Learning Markov chains with variable memory length from noisy output

Dana Angluin[†] and **Miklós Csűrös**
Department of Computer Science
Yale University
P.O Box 208285, New Haven CT 06520
E-mail: {angluin,csuros}@cs.yale.edu

Abstract

The problem of modeling complicated data sequences, such as DNA or speech, often arises in practice. Most of the algorithms select a hypothesis from within a model class assuming that the observed sequence is the direct output of the underlying generation process. In this paper we consider the case when the output passes through a memoryless noisy channel before observation. In particular, we show that in the class of Markov chains with variable memory length, learning is affected by factors, which, despite being super-polynomial, are still small in some practical cases.

Markov models with variable memory length, or probabilistic finite suffix automata, were introduced in learning theory by Ron, Singer and Tishby who also described a polynomial time learning algorithm [11, 12]. We present a modification of the algorithm which uses a noise-corrupted sample and has knowledge of the noise structure. The same algorithm is still viable if the noise is not known exactly but a good estimation is available. Finally, some experimental results are presented for removing noise from corrupted English text, and to measure how the performance of the learning algorithm is affected by the size of the noisy sample and the noise rate.

[†]Research supported in part by the National Science Foundation, grant CCR–9213881.

1 Introduction

Modeling complex data sequences is one of the main problems in many research areas. Examples can be taken from computational biology [4, 6] (DNA or protein sequences), from natural language processing or speech recognition [9] (phoneme or letter sequences), etc. Most of the model building methods assume that noiseless training data are available. A natural approach is, however, to consider the observed data sequence as being the output of a string generation process corrupted by some noise. The knowledge of the noise structure may allow one to estimate the parameters for the underlying "clean" string generation model using a sample of noisy strings.

The string generation process is often assumed to have only a finite memory, that is, the generation of the next symbol of the sequence depends on a fixed finite number of previous symbols and state transitions. Such models include Hidden Markov Models [9], FSM sources in data compression [10], and Markov chains with variable memory length [12].

In this paper we consider the case when the output of a Markov chain with variable memory length is passed through a memoryless noisy channel before observation. Among others, Lugosi [8] 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).

Markov models with variable memory length, otherwise known as probabilistic finite suffix automata, were introduced in learning theory by Ron, Singer and Tishby who also described a polynomial time learning algorithm [11, 12]. Similar models have been studied in information theory as well [10, 13, 14, 15]. Other models in the learning theory literature include Hidden Markov Models [9] and probabilistic finite automata. They both 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 Probabilistic Finite Automata are not learnable in the PAC-sense [2] unless noisy parity functions are learnable.

We proceed as follows. In Section 2 we describe the framework and algorithm of Ron *et al.* for learning PFSA. In Section 3 we modify the algorithm for learning from noisy data with exact knowledge of the noise structure. In Section 4 we present some experimental results for removing noise from corrupted English text with the knowledge of the noise rate. We also attempt to measure how the performance of the learning algorithm is affected by the size of the noisy sample and the noise rate. In Section 5 we 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.* [12].

2.1 How good is a hypothesis?

A data sequence is defined as a string over a finite alphabet Σ. For DNA sequences, the alphabet consists of the four bases, a set of sounds may serve as an alphabet in a pronunciation model, and so on. It is assumed 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 H belonging to some hypothesis class $\mathcal{H}$ that approximates M well. The learning algorithm selects the hypothesis after seeing a sufficiently large sample of strings generated by M.

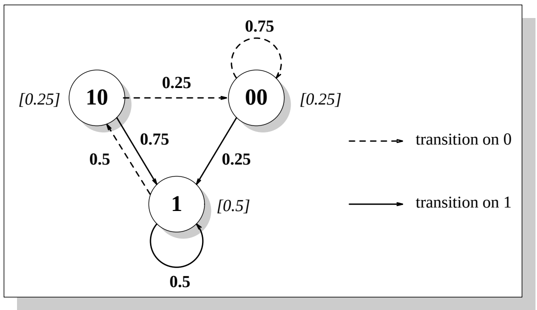
In this paper we consider probabilistic machines without a "stop state", that is, machines that generate an infinite string at each run. In order to pose practical questions and to deal with measurability, we restrict our attention to finite length prefixes of the machine output. We use prefix and suffix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

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 quantifying the difference between μ_M and μ_H. The chosen measure of difference is the relative entropy, also called the Kullback-Leibler divergence, defined as

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

(Throughout this paper, log denotes base 2 logarithm and ln denotes natural logarithm.)

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 $D_\ell[\,\mu_M \,\|\, \mu_H\,] < \ell\epsilon$.*



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. (Example taken from [12].)

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

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.

