Spectral Learning of Sequence Taggers Over Continuous Sequences *

Adrià Recasens and Ariadna Quattoni

Universitat Politècnica de Catalunya
Jordi Girona 1–3, 08034 Barcelona
arecasens@gmail.com, aquattoni@lsi.upc.edu

Abstract. In this paper we present a spectral algorithm for learning weighted finite-state sequence taggers (WFSTs) over paired input-output sequences, where the input is continuous and the output discrete. WFSTs are an important tool for modelling paired input-output sequences and have numerous applications in real-world problems. Our approach is based on generalizing the class of weighted finite-state sequence taggers over discrete input-output sequences to a class where transitions are linear combinations of elementary transitions and the weights of the linear combination are determined by dynamic features of the continuous input sequence. The resulting learning algorithm is efficient and accurate.

1 Introduction

Weighted Finite-state Sequence Taggers (WFSTs) are an important tool for modelling paired input-output sequences and have found numerous applications in areas such as natural language processing and computational biology (e.g. part of speech tagging, NP-chunking, entity recognition and protein folding prediction, to mention a few). The problem of modelling paired input-output sequences is usually referred in the literature as sequence tagging. In sequence tagging the goal is to predict a tag (i.e. a discrete output) for each symbol in the input sequence. And thus different from the general transduction problem where input and output sequences might be of different lengths, in the sequence tagging problem both input and output sequences are 'aligned' and have the same length. Still the problem of learning sequence taggers with latent states remains a challenging task.

The most popular methods for learning sequence taggers with hidden states are based on gradient-based or EM optimizations [12, 13], but these can be computationally expensive and are susceptible to local optima issues. Recently, an emerging line of work on spectral methods has proposed algorithms for latent-variable structure modelling that overcome some of the limitations of EM [11, 16, 21, 4, 15, 24, 3, 8, 6, 2]. Of these, [5] proposed a spectral method for learning

* This work has been partially funded by: the European Commission for project XLike (FP7-288342); the Spanish Government for project BASMATI (TIN2011-27479-C04-03); and the ERA-Net CHISTERA project VISEN.

WFSTs that learns distributions where both inputs and outputs are sequences from a discrete alphabet.

However, many real world problems require tagging sequences where the inputs are not discrete but continuous sequences. For example [22, 28, 20] consider the problem of human gesture or action recognition where given a video sequence the task is to predict the gesture that is been performed at each frame. Clearly, this can be framed as a sequence tagging problem where the continuous inputs correspond to real-valued features of the video sequence and the discrete outputs correspond to the gestures been performed at each point in time.

In this paper we extend the previous line of work on spectral learning for continuous sequences to handle the task of tagging sequences of continuous inputs. Our approach is based on generalizing the class of weighted finite-state sequence taggers over discrete input-output sequences to a class where transitions are linear combinations of elementary transitions and the weights of the linear combinations are determined by dynamic features of the continuous input sequence. One intuitive way to understand our approach is to think that we are learning a basis of the vectorial space of transition functions and that the weights of the linear combination depend only on the given features of the continuous input sequence.

Similar to [19, 6], we develop a spectral method for our model from forward-backward recursions which are used to derive useful matrix decompositions of observable statistics. These matrix decompositions are then in turn exploited to induce the relationships between observations and latent state dynamics. As with previous spectral methods our algorithm for learning finite-state sequence taggers is simple and efficient. It reduces to estimating simple statistics from samples of paired input-output sequences, performing a singular value decomposition and inversion of some matrices.

In summary the main contributions of this paper are: (1) We present a latent state model for sequence tagging over continuous inputs; (2) We derive an efficient spectral learning algorithm for this model from forward-backward recursions; and (3) We present experiments on a real-task that validate the effectiveness of our approach.

2 Models for Sequences of Continuous Inputs and Discrete Outputs

2.1 Preliminary: Weighted Finite-state Sequence Taggers

We start by defining a class of functions over pairs of discrete sequences. More specifically, let $x = x_1, \ldots, x_T$ be an input sequence and $y = y_1, \ldots, y_T$ be an output sequence, where $x \in \Delta^*$ and $y \in \Sigma^*$. Here both Δ and Σ are assumed to be discrete alphabets. We follow [5] in that we assume that x and y have the same length (i.e we model aligned sequences). Defining spectral learning algorithms over pairs of sequences of different lengths would require handling unobserved alignments which is outside the scope of this paper.

A weighted finite-state sequence tagger (WFST) over $\Delta \times \Sigma$ with m states can be defined as a tuple $A = \langle \alpha_1, \alpha_\infty, A^\sigma_\delta \rangle$ where α_1, $\alpha_\infty \in \mathbb{R}^m$ are the initial and final weight vectors and $A^\sigma_\delta \in \mathbb{R}^{m \times m}$ are the $|\Delta \times \Sigma|$ transition matrices associated to each pair of symbols $\langle \delta, \sigma \rangle \in \Delta \times \Sigma$. The function f_A realized by a WFST is defined as:

$$f_A(x, y) = \alpha_1^\top A^{y_1}_{x_1} \cdots A^{y_T}_{x_T} \alpha_\infty \quad . \tag{1}$$

The above equation is an algebraic representation of the computation performed by a WFST on a pair of sequences $\langle x, y \rangle$. To see this consider a state vector $s_t \in \mathbb{R}^m$ where the ith entry represents the sum of the weights of all the state paths that generate the prefix $\langle x_{1:t}, y_{1:t} \rangle$ and end in state i. Initially, $s_1 = \alpha_1$, and then $s^\top_{t+1} = s^\top_t A^{y_t}_{x_t}$ updates the state distribution by simultaneously emiting the symbol $\langle x_t, y_t \rangle$ and transitioning to the next state vector.

