Noname manuscript No.
(will be inserted by the editor)

Spectral Learning of Transducers Over Continuous Sequences

Adrià Recasens · Ariadna Quattoni

Received: date / Accepted: date

Abstract In this paper we present a spectral algorithm for learning weighted finite state transducers (WFSTs) over paired input-output sequences, where the input is continuous and the output discrete. WFSTs are an important tool for modeling paired input-output sequences and have numerous applications in real-world problems. Recently, Balle et al (2011) proposed a spectral method for learning WFSTs that overcomes some of the well known limitations of gradient-based or EM optimizations which can be computationally expensive and suffer from local optima issues. Their algorithm can model distributions where both inputs and outputs are sequences from a discrete alphabet.

However, many real world problems require modeling paired sequences where the inputs are not discrete but continuos sequences. Modelling continuous sequences with spectral methods has been studied in the context of HMMs (Song et al 2010), where a spectral algorithm for this case was derived. In this paper we follow that line of work and propose a spectral learning algorithm for modelling paired input-output sequences where the inputs are continuous and the outputs are discrete. Our approach is based on generalizing the class of weighted finite state transducers over discrete input-output sequences to a class where transitions are linear combinations of elementary transitions and the weights of this linear combinations are determined by dynamic features of the continuous input sequence.

At its core, the algorithm is simple and scalable to large data sets. We present experiments on a real task that validate the effectiveness of the proposed approach.

Keywords spectral learning · structure prediction · finite state transducers

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

1 Introduction

Weighted Finite State Transducers (WFSTs) are an important tool for modeling paired input-output sequences and have found numerous applications in areas such as natural language processing and computational biology. The most popular methods for learning WFSTs are based on gradient-based or EM optimizations, but these can be computationally expensive and are succeptible to local optima issues [(Clark 2001), (Eisner 2002)]. Other methods for learning transducers are based on grammar induction techniques [(Casacuberta 2000), (Bernard et al 2006)].

Recently, following an emerging line of work on spectral methods for latent variable sequence modelling (Chang 1996; Jaeger 2000; Mossel and Roch 2005; Bailly et al 2009; Hsu et al 2009; Siddiqi et al 2010b; Bailly 2011; Parikh et al 2011; Boots et al 2011; Balle et al 2012; Anandkumar et al 2012), Balle (Balle et al 2011) proposed a spectral method for learning WFSTs that overcomes some of the limitations of gradient-based and EM optimizations. Their algorithm can learn distributions where both inputs and outputs are sequences from a discrete alphabet.

However, many real world problems require modeling paired sequences where the inputs are not discrete but continuos sequences. For example, consider a robot moving in some environment, at each point in time the robot recieves readings from some sensors and must decide what action (out of a discrete set of actions) should be taken (Argall et al 2009). In this example, the continuous input would consist of a sequence of sensor readings and the output would consist of a sequence of discrete actions.

Many problems in computer vision can also be casted as modelling paired sequences of continuous inputs and discrete outputs (Quattoni et al 2007; Wang et al 2006; Morency et al 2007). For example, consider the problem of human gesture recognition where given a video sequence the task is to predict the gesture that is been performed at each frame. Clearly, this can be casted as a sequence prediction 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.

Modelling continuous sequences with spectral methods has been studied in the context of HMMs (Song et al 2010), where a spectral algorithm for this case was derived. Their approach builds on previous work [(Song et al 2009)] on Hilbert Space Embeddings of conditional distributions. The main idea is first to map continous distributions to points in a Hilbert Space and then derive a spectral method that works directly in the embedded space. Siddiqi et al (2010a) propose 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.

In this paper, we extend this previous line of work on spectral learning for continuous sequences to handle the task of modelling paired input-output sequences where the inputs are continuous and the outputs are discrete.

Our approach is based on generalizing the class of weighted finite state transducers 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.

Similar to (Luque et al 2012; Balle et al 2012), we develop a spectral method for our model from forward-backward recursions which are used to derive usefull matrix decompositions of observable statistics. These matrix decompositions are then in turn exploited to induce the relationships between observations and latent state dynamics.

Our derivations are rather simple and put emphasis on providing intuitions on the inner workings of the algorithm. We believe that this is important for making the work accessible to a general audience of machine learning practitioners and increasing the use of the method in real applications.

