MOX-Report No. 42/2019

hmmhdd Package: Hidden Markov Model for High Dimensional Data

Martino, A.; Guatteri, G.; Paganoni, A.M.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it http://mox.polimi.it

hmmhdd Package: Hidden Markov Model for High Dimensional Data

Andrea Martino[1], Giuseppina Guatteri[1] and Anna Maria Paganoni[1]

[1]Department of Mathematics, Politecnico di Milano, Milan, Italy

Abstract

The R package **hmmhdd** provides some tools to study times series and longitudinal datasets. In particular, the package is based on Hidden Markov Models, i.e. it considers an underlying structure defined by a Markov Model with non-observable states generating a certain type of data, in the multivariate or functional framework. In the former setting, a Gaussian copula models the correlation structure between the components of the observations while, in the latter setting, the data are multivariate functional data and the methods are based on distances between curves. The package is able to estimate all the parameters corresponding to the states of the underlying Markov model, while also computing the optimal state sequence and providing some further helpful tools.
Keywords: Functional Data; Hidden Markov Models; Multivariate Data.

1 Introduction

Hidden Markov Models (HMMs) are a very well-known method for the study of longitudal data or time series involving sequences of data, commonly used in many fields like biostatistics ([9]), finance ([15]) and signal processing ([5]). They consist in a generalization of mixture models, with the hidden variables being related through a Markov process (see [16] for further details). In the literature, HMMs are usually used to model univariate or multivariate data with independent components.

In this paper we present the **hmmhdd** package (Hidden Markov Models for High Dimensional Data), where some novel methods are applied both in the finite and the functional setting. For what concerns the multivariate HMM framework, there are examples of HMMs where the outcome is modelled using mixtures of multivariate Gaussian distributions (e.g. [3]) or multivariate distributions with independent nonnormal marginals, and there are some `R` packages able to do the task; see, e.g. `depmixS4` ([22]) and `msm` ([8]). Specifically, the **hmmhdd** package is able to study, using HMMs, two different types of data, by adding several novelties in both cases:

1. **Multivariate data**: the package extends the usual HMM algorithms in the multivariate framework, allowing a dependence between the components of the observations, which is done by means of gaussian copulas (see [12] for further details). A copula (see, e.g., [19], [14] and [10]) is a function that "couples" a multivariate distribution function to its marginal distribution functions, containing the information about the dependence structure between the components of a random vector. It is very useful since different marginals can be used, incorporating a flexible modelling of the dependence structure, without making any assumptions of normality.

2. **Functional data**: the package is able to study and model multivariate functional random curves, by extending the usual HMM algorithms from the finite dimensional framework to the infinite dimensional one (see [13] for further details). Therefore, this type of data falls into the well known Functional Data Analysis (FDA) setting (see, e.g. [17], [18], [6], [7]). The package models the functional data by considering a hidden Markov chain evolving in time, where each state emits a multivariate random curve and, estimating the parameters of the underlying Markov process, helps to understand the time series system that generated data.

The paper is organized as follows: in Section 2 we show how to set up a HMM using the package, for both multivariate and functional data; then, in Section 3 we present some background theory behind the model and algorithms, from the objective function computation to the parameter estimation and the computation of the optimal state sequence. In Section 4 and Section 5 we explain how to use the package to model multivariate and functional data, respectively, while in Section 6 and 7 we show how to use the package to estimate the optimal state sequence and make model selection.

2 Setting a Hidden Markov Model

A Hidden Markov Model [4] is a bivariate process $\{(Q_k, \boldsymbol{X}_k)\}_{k\geq 1}$ defined on a given probability space $(\Omega, \mathcal{F}, \mathbb{P})$ such that

- $\{Q_k\}_{k\geq 1}$ is a Markov chain with a discrete and finite state space $\{s_1, \ldots, s_N\}$, with $N \geq 1$, transition matrix $A = \{a_{ij}\} = \mathbb{P}(Q_k = s_j | Q_{k-1} = s_i)$ and initial distribution $\boldsymbol{\nu}$, where $\nu_i = \mathbb{P}(Q_1 = s_i)$;

- for each time k, the observation $\boldsymbol{X}_k$ is a d-dimensional random array. In particular, given the state process $\{Q_k\}_{k\geq 1}$, $\boldsymbol{X}_k$ is a sequence of conditionally independent random arrays (vectors or matrices, depending on the type of data).