Notice that since x and y are aligned sequences we could regard a WFST as a weighted finite-state automata (WFA) over a combined alphabet $\Gamma = \Delta \times \Sigma$. The reason why we maintain separate alphabets will become evident in the next sections when we will consider modelling pairs of *continuous* input sequences and discrete outputs.

We say that a WFST is stochastic if the function f_A is a probability distribution over $(\Delta \times \Sigma)^*$. That is, if $f_A(x, y) > 0$ for all $\langle x, y \rangle \in (\Delta \times \Sigma)^*$ and $\sum_{\langle x,y \rangle \in (\Delta \times \Sigma)^*} f_A(x, y) = 1$. When x is continuous we have the analogous condition: $\int_{(x,y) \in (\Delta \times \Sigma)^*} f_A(x, y)\, dx\, dy{=}1$. To make it clear that $f_A(x, y)$ represents the probability of pairs of sequences $\langle x, y \rangle$ we will sometimes write it as $\mathbb{P}[x, y]$.

2.2 Sequence Taggers over Continuous Sequences

We will now consider the case in which the input sequences are not discrete but continuous. More specifically, let $\mathcal{X}$ be an arbitrary domain of input symbols (possibly infinite) and $\Phi = \{\phi_1, \ldots, \phi_k\}$ be a set of feature functions over $\mathcal{X}$, where $\phi_i : \mathcal{X} \to \mathbb{R}$. For any symbol $a \in \mathcal{X}$ we regard the vector $\Phi(a) = [\phi_1(a), \ldots, \phi_k(a)] \in \mathbb{R}^k$ as the real representation of a under the $\mathcal{X} \to \mathbb{R}^k$ mapping induced by Φ. When necessary we will use $\Phi(\mathcal{X})$ to refer to the range of this mapping.

We could attempt to define a WFST over $(\mathcal{X} \times \Sigma)^*$ as:

$$f_A(x, y) = \alpha_1^\top A^{y_1}_{\Phi(x_1)} \cdots A^{y_T}_{\Phi(x_T)} \alpha_\infty \quad . \tag{2}$$

Clearly, there is a problem with the above formulation because there are an infinite number of transition matrices (i.e. one for each member of $\Phi(\mathcal{X}) \times \Sigma$), thus we need to impose some further restrictions on f_A. The first observation is that instead of regarding $A^\sigma_{\Phi(a)}$ as a matrix in $\mathbb{R}^{m \times m}$ we can define it as a function:

$$A(\Phi(a), \sigma) : \mathbb{R}^k \times \Sigma \to \mathbb{R}^{m \times m} \tag{3}$$

We can now restrict f_A by restricting A, in particular we will assume that:

$$A(\Phi(a), \sigma) = \sum_{l=1}^{k} \phi_l(a) O^\sigma_l \tag{4}$$

where $O_l^\sigma \in \mathbb{R}^{m \times m}$ is an operator associated with each of the k functions of Φ and each output symbol $\sigma \in \Sigma$. Thus for each output symbol we restrict our transition function to be a linear combination of a set of k elementary operators. The weights of the linear combination are those induced by Φ.

In summary, a *Continuous Weighted Finite-state Sequence Tagger (CWFST)* over $(\Phi(\mathcal{X}) \times \Sigma)^*$ with m states can be defined as a tuple $A = \langle \Phi, \alpha_1, \alpha_\infty, O_l^\sigma \rangle$ where Φ is a set of k functions, $\alpha_1, \alpha_\infty \in \mathbb{R}^m$ are the initial and final weight vectors, and $O_l^\sigma \in \mathbb{R}^{m \times m}$ are the $k \times |\Sigma|$ operator matrices associated with each each symbol in Σ and each function in Φ. The function f_A realized by a CWFST is defined as:

$$f_A(x, y) = \alpha_1^\top \; A(\Phi(x_1), y_1) \cdots A(\Phi(x_T), y_T) \; \alpha_\infty \tag{5}$$

$$= \alpha_1^\top \left(\sum_{l=1}^{k} \phi_l(x_1) O_l^{y_1} \right) \cdots \left(\sum_{l=1}^{k} \phi_l(x_T) O_l^{y_T} \right) \alpha_\infty \tag{6}$$

2.3 Some Examples

We give now some examples of classes of functions that can be computed by CWFSTs.

A WFST as a CWFST We start by considering WFSTs as they were defined in the previous section. It is easy to see that if we have a WFST defined over $\Delta \times \Sigma$ we can construct a CWFST that will compute the same function. The construction is very simple, to map a WFST $A = \langle \alpha_1, \alpha_\infty, A_\delta^\sigma \rangle$ to a CWFST $A' = \langle \Phi, \alpha_1', \alpha_\infty', O_l^\sigma \rangle$ we perform the following construction: (1) Define one indicator function $\phi_\delta : \Delta \to \mathbb{R}$ for each $\delta \in \Delta$ as: $\phi_\delta(a) = 1$ if $a = \delta$ and 0 otherwise; (2) Set the $|\Delta| \times |\Sigma|$ operators to $O_l^\sigma = A_l^\sigma$; (3) Define $\alpha_1' = \alpha_1$ and $\alpha_\infty' = \alpha_\infty$. Clearly, the CWFST A' resulting from this construction will compute the same function as A since by construction $A(\Phi(\delta), \sigma) = A_\delta^\sigma$.

Transitions as Mixture Models We will now describe a more interesting case of a distribution over $(\mathcal{X} \times \Sigma)^*$ that can be represented as a CWFST. To motivate this example consider a gesture recognition problem where given a sequence of video frames we wish to predict the gesture been performed at each point in time.

