Learning variable memory length Markov chains from noisy output

CS690/691 progress report

Miklós Csűrös
advisor: Dana Angluin

November 13, 1996

1 Introduction

Complex data sequences such as DNA or speech are often modeled by Hidden Markov Models [8] or Probabilistic Finite Automata. Both models have severe theoretical drawbacks. Abe and Warmuth [1] proved that Hidden Markov Models are not learnable in time polynomial in the alphabet size unless RP=NP. Kearns *et al.* [5] proved that nor are Probabilistic Finite Automata learnable in the PAC-sense [2] unless noisy parity functions are learnable. Ron, Singer and Tishby [9] proposed the use of Probabilistic Finite Suffix Automata (PFSA), or variable memory length Markov chains, which form a subclass of Probabilistic Finite Automata. They proved that PFSA are efficiently learnable and described a polynomial-time learning algorithm.

In this report we consider the case when the output of the target machine is seen only through a noisy channel. Among others, Lugosi [7] considered learning from noisy data and showed that consistent learning is possible in case of the binary symmetric channel even if the noise rate is not known. In particular, he proved that the use of a nearest neighbor or a histogram classification method is consistent, that is, the error rate converges to its minimal value (Bayes-error).

We will proceed as follows. In Section 2 we will present the framework and algorithm of Ron *et al.* for learning PFSA. In Section 3 we will modify the algorithm for learning from noisy data with exact knowledge of the noise structure. In Section 4 we will show that the same algorithm also works if we have a very good estimate of the noise.

2 Probabilistic Finite Suffix Automata

This section is mainly a summary of the results of Ron *et al.* [9].

2.1 How good is a hypothesis?

A data sequence is defined as a string over a finite alphabet Σ. For DNA sequencing, the alphabet consists of the four bases, a set of sounds may serve as alphabet in a pronunciation model and so on. It is supposed that the string is generated by a machine M belonging to some class $\mathcal{C}$ of generator machines. The purpose of learning is to find a hypothesis machine $H \in \mathcal{C}$ that approximates M well[1]. The learning algorithm makes its hypothesis after seeing a sufficiently large sample of strings generated by M.

In this paper we consider probabilistic machines without a "stop state", that is, machines that generate an infinite string at each run. In order to pose practical questions, we will restrict our attention to finite length prefixes of the machine output. We will use prefix and suffix functions defined the usual way.

$$\begin{aligned} \mathit{prefix}^{\ell}(s) &= q \in \Sigma^{\ell} \text{ iff } \exists s' \colon s = q \cdot s' \quad (s, s' \in \Sigma^{\infty} \text{ or } s, s' \in \Sigma^{*}) \\ \mathit{suffix}^{\ell}(s) &= q \in \Sigma^{\ell} \text{ iff } \exists s' \colon s = s' \cdot q \quad (s, s' \in \Sigma^{*}), \end{aligned}$$

where ℓ is a nonnegative integer and '$\cdot$' denotes concatenation.

Let $\mu_M(s)$ denote the probability that machine M generates a string with prefix s. The goodness of the hypothesis H is captured by some distance between μ_M and μ_H. The chosen distance measure is the relative entropy, also called Kullback-Leibler divergence. (It is not a distance in the mathematical sense because neither symmetry nor the triangle equality are necessarily satisfied.) The Kullback-Leibler divergence between μ_M and μ_H on length ℓ is defined as

$$D_{\ell}[\,\mu_M \,\|\, \mu_H\,] \stackrel{\text{def}}{=} \sum_{s \in \Sigma^{\ell}} \mu_M(s) \ln \frac{\mu_M(s)}{\mu_H(s)},$$

where ln denotes natural logarithm[2].

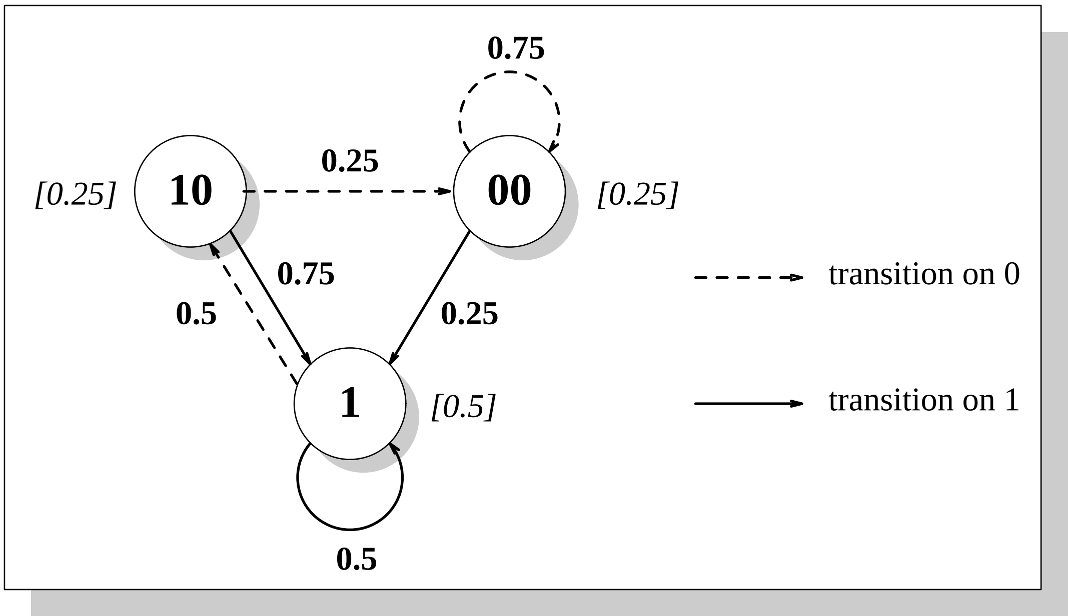
Definition 2.1 *The machine $H \in \mathcal{C}$ is an ϵ-good hypothesis for $M \in \mathcal{C}$ iff for all $\ell > 0$, the relative entropy $D_{\ell}[\,\mu_M \,\|\, \mu_H\,] < \ell\epsilon$, ie.,*

$$\frac{1}{\ell} D_{\ell}[\,\mu_M \,\|\, \mu_H\,] = \frac{1}{\ell} \sum_{s \in \Sigma^{\ell}} \mu_M(s) \ln \frac{\mu_M(s)}{\mu_H(s)} < \epsilon.$$

REMARK. If the class $\mathcal{C}$ is the class of Markov models of order less than ℓ, then the relative entropy is proportional to ℓ. Thus, it is necessary to divide it by ℓ to obtain a distance measure independent of the prefix length considered.

[1] In a more general framework, the hypothesis machine could be taken from a hypothesis class $\mathcal{H}$, not necessarily the same as the class $\mathcal{C}$. Here they are the same.

[2] The relative entropy $D[\,p \,\|\, q\,]$ between two probability mass functions is traditionally defined as $\sum_s p(s) \log \frac{p(s)}{q(s)}$ where log is taken base 2. It has an interesting coding theoretical interpretation. If the strings s have the distribution defined by p but we use the optimal code for the distribution defined by q, then $D[\,p \,\|\, q\,]$ is asymptotically the expected number of extra bits used compared to the optimal encoding. [9] uses natural logarithm instead of base 2, so we follow their definition.



The states of this binary PFA are $Q = \{1, 10, 00\}$, arrows represent the transition function with the transition probabilities. The initial probabilities are in brackets.

Figure 1: A Probabilistic Finite Automata working on a binary alphabet.

The relative entropy gives a bound on the L_1 distance, as

$$D[\,p \,\|\, q\,] \geq \frac{1}{2}\|p - q\|_1^2.$$

(See Cover and Thomas [3].) Since the L_1 norm bounds the $L_2, L_3, \ldots,$ and L_∞ norms from above, a good hypothesis remains a good one for other distance functions, as well. □

2.2 Model classes of Probabilistic Finite Automata

An example of a Probabilistic Finite Automaton (PFA) is shown in Figure 1.