As with previous spectral methods our algorithm for learning finite state transducers 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.

We test our method on the task of predicting robot actions from sensor readings. The results illustrate the effectiveness of the proposed algorithm.

In summary the main contributions of this paper are:

– We present a model for paired sequences of continuous inputs and discrete outputs.
– We derive an efficient spectral learning algorithm for this model from forward-backward recursions.
– We present experiments on a real-task that validate the effectiveness of our aproach.

The rest of the paper is organized as follows. Section 2 presents the proposed model for paired sequences of continuous inputs and discrete outputs and provides some examples of distributions that can be modelled with this approach. Section 3 derives a spectral learning algorithm for this model. Section 4 presents experiments on a real task and Section 5 concludes the paper.

2 Models for Sequences of Continuos Inputs and Discrete Outputs

2.1 Preliminary: Weighted Finite State Transducers

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 Balle et al (2011) 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 transducer (WFST) over $\Delta \times \Sigma$ with m states can be defined as a tuple:

$$A = \langle \alpha_1, \alpha_\infty, A_\delta^\sigma \rangle \tag{1}$$

where α_1, $\alpha_\infty \in \mathbb{R}^m$ are the initial and final weight vectors and $A_\delta^\sigma \in \mathbb{R}^{m \times m}$ 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:

$$f_A(x, y) = \alpha_\infty^\top A_{x_T}^{y_T} \cdots A_{x_1}^{y_1} \alpha_1 \tag{2}$$

The above equation is simply 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 $s_{t+1} = A_{x_t}^{y_t} s_t$ 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 (WFSA) 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 modeling 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 $\sum_{\langle x,y \rangle \in (\Delta \times \Sigma)^*} f_A(x, y) = 1$. 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 Transducers 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 linearly independent functions over $\mathcal{X}$, where:

$$\phi_i : \mathcal{X} \to \mathbb{R} \quad . \tag{3}$$

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:

$$f_A(x, y) = \alpha_\infty^\top A_{\Phi(x_T)}^{y_T} \cdots A_{\Phi(x_1)}^{y_1} \alpha_1 \tag{4}$$

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 $A_{\phi(a)}^\sigma$ as a matrix in $\mathbb{R}^{m \times m}$ we can define it as a function:

$$A(\phi(a), \sigma) : \mathbb{R}^k \times \Sigma \to \mathbb{R}^{m \times m} \tag{5}$$

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

$$A(\phi(a), \sigma) = \sum_{l=1}^{k} \phi_l(a) O_l^\sigma \tag{6}$$

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.

Wrapping up, a *Continuous Weighted Finite State Transducer (CWFST)* over $(\Phi(\mathcal{X}) \times \Sigma)^*$ with m states can be defined as a tuple:

$$A = \langle \Phi, \alpha_1, \alpha_\infty, O_l^\sigma \rangle \tag{7}$$

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:

$$\begin{aligned} f_A(x, y) &= \alpha_\infty^\top \; A(\phi(x_T), y_T) \cdots A(\phi(x_1), y_1) \; \alpha_1 & (8) \\ &= \alpha_\infty^\top \left(\sum_{l=1}^{k} \phi_l(x_T) O_l^{y_T} \right) \cdots \left(\sum_{l=1}^{k} \phi_l(x_1) O_l^{y} \right) \alpha_1 & (9) \\ & & (10) \end{aligned}$$

2.3 Some Examples

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

2.3.1 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:

– 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 otherwhise.
– Set the $|\Delta| \times |\Sigma|$ operators to $O_l^\sigma = A_l^\sigma$
– 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 $A(\phi(\delta), \sigma) = A_\delta^\sigma$.

2.3.2 Transitions as Mixture Models

We will now describe a more intersting 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 (Bosch et al 2006) we could learn a visual topic model (Blei et al 2003) 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 $H = \{c_1, \dots, c_m\}$ 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:

$$\begin{aligned}
\mathbb{P}[x,y] &= \sum_{h \in H^{T+1}} \mathbb{P}[x,y,h] && (11)\\
&= \sum_{h \in H^{T+1}} \mathbb{P}[h_0] \prod_{t=1}^{T-1} \mathbb{P}[h_{t+1}, x_t, y_t | h_t] && (12)\\
& && (13)
\end{aligned}$$

