A regularized spectral algorithm for Hidden Markov Models with applications in computer vision

Hà Quang Minh1  Marco Cristani1  Alessandro Perina2  Vittorio Murino1
1 Istituto Italiano di Tecnologia (IIT), Genoa 16163, Italy
2 Microsoft Research, WA, USA
{minh.haائع,marco.cristani,vittorio.murino}@iit.it, alperina@microsoft.com

Abstract

Hidden Markov Models (HMMs) are among the most important and widely used techniques to deal with sequential or temporal data. Their application in computer vision ranges from action/gesture recognition to video-surveillance through shape analysis. Although HMMs are often embedded in complex frameworks, this paper focuses on theoretical aspects of HMM learning. We propose a regularized algorithm for learning HMMs in the spectral framework, whose computations have no local minima. Compared with recently proposed spectral algorithms for HMMs, our method is guaranteed to produce probability values which are always physically meaningful and which, on synthetic mathematical models, give very good approximations to true probability values. Furthermore, we place no restriction on the number of symbols and the number of states. On various pattern recognition data sets, our algorithm consistently outperforms classical HMMs, both in accuracy and computational speed. This and the fact that HMMs are used in vision as building blocks for more powerful classification approaches, such as generative embedding approaches or more complex generative models, strongly support spectral HMMs (SHMMs) as a new basic tool for pattern recognition.

1. Introduction

Hidden Markov Models (HMM) are among the most important and widely used techniques in statistical learning, with numerous applications in various domains involving sequence modeling. Some areas in which HMMs have been applied successfully include speech recognition [11, 9], computer vision and pattern recognition [4, 6, 3], and bioinformatics [10, 5]. For the purpose of this paper, we limit our discussions to discrete HMMs, leaving continuous HMMs to future work.

Traditionally, algorithms for learning HMMs have mainly employed Expectation Minimization (EM) [2, 11]. While powerful and widely used, the main problem of EM methods is that they are prone to local minima. The quest for algorithms which are free of local minima and which are statistically consistent has been the focus of much research in the last decade. Two recent generalizations of HMMs, which are closely related, are Observable Operator Models (OMMs) [8] and Predictive State Representations (PSRs) [13]. Instead of the structure of unknown hidden states and emission probabilities, these models focus entirely on observation quantities and express sequence trajectories using linear operators, thus transforming probabilistic problems into linear algebraic ones.

Two recent algorithms implementing OMMs are [7, 12]. In [7], the authors proposed a spectral algorithm for learning joint and conditional probabilities that is free of local minima and is statistically consistent. Among the main assumptions by [7], which we describe mathematically below, are that the transition matrix is of full rank, and that the number of symbols is greater than or equal to the number of states. In [12], the authors extended the work in [7] to cover the case of low rank transition probability matrix. To handle the case the number of states is greater than the number of symbols, the authors proposed stacking several single observations together to build observation matrices.

1.1. Contributions of this work

Our present work builds upon the algorithms of [7, 12]. The main problem of these algorithms is that they are not very stable numerically. While they return exact results on exact observation statistics, on empirical observation statistics, which is what we have in practice, they often return probabilities which are negative or greater than one. As we discuss in detail below, this is due to their use of the Singular Value Decomposition and the pseudo-inverse operations.

To overcome this, we propose a regularized spectral algorithm. Compared to these methods, our algorithm

1. is guaranteed to produce probability values that are al-
ways physically meaningful, that is between 0 and 1;
2. on synthetic mathematical models, produces probabilities that approximate very well true theoretical values;
3. places no restriction on the number of symbols and number of states. In particular, when the number of symbols is smaller than the number of states, we do not require stacking several single observations together as in [12]. The theoretical justification for this case is significantly different from the case the number of states is smaller than or equal to the number of symbols.

1.2. Organization of the paper

We review the spectral algorithm for HMMs of [7] in Section 2. Our regularized algorithm is described in Section 3, with the case of low rank transition probability matrix in Section 3.4, and the case when the number of states is greater than the number of symbols in Section 3.5. We report our numerical experiments in Sections 4 and 5. The proofs for our mathematical results are given in Appendix A and in the Supplementary Material.

2. Spectral HMM

Let us first discuss the spectral algorithm for HMM from [7]. Let the hidden states be \(1, \ldots, n\), and the observation symbols be \(1, \ldots, m\). Let \(T \in \mathbb{R}^{m \times m}\) be the transition probability matrix: \(T_{ij} = \mathbb{P}(H_{t+1} = i | H_t = j)\). Let \(O \in \mathbb{R}^{n \times m}\) be the observation (emission) probability matrix: \(O_{ij} = \mathbb{P}(X_t = i | H_t = j)\). Then both \(T\) and \(O\) are column-stochastic matrices. Let the initial probability distribution be \(\mathbb{P} \in \mathbb{R}^m\); \(\mathbb{P}_i = \mathbb{P}(H_1 = i)\).

