A Spectral Learning Algorithm for Finite State Transducers [⋆]

Borja Balle, Ariadna Quattoni, and Xavier Carreras

Universitat Politècnica de Catalunya
{bballe,aquattoni,carreras}@lsi.upc.edu

Abstract. Finite-State Transducers (FSTs) are a popular tool for modeling paired input-output sequences, and have numerous applications in real-world problems. Most training algorithms for learning FSTs rely on gradient-based or EM optimizations which can be computationally expensive and suffer from local optima issues. Recently, Hsu et al. [13] proposed a spectral method for learning Hidden Markov Models (HMMs) which is based on an Observable Operator Model (OOM) view of HMMs. Following this line of work we present a spectral algorithm to learn FSTs with strong PAC-style guarantees. To the best of our knowledge, ours is the first result of this type for FST learning. At its core, the algorithm is simple, and scalable to large data sets. We present experiments that validate the effectiveness of the algorithm on synthetic and real data.

1 Introduction

Probabilistic Finite-State Transducers (FSTs) are a popular tool for modeling paired input-output sequences, and have found numerous applications in areas such as natural language processing and computational biology. Most training algorithms for learning FSTs rely on gradient-based or EM optimizations which can be computationally expensive and suffer from local optima issues [8,10]. There are also methods that are based on grammar induction techniques [5,3], which have the advantage of inferring both the structure of the model and the parameters.

At the same time, for the closely-related problem of learning Hidden Markov Models (HMMs), different algorithms based on the *Observable Operator Model* (OOM) representation of the HMM have been proposed [6,18,13,23]. The main idea behind OOMs is that the probabilities over sequences of observations generated by an HMM can be expressed as products of matrix operators [22,4,11,14]. This view of HMMs allows for the development of learning algorithms which are based on eigen-decompositions of matrices. Broadly speaking, these spectral

[⋆] This work was partially supported the EU PASCAL2 Network of Excellence (FP7-ICT-216886), and by a Google Research Award. B.B. was supported by an FPU fellowship (AP2008-02064) of the Spanish Ministry of Education. The Spanish Ministry of Science and Innovation supported A.Q. (JCI-2009-04240) and X.C. (RYC-2008-02223 and "KNOW2" TIN2009-14715-C04-04).

decompositions can reveal relationships between observations and hidden states by analyzing the dynamics between observations. Other approaches to language learning, also based on linear algebra, can be found in the literature, e.g. [9,2].

In this paper we show that the OOM idea can also be used to derive learning algorithms for parameter estimation of probabilistic non-deterministic FSTs. While learning FSTs is in general hard, here we show that a certain class of FSTs can be provably learned. Generalizing the work by Hsu et al. [13], we present a spectral learning algorithm for a large family of FSTs. For this algorithm we prove strong PAC-style guarantees, which is the first such result for FST learning to the best of our knowledge. Our sample complexity bound depends on some *natural* parameters of the input distribution, including the spread of the distribution and the expected length of sequences. Furthermore, we show that for input distributions that follow a Markov process, our general bound can be improved. This is important for practical applications of FSTs where the input distribution is well approximated by Markov models, such as in speech and language processing [15].

Like in the case for HMMs [6,18,13], our learning algorithm is based on spectral decompositions of matrices derived from estimated probabilities over triples of symbols. The method involves two simple calculations: first, counting frequencies on the training set; and second, performing SVD and inversions on matrices. The size of the input alphabet only has an impact on the first step, i.e. computing frequencies. Therefore, the algorithm scales well to very large training sets. Another good property of the method is that only one parameter needs to be tuned, namely the number of hidden states of the FST. Our theoretical analysis points to practical ways to narrow down the range of this parameter.

We present synthetic experiments that illustrate the properties of the algorithm. Furthermore, we test our algorithm for the task of transliterating names between English and Russian. In these experiments, we compare our method with an Expectation Maximization algorithm, and we confirm the practical utility of the spectral algorithm at learning FSTs on a real task.

The rest of the paper is organized as follows. Section 2 presents background materials on FSTs, together with their OOM representation. Section 3 presents the spectral learning algorithm, and Section 4 gives the main theoretical results of the paper. In sections 5 and 6 we present experiments on synthetic data and on a transliteration task. Section 7 concludes the paper.

2 Probabilistic Finite State Transducers

In this paper we use Finite-State Transducers (FSTs) that model the conditional probability of an output sequence $y = y_1 \cdots y_t$ given an input sequence $x = x_1 \cdots x_t$. Symbols x_s belong to an input alphabet $\mathcal{X} = \{a_1, \ldots, a_k\}$, symbols y_s belong to an output alphabet $\mathcal{Y} = \{b_1, \ldots, b_l\}$, and t is the length of both sequences.[1] We denote the cardinalities of these alphabets as $k = |\mathcal{X}|$ and $l = |\mathcal{Y}|$. Throughout the paper, we use x and y to denote two input-output sequences of length t, we use a and a' to denote arbitrary symbols in $\mathcal{X}$ and b to denote an arbitrary symbol in $\mathcal{Y}$. Finally, we use $x_{r:s}$ to denote the subsequence $x_r \cdots x_s$.

A *probabilistic non-deterministic FST* — which we simply call FST — defines a conditional distribution $\mathbb{P}$ of y given x using an intermediate hidden state sequence $h = h_1 \cdots h_t$, where each h_s belongs to a set of m hidden states $\mathcal{H} = \{c_1, \ldots, c_m\}$. Then, the FST defines:

$$\begin{aligned}\mathbb{P}(y|x) &= \sum_{h \in \mathcal{H}^t} \Pr_{\mathbb{P}}[y, h|x] \\ &= \sum_{h \in \mathcal{H}^t} \Pr_{\mathbb{P}}[h_1] \Pr_{\mathbb{P}}[y_1|h_1] \prod_{s=2}^{t} \Pr_{\mathbb{P}}[h_s|x_{s-1}, h_{s-1}] \Pr_{\mathbb{P}}[y_s|h_s] \qquad (1)\end{aligned}$$

The independence assumptions are that $\Pr[h_s|x, h_{1:s-1}] = \Pr[h_s|x_{s-1}, h_{s-1}]$ and $\Pr[y_s|x, h, y_{1:s-1}] = \Pr[y_s|h_s]$. That is, given the input symbol at time $s-1$ and the hidden state at time $s-1$ the probability of the next state is independent of anything else in the sequence, and given the state at time s the probability of the corresponding output symbol is independent of anything else. We usually drop the subscript when the FST is obvious from the context.

