq \frac{1}{s2^{10}} \tag{5.18}$$

Given the above, it follows from Lemma 5.6, parts 2 through 6, by an analogous argument as before, that the distortion of $M$ with respect to all letters is small, that is,

$$\delta(M) \leq \frac{1}{s2^{10}} \tag{5.19}$$

Finally, suppose that $F$ *is unsatisfiable*. Then by Lemma 5.6, part 7, we must have

$$M(w_5) < (\frac{1}{2})^{k_5} \cdot C[M](w_5)$$

The following technical lemmas are useful for proving Lemma 5.1, part 2. Their proofs are deferred to Appendix A.

**Lemma 5.2:** *For an arbitrary stochastic matrix $M \in \mathcal{M}_n$, we have:*

$$C(w) \geq C[M](w)$$

**Lemma 5.3:** *For arbitrary $M \in \mathcal{M}_n$ and $x \in \Sigma_n^*$ we have:*

1. $M(x) \leq \prod_{z \in \Sigma_n} (max\{M(0,*,z), M(1,*,z)\})^{\sharp(z,x)}$.
2. $M(x) \leq \prod_{z \in \Sigma_n} (max\{M(*,0,z), M(*,1,z)\})^{\sharp(z,x)}$.

**Lemma 5.4:** *For an arbitrary stochastic matrix $M \in \mathcal{M}_n$ and an arbitrary string $x \in \Sigma_n^*$, we have:*

$$M(x) \leq 2^{|x|} C[M](x)$$

**Lemma 5.5:** *Let $M \in \mathcal{M}_n$ be an arbitrary stochastic matrix. Let $\delta(M) = \Delta$ and let $M_\tau$ be a canonical stochastic matrix to which $M$ is closest, namely $\delta(M, M_\tau) = \Delta$. If $M_\tau(x) = 0$ then we have for an arbitrary string $x$:*

$$M(x) \leq |x| \cdot \Delta (1 + \sqrt{\Delta})^{|x|} C[M](x)$$

Using the above technical lemmas, we are now ready to prove Lemma 5.1, part 2. The following lemma summarizes the key steps of this proof. Its proof also uses the above technical lemmas and is relegated to Appendix B.

**Lemma 5.6:** *For an arbitrary stochastic matrix $M \in \mathcal{M}_n$ all of the following hold.*

1. *If $M(w) \geq \frac{1}{K} C(w)$, then $\delta^{(a,b)}(M) \leq \frac{1}{s 2^{10}}$.*
2. *$M(w_0) \leq C[M](w_0)$.*
3. *If $\delta^{(a,b)}(M) \leq \frac{1}{s 2^{10}}$ then $M(w_1) < (1 - \frac{\delta^{(a,b,f)}(M)}{2})^{k_1} C[M](w_1)$.*
4. *For each $i \leq n$ let $\Delta_i = min\{\delta^{(c_i,d_i)}(M, \{c_i : 0 \to 0, d_i : 0 \to 0\}), \delta^{(c_i,d_i)}(M, \{c_i : 0 \to 1, d_i : 1 \to 0\}), \delta^{(c_i,d_i)}(M, \{c_i : 1 \to 0, d_i : 0 \to 1\}), \delta^{(c_i,d_i)}(M, \{c_i : 1 \to 1, d_i : 1 \to 1\})\}$. and let $\Delta = max_{i \leq n} \Delta_i$. Then, if $M(w) \geq \frac{C(w)}{K}$ then $\Delta \leq \frac{1}{s 2^{10}}$.*
5. *If $\Delta \leq \frac{1}{s 2^{10}}$, where $\Delta$ is as defined above, and $\delta^{(f)}(M) \leq \frac{1}{s 2^{10}}$, then $M(w_3) < (1 - \frac{\delta^{(c,d,f)}(M)}{2})^{k_3} C[M](w_3)$.*
6. *If $\delta^{(a,b,c,d,f)}(M) > \frac{1}{s 2^{10}}$ then $M(w_4) < (1 - \frac{\delta^{(a,b,c,d,f,x,\bar{x},e)}(M)}{2})^{k_4} C[M](w_4)$.*
7. *If $\delta^{(a,b,c,d,f,x,\bar{x},e)}(M) \leq \frac{1}{s 2^{10}}$ and $F$ is unsatisfiable, then $M(w_5) < (\frac{1}{2})^{k_5} C[M](w_5)$.*

**Proof of Lemma 5.1, part 2, given lemmas 5.2, 5.4 and 5.6**

Assume that $M^*(w) \geq \frac{C(w)}{K}$, and let a particular $M$ witness this fact. We will show that this will imply that $F$ is satisfiable. First, the following follows immediately from Lemma 5.6, part 1.

$$\delta^{(a,b)}(M) \leq \frac{1}{s 2^{10}}. \tag{5.14}$$

Next, suppose for contradiction that $\delta^{(f)}(M) > \frac{1}{s 2^{10}}$. Then, $\delta^{(a,b,f)}(M) = \delta^{(f)}(M)$, since $\delta^{(f)}(M) > \delta^{(a,b)}(M)$. It follows from Lemma 5.6, part 3:

$$M(w_1) \leq (1 - \frac{\delta^{(f)}(M)}{2})^{k_1} C[M](w_1)$$

The *distortion* of $M$ is then defined as the distortion of $M$ with respect to the *closest* canonical stochastic matrix, including their 'mirror images.'

$$\delta(M) = min_{\tau \in T_n} min\{\delta(M, M_\tau), \delta(M, M_\tau^r)\}$$

For each of *leak*, *skew* and *distortion*, we define the *restriction* of it to an arbitrary subset $\Psi$ of the letter groups by maximizing over $\Psi$ instead of $\Gamma_n$ in the above definitions. We use readable names such as $(a,b)$ and $(a,b,c,d)$ to refer to subsets of $\Gamma_n$ such as $\{\{a,b\}\}$ and $\{\{a,b\}\} \cup \{\{c_i,d_i\} : i = 1,...,n\}$. We then let symbols such as $\lambda^{(a,b)}$, $\nu^{(f)}$, and $\delta^{(a,b,c,d)}$ denote the corresponding restrictions of $\lambda$, $\nu$ and $\delta$. Note that the only information in $M_\tau$ that was used to define $\lambda(M, M_\tau)$ is the transition function associated with $M_\tau$, and to define $\lambda^G(M, M_\tau)$, only the restriction of it to $G$. Thus, for any transition function $f : S \times \cup G \rightarrow S$, restricted to a letter group $G$, we define $\lambda^G(M, f)$ to be $max_{i \in S, G \in \Gamma_n} \sum_{z \in G, j \in S, f(i,z) \neq j} \frac{M(i,j,z)}{M(i,*,G)}$, and define $\delta^G(M, f)$ accordingly. For readability, we use notation such as $\lambda^{(a,b)}(M, \{a : 0 \rightarrow 1, b : 1 \rightarrow 0\})$ and $\delta^{(a,b,c_i,d_i)}(M, \{a : 0 \rightarrow 1, b : 1 \rightarrow 0, c_i : 0 \rightarrow 1, d_i : 1 \rightarrow 0\})$.

The notion of distortion just defined quantifies how bad an arbitrary stochastic matrix is in comparison to canonical stochastic matrices which are *near optimal*, from the point of view of maximizing the generation probability on the *unique* path on the substring $w_0w_1w_2w_3w_4$, which is assigned a positive generation probability by canonical stochastic matrices. Any leak causes a loss of probability on this path as is obvious from its definition. Any skew also causes a loss of probability on the path, essentially because the path in question for the string $w_0w_1w_2w_3w_4$ is almost perfectly symmetric in the following sense: For each letter group (except the dummy letters $v_i$) there are exactly the same number of transitions out of state 0 as there are out of state 1. We demonstrate this via an example.

**Example 5.1:** *Let $M$ be an arbitrary stochastic matrix, and $M_\tau$ a canonical stochastic matrix. Consider the string $ab$ and the path for $ab$ which is assigned a positive probability by $M_\tau$, namely, $0 \rightarrow 1 \rightarrow 0$. We will show that the probability assigned by $M$ on this path, $M(0,1,a) \cdot M(1,0,b)$ is at most $(1 - \delta^{(a,b)}(M))(\frac{M(*,*,(a,b))}{2})^2$. Notice that $(\frac{M(*,*,\{a,b\})}{2})^2$ is the maximum probability assignable on $ab$ by any matrix $\widehat{M}$ such that $\widehat{M}(*,*,(a,b)) = M(*,*,(a,b))$. To show the above inequality first observe that*

$$\begin{aligned} M(0,1,a) \cdot M(1,0,b) &\leq (1 - \lambda_0^{(a,b)}(M, M_\tau))M(0,*,(a,b)) \cdot (1 - \lambda_1^{(a,b)}(M, M_\tau))M(1,*,(a,b)) \\ &\leq (1 - max\{\lambda_0^{(a,b)}(M, M_\tau), \lambda_1^{(a,b)}(M, M_\tau)\})M(0,*,(a,b))M(1,*,(a,b)) \end{aligned}$$

*By (5.9) the above maximization clause can be replaced by $\lambda^{(a,b)}(M, M_\tau)$. Also assume without loss of generality that $M(0,*,(a,b)) \geq M(1,*,(a,b))$. From the definition of $\nu^{(a,b)}$ in (5.11), it follows that $M(0,*,(a,b)) = (1 + \nu^{(a,b)})\frac{M(*,*,(a,b))}{2}$ which implies that $M(1,*,(a,b)) = (1 - \nu^{(a,b)})\frac{M(*,*,(a,b))}{2}$ and we continue as follows:*

$$\begin{aligned} M(0,1,a) \cdot M(1,0,b) &\leq (1 - \lambda^{(a,b)}(M, M_\tau))(1 + \nu^{(a,b)}(M))(1 - \nu^{(a,b)}(M))(\frac{M(*,*,(a,b))}{2})^2 \\ &\leq (1 - \lambda^{(a,b)}(M, M_\tau))(1 - \nu^{(a,b)}(M)^2) \cdot (\frac{M(*,*,(a,b))}{2})^2 \\ &\leq (1 - max\{\lambda^{(a,b)}(M, M_\tau), \nu^{(a,b)}(M)^2\}) \cdot (\frac{M(*,*,(a,b))}{2})^2 \\ &= (1 - \delta^{(a,b)}(M, M_\tau)) \cdot (\frac{M(*,*,(a,b))}{2})^2 \text{ by (5.13)} \end{aligned}$$

It can be shown that $C(u)$ is the probability that any *satisfying* canonical stochastic matrix assigns on $u$, but in this proof we only use the above definition for $C(u)$. We also introduce a version of $C(u)$ which is *relativized* to the letter group shares of an arbitrary stochastic matrix $M$. That is, for an arbitrary string $u$ we define the quantity $C[M](u)$ as follows:

$$C[M](u) = \prod_{G \in \Gamma_n} (\frac{M(*,*,G)}{2})^{\sharp(G,u)} \tag{5.7}$$

Again there is an interpretation of $C[M](u)$ as the probability assigned on $u$, when $\tau$ satisfies $F$, by a variant $M_\tau[M]$ of the canonical stochastic matrix $M_\tau$ which is defined exactly analogously to $M_\tau$ except the letter group shares of $M_\tau[M]$ are the same as those of $M$. Note that $C[M](uv) = C[M](u)C[M](v)$

We now formalize the notion of distance between an arbitrary stochastic matrix $M$ and any canonical stochastic matrix $M_\tau$. The *G-leak* of an arbitrary stochastic matrix $M$ with respect to $M_\tau$ at state $i$, written $\lambda_i^G(M, M_\tau)$, is the fraction of $G$'s share out of $i$ which is assigned by $M$ to transitions assigned zero probability by $M_\tau$:

$$\lambda_i^G(M, M_\tau) = \sum_{z \in G,\ j \in S,\ M_\tau(i,j,a)=0} \frac{M(i,j,z)}{M(i,*,G)} \tag{5.8}$$