In the Observation Operator Representation model [8], for each symbol \(x \in \{1, \ldots, n\}\), we define
\[
A_x = TO_x = T \text{diag}(O_{x,1}, \ldots, O_{x,m}) \in \mathbb{R}^{m \times m},
\]
with
\[
(A_x)_{ij} = \mathbb{P}(H_{t+1} = i | H_t = j) \mathbb{P}(X_t = x | H_t = j).
\]
We have the following result [8].

**Lemma 1.** The joint probability of a sequence \(x_1, \ldots, x_t\), is given by
\[
\mathbb{P}(X_1 = x_1, \ldots, X_t = x_t) = \mathbf{1}_m^T A_{x_1} \ldots A_{x_t} \mathbb{P}_1,
\]
where \(\mathbf{1}_m = (1, \ldots, 1)^T\) is a column vector in \(\mathbb{R}^m\).

Alternatively, we have the following lemma for computing the joint and conditional probability distributions.

**Lemma 2.** For each \(1 \leq i \leq n\):
\[
\mathbb{P}(X_{t+1} = i, X_t = x_t, \ldots, X_1 = x_1) = (O A_{x_t} \ldots A_{x_1} \mathbb{P}_1).
\]
\[
\mathbb{P}(X_{t+1} = i | X_t = x_t, \ldots, X_1 = x_1) = \frac{(O A_{x_t} \ldots A_{x_1} \mathbb{P}_1)}{\mathbb{P}(x_1)}.
\]

2.1. Observable Operator Representation

The matrices \(A_x\)’s are not observable and thus in order to compute the parameters and/or probabilities associated with the HMM, we need to turn to observable quantities. Consider the following three matrices: \(P_1 \in \mathbb{R}^n\), \(P_{2,1} \in \mathbb{R}^{n \times n}\), \(P_{3,x,1} \in \mathbb{R}^{n \times n}\), which are defined by:
\[
(P_1)_i = \mathbb{P}(x_1 = i), \quad (P_{2,1})_{ij} = \mathbb{P}(x_2 = i, x_1 = j), \quad (P_{3,x,1})_{ij} = \mathbb{P}(x_3 = i, x_2 = x, x_1 = j),
\]
for \(1 \leq x \leq n\). In practice, these matrices are not known exactly, but can be approximated from training data. These observation matrices are expressed in terms of the hidden structure as follows [7]:
\[
P_1 = O \mathbb{P},
\]
\[
P_{2,1} = O T \text{diag}(\mathbb{F}) O^T,
\]
\[
P_{3,x,1} = O A_x T \text{diag}(\mathbb{F}) O^T.
\]
Consider the following set of Assumptions 1, which will subsequently be relaxed.

1. The number of symbols is greater than or equal to the number of states: \(n \geq m\).
2. \(\text{rank}(T) = m\).
3. \(\text{rank}(O) = m\).
4. \(\text{diag}(\mathbb{F}) > 0\).
5. There is a matrix \(U \in \mathbb{R}^{n \times m}\) such that the matrix \(U^T O\) is invertible.

In [7], the authors considered the following formulation. Under Assumptions 1, let
\[
b_1 = U^T P_1 \in \mathbb{R}^m, \quad b_\infty = (P_{21}^T U)^T P_1 \in \mathbb{R}^m,
\]
\[
B_x = (U^T P_{3,x,1})(U^T P_{2,1})^T \in \mathbb{R}^{m \times m}.
\]
Then
\[
b_1 = (U^T O) \mathbb{F}_1, \quad b_\infty = 1_m^T (U^T O)^{-1},
\]
\[
B_x = (U^T O) A_x (U^T O)^{-1}.
\]
Consequently,
\[
1_m^T A_{x_{t+1}} \mathbb{P}_1 = b_\infty B_{x_{t+1}} b_1,
\]
where we use the notation \(A_{x_{t+1}} = A_{x_t} \ldots A_{x_1}\). Thus the probability \(\mathbb{P}(x_{t+1}) = \mathbb{P}(X_1 = x_1, \ldots, X_t = x_t)\) can be expressed entirely in terms of the observation quantities \(b_1, b_\infty, B_{x_{t+1}}\).

**Choice of the matrix \(U\):** the matrix \(U\) is chosen by [7] to be the matrix whose columns are the first \(m\) left singular vectors of \(P_{2,1}\), that is those corresponding to the nonzero singular values.
2.1.1 Numerical instability

In practice, we only have access to the empirical versions \( P_1, P_{2,1}, P_{3,x,1} \) of the matrices \( P_1, P_{2,1}, P_{3,x,1} \), not their exact values. Numerical experiments indicate that the above formulation is not stable: it frequently produces invalid probabilities (negative or greater than one), and even when it does produce valid probabilities, they may not be close to theoretical ones. There are two main sources of numerical instability in this formulation:

1. The Singular Value Decomposition (SVD) operation

\[
P_{2,1} = W SV^T,
\]

(17)

taking the first \( m \) columns of \( W \) to form the matrix \( U \); the empirical singular vectors may not be very close to the theoretical ones (perhaps with exception of the first singular vector). Generally, the smaller \( m \) is compared to \( n \), the more stable the SVD operation in this situation becomes.