Equation (1) shows that the conditional distribution defined by an FST $\mathbb{P}$ can be fully characterized using standard transition, initial and emission parameters, which we define as follows. For each symbol $a \in \mathcal{X}$, let $T_a \in \mathbb{R}^{m \times m}$ be the state transition probability matrix, where $T_a(i,j) = \Pr[H_s = c_i | X_{s-1} = a, H_{s-1} = c_j]$. Write $\alpha \in \mathbb{R}^m$ for the initial state distribution, and let $O \in \mathbb{R}^{l \times m}$ be the emission probability matrix where $O(i,j) = \Pr[Y_s = b_i | H_s = c_j]$. Given $b_i \in \mathcal{Y}$ we write D_{b_i} to denote an $m \times m$ diagonal matrix with the ith row of O as diagonal elements. Similarly, we write D_α for the $m \times m$ diagonal matrix with α as diagonal values.

To calculate probabilities of output sequences with FSTs we employ the notion of *observable operators*, which is commonly used for HMMs [22,4,11,14]. The following lemma shows how to express $\mathbb{P}(y|x)$ in terms of these quantities using the observable operator view of FSTs.

Lemma 1. *For each $a \in \mathcal{X}$ and $b \in \mathcal{Y}$ define $A_a^b = T_a\ D_b$. Then, the following holds:*

$$\mathbb{P}(y|x) = 1^\top\ A_{x_t}^{y_t}\ \cdots\ A_{x_1}^{y_1}\ \alpha\ . \qquad (2)$$

To understand the lemma, consider a state-distribution vector $\alpha_s \in \mathbb{R}^m$, where $\alpha_s(i) = \Pr[y_{1:s-1}, H_s = c_i | x_{1:s-1}]$. Initially, α_1 is set to α. Then $\alpha_{s+1} = A_{x_s}^{y_s}\ \alpha_s$ updates the state distribution from positions s to $s+1$ by applying the appropriate operator, i.e. by emitting symbol y_s and transitioning with respect to x_s. The lemma computes α_{t+1} by applying a chain of computations to the

[1] For convenience we assume that input and output sequences are of the same length. Later in the paper we overcome this limitation by introducing special *empty* symbols in the input and output alphabets.

sequence pair x and y. Then, the probability of y given x is given by $\sum_i \alpha_{t+1}(i)$. Matrices A_a^b are the *observable operators* that relate input-output observations with state dynamics.

Our learning algorithms will learn model parameterizations that are based on this observable operators view of FSTs. As input they will receive a sample of input-output pairs sampled from an input distribution $\mathbb{D}$ over $\mathcal{X}^*$; the joint distribution will be denoted by $\mathbb{D} \otimes \mathbb{P}$. In general, learning FSTs is known to be hard. Thus, our learning algorithms need to make some assumptions about the FST and the input distribution. Before stating them we introduce some notation. For any $a \in \mathcal{X}$ let $p_a = \Pr[X_1 = a]$, and define an "average" transition matrix $T = \sum_a p_a T_a$ for $\mathbb{P}$ and $\mu = \min_a p_a$, which characterizes the *spread* of $\mathbb{D}$.

Assumptions. An FST can be learned when $\mathbb{D}$ and $\mathbb{P}$ satisfy the following: (1) $l \geq m$, (2) D_α and O have rank m, (3) T has rank m, (4) $\mu > 0$.

Assumptions 1 and 2 on the nature of the target FST have counterparts in HMM learning. In particular, the assumption on D_α requires that no state has zero initial probability. Assumption 3 is an extension for FSTs of a similar condition for HMM, but in this case depends on $\mathbb{D}$ as well as on $\mathbb{P}$. Condition 4 on the input distribution ensures that all input symbols will be observed in a large sample. For more details about the implications of these assumptions, see [13].

3 A Spectral Learning Algorithm

In this section we present a learning algorithm for FSTs based on spectral decompositions. The algorithm will find a set of operators B for an FST which are equivalent to the operators A presented above, in the sense that they define the same distribution. In section 4 we will present a theoretical analysis for this algorithm, proving strong generalization guarantees when the assumptions described in the previous section are fulfilled.

In addition, we also present an algorithm to directly retrieve the observation, initial and transition probabilities. The algorithm is based on a joint decomposition method which, to the best of our knowledge, has never been applied to OOM learning before.

We start by defining probabilities over bigrams and trigrams of output symbols generated by an FST. Let $P \in \mathbb{R}^{l \times l}$ be a matrix of probabilities over bigrams of output symbols, where

$$P(i,j) = \Pr[Y_{1:2} = b_j b_i] \ . \qquad (3)$$

Furthermore, for each two input-output symbols $a \in \mathcal{X}$ and $b \in \mathcal{Y}$ we define a matrix $P_a^b \in \mathbb{R}^{l \times l}$ of marginal probabilities over output trigrams as follows:

$$P_a^b(i,j) = \Pr[Y_{1:3} = b_j b b_i | X_2 = a] \ . \qquad (4)$$

```
Algorithm LearnFST($\mathcal{X}, \mathcal{Y}, S, m$)

Input:
  – $\mathcal{X}$ and $\mathcal{Y}$ are input-output alphabets
  – $S = \{(x^1, y^1), \dots, (x^n, y^n)\}$ is a training set of input-output sequences
  – $m$ is the number of hidden states of the FST
Output:
  – Estimates of the observable parameters $\widehat{\beta}_1$, $\widehat{\beta}_\infty$ and $\widehat{B}_a^b$ for all $a \in \mathcal{X}$ and $b \in \mathcal{Y}$

  1. Use $S$ to compute an empirical estimate of the probability matrices $\widehat{\rho}$, $\widehat{P}$, and
     $\widehat{P}_a^b$ for each pair of input-output symbols $a \in \mathcal{X}$ and $b \in \mathcal{Y}$
  2. Take $\widehat{U}$ to be the matrix of top $m$ left singular vectors of $\widehat{P}$
  3. Compute the observable FST parameters as: $\widehat{\beta}_1 = \widehat{U}^\top \widehat{\rho}$, $\widehat{\beta}_\infty^\top = \widehat{\rho}^\top (\widehat{U}^\top \widehat{P})^+$
     and $\widehat{B}_a^b = (\widehat{U}^\top \widehat{P}_a^b)(\widehat{U}^\top \widehat{P})^+$ for each $a \in \mathcal{X}$ and $b \in \mathcal{Y}$
```

Fig. 1. An algorithm for learning FST.