Definition 2.2 *The model class of Probabilistic Finite Automata is the set of five-tuples* $\langle Q, \Sigma, \tau, \gamma, \pi \rangle$*:*

- Q*: a finite set of states.*
- $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$*: finite alphabet.*
- $\gamma : Q \times \Sigma \mapsto [0, 1]$*: next symbol probability function,* $\forall q \in Q$: $\sum_{\sigma \in \Sigma} \gamma(q, \sigma) = 1$.
- $\tau : Q \times \Sigma \mapsto Q$*: transition function.*
- $\pi : Q \mapsto [0, 1]$*: initial probability distribution,* $\sum_{q \in Q} \pi(q) = 1$.

Definition 2.3 *The PFA $\langle Q, \Sigma, \tau, \gamma, \pi \rangle$ generates the string $s = s_1 s_2 \cdots s_\ell$ (where $s_i \in \Sigma$) with the following probability:*

$$p_s = \mu_M(s) = \sum_{q_0 \in Q} \pi(q_0) \prod_{i=1}^{\ell} \gamma(q_{i-1}, s_i),$$

with $q_i = \tau(q_{i-1}, s_i)$ for $i = 1, \ldots, \ell$.

Recall that by the result of Kearns *et al.* [5], it is unlikely to have an efficient learning algorithm for the model class of PFA. We will therefore impose further restrictions to make efficient learning possible. If we observe the output of a PFA, the only uncertainty about the internal state transitions is the value of the starting state q_0. If we learn it (for example, $\pi(q) = 1$ for some $q \in Q$), the whole sequence of state transitions is computable using the transition function τ. In general, however, the uncertainty about the starting state will not disappear. In other words, the automaton has unbounded memory, since the starting state is always remembered. In the class of Probabilistic Finite Suffix Automata, τ cannot be arbitrary. The machine has a finite memory length bounded by a nonnegative integer ℓ. That is, if in the generation of the string $s = s_1 s_2 \cdots$ the machine goes through the states $q_0, q_1, \ldots$, then for any $k \geq \ell$, the state q_k is uniquely determined by $s_{k-\ell+1}, s_{k-\ell+2}, \ldots, s_k$, the last ℓ symbols generated so far. The following definition captures this feature.

Definition 2.4 *An $\ell - PFSA$ is a PFA $\langle Q, \Sigma, \tau, \gamma, \pi \rangle$ with the additional requirement that the states can be labeled by strings of $\Sigma^{\leq \ell}$ so that for any $q \in Q$ and $\sigma \in \Sigma$, the state $\tau(q, \sigma)$ is labeled by a suffix of $q \cdot \sigma$. ('$\cdot$' denotes concatenation.)*

Figure 1 shows an example of a 2-PFSA. With this definition, the run of an $\ell - PFSA$ can be simulated by a Markov chain of order ℓ. The states of the chain are the symbols of the alphabet, and the probability of moving to a particular next state depends on the last ℓ states visited. The power of PFSA lies in the fact that the labeling of the states may involve only a small subset of Σ^ℓ in which case we do not have to deal with an exponential number of states. In many cases, the generation of the next symbol depends on fewer than ℓ previously generated symbols. This justifies the term "variable memory length Markov chain".

The PFSA learning algorithm of Ron, Singer and Tishby [9] uses a sample of prefixes of length $\ell + 1$ that are independently generated by an ℓ-PFSA $M = \langle Q, \Sigma, \tau, \gamma, \pi \rangle$. The parameters of the algorithm are $|\Sigma|$, ℓ, n, ϵ and δ, where n is a bound on $|Q|$ and $0 < \delta < 1$ is a confidence parameter. The algorithm outputs a hypothesis machine H that is ϵ-good with probability at least $1 - \delta$ and has fewer than $|Q||\Sigma|$ states. The running time is polynomial in ℓ, n, $|\Sigma|$, $1/\epsilon$, and $1/\delta$. The outline of the algorithm is shown in Figure 3:

(I) Estimate from the sample:
— $p'(s)$: the probability of string $s \in \Sigma^{\leq \ell+1}$
— $p'(\sigma|s) = p'(s \cdot \sigma)/p'(s)$: the next symbol probability.

(II.1) Start with state for empty string.

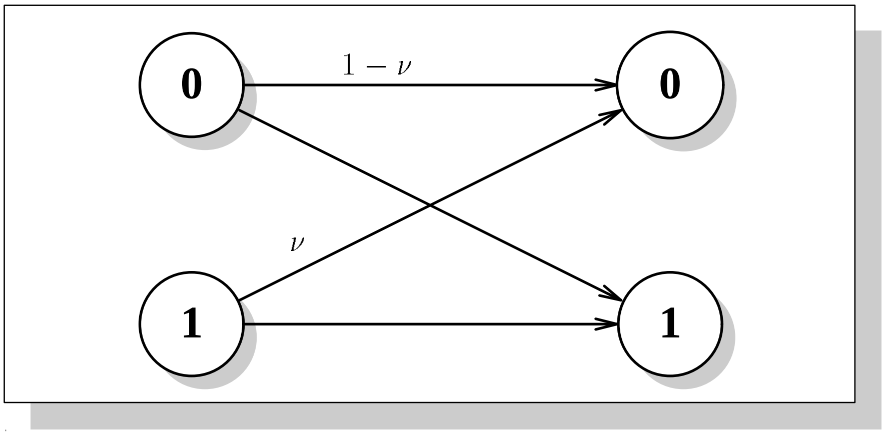


Figure 2: Binary symmetric channel.

(II.2) Repeat: add s if
$p'(\sigma|s) \gg p'(\sigma|suffix(s))$ for some σ, where $suffix(s)$ is the longest suffix of s that already has a corresponding state.

(II.3) Rescale probabilities.

The proof of correctness and running time of the algorithm is established by two theorems. The first theorem says that a random sample of polynomial size is typical with probability at least $1-\delta$. A typical sample is one for which (1) the relative frequency $p'(s)$ of the string s is close to the real probability p_s for any s that is a label of a state and (2) the relative next symbol frequencies $p'(\sigma|s)$ are close to the real probabilities $p(\sigma|s)$ for all s that often[3] appear in the sample. The second theorem states that if the algorithm is given a typical sample, then it outputs a hypothesis machine H with the features described above.

2.3 Noise model

In our approach we consider the case when the output of the machine passes through a noisy channel before being observed. Thus a learning algorithm has access only to a noisy sample. In speech recognition, for example, this type of model can reflect mistakes, differences in dialects or accents [same machine, different noise], etc. We can then use the hypothesis machine to remove the noise or to recognize patterns by finding the most likely state transition given the noisy output.

In the most general case, the noisy channel performs a transformation of the clean string probability measure μ_M to a noisy string probability measure μ^*_M. Here we consider only the case of a memoryless time-invariant channel. In particular, we consider a simple noise model: a memoryless symmetric channel. Each character of the output is altered by the same

[3]Throughout this part, qualitative notions as “often”, “close”, “much larger”: $\gg$ are precisely defined by factors and inequalities using the input parameters of the algorithm. For sake of simplicity, we omit them here.

probability $\nu \in [0,1]$. If a symbol gets corrupted, then it is replaced by a different symbol of the alphabet chosen with uniform probability. Figure 2 shows the binary symmetric channel. Each bit of the output of the machine is complemented with probability ν.

Notation — summary

- $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ finite alphabet of size $m \geq 2$
- ℓ suffix length
- p_s the probability of getting clean string $s \in \Sigma^*$ [*clean string probability*]
- p_s^* the probability of getting noisy string s [*noisy string probability*]
- $p(A) \stackrel{\text{def}}{=} \sum_{s \in A} p_s$, $p(s) = p_s$
- $p^*(A) \stackrel{\text{def}}{=} \sum_{s \in A} p_s^*$, $p^*(s) = p_s^*$
- $p(\sigma|s) = p(s \cdot \sigma)/p(s)$ next symbol probability
- ν (uniform) noise rate

3 Learning PFSA when noise is known

3.1 Inverse noise transformation

The main result of this section are the formulas expressing the relationship between the noisy and clean string probabilities.