One of the challenges in the gesture recognition task is that each video frame lies in a high-dimensional space which makes generalization to unseen samples difficult. To alleviate this problem we could consider a two step process where in the first step we induce a mapping from the high-dimensional space to a lower dimensional semantic space.

For example in the first step, like in [10] we could learn a visual topic model [7] over frames and represent each frame as a posterior distribution over visual topics. In the second step we need to be able to learn a sequence model from the topic space to gesture labels. To define such a model we will make use of some intermediate latent variables.

More precisely, let $H = \{c_1, \ldots, c_m\}$ be a set of m hidden states and Z be a k dimensional multinomial random variable. In the gesture recognition example Z would correspond to the latent topic variable for each video frame. Consider now the following distribution over paired $\langle x, y \rangle$ sequences:

$$\mathbb{P}[x, y] = \sum_{h \in H^{T+1}} \mathbb{P}[x, y, h] \tag{7}$$

$$= \sum_{h \in H^{T+1}} \mathbb{P}[h_0] \prod_{t=1}^{T-1} \mathbb{P}[h_{t+1}, x_t, y_t \mid h_t] \tag{8}$$

$\mathbb{P}[h_{t+1}, x_t, y_t | h_t]$ is the probability of emiting a pair of symbols (x, y) at time t and transitioning to a new state. Since x might lie in a high-dimensional space, to ease modelling of this conditional distribution we will define it as a mixture of k elementary conditional distributions:

$$\{\ \mathbb{P}_1[h_{t+1}, y_t \mid h_t], \ldots, \mathbb{P}_k[h_{t+1}, y_t \mid h_t]\ \} \tag{9}$$

More precisely, we define the transition function as:

$$\mathbb{P}[h_{t+1}, x_t, y_t \mid h_t] = \sum_{l=1}^{k} \mathbb{P}_l[h_{t+1}, y_t \mid h_t]\ \mathbb{P}[z = l \mid x_t]\ \mathbb{P}[x_t] \tag{10}$$

Thus, in this model the emission of an output symbol y is conditioned on z which is itself conditioned on the input variable x. Intuitively, we can think of $\mathbb{P}[z = l|x]$ as the probability of x taking discrete label l. In the gesture example, this would correspond to the posterior probability of a topic l given some input x. An alternative interpretation is that Z induces a soft partition of $\mathcal{X}$. The model exploits this partition to induce a better mapping between inputs and outputs.

Finally, we show how to construct a CWFST that realizes $\mathbb{P}[x, y]$. The idea is quite simple, we define a feature function for each of the k possible values that Z can take. More precisely, we define a CWFST A in the following manner: (1) Define one feature function $\phi_l(x)$ for each possible value of Z as $\mathbb{P}[z = l \mid x]\ \mathbb{P}[x]$; (2) Define the $k \times |\Sigma|$ operators as $O_l^{\sigma}(i, j) = \mathbb{P}_l[h_{t+1} = i, \sigma \mid h_t = j]$; (3) Define $\alpha_1(i) = \mathbb{P}[h_0 = i]$ and $\alpha_\infty = \mathbf{1}$. It is easy to see that A computes $\mathbb{P}[x, y]$ since by definition $A(\Phi(\delta), \sigma) = \mathbb{P}[h_{t+1}, \delta, \sigma \mid h_t]$.

We would like to end this section with a note on the limitations of the model. One of the key assumptions is that the feature functions depend only on the input. This means that the feature function needs to capture enough information so that at any point in time the output y_t can be predicted knowing the current state vector (which is a summary of the $[x_{1:t-1}; y_{1:t-1}]$ history) and the input features at time t. In other words, there must be enough information in the feature functions to explain the dynamics of the output symbols.

3 Spectral Learning of Stochastic CWFSTs

Recall that a stochastic CWFST computes a function $f_{\mathbf{A}}$ that is a probability distribution over $(\Delta \times \Sigma)^*$. In this section we will derive a learning algorithm for inducing the parameters of the CWFST from samples. We begin by defining some expectation matrices induced by f_A. We continue by presenting a duality result between a subclass of stochastic CWFSTs and factorizations of these matrices. Finally, we describe the spectral method for CWFSTs which is a statistically robust implementation of the arguments used in the duality proof.

3.1 Duality and Minimal CWFST

Let $\mathbb{P}$ be the function computed by f_A, we define functions: $\phi_{ij}(aa') = \phi_i(a)\phi_j(a')$ and $\phi^{\sigma}_{ilj}(aa'a'', \sigma'\sigma''\sigma''') = \phi_i(a)\phi_l(a')\phi_j(a'')\mathbb{P}(a')$ if $\sigma'' = \sigma$ and 0 otherwise. Using these functions we construct the observable statistics matrices $H_1 \in \mathbb{R}^k$, $H_2 \in \mathbb{R}^{k\times k}$, $H^{\sigma}_l \in \mathbb{R}^{k\times k}$ and $C \in \mathbb{R}^{k\times k}$ as:

$$H_1(i) = \mathbb{E}_{\mathbb{P}}[\phi_i(a)] \tag{11}$$

$$H_2(i,j) = \mathbb{E}_{\mathbb{P}}[\phi_{ij}(aa')] \tag{12}$$

$$H^{\sigma}_l(i,j) = \mathbb{E}_{\mathbb{P}}[\phi^{\sigma}_{ilj}(aa'a'', \sigma'\sigma''\sigma''')] \tag{13}$$

$$C(i,j) = \mathbb{E}_{\mathbb{P}}[\phi_i(a)\phi_j(a)] \tag{14}$$

We say that a CWFST $A = \langle \Phi, \alpha_1, \alpha_\infty, O^{\sigma}_l \rangle$ with $O^{\sigma}_l \in \mathbb{R}^{m\times m}$ for $l = 1:k$ is minimal for f_A if $\text{rank}(H_2) = m$ and $\text{rank}(C) = k$.