$\mathbb{P}[h_{t+1}, x_t, y_t | h_t]$ 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:

$$\{\mathbb{P}_1[h_{t+1}, y_t | h_t] \dots \mathbb{P}_k[h_{t+1}, y_t | h_t]\} \qquad (14)$$

More precisely, we define the transition function as:

$$\mathbb{P}[h_{t+1}, x_t, y_t | h_t] = \sum_{l=1}^{k} \mathbb{P}_l[h_{t+1}, y_t | h_t] \mathbb{P}[z = l | x_t] \mathbb{P}[x_t] \qquad (15)$$

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 will 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:

– Define one feature function $\phi_l(x)$ for each possible value of Z as $\mathbb{P}[z = l|x]\mathbb{P}[x]$
– Define the $k \times |\Sigma|$ operatos as $O_l^\sigma(i, j) = \mathbb{P}_l[h_{t+1} = i, \sigma | h_t = j]$
– 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 $A(\phi(\delta), \sigma) = \mathbb{P}[h_{t+1}, \delta, \sigma | h_t]$

3 Spectral Learning Algorithm for CWFST

In this section we present a learning algorithm for stochastic CWFST based on spectral decompositions of observable statistics. Our task is given samples from the joint distribution of paired input-output sequences $\mathbb{P}[x, y]$ and feature functions Φ 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 linearly independent feature functions $\Phi = \{\phi_1(a) \ldots \phi_k(a)\}$
– The desired number of states: m

A couple of observations before describing the algorithm. First, notice that we assume that the set of feature functions is given. In the gesture recognition example this corresponds to the fact that the distribution over latent topics for each video frame is learned in a pre-processing stage. Learning the feature functions jointly with the parameters of the CWFST is an interesting problem but it is outside the scope of this paper.

Second, observe that we require the feature functions to be linearly independent. Notice however, that this is not restrictive since for any set of feature functions we can find an equivalent set of linearly independent features by finding an appropriate basis.

As it is standard with spectral methods, to derive the learning algorithm for CWFST we will do the following:

– Define some matrices of observable statitstics.
– Show how we can derive usefull matrix factorizations from forward-backward recursions.
– Show how we can exploit these factorizations to recover model parameters.

Algorithm `LearnCWFT`$(\mathcal{X}, \Phi, \Sigma, S, m)$

Input:

- $\mathcal{X}$ is some input set.
- $\Phi = \{\phi_1(a), \ldots, \phi_k(a)\}$ is a set of k feature functions.
- Σ is the output alphabet
- $S = \{(x^1, y^1), \ldots, (x^n, y^n)\}$ is a training set of input-output sequences
- m is the number of hidden states of the CWFST

Output:

- CWFT parameters: $\langle \alpha_1,\ \alpha_\infty, O_l^\sigma \rangle$ for every feature function l and every output symbol.

---

1. For every pair of sequences (x, y) in S and every index $1 < t < |x|$ compute the feature vector $\phi(x_t) \in \mathbb{R}^k$ where $\phi(x_t) = [\phi_1(x_t), \ldots, \phi_k(x_t)]$
2. Use S to estimate matrix statistics $H_1 \in \mathbb{R}^k$, $H_2 \in \mathbb{R}^{k\times k}$, $H_l^\sigma \in \mathbb{R}^{k\times k}$ for every output symbol $\sigma \in \Sigma$ and every function $l \in \Phi$ and covariance matrix $C \in \mathbb{R}^{k\times k}$:

$$H_1(i) = \frac{1}{n} \sum_{(x_{1:T}, y_{1:T}) \in S} \frac{1}{T} \sum_{t=1}^{T} \phi_i(x_t) \tag{16}$$

$$H_2(j,i) = \frac{1}{n} \sum_{(x_{1:T}, y_{1:T}) \in S} \frac{1}{T} \sum_{t=2}^{T} \phi_j(x_t)\phi_i(x_{t-1}) \tag{17}$$

$$H_l^\sigma(j,i)) = \frac{1}{n} \sum_{(x_{1:T}, y_{1:T}) \in S} \frac{1}{\#(\sigma, \mathrm{y}_{2:\mathrm{T}-1})} \sum_{\substack{1<t<T \\ \text{s.t.}\, y_t = \sigma}} \phi_j(x_{t+1})\phi_l(x_t)\phi_i(x_{t-1}) \tag{18}$$

$$C(l,r) = \frac{1}{n} \sum_{(x_{1:T}, y_{1:T}) \in S} \frac{1}{T} \sum_{t=1}^{T} \phi_l(x_t)\phi_r(x_t) \tag{19}$$