2. The pseudo-inverse operation

\[
(U^T P_{2,1})^+ = (U^T P_{2,1})^T (U^T P_{2,1})^{-1}.
\]

(18)

can be unstable: a very small change in the given matrix can cause a very big change in the pseudo-inverse.

We discuss ways to overcome these issues next.

3. Regularized spectral HMM

We first replace the pseudo-inverses in the above formulation with true inverses of appropriate matrices. This new formulation gives rise to a natural regularization step and, as we show below, covers also the case \( n \geq m \).

Lemma 3. Under Assumptions 1, consider the following quantities:

\[
b_\infty = (U^T P_{2,1} P_{2,1}^T U)^{-1}(U^T P_{2,1} P_{2,1}),
\]

(19)
\[
B_x = (U^T P_{3,x,1} P_{2,1}^T U)(U^T P_{2,1} P_{2,1}^T U)^{-1},
\]

(20)
\[
B_1 = U^T P_1 = (U^T O)^\frac{1}{\gamma}.
\]

(21)

Then

\[
b_\infty^T = 1_m (U^T O)^{-1},
\]

\[
B_x = (U^T O) A_x (U^T O)^{-1},
\]

and consequently

\[
b_\infty^T B_{x,1} b_1 = 1_m A_{x,1} \frac{1}{\gamma}.
\]

(22)

3.1. Choosing the matrix \( U \)

According to Assumptions 1, we require that the matrix \( U \) be such that \( U^T O \) is invertible. We propose to choose \( U \) randomly. We have that

\[
P_{2,1} = OT \text{diag}(\frac{1}{\gamma}) O^T.
\]

(23)

For a matrix \( U \in \mathbb{R}^{n \times m} \), we have

\[
(U^T P_{2,1} P_{2,1}^T U) = (U^T O)(T \text{diag}(\frac{1}{\gamma})) O^T (T \text{diag}(\frac{1}{\gamma}))^T (U^T O)^T.
\]

Since \( T \text{diag}(\frac{1}{\gamma}) \) is an \( m \times m \) matrix of full rank, \( OT \) is an \( m \times m \) matrix of full rank (\( \text{rank}(OT) = \text{rank}(O) = m \)), the matrix \( U^T O \) is invertible iff \( U^T P_{2,1} P_{2,1}^T U \) is of full rank. Thus we can choose \( U \) as follows:

Repeat

1. Randomly generate \( U \) of size \( n \times m \), with positive entries. Normalize so that \( \sum_{i,j} U_{ij} = 1 \).
2. Until \( \text{rank}(U^T P_{2,1} P_{2,1}^T U) = m \).

Note that we normalize \( \sum_{i,j} U_{ij} = 1 \), so that \( U \) does not dominate in products such as \( U^T P_1 \) or \( U^T P_{2,1} \).

3.2. Regularization of the inverse

To avoid instability in the inverse operation, it is necessary to carry out regularization. For \( \gamma > 0 \), we define

\[
b_{\infty,\gamma} = (U^T P_{2,1} P_{2,1}^T U + \gamma I)^{-1}(U^T P_{2,1} P_{2,1}),
\]

(24)
\[
B_{x,\gamma} = (U^T P_{3,x,1} P_{2,1}^T U)(U^T P_{2,1} P_{2,1}^T U + \gamma I)^{-1},
\]

(25)
and compute

\[
b_{\infty,\gamma}^T \left( \prod_{i=1}^t B_{x,i,\gamma} \right) b_1.
\]

(26)

Remark 1. For each regularization, we need to choose an appropriate parameter \( \gamma > 0 \). For our numerical experiments, we have chosen \( \gamma \) in the range \( 10^{-7} - 10^{-5} \).

3.3. Algorithm

We will focus on computing the conditional probabilities

\[
\mathbb{P}(x_t | x_1, \ldots, x_{t-1}) = \frac{\mathbb{P}(x_1, \ldots, x_t)}{\mathbb{P}(x_1, \ldots, x_{t-1})} = \frac{b_{\infty}^T B_{x,1} b_1}{b_{\infty}^T B_{x,t-1} b_1}.
\]

(27)

The joint probability is then

\[
\mathbb{P}(x_1, \ldots, x_t) = \prod_{\tau=1}^t \mathbb{P}(x_{\tau} | x_1, \ldots, x_{\tau-1}).
\]

(28)

Since the matrix \( U \) can be chosen randomly, we can compute the sequence of conditional probabilities \( R \) times, each time with a different \( U \), and take the average, excluding sequences that contain invalid probability values, which could still occur occasionally. Our algorithm is thus
Algorithm 1. Input:  
- The empirical matrices $\hat{P}_1$, $\hat{P}_{2,1}$, $\hat{P}_{3,2,1}$ from training data.  
- Test sequence $(x_t)_{t=1}^N$.  