Also,

$$\lambda^G(M, M_\tau) = max_{i \in S} \lambda_i^G(M, M_\tau) \tag{5.9}$$

We then define the *leak* of $M$ with respect to $M_\tau$ as the maximum $\lambda_i^G(M, M_\tau)$ over all states and letter groups.

$$\lambda(M, M_\tau) = max_{G \in \Gamma_n} \lambda^G(M, M_\tau) \tag{5.10}$$

The *skew*, written $\nu(M)$, is twice the maximum ratio at which the ratio $\frac{M(0,*,G)}{M(*,*,G)}$ deviates from a half for some letter group $G$.

$$\begin{aligned} \nu^G(M) &= 2 \mid \frac{M(0,*,G)}{M(*,*,G)} - \frac{1}{2} \mid \text{ and} \\ \nu(M) &= max_{G \in \Gamma_n} \nu^G(M) \end{aligned} \tag{5.11}$$

Note that since $0 \leq \frac{M(0,*,G)}{M(*,*,G)} \leq 1$, we have, for an arbitrary letter group $G$:

$$0 \leq \nu^G(M) \leq 1 \tag{5.12}$$

Recall that $\frac{M_\tau(0,*,G)}{M_\tau(*,*,G)} = \frac{1}{2}$ and thus for motivational purposes the skew of $M$ for a letter group $G$ can be expressed as the ratio at which the ratio $\frac{M(0,*,G)}{M(*,*,G)}$ deviates from $\frac{M_\tau(0,*,G)}{M_\tau(*,*,G)}$:

$$\nu^G(M) = \frac{\mid \frac{M(0,*,G)}{M(*,*,G)} - \frac{M_\tau(0,*,G)}{M_\tau(*,*,G)} \mid}{\frac{M_\tau(0,*,G)}{M_\tau(*,*,G)}}$$

We then define the *distortion* of $M$ with respect to any deterministic stochastic matrix $M_\tau$, written $\delta(M, M_\tau)$, as follows:

$$\delta(M, M_\tau) = max\{\lambda(M, M_\tau), \nu(M)^2\} \tag{5.13}$$

$$M(*,*,z) = \sum_{i \in S, j \in S} M(i,j,z)$$

$$M(i,*,z) = \sum_{j \in S} M(i,j,z)$$

$$M(*,j,z) = \sum_{i \in S} M(i,j,z)$$

We call $M(i,*,z)$ the *out-share* of $z$ at $i$ in $M$, $M(*,j,z)$ the *in-share* of $z$ at $j$ in $M$, and $M(*,*,z)$ the *share* of $z$ in $M$. We partition our alphabet $\Sigma_n$ into *letter groups*, in which the pairs of partial letters $(a,b)$ and $(c_i,d_i)$'s are grouped together and all total letters form their own group. Let $\Gamma_n$ denote the set of these letter groups, that is, $\Gamma_n = \{\{a,b\}\} \cup \{\{c_i,d_i\} : i = 1,...,n\} \cup \{\{z\} : z \text{ total}\}$. We then define the *share of a letter group* $G$ in a stochastic matrix $M$, written $M(*,*,G)$, to be the total sum of the probabilities of all transitions in $M$ labeled with a letter in $G$. That is,

$$M(*,*,G) = \sum_{i \in S, j \in S, z \in G} M(i,j,z)$$

The *out-share* and *in-share* of a *letter group* are defined analogously to those of a letter and $\sharp(G,w) = \sum_{z \in G} \sharp(z,w)$. Recall that in a canonical stochastic matrix, there is one transition out of each state for each total letter whose probability is the frequency of that letter, and there is exactly one transition for each partial letter, whose probability is twice the frequency of that letter. This implies that the share of each letter group in $M_\tau$ is set according to twice the total frequency of the letters from that letter group:

$$\forall G \in \Gamma_n \; M_\tau(*,*,G) = 2\frac{\sharp(G,w)}{\mid w \mid} \tag{5.2}$$

For example, for the letter group consisting of a single total letter $f$, we have the following from the definition of $M_\tau$.

$$M_\tau(*,*,f) = M_\tau(0,*,f) + M_\tau(1,*,f) = 2(\frac{\sharp(f,w)}{|w|}) \tag{5.3}$$

For a letter group consisting of two partial letters, for example $(a,b)$, we can derive the following also from the definition of $M_\tau$.

$$M_\tau(*,*,(a,b)) = M_\tau(*,*,a) + M_\tau(*,*,b) = 2\frac{2\sharp(a,w)}{|w|} = 2\frac{\sharp((a,b),w)}{|w|} \tag{5.4}$$

For each letter group the share is split evenly to the two states, and hence $M_\tau(0,*,G) = M_\tau(1,*,G) = \frac{\sharp(G,w)}{|w|}$. By summing over all letter groups it follows that $M_\tau(0,*,*) = M_\tau(1,*,*) = 1$ and thus $M_\tau$ of Definition 5.1 is stochastic. The generation probability $C(w)$ can now be rewritten as follows, by plugging (5.2) into (5.1).

$$C(w) = \prod_{G \in \Gamma_n} \frac{\sharp(G,w)}{\mid w \mid} = \prod_{G \in \Gamma_n} (\frac{M_\tau(*,*,G)}{2})^{\sharp(G,w)}, \tag{5.5}$$

Note that we can straightforwardly generalize $C(w)$ to an arbitrary substring $u$ of $w$: We define $C(u)$ by replacing each occurrence of $\sharp(G,w)$ by $\sharp(G,u)$:

$$C(u) = \prod_{G \in \Gamma_n} (\frac{M_\tau(*,*,G)}{2})^{\sharp(G,u)} \tag{5.6}$$

3. *For each variable* $X_i$,

$$\begin{array}{lll}(c_i, d_i) =_{M_\tau} (1 \to 0, 0 \to 1) & \text{if } \tau(X_i) = T. \\ \quad =_{M_\tau} (0 \to 1, 1 \to 0) & \text{if } \tau(X_i) = F. \\ v_i \quad =_{M_\tau} \textbf{flip} & \text{if } \tau(X_i) = T. \\ \quad =_{M_\tau} \textbf{id} & \text{if } \tau(X_i) = F. \\ (x_i, \bar{x}_i) =_{M_\tau} (\textbf{1}, \textbf{id}) & \text{if } \tau(X_i) = T. \\ \quad =_{M_\tau} (\textbf{id}, \textbf{1}) & \text{if } \tau(X_i) = F. \end{array}$$

4. *Each non-zero transition probability in* $M_\tau$ *is either equal to or twice the* frequency *in* $w$ *of the letter* $z$ *labeling the transition where this frequency is defined as the number of occurrences of* $z$ *in* $w$*, written* $\sharp(z, w)$*, divided by* $|w|$*. (Notice that* $a$ *and* $b$*, and all* $c_i$ *and* $d_i$ *are partial letters and all other letters are total.)*
   *(i) For any partial letter* $z$*,* $M_\tau(i, j, z) = 2\frac{\sharp(z,w)}{|w|}$ *if and only if* $M_\tau(i, j, z) \neq 0$*.*
   *(ii) For any total letter* $z$*,* $M_\tau(i, j, z) = \frac{\sharp(z,w)}{|w|}$ *if and only if* $M_\tau(i, j, z) \neq 0$*.*

We still need to show that $M_\tau$ is stochastic. For convenience we will defer this proof to later after we have developed some more notation for stochastic matrices. Note that in the above definition we have arbitrarily chosen $a$ to go from 0 to 1 and $b$ to go from 1 to 0. Given a canonical stochastic matrix $M_\tau$, we let $M_\tau^r$ denote its *mirror image*, obtained by reversing the states 0 and 1 in $M_\tau$. $M_\tau(w)$ can be calculated easily because canonical stochastic matrices are deterministic: $M_\tau(w)$ is zero, if $\tau$ does not satisfy $F$, and otherwise is just the product of the probabilities assigned on all the transitions occurring in *the* unique path for $w$ that is assigned a positive probability by $M_\tau$. If we define $C(w)$ as follows, regardless of whether $F$ is satisfiable or not,

$$C(w) = \prod_{z \text{ partial}} (2\frac{\sharp(z, w)}{\mid w \mid})^{\sharp(z,w)} \cdot \prod_{z \text{ total}} (\frac{\sharp(z, w)}{\mid w \mid})^{\sharp(z,w)} \tag{5.1}$$

then we have $M_\tau(w) = C(w)$ just in case $\tau$ satisfies $F$, and $M_\tau(w) = 0$ otherwise. We are now ready to state the key lemma in the proof of Theorem 5.1.

**Lemma 5.1:** *Let* $\omega_\alpha$ *be as defined earlier, and let* $M^*(w)$ *denote the maximum probability assignable on* $w$ *by any 2-state PA, i.e.* $M^*(w) = \max_{M \in \mathcal{M}_n} M(w)$*. Then for any CNF formula* $F$ *and* $K = 2^{|w|^{1-\alpha}}$*, we have:*

1. *If* $F$ *is satisfiable, then* $M^*(\omega_\alpha(F)) \geq C(\omega_\alpha(F))$*.*
2. *If* $F$ *is unsatisfiable, then* $M^*(\omega_\alpha(F)) < \frac{C(\omega_\alpha(F))}{K}$*.*

**Proof of Lemma 5.1**

The proof of part 1 is immediate: If $F$ is satisfiable then let $\tau$ satisfy $F$ and we have $M_\tau(w) = C(\omega_\alpha(F))$. The other direction is more involved and requires more definitions. The key is to find a useful way of quantifying the distance $\delta$ between an *arbitrary* stochastic matrix $M$ and canonical stochastic matrices such that the generation probability assigned on $w$ by $M$ can be shown to degrade rapidly as a function of $min_{\tau \in T_n} \delta(M, M_\tau)$. We can then use it to carry out the dilemma argument described in the proof sketch. We need some preliminary definitions.

Recall that for a given stochastic matrix $M$, $M(i, j, z)$ denotes the transition probability from state $i$ to $j$ labeled with letter $z$. We introduce the following notation for sums of transition probabilities of various forms:

of $2^n$ many *deterministic* stochastic matrices corresponding to truth assignments, and $w_b = w_5$ distinguishes those corresponding to satisfying assignments from those corresponding to non-satisfying assignments. Below we describe how this is intended to be achieved at a high level, leaving the exact nature of how the notion of 'closeness' among stochastic matrices to be specified later. The intended function of the first part $w_0$ of the string $w$ is to force the two 'control letters' $a$ and $b$ to go from 0 to 1, and 1 to 0, respectively or vice-versa. The intuitive reason why $(ab)^{k_0}$ forces these settings is as follows: Any stochastic matrix which has two transitions for either $a$ or $b$ will lose probability, and hence there can be only one transition for each of $a$ and $b$. Furthermore, to be able to generate $(ab)^{k_0}$, the single transition for $a$ and the single transition for $b$ must form a cycle. So, we must have (i) $(a,b) = (0 \to 0, 0 \to 0)$, (ii) $(a,b) = (1 \to 1, 1 \to 1)$, (iii) $(a,b) = (1 \to 0, 0 \to 1)$, or (iv) $(a,b) = (0 \to 1, 1 \to 0)$. But because the length of $w_0$ is almost the entire length of $w$, approximately optimal stochastic matrices must let these two transitions have *very* large probabilities (close to one). This is impossible if both of these transitions are out of the same state, so the options (i) and (ii) are eliminated, leaving (iii) and (iv). We assume without loss of generality that we have (iv), that is, $(a,b) = (0 \to 1, 1 \to 0)$. It is easy to see that with these particular settings of $a$ and $b$, $w_1$ forces $f$ to be a **flip**. $w_2$ performs the analogous function for each $(c_i, d_i)$ pair as $w_0$ did for $(a,b)$, but since $w_2$ is *not* the overwhelming majority (and there are $n$ such pairs), at this point all four (i - iv) options are possible for each $(c_i, d_i)$ pair. $w_3$ uses $f$, which has been set to **flip** by $w_1$, to eliminate (i) and (ii), and forces each $(c_i, d_i)$ pair to be either $(1 \to 0, 0 \to 1)$ or $(0 \to 1, 1 \to 0)$. The crucial observation is that for each $i$, the direction of $(c_i, d_i)$ *in relationship to* the direction of $(a,b)$ is left unspecified. Utilizing this degree of freedom, $w_4$ sets the literal letters $x_i$, $\bar{x_i}$ in a particular way: For each $i$, $(x_i, \bar{x_i})$ is forced to be either $(\mathbf{1}, \mathbf{id})$ (see Figure C.3) or $(\mathbf{id}, \mathbf{1})$ (see Figure C.4), corresponding respectively to the assignment of 'true' and 'false' to the variable $x_i$. In this way, stochastic matrices assigning a near optimal generation probability on $w_0 w_1 w_2 w_3 w_4$ are forced to be *close to* one of these deterministic stochastic matrices, which we called earlier 'canonical stochastic matrices.' Observe that there are $2^n$ of these, corresponding to the $2^n$ truth assignments on $n$ variables. Finally, $w_5$ is designed so that any canonical matrix 'satisfying' $F$ will assign it a probability exceeding some bound, whereas any canonical matrix not satisfying $F$ will assign it probability zero. Here we use a trick related to that used by Angluin in [Ang89]. For a given clause $C_j$ in $F$, $(x_1, \bar{x_4}, x_2)$ for example, $w_{4,j}$ is $ab(x_1 \bar{x_4} x_2)b$. Since each $w_{4,j}$ is preceded by an '$ab$' and followed by a '$b$,' $(x_1 \bar{x_4} x_2)$ is forced to map 0 to 1. Now the crucial observation is that if all three letters $x_1, \bar{x_4}$, and $x_2$ are set **id** (or the corresponding truth assingment assigns all three literals 'false') then so is $(x_1, \bar{x_4}, x_2)$, and hence it must map 0 to 0. Since for any non-satisfying truth assignment there is a clause not satisfied by it, any canonical matrix corresponding to a non-satisfying assignment assigns the string $w$ probability zero. Hence if $F$ is unsatisfiable then any canonical matrix must assign zero probability on $w = \omega_\alpha(F)$.