The first step for defining a HMM in the **hmmhdd** package consists in defining an object corresponding to the **S3** classes `mhmm` or `fhmm`, depending on the data being vectors or functions. These classes can be used to characterize a HMM and they are the initialization for the other functions that will be used for further analysis. Both classes require `bT`, i.e. the beginning times of each observation for every statistical unit, the number of states `nStates`

of the HMM, the vector of the initial probabilities `nu` and the transition matrix `A` of the HMM.

When dealing with a multivariate dataset, `Obs` is a matrix with n rows and d columns, where n is the number of observations while d is the number of components of the observations. The function also admits the values of the correlation matrices `corr` for the gaussian copula, passed as a N-dimensional array of matrices, the values of the state-dependent parameters `params`, to be passed as a list of matrices, and a vector of strings `distr` specifying the distribution for each component. If the dataset is functional, `Obs` is a `funData` object from the `gmfd` package, i.e. a list containing the grid $I = [t_0, t_1, \ldots, t_{P-1}]$ over which the data is defined and the values of the observations in the functional dataset, provided as a matrix with the rows as observations and the columns as their measurements over the grid of length P. Moreover, the function requires `centroids`, i.e. the functional parameters representing the functional means of each state of the HMM, passed as a `funData` where the first element is again the grid over which the data is defined and the others are N x P matrices, one for each component of the data. The package contains two simulated dataset, denoted as `copuladata` and `copulahmmdata`, to test the functions for the multivariate case and one denoted as `simulatedFD` to test the functions in the functional case. A call's example for a multivariate model, using the simulated data `copulahmmdata` in the package, with some data visualization given using the `summary` function is the following:

```
data(copulahmmdata)
Obs <- copulahmmdata
n <- 20 #number of observations per statistical unit
n_tot <- dim(Obs)[1]
bt <- seq(1, n_tot, by = n) #beginning times for each statistical unit
distr <- c("exp", "gaussian")

#Initialize the HMM
hmm <- set_mhmm(Obs, bT = bt, nStates = 2, distr = distr)
summary(hmm)

Initial state probabilities:
S_1 S_2
0.5 0.5

Transition matrix:
toS_1 toS_2
fromS_1   0.5   0.5
fromS_2   0.5   0.5

Response 1 :Exponential
Response 2 :Gaussian
```

```
Response parameters
rate.Comp_1 mean.Comp_2 sd.Comp_2
S_1   0.5006811   -1.998299  1.005719
S_2   2.0303610    3.988309  1.006424
Correlation matrix for State 1
[,1]      [,2]
[1,] 1.0000000 0.1469427
[2,] 0.1469427 1.0000000
Correlation matrix for State 2
[,1]      [,2]
[1,] 1.0000000 0.6044778
[2,] 0.6044778 1.0000000
```

where the initial state probabilities and the transition matrix have a default initialization, where each term is set to $1/$`nStates` while the response parameters and the correlation matrices are initialized using the `kmeans` function. where we are not setting any initialization for the state-dependent parameters. More details will be given later, where we will cover the case with functional response too.

The implementation of **S3** classes allows us to use typical methods like `summary` and `plot` to help visualizing outputs of the HMM. Specifically, for both classes we implemented a method for visualizing all the elements of the HMM, `summary.mhmm` and `summary.fhmm` respectively. Moreover, in the case of data with functional response, it is possible to visualize all the data and the curves corresponding to the states with the function `plot.fhmm`; in particular, in this case, the graphical window is split in a rectangular lattice so that each component is plotted singularly, with each state represented with a different colour.

3 Fitting a Hidden Markov Model

After the initialization of a HMM, there are usually three fundamental problems (see for instance [16] and [25]) to be solved:

1. find the objective function $\mathcal{L}(\lambda|\boldsymbol{x})$ for the observations $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K)$. In particular, if the data is multivariate, this is a likelihood function for the data;

2. given some $\boldsymbol{x}$ and λ, find the optimal state sequence $Q = (Q_1, \ldots, Q_K)$ that explains $\boldsymbol{x}$;

3. find $\lambda^* = \underset{\lambda}{\operatorname{argmax}}\, \mathcal{L}(\lambda|\boldsymbol{x})$, i.e. the best parameters of the models that maximize the objective function.