Procedure:  
Choose $R \in \mathbb{N}$.  
Choose a regularization parameter $\gamma > 0$.  
For $r = 1$ to $R$:  
beginFor  
1. Generate $U$ of size $n \times m$ randomly, such that  
$\text{rank}(U^T \hat{P}_{2,1} \hat{P}_{2,1}^T) = m$.  
2. Compute, for $t = 1, \ldots, N$:  
\[
\bar{\hat{p}}_\gamma(x_t | x_1, \ldots, x_{t-1}) = \frac{\hat{b}_{t,\infty} \prod_{t'=t}^T (\hat{B}_{x,t'}) \hat{b}_1}{\hat{b}_{t,\infty} \prod_{t'=t-1}^T (\hat{B}_{x,t'}) b_1},
\]
where  
$\hat{b}_{t,\infty} = (U^T \hat{P}_{2,1} \hat{P}_{2,1}^T U + \gamma I)^{-1}(U^T \hat{P}_{2,1} \hat{P}_1)$. \hspace{1cm} (29)  
$\hat{B}_{x,t} = (U^T \hat{P}_{3,x,t} \hat{P}_{3,2,1}^T U)(U^T \hat{P}_{2,1} \hat{P}_{2,1}^T U + \gamma I)^{-1}$.  
3. If the sequence $(\bar{\hat{p}}_\gamma(x_t | x_1, \ldots, x_{t-1}))_{t=1}^N$ contains a negative or greater than 1 value, discard it.  
endFor  
Output:  
The average of all the sequences $(\bar{\hat{p}}_\gamma(x_t | x_1, \ldots, x_{t-1}))_{t=1}^N$ above.  
By its design, the algorithm is guaranteed to produce only meaningful probability values.  

3.3.1 On the use of randomness  
It might appear that the use of a random $U$ is similar to the use of a random initialization in an EM method. However, this is not the case. The use of randomized $U$s is only for numerical reasons. If we have exact values for $P_1, P_{2,1}, P_{3,2,1}$, then no regularization is necessary, $U$ simply gets canceled out, and one obtains the true global solution. If only empirical versions of $P_1, P_{2,1}, P_{3,2,1}$ are known, which is what happens in practice, then regularization is necessary to obtain physically meaningful results and for each $U$ one gets an approximate version of the true global solution. It is not a local minimum, but an empirical version of the true global minimum.  

For a fixed $U$, such as one obtained by performing SVD on $P_{2,1}$, there is no guarantee that the probabilities one obtains are valid. When allowing $U$ to be random, we can choose from an infinite number of $U$’s that give rise to meaningful results and discard those that do not. These are all approximate versions of the true global minimum. Thus the method is fundamentally different from EM algorithms.  

3.4. Low rank transition probability matrix  
Let us now relax Assumptions 1 and consider the case when the transition probability matrix $T$ is not of full rank. Let us adapt the techniques in [12] for our current formulation. Let $k \in \mathbb{N}, k \leq m$. The transition matrix $T$ admits the decomposition  
\[
T = RS, \quad R \in \mathbb{R}^{m \times k}, \quad S \in \mathbb{R}^{k \times m},
\]
with $\text{rank}(R) = \text{rank}(S) = \text{rank}(T) = k$. Assume that there is a matrix $U \in \mathbb{R}^{m \times k}$ such that  
\[
\text{rank}(U^T OR) = \text{rank}(SDiag(\vec{\pi})O^T) = k.
\]  
Then  
\[
\text{rank}(U^T P_{2,1} P_{2,1}^T U) = \text{rank}(P_{2,1}) = k.
\]  
Assume further that there is a vector $\vec{\pi} \in \mathbb{R}^k$ such that  
\[
\vec{\pi} = R \vec{\pi}_k.
\]  
Then  
\[
b_1 = U^T P_1 = U^T O \vec{\pi} = (U^T OR) \vec{\pi}_k \in \mathbb{R}^k.
\]  
Lemma 4. Consider the following quantities:  
\[
b_\infty = (U^T P_{2,1} P_{2,1}^T U)^{-1}(U^T P_{2,1} P_1) \in \mathbb{R}^k,
\]
\[
B_x = (U^T P_{3,x,1} P_{3,2,1}^T U) (U^T P_{2,1} P_{2,1}^T U)^{-1} \in \mathbb{R}^{k \times k},
\]  
Then  
\[
b_\infty^T b_1 = 1_m^T R(U^T OR)^{-1},
\]
\[
B_x = (U^T OR)SDiag(\vec{\pi}_k) OR^{-1},
\]  
and consequently  
\[
b_\infty^T B_{x,t} b_1 = 1_m^T A_{x,t} \vec{\pi}.
\]  
3.5. Number of symbols less than number of states  
Consider now the case when the number of symbols is less than the number of states, that is $n \leq m$. We show that the above formulation also covers this case. First note that if $k \leq n \leq m$, then the same reasoning as in Section 3.4 applies. However, for $n < k \leq m$, the theoretical justification is different and the mathematical proof more involved.  