**Proof of Theorem 5.1**

Now we make our argument precise. We begin by defining the notion of canonical (deterministic) stochastic matrices.

**Definition 5.1 (Canonical Stochastic Matrices):** *Let $T_n$ denote the set of truth assignments on $n$ variables, each mapping $\{X_i \mid 1 \le i \le n\}$ to $\{T, F\}$. For each $\tau \in T_n$, we define the 'canonical stochastic matrix for $\tau$,' written $M_\tau$ as follows.*

1. *$M_\tau$ is deterministic.*
2. *$(a,b) =_{M_\tau} (0 \to 1, 1 \to 0)$, $f =_{M_\tau}$ **flip**, and $e =_{M_\tau}$ **0**.*

over $\Sigma_n$. We begin by describing the alphabet $\Sigma_n$. The letters in $\Sigma_n$ can be classified into the following categories: *global control letters* $a$ and $b$, *fixed function letters* $f$ and $e$, and for each variable $x_i$, *literal letters* $x_i, \bar{x}_i$, *control letters* $c_i, d_i$, and a *dummy letter* $v_i$. The literal letters directly correspond to the literals in the formula $F$, whereas the remaining letters play a support role. We refer to the two states as state 0 and state 1, and let $S$ denote the state set $\{0,1\}$. We now give the string $\omega_\alpha(F)$, or $w$ for short.

$$
\begin{aligned}
w &= w_0 w_1 w_2 w_3 w_4 w_5 \\
w_0 &= (ab)^{k_0} \\
w_1 &= (afabfb)^{k_1} \\
w_2 &= \textstyle\prod_{i=1}^{n} v_i (c_i d_i)^{k_2} v_i \\
w_3 &= \textstyle\prod_{i=1}^{n} (c_i f c_i d_i f d_i)^{k_3} \\
w_4 &= \textstyle\prod_{i=1}^{n} (a x_i b x_i c_i \bar{x}_i d_i \bar{x}_i b e a e a b)^{k_4} \\
w_5 &= \textstyle\prod_{j=1}^{s} (ab(l_{j,1} l_{j,2} l_{j,3})b)^{k_5} \\
&\quad \text{(where } l_{j,k} \text{ denotes the } k\text{-th literal in the } j\text{-th clause of } F.)
\end{aligned}
$$

$$
\begin{aligned}
k_5 &= \log K \\
k_4 &= s 2^{11} (\log K + \mid w_5 \mid) \\
k_3 &= s 2^{11} (\log K + \mid w_4 w_5 \mid) \\
k_2 &= s 2^{11} (\log K + \mid w_3 w_4 w_5 \mid) \\
k_1 &= s 2^{11} (\log K + \mid w_2 w_3 w_4 w_5 \mid) \\
k_0 &= s 2^{11} (\log K + \mid w_1 w_2 w_3 w_4 w_5 \mid)
\end{aligned}
$$

The $k_i$'s were chosen as moderately growing functions of $K$, the intended *gap*, so that $|w| \leq p(n,s) \log K$ for some polynomial $p$. If $\alpha > 1$ then reset it to 1. Now $\omega_\alpha$ is obtained by setting $\log K = p(n,s)^{\frac{1-\alpha}{\alpha}}$ which makes $\mid w \mid$ less than or equal to $p(n,s)^{\frac{1}{\alpha}}$ and hence $\log K \geq \mid w \mid^{1-\alpha}$ as desired. We may assume without loss of generality that $\log K$ as set above is an integer because if $p(n,s)^{\frac{1-\alpha}{\alpha}}$ was not an integer then by at most halving $\alpha$ one could find a smaller $\alpha'$ such that $p(n,s)^{\frac{1-\alpha'}{\alpha'}}$ is an integer.

To explain the intent of the transformation given above, we need to introduce some terminology concerning stochastic matrices. A *deterministic* stochastic matrix is a stochastic matrix in which for each state $s$ and each letter $z$, there is at most one transition with a positive probability out of $s$ labeled with $z$. Thus any deterministic stochastic matrix $M$, induces for each letter $z$ a (possibly partial) transition function from states to states. Since for this proof the number of states is 2, these transition functions are Boolean. If the letter has transitions out of each of the two states, then the associated function must be one of the four possible total boolean functions over one variable. Borrowing Angluin's terminology [Ang89], these are: 0-reset (**0**), 1-reset (**1**), *identity* (**id**) and *flip* (**flip**), and are defined as follows: (i) $\mathbf{0}(0) = \mathbf{0}(1) = 0$, (ii) $\mathbf{1}(0) = \mathbf{1}(1) = 1$, (iii) $\mathbf{id}(0) = 0, \mathbf{id}(1) = 1$, and (iv) $\mathbf{flip}(0) = 1, \mathbf{flip}(1) = 0$. (See Figure C.2.) With a letter that has a transition out of only one of the two states we associate one of the four possible *partial* one-variable boolean functions: (v) $0 \to 0$, (vi) $0 \to 1$, (vii) $1 \to 0$, and (viii) $1 \to 1$. We refer to letters with a total (partial) transition function as *total* (respectively *partial*) letters. If in a stochastic matrix $M$ a letter $z$ is associated with **flip**, for example, we write $z =_M \mathbf{flip}$. If a pair of letters $x$ and $y$ are **id** and **flip** respectively, then we write $(x,y) =_M (\mathbf{id}, \mathbf{flip})$. (When $M$ is clear from the context, we will drop $M$ and write $(x,y) = (\mathbf{id}, \mathbf{flip})$.)

As explained in the high level description at the beginning of this section, the idea of our proof is that $w_a = w_0 w_1 w_2 w_3 w_4$ forces any near-optimal stochastic matrix to be *close to* one

**Proof of Corollary 5.1 given Theorem 5.1**

Suppose that a training algorithm $A$ trains 2-state PA constraints in time polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, n$ and $a$. Then, by Theorem 4.1, it follows that there exists an algorithm $B$ which approximates the MLM problem within factor $1+\epsilon$ in random time polynomial in $n$, $m$, and $a$. We can then use $B$ to solve 3-SAT in random polynomial time, using the reduction of Theorem 5.1. Thus 3-SAT, which is NP-complete, would be shown to be in **RP**, which would imply that **RP** = **NP**. □

The proof of Theorem 5.1 is lengthy, so we will begin by giving a very high level description of the proof. We will then give a proof sketch, introducing some key definitions and then give the formal proof. We reduce 3-SAT, the satisfiability problem for 3-CNF formulas, to the approximation problem for the single string MLM problem for the 2-state complete constraints with a guaranteed approximation ratio of $2^{|w|^{1-\alpha}}$ for any fixed $\alpha > 1$. For an arbitrary $\alpha > 0$, we exhibit a polynomial time reduction which maps any CNF formula $F$ to a string $w$ such that the maximum probability assignable by a 2-state stochastic matrix on the string $w$ is at least $C(F)$ if the formula is satisfiable, and less than $\frac{1}{2^{|w|^{1-\alpha}}}C(F)$ otherwise, where $C(F)$ is easily computable. Thus any approximation algorithm for the single string MLM for this class with a guaranteed approximation ratio of $2^{|w|^{1-\alpha}}$ can be used to solve 3-SAT, and hence the problem is NP-hard. The rough idea of the reduction is as follows: Let us imagine that an agent (the hypothetical approximation algorithm) is attempting to find a stochastic matrix which assigns an approximately maximum probability to the string. The string $w$ is conceptually divided into two parts: $w = w_a w_b$. We design the first half of the string in such a way that if the agent is to maximize the probability on it, it will have to 'lean towards' one of $2^n$ *deterministic* stochastic matrices of a particular kind, which we call '*canonical stochastic matrices*.' These correspond to the $2^n$ truth assignments for the $n$ variables in $F$. More precisely, we define a notion of distance between stochastic matrices and show that if the matrix in question is $\Delta$-far from the closest canonical stochastic matrix, then the probability assigned on the first half is less than the optimal by roughly a factor of $(1-\Delta)^{|w_a|}$. Now the second half of the string 'tests' whether any of these canonical matrices corresponds to a satisfying assignment for the formula $F$. For any canonical stochastic matrix, or one that is $\Delta$-close to one, the generation probability assigned on the second half of the string will be high if the corresponding truth assignment satisfies $F$ and otherwise will be less by roughly a multiplicative factor of $\Delta^{|w_b|}$. Thus the agent faces the following dilemma: (i) If it tries to be near-optimal on the first half and chooses a small value of $\Delta$, that is, its stochastic matrix is in fact close to one of the canonical matrices, then it would have to in effect solve 3-SAT to determine an approximately maximum generation probability assignable on the second part. (ii) If it tries to avoid solving 3-SAT on the second part and chooses a large enough value of $\Delta$, that is, its stochastic matrix is sufficiently far from any of the canonical matrices, then it loses so much probability on the first half that it cannot guarantee an approximately optimal generation probability.

**Proof Sketch of Theorem 5.1**

We now give a fuller proof sketch, defining key notions used in the proof in so doing. For each $\alpha > 0$, we exhibit a transformation $\omega_\alpha$ mapping any 3-CNF-formula $F$ to a string over an alphabet that depends on the number of variables, for which a gap of factor $K = 2^{|\omega_\alpha(F)|^{1-\alpha}}$ is forced in the optimal solution depending on the satisfiability of the formula. Let $n$ be the number of variables in $F$, and $s$ the number of clauses in $F$. We let $\Sigma_n$ denote the alphabet used for $\omega_\alpha(F)$, and $\mathcal{M}_n$ the set of all 2-state stochastic matrices

We wish to maximize the above expression, subject to the constraints:

$$\forall i \in S \sum_{j \in S, z \in \Sigma} M_P(i,j,z) = 1$$

$$\sum_{i \in S} \pi_P(i) = 1$$

We can show that this is maximized when we define $P$ as follows, using the following well-known technical lemma, which we state without proof.

$$\forall i,j \in S \, \forall z \in \Sigma \; M_P(i,j,z) = \frac{\sharp(j,z \mid i)}{\sharp(i)}$$

$$\forall i \in S \, \pi_P(i) = \frac{\sharp_0(i)}{m}$$

**Lemma 4.2:** *A function of the form* $f(x_1,...,x_n) = x_1^{a_1} \cdot x_2^{a_2} \cdot ... \cdot x_n^{a_n}$ *subject to the constraint* $x_1 + x_2 + ... + x_n = c$ *attains its maximum when* $x_i = (\frac{a_i}{\sum_{i=1}^{n} a_i}) \cdot c$.