3. Take U to be the matrix of top m left singular vectors of H_2
4. Compute the inverse of C
5. Compute $Q_l^\sigma = (U^\top H_l^\sigma)(U^\top H_2)^+$ for every function l and every output σ.
6. Compute the start and ending parameters of the CWFT as:

$$\alpha_1 = U^\top H_1 \tag{20}$$

$$\alpha_\infty = H_1^\top (U^\top H_2)^+ \tag{21}$$

7. Compute the transition matrices O_l^σ for every function l and every output so that for every transitions $\langle i, j\rangle$, $\sigma \in \Sigma$ and $l \in \Phi$ we have that:

$$\begin{bmatrix} O_1^\sigma(i,j) \\ \vdots \\ O_k^\sigma(i,j) \end{bmatrix} = C^{-1} \begin{bmatrix} Q_1^\sigma(i,j) \\ \vdots \\ Q_k^\sigma(i,j) \end{bmatrix} \tag{22}$$

Fig. 1 An algorithm for learning CWFST

We start by defining the observable matrix statistics $H_1 \in \mathbb{R}^k$, $H_2 \in \mathbb{R}^{k\times k}$ and $H_l^\sigma \in \mathbb{R}^{k\times k}$ as:

$$H_1(j) = \mathbb{E}_{\mathbb{P}}[\phi_j(x_t)] \tag{23}$$

$$H_2(j,i) = \mathbb{E}_{\mathbb{P}}[\phi_j(x_{t+1})\phi_i(x_t)] \tag{24}$$

$$H_\delta^r(j,i) = \mathbb{E}_{\mathbb{P}}[\phi_j(x_{t+1})\phi_r(x_t)\phi_i(x_{t-1})\mathrm{I}(\delta, y_t, x_t)] \tag{25}$$

where, $\mathrm{I}(\delta, y, x) = \mathbb{P}(x)$ if $\delta = y$, and otherwise is 0.
Our algorithm will also use the covariance matrix defined as

$$C(l, r) = \mathbb{E}_{\mathbb{P}}[\phi_l(x)\phi_r(x)] \tag{26}$$

We will now derive some usefull matrix factorizations from forward-backward mappings. We start by defining a forward mapping from features to $\mathbb{R}^m$:

$$b_l = \alpha_\infty^\top \int_{(x,y)\in(\mathcal{X}\times\Sigma)^*} A(x,y)\phi_l(x)\ dx\ dy \tag{27}$$

(where we use the shorthand notation $A(x,y) = A(\Phi(x_T), y_T)\cdots A(\Phi(x_1), y_1)$) Thus, for each feature l we defined a vector $\in \mathbb{R}^m$, we can think of the s entry of this vector as the expectation of feature l under the conditional distribution $\mathbb{P}[x, y|s]$. Similarly, we define a backward mapping:

$$f_l = \int_{(x,y)\in(\mathcal{X}\times\Sigma)^*} A(x,y)\phi_l(x)\ dx\ dy\ \ \alpha_1 \tag{28}$$

In this case we regard the s entry of vector $f_l \in \mathbb{R}^m$ as the expectation of feature l under the joint distribution $\mathbb{P}[x, y, s]$.

Now consider a backward matrix $B \in \mathbb{R}^{k\times m}$ where each row corresponds to a backward vector and a forward matrix $F \in \mathbb{R}^{m\times k}$ where each column corresponds to a forward vector. Some simple algebraic manipulations allows us to derive the following factorization of H_2:

$$H_2 = BF \tag{29}$$

A few more algebraic manipulations give us the following useful factorization for H_l^σ:

$$H_l^\sigma = B \sum_{r=1}^{k} O_r^\sigma C(l, r) F \tag{30}$$

Finally we also have that:

$$H_1 = B\alpha_1 = \alpha_\infty^\top F \tag{31}$$

The reason why Eq. (30) is usefull is that if we knew B and F in (29) we coud recover O_l^σ from Eq. (30). This is because we would only need to solve a system of km^2 equations on km^2 unknowns. More specifically, we could recover the model parameters by solving the system given by:

$$C \begin{bmatrix} O_1^\sigma(i,j) \\ \vdots \\ O_k^\sigma(i,j) \end{bmatrix} = \begin{bmatrix} Q_1^\sigma(i,j) \\ \vdots \\ Q_k^\sigma(i,j) \end{bmatrix} \tag{32}$$

Notice that we could also recover α_1 and α_∞ from Eq. (31).