The rank-m restriction on H_2 is analogous to the restriction that O and T be rank m in [15, 6]. There are ways to relax this assumption by considering higher order expectations, but this is outside the scope of the paper. Now we will show a relationship between minimal A and some useful rank-m factorizations of H_2. Under appropriate stationary assumptions:

Lemma 1. *Let H_1, H_2, H^{σ}_l and C be the expectation matrices induced by an m-state minimal A. There exist matrices $F \in \mathbb{R}^{k\times m}$ and $B \in \mathbb{R}^{m\times k}$ such that the following holds:*

$$H_2 = FB \tag{15}$$

$$H^{\sigma}_l = F \sum_{i=1}^{k} O^{\sigma}_i C(l,i) B \tag{16}$$

$$H_1 = F\alpha_\infty = \alpha_1^\top B \tag{17}$$

Proof. Define a *forward* vector $f_l \in \mathbb{R}^m$ and a *backward* vector $b_l \in \mathbb{R}^m$ for each feature function:

$$f_l = \alpha_1^\top \int_{(x,y)\in(\mathcal{X}\times\Sigma)^*} A(x,y)\phi_l(x)\ dx\ dy \tag{18}$$

$$b_l = \int_{(x,y)\in(\mathcal{X}\times\Sigma)^*} A(x,y)\phi_l(x)\ dx\ dy\ \alpha_\infty \tag{19}$$

where we use the shorthand notation $A(x,y) = A(\Phi(x_1), y_1) \cdots A(\Phi(x_T), y_T))$. It is not hard to see that H_2 can be written as:

$$\begin{aligned} H_2(i,j) &= E[\phi_i(a)\phi_j(a')] \\ &= \int_{(xa,y\sigma)} \int_{(a'x',\sigma'y')} f_A(xaa'x', y\sigma\sigma'y')\ \phi_i(a)\phi_j(a') \\ &= \int_{(xa,y\sigma)} \int_{(a'x',\sigma'y')} \alpha_1^\top A(xa,y\sigma) A(a'x',\sigma'y')\alpha_\infty\ \phi_i(a)\phi_j(a') \\ &= \langle f_i, b_j \rangle \end{aligned}$$

Thus if we define a forward matrix $F \in \mathbb{R}^{k \times m}$ where each row corresponds to a forward vector and a backward matrix $B \in \mathbb{R}^{m \times k}$ where each column corresponds to a backward vector we get $H_2 = FB$ as desired. For the next claim we have:

$$\begin{aligned} H_l^\sigma(i,j) &= \mathbb{E}_{\mathbb{P}}[\phi_{ilj}^\sigma(a'aa'', \sigma''\sigma'\sigma''')] \\ &= \int_{(xa',y\sigma'')} \int_a \int_{(a''x',\sigma'''y')} f_A(xa'aa''x', y\sigma''\sigma\sigma'''y')\phi_i(a')\phi_l(a)\phi_j(a'')\mathbb{P}(a) \\ &= f_i^t \int_a A(a,\sigma)\phi_l(a)\mathbb{P}(a)\ b_j \\ &= f_i^t \ \sum_{r=1}^{k} O_r^\sigma C(l,r)\ \ b_j \end{aligned}$$

A few extra algebraic manipulations using F and B give us the remaining claims. □

We now develop the opposite direction of the duality between factorizations and minimal CWFSTs, which is the key to understand the spectral learning algorithm. The following theorem shows that given any rank factorization of H_2 we can compute a CWFST for f.

Theorem 1. *Let H_1, H_2, H_l^σ and C be the expectation matrices of some function f computed by a minimal CWFST and let $H_2 = FB$ be a rank factorization, then $A = \langle \Phi, \alpha_1, \alpha_\infty, O_l^\sigma \rangle$ defined as:*

$$\alpha_\infty = F^+ H_1 \tag{20}$$

$$\alpha_1^\top = H_1 B^+ \tag{21}$$

$$[O_1^\sigma(i,j), \ldots, O_k^\sigma(i,j)]^\top = C^{-1}[Q_1^\sigma(i,j), \ldots, Q_k^\sigma(i,j)] \tag{22}$$

$$Q_l^\sigma = F^+ H_l^\sigma B^+ \tag{23}$$

computes f.

Proof. Let $\widetilde{A} = \langle \Phi, \widetilde{\alpha}_1, \widetilde{\alpha}_\infty, \widetilde{O}_l^\sigma \rangle$ be a minimal CWFST for f that induces rank factorization $H_2 = \widetilde{F}\widetilde{B}$. We first show that there exists an invertible matrix M

such that for all $(x, y) \in (\mathcal{X} \times \Sigma)^*$ we have that: $M^{-1}\widetilde{A}(x, y)M = A(x, y)$. Define $M = \widetilde{B}B^+$, we have that: $F^+\widetilde{F}\widetilde{B}B^+ \implies F^+H_2B^+ = I \implies M^{-1} = F^+\widetilde{F}$. Thus M is invertible. We also have that for every σ and every l the following holds:

$$\sum_{i=1}^{k} O_i^\sigma C(l, i) = F^+ H_l^\sigma B^+ = F^+\widetilde{F} \sum_{i=1}^{k} \widetilde{O}_i^\sigma C(l, i)\widetilde{B}B^+$$

$$= M^{-1} \sum_{i=1}^{k} \widetilde{O}_i^\sigma C(l, i)M \tag{24}$$

For each σ we have k different equations, one per feature l:

$$\sum_{i=1}^{k} O_i^\sigma C(l, i) = M^{-1} \sum_{i=1}^{k} \widetilde{O}_i^\sigma C(l, i)M \tag{25}$$