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

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

(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

Definition 2.2 *The model class of Probabilistic Finite Automata is the set of five-tuples $\langle Q, \Sigma, \tau, \gamma, \pi\rangle$: a finite set of states Q, a finite alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, a next symbol probability function $\gamma : Q \times \Sigma \mapsto [0,1]$ ($\forall q \in Q$: $\sum_{\sigma\in\Sigma}\gamma(q,\sigma) = 1$), a state transition function $\tau : Q \times \Sigma \mapsto Q$, and an initial probability distribution $\pi : Q \mapsto [0,1]$ ($\sum_{q\in Q}\pi(q) = 1$).*

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 probability $p_s = \mu_M(s) = \sum_{q_0\in Q}\pi(q_0)\prod_{i=1}^{\ell}\gamma(q_{i-1}, s_i)$, where $q_i = \tau(q_{i-1}, s_i)$ for $i > 0$.*

Recall that by the result of Kearns *et al.* [5], it is unlikely that there is an efficient learning algorithm for the model class of PFA. We 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 seems to have unbounded memory for the observer of the output, since the starting state is always remembered. In the class of Probabilistic Finite Suffix Automata, $\tau(\cdot)$ 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 a suffix-free subset 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 [12] 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 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 algorithm starts with a state for the empty string and keeps splitting states whenever necessary. Without going into details here, "splitting" means that the state for suffix s is replaced by states for longer suffixes $s's$, that is, suffixes are grown "from right to left". In the course of the algorithm, a set S is maintained that consists of the strings that occur sufficiently often in the sample and are therefore worth considering. The outline of the algorithm is the following (cf. Figure 2).

(I) Estimate from the sample:
— $p'(s)$: the probability of string $s \in \Sigma^{\leq \ell+1}$, and
— for $p'(s) > 0$, $p'(\sigma|s)$: the next symbol probability.

(II.1) Start with a state for the empty string. Initialize $S \leftarrow \{\sigma : \sigma \in \Sigma,\, p'(\sigma) \geq (1-\epsilon_{\mathrm{str}})\gamma_{\mathrm{str}}\}$

(II.2) While $S \neq \emptyset$ do

(II.2.a) Remove an arbitrary s from S.

(II.2.b) If for some σ

$$p'(\sigma|s) \geq (1+\epsilon_{\mathrm{sym}})\gamma_{\mathrm{sym}}$$

and

$$p'(\sigma|s) > (1+3\epsilon_{\mathrm{sym}})p'(\sigma|\mathit{suffix}(s)),$$

then add s to the set of states. *suffix*(s) is the longest suffix of s that already has a corresponding state.

(II.2.c) If $|s| < \ell$, then for every $\sigma \in \Sigma$ if $p'(\sigma s) \geq (1-\epsilon_{\mathrm{str}})\gamma_{\mathrm{str}}$, then add σs to S.

(II.3) Rescale probabilities and add missing states.

The threshold parameters γ_{str} and γ_{sym}, and the precision parameters[1] ϵ_{str} and ϵ_{sym} for the string and next symbol probabilities are relatively simple functions of n, ℓ, $|\Sigma|$, ϵ, and δ.

The proof of correctness and the estimation of running time of the algorithm are 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 state s, and (2) the relative next symbol frequencies $p'(\sigma|s)$ are close to the real probabilities $p(\sigma|s)$ for all s that often appear in the sample. The second theorem states that if the algorithm is given a typical sample, then it outputs an ϵ-good hypothesis H with the features described above.

We mention here that similar frameworks and algorithms can be found in the information theory literature for data compression (FSM and FSMX sources, context trees, see, eg., [10], [13] and [14]). The per-symbol entropy $D_\ell[\,\mu_M \,\|\, \mu_H\,]/\ell$ equals the minimum expected number of extra bits when the data sequence generated by M is encoded using the hypothesis H instead of the optimal code based on M. The main difference is that here H is selected based on a sample and its performance is measured on all the strings, whereas in data compression, the performance is measured on the same "sample" string.

We use the algorithm of Ron, Singer and Tishby as a "core" algorithm for learning from noisy data. We believe that it is possible to use a different algorithm with minor modifications for the "core" in the same manner. The asymptotically optimal methods used for data compression are good candididates but their application is not trivial. For example, the very effective approach of

[1] The original algorithm of [12] uses the notation ϵ_0, $\gamma_{\min}$, ϵ_1, and ϵ_2 for our γ_{str}, γ_{sym}, ϵ_{str}, and ϵ_{sym}, respectively.

Willems *et al.* [15] outputs a distribution over PFSA's instead of one hypothesis machine. Naively using the tree building methods of [10] or [13] may result in an unduly large number of states, etc. In either way, using our approach (or a similar one) for learning from noisy output most likely causes similar effects in running time and sample size. Therefore, we proceed with studying the PFSA learning algorithm of Ron *et al.*

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 Noise model

We consider the case when the output of the machine passes through a memoryless 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 analyze first a simple noise model: a memoryless symmetric channel. Each character of the output is altered by the same 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.