¿From the above observation and Theorem 4.1, it follows that the class of deterministic constraints is polynomially trainable. The sample complexity obtained in this way can probably be improved significantly for this restricted case.

**Corollary 4.2:** *The class of deterministic PA constraints is polynomially trainable.*

## 5 Computational Complexity of Training Probabilistic Automata

The hardness of training the 2-state complete PA constraints is shown via a strong non-approximability result for the *single string* MLM problem for the same class. We emphasize again that the training problem for the class of $s$-state complete constraints is the natural problem of finding a near optimal probabilistic automaton of a *given number of states*. It should be noted that showing that the class of complete constraints is hard to train is much more significant and also more difficult than constructing an artificial class of constraints that is hard to train.

**Theorem 5.1:** *For any* $\alpha > 0$, *the single string MLM problem for the class of 2-state complete PA constraints is not approximable within* $2^{|w|^{1-\alpha}}$ *in time polynomial in the alphabet size* $a$ *and* $\mid w \mid$, *where* $w$ *is the input word, unless* $\mathbf{P} = \mathbf{NP}$.

As we noted in Section 1, Theorem 5.1 is a strong non-approximability result, since guaranteed approximation ratio of $2^{|w|}$ for the MLM problem for 2-state PAs is trivially achievable.

**Theorem 5.2:** *For arbitrary* $s \in N$, *the single string MLM problem for the* $s$*-state complete constraints is approximable within* $s^{|w|}$ *in time polynomial in the alphabet size* $a$ *and* $\mid w \mid$, *where* $w$ *is the input word.*

The proof of Theorem 5.2 is simple but uses one of the technical lemmas to be stated and used in the proof of Theorem 5.1, so we defer the proof of Theorem 5.2 till after the proof of Theorem 5.1.

**Corollary 5.1:** *The class of 2-state complete PA constraints is not polynomially trainable, unless* $\mathbf{RP} = \mathbf{NP}$.

$$\begin{aligned}\frac{\prod_{i=1}^{m} OPT_T(w_i)}{\prod_{i=1}^{m} Q(w_i)} &\leq 2^{\frac{\epsilon}{2}\log e} \\ &= e^{\frac{\epsilon}{2}} \\ &\leq 1+\epsilon, \text{ for } \epsilon \in (0,1]. \end{aligned} \tag{4.13}$$

But then $OPT_S$ and $OPT_T$ assign the same likelihood on $S$ as shown below.

$$\begin{aligned}\prod_{i=1}^{m} OPT_S(w_i) &= max\{\prod_{i=1}^{m} P(w_i) : P \in \mathcal{PA}(C)\} \\ &= (max\{\prod_{i=1}^{mr} P(v_i) : P \in \mathcal{PA}(C)\})^{\frac{1}{r}} \\ &= (\prod_{i=1}^{mr} OPT_T(v_i))^{\frac{1}{r}} \\ &= \prod_{i=1}^{m} OPT_T(w_i) \end{aligned} \tag{4.14}$$

By plugging (4.14) in (4.13), we finally obtain:

$$\frac{\prod_{i=1}^{m} OPT_S(w_i)}{\prod_{i=1}^{m} Q(w_i)} \leq 1+\epsilon$$

$(3 \rightarrow 4)$ This is obvious from the definitions. □

¿From Theorem 4.1 and Corollary 3.2, the following positive result on the approximation problem for the sample MLM problem follows at once.

**Corollary 4.1:** *The sample MLM problem for any finite class of PA constraints is approximable within a factor of* $1+\epsilon$ *for any* $\epsilon > 0$*, in time polynomial in* $\frac{1}{\epsilon}$ *and the total length of the sample.*

Note that this does not give rise to a practical training algorithm, since the running time of the algorithm we exhibit is exponential in the number of probability parameters specified by the input constraint, which may grow quite large in practical applications.

We mention a simple class of constraints which can be trained efficiently. A constraint $C = \langle I, G\rangle$ is said to be *deterministic*, if for a given state $i$ and a given letter $a$, there is at most one transition from $i$ labeled with $a$, in $G$. It is well known in the literature that the MLM problem for the class of deterministic constraints is solvable in polynomial time. We repeat the proof of this fact briefly, as it makes use of a lemma that will be used again in the hardness proof in Section 5. Let a deterministic constraint $C$ and a sample $S = \langle w_1, ..., w_m\rangle$ be given. For each $w_k \in S$, let $\sigma^k$ be the unique path in $C$ labeled with $w_k$ (if one exists). Here, note that since $C$ is deterministic, there is at most one path for each word, and if there is a word in $S$ for which there is no path in $C$, then the maximum probability assignable on $S$ is zero and the problem is trivial. Let $\sharp(j, z \mid i)$ denote the total number of times the transition to state $j$ outputting letter $z$ was taken from state $i$ in all of pathes $\sigma^k$, and $\sharp(i)$ denote the number of times a transition was taken from state $i$ in all of pathes $\sigma^k$. Let $\sharp_0(i)$ denote the total number of times state $i$ was entered as the *initial* state in all paths $\sigma^k$. Clearly $\sum_{j\in S, z\in\Sigma} \sharp(j, z \mid i) = \sharp(i)$ and $\sum_{i\in S} \sharp_0(i) = m$. Now note that for any probabilistic automaton $P$, we can calculate $P(S) = \prod_{i=1}^{m} P(w_i)$ as follows.

$$P(S) = \prod_{i\in S} \pi_P(i)^{\sharp_0(i)} \prod_{i\in S, j\in S, z\in\Sigma} M_P(i,j,z)^{\sharp(j,z|i)}$$

Hence from (4.5) and (4.10), it follows that whenever $m \geq max\{q(\frac{4}{\epsilon}, \log\frac{2}{\delta}, n, t), \log\frac{2}{\delta}\}$, with probability at least $1-\delta$,

$$E_D(\log\frac{1}{Q_B(\cdot)}) - E_D(\log\frac{1}{Opt(\cdot)}) \leq \frac{\epsilon}{2} + \frac{5(n+1)}{m-1} \tag{4.11}$$

Hence, whenever $m \geq max\{q(\frac{\epsilon}{4}, \log\frac{2}{\delta}, n, t), \frac{10(n+1)}{\epsilon} + 1, \log\frac{2}{\delta}\}$, with probability at least $1-\delta$,

$$E_D(\log\frac{1}{Q_B(\cdot)}) - E_D(\log\frac{1}{Opt(\cdot)}) \leq \epsilon$$

We have thus shown that the above algorithm trains $C$, which was an arbitrary constraint in $\mathcal{C}$, with sample complexity polynomial in $\frac{1}{\epsilon}, \log\frac{1}{\delta}, n, t$ and running time polynomial in the same parameters. It is easy to convert such an algorithm to one whose running time is polynomial in the total sample length and which has sample complexity still polynomial in $\frac{1}{\epsilon}, \log\frac{1}{\delta}, n, t$.

$(2 \rightarrow 1)$

This is obvious from the definitions.

Thus far, we have shown that the first three statements of the theorem are equivalent. Now we proceed to show that *3* is equivalent to *4*.

$(4 \rightarrow 3)$ Suppose that algorithm $A$, running in random time polynomial in $\frac{1}{\epsilon}, n, t, m$, approximates the sample MLM problem for $\mathcal{C}$ within factor $2^{p(n,t)m^\alpha}$ for some polynomial $p$. Let a constraint $C \in \mathcal{C}$ and a sample $S = \langle w_1, ..., w_m\rangle$ be given. Again, assume without loss of generality that $\prod_{i=1}^m OPT_S(w_i) = max\{\prod_{i=1}^m P(w_i) : P \in \mathcal{PA}(C)\}$ is positive. We then repeat the sample $r = \lceil(\frac{2}{\epsilon}(\ln 2)p(n,t)m^\alpha)^{\frac{1}{1-\alpha}}\rceil$ times to obtain a new sample $T = \langle v_1, ..., v_{rm}\rangle$ of length $rm$, and feed this into $A$ to obtain a hypothesis $Q$. Then, by definition, $Q$ must satisfy:

$$\frac{\prod_{i=1}^{rm} OPT_T(t_i)}{\prod_{i=1}^{rm} Q(t_i)} \leq 2^{p(n,t)(rm)^\alpha}$$

where we used $OPT_T$ to denote the PA in $\mathcal{PA}(C)$ that assigns the maximum probability on the sample $T$. Since each example of $S$ is repeated exactly $r$ times in $T$,

$$\begin{aligned}\frac{\prod_{i=1}^m OPT_T(w_i)}{\prod_{i=1}^m Q(w_i)} &\leq (2^{p(n,t)(mr)^\alpha})^{\frac{1}{r}} \\ &\leq 2^{p(n,t)m^\alpha r^{\alpha-1}}\end{aligned} \tag{4.12}$$

By substituting $r$ into the exponent in (4.12) we obtain:

$$\begin{aligned}\log\frac{\prod_{i=1}^m OPT_T(w_i)}{\prod_{i=1}^m Q(w_i)} &\leq p(n,t)m^\alpha\lceil(\frac{2}{\epsilon}(\ln 2)p(n,t)m^\alpha)^{\frac{1}{1-\alpha}}\rceil^{\alpha-1} \\ &\leq p(n,t)m^\alpha(\frac{2}{\epsilon}(\ln 2)p(n,t)m^\alpha)^{\frac{1}{1-\alpha}})^{\alpha-1}, \text{ since } \alpha - 1 < 0 \\ &= p(n,t)m^\alpha(\frac{2}{\epsilon}(\ln 2)p(n,t)m^\alpha)^{-1} \\ &= \frac{\epsilon}{2}\log e\end{aligned}$$

Hence, we have:

Now, using the trick of shifting, rounding-off and stochastic correction as done in the proof of Theorem 3.1, we obtain from $P$ a member $Q_B$ of $\mathcal{FPA}(C)_m$, such that

$$\forall x \in \Sigma^n \ \log \frac{1}{Q_B(x)} - \log \frac{1}{P(x)} \leq \frac{2(n+1)}{m-1} \tag{4.4}$$

From (4.3) and (4.4), (4.2) follows.

Next we can show that any algorithm having the above performance indeed trains $\mathcal{C}$. This can be shown similarly to the way the algorithm exhibited in the proof of Theorem 3.1 was proved to successfully train $\mathcal{C}$. Let us first recall the notation of that proof: We denoted by $Q$ the output of the training algorithm which, for the input sample $S$, minimized $E_{\widehat{D}_S}(\log \frac{1}{F(\cdot)})$ over $F \in \mathcal{FPA}(C)_m$. The PA $Opt$ was a member of $\mathcal{PA}$ with the minimum divergence with respect to the target distribution. $F^*$ denoted a member of $\mathcal{FPA}(C)_m$ that is closest to $Opt$. In order to prove that $E_D(\log \frac{1}{Q(\cdot)})$ is close to $E_D(\log \frac{1}{Opt(\cdot)})$ with high probability (inequality (3.9)), we showed in the proof of Theorem 3.1 that:

$$E_D(\log \frac{1}{F^*(\cdot)}) - E_D(\log \frac{1}{Opt(\cdot)}) \leq \frac{2(n+1)}{m-1} \tag{4.5}$$

and whenever $m \geq q(\frac{4}{\epsilon}, \frac{1}{\delta}, n, t)$ with probability at least $1-\delta$,

$$E_D(\log \frac{1}{Q(\cdot)}) - E_D(\log \frac{1}{F^*(\cdot)}) \leq \frac{\epsilon}{2} \tag{4.6}$$

Since (4.5) always holds, we only need to show an analogue of (4.6) for any algorithm satisfying (4.2). Recall that in the earlier proof, in order to show (3.9), we applied triangle inequality to the inequalities (3.6), (3.7), and (3.8). (3.6) and (3.7) followed from the uniform convergence for $\mathcal{FPA}(C)_m$, and (3.8) followed from the fact that the training algorithm minimized $E_{\widehat{D}_S}(\log \frac{1}{Q(\cdot)})$ within $\mathcal{FPA}(C)_m$.