Fixing the right part of the equation we observe that it is a system of k equations with k variables O_i^σ. Since f is minimal C is invertible and we can perform Gauss elimination and end up with a unique solution for the system. Since $O_i^\sigma = M^{-1}\widetilde{O}_i^\sigma M$ is a solution we must have that

$$\forall (a, \sigma) \in (\mathcal{X} \times \Sigma) \ : \ \sum_{i=1}^{k} O_i^\sigma \phi_i(a) = M^{-1} \sum_{i=1}^{k} \widetilde{O}_i^\sigma \phi_i(a)M \qquad . \tag{26}$$

Some algebraic manipulations give: $\alpha_1^\top = \widetilde{\alpha}_1^\top M$ and $\alpha_\infty = M^{-1}\widetilde{\alpha}_\infty$. Therefore, we can compute f as:

$$\begin{aligned} f_A(x, y) &= \alpha_1^\top \left(\sum_{l=1}^{k} O_l^{y_1} \phi_l(x_1) \right) \cdots \left(\sum_{l=1}^{k} O_l^{y_t} \phi_l(x_t) \right) \alpha_\infty \\ &= \widetilde{\alpha}_1^\top MM^{-1} \left(\sum_{l=1}^{k} \widetilde{O}_l^{y_1} \phi_l(x_1) \right) M \cdots M^{-1} \left(\sum_{l=1}^{k} \widetilde{O}_l^{y_t} \phi_l(x_t) \right) MM^{-1}\widetilde{\alpha}_\infty \\ &= f_{\widetilde{A}}(x, y) = f \end{aligned} \tag{27}$$

□

This result shows that there exists a duality between rank factorizations of H_2 and minimal CWFST for f. A consequence of this is that minimal CWFSTs for a function f with covariance C are related to each other via some change of basis.

Corollary 1. *Let $A = \langle \Phi, \alpha_1, \alpha_\infty, O_l^\sigma \rangle$ and $\widetilde{A} = \langle \Phi, \widetilde{\alpha}_1, \widetilde{\alpha}_\infty, \widetilde{O}_l^\sigma \rangle$ be two minimal CWFSTs for some f of rank m with covariance C. Then there exists an invertible matrix $M \in \mathbb{R}^{m \times m}$ such that $\alpha_1^\top = \widetilde{\alpha}_1^\top M$, $\alpha_\infty = M^{-1}\widetilde{\alpha}_\infty$ and*

$$\forall (a, \sigma) \in (\mathcal{X} \times \Sigma) \ : \ \sum_{l=1}^{k} O_l^\sigma \phi_l(a) = M^{-1} \sum_{l=1}^{k} \widetilde{O}_l^\sigma \phi_l(a)M \qquad .$$

In practice, we do not observe H_2, H_l^σ and C but we can estimate them from n training samples (x, y). The errors in the estimation can be bounded using the Hoeffding inequality, for example for H_2 we would get:

$$P(|\widehat{\mathbb{E}}[\phi_i(a)\phi_j(a')] - \mathbb{E}[\phi_i(a)\phi_j(a')]| > \epsilon) \leq 2\exp^{\frac{-2n\epsilon^2}{(\mu-\lambda)^2}}$$

where a and a' are any two input symbols, and λ and μ are bounds on the minimum and maximum values for $\phi_i(a) \cdot \phi_j(a')$.

The spectral learning algorithm that we present in the next section uses the singular value decomposition of H_2; this choice of low rank decomposition is appealing because its robustness to noise in the estimation of H_2. Using results from the stability of the singular value decomposition it is possible to show that the CWFST obtained from an approximate H_2 will be close to the one obtained using the exact statistics and thus the algorithm is statistically consistent. Also, it is not hard to see that one could use techniques similar to [15, 4, 6] to prove sample complexity bounds that depend linearly in the number of input features and $|\Sigma|$.

3.2 Spectral Algorithm

In this section we present a learning algorithm for stochastic CWFST based on spectral decompositions of observable statistics. Given samples from the joint distribution of paired input-output sequences $\mathbb{P}[x, y]$ and feature functions Φ, the task is to induce a CWFST: $A = \langle \Phi, \alpha_1, \alpha_\infty, O_l^\sigma \rangle$ that approximates $\mathbb{P}$.

More precisely, we are given:

- A set of n training samples $S = \{(x^1, y^1), \ldots, (x^n, y^n)\}$ of input-output sequences (where $x \in \mathcal{X}^T$ and $y \in \Sigma^T$ for some T), sampled from $\mathbb{P}[x, y]$
- A set of k feature functions $\Phi = \{\phi_1(a), \ldots, \phi_k(a)\}$
- The desired number of states m

We first use the training samples to compute estimates of H_1, H_2, H_l^σ and C. Recall that the compact SVD of a $k \times k$ matrix of rank m is given by: $H_2 = U\Lambda V^\top$ where $U \in k \times m$ and $V \in k \times m$ are orthogonal matrices. Clearly, $H_2 = (U\Lambda)V^\top$ is a rank m factorization of H_2, note that since $VV^\top = I$ this factorization is equivalent to $H_2 = (H_2V)V^\top$. Applying the ideas of the duality theorem for the factorization $F = H_2V$ and $B = V^\top$ we get that the model parameters are given by:

$$\alpha_\infty = (HV)^+ H_1 \tag{28}$$

$$\alpha_1^\top = H_1 V \tag{29}$$

$$[O_1^\sigma(i,j), \ldots O_k^\sigma(i,j)]^\top = C^{-1}[Q_1^\sigma(i,j), \ldots Q_k^\sigma(i,j)] \tag{30}$$

$$Q_l^\sigma = (HV)^+ H_l^\sigma V \tag{31}$$

Computing the expectation matrices is linear in the size of the training set and the number of features and output symbols. The cost of the algorithm is dominated by the singular value decomposition of the $k \times k$ matrix H_2 and therefore the overall cost is at most cubic on the number of features.