Some algebraic manipulations show the following equivalences (recall that $T = \sum_a T_a \Pr[X_1 = a]$):

$$P = O\ T\ D_\alpha\ O^\top \ , \tag{5}$$

$$P_a^b = O\ T_a\ D_b\ T\ D_\alpha\ O^\top \ . \tag{6}$$

Note that if Assumptions 1–3 are satisfied, then P has rank m. In this case we can perform an SVD on $P = U \Sigma V^*$ and take $U \in \mathbb{R}^{l \times m}$ to contain its top m left singular vectors. It is shown in [13] that under these conditions the matrix $U^\top O$ is invertible. Finally, let $\rho \in \mathbb{R}^l$ be the initial symbol probabilities, where $\rho(i) = \Pr[Y_1 = b_i]$.

Estimations of all these matrices can be efficiently computed from a sample obtained from $\mathbb{D} \otimes \mathbb{P}$. Now we use them to define the following *observable representation* for $\mathbb{P}$:

$$\beta_1 = U^\top \rho \ , \tag{7}$$

$$\beta_\infty^\top = \rho^\top (U^\top P)^+ \ , \tag{8}$$

$$B_a^b = (U^\top P_a^b)(U^\top P)^+ \ . \tag{9}$$

Next lemma shows how to compute FST probabilities using these new observable operators. Fig. 1 presents `LearnFST`, an algorithm that estimates the operators.

Lemma 2 (Observable FST representation). *Assume $\mathbb{D}$ and $\mathbb{P}$ obey Assumptions 1–3. For any $a \in \mathcal{X}$, $b \in \mathcal{Y}$, $x \in \mathcal{X}^t$ and $y \in \mathcal{Y}^t$, the following hold.*

$$\beta_1 = (U^\top O)\alpha \ , \tag{10}$$

$$\beta_\infty^\top = 1^\top (U^\top O)^{-1} \ , \tag{11}$$

$$B_a^b = (U^\top O) A_a^b (U^\top O)^{-1} \ , \tag{12}$$

$$\mathbb{P}(y|x) = \beta_\infty^\top B_{x_t}^{y_t} \ \cdots \ B_{x_1}^{y_1} \beta_1 \ . \tag{13}$$

The proof is analogous to that of Lemma 3 of [13]. We omit it for brevity.

3.1 Recovering the original FST parameters

We now describe an algorithm for recovering the standard FST parameters, namely O, α and T_a for $a \in \mathcal{X}$. Though this is not necessary for computing sequence probabilities, it may be an appealing approach for applications that require computing quantities which are not readily available from the observable representation, e.g. state marginal probabilities.

Similar to before, we will define some probability matrices. Let $P_3^b \in \mathbb{R}^{l \times l}$ be a matrix of probabilities over output trigrams, where

$$P_3^b(i,j) = \Pr[Y_{1:3} = b_j b b_i] \ . \tag{14}$$

Let $P_3 \in \mathbb{R}^{l \times l}$ account for probabilities of output trigrams, marginalizing the middle symbol,

$$P_3(i,j) = \Pr[Y_1 = b_j, Y_3 = b_i] \ . \tag{15}$$

This matrix can be expressed as

$$P_3 = \sum_a \sum_b P_a^b \Pr[X_2 = a] \ . \tag{16}$$

Finally, let $P_a \in \mathbb{R}^{l \times l}$ be probabilities of output bigrams, where

$$P_a(i,j) = \Pr[Y_{1:2} = b_j b_i | X_1 = a] \ . \tag{17}$$

Now, for every $b \in \mathcal{Y}$ define $Q^b = P_3^b P_3^+$. Writing $T_2 = \sum_a T_a \Pr[X_2 = a]$, one can see that

$$Q^b = (OT_2) D_b (OT_2)^+ \ . \tag{18}$$

The equation above is an eigenvalue-eigenvector decomposition of the matrices Q^b. These matrices allow for a *joint eigen-decomposition* where the eigenvalues of Q^b correspond to the row of O associated with b.

Our algorithm first computes empirical estimates $\widehat{\rho}$, $\widehat{P}_3^b$, $\widehat{P}_3$ and $\widehat{P}_a$, and builds $\widehat{Q}^b$. Then it performs a Joint Schur Decomposition of the matrices $\widehat{Q}^b$ to retrieve the joint eigenvalues and compute $\widehat{O}$. We use the optimization algorithm from [12] to perform the joint Schur decomposition. Finally, estimates of transition matrices for all $a \in \mathcal{X}$ and initial state probabilities are obtained as:

$$\widehat{\alpha} = \widehat{O}^+ \ \widehat{\rho} \ , \tag{19}$$

$$\widehat{T}_a = \widehat{O}^+ \ \widehat{P}_a (D_{\widehat{\alpha}} \ \widehat{O})^+ \ . \tag{20}$$

The correctness of these expressions in the error-free case can be easily verified.

Though this method is provided without an error analysis, some experiments in Section 5 demonstrate that in some cases the parameters recovered with this algorithm can approximate the target FST better than the observable representation obtained with `LearnFST`.

Note that this method is different from those presented in [18,13] for recovering parameters of HMMs. Essentially, their approach requires to find a set of eigenvectors, while our method recovers a set of joint eigenvalues. Furthermore, our method could also be used to recover parameters from HMMs.