In this proof, the output $Q_B$ of our training algorithm *approximately* minimizes $E_{\widehat{D}_S}(\log \frac{1}{Q(\cdot)})$ within $\mathcal{FPA}(C)_m$. More precisely, by the optimality of $OPT$, it follows from (4.2) that the following holds with probability at least $1-(\frac{1}{2})^m$:

$$E_{\widehat{D}_S}(\log \frac{1}{Q_B(\cdot)}) - E_{\widehat{D}_S}(\log \frac{1}{F^*(\cdot)}) \leq \frac{3(n+1)}{m-1} \tag{4.7}$$

As before, for $m \geq q(\frac{4}{\epsilon}, \log \frac{2}{\delta}, n, t)$ both of the following hold with probability at least $1-\frac{\delta}{2}$:

$$E_D(\log \frac{1}{Q_B(\cdot)}) - E_{\widehat{D}_S}(\log \frac{1}{Q_B(\cdot)}) \leq \frac{\epsilon}{4} \tag{4.8}$$

$$E_{\widehat{D}_S}(\log \frac{1}{F^*(\cdot)}) - E_D(\log \frac{1}{F^*(\cdot)}) \leq \frac{\epsilon}{4} \tag{4.9}$$

By summing up the inequalities (4.7), (4.8), and (4.9), and noting that $(\frac{1}{2})^m \leq \frac{\delta}{2}$ when $m \geq \log \frac{2}{\delta}$, we obtain that the following holds with probability at least $1-\delta$ whenever $m \geq max\{q(\frac{4}{\epsilon}, \log \frac{2}{\delta}, n, t), \log \frac{2}{\delta}\}$:

$$E_D(\log \frac{1}{Q_B(\cdot)}) - E_D(\log \frac{1}{F^*(\cdot)}) \leq \frac{\epsilon}{2} + \frac{3(n+1)}{m-1} \tag{4.10}$$

We thus obtain $\sum_{i=1}^{m} \log \frac{OPT(w_i)}{Q(w_i)} \leq \frac{\epsilon}{2}$. So for $\epsilon \in (0,1]$, we have

$$\prod_{i=1}^{m} \frac{OPT(w_i)}{Q(w_i)} \leq 2^{\frac{\epsilon}{2}} \leq 1+\epsilon$$

Since $OPT$ is a probabilistic automaton in $\mathcal{PA}(C)$ minimizing the divergence with respect to $\widehat{D}_S$ on $S$, by the 'classical equivalence' given in Lemma 2.1, it is also an optimal solution in $\mathcal{PA}(C)$ of the sample MLM problem on $S$. Thus we have obtained a $1+\epsilon$ approximation to the sample MLM problem with probability at least a half. The running time of $B$ is clearly bounded by a polynomial in $\frac{1}{\epsilon}, n, t, m$, since $m'$ is polynomial in these parameters, the time spent on generating the sample $S'$ using the algorithm $U$ is polynomial in $n$, $m$ and $m'$, and the running time of $A$ is polynomial in the total length of the sample $S'$, i.e. polynomial in $m'$ and $n$.

$(3 \rightarrow 2)$

We will use the hypothetical approximation algorithm for the MLM problem for a class of constraints $\mathcal{C}$ to construct $B$ that trains $\mathcal{C}$. Let $D$ be the target distribution over $\Sigma^n$ and let a constraint $C \in \mathcal{C}$ be given as input. Assume without loss of generality that $min\{d_{KL}(D,P) : P \in \mathcal{PA}(C)\}$ is finite, since otherwise the training problem is trivial. The finiteness of $min\{d_{KL}(D,P) : P \in \mathcal{PA}(C)\}$ guarantees that for each $x \in \Sigma^n$ assigned a positive probability by $D$, there is a path in $C$ labeled with $x$. Hence it follows that for an arbitrary sample $S$ generated by $D$, $max\{\prod_{i=1}^{m} P(w_i) : P \in \mathcal{PA}(C)\}$ is positive. First we show that any approximation algorithm (in the sense of *3* in the statement of the lemma) can be modified so as to output a PA $Q_B$ in the finite class $\mathcal{FPA}(C)_m$, defined for $C$ as in the proof of Theorem 3.1, which with probability at least $1-(\frac{1}{2})^m$ satisfies:

$$E_{\widehat{D}_S}(\log \frac{1}{Q_B(\cdot)}) - E_{\widehat{D}_S}(\log \frac{1}{OPT(\cdot)}) \leq \frac{3(n+1)}{m-1} \tag{4.2}$$

where $OPT \in \mathcal{PA}(C)$ is an optimal solution for the sample MLM problem with input sample $S$ satisfying the input constraint $C$. Second, we show that in fact such an algorithm is a training algorithm for PAs with a slightly larger sample complexity than the training algorithm exhibited in the proof of Theorem 3.1.

It is easy to verify the first of these two claims. First note that any algorithm which approximates the sample MLM problem for $\mathcal{C}$ within factor $1+\epsilon$ with probability at least a half in time polynomial in $\frac{1}{\epsilon}, n, t$ and $m$ can be boosted to one which achieves the same approximation factor with probability at least $1-\delta$ in time polynomial in $\log \frac{1}{\delta}, \frac{1}{\epsilon}, t, n$ and $m$. This can be done by iteratively running the former algorithm $\lceil \log \frac{1}{\delta} \rceil$ times and then selecting, from among its outputs, one that assigns the maximum likelihood on the input sample. (Note that using dynamic programming it is easy to compute for a given sample and PA the likelihood of that sample in time polynomial in the total length of the sample and $t$.) Now set $\epsilon = n$ and $\delta = (\frac{1}{2})^m$ and run this algorithm on the input sample $S$ and obtain a PA $P$, which with probability at least $1-(\frac{1}{2})^m$ approximates the sample MLM problem for $\mathcal{C}$ within factor $1+n$:

$$\prod_{i=1}^{m} \frac{OPT(w_i)}{P(w_i)} \leq 1+n \leq e^{1+n}$$

Hence we have:

$$E_{\widehat{D}_S}(\log \frac{1}{P(\cdot)}) - E_{\widehat{D}_S}(\log \frac{1}{OPT(\cdot)}) = \frac{1}{m}\sum_{i=1}^{m} \log \frac{OPT(w_i)}{P(w_i)} \leq \frac{n+1}{m} < \frac{n+1}{m-1} \tag{4.3}$$

Let $\epsilon > 0$, a finite sample $S = \langle w_1, ..., w_m \rangle$ with $|w_i| = n$ for each $w_i$, and a constraint $C \in \mathcal{C}$ be given. We use the hypothetical training algorithm $A$ on the empirical distribution $\widehat{D}_S$ of $S$, as defined in Definition 2.1. We can assume without loss of generality that $min\{d_{KL}(\widehat{D}_S, P) : P \in \mathcal{PA}(C)\}$ is finite, since otherwise $max\{\prod_{i=1}^{m} P(w_i) : P \in \mathcal{PA}(C)\} = 0$ and the MLM problem is trivial. $B$ then computes a sample size $m'$ for $A$ large enough for accuracy $\frac{\epsilon}{2m}$ and confidence $1 - \frac{1}{4}$. For example, $m' = \lceil q(\frac{2m}{\epsilon}, 4, n, t) \rceil$ suffices where $q(...)$ is an upper bound on the sample complexity of the hypothetical training algorithm $A$, which by assumption is polynomially bounded. Now, $B$ gives $A$ a sample $S'$ of size $m'$ obtained by sampling from $S$ according to $\widehat{D}_S$, the empirical distribution of $S$, or the uniform distribution over the $m$ elements of the sequence $S$. Here note that while it is not always possible to simulate the empirical distribution of a finite sample with a fair coin, there exists an algorithm which generates a new sample of an arbitrary size according to the empirical distribution of the original sample with high probability. We make this precise below.

**Lemma 4.1:** *There exists a randomized algorithm, $U$, which, given a finite sample $S = \{w_1, ..., w_m\}$ of size $m$, an integer $m'$, and a confidence parameter $\nu > 0$, always terminates in time polynomial in $m$, $m'$ and $\frac{1}{\nu}$ and outputs a sample $S'$ of size $m'$ which with probability at least $1 - \nu$ is drawn according to the empirical distribution $\widehat{D}_S$ of $S$.*

**Proof of Lemma 4.1** The algorithm $U$ first calculates $i = \lceil \log m \rceil$ and iterates the following: $U$ flips a fair coin $i$ times to obtain a bit string $x$ of length $i$. If $x \leq m$, then $U$ appends $w_x$ to the end of the sequence $S'$, and does nothing otherwise. It is clear that each example of $S'$ is drawn independently at random according to $\widehat{D}_S$. Notice that at each iteration the length of $S'$ increases by one with probability at least a half. It is easy to see, by applying Chernoff's bound (c.f. [Val84]) that in $p(m', \frac{1}{\nu})$ many iterations, the length of $S'$ becomes $m'$ with probability at least $1 - \nu$, where $p$ is a certain polynomial. If this fails to occur, i.e. the length of $S'$ is shorter than $m'$ after $p(m', \frac{1}{\nu})$ many iterations, $U$ pads $S'$ with arbitrary examples to make its length $m'$.

**End of proof of Lemma 4.1**

We run $U$ on $S$, $m'$, and $\frac{1}{3}$ to obtain a sample $S'$, and let $Q$ be the output obtained by running $A$ on $S'$. Thus, $A$ is run on a sample of size $m'$ generated with respect to $\widehat{D}_S$ with probability at least $\frac{2}{3}$. Furthermore, when this is the case, by the performance guarantee on $A$, the divergence of $Q$ with respect to $\widehat{D}_S$ is $\frac{\epsilon}{2m}$ close to the divergence of the best PA satisfying $C$, with probability at least $\frac{3}{4}$. Hence, with probability at least $\frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}$, the following holds:

$$d_{KL}(\widehat{D}_S, P) - d_{KL}(\widehat{D}_S, OPT) = \sum_{x \in \Sigma^n} \widehat{D}_S(x) \log \frac{OPT(x)}{P(x)} \leq \frac{\epsilon}{2m} \tag{4.1}$$

where $OPT$ is a PA in $\mathcal{PA}(C)$ satisfying:

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

Substituting $\widehat{D}_S(x) = \frac{\sharp(x,S)}{m}$ in (4.1) above,

$$d_{KL}(\widehat{D}_S, Q) - d_{KL}(\widehat{D}_S, OPT) = \frac{1}{m} \sum_{i=1}^{m} \log \frac{OPT(w_i)}{Q(w_i)} \leq \frac{\epsilon}{2m}$$

## 4 The Equivalence Between the Training and Maximum Likelihood Model Problems

The sample complexity bounds of Theorem 3.1 can be used to establish the equivalence between the efficient trainability of a class of constraints and the efficient approximability of the sample MLM problem for the same class. We first define what it means for a randomized algorithm to *approximate* the sample MLM problem within a given factor.

**Definition 4.1 (Approximate Sample MLM problem):** *A randomized algorithm $A$ is said to approximate the sample MLM problem for a class of constraints $\mathcal{C}$ within factor $K$, possibly a function of various parameters of the problem, in random $T(...)$ time, if given a constraint $C \in \mathcal{C}$, any $\epsilon > 0$, and an input sample $S = \langle w_1, ..., w_m \rangle$ of $\Sigma^n$ for some $n > 0$ and some finite alphabet $\Sigma$, $A$ terminates in $T(...)$ many steps and outputs a PA $Q \in \mathcal{PA}(C)$, which with probability at least a half satisfies:*

$$\frac{\prod_{i=1}^{m} OPT(w_i)}{\prod_{i=1}^{m} Q(w_i)} \leq K$$

*where $OPT$ is a member of $\mathcal{PA}(C)$ which maximizes the likelihood of $S$, i.e.*

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

*By convention, we let $\frac{0}{0} = 1$*

As before, $OPT$ is guaranteed to exist because of the compactness of $\mathcal{PA}(C)$ and the continuity of the likelihood function on a finite sample.