Proposition 1. Assume that $n \leq k = \text{rank}(T) \leq m$. Assume that $\text{rank}(O) = \text{rank}(P_{2,1}) = n$. Let $U$ be an invertible $n \times n$ matrix. Let $b_\infty, B_x, b_1$ be defined as in Lemma 3. Then  
\[
b_\infty^T p_1 P_{2,1}^{-1}(U^T)^{-1} = 1_n^T (U^T)^{-1},
\]
\[
B_x = U^T P_{3,x,1} P_{2,1}^{-1}(U^T)^{-1},
\]
\[
b_\infty^T B_{x,t} b_1 = 1_m^T A_{x,t} \vec{\pi} = \mathbb{P}(X_t = x_t, \ldots, X_1 = x_1).
\]
While the first three expressions are straightforward to derive from assumptions, the last one requires a substantially different proof than that for the case \( n \geq m \). We prove this in Appendix A.

To cover both this case and the case \( T \) is of low rank, we modify Algorithm 1 by just one line:

**Algorithm 2.** Same as Algorithm 1, except that \( U \) is chosen as follows:

1. Generate \( U \) of size \( n \times \text{rank}(\hat{P}_{2,1}) \) such that \( \text{rank}(U^T \hat{P}_{2,1} \hat{P}_{2,1}^T U) = \text{rank}(\hat{P}_{2,1}) \).

### 4. Experiments on synthetic models

In experiments on synthetic mathematical models, our Algorithms 1 and 2 produce joint and conditional probability values that are: (1) always physically meaningful, i.e. between 0 and 1; (2) generally much closer to the true theoretical values than those produced by the original non-regularized methods, given an appropriate choice for \( \gamma > 0 \).

In the following experiments, we fixed \( \gamma = 10^{-7} \), which in our experience gave very good numerical results.

**Example 1:** \( m = 3, n = 5, k = 3 \). We randomly generated 20 sets of column-stochastic matrices \((T, O, N)\) such that \( \text{rank}(T) = m, \text{rank}(O) = m \). We used the \texttt{rand} routine from MATLAB, which generated numbers uniformly from the interval \([0,1]\). For each set of matrices, which define one HMM, we generated a sequence of triples \((x_1, x_2, x_3)^L_{i=1}\), where \( L = 5000 \), on which we built the observation matrices \((\hat{P}_1, \hat{P}_{2,1}, \hat{P}_{3,x,1})\). With the corresponding HMM, we generated 20 different sequences \((x_t)_{t=1}^N\), where \( N = 100 \). For each sequence \((x_t)_{t=1}^N\), we computed the corresponding sequence of conditional probabilities \((\hat{P}_n, (x_1, x_2, \ldots, x_{n-1}))_{t=1}^N\) using 20 random matrices \( U \). In Table 1, we report the errors of these probabilities compared to the true theoretical values, averaged over 400 sequences (20 sequences for each of 20 HMMs).

**Example 2:** \((m = 3, n = 5, k = 3)\) and \((m = 4, n = 5, k = 3)\). We carried out the same procedure, now with \( \text{rank}(T) = k, \text{rank}(O) = m \). These scenarios are not covered by [7], but are covered by [12].

**Example 3:** \((m = 6, n = 4, k = 2), (m = 6, n = 4, k = 3), \) and \((m = 6, n = 2, k = 4)\). We carried out the same procedure, now with \( \text{rank}(O) = n, \text{rank}(T) = k \). In [12], the authors proposed stacking several observations per time step. However, this was not clearly spelled out and in any case, without regularization, it would still suffer from numerical instability. The first two cases in fact are covered by the method of [12], still using single observations.

**Discussion:** One can see that the errors by the method in [12] are smaller when \( k = 2 \). This is because here we take the first 2 left singular vectors of \( \hat{P}_{2,1} \), which is more stable than taking the first 3 left singular vectors. This is in agreement with our discussion above in Section 2. In comparison, our method not only has much smaller errors, but the errors are very consistent across different scenarios.

### 5. Experiments on real data sets

In the experiments, we dealt with supervised classification tasks, where HMMs applied most consistently. We considered many datasets, each of them exhibiting a particular challenge (few training data, overlapped classes, noise, short sequences). In particular, we tested on the following datasets: 1) E. coli promoter gene sequences\(^1\), 2) Chicken Pieces [1], 3) TwoLeadECG, 4) Symbols, 5) FaceUCR, 6) 50words. Datasets 3-6 come from the UCR time series repository\(^2\), and are defined on continuous data, so we performed symbol quantization, choosing opportune the number of codewords. We tried different model topologies with varying number of hidden states. It is worth noting that our aim was not to reach state-of-the-art classification performances, but to thoroughly evaluate the relation between classical HMMs and spectral HMMs (SHMMs). In all cases SHMMs were shown to be strongly superior to classical HMMs.