Theorem 3.1 *Set*

$$\theta \stackrel{\text{def}}{=} \frac{\nu}{1 - \frac{m}{m-1}\nu} \tag{1}$$

Then the clean and noisy string probabilities can be expressed in terms of each other as

$$p_s^* = \sum_{t \in \Sigma^\ell} \left(\frac{\nu}{m-1}\right)^{d(s,t)} (1-\nu)^{\ell - d(s,t)} p_t \tag{2}$$

$$p_s = \sum_{t \in \Sigma^\ell} \left(-\frac{\theta}{m-1}\right)^{d(s,t)} (1+\theta)^{\ell - d(s,t)} p_t^*, \tag{3}$$

where $d(s,r)$ is the number of different letters in s and r (Hamming-distance between s and r).

PROOF. Equation (2) is clear from the probability model.

Let v_k be the (row) vector of clean string probabilities (in lexicographic order of the strings) for length k. Let v_k^* be defined similarly for the noisy strings. We will derive formulas for the noise transformation matrix $\mathbf{N}_k$ and the inverse noise transformation matrix $\mathbf{N}_k^{-1}$:

$$v_k^* = v_k \cdot \mathbf{N}_k \quad \text{and} \quad v_k = v_k^* \cdot \mathbf{N}_k^{-1}.$$

Notice that these matrices are of size $m^k \times m^k$. For $k = 1$, we have

$$(p_{\sigma_1}^* \cdots p_{\sigma_m}^*) = (p_{\sigma_1} \cdots p_{\sigma_m})\mathbf{N}_1 = (p_{\sigma_1} \cdots p_{\sigma_m}) \begin{pmatrix} 1-\nu & \cdots & \nu/(m-1) \\ \vdots & \ddots & \vdots \\ \nu/(m-1) & \cdots & 1-\nu \end{pmatrix},$$

that is, the entries of $\mathbf{N}_1$ are $(1-\nu)$ along the diagonal and $\nu/(m-1)$ everywhere else. By inverting the matrix we get

$$(p_{\sigma_1} \cdots p_{\sigma_m}) = (p_{\sigma_1}^* \cdots p_{\sigma_m}^*)\mathbf{N}_1^{-1} = (p_{\sigma_1}^* \cdots p_{\sigma_m}^*) \begin{pmatrix} \frac{1-\nu/(m-1)}{1-\frac{m}{m-1}\nu} & \cdots & \frac{-\nu/(m-1)}{1-\frac{m}{m-1}\nu} \\ \vdots & \ddots & \vdots \\ \frac{-\nu/(m-1)}{1-\frac{m}{m-1}\nu} & \cdots & \frac{1-\nu/(m-1)}{1-\frac{m}{m-1}\nu} \end{pmatrix},$$

that is, the entries of $\mathbf{N}_k^{-1}$ are $(1+\theta)$ along the diagonal and $-\theta/(m-1)$ everywhere else.

For $k > 1$, the transformation matrices can be obtained by using the recursion

$$\mathbf{N}_{k+1} = \begin{pmatrix} (1-\nu)\mathbf{N}_k & \cdots & \frac{\nu}{m-1}\mathbf{N}_k \\ \vdots & \ddots & \vdots \\ \frac{\nu}{m-1}\mathbf{N}_k & \cdots & (1-\nu)\mathbf{N}_k \end{pmatrix}$$

that gives the recursive form for $\mathbf{N}_{k+1}^{-1}$:

$$\mathbf{N}_{k+1}^{-1} = \begin{pmatrix} (1+\theta)\mathbf{N}_k^{-1} & \cdots & -\frac{\theta}{m-1}\mathbf{N}_k^{-1} \\ \vdots & \ddots & \vdots \\ -\frac{\theta}{m-1}\mathbf{N}_k^{-1} & \cdots & (1+\theta)\mathbf{N}_k^{-1} \end{pmatrix}$$

Therefore, the entry in row s and column t of $\mathbf{N}_\ell^{-1}$ is $\left(-\frac{\theta}{m-1}\right)^{d(s,t)} (1+\theta)^{\ell-d(s,t)}$. ■

Definition 3.1 *Define the Hamming-ball of radius d around the string $s \in \Sigma^\ell$ as*

$$B_s(d) \stackrel{\text{def}}{=} \{r : r \in \Sigma^\ell, d(s,r) = d\}.$$

Then instead of Equations (2) and (3) we can use the equivalent forms

$$p_s^* = \sum_{d=0}^{\ell} \left(\frac{\nu}{m-1}\right)^d (1-\nu)^{\ell-d} p(B_s(d)) \tag{4}$$

$$p_s = \sum_{d=0}^{\ell} \left(-\frac{\theta}{m-1}\right)^d (1+\theta)^{\ell-d} p^*(B_s(d)), \tag{5}$$

for computing upper bounds on the errors. Equation (3) is used in the algorithms.

Remark. One can verify that $(1+\theta)$ decreases as the alphabet size m increases if ν is fixed. We have

$$\frac{1}{1-\nu} < 1+\theta \leq \frac{1-\nu}{1-2\nu},$$

where the upper bound is reached in the case of a binary alphabet. This is important in the following results because the lower bounds on sample size for effective learning will grow with $(1+\theta)$ and thus the worst case will be that of the binary alphabet. The lower bound on $(1+\theta)$ is the limit as $m \to \infty$.

Furthermore, θ increases with ν, it is positive for $\nu < 1-1/m$ and it is negative between $1-1/m$ and 1. We will consider usually only $\nu < 1-1/m$. □

3.2 Inverse noise transformation with empirical probabilities

Suppose now that we use Equations (3) and (5) with estimated probabilities. How does the estimation error propagate?

The estimated noisy string probabilities obtained from the sample are denoted by $\hat{p}^*(\cdot)$.

$$\hat{p}_s \stackrel{\text{def}}{=} \sum_{t\in\Sigma^\ell} \left(-\frac{\theta}{m-1}\right)^{d(s,t)} (1+\theta)^{\ell-d(s,t)} \hat{p}^*_t \tag{6}$$

$$= \sum_{d=0}^{\ell} \left(-\frac{\theta}{m-1}\right)^{d} (1+\theta)^{\ell-d} \hat{p}^*(B_s(d)). \tag{7}$$

Lemma 3.1 *If, given a string s of length ℓ, for the estimated noisy string probabilities we have*

$$\forall\, d\ \ |\hat{p}^*(B_s(d)) - p^*(B_s(d))| \leq \epsilon^*, \tag{8}$$

then for the clean string probabilities estimated using Equation (6)

$$|\hat{p}_s - p_s| \leq \epsilon^*(1+\theta)^{\ell+1},$$

for $\nu < 1-1/m$.

The sample size required to obtain the estimation error of Equation (8) is set by the following lemma.

Lemma 3.2 *If $\hat{p}^*(\cdot)$ denotes relative frequency in a sample of size*

$$N \geq \frac{2}{\epsilon^{*2}} \ln \frac{2(\ell+1)}{\delta}, \tag{9}$$

then for arbitrary s,

$$\mathbb{P}\{\,\forall\, d\ \ |\hat{p}^*(B_s(d)) - p^*(B_s(d))| \leq \epsilon^*\,\} \geq 1-\delta. \tag{10}$$

With the aid of these lemmas, we can prove the following theorem.

Theorem 3.2 *If the clean string probabilities are estimated using Equation (6), where the noisy string probabilities are derived from a sample of size*

$$N \geq (1+\theta)^{2(\ell+1)} \frac{2}{\epsilon^2} \ln \frac{2(\ell+1)}{\delta}, \tag{11}$$

then

$$\mathbb{P}\{\, |\hat{p}_s - p_s| \leq \epsilon \,\} \geq 1 - \delta \tag{12}$$

for $\nu < 1 - 1/m$.