4 Theoretical Analysis

In this section the algorithm `LearnFST` is analyzed. We show that, under some assumptions on the target FST and the input distribution, it will output a good hypothesis with high probability whenever the sample is large enough. First we discuss the learning model, then we state our main theorem, and finally we sketch the proof. Our proof schema follows closely that of [13]; therefore, only the key differences with their proof will be described, and, in particular, the lemmas which are stated without proof can be obtained by mimicking their techniques.

4.1 Learning Model

Our learning model resembles that in [1] for learning stochastic rules, but uses a different loss function. As in the well-known PAC model, we have access to examples (x, y) drawn i.i.d. from $\mathbb{D}\otimes\mathbb{P}$. The difference with concept learning is that now, instead of a *deterministic* rule, the learning algorithm outputs a *stochastic* rule modelling a conditional distribution $\widehat{\mathbb{P}}$ that given an input sequence x can be used to predict an output sequence y. As in all models that learn input-output relations, the accuracy of the hypothesis is measured relatively to the same input distribution that was used to generate the training sample. In our case, we are interested in minimizing

$$d_{\mathbb{D}}(\mathbb{P}, \widehat{\mathbb{P}}) = \mathrm{E}_{X\sim\mathbb{D}}\left[\sum_y |\mathbb{P}(y|X) - \widehat{\mathbb{P}}(y|X)|\right] \ . \tag{21}$$

This loss function corresponds to the L_1 distance between $\mathbb{D}\otimes\mathbb{P}$ and $\mathbb{D}\otimes\widehat{\mathbb{P}}$.

4.2 Results

Our learning algorithm will be shown to work whenever $\mathbb{D}$ and $\mathbb{P}$ satisfy Assumptions 1–4. In particular, note that 2 and 3 imply that the mth singular values of O and P, respectively σ_O and σ_P, are positive.

We proceed to state our main theorem. There, instead of restricting ourselves to input-output sequences of some fixed length t, we consider the more general, practically relevant case where $\mathbb{D}$ is a distribution over $\mathcal{X}^*$. In this case the bound depends on $\lambda = \mathrm{E}_{X\sim\mathbb{D}}[|X|]$, the expected length of input sequences.

Theorem 1. *For any $0 < \epsilon, \delta < 1$, if $\mathbb{D}$ and $\mathbb{P}$ satisfy Assumptions 1–4, and `LearnFST` receives as input m and a sample with $n \geq N$ examples for some N in*

$$O\left(\frac{\lambda^2 m l}{\epsilon^4 \mu \sigma_O^2 \sigma_P^4} \log\frac{k}{\delta}\right) \ , \tag{22}$$

then, with probability at least $1-\delta$, the hypothesis $\widehat{\mathbb{P}}$ returned by the algorithm satisfies $d_{\mathbb{D}}(\mathbb{P}, \widehat{\mathbb{P}}) \leq \epsilon$.

Note that function N in Theorem 1 depends on $\mathbb{D}$ through λ, μ and σ_P — this situation is quite different from the setting found in HMMs. The price we pay for choosing a setting with input strings of arbitrary length is a dependence of type $O(\lambda^2/\epsilon^4)$ in the bound. A dependence of type $O(t^2/\epsilon^2)$ can be obtained, using similar techniques, in the setting where input strings have fixed length t. However, we believe the latter setting to be *less realistic* for practical applications. Furthermore, a better dependence on ϵ can be proved for the following particular case of practical interest.

Recall that if $X \sim \mathbb{D}$ is modeled by an HMM with an absorbing state — equivalently, an HMM with stopping probabilities — the random variable $|X|$ follows a phase-type (PH) distribution [19]. It is well known that after a transient period, these distributions present an exponential rate of decay. In particular, for any such $\mathbb{D}$ there exist positive constants τ_1, τ_2 such that if $t \geq \tau_1$, then $\Pr[|X| \geq t] = O(e^{-t/\tau_2})$. In this case our techniques yield a bound of type $O(\tau_1^2 \tau_2^2/\epsilon^2 \log(1/\epsilon))$. In many practical problems it is not uncommon to assume that input sequences follow some kind of markovian process; this alternative bound can be applied in such cases.

Though it will not be discussed in detail here due to space reasons, the dependence on l in Equation 22 can be relaxed to take into account only the most probable symbols; this is useful when $\mathbb{D}$ exhibits a power law behavior. Furthermore, our algorithm provably works (cf. [13]) with similar guarantees in the agnostic setting where $\mathbb{P}$ cannot be *exactly* modelled with an FST, but is *close* to some FST satisfying Assumptions 1–3. Finally, in application domains where k is large, there may be input symbols with very low probability that are not observed in a sample. For these cases, we believe that it may be possible to soften the (implicit) dependence of N on k through $1/\mu$ using smoothing techniques. Smoothing procedures have been used in practice to solve these issues in many related problems [7]; theoretical analyses have also proved the validity of this approach [21].

We want to stress here that our result goes beyond a naive application of the result by Hsu et al. [13] to FST learning. One could try to learn an HMM modeling the joint distribution $\mathbb{D} \otimes \mathbb{P}$, but their result would require that this distribution can be modeled by some HMM; we do not need this assumption in our result. Another approach would be to learn k distinct HMMs, one for each input symbol; this approach would miss the fact that the operator O is the same in all these HMMs, while our method is able to exploit this fact to its advantage by using the same $\widehat{U}$ for every operator. In Section 5 we compare our algorithm against these two baselines and show that it behaves better in practice.

4.3 Proofs

The main technical difference between algorithm `LearnFST` and spectral techniques for learning HMM is that in our case the operators B_a^b depend on the *input* symbol as well as the output symbol. This implies that estimation errors of B_a^b for different input symbols will depend on the input distribution; the occurrence of μ in Equation 22 accounts for this fact.