#### 5.1. E. coli promoter gene sequences dataset

The standard task on this dataset is binary classification, i.e., recognizing promoters in strings of nucleotides (A, G, T, or C). A promoter is a genetic region which facilitates the transcription of genes located nearby. The dataset is composed of 106 strings (53 per class) of 57 sequential DNA nucleotides. The classification results are obtained using leave-one-out (LOO). We trained a generative model HMM (SHMM) once for each left-out sample. For each test point, the model is learned only on the training set consisting of all data but that point. As number of states, we tried with \( m = 2,3,4,5,6,7,8 \). Each experiment was repeated 50 times, to account for the variability of the results due to local minima (for classical HMMs) and the random U computation (for SHMMs). The results are reported in Table 2. As can be seen, SHMMs have higher performances for

<table>
<thead>
<tr>
<th>Parameters</th>
<th>[7]</th>
<th>[12]</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>((m = 3, n = 5, k = 3))</td>
<td>0.7770</td>
<td>0.7770</td>
<td>0.0063</td>
</tr>
<tr>
<td>((m = 3, n = 5, k = 2))</td>
<td>N/A</td>
<td>0.0350</td>
<td>0.0064</td>
</tr>
<tr>
<td>((m = 4, n = 5, k = 3))</td>
<td>N/A</td>
<td>0.5450</td>
<td>0.0065</td>
</tr>
<tr>
<td>((m = 6, n = 4, k = 2))</td>
<td>N/A</td>
<td>0.0729</td>
<td>0.0070</td>
</tr>
<tr>
<td>((m = 6, n = 4, k = 3))</td>
<td>N/A</td>
<td>1.4958</td>
<td>0.0073</td>
</tr>
<tr>
<td>((m = 6, n = 2, k = 4))</td>
<td>N/A</td>
<td>N/A</td>
<td>0.0071</td>
</tr>
</tbody>
</table>

\(^1\)Available at http://archive.ics.uci.edu/ml

Table 2. E. coli promoter gene sequences dataset.

<table>
<thead>
<tr>
<th>classifier</th>
<th>m</th>
<th>mean acc.(%) (std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>2</td>
<td>69.7 (1.7)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>70.6 (2)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>71.3 (2)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>70.6 (2)</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>71.9 (2.1)</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>71.1 (2.2)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>71.6 (1.6)</td>
</tr>
<tr>
<td>SHMM</td>
<td></td>
<td>77.5 (1.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76.9 (1.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77.3 (1.4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77.4 (1.3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77.4 (1.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77.6 (1.4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77.3 (1.5)</td>
</tr>
</tbody>
</table>

Table 3. Chicken Pieces database.

<table>
<thead>
<tr>
<th>met.</th>
<th>m=2</th>
<th>m=3</th>
<th>m=4</th>
<th>m=6</th>
<th>m=8</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>ML(%)</td>
<td>16.3</td>
<td>18.3</td>
<td>15.2</td>
<td>20.8</td>
</tr>
<tr>
<td></td>
<td>1NN(%)</td>
<td>46.6</td>
<td>31.4</td>
<td>22.0</td>
<td>20</td>
</tr>
<tr>
<td>SHMM</td>
<td>ML(%)</td>
<td>32.8</td>
<td>32.5</td>
<td>29.7</td>
<td>32.2</td>
</tr>
<tr>
<td></td>
<td>1NN(%)</td>
<td>49.4</td>
<td>54</td>
<td>47.9</td>
<td>50.9</td>
</tr>
</tbody>
</table>

Table 4. UCR time series datasets.

<table>
<thead>
<tr>
<th>dataset</th>
<th>classes</th>
<th># train</th>
<th># test</th>
<th>seq. length</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwoLeadECG</td>
<td>2</td>
<td>23</td>
<td>1139</td>
<td>82</td>
</tr>
<tr>
<td>Symbols</td>
<td>6</td>
<td>25</td>
<td>995</td>
<td>398</td>
</tr>
<tr>
<td>FaceUCR</td>
<td>14</td>
<td>200</td>
<td>2050</td>
<td>131</td>
</tr>
<tr>
<td>Adiac</td>
<td>37</td>
<td>390</td>
<td>391</td>
<td>176</td>
</tr>
<tr>
<td>50words</td>
<td>50</td>
<td>450</td>
<td>455</td>
<td>270</td>
</tr>
</tbody>
</table>

Table 5. UCR time series datasets.