4 Related Work

Modelling continuous sequences with spectral methods has been studied in the context of HMMs [26], where a spectral algorithm for this case was derived. Their approach builds on previous work [27] on Hilbert Space Embeddings of conditional distributions. The main idea is first to map continuous distributions to points in a Hilbert Space and then derive a spectral method that works directly in the embedded space. [25] proposed an alternative spectral algorithm for continuous HMMs which is based on kernels. In spirit, our algorithm shares some similarities with all these methods since all of them work by embedding the transition function in some vectorial space.

Modelling continuous sequences has also been addressed in the original work by Jaeger [17, 18] on observable operator models (OOMs). Similar to that approach, we also consider operators that can be written as linear combinations of some *basis operators*. The main difference is that while they consider modelling continuous sequences, we consider the special case of *tagging* continuous sequences, that is, modelling paired sequences of continuous inputs and discrete outputs. Furthermore, we study the case in which the weights of the linear combination are provided in the form of feature functions that depend only on the continuous input.

More closely related to our approach is the work by [9] on transformed predictive state representations (TPSR). Although they do not directly address the sequence tagging problem (they are interested in predicting the conditional output of a dynamical system), implicitly they do consider paired sequences which can be sampled from a continuous space. Furthermore, they also use feature representations and operators that can be seen as linear combinations of elementary operators. One of the main difference between our work is that we focus on the case in which the following holds: (1) one of the two sequences comes from a small discrete alphabet; and (2) the weights of the linear operators depend on features of the continuous sequence only (in their case the feature function depends on both sequences). We show that for this special case the observable statistics on past and future events that are used to compute the basis of the operators depend only on the continuous input sequence. In their case, all observable statistics depend on both sequences. In this sense the difference between our work and theirs is analogous to the difference of a vanilla approach for computing joint distributions of discrete paired input/output sequences versus the work by [5], where they show that the basis can be computed from one of the two sequences alone.

Thus, although the learning algorithms might seem similar at first hand, the observable statistics on which they rely are quite different and thus they both have different properties. For example, we can consider cases in which we can easily estimate the statistics of the input distribution needed to compute the basis but in which estimating the joint input/output statistic might be hard. Another property of our model is that since some observable statistics depend only on the input we could easily use unlabeled samples (i.e. samples for which the output sequences are unknown) to better estimate them.

Apart from the differences mentioned above, the techniques that we use to prove the correctness of our algorithm are different. We derive the algorithm directly from a duality between low-rank factorizations of certain observable statistics and the parameters of the model. Finally, at the experimental level the two works are quite different. We test the accuracy of our learning algorithm in sequence tagging tasks while they test their model in tasks that involve predicting the future state of a dynamic system conditioned on the observed history.

5 Experiments

We conducted experiments on the Wall-following Navigation dataset of the UCI repository [1]. Given a sequence of sensor readings, the task is to predict an appropriate movement action out of a set of discrete actions. There are four possible actions: move-right, move-left, right-turn, left-turn. The sensor readings are the outputs of 24 ultrasound sensors sampled at a rate of 9 samples per second. When we frame this task as a sequence prediction problem over continuous inputs we have that x consists of sequences of sensor readings and y consists of sequences of appropriate actions.

The dataset consists of one long sequence of sensor readings and corresponding robot actions. For our experiments we split this sequence into 150 contiguous sequences of approximately 4 seconds each (36 contiguous samples per sequence). We then randomly partition these sequences and use 100 sequences as training data 25 sequences as validation and the remaining 25 sequences as test. When we report optimal performance for a given model, the validation sequences were used to pick the optimal number of states and to choose the optimal parameters of the feature functions.

5.1 Feature Functions

In general the feature functions can be validated using a held-out validation data. The goal of the first set of experiments is to test the robustness of our method with respect to different feature functions.

In kernel learning one usually assumes that a kernel function is provided, analogously a natural way to define feature functions in our setting it so assume that we are provided with some distance function between elements in $\mathcal{X}$. Once we have the distance function we can obtain centroids on the input space by performing vector-quantization (e.g. k-means) using the given distance. If a kernel was provided instead we could also perform kernel k-means to obtain centroids. Finally, we compute features as similarities to each of the centroids. More specifically, to obtain features for these experiments we do the following: (1) Perform k-means (with the provided distance function) on the input training samples to obtain k cluster centroids; and (2) For each cluster centroid c define the corresponding feature function: $\phi_c(x) = \frac{\exp \frac{-d(c,x)}{\tau}}{z}$. Here $-d(x, x')$ is the provided distance function, we will compare three distance functions: (1) Square Euclidean; (2) Correlation, computed as 1-sample correlation between

points; and (3) Cosine, computed as 1-the cosine of the included angle between points. The other parameter τ defines the width of the kernel function and z is a normalization constant. A small τ will result in sparse feature vectors for each point, where most of the mass will be concentrated around a few features. To compare against the discrete WFST we create a discrete alphabet by mapping each point to its closest cluster centroid according to the provided distance function.

In all experiments as a performance metric we report the accuracy on predicting actions for the test sequences. To predict the most probable sequence of actions y for a given test sequence x we must compute:

$$\text{argmax}_y \mathbb{P}(y|x) = \text{argmax}_y \mathbb{P}(x; y) \tag{32}$$

Due to the presence of the latent state variables the above computation is known to be untractable. Instead we use the standard approximation of maximizing the marginal probability at each time, that is we compute:

$$\text{argmax}_{y_t} \sum_{y_{1:t-1}, y_{t+1:T}} \mathbb{P}(x_{1:T}, y_{1:t-1} y_t y_{t+1:T}) \tag{33}$$

In the next section we validate the accuracy of this approximation.