**Theorem 4.1:** *For an arbitrary class of PA constraints $\mathcal{C}$, the following three statements are equivalent. Below, we let $t$ denote the size of the input constraint $C \in \mathcal{C}$ to be trained, $m$ the sample size, and $n$ the length of each example.*

1. *There exists a training algorithm for $\mathcal{C}$ with sample complexity polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, t, n$ and $m$, running in time polynomial in the total sample length.*
2. *There exists a training algorithm for $\mathcal{C}$ with sample complexity polynomial in $\frac{1}{\epsilon}, \log \frac{1}{\delta}, t, n$ and $m$, running in time polynomial in the total sample length.*
3. *The sample MLM problem for $\mathcal{C}$ is approximable within a factor $1 + \epsilon$ in random time polynomial in $\frac{1}{\epsilon}, t, n$ and $m$.*
4. *The sample MLM problem for $\mathcal{C}$ is approximable within factor $2^{p(n,t)m^{\alpha}}$ in random time polynomial in $t, n$ and $m$, for some $\alpha < 1$.*

**Proof of Theorem 4.1**
$(1 \rightarrow 3)$ The idea of the proof is as follows. We use the hypothetical training algorithm $A$ for a class of constraints $\mathcal{C}$ to construct a randomized approximation algorithm $B$ for the sample MLM problem for the same class.

In order to do this, we take advantage of the *robustness* of the training algorithm. In particular, the algorithm must meet its performance guarantee when it is fed examples generated by the *empirical distribution* observed in the input sample $S$ for the sample MLM problem. We then appeal to the classical equivalence between minimizing the divergence with respect to the empirical distribution and maximizing the likelihood of a given sample (see Lemma 2.1).

initial state with probability at least $\frac{1}{2tm}$ and each transition in the path has probability at least $\frac{1}{2tm}$. Hence $\log \frac{1}{F(x)}$ is bounded above by $(n+1)\log 2tm$. Since $\mathcal{FPA}(C)_m$ equals $\mathcal{G}'(\frac{1}{m}, \frac{1}{2tm})$, and the cardinality of $\mathcal{G}'(\gamma, \theta)$ is at most $(\frac{1}{\gamma}\ln\frac{1}{\theta})^t$, the cardinality of $\mathcal{FPA}(C)_m$ is at most $(m \ln 2tm)^t$. Plugging in $M = (n+1)\log 2tm$ and $\mid \mathcal{FPA}(C)_m \mid \leq (m \ln 2tm)^t$ into the inequality in Lemma 3.2 gives the following inequality:

$$m \geq \frac{(n+1)^2 \log^2 2tm}{\epsilon^2}(t \ln(m \ln 2tm) + \ln \frac{1}{\delta}) \tag{3.15}$$

To show that if $m$ is at least the bound in Lemma 3.1 then (3.15) holds, we first note that (3.15) is implied by the following two inequalities:

$$\frac{m}{2} \geq \frac{(n+1)^2 \log^2 2tm}{\epsilon^2} t \ln(m \ln 2tm) \tag{3.16}$$

$$\frac{m}{2} \geq \frac{(n+1)^2 \log^2 2tm}{\epsilon^2} \ln \frac{1}{\delta} \tag{3.17}$$

To get a simple argument for (3.16) we first show that

$$\log^2 2tm \ln(m \ln 2tm) \leq \frac{\ln^2 2tm}{\ln^2 2} \ln(tm \ln 2tm) \leq 2 \ln^3 2tm \tag{3.18}$$

For $tm \geq 1$ the latter inequality is equivalent to $\ln \ln 2tm \leq (2 \ln^2 2 - 1) \ln 2tm + \ln 2$ which holds for $tm = 1$ and only improves for larger $m$. We now return to the proof of (3.16): Since (3.18) holds it suffices to show

$$m \geq \alpha \ln^3 \beta m, \text{ for } \alpha = \frac{4(n+1)^2 t}{\epsilon^2} \text{ and } \beta = 2t. \tag{3.19}$$

If $m \geq \alpha \ln^3 \beta m$ holds for some choice of $m$ then it also does for all larger $m$. We set $m$ to $8\alpha \ln^3 8\alpha\beta$ which is the first bound in the maximization clause of $q(\frac{1}{\epsilon}, \log \frac{1}{\delta}, n, t)$. Then $m \geq \alpha \ln^3 \beta m$ is equivalent to $8\alpha\beta \geq \ln^3 8\alpha\beta$. Since $n \geq 1$ and $t \geq 2$, $\alpha\beta = \frac{4(n+1)^2 t}{\epsilon^2} \cdot (2t)$ is always at least 64. Now observe that the last inequality holds for $\alpha\beta = 64$ because $(\ln 512)^3 \approx 242.77 < 512$, and thus for all larger $\alpha\beta$. We conclude that (3.19) and (3.16) hold.

The proof of (3.17) is simpler. We need to show

$$m \geq \alpha \log^2 \beta m, \text{ for } \alpha = \frac{2(n+1)^2}{\epsilon^2} \ln \frac{1}{\delta} \text{ and } \beta = 2t.$$

One can show that if $m \geq 4\alpha \log^2 2\alpha\beta$ (which is the second bound in the maximization clause of $q(\frac{1}{\epsilon}, \log \frac{1}{\delta}, n, t)$) then $m \geq \alpha \ln^2 \beta m$.

To complete the proof of Theorem 3.1 observe that $q(\frac{4}{\epsilon}, \log \frac{1}{\delta}, n, t)$ of Lemma 3.1 is at least $\frac{4(n+1)}{\epsilon} + 1$ and fulfills the order bound promised in the statement of the theorem. □

As an immediate corollary to the proof of Theorem 3.1, we have the following positive result for training PAs of a fixed number of parameters.

**Corollary 3.2:** *Any finite class of PA constraints is polynomially trainable.*

**Proof of Corollary 3.2**

Since the sample size $m$ required in Theorem 3.1 is polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, n$ and $t$, this immediately follows from the observation that $\mid \mathcal{FPA}(C)_m \mid \leq (2m \log 2tm)^t$ is polynomial in $m$ when $t$ is bounded from above by a constant. □

This completes the proof of (3.11). The second inequality (3.12) is straightforward to verify. Let $B$ be an arbitrary member of $\mathcal{BPA}(C)_m$. We round off each initial probability and each transition probability in $B$ to a power of $(1-\frac{1}{m})$. Denote the obtained PA by $R$ and its stochastic correction by $F$. Each probability in $F$ is at least $(1-\frac{1}{m})$ times the corresponding probability in $B$ and this leads to two consequences. First, since $B \in \mathcal{BPA}(C)_m$ each initial and transition probability in $F$ is at least $\frac{1}{tm}(1-\frac{1}{m}) \geq \frac{1}{2tm}$, since the final choice of $m$ will be larger than 2. This implies that the 'nearest grid point' $R$ next to $B$ lies in $\mathcal{C}(\frac{1}{m}, \frac{1}{2tm})$ and the stochastic correction $F$ lies in $\mathcal{FPA}(C)_m = \mathcal{G}'(\frac{1}{m}, \frac{1}{2tm})$. Second, each probability in $B$ is at most $(1+\frac{1}{m-1})$ times the corresponding probability in $F$. Therefore, for any $x \in \Sigma^n$ we must have:

$$\frac{B(x)}{F(x)} \leq (1+\frac{1}{m-1})^{n+1} \leq e^{\frac{n+1}{m-1}}$$

and hence:

$$\log\frac{1}{F(x)} - \log\frac{1}{B(x)} \leq \frac{n+1}{m-1}$$

This proves (3.12) and completes the proof of (3.2).

The next lemma shows that whenever $m \geq q(\frac{4}{\epsilon}, \log\frac{1}{\delta}, n, t)$ the inequality (3.5) holds with probability at least $1-\delta$.

**Lemma 3.1:** *Let $\mathcal{F}(\mathcal{FPA}(C)_m) = \{\log\frac{1}{F(\cdot)} : F \in \mathcal{FPA}(C)_m\}$. Let $D$ be an arbitrary distribution over $\Sigma^n$ for some $n \in N$, and $D^m$ denote the product distribution induced by $D$ over $(\Sigma^n)^m$. If $S = \langle w_1, ..., w_m\rangle \in (\Sigma^n)^m$ then we let $E_{\widehat{D}_S}(f)$ denote the empirical estimate of the random variable $f$ by the sample $S$, i.e. $E_{\widehat{D}_S}(f) = \frac{\sum_{i=1}^m f(w_i)}{m}$, and let $E_D(f)$ denote the expectation (according to $D$) of $f$. Then we have, for all $\epsilon \leq 1$:*

$$\forall n, t \in N \quad \textit{If } m \geq q(\frac{1}{\epsilon}, \log\frac{1}{\delta}, n, t),$$

$$\textit{then } D^m\{S \in (\Sigma^n)^m : \exists f \in \mathcal{F}(\mathcal{FPA}(C)_m) \textit{ such that } \mid E_{\widehat{D}_S}(f) - E_D(f) \mid > \epsilon\} < \delta,$$

$$\textit{where } q(\frac{1}{\epsilon}, \log\frac{1}{\delta}, n, t) \textit{ is defined as}$$

$$max\{\frac{32(n+1)^2t}{\epsilon^2}\ln^3\frac{64t(n+1)^2t}{\epsilon^2}, \frac{8(n+1)^2\ln\frac{1}{\delta}}{\epsilon^2}\log^2\frac{8t(n+1)^2\ln\frac{1}{\delta}}{\epsilon^2}\}$$

**Proof of Lemma 3.1**
We use the following lemma due to Hoeffding. (See for example [Pol84].)

**Lemma 3.2 (Hoeffding):** *Let $\mathcal{F}$ be a finite class of bounded random variables on a set $X$, that is for each $f \in \mathcal{F}$, $f : X \to [0, M]$ for some real $M \in R$. Let $D$ be an arbitrary distribution over $X$. Then we have:*

$$\textit{If } m \geq \frac{M^2}{\epsilon^2}(\ln \mid \mathcal{F} \mid + \ln\frac{1}{\delta}) \textit{ then } D^m\{S \in X^m : \exists f \in \mathcal{F} \mid E_{\widehat{D}_S}(f) - E_D(f) \mid > \epsilon\} < \delta$$

To apply Lemma 3.2, we compute an upper bound on the random variables in $\mathcal{F}(\mathcal{FPA}(C)_m)$, and the cardinality of $\mathcal{FPA}(C)_m$. Since by our assumption at the beginning of the proof $min\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$ is finite, for any string $x$ in $\Sigma^n$ assigned a positive probability by the target distribution $D$, there is a path for $x$ in the input constraint $C$. Now any path in $C$ is produced with probability at least $(\frac{1}{2tm})^{n+1}$, since any state is chosen as the

We next verify (3.2) in two steps (inequalities (3.11) and (3.12) below). We define $\mathcal{BPA}(C)_m$, a subclass of $\mathcal{PA}(C)$ in which all transition and initial probabilities are bounded from below by[6] $\frac{1}{tm}$.

$$\mathcal{BPA}(C)_m = \{B \in \mathcal{PA}(C) : \forall i,j \in S, z \in \Sigma \; M_B(i,j,z) \geq \frac{1}{tm} \text{ and } \pi_B(i) \geq \frac{1}{tm}\}$$

We then show that for arbitrary $P$ in $\mathcal{PA}(C)$, there exists $B \in \mathcal{BPA}(C)_m$ which is close to $P$ in the following sense:

$$\forall P \in \mathcal{PA}(C) \; \exists B \in \mathcal{BPA}(C)_m \; \forall x \in \Sigma^n \; \log \frac{1}{B(x)} - \log \frac{1}{P(x)} \leq \frac{n+1}{m-1} \tag{3.11}$$

