DBHC

Discrete Bayesian HMM Clustering

Budel, Gabriel; Frasincar, Flavius; Boekestijn, David

DOI
10.1007/s13042-024-02102-w

Publication date
2024

Document Version
Final published version

Published in
International Journal of Machine Learning and Cybernetics

International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-024-02102-w

ORIGINAL ARTICLE

DBHC: Discrete Bayesian HMM Clustering

Gabriel Budel[1,2] · **Flavius Frasincar**[1] · **David Boekestijn**[1]

Received: 23 December 2022 / Accepted: 15 January 2024

Abstract
Sequence data mining has become an increasingly popular research topic as the availability of data has grown rapidly over the past decades. Sequence clustering is a type of method within this field that is in high demand in the industry, but the sequence clustering problem is non-trivial and, as opposed to static cluster analysis, interpreting clusters of sequences is often difficult. Using Hidden Markov Models (HMMs), we propose the Discrete Bayesian HMM Clustering (DBHC) algorithm, an approach to clustering discrete sequences by extending a proven method for continuous sequences. The proposed algorithm is completely self-contained as it incorporates both the search for the number of clusters and the search for the number of hidden states in each cluster model in the parameter inference. We provide a working example and a simulation study to explain and showcase the capabilities of the DBHC algorithm. A case study illustrates how the hidden states in a mixture of HMMs can aid the interpretation task of a sequence cluster analysis. We conclude that the algorithm works well as it provides well-interpretable clusters for the considered application.

Keywords Sequence data mining · Sequence clustering · Mixture hidden Markov models · Graphical models · Probability smoothing

1 Introduction

Clustering sequence data is inherently different than clustering static data. One of the main reasons for this is the intransitivity of the problem [5]. For example, if two sequences AAAA and BBB are regarded to be similar to AAAABBB, it does not mean that AAAA is also similar to BBB [5]. Intransitivity makes the sequence clustering problem non-trivial. However, by using an adjusted distance function that incorporates the sequential structure it becomes possible to use traditional clustering methodology for sequence data [13]. A special type of distance function is the probabilistic distance function: using a probability distribution to describe differences between clusters, which is the main idea behind mixture models (in general) [13]. Using a mixture of Hidden Markov Models (HMMs), the model can also account for differences in sequence-time variation across clusters, as demonstrated by e.g. Lagona et al. [11].

Because of the dimensionality of the problem and the dynamic nature of sequence data, it is often difficult to interpret clusters of sequences [15]. This is a major drawback for analysis of this data type, because the main goal of cluster analysis is usually description, not prediction [23]. For the purpose of solving this problem, visualising cluster patterns in sequence data might aid the interpretation task. Cadez et al. [3] consider sequences where the observations are assumed to develop according to a first-order Markov process, i.e., observations are assumed to be drawn from a distribution that only depends on the previous observation in a sequence. They developed a tool that visualises discrete sequence observations for each cluster using colour coding. In this case the sequential ordering in the data is not a drawback of the problem setting anymore, it helps interpreting the clusters. Clustering sequences using multiple instances of an HMM may also aid the interpretation task. An HMM allows for modelling dynamic processes with an unobserved variable that traverses through different states, which are

✉ Flavius Frasincar
frasincar@ese.eur.nl

Gabriel Budel
g.j.a.budel@tudelft.nl

David Boekestijn
boekestijn@ese.eur.nl

[1] Erasmus University Rotterdam, PO Box 1738, 3000 DR Rotterdam, The Netherlands

[2] Delft University of Technology, PO Box 5031, 2600 GA Delft, The Netherlands

Published online: 27 February 2024

called 'hidden' states [19]. In this model, not the observations, but the hidden states are assumed to follow a first-order Markov process. Visualising the estimated probabilities of such a model in heatmaps can also help understand differences between clusters.

In the past, HMMs have been used for clustering and modelling continuous sequence data [7, 12]. This paper contributes to the literature as follows. We propose an approach to clustering discrete sequences using HMMs: the Discrete Bayesian HMM Clustering (DBHC) algorithm, an extension of the Bayesian HMM Clustering algorithm of Li and Biswas [12] to discrete sequence data. The algorithm incorporates the searches for the number of clusters and for the number of hidden states in each cluster model, two classical problems that one is faced with when working with mixture models and HMMs [14, 20]. We follow the approach of Li and Biswas [12] and incorporate the searches for these unknown parameters in the model by casting them as Bayesian model selection problems. In the work of Li and Biswas [12], clusters are iteratively added to the total partition, while each new cluster initially contains a carefully selected set of observations. This set of observations is called a 'seed', and the procedure that initialises the cluster models by evaluating them on a seed is called the 'seed selection procedure'. Since the method of Li and Biswas [12] is intended for continuous observations only, we extend the seed selection procedure to be applicable for discrete observations by adding a probability smoothing step. To the best of our knowledge, the framework of Li and Biswas [12] has never been extended to discrete sequence data classification. Furthermore, we propose an implementation of the DBHC algorithm in the R statistical software [16], which can be readily used.