First we introduce some notation. We will use $\|\cdot\|_p$ to denote the usual ℓ_p norms for vectors and the corresponding induced norms for matrices, and $\|\cdot\|_F$ will be used to denote the Frobenius norm. Given n examples $(x^1, y^1), \ldots, (x^n, y^n)$ drawn i.i.d. from $\mathbb{D}\otimes\mathbb{P}$, we denote by n_a the number of samples such that $x_2 = a$, which measures how well $\widehat{P}_a^b$ is estimated. Also define the following estimation errors: $\epsilon_\rho = \|\rho - \widehat{\rho}\|_2$, $\epsilon_P = \|P - \widehat{P}\|_F$ and $\epsilon_a = \sum_b \|P_a^b - \widehat{P}_a^b\|_F$. The first lemma bounds these errors in terms of the sample size. The results follow from a simple analysis using Chernoff bounds and McDiarmid's inequality.

Lemma 3. *With probability at least $1-\delta$, the following hold simultaneously:*

$$\epsilon_\rho \leq \sqrt{1/n}\left(1 + \sqrt{\log(4/\delta)}\right) \ , \tag{23}$$

$$\epsilon_P \leq \sqrt{1/n}\left(1 + \sqrt{\log(4/\delta)}\right) \ , \tag{24}$$

$$\forall a \ \epsilon_a \leq \sqrt{l/n_a}\left(1 + \sqrt{\log(4/\delta)}\right) \ , \tag{25}$$

$$\forall a \ n_a \geq np_a - \sqrt{2np_a \log(4k/\delta)} \ . \tag{26}$$

Next lemma is almost identical to Lemma 10 in [13], and is repeated here for completeness. Both this and the following one require that $\mathbb{D}$ and $\mathbb{P}$ satisfy Assumptions 1–3 described above. These three quantities are used in both statements:

$$\widetilde{\epsilon}_1 = \|(\widehat{U}^\top O)^{-1}(\widehat{\beta}_1 - \widetilde{\beta}_1)\|_1 \ , \tag{27}$$

$$\widetilde{\epsilon}_\infty = \|(\widehat{U}^\top O)^\top(\widehat{\beta}_\infty - \widetilde{\beta}_\infty)\|_\infty \ , \tag{28}$$

$$\widetilde{\epsilon}_a = \sum_b \|(\widehat{U}^\top O)^{-1}(\widehat{B}_a^b - \widetilde{B}_a^b)(\widehat{U}^\top O)\|_1 \ . \tag{29}$$

Here, the definitions of $\widetilde{\beta}_1$, $\widetilde{\beta}_\infty$ and $\widetilde{B}_a^b$ correspond to substituting U by $\widehat{U}$ in the expressions for β_1, β_∞ and B_a^b respectively.

Lemma 4. *If $\epsilon_P \leq \sigma_P/3$, then*

$$\widetilde{\epsilon}_1 \leq (2/\sqrt{3})\epsilon_\rho\sqrt{m}/\sigma_O \ , \tag{30}$$

$$\widetilde{\epsilon}_\infty \leq 4\left(\epsilon_P/\sigma_P^2 + \epsilon_\rho/(3\sigma_P)\right) \ , \tag{31}$$

$$\widetilde{\epsilon}_a \leq (8/\sqrt{3})\sqrt{m}/\sigma_O\left(\epsilon_P/\sigma_P^2 + \epsilon_a/(3\sigma_P)\right) \ . \tag{32}$$

The lemma follows from a perturbation analysis on the singular values of $\widehat{U}^\top O$, $\widehat{U}^\top P$ and $\widehat{U}^\top \widehat{P}$. In particular, the condition on ϵ_P ensures that $\widehat{U}^\top O$ is invertible.

Our next lemma gives two inequalities useful for bounding the error between $\mathbb{P}$ and the output from `LearnFST`. The proof extends that of Lemmas 11 and 12 from [13] and is omitted in this version; the main difference is that now bounds depend on the input sequence. Note that the second inequality is a consequence of the first one.

Lemma 5. *For all $x \in \mathcal{X}^t$, let $\varepsilon_x = \prod_{s=1}^{t}(1 + \widetilde{\epsilon}_{x_s})$. The following hold:*

$$\sum_{y \in \mathcal{Y}^t} \|(\widehat{U}^\top O)^{-1}(\widehat{B}_x^y \widehat{\beta}_\infty - \widetilde{B}_x^y \widetilde{\beta}_\infty)\|_1 \leq (1 + \widetilde{\epsilon}_1)\varepsilon_x - 1 \ , \tag{33}$$

$$\sum_{y \in \mathcal{Y}^t} |\mathbb{P}(y|x) - \widehat{\mathbb{P}}(y|x)| \leq (1 + \widetilde{\epsilon}_1)(1 + \widetilde{\epsilon}_\infty)\varepsilon_x - 1 \ . \tag{34}$$

Now we proceed to prove our main theorem.

Proof (Proof of Theorem 1). First note that by the assumptions on $\mathbb{D}$ and $\mathbb{P}$ all the above lemmas can be used. In particular, by Lemmas 3 and 4 we have that, for some constants $c_1, c_2, c_3, c_4 > 0$, the following hold simultaneously with probability $1 - \delta$:

1. $n \geq c_1/\sigma_P^2 \log(1/\delta)$ implies $\epsilon_P \leq \sigma_P/3$,
2. $n \geq c_2 m/(\epsilon^2 \sigma_O^2) \log(1/\delta)$ implies $\widetilde{\epsilon}_1 \leq \epsilon/40$,
3. $n \geq c_3/(\epsilon^2 \sigma_P^4) \log(1/\delta)$ implies $\widetilde{\epsilon}_\infty \leq \epsilon/40$, and
4. $n \geq c_4 \lambda^2 l m/(\epsilon^4 \mu \sigma_O^2 \sigma_P^2) \log(k/\delta)$ implies $\forall a \ \widetilde{\epsilon}_a \leq \epsilon^2/(20\lambda)$.