3.2 Inverse noise transformation

The Hamming-distance $d(s, s')$ between two strings $s = \sigma_1\sigma_2 \cdots \sigma_\ell$ and $s' = \sigma'_1\sigma'_2 \cdots \sigma'_\ell$ is defined as

$$d(s, s') \stackrel{\text{def}}{=} \sum_{i=1}^{\ell} \mathrm{I}_{\{\sigma_i \neq \sigma'_i\}},$$

where $\mathrm{I}_{\{\sigma_i \neq \sigma'_i\}}$ equals 1 if the two symbols differ and equals 0 if they are the same.

Theorem 3.1 *The clean and noisy string probabilities can be expressed in terms of each other as*

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

$$p_s = \sum_{t \in \Sigma^\ell} \left(-\frac{\theta}{m-1}\right)^{d(s,t)} (1+\theta)^{\ell - d(s,t)} p^*_t, \qquad (2)$$

where $d(s,t)$ is the Hamming-distance between s and t, and

$$\theta \stackrel{\text{def}}{=} \frac{\nu}{1 - \frac{m}{m-1}\nu}. \qquad (3)$$

Proof. Equation (1) is clear from the noise 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. Define the noise transformation matrices $\mathbf{N}_k$ and the inverse noise transformation matrices $\mathbf{N}_k^{-1}$ such that $v^*_k = v_k \cdot \mathbf{N}_k$ and $v_k = v^*_k \cdot \mathbf{N}_k^{-1}$. Notice that these matrices have size $m^k \times m^k$. For $k = 1$, we have

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

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

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

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

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

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

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

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

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

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

where the upper bound is reached in the case of a binary alphabet. This is important in the following results because the required sample size for effective learning grows with $(1+\theta)$ and thus the worst case is that of a 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 usually consider only $\nu < 1-1/m$. □

3.3 Inverse noise transformation with empirical probabilities

Suppose now that we use Equation (2) with estimated probabilities. How does the estimation error propagate?

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

$$\hat{p}_s \stackrel{\text{def}}{=} \sum_{t\in\Sigma^\ell} \left(-\frac{\theta}{m-1}\right)^{d(s,t)} (1+\theta)^{\ell-d(s,t)} \hat{p}^*_t \qquad (4)$$

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

with the Hamming-ball of radius d around s defined as $B_s(d) \stackrel{\text{def}}{=} \{r : r \in \Sigma^\ell, d(s,r) = d\}$.

Lemma 3.1 *If, given a string s of length ℓ, all the probabilities $p^*(B_s(d))$ are estimated within ϵ^* error, then the clean string probabilities are within $\epsilon^*(1+\theta)^{\ell+1}$ error when estimated using Equation (4).*

Theorem 3.2 *If the clean string probabilities are estimated using Equation (4), where the noisy string probabilities are derived from a sample of size*

$$N \geq (1+\theta)^{2(\ell+1)} \frac{2}{\epsilon^2} \ln \frac{2(\ell+1)}{\delta}, \qquad (6)$$

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

PROOF OF THEOREM 3.2. Use Lemma 3.1 by setting $\epsilon^* = \epsilon/(1+\theta)^{\ell+1}$ together with Hoeffding's inequality. ■

PROOF OF LEMMA 3.1. From Equations (2) and (5) we have

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

$$< \epsilon^* \frac{(1+\theta)^\ell}{1-\frac{\theta/(m-1)}{1+\theta}} \leq \epsilon^*(1+\theta)^{\ell+1} \qquad ■$$

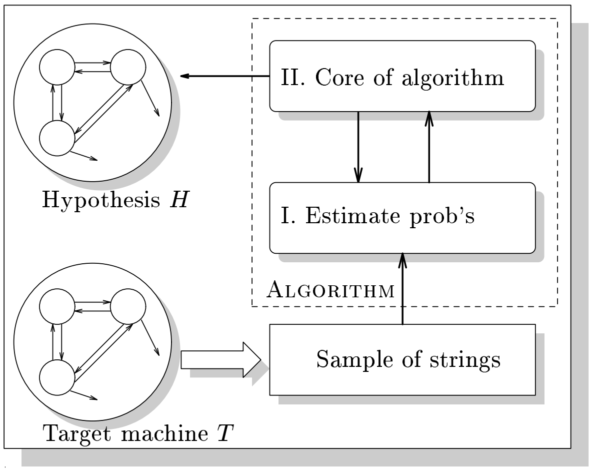


Figure 2: PFSA learning process

REMARK. Note that we could have obtained an upper bound of the form $\epsilon^*(\ell+1)(1+\theta)^\ell$ but $(\ell+1)$ is expected to be larger than $(1+\theta)$ for small[2] values of ν. □