3.1 Objective function computation

First, we have to define the objective function of a HMM, which in general can be written as

$$\log(\mathcal{L}(\lambda|\mathbf{x})) = \underbrace{\sum_{i=1}^{N} \gamma_1(i) \log \nu_i}_{\text{term 1}} + \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\sum_{k=1}^{K-1} \xi_k(i,j)\right)\log a_{ij}}_{\text{term 2}} + \underbrace{\sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_k(i) \log b_i(\mathbf{x}_k;\boldsymbol{\theta}_i)}_{\text{term 3}}. \tag{1}$$

where $\xi_k(i,j) = \mathbb{P}(Q_k = s_i, Q_{k+1} = s_j \mid \mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_k = \mathbf{x}_k, \lambda)$ is the probability of being in state s_i at time k and state s_j at time $k+1$, given the model and the observations, $\gamma_k(i) = \mathbb{P}(Q_k = s_i \mid \mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_k = \mathbf{x}_k, \lambda)$ is the probability of being in the state s_i at time k, while $b_i(\mathbf{x}_k;\boldsymbol{\theta}_i)$ is the emission function of $\mathbf{X}_k$ conditionally on the event $\{Q_k = s_i\}$ for any $i = 1, \ldots, N$. To estimate the value of the function in (1), we perform the so called forward-backward algorithm. If we define the forward quantities, i.e. the probability density functions of observing the partial sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ and ending in the state Q_k, given the model λ, as

$$\alpha_k(j) \;=\; f_{(\boldsymbol{X}_1,\ldots,\boldsymbol{X}_k,Q_k)|\lambda}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, s_j),$$

then we can find the objective function by solving for $\alpha_k(j)$ inductively, as follows:

(1) $\alpha_1(i) = \nu_i b_i(\mathbf{x}_1;\boldsymbol{\theta}_i)$, for $1 \le i \le N$;

(2) $\alpha_{k+1}(i) = \left[\sum_{j=1}^{N} \alpha_k(j) a_{ji}\right] b_i(\mathbf{x}_{k+1};\boldsymbol{\theta}_i)$, for any $1 \le k \le K-1$ and $1 \le i \le N$.

The objective function can be computed as $\mathcal{L}(\lambda|\mathbf{x}) = \sum_{j=1}^{N} \alpha_K(j)$. The backward procedure is similar (see [16] for further details) and it is implemented along with the forward one in the `forwardbackward` function of the package.

3.2 Parameter estimation

The forward-backward algorithm is implemented together with the Baum-Welch algorithm, to fit a HMM and estimate all its parameters starting from the data [16]. Parameters are estimated in **hmmhdd** using the expectation-maximization (EM) algorithm. In particular, in the HMM setting, the algorithm is known as Baum-Welch algorithm (see for instance [1], [2], [24], [3]).

Starting from the expression (1), it is possible to perform the EM algorithm for HMMs by solving iteratively the two steps:

- **E step** replace the quantities $\xi_k(i,j)$ and $\gamma_j(k)$ by their conditional expectations given the current parameter estimates and the observations (see, e.g. [16])

- **M step** maximize the logarithm of the objective function in (1). Each term of the expression depends on different parameters, so it can be split into three parts in order to maximize each term separately. The maximization of the first two terms of the expression is done as in [16], while the maximization of third term depends on the type of data of the response.

The **hmmhdd** package is designed to model longitudinal datasets or time series with several observations for each statistical unit; until now, all the formulas only considered a single observation sequence. Since the package models multiple sequences, we denote the set of observation sequences as

$$\mathcal{X} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)})$$

for M statistical units, each one of them being a sequence of length K_m, $m = 1, \ldots, M$. Then, $\boldsymbol{x}^{(m)} = (\boldsymbol{x}_1^{(m)} \boldsymbol{x}_2^{(m)} \cdots \boldsymbol{x}_{K_m}^{(m)})$ is the m-th observation sequence of length K; all the sequences are considered independent from each other. The main goal consists in adjusting the parameters of the model λ to maximize the following likelihood:

$$\mathcal{L}(\lambda|\mathcal{X}) = \prod_{m=1}^{M} \mathcal{L}(\lambda|\boldsymbol{x}^{(m)}). \tag{2}$$

The use of the package `hmmhdd` for parameter estimation mainly consists of two steps:

1. model initialization, as described in Section 2, through the use of the `setHMM_mhmm` and `setHMM_fhmm` functions;

2. model fitting, through the use of the algorithms just described, with the functions `fitBM_mhmm` and `fitBM_fhmm`.