The main idea of the learning algorithm described in figure 1 is to first find an m rank factorization $H_2 = BF$ by computing the thin singular value decomposition: $H_2 = [USV^T]$ and setting :

$$B = U(:, 1 : m) \tag{33}$$

$$F = (U^\top H_2) \tag{34}$$

$$\tag{35}$$

and then recover O_l^σ by solving Eq. (32). The cost of the algorithm is dominated by the singular value decomposition of the $k \times k$ matrix H_2.

4 Experiments

The goal of these experiments is to validate the effectiveness of the proposed algorithm (CWFST) on a real sequence prediction task. We will compare our method with two alternative approaches:

- (WFST) This approach consists of first discretazing the input space via k-means and then learning a WFST as defined in section 2.1.
- (EM) The second approach trains a model as defined in section 2.3.2 using expectation maximization.

4.1 Dataset and Task

We conducted experiments on the wall-following-navegation dataset of the UCI repository (A. Asuncion 2007). Given a sequence of sensor readings, the task is to predict an appropiate movement action out of a set of discrete actions. There are four possible accions: 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 appropiate 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 approximatly 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.

4.2 Feature Function

To generate feature functions Φ, we do the following:

1. Perform k-means using the input training samples to obtain k cluster centroids.
2. For each cluster centroid c define the corresponding feature function:

$$\phi_c(x) = \frac{\exp \frac{-d(c,x)}{\tau}}{z}$$

Where τ is a parameter to optimize and z is a normalization constant. All free parameters are optimized using the validation data. To create the discrete alphabet for the WFST we use the same k cluster centroids which are used to create the features of the CWFST. 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:

$$\text{argmax}_y \mathbb{P}(y|x) = \text{argmax}_y \mathbb{P}(x; y) \tag{36}$$

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

$$\text{argmax}_{y_t} \sum_{y_{1:t-1}, y_{t+1:T}} \mathbb{P}(x_{1:T}, y_{1:t-1} y_t y_{t+1:T}) \tag{37}$$

4.3 Results

Figures 2 and 3 show the accuracy of CWFST and WFST as a function of the number of latent states m, for models trained with 30 and 60 clusters. That means that we show performances for WFSTs with discrete alphabet sizes of 30 and 60 and for CWFSTs trained with 30 and 60 features.

In both figures we can see the performance of CWFST for different values of τ (i.e. different feature functions). Larger values of τ will result in feature functions that induce a softer partition of the input space. Smaller values of τ will tend to induce a hard partition where for each x only one of the feature functions will be non-zero. Thus we would expect that for small values of τ CWFST and WFST would give very similar performance, and this is case. However, for cases in which the feature function induces a soft partition of the input space CWFST outperforms significantly WFST.

Figure 4 shows accuracy as a function of the number of clusters (i.e. for optimal number of states and τ). As we can see CWFST significantly outperforms WFST for all numbers of clusters. 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.

Figure 5 compares the performance of CWFST, WFST and a CWFST trained with expectation maximization (EM), as a function on the number of states.

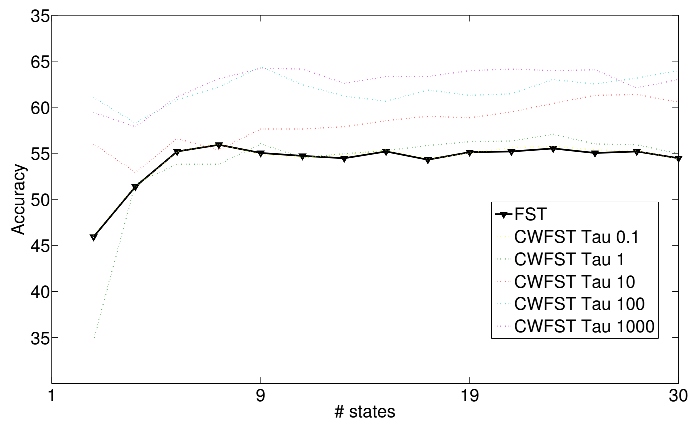

Fig. 2 Accuracy as a function of the number of states for a feature function with 30 clusters.