3.4 Algorithm when noise is known

We use the algorithm of Ron, Singer and Tishby [12]. The main structure of the learning process is sketched in Figure 2. From a sample of strings generated by the target automaton T, the algorithm estimates the string probabilities and outputs a hypothesis machine H. No noise is introduced in the original model, 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. It is proved 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 has the same outline. We use a different method for estimating 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 are estimated using Equation (4). The other difference is that we only compute estimations as needed, whereas in the original algorithm, the estimates can be precomputed before starting the core part. Therefore, Step (I) is missing, and Step (II) differs only in using $\hat{p}(\cdot)$ instead of $p'(\cdot)$. In the original algorithm, $p'(\cdot)$ was found by a table-lookup, whereas in the modified one, $\hat{p}(\cdot)$ is a call to a subroutine. Since we only need to consider the noisy strings that appear in the sample, one clean string probability estimation takes linear time in the sample size and ℓ,

[2] $\ell+1 \geq 1+\theta$, if $\nu \leq \frac{\ell}{1+\frac{m}{m-1}\ell}$, which is always satisfied for $\nu \leq 1/3$.

the length of the longest suffix considered. This results in a new factor in the running time of the algorithm. We shall tighten the notion of a typical sample so that we can have an algorithm that works on a noisy sample. Since we require somewhat better accuracy in the clean string probability estimates than the original definition, and use the same core algorithm, we still get 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 can be obtained easily directly from the sample. For $p'(\sigma|s)$, we count all the occurrences of σ following s. In the case of a noisy sample, however, this is not feasible. We define the estimated clean next symbol probability as $\hat{p}(\sigma|s) \stackrel{\text{def}}{=} \frac{\hat{p}(s\sigma)}{\hat{p}(s)}$ if $\hat{p}(s) > 0$.

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 3.1 *A sample generated by an ℓ-PFSA is $(\epsilon_{\text{str}}, \gamma_{\text{str}}, \epsilon_{\text{sym}}, \gamma_{\text{sym}})$-typical if for the probabilities $\hat{p}_s$ estimated using the sample, the following two properties hold:*

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

REMARK. This definition differs in property 1 from that of the paper of Ron *et al.* In that paper, 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. □

Lemma 3.2 *Property 2 is implied by property 1 if $4\epsilon_{\text{str}} \leq \epsilon_{\text{sym}}\gamma_{\text{sym}}$.*

(Proof in Appendix.)

Theorem 3.3 *A sample of size*

$$N \geq \frac{2}{\epsilon_{\text{str}}^2 \gamma_{\text{str}}^2}(1+\theta)^{2(\ell+2)}(\ell+2)\ln\frac{2|\Sigma|}{\delta} \tag{7}$$

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. By Lemma 3.2, we only need to prove that the sample satisfies property 1 with probability at least $1-\delta$. Since $\left|\Sigma^{\leq \ell+1}\right| < m^{(\ell+2)}$, if the inequality holds for an arbitrary string with probability $\geq 1-\delta/m^{(\ell+2)}$ then it holds for all strings in $\Sigma^{\leq \ell+1}$ with probability at least $1-\delta$. Thus, in order to prove property 1, just plug $\ell \leftarrow \ell+1, \epsilon \leftarrow \epsilon_{\text{str}}\gamma_{\text{str}}$, and $\delta \leftarrow \delta/m^{(\ell+2)}$ into Equation (6) and notice that

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

3.5 Running time and sample size

The constraint of Lemma 3.2 and the constraints of [11] help us to set the threshold and precision parameters. The following values can be used.

$$\epsilon_{\text{str}} = \frac{\epsilon_{\text{sym}}\gamma_{\text{sym}}}{4} = \frac{\epsilon^2}{9216\ell^2|\Sigma|}, \tag{8}$$

$$\gamma_{\text{str}} = \frac{\epsilon}{2n\ell\log(1/\gamma_{\text{sym}})} = \frac{\epsilon}{2n\ell\log(48\ell|\Sigma|/\epsilon)}, \tag{9}$$

$$\epsilon_{\text{sym}} = \frac{\epsilon}{48\ell}, \tag{10}$$

$$\gamma_{\text{sym}} = \frac{\epsilon_{\text{sym}}}{|\Sigma|} = \frac{\epsilon}{48\ell|\Sigma|}. \tag{11}$$

Plugging these values into Equation (7) we get

Theorem 3.4 *There exists an*

$$N = O\left(\Lambda n^2 \ell^7 |\Sigma|^2 \epsilon^{-6} \log^2\frac{\ell|\Sigma|}{\epsilon}\log\frac{|\Sigma|}{\delta}\right),$$

such that a sample of size at least N can be used for efficient learning of PFSA from noisy output if the noise rate is known. Λ denotes the exponential term

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

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 uses Equation (4).