4 Multivariate Hidden Markov Model

Throughout this work, two simulated datasets will be used. For what concerns the multivariate case, the dataset is the `copulahmmdata`. Specifically, the dataset was simulated starting from a Markov Model with 2 states and the following parameters:

$$\nu = (1 \quad 0), \qquad A = \begin{pmatrix} 0.70 & 0.30 \\ 0.10 & 0.90 \end{pmatrix},$$

with 20 observations for 250 statistical units. The process is bivariate, with an exponential and a gaussian component. Each state emits realizations of the joint distribution (X_1, X_2) where, for every state i, we have:

- $X_1|s_i \sim \mathcal{E}(\lambda_i)$;
- $X_2|s_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

In particular, the parameters of the first state are $\lambda_1 = 2, \mu_1 = 4, \sigma_1 = 1$ while the parameters of the second state are $\lambda_2 = 0.5, \mu_2 = -1, \sigma_2 = 1$. Moreover, the components of the first state have a stronger correlation ($\rho_1 = 0.7$) than the ones related to the second state ($\rho_2 = 0.15$). The data was generated using the `rmdistr` function of the `hmmhdd` package, which returns a multivariate sample after taking as arguments the number `n` of observations, a list of the distribution parameters `params`, an array containing the true correlation matrices `corr` and a vector of strings `distr`, for the names of the distribution for each component (the allowed options are the exponential, the gaussian and the gamma distribution, represented by `exp`, `gaussian` and `gamma`, respectively). The package does not necessarily require any initialization, since the transition matrix `A` and the vector `nu` have a default initialization, with every element set to 1/`nStates` while the parameters are initialized with a k-means algorithm, performed with the `kmeans` function of the `stats` package. As seen before, the Baum-Welch algorithm requires the computation of the emission functions, which represent the likelihood of emitting a certain observation by a certain state. In our case, since the response is multivariate but we want to consider the correlation between the components, we implemented a gaussian copula model using a copula density function c [12]. In general, for a d-variate data, the emission function related to state s_j for the observation $\mathbf{x}_k$, can be written as

$$b_j(\boldsymbol{x}_k; \boldsymbol{\theta}_j) \;=\; c[F_{1j}(x_{1k}; \boldsymbol{\theta}_{1j}), \ldots, F_{dj}(x_{dk}; \boldsymbol{\theta}_{dj});\, \rho_j] \cdot \prod_{i=1}^{d} f_{ij}(x_{ik}; \boldsymbol{\theta}_{ij})$$

where f_{ij} and F_{ij}, $i = 1, \ldots, d$, are the pdf and the cdf of a given distribution. Therefore, taking into account the equations (1) and (2), the proper estimates of the model parameters can be found by maximizing the following quantity:

$$\sum_{m=1}^{M} \sum_{k=1}^{K_m} \sum_{j=1}^{N} \gamma_k(j) \left(\log c\Big[F_{1j}(x_{1k}; \lambda_{1j}), \ldots, F_{dj}(x_{dk}; \lambda_{dj});\, \rho_j\Big] + \sum_{i=1}^{d} \log f_{ij}(y_{1k}; \boldsymbol{\theta}_{dj}) \right)$$

To compute the values of the emission functions, we implemented in the package the function `dmdistr` that takes as arguments the matrix of the data `x`, a list of the parameters `params`, an array of correlation matrices `corr` and a vector of strings for the distributions of each components, returning a numeric value corresponding to the value of the multivariate density function that uses gaussian copulas to model the correlation between the marginals.

To set the right class of this model, which is `mhmm` for multivariate data, and estimate its parameters, a call's example along with its output is the following:

```
data(copulahmmdata)
Obs <- copulahmmdata
n <- 20 #number of observations per statistical unit
n_tot <- dim(Obs)[1]
bt <- seq(1, n_tot, by = n) #beginning times for each statistical unit
distr <- c("exp", "gaussian")
```

```
#Initialize the HMM
hmm <- set_mhmm(Obs, bT = bt, nStates = 2, distr = distr)

#Parameter estimation
bw <- fitBM_mhmm(hmm)
Initial state probabilities:
S_1         S_2
0.995151673 0.004848327
Transition matrix:
toS_1     toS_2
fromS_1 0.70010846 0.2998915
fromS_2 0.09917526 0.9008247

Response 1: Exponential
Response 2: Gaussian

Response parameters
rate.Comp_1 mean.Comp_2 sd.Comp_2
S_1   2.0296898    4.005769  0.988701
S_2   0.5053992   -1.988103  1.022097
Correlation matrix for State 1
[,1]     [,2]
[1,] 1.000000 0.622236
[2,] 0.622236 1.000000
Correlation matrix for State 2
[,1]     [,2]
[1,] 1.000000 0.138430
[2,] 0.138430 1.000000
```

where `bw` contains the output of the function, i.e. all the estimations of the parameters. Although it is advised to initialize the data if possible to reduce convergence problems, since these methods are guaranteed to find a local maximum, all the obtained estimates using the gaussian copula are quite accurate. The code snippet shows first the parameters related to the Markov Model, where `S_1` and `S_2` represent the states of the HMM, and then the parameters related to the response, along with the correlation matrices of the gaussian copula for each state.