Proof of Theorem 3.2. Use Lemma 3.1 and Lemma 3.2 with $\epsilon = \epsilon^*(1+\theta)^{\ell+1}$. ■

Proof of Lemma 3.1. If $\nu < 1 - 1/m$, then $\theta > 0$. From Equations (5) and (7) we have

$$\begin{aligned}
|\hat{p}_s - p_s| &= \left| \sum_{d=0}^{\ell} \left(-\frac{\theta}{m-1} \right)^d (1+\theta)^{\ell-d} \Big(\hat{p}^*(B_s(d)) - p^*(B_s(d)) \Big) \right| \\
&\leq \epsilon^* \sum_{d=0}^{\ell} \left(\frac{\theta}{m-1} \right)^d (1+\theta)^{\ell-d} = \epsilon^* (1+\theta)^{\ell} \sum_{d=0}^{\ell} \left(\frac{\theta/(m-1)}{1+\theta} \right)^d \\
&< \epsilon^* (1+\theta)^{\ell} \frac{1}{1 - \frac{\theta/(m-1)}{1+\theta}} \\
&\leq \epsilon^* (1+\theta)^{\ell+1}
\end{aligned}$$

■

Remark. Note that we could have obtained an upper bound of the form $\epsilon^*(1+\theta)^{\ell}(\ell+1)$ but $(\ell+1)$ is expected to be larger than $(1+\theta)$ for small values of ν. (cf. Figure 4.) □

Proof of Lemma 3.2. For sample size N satisfying Equation (9), we have

$$\begin{aligned}
&\mathbb{P}\{\, \forall\, d \;\; |\hat{p}^*(B_s(d)) - p^*(B_s(d))| \leq \epsilon^* \,\} \\
&= 1 - \mathbb{P} \bigcup_{d=0}^{\ell} \{ |\hat{p}^*(B_s(d)) - p^*(B_s(d))| > \epsilon^* \} \\
&\geq 1 - \sum_{d=0}^{\ell} \mathbb{P}\{\, |\hat{p}^*(B_s(d)) - p^*(B_s(d))| > \epsilon^* \,\} \\
&\geq 1 - (\ell+1) \frac{\delta}{\ell+1} = 1 - \delta,
\end{aligned}$$

by Hoeffding's inequality[4] [4]. ■

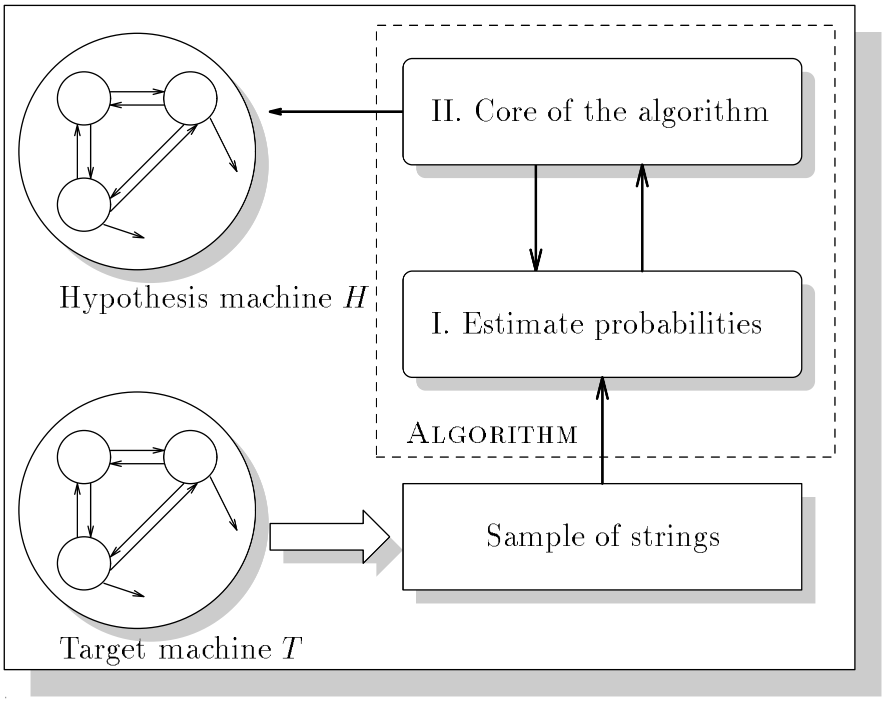


Figure 3: PFSA learning process

3.3 Algorithm when noise is known

We will use the algorithm of Ron, Singer and Tishby [9]. The main structure of the learning process is sketched in Figure 3. From a sample of strings generated by the target automaton T, the algorithm estimates the string probabilities and outputs a hypothesis machine H. There is no noise introduced in the original algorithm, so the estimation of probabilities is simple: it is enough to take the relative frequencies. The original paper defined the notion of a "typical" sample. They prove that if the algorithm is given a typical sample, then the output is an ϵ-good hypothesis with probability at least $1 - \delta$.

Our learning process will have the same outline. We change the method of estimation of probabilities from the sample, but the core remains the same. The clean string probabilities that would be derived from a clean sample in the original algorithm will be estimated using Equation (6). The other difference is that we only compute estimation if needed, whereas in the original algorithm, the estimates can be precomputed before starting the core part. Since we only need to consider the noisy strings that are present in the sample, one clean string probability estimation takes linear time in the sample size and ℓ, the length of the longest suffix considered. This results in a new factor in the running time of the algorithm. We shall tighten the definition of a typical sample so that we can have an algorithm that works on a noisy sample. Since we require a somewhat better accuracy in the clean string probability estimates than the original definition, and use the same core algorithm, we will

[4] We use the corollary that $\mathbb{P}\{\,|\hat{p}^*(B_s(d)) - p^*(B_s(d))| > \epsilon^*\,\} \leq 2\exp(-2n{\epsilon^*}^2)$.

still output an ϵ-good hypothesis with probability at least $1-\delta$.

A better accuracy of the estimation is needed because a new problem arises here when we estimate the clean next symbol probabilities. If we have access to a sample of uncorrupted strings, then the next symbol probabilities are obtained easily directly from the sample. For $\hat{p}(\sigma|s)$, we count all the occurrences of σ following s. In the case of a noisy sample though, this is not feasible. We define the estimated clean next symbol probability as

$$\hat{p}(\sigma|s) \stackrel{\text{def}}{=} \frac{\hat{p}(s\sigma)}{\hat{p}(s)}. \tag{13}$$

The notion of a typical sample has to be slightly modified in order to ensure that the algorithm still works. The modified definition of a typical sample suited for our needs and notation follows.

Definition 3.2 (cf. Definition 6.1 in [9]) *A sample generated according to PFSA M is $(\epsilon_0, \epsilon_1, \epsilon_2, \gamma_{\min})$-typical if for the probabilities $\hat{p}_s$ estimated using the sample, the following two properties hold:*

1. *For all $s \in \Sigma^{\leq \ell+1}$: $|\hat{p}_s - p_s| \leq \epsilon_1\epsilon_0$.*
2. *For all $s \in \Sigma^{\leq \ell}$: if $\hat{p}_s \geq (1-\epsilon_1)\epsilon_0$ then for every $\sigma \in \Sigma$,*

$$|\hat{p}(\sigma|s) - p(\sigma|s)| \leq \epsilon_2\gamma_{\min}. \tag{14}$$

Remark. The parameters $\epsilon_0, \epsilon_1, \epsilon_2$ and $\gamma_{\min}$ are used in the core of the algorithm. They are relatively simple functions of $\ell, n, |\Sigma|, \epsilon$, and δ.