<table>
<thead>
<tr>
<th>dataset</th>
<th>HMM</th>
<th>SHMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwoLeadECG</td>
<td>53%</td>
<td>80.9%</td>
</tr>
<tr>
<td>Symbols</td>
<td>16.4%</td>
<td>45.3%</td>
</tr>
<tr>
<td>FaceUCR</td>
<td>15.9%</td>
<td>44.6%</td>
</tr>
<tr>
<td>Adiac</td>
<td>15%</td>
<td>35.3%</td>
</tr>
<tr>
<td>50words</td>
<td>8.5%</td>
<td>14.5%</td>
</tr>
</tbody>
</table>

5.2. Chicken Pieces Database

This set consists of 446 binary images of chicken pieces. Each piece belongs to one of the five classes, representing specific parts of a chicken: wing (117 samples), back (76), drumstick (96), thigh and back (61), and breast (96). The shapes are usually first described by contours, which have been also quantized in sequences of 8 different levels of curvature. The conventional approach for classification is to train one HMM for each class and then apply Maximum Likelihood classification. Nevertheless, since the dataset has high intra-class variability, another classification routine has been applied in the literature: the idea is to train one model for each training sequence, assigning an unknown sequence to the class of the model showing the highest likelihood. This may be seen as a 1-nearest-neighbor (1-NN) classifier, with the proximity measure defined by the likelihood function. We report results for both strategies in Table 3, where SHMMs can be seen to be superior in all settings.

5.3. Other datasets

We selected five datasets dealing with computing vision and pattern recognition problems: ECG signal recognition (TwoLeadECG), stroke recognition (Symbols), person recognition through profile images (FaceUCR), shape recognition (Adiac), handwritten character recognition (50words). Each of them has inherent challenges, as can be seen in Table 4: small training set (TwoLead-ECG, Symbols), short sequences (TwoLeadECG); moreover, there are overlapped classes (Symbols, TwoLeadECG, FaceUCR, Adiac, 50words). The idea of the experiments was that of finding the best configuration in terms of number of states for classical HMMs, and test SHMMs on that setting. Furthermore, since the data is continuous, we performed symbol quantization employing k-means, selecting the number of symbols that ensured the highest HMM performance. Model selection (number of states/number of symbols) was performed on half the training dataset, spanning a range of 2-28 states and 2-28 symbols. After that, we performed classification employing LOO cross-validation. The results are shown in Table 5. In general, they are below state-of-the-art performance. This happened since we applied quantization, which eliminates much information from the data, diminishing classification performances. A very enlightening paper in this regards, focused on the UCR dataset, is [14]. Even here, SHMMs did very well, achieving performances which were two (Adiac, 50words) or three times (Symbols, FaceUCR) better that classical HMMs.

5.4. Computational complexity

Unlike the standard Baum-Welch algorithm for classical HMMs, SHMMs do not have parameter re-estimation and are not iterative (the runs over different Us are averaged), and thus do not have a convergence criterion. In our experiments, our method is often considerably faster than classical HMMs, typically by one order of magnitude. It also scales very well with large transition and emission matrices.

6. Conclusion and future work

We have presented a mathematically sound algorithm for learning probabilities in HMMs in the spectral framework. Our algorithm has been shown to outperform classical HMMs on many common data sets. Two major problems to consider for our future research are the problem of parameter estimation for discrete HMMs, and the generalization of the present framework to continuous HMMs.
References


A. Proofs

We provide the proof of Proposition 1 here, leaving all other proofs to the Supplementary Material.

Proof of Proposition 1. The first three expressions are straightforward. Let us prove the last:

\[ b_T \equiv B_{t \infty} b_1 = 1_{T_n} A_{x \infty} \Omega. \]

The case \( n = 1 \) is trivial, since then \( P_1 = P_{2,1} = P_{3,1} = 1 \) and \( \Omega(x_{1:1}) = 1 \). Thus we can assume that \( n \geq 2 \). By assumptions, the \( n \times n \) matrix \( P_{2,1} \) is of full rank, hence invertible. For any \( x \), \( 1 \leq x \leq n \), we have

\[ P_{x,1}^T 1_n = P_{2,1}^T (x, x). \]

\[ P_1^T = 1_{T_n} P_{2,1} \Leftrightarrow (P_{2,1})^{-1} P_1 = 1_n. \]

It will be easier for us to consider the quantity \((b_T \equiv B_{t \infty} b_1)\). We have

\[ (b_T \equiv B_{t \infty} b_1) = P_1^T (P_{2,1}^T)^{-1} \left( P_{3,1}^T P_{2,1}^T (P_{2,1}^T)^{-1} \right)^{t=1} P_1. \]

(39)

Consider for the moment the case \( t = 3 \). We have

\[ P_{3,1}^T (P_{2,1}^T)^{-1} P_1 = P_{2,1}^T (x, x_3) = \text{Odiag}(\Omega) T^T O_{x_3} 1_m. \]

Consider the next vector

\[ P_{3,2,1} (P_{2,1}^T)^{-1} \text{Odiag}(\Omega) T^T O_{x_3} 1_m. \]