5 Functional Hidden Markov Model

If dealing with a functional dataset, like the one implemented in the package `simulatedFD`, we have to set a different class and compute some of the quantities in a different way. This dataset was generated using the `R` package **roahd** [20], starting from a 3-state Markov model

with the following parameters:

$$\nu = (1 \quad 0 \quad 0), \qquad A = \begin{pmatrix} 0.60 & 0.30 & 0.10 \\ 0.10 & 0.80 & 0.10 \\ 0.00 & 0.00 & 1.00 \end{pmatrix},$$

$$\boldsymbol{\theta}_1(t) = \begin{pmatrix} t(1-t) \\ 2t \end{pmatrix}; \quad \boldsymbol{\theta}_2(t) = \begin{pmatrix} t^2(1-t) \\ t^2 \end{pmatrix}; \quad \boldsymbol{\theta}_3(t) = \begin{pmatrix} t(1-t)^2 \\ \frac{1}{2}t^3 \end{pmatrix},$$

with 20 observations for 100 statistical units. Each functional data consists of a bivariate random curve with independent components, defined on a grid of $P = 100$ points, with $\boldsymbol{\theta}_1(t), \boldsymbol{\theta}_2(t), \boldsymbol{\theta}_3(t)$ being the means of each multivariate random curves for state s_1, s_2 and s_3, respectively. A call's example, where we initialize the functional HMM and then estimate all the parameters is given by:

```
data( simulatedFD )
FD <- simulatedFD
n <- 20 #number of observations per statistical unit
n_tot <- dim( FD$data[[1]] )[1]
bt <- seq(1, n_tot, by = n) #beginning times for each statistical unit
bt <- seq( 1, n_tot, by = n )
# Initialize the HMM
hmm <- set_fhmm( FD, nStates = 3, bT = bt )

#Parameter estimation
bw <- fitBM_fhmm(hmm)
summary(bw)
Initial state probabilities:
S_1          S_2          S_3
1.000000e+00 3.002669e-50 5.263375e-65

Transition matrix:

toS_1        toS_2      toS_3
fromS_1 6.955202e-01 1.882253e-01 0.11625450
fromS_2 4.809883e-02 8.548413e-01 0.09705983
fromS_3 1.187856e-26 1.529422e-09 1.00000000
```

In this case, the initial probabilities `nu` and the transition matrix `A` are still set to default, with each element being equal to `1/nStates`. If no initialization is given for the `centroids`, then a functional k-means is performed using the `gmfd_kmeans` function of the **gmfd** package [11]. The algorithm alternates a step where each multivariate curve is assigned to a state with a step where the centroids of the states are computed, until convergence is reached (see [21] for further details).

After this initialization step, the Baum-Welch algorithm is performed to compute all the parameters of the Markov model and the new functional parameters representing the states. To be able to do that, it is necessary to compute the emission functions $b_i(\,\cdot\,;\boldsymbol{\theta}_i)$, $i = 1, \ldots, N$ for each functional response. Let us recall the definition of L^2 distance between two multivariate curves $\mathbf{a}$ and $\mathbf{b}$ defined on I, compact interval of $\mathbb{R}$:

$$d_{L^2}(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\| = \sqrt{\sum_{l=1}^{d} \int_I (a_l(t) - b_l(t))^2 dt}.$$

Therefore, in the `fitBM_fhmm` function, the computation of the emission functions related to the states of the HMM is performed as

$$b_i(\,\cdot\,;\boldsymbol{\theta}_i) = h\big(d(\,\cdot\,, \boldsymbol{\theta}_i)\big), \qquad i = 1, \ldots, N \tag{3}$$

where $h(y)$ is a function that transforms the distance in a similarity measure; in this case, we have chosen $h(y) = 1/y^2$, but other choices for the function are possible. By recalling equations 1 and 2, in the case of functional data, to estimate the functional parameters of the HMM, the package maximizes the following quantity:

$$\sum_{m=1}^{M} \sum_{k=1}^{K_m} \sum_{j=1}^{N} \gamma_k(j) \log b_i(\mathbf{x}_k; \boldsymbol{\theta}_j).$$