REMARK. The presence of an exponential factor (which is expected when learning from noisy data) is not as inconvenient as it may seem at first sight. Λ is not a particularly high value (cf. Figure 3) since ν is typically small. □

3.6 General memoryless channel with known noise structure

Similar results can be derived in the general case when the letters of the output are altered independently, determined by a fixed noise matrix $\mathbf{N}_1 = [\nu_{ij}]_{i,j=1,\ldots,m}$ with $\sum_{j=1}^m \nu_{ij} = 1$ for any row i. The general idea is the following. Let v_k and v_k^* be defined as in the proof of Theorem 3.1, that is, the (row) vectors of clean and noisy string probabilities at length k. Then $v_1\mathbf{N}_1 = v_1^*$. The entries of the noise matrix $\mathbf{N}_k$ used in the noise

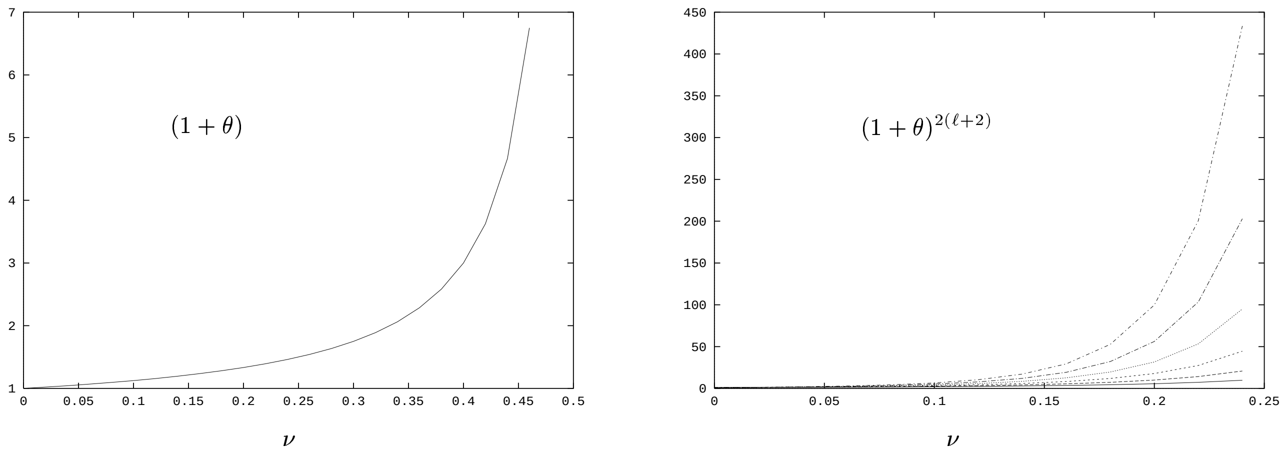


Figure 3: The values of $(1+\theta)$ and the exponential factor $\Lambda = (1+\theta)^{2(\ell+2)}$ for memory lengths $\ell = 1, \dots, 6$ (binary alphabet).

transformation $v_k \mathbf{N}_k = v_k^*$ can be derived from the entries of $\mathbf{N}_1$. 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 by $\hat{v}_k = \hat{v}_k^* \cdot \mathbf{N}_k^{-1}$. The entries of $\mathbf{N}_k^{-1}$ are derived from the entries of $\mathbf{N}_1^{-1}$.

For the detailed discussion we need some tools from matrix theory.

Definition 3.2 *For matrices $\mathbf{A} = [a_{ij}]_{i,j=1}^m$ and $\mathbf{B}$ define the Kronecker-product $\mathbf{A} \otimes \mathbf{B}$ as*

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1m}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{1m}\mathbf{B} \\ \vdots & \vdots & & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \dots & a_{mm}\mathbf{B} \end{pmatrix}$$

Lemma 3.3 *The noise transformation matrices $\mathbf{N}_k$ and the 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, $\mathbf{N}_k$ is the k-th "Kronecker-power" of $\mathbf{N}_1$, and $\mathbf{N}_k^{-1}$ is the k-th Kronecker-power of $\mathbf{N}_1^{-1}$.

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

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

(see, eg., [7]). ■

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

$$\hat{v}_k \stackrel{\text{def}}{=} \hat{v}_k^* \cdot \mathbf{N}_k^{-1}. \tag{12}$$

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, $\|\hat{v}_k^* - v_k^*\|_\infty \le \epsilon^*$, then we have

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

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

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

for matrix $\mathbf{A} = [a_{ij}]_{i,j=1}^m$. (See [7].)

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

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

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

So we have the following lemma.

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

$$\forall\, s \ |\hat{p}_s^* - p_s^*| \le \epsilon^*, \tag{13}$$

then for the clean string probabilities estimated using Equation (12)

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

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

PROOF. Similar to that of Lemma 3.1. Observe that we require precision in the estimation of individual string probabilities instead of Hamming-balls: we do not assume symmetry in the noise matrix. ■

The following theorem corresponds to Theorem 3.2.