In addition, we show that for each member of $\mathcal{BPA}(C)_m$ there is one in $\mathcal{FPA}(C)_m$ that is close to it:

$$\forall B \in \mathcal{BPA}(C)_m \; \exists F \in \mathcal{FPA}(C)_m \; \forall x \in \Sigma^n \; \log \frac{1}{F(x)} - \log \frac{1}{B(x)} \leq \frac{n+1}{m-1} \tag{3.12}$$

(3.2) clearly follows from (3.11) and (3.12). To verify the first inequality (3.11), let an arbitrary $P \in \mathcal{PA}(C)$ be given. We obtain $P_m$ from $P$ by shifting each $M_P(i,j,z)$ towards the '*uniform* stochastic matrix' to obtain $P_m$ – the matrix in which each transition in $G$ out of any state $i$ receives the same probability $\frac{1}{t(i)}$, where we let $t(i)$ denote the number of transitions out of state $i$. Formally:

$$\forall i,j \in S \; \forall z \in \Sigma \; M_{P_m}(i,j,z) = \frac{(m-1) \cdot M_P(i,j,z) + \frac{1}{t(i)}}{m} \tag{3.13}$$

Similarly we shift each $\pi_P(i)$ towards the uniform distribution over $I$:

$$\forall i \in S \; \pi_{P_m}(i) = \frac{(m-1) \cdot \pi_P(i) + \frac{1}{|I|}}{m} \tag{3.14}$$

Note that for all $i,j,z$, $M_{P_m}(i,j,z) \geq \frac{1}{mt(i)} \geq \frac{1}{|G|m} \geq \frac{1}{tm}$ and $\pi_{P_m}(i) \geq \frac{1}{|I|m} \geq \frac{1}{tm}$, and hence $P_m \in \mathcal{BPA}(C)_m$. Now since each of the initial and transition probabilities in $P_m$ is at least as much as $\frac{m-1}{m}$ times the corresponding probability in $P$, the probability assigned by $P_m$ on any path (and hence on any string) is at least $(\frac{m-1}{m})^{n+1}$ times that assigned by $P$. Hence we have for any string $x \in \Sigma^n$:

$$\frac{P(x)}{P_m(x)} \leq (1 + \frac{1}{m-1})^{n+1} \leq e^{\frac{n+1}{m-1}}$$

Hence, for every $x \in \Sigma^n$, we have

$$\log \frac{1}{P_m(x)} - \log \frac{1}{P(x)} \leq \frac{n+1}{m-1}$$

[6] Note that these bounds can be made slightly smaller, which would result in a slight improvement on the sample complexity. In particular, the bound on the transition probabilities can be $\frac{1}{t^* m}$ where $t^*$ denotes the maximum number of transitions in $G$ out of any state. The bound on the initial probabilities can be $\frac{1}{|I|m}$.

Suppose that $m$ is at least $\frac{4(n+1)}{\epsilon}+1$ and thus $\frac{2(n+1)}{m-1} \leq \frac{\epsilon}{2}$. Then if we let $F^*$ be a member of $\mathcal{FPA}(C)_m$ such that for all $F \in \mathcal{FPA}(C)_m$, $d_{KL}(D,F^*) \leq d_{KL}(D,F)$, then we have:

$$d_{KL}(D,F^*) - d_{KL}(D,Opt) \leq \frac{\epsilon}{2} \tag{3.4}$$

Now by the uniform convergence for the random variables $\mathcal{F}(\mathcal{FPA}(C)_m)$ as stated in Lemma 3.1, for moderately large $m$ ($m \geq q(\frac{4}{\epsilon}, \log\frac{1}{\delta}, n, t)$ where $q$ is polynomially bounded and will be specified in Lemma 3.1), we have with probability at least $1-\delta$:

$$\forall F \in \mathcal{FPA}(C)_m \mid E_{\widehat{D}_S}(\log\frac{1}{F(\cdot)}) - E_D(\log\frac{1}{F(\cdot)}) \mid \leq \frac{\epsilon}{4} \tag{3.5}$$

Now let $Q \in \mathcal{FPA}(C)_m$ be the output of our training algorithm, minimizing the empirical estimate of $\log\frac{1}{Q(\cdot)}$ over $Q \in \mathcal{FPA}(C)_m$. Then, by (3.5), we have that with probability at least $1-\delta$, both of the following hold:

$$E_D(\log\frac{1}{Q(\cdot)}) - E_{\widehat{D}_S}(\log\frac{1}{Q(\cdot)}) \leq \frac{\epsilon}{4} \text{ and} \tag{3.6}$$

$$E_{\widehat{D}_S}(\log\frac{1}{F^*(\cdot)}) - E_D(\log\frac{1}{F^*(\cdot)}) \leq \frac{\epsilon}{4} \tag{3.7}$$

Also by the definition of our training algorithm

$$E_{\widehat{D}_S}(\log\frac{1}{Q(\cdot)}) - E_{\widehat{D}_S}(\log\frac{1}{F^*(\cdot)}) \leq 0 \tag{3.8}$$

By summing the inequalities, (3.6), (3.7) and (3.8), we have the following with probability at least $1-\delta$:

$$E_D(\log\frac{1}{Q(\cdot)}) - E_D(\log\frac{1}{F^*(\cdot)}) \leq \frac{\epsilon}{2} \tag{3.9}$$

This implies:

$$d_{KL}(D,Q) - d_{KL}(D,F^*) \leq \frac{\epsilon}{2} \tag{3.10}$$

Then from (3.4) and (3.10) we have with probability at least $1-\delta$:

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

Thus any training algorithm that minimizes the empirical divergence over $\mathcal{FPA}(C)_m$ also approximately minimizes the actual divergence over $\mathcal{PA}(C)$ within accuracy $\epsilon$ with probability $1-\delta$, whenever $m \geq max\{\frac{4(n+1)}{\epsilon}+1, q(\frac{4}{\epsilon}, \log\frac{1}{\delta}, n, t)\}$.

**Proof of Theorem 3.1**

We need to show that (3.2) holds and that whenever $m \geq q(\frac{4}{\epsilon}, \log\frac{1}{\delta}, n, t)$ the inequality (3.5) holds with probability at least $1-\delta$, for some $q$ for which $q(\frac{4}{\epsilon}, \log\frac{1}{\delta}, n, t)$ is at least $\frac{4(n+1)}{\epsilon}+1$ and which fulfills the order bound promised in statement of the theorem.

We begin by defining $\mathcal{FPA}(C)_m$. We define a finite subset (grid points), denoted by $\mathcal{G}(\gamma,\theta)$, of $\mathcal{PA}(C)$ as the set of all members $P$ in $\mathcal{PA}(C)$, such that each of $\pi_P(i)$ and $M_P(i,j,z)$ is some power of $1-\gamma$, and is at least $\theta$. Let $\mathcal{G}'(\gamma,\theta)$ be the set of stochastic corrections of the members of $\mathcal{G}(\gamma,\theta)$. We then define $\mathcal{FPA}(C)_m$ as $\mathcal{G}'(\frac{1}{m}, \frac{1}{2tm})$.

and show that the empirical estimates converge to their true means *uniformly* for the class of random variables $\mathcal{F}(\mathcal{PA}(C)) = \{\log \frac{1}{P(\cdot)} : P \in \mathcal{PA}(C)\}$ for moderate sample size. The difficulty here is the fact that $\mathcal{F}(\mathcal{PA}(C))$ is unbounded in the sense that $\log \frac{1}{P(x)}$ diverges to infinity when $P(x)$ goes to zero. This fact prohibits the direct application of certain lemmas on the convergence of random variables, such as Hoeffding's inequality.

It turns out, however, that we do not need to show uniform convergence for the entire class. We let our training algorithm output a probabilistic automaton $Q$ for which $E_{\widehat{D}_S}(\log \frac{1}{Q(\cdot)})$ is minimized when $Q$ is restricted to be in a finite subclass $\mathcal{FPA}(C)_m$ of $\mathcal{PA}(C)$. (If there is more than one member in the finite class that achieves the minimum we make no further assumption about how our algorithm picks its hypothesis.) To describe $\mathcal{FPA}(C)_m$, first note that $\mathcal{PA}(C)$ is nothing but the subset of the 'parameter space' $[0,1]^I \times [0,1]^G$ satisfying the stochastic condition. For convenience we expand $\mathcal{PA}(C)$ by relaxing the equalities in the stochastic condition to inequalities:

$$\forall i \in S \sum_{j \in S, z \in \Sigma} M_P(i,j,z) \leq 1 \text{ and } \sum_{j \in I} \pi_P(j) \leq 1 \tag{3.1}$$

Note that any member $P$ of the parameter space satisfying this weaker condition can easily be converted to a probabilistic automaton $P'$ satisfying the strict stochastic condition which assigns at least as large a probability on every word:

$$\forall i,j \in S, z \in \Sigma \; M_{P'}(i,j,z) = \frac{M_P(i,j,z)}{\sum_{k \in S, z' \in \Sigma} M_P(i,k,z')} \text{ and } \pi_{P'}(i) = \frac{\pi_P(i)}{\sum_{j \in I} \pi_P(j)}$$

Let us say that $P'$ is the *stochastic correction* of $P$. Note that $P'$ has less divergence than $P$ with respect to *any* distribution[5].

$\mathcal{FPA}(C)_m$ is defined for each sample size $m$ and is the set of stochastic corrections of all 'bounded grid points' of $\mathcal{PA}(C)$, that is, those members of $\mathcal{PA}(C)$ in which all transition and initial probabilities are bounded from below by some decreasing function of $m$ and are powers of $(1-\gamma)$ where $\gamma$ also is a decreasing function of $m$.

Now, since $\mathcal{F}(\mathcal{FPA}(C)_m) = \{\log \frac{1}{F(\cdot)} : F \in \mathcal{FPA}(C)_m\}$ is a finite class of bounded random variables of moderate cardinality, we can show *fast* uniform convergence (of empirical estimates to true means) for them. We then show that for sufficiently large $m$, for each $P$ in $\mathcal{PA}(C)$, there exists $F \in \mathcal{FPA}(C)_m$ which is 'close to' $P$ everywhere in the domain:

$$\forall P \in \mathcal{PA}(C) \, \exists F \in \mathcal{FPA}(C)_m \, \forall x \in \Sigma^n \; \log \frac{1}{F(x)} - \log \frac{1}{P(x)} \leq \frac{2(n+1)}{m-1} \tag{3.2}$$

This immediately implies that for an arbitrary source distribution $D$ over $\Sigma^n$, we have:

$$\exists F \in \mathcal{FPA}(C)_m \; d_{KL}(D,F) - d_{KL}(D,P) \leq \frac{2(n+1)}{m-1} \tag{3.3}$$

So if we let $Opt$ be a member of $\mathcal{PA}(C)$ satisfying $d_{KL}(D,Opt) = min\{d_{KL}(D,P) : P \in \mathcal{PA}(C)\}$, then we have:

$$\exists F \in \mathcal{FPA}(C)_m \; d_{KL}(D,F) - d_{KL}(D,Opt) \leq \frac{2(n+1)}{m-1}$$

[5]We have also implicitly extended the notion of divergence for the generalized notion of probability distributions in which the total sum of probabilities over the domain may be less than one.

probability zero. 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 2.6: Single-String MLM Problem for $\mathcal{C}$**
*Input: A constraint $C \in \mathcal{C}$ and a string $w$ in $\Sigma^*$.*
*Output: A stochastic matrix $Q$ satisfying $C$ which assigns the maximum generation probability on $w$ among all such stochastic matrices, i.e.*

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

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 [Gil77]: 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 $\mathcal{C}$ is trainable with sample complexity $O((\frac{n}{\epsilon})^2 t \cdot \log^3 \frac{nt}{\epsilon} \cdot \log \frac{1}{\delta} \cdot \log^2 \log \frac{1}{\delta})$, 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 $n$, $\frac{1}{\epsilon}$, and $\log \frac{1}{\delta}$. As an easy corollary, the following bound on the sample complexity of the training problem for the complete constraints follows.