This definition differs in property 1 from that of the article. In the article, it is required to hold only for strings in Q, the set of states of the PFSA. We need more here so that property 2 is implied by property 1 modulo certain assumptions. □

The theorem of this section is

Theorem 3.3 *A sample of size*

$$N \geq \frac{2}{\epsilon_1^2\epsilon_0^2}(1+\theta)^{2(\ell+2)}(\ell+2)\ln\frac{2|\Sigma|}{\delta} \tag{15}$$

is typical with probability $\geq 1-\delta$, where

$$\theta = \frac{\nu}{1 - \frac{|\Sigma|}{|\Sigma|-1}\nu} \quad \textit{and} \quad \nu < 1 - \frac{1}{|\Sigma|}.$$

Proof of Theorem 3.3. The proof of property 1 of Definition 3.2 is similar to the proof in the article. The only difference is that we have to use Theorem 3.2 instead of a simple application of Hoeffding's inequality. Since $\left|\Sigma^{\leq \ell+1}\right| < m^{(\ell+2)}$, if the inequality holds for any particular string with probability $\geq 1 - \delta/m^{(\ell+2)}$ then it holds for all strings in $\Sigma^{\leq \ell+1}$ with

	Constraint	**Reference**
C1	$\epsilon_0 \le \epsilon/(2n\log(1/\gamma_{\min}))$	Page 42, below Equation (8)
C2	$\epsilon_3 \le \epsilon/2$	Page 42, below Equation (8)
C3	$8n\epsilon_1\epsilon_0 \le \epsilon_2\gamma_{\min}$	Page 44, below Equation (11), obsolete
C4	$\gamma_{\min} \le \epsilon_2/\|\Sigma\|$	Page 45, above Equation (15)
C5	$\epsilon_2 \le \epsilon_3/6$	Page 45, above Equation (18)
C6	$\epsilon_3 \le 1/2$	Page 45, above Equation (18)
C7	$\epsilon_2 \le \epsilon_3/21$	Page 46, above Equation (20)
C8	$\epsilon_2 \le \epsilon_3/(24\ell)$	Page 46, below Equation (26)
C9	$\epsilon_1 \le \epsilon_2\gamma_{\min}/4$	Equation (16), replaces C3

(The page and equation numbers refer to the article, except in the last row where it is our new constraint.)

Table 1: Constraints on the parameters

probability at least $1-\delta$. For this step just plug $\ell \leftarrow \ell+1, \epsilon \leftarrow \epsilon_1\epsilon_0$ and $\delta \leftarrow \delta/(m^{(\ell+2)})$ into Equation (11) and notice that

$$\ln\frac{2(\ell+2)m^{(\ell+2)}}{\delta} < (\ell+2)\ln\frac{2m}{\delta}.$$

We show now that property 2 is implied by property 1. This step is different from the method of the article. Consider the following:

$$\begin{aligned}
|\hat{p}(\sigma|s) - p(\sigma|s)| &= \left|\frac{\hat{p}(s\sigma)}{\hat{p}(s)} - \frac{p(s\sigma)}{p(s)}\right| \\
&= \frac{1}{\hat{p}(s)p(s)}\Big|\hat{p}(s\sigma)p(s) - p(s\sigma)\hat{p}(s)\Big| \\
&= \frac{1}{\hat{p}(s)p(s)}\Big|\Big(\hat{p}(s\sigma)p(s) - p(s\sigma)p(s)\Big) + \Big(p(s\sigma)p(s) - p(s\sigma)\hat{p}(s)\Big)\Big| \\
&\le \frac{1}{\hat{p}(s)}\Big(\Big|\hat{p}(s\sigma) - p(s\sigma)\Big| + p(\sigma|s)\Big|\hat{p}(s) - p(s)\Big|\Big) \\
&\le \frac{1}{(1-\epsilon_1)\epsilon_0}\epsilon_1\epsilon_0(1 + p(\sigma|s)) \le \frac{2\epsilon_1}{1-\epsilon_1}.
\end{aligned}$$

If $\epsilon_1 \le 1/2$, then the right-hand side is not greater than $4\epsilon_1$. Therefore we introduce a new constraint on the parameters:

$$4\epsilon_1 \le \epsilon_2\gamma_{\min}. \tag{16}$$

This replaces the condition $8n\epsilon_1\epsilon_0 \le \epsilon_2\gamma_{\min}$ set in the article in order to prove our second property. See the Table of Constraints (1) for detailed reference. ■

The rest of the results in the article are applicable here.

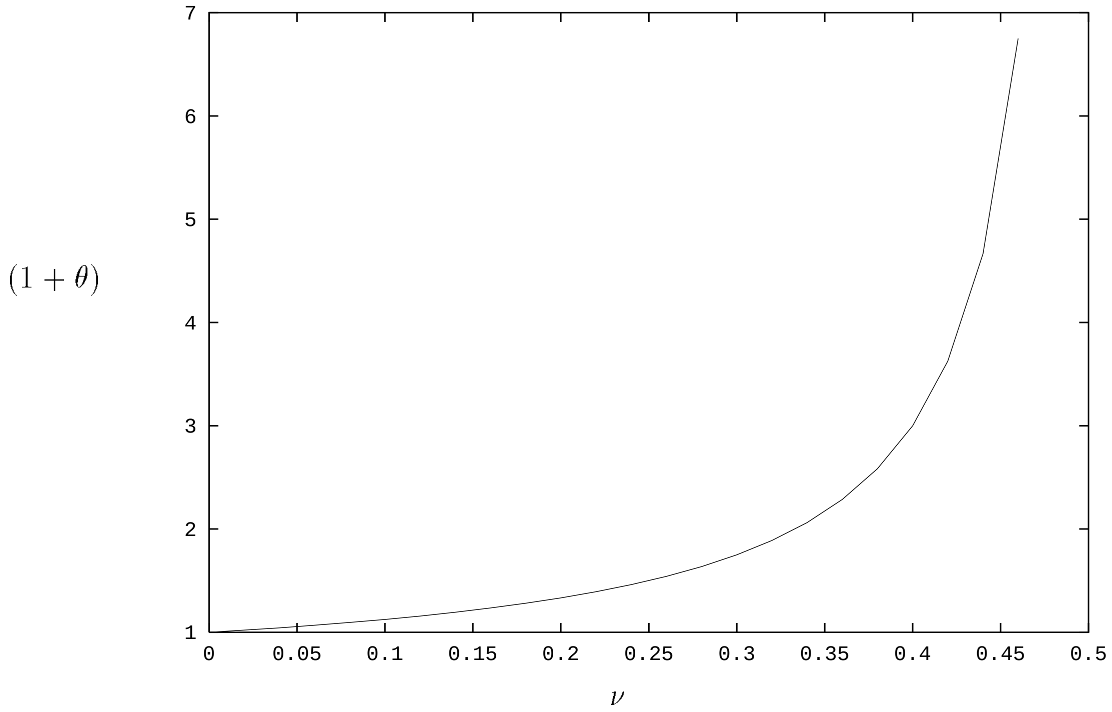


Figure 4: The values of $(1+\theta)$ (binary alphabet)

3.4 Running time and sample size

The various constraints set by the article and by the proof of Theorem 3.3 are summarized in Table 1. Combining these values, we obtain the following for the parameters in terms of the input values $\ell, n, \Sigma, \epsilon$, and δ:

	Parameter	**Constraints**
C10	$\epsilon_3 \le \frac{\epsilon}{2}$	C2, C6
C11	$\epsilon_2 \le \frac{\epsilon}{48\ell}$	C5, C7, C8, C10
C12	$\gamma_{\min} \le \frac{\epsilon}{48\ell\|\Sigma\|}$	C4, C11
C13	$\epsilon_0 \le \frac{\epsilon}{2n\log(48\ell\|\Sigma\|/\epsilon)}$	C1, C12
C14	$\epsilon_1 \le \frac{\epsilon^2}{9216\ell^2\|\Sigma\|}$	C9, C11, C12