For further details regarding this method, see [13]. Since the output of the `fitBM_fhmm` function is an object of class `fhmm`, we implemented a `summary` method to be used for this class. As we can see from the output, the estimates of the values related to the initial probabilities `nu` and the transition matrix `A` are quite accurate. Other than these parameters, which are just related to the Markov structure underlying the succession of states, we can also obtain some estimates for the functional means $\boldsymbol{\theta}_1(t), \boldsymbol{\theta}_2(t), \boldsymbol{\theta}_3(t)$ generating the data. Since they are functional data, for a better visualization we implemented a function based on the `plot` method for objects belonging to the `fhmm` classes. In Fig. 1, we can see the output related to the call `plot(bw)`.

6 Optimal state sequence

When dealing with HMMs, a problem that is usually considered consists in finding the optimal state sequence $Q = (Q_1, \ldots, Q_K)$ that generated the observations. The most popular approach to solve this problem is based on the Viterbi algorithm, see [23]. Let us fix an observed sequence $q_1, \ldots, q_k$ and define the quantity

$$\delta_k(i) = \max_{q_1, \ldots, q_{k-1}} \mathbb{P}(Q_1 = q_1, \ldots, Q_{k-1} = q_{k-1}, Q_k = s_i, \mathbf{x}_1, \ldots, \mathbf{x}_k | \lambda). \tag{4}$$

which is the highest probability on a single path at time k by taking into account the partial sequence $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Since we need to retrieve the state sequence, we have to keep track of the argument that maximizes (4) for each k and j, through the array $\psi_k(j)$. In particular, the procedure can be divided into four steps:

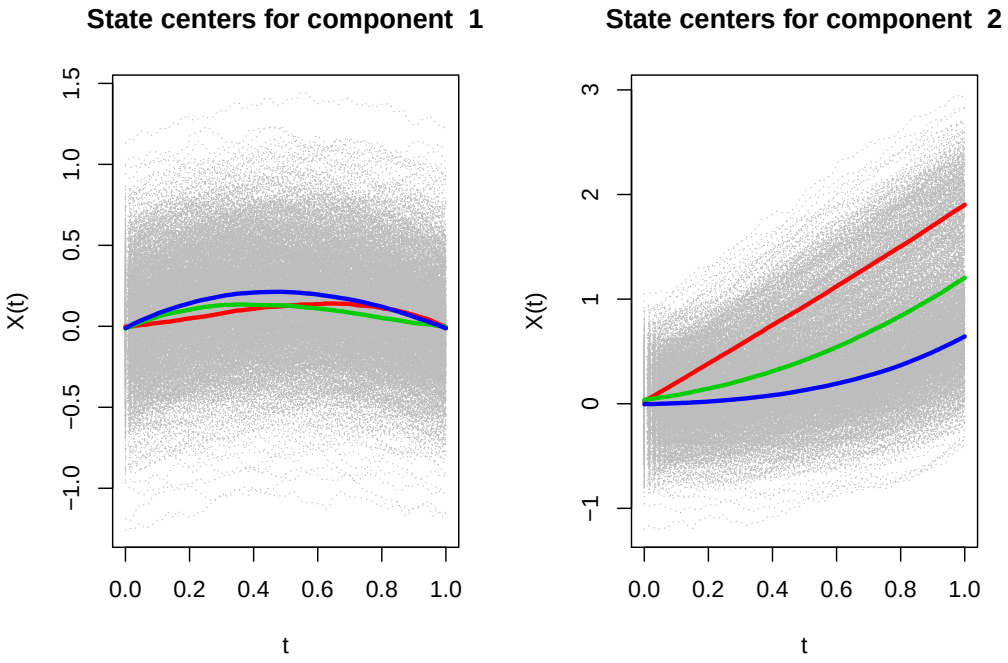

Figure 1: Plot of the functional parameters related to the states of the HMM, using the function `plot.fhmm`.

1. Initialization: $\delta_1(i) = \nu_i b_{\mathbf{X}_k|Q_k=s_i}(\mathbf{x}_1; \boldsymbol{\theta}_i)$ and $\psi_1(i) = 0$ for $1 \leq i \leq N$.