The vector

\[ (P_{2,1}^T)^{-1} \text{Odiag}(\Omega) T^T O_{x_3} 1_m \]

is simply the \( x_3 \)th column of \( I_n \). Thus

\[ P_{3,2,1} (P_{2,1}^T)^{-1} \text{Odiag}(\Omega) T^T O_{x_3} 1_m \]

is simply the \( x_3 \)th column of \( P_{3,2,1}^T \), that is the \( x_3 \)th row of \( P_{3,2,1}^T \). By definition,

\[ P_{3,2,1} = O'_T O_{x_3} T \text{diag}(\Omega) O' T. \]

Thus

\[ P_{3,2,1} (x, x_3) = \text{Odiag}(\Omega) T^T O_{x_3} T^T O' T (x, x_3), \]

or equivalently,

\[ P_{3,2,1} (x, x_3) = \text{Odiag}(\Omega) T^T O_{x_3} T^T O_{x_3} 1_m. \]

Thus

\[ (P_{3,2,1} (P_{2,1}^T)^{-1})^3_{j=2} P_1 = \text{Odiag}(\Omega) T^T O_{x_3} T^T O_{x_3} 1_m. \]

For a general \( t \in \mathbb{N} \), by iteration, we obtain

\[ (P_{3,2,1} (P_{2,1}^T)^{-1})^T_{j=1} P_1 = \text{Odiag}(\Omega) T^T O_{x_3} 1_m. \]

(40)

Noting that \( 1_m = T^T 1_m \), we get an equivalent expression

\[ (P_{3,2,1} (P_{2,1}^T)^{-1})^T_{j=1} P_1 = \text{Odiag}(\Omega) T^T (O_{x_3} T^T)_{j=1}^T 1_m. \]

(41)

Our final expression is

\[ (b_T \equiv B_{t \infty} b_1)^T = P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\Omega) T^T (O_{x_3} T^T)_{j=1}^T 1_m. \]

(42)

By Corollary 1 below,

\[ (b_T \equiv B_{t \infty} b_1)^T = \text{O} T^T (O_{x_3} T^T)_{j=1}^T 1_m = \text{O} T^T (A_{x_3} T^T)_{j=1}^T 1_m. \]

Taking transpose gives us the final answer. \( \square \)
Lemma 5. Assuming the hypothesis of Proposition 1. For each \( 1 \leq x \leq n \),
\[
P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T O_x T^T = \vec{\pi}^T O_x T^T. \tag{43}\]

Corollary 1. Assuming the hypothesis of Proposition 1. For \( 1 \leq x_1, \ldots, x_t \leq n \),
\[
P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T (O_{x_j} T^T)_{j=1}^t = \vec{\pi}^T (O_{x_j} T^T)_{j=1}^t. \tag{44}\]

Proof. This follows immediately from Lemma 5. \( \square \)

Proof of Lemma 5. Consider three different cases.

First, the case \( n = m \). Then \( O \) and \( O_{P,2} \) are all invertible, thus
\[
P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T = \vec{\pi}^T O_{P,2} T^T. \]

It follows that
\[
P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T O_x T^T = \vec{\pi}^T O_x T^T. \]

Second, the case \( n < k = m \). Then \( T \) and \( P_{2,1} \) are both invertible, but \( O \) is not. We show that in this case we also have
\[
P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T = \vec{\pi}^T. \tag{45}\]

Now
\[
P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T = \vec{\pi}^T O_T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T. \]

Let \( f \in \mathbb{R}^m \) be such that
\[
D_m f = 1_{m+1},
\]
where
\[
D_m = \begin{pmatrix} O^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T & \vec{\pi}^T \end{pmatrix}. \]

This is a system of linear equations of size \((m + 1) \times m \), of rank at most \( n + 1 \), with at least one solution, namely \( f = 1_m \), since clearly \( \vec{\pi}^T 1_m = 1 \) and
\[
O^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T 1_m = O^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T 1_n = O^T (P_{2,1}^T)^{-1} 1_n = O^T 1_n = 1_m. \]

If rank\( (D_m) \leq n + 1 < m \), which always happens if \( n < k - 1 = m - 1 \), there are infinitely many \( f \) such that \( D_m f = 1_{m+1} \), with the form
\[
f = 1_m + n_{D_m}, \]
where \( n_{D_m} \in \text{nulspace}(D_m) \). Let \( F = [f_1, \ldots, f_m] \) be a matrix whose columns are \( m \) such vectors, which are linearly independent, then
\[
\vec{\pi}^T O_T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T F = \vec{\pi}^T F = 1^T.
\]

Since \( F \) is invertible, right multiplying by \( F^{-1} \) gives
\[
P_1^T (P_{2,1}^T)^{-1} \text{Odiag}(\vec{\pi}) T^T O_T F = \vec{\pi}^T O_T F = 1^T,
\]
as we claimed. This completes our proof. \( \square \)