Plugging these values into Equation (15) we get

$$N = \Omega\left(\Lambda n^2 \ell^5 |\Sigma|^2 \epsilon^{-6} \log^2 \frac{\ell|\Sigma|}{\epsilon} \log \frac{|\Sigma|}{\delta}\right),$$

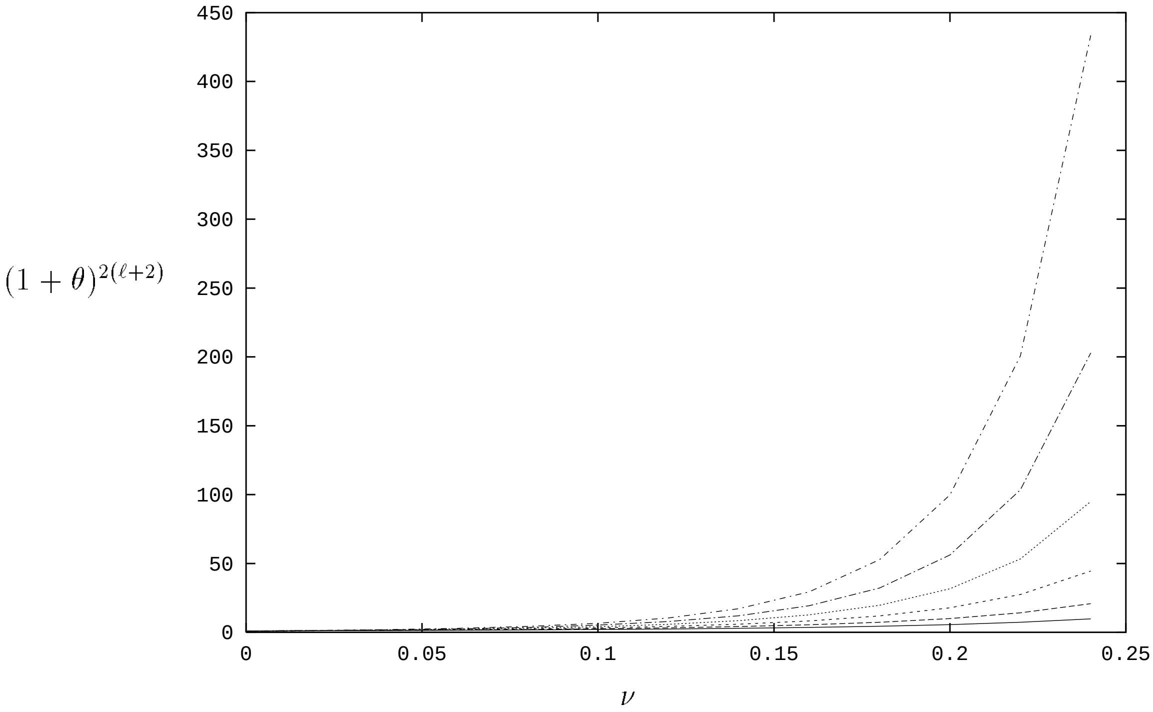


Figure 5: The exponential factor Λ for memory lengths $\ell = 1, \ldots, 6$ (binary alphabet).

for the sample size, where Λ hides the exponential term

$$\Lambda = (1+\theta)^{2(\ell+2)} = \left(\frac{1 - \nu/(|\Sigma|-1)}{1 - \frac{|\Sigma|}{|\Sigma|-1}\nu} \right)^{2(\ell+2)} \leq \left(\frac{1-\nu}{1-2\nu} \right)^{2(\ell+2)} .$$

This is not a particularly high value (cf. Figure 4 and 5) since ν is typically small. The running time of the algorithm is bounded by order $N^2\ell^2$. The original algorithm had a running time of order $N\ell$, and here we have an additional order $N\ell$ factor because the process of estimation of the clean probabilities is using Equations (6) and (13).

3.5 General memoryless channel with known noise structure

Suppose that the letters of the output are altered independently, determined by a fixed noise matrix $\mathbf{N}_1$:

$$\mathbf{N}_1 = \begin{pmatrix} \nu_{11} & \nu_{12} & \ldots & \nu_{1m} \\ \nu_{21} & \nu_{22} & \ldots & \nu_{2m} \\ \vdots & \vdots & & \vdots \\ \nu_{m1} & \nu_{m2} & \ldots & \nu_{mm} \end{pmatrix}$$

with $\sum_{j=1}^{m} \nu_{ij} = 1$ for any row i. Let v_k and v_k^* be defined as in the proof of Theorem 3.1: the (row) vectors of clean and noisy string probabilities. Then $v_1 \mathbf{N}_1 = v_1^*$.

Definition 3.3 *For matrices* $\mathbf{A} = [a_{ij}]_{i,j=1}^{n}$ *and* $\mathbf{B}$ *define the Kronecker-product* $\mathbf{A} \otimes \mathbf{B}$ *as*

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \ldots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \ldots & a_{1n}\mathbf{B} \\ \vdots & \vdots & & \vdots \\ a_{n1}\mathbf{B} & a_{n2}\mathbf{B} & \ldots & a_{nn}\mathbf{B} \end{pmatrix}$$

Lemma 3.3 *The noise transformation matrices* $\mathbf{N}_k$ *and inverse noise transformation matrices* $\mathbf{N}_k^{-1}$ *defined by*

$$v_k \cdot \mathbf{N}_k = v_k^* \quad \textit{and} \quad v_k^* \cdot \mathbf{N}_k^{-1} = v_k$$

can be obtained by the recursions

$$\mathbf{N}_k = \mathbf{N}_1 \otimes \mathbf{N}_{k-1} \qquad \mathbf{N}_k^{-1} = \mathbf{N}_1^{-1} \otimes \mathbf{N}_{k-1}^{-1}$$

for $k > 1$*. That is,* $\mathbf{N}_k$ *(vs.* $\mathbf{N}_k^{-1}$*) is the* k*-th "Kronecker-power" of* $\mathbf{N}_1$ *(vs.* $\mathbf{N}_1^{-1}$*).*

Proof. The recursion for the noise transformation matrices is clear from the noise model. For the inverses, use the fact that

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \left(\mathbf{A}^{-1}\right) \otimes \left(\mathbf{B}^{-1}\right)$$

(see, eg., [6]). ■

Now, if we have an estimation $\hat{v}_k^*$ of the noisy string probabilities, then we can use it to obtain an estimation of the clean string probabilities $\hat{v}_k$ as

$$\hat{v}_k \stackrel{\text{def}}{=} \hat{v}_k^* \cdot \mathbf{N}_k^{-1}. \tag{17}$$

Our goal is to derive a bound on the error of the clean string probability estimation as a function of the error of the noisy string estimation and some characteristic of the noise matrix. If all noisy string probabilities are estimated within ϵ^* error, that is, $\|\hat{v}_k^* - v_k^*\|_\infty \leq \epsilon^*$, then we have

$$\|\hat{v}_k - v_k\|_\infty = \left\|(\hat{v}_k^* - v_k^*) \cdot \mathbf{N}_k^{-1}\right\|_\infty \leq \|\hat{v}_k^* - v_k^*\|_\infty \left\|\mathbf{N}_k^{-1}\right\| \leq \left\|\mathbf{N}_k^{-1}\right\| \epsilon^*,$$

where the matrix norm $\|\cdot\|$ is compatible with the infinity norm $\|\cdot\|_\infty$. A compatible matrix norm is the *column norm* defined as

$$\|\mathbf{A}\|_\gamma \stackrel{\text{def}}{=} \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}|$$

for matrix $\mathbf{A} = [a_{ij}]_{i,j=1}^{n}$. (See [6].)

It is easy to prove that for any matrices $\mathbf{A}$ and $\mathbf{B}$,

$$\|\mathbf{A} \otimes \mathbf{B}\|_\gamma \leq \|\mathbf{A}\|_\gamma \cdot \|\mathbf{B}\|_\gamma .$$

Therefore, $\left\|\mathbf{N}_k^{-1}\right\|_\gamma \leq \left(\left\|\mathbf{N}_1^{-1}\right\|_\gamma\right)^k$.