2. Recursion:

$$\delta_k(j) = \max_{1 \leq i \leq N} [\delta_{k-1}(i) a_{ij}] b_{\mathbf{X}_k|Q_k=s_j}(\mathbf{x}_k; \boldsymbol{\theta}_i), \quad 1 \leq k \leq K, \quad 1 \leq j \leq N;$$
$$\psi_k(j) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_{k-1}(i) a_{ij}], \quad 1 \leq k \leq K, \quad 1 \leq j \leq N.$$

3. Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_K(i)];$$
$$q_K^* = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_K(i)].$$

4. State sequence backtracking: $q_k^* = \psi_{k+1}(q_{k+1}^*), \quad k = K-1, K-2, \ldots, 1.$

Except for the last step, we can notice this procedure is similar to the forward one, where we have a maximization step instead of the summing one. By following this procedure, we can retrieve the optimal state sequence q_k^*, $k = 1, \ldots, K$. (see [16] for further details).If `bw` is an object of class `mhmm` or `fhmm`, then the state sequence can be computed with the function `viterbi` as follows:

```
viterbi(bw)
[1] 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2
[26] 2 2 1 1 1 1 3 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 3 3
[51] 3 3 3 3 3 3 3 3 3 3 2 2 1 1 1 3 3 3 3 3 3 3 3 3 3
[76] 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
...
```

where in this case we are referring to the `bw` object obtained in Section 5. As we can see from the output, the function `viterbi` returns a vector of numbers, each one being a label for each observation.

7 Model Selection

Since the number of states for a HMM is not usually known a priori, it is useful to use some model selection criteria to be able to compare some model results and choose the one that performs better. In this package, we implemented two criteria often used in the HMM framework, AIC and BIC, to perform model selection; see, e.g., [25]. Given a dataset with n statistical units and p unknown parameters, the two quantities are computed as:

$$\text{AIC} = -2\log(\mathcal{L}(\lambda|\mathbf{x})) + 2p \qquad \text{BIC} = -2\log(\mathcal{L}(\lambda|\mathbf{x})) + p\log(n), \tag{5}$$

with the functions `aic` and `bic`, respectively. Let us consider the functional dataset just used in Section 5. If we want to compare some HMMs to choose the optimal number of

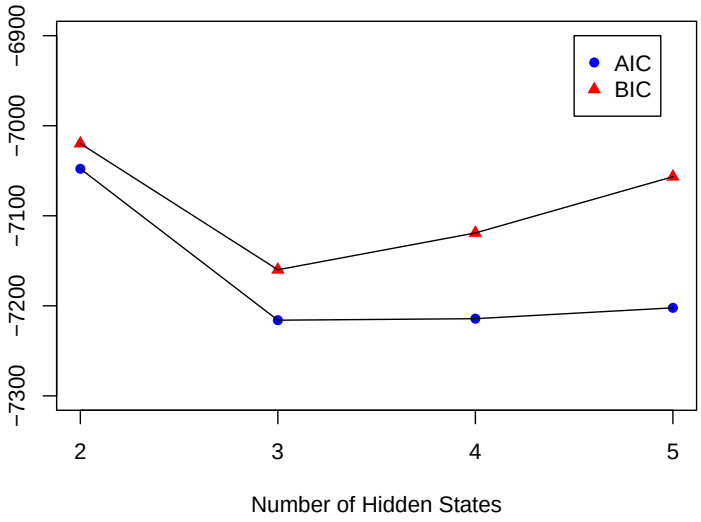

Figure 2

states, we have to use the `aic` and `bic` functions for each one of it. In particular, we can see in Fig. 2 the results for $n = 2, \ldots, 6$ number of states. The smallest number for both criteria is for $n = 3$, which represents the optimal number of states to model the data. The calls to both of these functions are also included inside the `summary.mhmm` and `summary.fhmm` functions of the package.

8 Conclusion

In this work, we presented the `R` package `hmmhdd` (Hidden Markov Models for High-Dimensional Data) that we developed for the study of both multivariate and functional data using HMMs. In particular, our package can deal with high-dimensional datasets, with little a priori knowledge on data, and provides useful tools for describing the time series system of data. `hmmhdd` is available on the Comprehensive R Archive Network (CRAN), with current release version 1.0, at https://cran.r-project.org/web/packages/hmmhdd and the source code is fully documented and commented.