Theorem 3.5 *If the clean string probabilities are estimated using Equation (12), where the noisy string probabilities are derived from a sample of size*

$$N \geq \zeta^{2\ell} \frac{2}{\epsilon^2} \ln \frac{2m^\ell}{\delta}, \tag{14}$$

then

$$\mathbb{P}\{\, |\hat{p}_s - p_s| \leq \epsilon \,\} \geq 1 - \delta. \tag{15}$$

PROOF. Use Lemma 3.4 by setting $\epsilon^* = \epsilon/\zeta^\ell$ together with Hoeffding's inequality. ■

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

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

it decreases with the alphabet size m and

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

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

Similarly to Theorem 3.3, one can prove that a sample of size

$$N \geq \frac{2}{\epsilon_{\mathrm{str}}^2 \gamma_{\mathrm{str}}^2} \zeta^{2(\ell+1)} (2\ell + 3) \ln \frac{2m}{\delta}$$

is typical with probability at least $1 - \delta$. Plugging in the values for ϵ_{str} and γ_{str} gives the following theorem for the sample size.

Theorem 3.6 *There exists an*

$$N = O\left(\Lambda n^2 \ell^7 |\Sigma|^2 \epsilon^{-6} \log^2 \frac{\ell|\Sigma|}{\epsilon} \log \frac{|\Sigma|}{\delta}\right)$$

such that a sample of size at least N can be used for efficient learning of PFSA from noisy output if the noise matrix $\mathbf{N}_1$ is known. Λ denotes the exponential term $\Lambda = \left(\|\mathbf{N}_1^{-1}\|_\gamma\right)^{2(\ell+1)}$.

Notice that the algorithm does not actually have to carry out the costly matrix multiplications with matrices of size $m^k \times m^k$, when Equation (12) is used, since most of the entries of $\hat{v}_k^*$ are 0. It is enough to go through the noisy strings that appear in the sample whenever $\hat{p}_s$ needs to be computed for some s, and sum up the probabilities of altering s into the observed noisy strings using $\mathbf{N}_1^{-1}$.

Original text: alice opened the door and found that it led into a small passage not much larger than a rat hole she knelt down and looked along the passage into the loveliest garden you ever saw

Corrupted text: aliceydpeneb uhexdohc and ftund nhat itzljd ivoi a smfll passage not ruch laraer qshhq r rat hgle xhe kdelj downfand lookev as ng the pmssage knto the loveldest garden yojdepeitsaw

Corrected text: alice spened the dohe and found that it led it i a small passage not ruch larker shhe r hat hile the keele down and looked as ng the pessage knto the loveldest garden yo depeitsaw

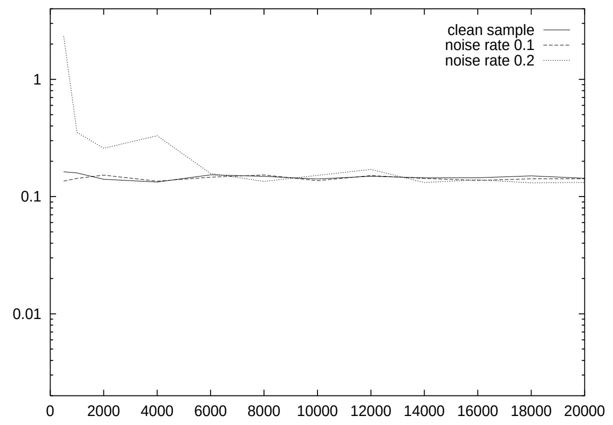
Figure 4: Alice in Wonderland

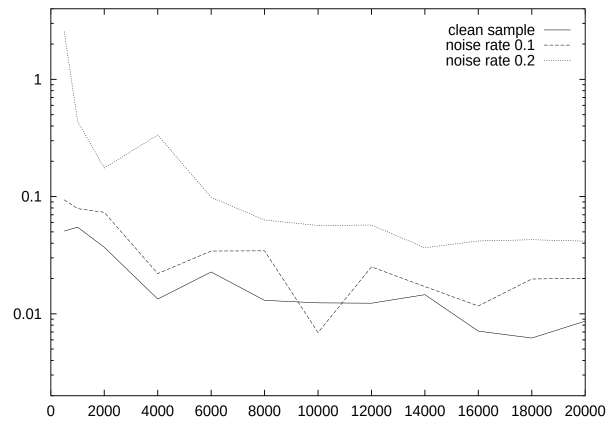
4 Experimental results

The learning algorithm was used to construct a model of English text for removing noise from corrupted text. The sample consisted of the text of Lewis Carroll's *Alice in Wonderland* without the first chapter, corrupted using a noise rate of $\nu = 0.2$. The suffix length was set to $\ell = 5$. The model obtained was used on the corrupted text of the first chapter to find the most probable state sequence via dynamic programming. The results are shown in Figure 4. Notice that although modeling Lewis Carroll by a PFSA of order 5 is clearly an oversimplification, the correction is (more or less) successful. As Ron *et al.* point it out, the PFSA model of English captures some aspects of the language, having states such as "in th" and "there", for example.