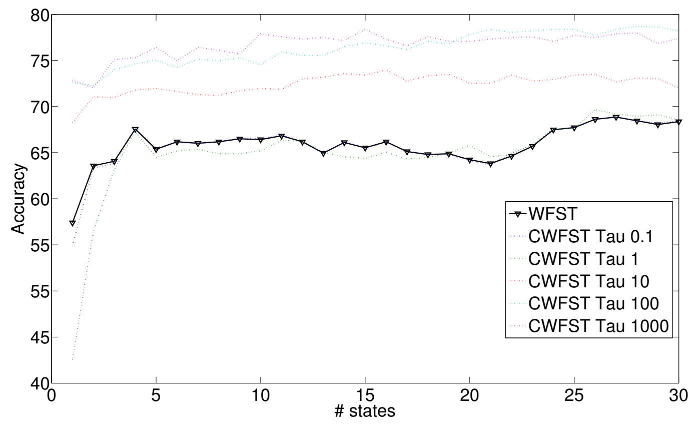

Fig. 3 Accuracy as a function of the number of states for a feature function with 60 clusters.

time:	248s	1700s	2000s	6400s	10000s	15000
accuracy:	67%	69%	70%	72%	75%	75%

Table 1 Training time (in seconds) and accuracy for Expectation Maximization

Finally, Table 1 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.

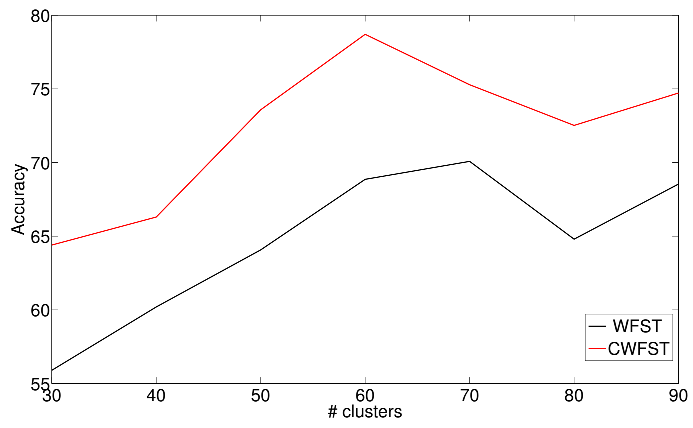

Fig. 4 Accuracy as a function of the number of clusters.

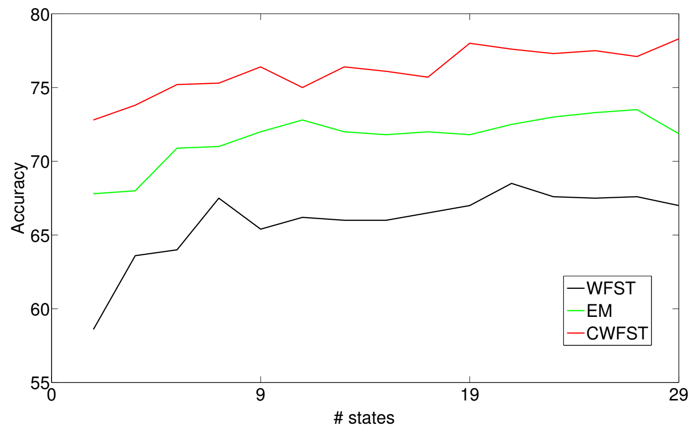

Fig. 5 Accuracy as a function of the number of states.

5 Conclusions

In this paper we presented a novel spectral learning algorithm that allows us to exploit the representational power of latent variables to solve sequence prediction problems where the input is a continous sequence and the output is discrete.

Our approach is based on regarding the transition function of a weighted finite state transducer 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. When compared to standard alternatives (i.e. EM optimization) we observe 2 to 3 orders of magnitude reduction in training time.

Intuitivelly, the atomic transition functions operate on a soft partition of the input space. Experiments on a real task have shown the effectivness of the method and its ability to take full advantage of these soft partitions.