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

**Outline of the proof of Theorem 3.1**
Let an input constraint $C = \langle I, G \rangle \in \mathcal{C}$ be given, and let $t$ be its size, i.e. $t = \mid I \mid + \mid G \mid$. 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 $S$, its output $Q \in \mathcal{PA}(C)$ is likely to approximately minimize $d_{KL}(D, Q)$, where $D$ is the source distribution. Recall that $d_{KL}(D; Q) = E_D(\log \frac{1}{Q(\cdot)}) - H(D)$. 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 $E_D(\log \frac{1}{Q(\cdot)})$. Thus a natural attempt would be to find a $Q$ that minimizes $E_{\widehat{D}_S}(\log \frac{1}{Q(\cdot)})$

when $inf\{d_{KL}(D,P) : P \in \mathcal{PA}(C)\}$ is finite. Define a function $\xi_D : \mathcal{PA}(C) \rightarrow [0,1]$ 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 $[0,1]^G$, it attains a maximum at a particular PA, say $Opt'$:

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

Hence,

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

Noting that $d_{KL}(D,P) = E_D(\log \frac{1}{P(\cdot)}) - H(D)$, the inequality (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 $\mathcal{C}$**
*Input: A constraint $C \in \mathcal{C}$, a string length $n$ and a finite sample $S = \langle w_1, ..., w_m \rangle$ of strings from $\Sigma^n$.*
*Output: A probabilistic automaton $Q$ satisfying $C$ which assigns the maximum generation probability (or the maximum likelihood) on $S$ among all such probabilistic automata, i.e.*

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

Again note that $max\{\prod_{i=1}^{m} P(w_i) : P \in \mathcal{PA}(C)\}$ is well-defined because $\zeta : \mathcal{PA}(C) \rightarrow [0,1]$ 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 all other states have

the Kullback-Leibler divergence with respect to the source distribution also minimizes the expected code length of the data. It is also well-known that minimizing the Kullback-Leibler divergence with respect to the *empirical* distribution $\widehat{D}_S$ observed in a sample $S = \langle w_1, ..., w_m \rangle$ (Definition 2.1) corresponds to maximizing the likelihood of the sample, as demonstrated below. Minimizing $d_{KL}(\widehat{D}_S; Q)$ corresponds to minimizing $E_{\widehat{D}_S}(\log \frac{1}{Q(\cdot)})$, and the following always holds:

$$E_{\widehat{D}_S}(\log \frac{1}{Q(\cdot)}) = \frac{1}{m} \cdot \sum_{i=1}^{m} \log \frac{1}{Q(w_i)} = \frac{1}{m} \log \prod_{i=1}^{m} \frac{1}{Q(w_i)} \tag{2.4}$$

Thus, $d_{KL}(\widehat{D}_S; Q)$ is minimized when $\prod_{i=1}^{m} Q(w_i)$ 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 $\Sigma^n$, $S = \langle w_1, ..., w_m \rangle$ an arbitrary sample of $\Sigma^n$, and $\widehat{D}_S$ the empirical distribution of $S$. Then,*

$$E_{\widehat{D}_S}(\log \frac{1}{Q(\cdot)}) = inf\{E_{\widehat{D}_S}(\log \frac{1}{P(\cdot)}) : P \in \mathcal{P}\}$$

*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 $S$ 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 $q(\frac{1}{\epsilon}, \frac{1}{\delta}, n, t)$, if $A$, when given as input an arbitrary constraint $C \in \mathcal{C}$ of size $t$, a string length $n$, and a finite sample $S$ drawn independently at random from an arbitrary unknown distribution $D$ over $\Sigma^n$, is such that whenever the sample size $m$ exceeds $q(\frac{1}{\epsilon}, \frac{1}{\delta}, n, t)$, then provided that $min\{d_{KL}(D, P) : P \in \mathcal{PA}(C)\}$ 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 probabilistic. If there exists such an algorithm then we say that $\mathcal{C}$ is trainable with sample complexity $q(\frac{1}{\epsilon}, \frac{1}{\delta}, n, t)$. If there exists such an algorithm running 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,

is a pair $C = \langle I, G \rangle$ where $I$ is the *initial 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 $\mid I \mid$ for the number of states in $I$, $\mid G \mid$ for the number of transitions in $G$. We then define the *size* of a constraint $C$, written $\mid C \mid$, as $\mid C \mid = \mid I \mid + \mid G \mid$. Note that the size of $C$ corresponds to the number of probability parameters specified by $C$.

We say that there is a path in $C$ for a string $w_1 ... w_n \in \Sigma^n$, if there is a sequence of states $\sigma = \langle \sigma_0, ..., \sigma_{n+1} \rangle$ from $S^{n+1}$ such that $\sigma_0 \in I$ and for all $i$, $0 \leq i < n$, $(\sigma_i, \sigma_{i+1}, w_{i+1}) \in G$. We say that a probabilistic automaton $P$ *satisfies* a constraint $C = \langle I, G \rangle$, 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 $[0, 1]$. (In general, we use $Y^X$ to denote the class of functions from domain $X$ into range $Y$.) 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 $\frac{0}{0} = 1$.)*

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 $E_D(\log \frac{1}{Q(\cdot)}) - H(D)$ where $E_D(\log \frac{1}{Q(\cdot)})$ is the expectation according to $D$ of the random variable $\log \frac{1}{Q(\cdot)}$, and $H(D)$ is the 'entropy' of $D$, defined as $\sum_{x \in X} D(x) \log \frac{1}{D(x)}$. Recall that $\log \frac{1}{D(x)}$ is the code length for $x$ with respect to the ideal code for $D$, and $H(D)$ is the expected code length[4] of that code [Ham86] for the source distribution $D$. 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

[4] In Physics, it is customary to use the natural logarithm for the definition of entropy. We use the binary logarithm for the entropy and the Kullback-Leibler divergence as is done in information and coding theory.

*Note that for any $y$ not in $S$, we have $\widehat{D}_S(y) = 0$.*

The probabilistic automaton is formalized as a stochastic matrix $M$ together with an 'initial distribution' $\pi$ over the set of states. Intuitively, the probabilistic automaton is much like a non-deterministic finite state automaton except that the transitions take place with probabilities prescribed by $M$. (See Figure C.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 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 $n$.

**Definition 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[3], 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 $w = w_1 ... w_n \in \Sigma_P^*$, the generation probability assigned on it by $P = \langle S_P, \Sigma_P, \pi_P, M_P\rangle$ is computed as follows.*

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

*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 C.1 on the string $w = aab$ is calculated as follows:

$$\begin{aligned}
P(w) &= \pi_P(0) \cdot M_P(0, 0, a) \cdot M_P(0, 1, a) \cdot M_P(1, 1, b) \\
&\quad + \pi_P(0) \cdot M_P(0, 1, a) \cdot M_P(1, 1, a) \cdot M_P(1, 1, b) \\
&\quad + \pi_P(0) \cdot M_P(0, 1, a) \cdot M_P(1, 2, a) \cdot M_P(2, 0, b) \\
&\quad + \pi_P(1) \cdot M_P(1, 1, a) \cdot M_P(1, 1, a) \cdot M_P(1, 1, b) \\
&\quad + \pi_P(1) \cdot M_P(1, 1, a) \cdot M_P(1, 2, a) \cdot M_P(2, 0, b) \\
&= 0.5 \cdot 0.4 \cdot 0.6 \cdot 0.1 + 0.5 \cdot 0.6 \cdot 0.1 \cdot 0.1 + 0.5 \cdot 0.6 \cdot 0.8 \cdot 0.6 + \\
&\quad 0.5 \cdot 0.1 \cdot 0.1 \cdot 0.1 + 0.5 \cdot 0.1 \cdot 0.8 \cdot 0.6 \\
&= 0.012 + 0.003 + 0.072 + 0.0005 + 0.024 = 0.1115
\end{aligned}$$

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*

[3]The probabilistic automaton is often formulated as a probabilistic acceptor in the literature. Here we view PAs as generators. Thus the stochastic condition in this definition states that, for each state, the total probability of transitions out of that state sums to one, rather than the total probability for each state-letter pair as is the case for PAs as acceptors. Tzeng considers the incomparable problem of learning PAs as acceptors from queries [Tze89].

approximate (unless $\mathbf{P} = \mathbf{NP}$), and hence the class of 2-state complete constraints is not polynomially trainable (unless $\mathbf{RP} = \mathbf{NP}$).

The hardness result for the MLM problem for 2-state complete constraints 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 complete 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 $\mathbf{P} = \mathbf{NP}$. Note that it is a very strong non-approximability result, 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 [Gol78,Ang78,PW89]. 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 [Ang89]. 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 complete constraints, it is already non-trivial to formalize how a truth assignment 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 $\frac{1}{2^{|w|^{1-\alpha}}}$ 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 complete 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 HHMs. 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* $S$ of $\Sigma^n$ is a finite sequence of examples of $\Sigma^n$, $S = \langle w_1, ..., w_m \rangle$, where $m$ is the sample size. We abuse notation and write $x \in S$ to mean that $x$ appears in the sequence $S$. We let $\sharp(x, S)$ denote the number of occurrences of example $x$ in sample $S$. Using the above notation, we define the notion of the empirical distribution of a sample.

**Definition 2.1:** *Given a sample $S$ of size $m$ of $\Sigma^n$, the empirical distribution of $S$ over $\Sigma^n$, written $\widehat{D}_S$, is defined by:*

$$\forall x \in \Sigma^n \; \widehat{D}_S(x) = \frac{\sharp(x, S)}{m}$$

Using this model, we show a number of results: We show that an arbitrary class of constraints is trainable by exhibiting a training algorithm whose sample complexity 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 complete 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' [LRS83]. 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 $\mathcal{C}$ is equivalent to the polynomial time approximability of the 'maximum likelihood model' problem (MLM) for the same class $\mathcal{C}$ – the problem of setting the initial and transition probabilities in a given constraint in $\mathcal{C}$ 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 $\mathcal{C}$ is equivalent to the approximability of the MLM problem for $\mathcal{C}$ within a factor $1+\epsilon$ in random time polynomial in $\frac{1}{\epsilon}$ and the size $t$ of the input constraint. Furthermore, we show that this latter notion of approximability of the MLM problem for $\mathcal{C}$ 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 *complete* constraints, or the problem of finding a 2-state PA assigning the maximum likelihood on the input sample, is hard to

[2]The existence of an algorithm for training hidden Markov models which always outputs a *near local optimum* on a given sample is well-known ('Baum-Welch' algorithm [Bau72]) and is used extensively in practice. Note that in applications to speech recognition, the alphabet size is determined by how precise the acoustic signals are quantized. The alphabet size is often in the hundreds.

## 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 [LRS83]). 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 $q(...)$ if there exists an algorithm which trains every constraint in the class, and the sample size required for a given accuracy, 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 complete, 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 [Val84, BEHW89] and is inspired by the model of efficient unsupervised learning of Laird [Lai88]. It is also related to the models for learning languages from stochastic data in the limit proposed and studied by Angluin [Ang88]. 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 in [Hau89]. 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.' Other commonly used measures of distance between probability distributions are, for example, the $\chi^2$ distance, the variation distance, the quadratic distance [KS90], and the Hellinger distance. 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 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 in [Yam91].

[1] HMMs are similar to probabilistic automata, except that outputs in an HMM are associated with the states rather than the transitions, and thus the transitions are unlabeled state to state pairs.

## 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 – essentially 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 problem for 2-state PAs, which is of independent interest.

# On the Computational Complexity of Approximating Distributions by Probabilistic Automata*

Naoki Abe
Manfred K. Warmuth

UCSC-CRL-90-63

December 28, 1990

Baskin Center for
Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

*Supported by the Office of Naval Research, contract number N0014-86-K-0454. Part of this work was done after the first author began employment by C & C Information Technology Research Labs, NEC, 4-1-1 Miyazaki Miyamae-ku, Kawasaki, 216 Japan, and the second author was employed by IIAS-SIS Fujitsu Limited, Numazu, Japan. Email addresses: abe@ibl.cl.nec.co.jp and manfred@cis.ucsc.edu.