The lower bounds on the sample sizes are of rather theoretical interest since the $O(\cdot)$ notations hide large factors. Therefore, it is not recommended to use the parameter settings of Equations (8)...(11) for practical purposes. We conducted some experiments to measure the effect of noise on the required sample size. Figure 5 shows how the per-symbol divergence between the target machine and the hypothesis depends on the size of the noisy sample and the noise rate. For the experiments, the target was chosen to be the simple machine M depicted on Figure 1. The learning algorithm used the parameter settings $\epsilon_{\mathrm{str}} = 0.001$, $\gamma_{\mathrm{str}} = 0.05$, $\epsilon_{\mathrm{sym}} = 0.01$, $\gamma_{\mathrm{sym}} = 1e-5$ and $\ell = 3$. The Kullback-Leibler divergence between M and the hypothesis was estimated using a sample consisting of 10000 strings of length 6 generated by M. The average and standard deviation of the Kullback-Leibler divergence was calculated by carrying out the learning 10 times for each sample size and noise rate.

As it can be seen from the graphs, the average hypothesis learnt from a noisy sample is not much worse than the average one learnt from a clean sample for reasonable sample sizes. However, the deviation from the average is clearly affected by the noise rate. Our ap-





The graphs show how the per-symbol entropy $D_\ell[\,\mu_M \,\|\, \mu_H\,]/\ell$ depends on the size of the sample on which H is based. The target machine M was the machine depicted on Figure 1. The graph on the left hand side shows the average per-symbol entropy. The graph on the right hand side shows the standard deviation.

Figure 5: Per-symbol relative entropy vs. sample size

proach predicts that the sample size must be increased by a factor of $(1+\theta)^{2(\ell+2)}$ to get the same performance. For $\nu = 0.1$, this means a factor of 3, for $\nu = 0.2$, a factor of 18 roughly. A very similar behavior can be seen on the graph for the standard deviation.

5 Learning when noise is not known exactly

In this section we attempt to derive some way to learn the suffix machine if we have access only to a noisy sample and the noise rate is not known exactly. We 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 consider only the m-ary symmetric channel.

5.1 Noise deviation

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

$$\begin{aligned} \tilde{\theta} &\stackrel{\text{def}}{=} \frac{\tilde{\nu}}{1 - \frac{m}{m-1}\tilde{\nu}}, \quad \text{and} \\ \theta_{\max} &= \max(\tilde{\theta}, \theta) = \frac{\nu_{\max}}{1 - \frac{|\Sigma|}{|\Sigma|-1}\nu_{\max}}. \end{aligned}$$

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

$$\begin{aligned} \tilde{p}_s &\stackrel{\text{def}}{=} \sum_{t \in \Sigma^\ell} \left(-\frac{\tilde{\theta}}{m-1}\right)^d (1+\tilde{\theta})^{\ell-d} \hat{p}^*_t && (16) \\ &= \sum_{d=0}^{\ell} \left(-\frac{\tilde{\theta}}{m-1}\right)^d (1+\tilde{\theta})^{\ell-d} \hat{p}^*(B_s(d)), && (17) \end{aligned}$$

analogously to Equations (4) and (5).

Lemma 5.1 *Suppose that the noisy string probabilities are estimated within ϵ^* error. Then*

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

(Proof in Appendix.)

Theorem 5.1 *If*

$$\epsilon_\nu \leq \epsilon \frac{(1-1/m)^2}{2\ell(1+\theta_{\max})^{\ell+1}} \qquad (18)$$

and the clean string probabilities are estimated using Equation (16), 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 5.1. Use Lemma 5.1 by setting $\epsilon^* = \epsilon/(2(1+\theta_{\max})^{\ell+1})$ together with Hoeffding's inequality. ∎

Theorem 5.1 shows that even if the noise is not known exactly, learning of the PFSA is possible.

Theorem 5.2 *Suppose that for the maximal error ϵ_ν of the real noise ν, we have*

$$\begin{aligned} \nu_{\max} - 2\epsilon_\nu &\leq \nu \leq \nu_{\max} < 1 - \frac{1}{|\Sigma|} \quad \textit{and} \\ \epsilon_\nu &\leq \epsilon_{\rm str}\gamma_{\rm str}\frac{(1-1/|\Sigma|)^2}{2\ell(1+\theta_{\max})^{\ell+1}} \end{aligned}$$

Then a sample of size

$$N \geq \frac{8}{\epsilon_{\rm str}^2\gamma_{\rm str}^2}(1+\theta_{\max})^{2(\ell+2)}(\ell+2)\ln\frac{2|\Sigma|}{\delta} \tag{19}$$

is typical with probability $\geq 1-\delta$.