Item 4 above uses the fact that $u - \sqrt{cu} \geq u/2$ for $u \geq 4c$. Finally, we use that $(1 + u/t)^t \leq 1 + 2u$ for all $t \geq 0$ and $u \leq 1/2$ to obtain the bound:

$$d_{\mathbb{D}}(\mathbb{P}, \widehat{\mathbb{P}}) \leq \sum_{|x| < 4\lambda/\epsilon} \mathbb{D}(x) \sum_{y \in \mathcal{Y}^{|x|}} |\mathbb{P}(y|x) - \widehat{\mathbb{P}}(y|x)| + \sum_{|x| \geq 4\lambda/\epsilon} 2\,\mathbb{D}(x) \leq \epsilon \ , \tag{35}$$

where the first term is at most $\epsilon/2$ by Lemma 5, and the second is bounded using Markov's inequality. □

5 Synthetic Experiments

In this section we present experiments using our FST learning algorithm with synthetic data. We are interested in four different aspects. First, we want to evaluate how the estimation error of the learning algorithm behaves as we increase the training set size and the difficulty of the target. Second, how the estimation error degrades with the length of test sequences. In the third place, we want to compare our algorithms with other, more naive, spectral methods for learning FST. And four, we compare `LearnFST` with our other algorithm for recovering the parameters of an FST using a joint Schur decomposition.

For our first experiment, we generated synthetic data of increasing difficulty as predicted by our analysis, as follows. First, we randomly selected a distribution over input sequences of length three, for input alphabet sizes ranging from 2 to 10, and choosing among uniform, gaussian and power distributions with random parameters. Second, we randomly selected an FST, choosing from output alphabet sizes from 2 to 10, choosing a number of hidden states and randomly generating initial, transition and observation parameters. For a choice of input distribution and FST, we computed the quantities appearing in the bound except for the logarithmic term, and defined $c = (\lambda^2 m l)/(\mu \sigma_O^2 \sigma_P^4)$. According to

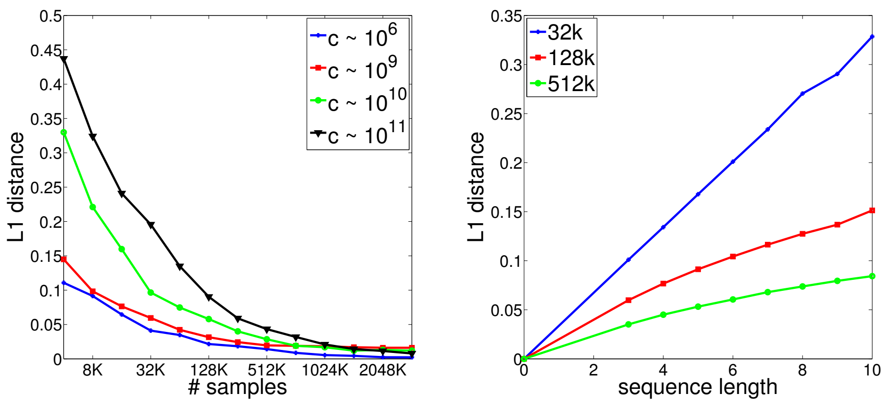

Fig. 2. *(Left)* Learning curves for models at increasing difficulties, as predicted by our analysis. *(Right)* L_1 distance with respect to the length of test sequences, for models trained with 32K, 128K and 512K training examples ($k = 3$, $l = 3$, $m = 2$).

our analysis, the quantity c is an estimate of the difficulty of learning the FST. In this experiment we considered four random models, that fall into different orders of c. For each model, we generated training sets of different sizes, by sampling from the corresponding distribution. Figure 2 (left) plots $d_{\mathbb{D}}(\mathbb{P}, \widehat{\mathbb{P}})$ as a function of the training set size, where each curve is an average of 10 runs. The curves follow the behavior predicted by the analysis.

The results from our second experiment can be seen in Figure 2 (right), which plots the error of learning a given model (with $k = 3$, $l = 3$ and $m = 2$) as a function of the test sequence lengths t, for three training set sizes. The plot shows that increasing the number of training samples has a clear impact in the performance of the model on longer sequences. It can also be seen that, as we increase the number of training samples, the curve seems to flatten faster, i.e. the growth rate of the error with the sequence length decreases nicely.

In the third experiment we compared `LearnFST` to another two *baseline* spectral algorithms. These baselines are naive applications of the algorithm by Hsu et al. [13] to the problem of FST learning. The first baseline (HMM) learns an HMM that models the joint distribution $\mathbb{D} \otimes \mathbb{P}$. The second baseline ($k$-HMM) learns k different HMMs, one for each input symbol. This correponds to learning an operator B_a^b for each pair $(a, b) \in \mathcal{X} \times \mathcal{Y}$ using *only* the observations where $X_2 = a$ and $Y_2 = b$, ignoring the fact that one can use the same U, computed with all samples, for every operator B_a^b. In this experiment, we randomly created an input distribution $\mathbb{D}$ and a target FST $\mathbb{P}$ (using $k = 3, l = 3, m = 2$). Then we randomly sampled training sequence pairs from $\mathbb{D} \otimes \mathbb{P}$, and trained models using the three spectral algorithms. To evaluate the performance we measured the L_1 distance on all sequence pairs of length 3. Figure 3 (left) plots learning curves resulting from averaging performance across 5 random runs of the experiment. It can be seen that with enough examples the baseline algorithms