Figure 1 shows the accuracy of CWFST and WFST as a function of the number of latent states m for the Euclidean, Correlation and Cosine feature functions. The number of features for these graphs is 80. As we can see CWFST outperforms WFST for all feature functions. In the three figures we can see the performance of CWFST for different values of τ (i.e. different feature functions). Larger values of τ result in feature functions that induce a softer partition of the input space. Thus we expect that for small τ values CWFST and WFST give similar performance, and this seems to be the case.

CWFST seems to be quite robust to the particular choice of feature function and what seems to change in each case is the optimal kernel width τ. For the cosine and correlation functions sparser feature vectors seem to be preferred (i.e. smaller τ) than for the Euclidean distance function. WFST on the other hand seems to be less robust to the choice of distance function (used in this case to discretize the inputs) and Cosine and Correlation seem to perform significantly better than the Euclidean distance.

Figure 2 (Left) shows accuracy as a function of the number of features (i.e. for optimal number of states and τ). As we can see CWFST significantly outperforms WFST for any number of features. This seems to suggest that working with a soft partition of the input space always results in better performance, regardless of the number of partitions. This appears to be true independent of the particular choice of feature function.

We end this subsection with a note on how to pick the optimal number of states. In general, one should use a validation set to pick the optimal number of states. One advantage of spectral learning algorithms is that they are very fast, hence parameter validation is cost-less. Still, we can use information of the spectrum of H_2 to guide our search for optimal m. Figure 2 (Center) shows

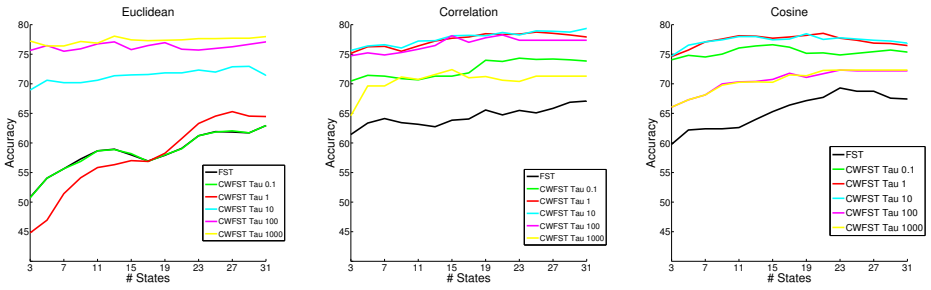

Fig. 1. Accuracy as a function of the number of states for different feature functions.

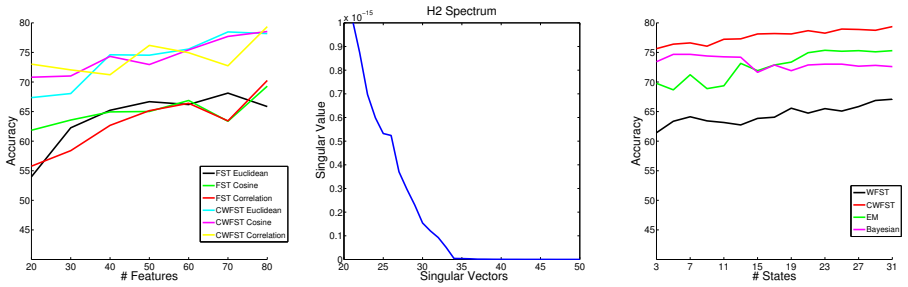

Fig. 2. (Left) Accuracy as a function of the number of features. (Center) Singular Values of H_2. (Right) Accuracy as a function of number of states for different methods

sorted singular singular values of H_2 for the correlation model with 80 features. As we can see the singular values drop to almost 0 after the 34th singular vector. Most likely, the optimal number of states for the spectral method will be less than 34 and probably in between 30 and 35 states.

5.2 Comparison with other methods

In the second set of experiments we fix the feature function to be Correlation and compare CWFST against two other methods:

- (EM) We train a model as defined in section 2.3.2 using expectation maximization. The models were run for a maximum of 400 iterations but the actual stopping criteria was chosen using the held-out validation data. That is we picked the model resulting from the iteration that performed best in validation data, which was less than 400 iterations (see table 2).
- (Bayesian) As a second model to compare we choose the winner algorithm of a recent probabilistic automata competition [1]. The winner algorithm [23] was a Bayesian method that implements Collapsed Gibbs Sampling [14]. Since

[1] http://ai.cs.umbc.edu/icgi2012/challenge/Pautomac

this method assumes discrete inputs, we discretize the continuous inputs following the same approach that was discussed in the previous section for WFST.

Figure 2 (Right) compares the performance of CWFST, EM and Bayesian as a function of the number of states. As we can see CWFST outperforms both the EM and Bayesian algorithms. The Bayesian algorithm seems to be able to provide more compact models than EM (i.e. fewer number of states). Table 1 (Left) shows the performance of the best models for each learning algorithm. Recall that we resorted to approximate max marginal inference. Given that the average length of each sequence in the test sample is 10 it is still possible (though costly) to perform exact inference. That is to compute: $\text{argmax}_y \mathbb{P}(y|x) = \text{argmax}_y \mathbb{P}(x; y)$ by doing exhaustive search. The last row of Table 1 (Left) shows the accuracy of each model when the approximate inference is replaced by exact inference. In all cases we see an improvement in between 1 % and 2 %. This seems to suggest that the approximation is a good trade-off of accuracy vs inference time. Table 2 shows accuracy of EM as a function of training time (for optimal m and τ). For time comparison, the spectral training algorithm takes less than 30 seconds to train.