In our current presentation the feature functions depend on the input x_t, but we could easily extent the model so as to consider feature functions that depend on both on x_t and y_t. Future work will explore this possibility.

Finally, a challenging and interensting open research question is wether we can derive spectral algorithms that learn simultaneously the optimal latent state parameters and a soft partition of the input space.

References

A Asuncion DN (2007) UCI machine learning repository. URL http://www.ics.uci.edu/∼mlearn/MLRepository.html

Anandkumar A, Hsu D, Kakade SM (2012) A method of moments for mixture models and hidden markov models. CoRR abs/1203.0683

Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(5):469–483, DOI 10.1016/j.robot.2008.10.024, URL http://dx.doi.org/10.1016/j.robot.2008.10.024

Bailly R (2011) Quadratic weighted automata: Spectral algorithm and likelihood maximization. Journal of Machine Learning Research

Bailly R, Denis F, Ralaivola L (2009) Grammatical inference as a principal component analysis problem. In: Proc. ICML

Balle B, Quattoni A, Carreras X (2011) A spectral learning algorithm for finite state transducers. ECML–PKDD

Balle B, Quattoni A, Carreras X (2012) Local loss optimization in operator models: A new insight into spectral learning. In: Langford J, Pineau J (eds) Proceedings of the 29th International Conference on Machine Learning (ICML-12), Omnipress, New York, NY, USA, ICML '12, pp 1879–1886

Bernard M, Janodet JC, Sebban M (2006) A discriminative model of stochastic edit distance in the form of a conditional transducer. Grammatical Inference: Algorithms and Applications 4201

Blei DM, Ng AY, Jordan MI, Lafferty J (2003) Latent dirichlet allocation. Journal of Machine Learning Research 3:2003

Boots B, Siddiqi S, Gordon G (2011) Closing the learning planning loop with predictive state representations. I J Robotic Research

Bosch A, Zisserman A, Munoz X (2006) Scene classification via pLSA. In: European Conference on Computer Vision

Casacuberta F (2000) Inference of finite-state transducers by using regular grammars and morphisms. Grammatical Inference: Algorithms and Applications 1891

Chang JT (1996) Full reconstruction of markov models on evolutionary trees: Identifiability and consistency. Mathematical Biosciences 137:51–73

Clark A (2001) Partially supervised learning of morphology with stochastic transducers. In: Proc. of NLPRS, pp 341–348

Eisner J (2002) Parameter estimation for probabilistic finite-state transducers. In: Proc. of ACL, pp 1–8

Hsu D, Kakade SM, Zhang T (2009) A spectral algorithm for learning hidden markov models. In: Proc. of COLT

Jaeger H (2000) Observable operator models for discrete stochastic time series. Neural Computation 12:1371–1398

Luque F, Quattoni A, Balle B, Carreras X (2012) Spectral learning in non-deterministic dependency parsing. EACL

Morency LP, Quattoni A, Darrell T (2007) Latent-dynamic discriminative models for continuous gesture recognition. In: CVPR, IEEE Computer Society

Mossel E, Roch S (2005) Learning nonsingular phylogenies and hidden markov models. In: Proc. of STOC

Parikh A, Song L, Xing E (2011) A spectral algorithm for latent tree graphical models. ICML

Quattoni A, Wang S, p Morency L, Collins M, Darrell T, Csail M (2007) Hidden-state conditional random fields. In: IEEE Transactions on Pattern Analysis and Machine Intelligence

Siddiqi S, Boots B, Gordon GJ (2010a) Reduced-rank hidden Markov models. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010)

Siddiqi SM, Boots B, Gordon GJ (2010b) Reduced-Rank Hidden Markov Models. In: Proc. AISTATS, pp 741–748

Song L, Huang J, Smola A, Fukumizu K (2009) Hilbert space embeddings of conditional distributions with applications to dynamical systems

Song L, Boots B, Siddiqi SM, Gordon GJ, Smola AJ (2010) Hilbert space embeddings of hidden Markov models. In: Proc. 27th Intl. Conf. on Machine Learning (ICML)

Wang SB, Quattoni A, Morency LP, Demirdjian D (2006) Hidden conditional random fields for gesture recognition. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2, IEEE Computer Society, Washington, DC, USA, CVPR '06, pp 1521–1527, DOI 10.1109/CVPR.2006.132, URL http://dx.doi.org/10.1109/CVPR.2006.132