So we have the following lemma (cf. Lemma 3.1).

Lemma 3.4 *If, for the estimated noisy string probabilities on length ℓ we have*

$$\forall\, s \;\; |\hat{p}^*_s - p^*_s| \leq \epsilon^*, \tag{18}$$

then for the clean string probabilities estimated using Equation (17)

$$\forall\, s \;\; |\hat{p}_s - p_s| \leq \epsilon^* \zeta^\ell,$$

where $\zeta = \left\| \mathbf{N}_1^{-1} \right\|_\gamma$.

The following lemma corresponds to Lemma 3.2

Lemma 3.5 *If $\hat{p}^*(\cdot)$ denotes relative frequency in a sample of size*

$$N \geq \frac{2}{\epsilon^{*2}} \ln \frac{2m^\ell}{\delta}, \tag{19}$$

then

$$\mathbb{P}\left\{ \forall\, s \in \Sigma^\ell \colon\; |\hat{p}^*_s - p^*_s| \leq \epsilon^* \right\} \geq 1 - \delta. \tag{20}$$

The following theorem corresponds to Theorem 3.2

Theorem 3.4 *If the clean string probabilities are estimated using Equation (17), where the noisy string probabilities are derived from a sample of size*

$$N \geq \zeta^{2\ell} \frac{2}{\epsilon^2} \ln \frac{2m^\ell}{\delta}, \tag{21}$$

then

$$\mathbb{P}\{\, |\hat{p}_s - p_s| \leq \epsilon \,\} \geq 1 - \delta. \tag{22}$$

Remark. For the case of the symmetric channel, the value of ζ is

$$\zeta = \frac{1 + \left(1 - \frac{1}{m-1}\right)\nu}{1 - \left(1 + \frac{1}{m-1}\right)\nu},$$

it decreases with the alphabet size m and

$$\frac{1+\nu}{1-\nu} < \zeta \leq \frac{1}{1-2\nu}.$$

The value ζ is always larger than $(1+\theta)$. □

Using these values for the sample size and the same approach as before, we get that a sample of size

$$N = \Omega\left(\zeta^{2(\ell+1)} n^2 \ell^6 |\Sigma|^2 \epsilon^{-6} \log^2 \frac{\ell|\Sigma|}{\epsilon} \log |\Sigma| \delta \right)$$

can be used with the inverse noise transformation and the original algorithm.

4 Learning when noise is not known exactly

In this section I attempt to derive some way to learn the suffix machine if we have only access to a noisy sample and the noise rate is not known exactly. I will show that if the noise rate is estimated sufficiently well, then learning is possible from a sample four times as large as in the case when the noise rate is exactly known. We only consider the m-ary symmetric channel.

4.1 Noise deviation

Suppose that we use a bogus $\tilde{\nu} : |\tilde{\nu} - \nu| \leq \epsilon_\nu$ instead of the real noise rate ν in the inverse noise transformation. Define

$$\tilde{\theta} \stackrel{\text{def}}{=} \frac{\tilde{\nu}}{1 - \frac{m}{m-1}\tilde{\nu}} \quad \text{and} \quad \nu_{\max} = \tilde{\nu} + \epsilon_\nu.$$

Throughout this section we will suppose that $\nu_{\max} < 1 - 1/m$. We will denote the estimated clean string probabilities using the wrong $\tilde{\nu}$ by $\tilde{p}_s$:

$$\tilde{p}_s \stackrel{\text{def}}{=} \sum_{t \in \Sigma^\ell} \left(-\frac{\tilde{\theta}}{m-1} \right)^d (1 + \tilde{\theta})^{\ell - d} \hat{p}^*_t \tag{23}$$

$$= \sum_{d=0}^{\ell} \left(-\frac{\tilde{\theta}}{m-1} \right)^d (1 + \tilde{\theta})^{\ell - d} \hat{p}^*(B_s(d)), \tag{24}$$

similarly to Equations (6) and (7). We will proceed as follows. First we establish a bound on $|\tilde{p}_s - \hat{p}_s|$ in terms of $|\tilde{\theta} - \theta|$ and then we bound the values of $|\tilde{p}_s - p_s|$ using $|\tilde{p}_s - p_s| \leq |\tilde{p}_s - \hat{p}_s| + |\hat{p}_s - p_s|$.

The following lemma provides an upper bound for the deviation from the clean string probabilities that are estimated using the real ν.

Lemma 4.1 *For the estimation of the clean string probabilities using $\tilde{\nu}$, we have*

$$|\tilde{p}_s - \hat{p}_s| \leq \frac{|\tilde{\theta} - \theta|}{m-1} \ell (1 + \theta_{\max})^{\ell - 1},$$

where

$$\theta_{\max} = \max(\tilde{\theta}, \theta) = \frac{\nu_{\max}}{1 - \frac{|\Sigma|}{|\Sigma| - 1}\nu_{\max}}.$$

The following lemma sets a bound on how wrong the guess of the noise rate can be. It corresponds to Lemma 3.1.

Lemma 4.2 *Suppose that the noisy string probabilities are estimated within ϵ^* error, that is,*

$$\forall\, d \;\; |\hat{p}^*(B_s(d)) - p^*(B_s(d))| \leq \epsilon^*.$$

Then

$$|\tilde{p}_s - p_s| \leq \left(\epsilon^* + \left(1 - \frac{1}{m}\right)^{-2} \ell\epsilon_\nu\right)(1+\theta_{\max})^{\ell+1}.$$

The following theorem gives a lower bound for the sample size in case of estimated noise.

Theorem 4.1 *If*

$$\epsilon_\nu \leq \epsilon \frac{(1-1/m)^2}{2\ell(1+\theta_{\max})^{\ell+1}} \tag{25}$$

and the clean string probabilities are estimated using Equation (23), where the noisy string probabilities are derived from a sample of size

$$N \geq (1+\theta_{\max})^{2(\ell+1)} \frac{8}{\epsilon^2} \ln \frac{2(\ell+1)}{\delta},$$

then

$$\mathbb{P}\{\, |\tilde{p}_s - p_s| \leq \epsilon \,\} \geq 1 - \delta.$$

Proof of Theorem 4.1. Let

$$\epsilon^* \leq \frac{\epsilon}{2(1+\theta_{\max})^{\ell+1}},$$

and

$$\left(1 - \frac{1}{m}\right)^{-2} \ell\epsilon_\nu \leq \frac{\epsilon}{2(1+\theta_{\max})^{\ell+1}}.$$

By Lemma 4.2, we get $|\tilde{p}_s - p_s| \leq \epsilon$, if the empirical noisy string probabilities are within ϵ^* error.

So we have the upper bound

$$\epsilon_\nu \leq \epsilon \frac{(1-1/m)^2}{2\ell(1+\theta_{\max})^{\ell+1}}$$

on the error of the noise estimation. By Lemma 3.2, the lower bound on the sample size ensures that the noisy string probabilities are estimated within ϵ^* error with probability $\geq 1 - \delta$. ■

Theorem 4.1 shows us that even if the noise is not known exactly, learning of the PFSA is possible.