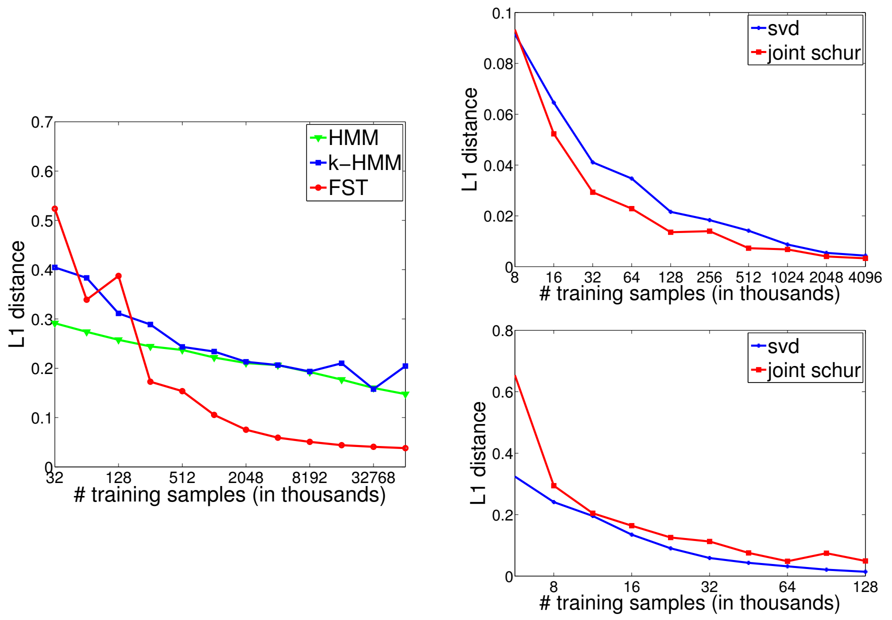

Fig. 3. *(Left)* Comparison with spectral baselines. *(Right)* Comparison with joint decomposition method

are outperformed by our method. Furthermore, the fact that the joint HMM outperforms the conditional FST with small sample sizes is consistent with the well-known phenomena in classification where generative models can outperform discriminative models with small sample sizes [20].

Our last experiment's goal is to showcase the behavior of the algorithm presented in Section 3.1 for recovering the parameters of an FST using a joint Schur decomposition. Though we do not have a theoretical analysis of this algorithm, several experiments indicate that its behavior tends to depend more on the particular model than that of the rest of spectral methods. In particular, in many models we observe an asymptotic behavior similar to the one presented by `LearnFST`, and for some of them we observe better absolute performance. Two examples of this can be found in Figures 3 (right), where the accuracy versus the number of examples is plotted for two different, randomly selected models (with $k = 3$, $l = 3$, $m = 2$).

6 Experiments on Transliteration

In this section we present experiments on a real task in Natural Language Processing, machine transliteration. The problem consists of mapping named entities (e.g. person names, locations, etc.) between languages that have different alphabets and sound systems, by producing a string in the target language that is

Table 1. Properties of the transliteration dataset. "length ratio" is the average ratio between lengths of input and output training sequences. "equal length" is the percentage of training sequence pairs of equal length.

number of training sequences	6,000	average length x	7.84
number of test sequences	943	average length y	8.20
size of $\mathcal{X}$	82	length ratio	0.959
size of $\mathcal{Y}$	34	equal length	53.42%

phonetically equivalent to the string in the source language. For example, the English word "brooklyn" is transliterated into Russian as "бруклин". Because orthographic and phonetic systems across languages differ, the lengths of paired strings also differ in general. The goal of this experiment is to test the performance of our learning algorithm in real data, and to compare it with a standard EM algorithm for training FSTs.

We considered the English to Russian transliteration task of the News shared task [17]. Training and test data consists of pairs of strings. Table 1 gives additional details on the dataset.

A standard metric to evaluate the accuracy of a transliteration system is the normalized edit distance (NED) between the correct and predicted transliterations. It counts the minimum number of character deletions, insertions and substitutions that need to be made to transform the predicted string into the correct one, divided by the length of the correct string and multiplied by 100.

In order to apply FSTs to this task we need to handle sequence pairs of unequal lengths. Following the classic work on transliteration by Knight and Graehl [16] we introduced special symbols in the output alphabet which account for an empty emission and every combination of two output symbols; thus, our FSTs can map an input character to zero, one or two output characters. However, the correct character alignments are not known. To account for this, for every training pair we considered all possible alignments as having equal probability.[2] It is easy to adjust our learning algorithm such that when computing the probability estimates (step 1 in the algorithm of Figure 1) we consider a distribution over alignments between training pairs. This can be done efficiently with a simple extension to the classic dynamic programming algorithm for computing edit distances.

At test, predicting the best output sequence (summing over all hidden sequences) is not tractable. We resorted to the standard approach of sampling, where we used the FST to compute conditional estimates of the next output symbol (see [13] for details on these computations).

[2] The alignments between sequences are a missing part in the training data, and learning such alignments is in fact an important problem in FST learning (e.g., see [16]). However, note that our focus is not on learning alignments, but instead on learning non-deterministic transductions between aligned sequences. In practice, our algorithm could be used with an iterative EM method to learn both alignment distributions and hidden states, and we believe future work should explore this line.

Table 2. Normalized edit distance at test (NED) of a model as a function of the number of hidden states (m), using all training samples. σ is the mth singular value of $\hat{P}$.

m	1	2	3	4	5
σ	0.0929	0.0914	0.0327	0.0241	0.0088
NED	21.769	21.189	21.224	26.227	71.780

Table 3. Running times (averaged, in seconds) of a *single* EM iteration, for different number of training pairs and two different values for m. The number in parenthesis is the number of iterations it takes to reach the best test performance (see Figure 4).

	75	350	750	1500	3000	6000
$m = 2$	1.15 (120)	3.53 (70)	6.73 (140)	10.60 (120)	19.8 (50)	37.74 (40)
$m = 3$	1.16 (50)	3.55 (80)	6.75 (40)	10.62 (180)	19.9 (180)	37.78 (30)

The only free parameter of the FST is the number of hidden states (m). There is a trade-off between increasing the number of hidden states, yielding lower approximation error, and increasing the estimation error. In particular, our analysis states that the estimation error depends on the mth singular value of $\hat{P}$. Table 2 illustrates this trade-off. Clearly, the singular values are a good indicator of the proper range for m.

We also trained FST models using EM, for different number of hidden states. We tried multiple random initializations and ran EM for a sufficiently large number of iterations (200 in our experiments). We evaluated each EM model at every 10 iterations on the test, and chose the best test performance.

Figure 4 shows the best learning curves using the spectral algorithm and EM, for $m = 2$ and $m = 3$. The performance of the spectral method is similar to that of EM for large training sizes while for smaller training sizes EM seems to be unable to find a good model. Our experience at running EM was that for large training sizes, the performance of different runs was similar, while for small training sets the error rates of different runs had a large variance. The spectral method seems to be very stable at finding good solutions.