Proof of Theorem 5.2. Use Theorem 5.1 and proceed as in the proof of Theorem 3.3. ■

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

6 Conclusion

We showed that the PFSA learning algorithm of Ron, Singer and Tishby can be used as the core of a learning algorithm for learning from noisy data if

- either the noise rate is known in the case of symmetric noisy channel,
- or the noise matrix is known in the case of general memoryless noisy channel,
- or at least the noise rate is very well estimated in the case of symmetric noisy channel.

Among others, the following open questions arise from these results.

- Can a more efficient learning algorithm be designed using a different technique based on data compression methods?
- What lower bounds on sample size can be proved?
- Is there an elegant way to handle the case of estimated general memoryless noise? What is the effect of a bad noise estimation? Is it necessary to estimate the noise rate well?
- Can the results be extended to more complicated noise models?

Acknowledgment

We would like to thank James Aspnes for his advice on noise transformation and comments on earlier versions of this paper, and Andrew Barron for calling our attention to the data compression literature.

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] S. Eddy. Hidden Markov models. *Current Opinion in Structural Biology*, **6:**361–365, 1996.

[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.

[6] A. Krogh, B. Brown, I. S. Mian, K. Sjölander,, D. Haussler. Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, **235:**1501–1531, 1994.

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

[8] G. Lugosi. Learning with an unreliable teacher. *Pattern Recognition*, **25:**79–87, 1992.

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

[10] J. Rissanen. Complexity of strings in the class of Markov sources. *IEEE Transactions on Information Theory*, **IT-32:**526–532, 1986.

[11] D. Ron, Y. Singer and N. Tishby. Learning probabilistic automata with variable memory length. *COLT 94*, 35–46, 1994.

[12] D. Ron, Y. Singer and N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, **25:**117–149, 1996.

[13] M. J. Weinberger, A. Lempel, J. Ziv. A sequential algorithm for the universal coding of finite memory sources. *IEEE Transactions on Information Theory*, **IT-38:**1002–1014, 1992.

[14] M. J. Weinberger, J. J. Rissanen, M. Feder. A universal finite memory source. *IEEE Transactions on Information Theory*, **IT-41:**643–652, 1995.

[15] F. M. J. Willems, Y. M. Shtarkov, Tj. J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, **IT-41:**653–664, 1995.

Appendix — proof of technical lemmas

PROOF OF LEMMA 3.2. Consider any $s \in \Sigma^{\leq \ell}$ with $\hat{p}_s \geq (1-\epsilon_{\mathrm{str}})\gamma_{\mathrm{str}}$. Then

$$\begin{aligned}
&|\hat{p}(\sigma|s) - p(\sigma|s)| \\
&\quad = \left|\frac{\hat{p}(s\sigma)}{\hat{p}(s)} - \frac{p(s\sigma)}{p(s)}\right| \\
&\quad = \frac{1}{\hat{p}(s)p(s)}\left|\hat{p}(s\sigma)p(s) - p(s\sigma)\hat{p}(s)\right| \\
&\quad = \frac{\left|\hat{p}(s\sigma)p(s) - p(s\sigma)p(s) + p(s\sigma)p(s) - p(s\sigma)\hat{p}(s)\right|}{\hat{p}(s)p(s)} \\
&\quad \leq \frac{\left|\hat{p}(s\sigma) - p(s\sigma)\right| + p(\sigma|s)\left|\hat{p}(s) - p(s)\right|}{\hat{p}(s)} \\
&\quad \leq \frac{\epsilon_{\mathrm{str}}\gamma_{\mathrm{str}}}{(1-\epsilon_{\mathrm{str}})\gamma_{\mathrm{str}}}(1 + p(\sigma|s)) \leq \frac{2\epsilon_{\mathrm{str}}}{1-\epsilon_{\mathrm{str}}}.
\end{aligned}$$

If $\epsilon_{\mathrm{str}} \leq 1/2$, then the right-hand side is not greater than $4\epsilon_{\mathrm{str}}$. Therefore we impose the constraint $4\epsilon_{\mathrm{str}} \leq \epsilon_{\mathrm{sym}}\gamma_{\mathrm{sym}}$. It replaces the condition $8n\epsilon_{\mathrm{str}}\gamma_{\mathrm{str}} \leq \epsilon_{\mathrm{sym}}\gamma_{\mathrm{sym}}$ set in [11] in order to prove the second property in a different way. ∎

PROOF OF LEMMA 5.1. We use the fact that $|\tilde{p}_s - p_s| \leq |\tilde{p}_s - \hat{p}_s| + |\hat{p}_s - p_s|$. For the second term we have the bound

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

from Lemma 3.1. For the first term we have

$$\tilde{p}_s - \hat{p}_s = \sum_{d=0}^{\ell} \hat{p}_s^*(B_s(d))\Delta_d,$$

where Δ_d is defined as

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

It is not too hard to prove that for all $d = 0, \ldots, \ell$,

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

Therefore the first term can be bounded as

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

The difference $|\tilde{\theta} - \theta|$ can be bounded from above as

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

Therefore,

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

since

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

The final step is to observe that

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

∎