Theorem 4.2 *Suppose that for the maximal error ϵ_ν of the real noise ν, we have*

$$\nu_{\max} - 2\epsilon_\nu \leq \nu \leq \nu_{\max} < 1 - \frac{1}{|\Sigma|} \quad \textit{and} \quad \epsilon_\nu \leq \epsilon_0\epsilon_1 \frac{(1-1/|\Sigma|)^2}{2\ell(1+\theta_{\max})^{\ell+1}}$$

Then a sample of size

$$N \geq \frac{8}{\epsilon_1^2\epsilon_0^2}(1+\theta_{\max})^{2(\ell+2)}(\ell+2)\ln\frac{2|\Sigma|}{\delta} \tag{26}$$

is typical with probability $\geq 1 - \delta$, where

$$\theta_{\max} = \frac{\nu_{\max}}{1 - \frac{|\Sigma|}{|\Sigma|-1}\nu_{\max}}$$

Proof. Use Theorem 4.1 and proceed as in the proof of Theorem 3.3. ■

REMARK. The bound for ϵ_ν is small, we can expect typically a factor of $10^{-5} \ldots 10^{-2}$ with $\epsilon_0\epsilon_1$. These theorems also show that the algorithm given for the case when the noise is exactly known is robust for a small round-off error of θ. Notice that the sample size is only four times as large as that of Theorem 3.3. $\square$

The core of the learning algorithm is the same as when the noise is exactly known, except that we use $\tilde{p}$ instead of $\hat{p}$.

In the rest of the section we prove Lemmas 4.1 and 4.2.

Proof of Lemma 4.1. From Equations (7) and (24) we have

$$|\tilde{p}_s - \hat{p}_s| = \left| \sum_{d=0}^{\ell} \left(\left(-\frac{\tilde{\theta}}{m-1} \right)^d (1+\tilde{\theta})^{\ell-d} - \left(-\frac{\theta}{m-1} \right)^d (1+\theta)^{\ell-d} \right) \hat{p}_s^*(B_s(d)) \right|.$$

Define

$$\Delta_d \stackrel{\text{def}}{=} \left(-\frac{\tilde{\theta}}{m-1} \right)^d (1+\tilde{\theta})^{\ell-d} - \left(-\frac{\theta}{m-1} \right)^d (1+\theta)^{\ell-d}. \tag{27}$$

Now,

$$\begin{aligned}
\Delta_d &= \left(-\frac{\tilde{\theta}}{m-1} \right)^d (1+\tilde{\theta})^{\ell-d} - \left(-\frac{\tilde{\theta}}{m-1} \right)^d (1+\theta)^{\ell-d} \\
&\quad + \left(-\frac{\tilde{\theta}}{m-1} \right)^d (1+\theta)^{\ell-d} - \left(-\frac{\theta}{m-1} \right)^d (1+\theta)^{\ell-d} \\
&= \left(-\frac{\tilde{\theta}}{m-1} \right)^d \left((1+\tilde{\theta})^{\ell-d} - (1+\theta)^{\ell-d} \right) \\
&\quad + (1+\theta)^{\ell-d} \left(\left(-\frac{\tilde{\theta}}{m-1} \right)^d - \left(-\frac{\theta}{m-1} \right)^d \right).
\end{aligned} \tag{28}$$

For the factors in Equation (28) we have

$$\begin{aligned}
\left| \left(-\frac{\tilde{\theta}}{m-1} \right)^d - \left(-\frac{\theta}{m-1} \right)^d \right| &= \frac{1}{(m-1)^d} \left| \left(\theta - \tilde{\theta} \right) \sum_{i=0}^{d-1} (-\tilde{\theta})^i (-\theta)^{(d-1)-i} \right| \\
&\le d \frac{|\tilde{\theta} - \theta|}{(m-1)^d} \theta_{\max}^{d-1},
\end{aligned}$$

for $d = 1, \ldots, \ell$.

Similarly,

$$|(1+\tilde{\theta})^{\ell-d} - (1+\theta)^{\ell-d}| \le (\ell - d)|\tilde{\theta} - \theta|(1+\theta_{\max})^{\ell-d-1}$$

for $d = 0, \ldots, \ell - 1$.

Therefore, for $d = 1, \ldots, \ell - 1$,

$$\begin{aligned}
|\Delta_d| &\le \frac{|\tilde{\theta} - \theta|}{(m-1)^d} \left((\ell - d)\tilde{\theta}^d (1+\theta_{\max})^{\ell-d-1} + d(1+\theta)^{\ell-d} \theta_{\max}^{d-1} \right) \\
&\le \frac{|\tilde{\theta} - \theta|}{(m-1)^d} \left((\ell - d)\theta_{\max}^d (1+\theta_{\max})^{\ell-d-1} + d(1+\theta_{\max})^{\ell-d} \theta_{\max}^{d-1} \right) \\
&\le \frac{|\tilde{\theta} - \theta|}{(m-1)^d} \ell \theta_{\max}^{d-1} (1+\theta_{\max})^{\ell-d}.
\end{aligned}$$

Also,

$$|\Delta_0| \leq |\tilde{\theta}-\theta|\ell(1+\theta_{\max})^{\ell-1} \quad \text{and} \quad |\Delta_\ell| \leq \frac{|\tilde{\theta}-\theta|}{(m-1)^\ell}\ell\theta_{\max}^{\ell-1}.$$

Therefore, for all $d = 0, \ldots, \ell$, we have

$$|\Delta_d| \leq |\tilde{\theta}-\theta|\ell(1+\theta_{\max})^{\ell-1}.$$

The rest is simple:

$$\begin{aligned} |\tilde{p}_s - \hat{p}_s| &= \left|\sum_{d=0}^{\ell} \Delta_d \hat{p}_s^*(B_s(d))\right| \leq |\tilde{\theta}-\theta|\ell(1+\theta_{\max})^{\ell-1}\sum_{d=0}^{\ell}\hat{p}_s^*(B_s(d)) \\ &\leq |\tilde{\theta}-\theta|\ell(1+\theta_{\max})^{\ell-1}. \end{aligned}$$

■

Proof of Lemma 4.2. For the difference $|\tilde{\theta}-\theta|$, we get

$$|\tilde{\theta}-\theta| = \left|\left(\frac{\tilde{\nu}}{1-\frac{m}{m-1}\tilde{\nu}}\right) - \left(\frac{\nu}{1-\frac{m}{m-1}\nu}\right)\right| = \frac{|\tilde{\nu}-\nu|}{\left(1-\frac{m}{m-1}\tilde{\nu}\right)\left(1-\frac{m}{m-1}\nu\right)} \leq \frac{\epsilon_\nu}{\left(1-\frac{m}{m-1}\nu_{\max}\right)^2}.$$

Thus we obtain by using Lemmas 4.1 and 3.1 that

$$\begin{aligned} |\tilde{p}_s - p_s| &\leq |\tilde{p}_s - \hat{p}_s| + |\hat{p}_s - p_s| \leq \frac{\epsilon_\nu}{\left(1-\frac{m}{m-1}\nu_{\max}\right)^2}\ell(1+\theta_{\max})^{\ell-1} + \epsilon^*(1+\theta)^{\ell+1} \\ &\leq \left(\epsilon^* + \left(1-\frac{\nu_{\max}}{m-1}\right)^{-2}\ell\epsilon_\nu\right)(1+\theta_{\max})^{\ell+1}, \end{aligned}$$

since

$$1+\theta_{\max} = \frac{1-\frac{\nu_{\max}}{m-1}}{1-\frac{m}{m-1}\nu_{\max}}.$$

The final step is to observe

$$\left(1-\frac{\nu_{\max}}{m-1}\right)^{-2} < \left(1-\frac{1}{m}\right)^{-2}.$$

■

Acknowledgment

I would like to say thanks to James Aspnes for his advice on noise transformation and comments on earlier versions of this report.

References

[1] N. Abe and M. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260, 1992.

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

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley and Sons, New York, 1991.

[4] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[5] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire and L. Sellie. On the Learnability of discrete distributions. *STOC 94*, 273–282, 1994. Available electronically as `http://www.cs.huji.ac.il/labs/learning/Papers/dists-stoc94.ps.gz`

[6] P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, Orlando, 1985.

[7] G. Lugosi. Learning with an unreliable teacher. *Pattern Recognition*, 79–87, 1992.

[8] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–285, 1989.

[9] D. Ron, Y. Singer and N. Tishby. Learning probabilistic automata with variable memory length. *COLT 94*, 35–46, 1994. Available electronically as `http://www.cs.huji.ac.il/labs/learning/Papers/psa_colt94.ps.gz`