We also compared the two learning methods in terms of computation time.[3]. Table 3 shows the time it takes to complete an iteration under EM, together with the number of iterations it takes to reach the best error rates at tests. In comparison, the spectral method takes about 13 seconds to compute the statistics on the larger training set (step 1 on algorithm 1) and about 13 seconds to perform the SVD and compute the operators (steps 2 and 3 on algorithm 1) which gives a total of 26 seconds for the largest setting.

Comparing to state-of-the-art on the NEWS data, our model obtains 87% on the F1 metric, while the range of performances goes from 86.5% to 93% [17][4].

[3] We used Matlab on an Intel Xeon 2.40GHz machine with 12Gb of RAM running Linux.

[4] Normalized edit distance was not measured in the News'09 Shared Task.

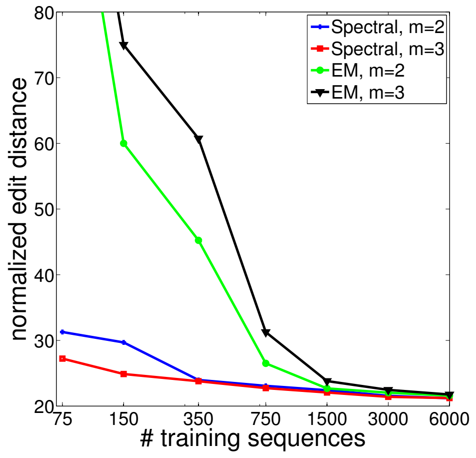

Fig. 4. Learning curves for transliteration experiments using the spectral algorithm and EM, for different number of hidden states. Error is measured as Normalized Edit Distance.

It should be noted that transliteration systems exploit combinations of several models optimized for the task. In contrast, we use out-of-the-box FSTs.

7 Conclusions

In this paper we presented a spectral learning algorithm for probabilistic non-deterministic FSTs. The main result are strong PAC-style guarantees, which, to our knowledge, are the first for FST learning. Furthermore, we present extensive experiments demonstrating the effectiveness of the proposed method in practice, when learning from synthetic and real data.

An attractive property of our algorithm is its speed and scalability at training. Experiments on a transliteration task show that, in practice, it is an effective algorithm for learning FSTs. Our models could be used as building blocks to solve complex tasks, such as parsing and translation of natural languages, and planning in reinforcement learning.

Future work should improve the behavior of our algorithm in large input alphabets by means of smoothing procedures. In practice, this should improve the robustness of the method and make it applicable to a wider set of tasks. Other lines of future research include: conducting a theoretical analysis of the joint Schur approach for recovering parameters of HMM and FST, and exploring the power of our algorithm for learning more general families of transductions.

References

1. Abe, N., Takeuchi, J., Warmuth, M.: Polynomial Learnability of Stochastic Rules with Respect to the KL-Divergence and Quadratic Distance. IEICE Transactions

on Information and Systems 84(3), 299–316 (2001)
2. Bailly, R., Denis, F., Ralaivola, L.: Grammatical inference as a principal component analysis problem. In: Proc. ICML (2009)
3. Bernard, M., Janodet, J.C., Sebban, M.: A discriminative model of stochastic edit distance in the form of a conditional transducer. Grammatical Inference: Algorithms and Applications 4201 (2006)
4. Carlyle, J., Paz, A.: Realization by stochastic finite automaton. Journal of Computer and System Sciences 5, 26–40 (1971)
5. Casacuberta, F.: Inference of finite-state transducers by using regular grammars and morphisms. Grammatical Inference: Algorithms and Applications 1891 (2000)
6. Chang, J.T.: Full reconstruction of markov models on evolutionary trees: Identifiability and consistency. Mathematical Biosciences 137, 51–73 (1996)
7. Chen, S., Goodman, J.: An empirical study of smoothing techniques for language modeling. In: Proc. of ACL. pp. 310–318 (1996)
8. Clark, A.: Partially supervised learning of morphology with stochastic transducers. In: Proc. of NLPRS. pp. 341–348 (2001)
9. Clark, A., Costa Florêncio, C., Watkins, C.: Languages as hyperplanes: grammatical inference with string kernels. Machine Learning pp. 1–23 (2010)
10. Eisner, J.: Parameter estimation for probabilistic finite-state transducers. In: Proc. of ACL. pp. 1–8 (2002)
11. Fliess, M.: Matrices de Hankel. Journal de Mathematiques Pures et Appliquees 53, 197–222 (1974)
12. Haardt, M., Nossek, J.A.: Simultaneous schur decomposition of several nonsymmetric matrices to achieve automatic pairing in multidimensional harmonic retrieval problems. IEEE Transactions on Signal Processing 46(1) (1998)
13. Hsu, D., Kakade, S.M., Zhang, T.: A spectral algorithm for learning hidden markov models. In: Proc. of COLT (2009)
14. Jaeger, H.: Observable operator models for discrete stochastic time series. Neural Computation 12, 1371–1398 (2000)
15. Jelinek, F.: Statistical Methods for Speech Recognition (Language, Speech, and Communication). MIT Press (1998)
16. Knight, K., Graehl, J.: Machine transliteration. Computational Linguistics 24(4), 599–612 (1998)
17. Li, H., Kumaran, A., Pervouchine, V., Zhang, M.: Report of news 2009 machine transliteration shared task. In: Proc. Named Entities Workshop (2009)
18. Mossel, E., Roch, S.: Learning nonsingular phylogenies and hidden markov models. In: Proc. of STOC (2005)
19. Neuts, M.F.: Matrix-geometric solutions in stochastic models : an algorithmic approach. Johns Hopkins University Press (1981)
20. Ng, A., Jordan, M.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: NIPS (2002)
21. Ron, D., Singer, Y., Tishby, N.: On the learnability and usage of acyclic probabilistic finite automata. In: Proc. of COLT. pp. 31–40 (1995)
22. Schützenberger, M.: On the definition of a family of automata. Information and Control 4, 245–270 (1961)
23. Siddiqi, S.M., Boots, B., Gordon, G.J.: Reduced-Rank Hidden Markov Models. In: Proc. AISTATS. pp. 741–748 (2010)