Table 1. Comparison with other methods

	#states	Acc Marginals	Acc Exact
EM:	23	75.36%	76.32%
Bayesian:	5	74.67%	75.90%
CWFST:	31	79.36%	81.12%
FST:	31	67.09%	68.06%

Table 2. Training time (in seconds) and accuracy for Expectation Maximization for optimal model with 23 states

iters:	1	20	60	80	140	180	200	210	400
time:	14s	300s	1100s	2000s	4200s	5400s	6000s	6400s	10000s
acc.:	68%	74%	73.8%	74.8%	75%	75.36*%	75.36%	75.31%	75.17%

6 Conclusions

In this paper we presented a novel spectral learning algorithm that allows to exploit the representational power of latent variables to solve sequence tagging

problems where the input is a continuous sequence and the output is discrete. Our approach is based on regarding the transition function of a weighted finite-state sequence tagger as a linear combination of atomic transition functions. We derive a spectral learning algorithm for this model from forward-backward mappings. The resulting algorithm is both simple and fast. Intuitively, the atomic transition functions operate on a soft partition of the input space. Experiments on a real task have shown the effectiveness of the method and its ability to take full advantage of these soft partitions.

References

1. A. Asuncion, D.N.: UCI machine learning repository (2007), http://www.ics.uci.edu/∼mlearn/MLRepository.html
2. Anandkumar, A., Hsu, D., Kakade, S.M.: A method of moments for mixture models and hidden markov models. CoRR abs/1203.0683 (2012)
3. Bailly, R.: Quadratic weighted automata: Spectral algorithm and likelihood maximization. Journal of Machine Learning Research (2011)
4. Bailly, R., Denis, F., Ralaivola, L.: Grammatical inference as a principal component analysis problem. In: Proc. ICML (2009)
5. Balle, B., Quattoni, A., Carreras, X.: A spectral learning algorithm for finite state transducers. ECML–PKDD (2011)
6. Balle, B., Quattoni, A., Carreras, X.: Local loss optimization in operator models: A new insight into spectral learning. In: Proceedings of ICML. pp. 1879–1886 (2012)
7. Blei, D.M., Ng, A.Y., Jordan, M.I., Lafferty, J.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 2003 (2003)
8. Boots, B., Siddiqi, S., Gordon, G.: Closing the learning planning loop with predictive state representations. I. J. Robotic Research (2011)
9. Boots, B., Gordon, G.J.: An online spectral learning algorithm for partially observable nonlinear dynamical systems. In: In Proceedings of the 25th National Conference on Artificial Intelligence (2001)
10. Bosch, A., Zisserman, A., Munoz, X.: Scene classification via pLSA. In: European Conference on Computer Vision (2006)
11. Chang, J.T.: Full reconstruction of markov models on evolutionary trees: Identifiability and consistency. Mathematical Biosciences 137, 51–73 (1996)
12. Clark, A.: Partially supervised learning of morphology with stochastic transducers. In: Proc. of NLPRS. pp. 341–348 (2001)
13. Eisner, J.: Parameter estimation for probabilistic finite-state transducers. In: Proc. of ACL. pp. 1–8 (2002)
14. Gao, J., Johnson, M.: A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In: Proceedings of EMNLP. pp. 344–352 (2008)
15. Hsu, D., Kakade, S.M., Zhang, T.: A spectral algorithm for learning hidden markov models. In: Proc. of COLT (2009)
16. Jaeger, H.: Observable operator models for discrete stochastic time series. Neural Computation 12, 1371–1398 (2000)
17. Jaeger, H.: Characterizing distributions of stochastic processes by linear operators. Tech. Rep. 62, German National Research Center for Information Technology (1999)

18. Jaeger, H.: Modeling and learning continuous-valued stochastic processes with ooms. Tech. Rep. 102, German National Research Center for Information Technology (2001)
19. Luque, F., Quattoni, A., Balle, B., Carreras, X.: Spectral learning in non-deterministic dependency parsing. EACL (2012)
20. Morency, L.P., Quattoni, A., Darrell, T.: Latent-dynamic discriminative models for continuous gesture recognition. In: CVPR (2007)
21. Mossel, E., Roch, S.: Learning nonsingular phylogenies and hidden markov models. In: Proc. of STOC (2005)
22. Quattoni, A., Wang, S., Morency, L., Collins, M., Darrell, T.: Hidden-state conditional random fields. Pattern Analysis and Machine Intelligence (2007)
23. Shibata, C., Yoshinaka, R.: Marginalizing out transition probabilities for several subclasses of pfas. In: JMLR Workshop and Conference Proceedings, ICGI 2012. vol. 21, pp. 259–263 (2012)
24. Siddiqi, S.M., Boots, B., Gordon, G.J.: Reduced-Rank Hidden Markov Models. In: Proc. AISTATS. pp. 741–748 (2010)
25. Siddiqi, S., Boots, B., Gordon, G.J.: Reduced-rank hidden Markov models. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010) (2010)
26. Song, L., Boots, B., Siddiqi, S.M., Gordon, G.J., Smola, A.J.: Hilbert space embeddings of hidden Markov models. In: Proc. 27th Intl. Conf. on Machine Learning (ICML) (2010)
27. Song, L., Huang, J., Smola, A., Fukumizu, K.: Hilbert space embeddings of conditional distributions with applications to dynamical systems (2009)
28. Wang, S.B., Quattoni, A., Morency, L.P., Demirdjian, D.: Hidden conditional random fields for gesture recognition. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2. pp. 1521–1527. CVPR '06 (2006)