References

[1] L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8, 1972.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[3] J. A. Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[4] O. Cappé, E. Moulines, and T. Ryden. *Inference in Hidden Markov Models (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[5] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk. Wavelet-based statistical signal processing using hidden markov models. *IEEE Transactions on signal processing*, 46(4):886–902, 1998.

[6] F. Ferraty and P. Vieu. *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media, 2006.

[7] L. Horváth and P. Kokoszka. *Inference for functional data with applications*, volume 200. Springer Science & Business Media, 2012.

[8] C. H. Jackson. Multi-state models for panel data: The msm package for R. *Journal of Statistical Software*, 38(8):1–29, 2011.

[9] C. H. Jackson and L. D. Sharples. Hidden markov models for the onset and progression of bronchiolitis obliterans syndrome in lung transplant recipients. *Statistics in medicine*, 21(1):113–128, 2002.

[10] H. Joe. *Multivariate models and multivariate dependence concepts*. Chapman and Hall/CRC, 1997.

[11] A. Martino, A. Ghiglietti, F. Ieva, and A. M. Paganoni. *gmfd: Inference and Clustering of Functional Data*, 2018. R package version 1.0.1.

[12] A. Martino, G. Guatteri, and A. M. Paganoni. Multivariate hidden markov models for disease progression. *Mox Report 59/2018*, 2018.

[13] A. Martino, G. Guatteri, and A. M. Paganoni. Hidden markov models for multivariate functional data. *Mox Report 20/2019*, 2019.

[14] R. B. Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007.

[15] L. J. Paas, J. K. Vermunt, and T. H. Bijmolt. Discrete time, discrete state latent markov modelling for assessing and predicting household acquisitions of financial products. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(4):955–974, 2007.

[16] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[17] J. O. Ramsay. Functional data analysis. *Encyclopedia of Statistical Sciences*, 4, 2004.

[18] J. O. Ramsay and B. W. Silverman. *Applied functional data analysis: methods and case studies*. Springer, 2007.

[19] M. Sklar. Fonctions de repartition an dimensions et leurs marges. *Publ. inst. statist. univ. Paris*, 8:229–231, 1959.

[20] N. Tarabelloni, A. Arribas-Gil, F. Ieva, A. M. Paganoni, and J. Romo. *roahd: Robust Analysis of High Dimensional Data*, 2018. R package version 1.4.1.

[21] T. Tarpey and K. K. Kinateder. Clustering functional data. *Journal of classification*, 20(1):093–114, 2003.

[22] I. Visser and M. Speekenbrink. depmixS4: An R package for hidden markov models. *Journal of Statistical Software*, 36(7):1–21, 2010.

[23] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[24] L. R. Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):10–13, 2003.

[25] W. Zucchini, I. L. MacDonald, and R. Langrock. *Hidden Markov models for time series: an introduction using R*. Chapman and Hall/CRC, 2016.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

41/2019 Abbà, A.; Bonaventura, L.; Recanati, A.; Tugnoli, M.;
Dynamical p-adaptivity for LES of compressible flows in a high order DG framework

39/2019 Lovato, I.; Pini, A.; Stamm, A.; Taquet, M.; Vantini, S.
Multiscale null hypothesis testing for network-valued data: analysis of brain networks of patients with autism

40/2019 Lovato, I.; Pini, A.; Stamm, A.; Vantini, S.
Model-free two-sample test for network-valued data

38/2019 Massi, M.C.; Ieva, F.; Gasperoni, F.; Paganoni, A.M.
Minority Class Feature Selection through Semi-Supervised Deep Sparse Autoencoders

36/2019 Salvador, M.; Dede', L.; Quarteroni, A.
An intergrid transfer operator using radial basis functions with application to cardiac electromechanics

37/2019 Menafoglio, A.; Secchi, P.
O2S2: a new venue for computational geostatistics

35/2019 Zancanaro, M.; Ballarin, F.; Perotto, S.; Rozza, G.
Hierarchical model reduction techniques for flow modeling in a parametrized setting

33/2019 Regazzoni, F.; Dede', L.; Quarteroni, A.
Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics

34/2019 Antonietti, P. F.; Mazzieri, I.; Melas, L.; Paolucci, R.; Quarteroni, A.; Smerzini, C.; Stupazzini,
Three-dimensional physics-based earthquake ground motion simulations for seismic risk assessment in densely populated urban areas

32/2019 Fedele, M.
Polygonal surface processing and mesh generation tools for numerical simulations of the complete cardiac function.