The remainder of this paper is structured as follows. First, related work in this field is described in Sect. 2. Second, Sect. 3 describes the model underlying the mixture used for sequence clustering in the DBHC algorithm: the discrete-output HMM. Then follows a detailed description of the methodology of the DBHC algorithm in Sect. 4. A working example of the proposed algorithm is provided in Sect. 5. In Sect. 6, a simulation study and a case study of sequence clustering with the DBHC algorithm are discussed. Last, a summary of the findings in this paper and concluding remarks are provided in Sect. 7.

2 Related work

In the current literature, most applications of sequence clustering can be found in the fields of biology and Web mining. Dong and Pei [5] provide a complete overview of the approaches to sequence clustering in the current literature. They argue that it is important to select a distance function that incorporates the sequential structure of the data, and, next, to select an algorithm that is suitable for this distance function. They conclude that the choice for the distance function should be based on domain knowledge of the subject of interest. A popular algorithm that naturally fits a wide range of distance functions is the hierarchical clustering algorithm. For example, Burke et al. [2] adapt the single-linkage hierarchical clustering algorithm to cluster DNA sequences by using a distance function that is suitable for DNA sequences. They conclude that their algorithm works well for their application in the field of DNA research.

It is also possible to cluster sequences using a model-based approach, which makes use of a probabilistic distance function. Cadez et al. [3] present a mixture of first-order Markov models for clustering visitors of a Web page, where the sequence data is assumed to be generated by a Markov model. They also present a visualisation tool for the clusters, which aids interpretation tasks. They state that a mixture of first-order models is not a first-order model, because it can model much more flexible relations. However, while estimating the parameters of the mixture model, the number of clusters is fixed; the number of clusters is chosen in a second step based on the model that minimises an out-of-sample predictive log score. In the current work, the proposed DBHC algorithm learns the initial, transition, and emission probabilities for each cluster, as well as the number of clusters, altogether.

The usage of graphical models for the purpose of sequence clustering was first employed by Rabiner et al. [18]. They propose an algorithm for speech recognition that makes use of clustering using an HMM with continuous output. The problem of clustering speech data is inherently dynamic when audio is logged over time. They are the first to use an HMM in a clustering problem, which they do by fitting an HMM for each cluster, known as a finite mixture of HMMs. An advantage of formulating the problem in this way is that the clustering can now be incorporated in the HMM parameter estimation procedure. The authors propose two methods for obtaining clusters in an HMM. The first is incorporating the clustering in the likelihood function, which can then easily be maximised. However, they conclude that this method appears not to work well in practice, as it only improves the fit for observations that were already represented well by an initial model in the procedure. The second method, which the authors conclude works much better, is a cluster splitting procedure. In this procedure, clusters are iteratively split into smaller clusters, until a threshold is reached in the likelihood objective. In such a split they fit an extra HMM to previously poorly represented observations. A major disadvantage of their work is that the threshold can usually only be based on a simple guess; another disadvantage is that the model performance is very sensitive to the chosen threshold. Following the approach of Li and Biswas

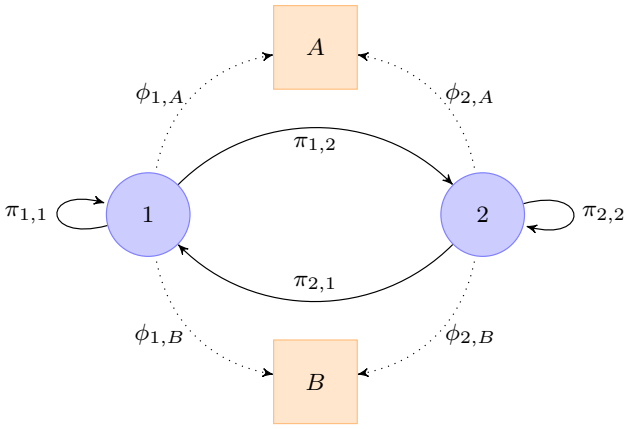


Fig. 1 Visual representation of an HMM with two states $\boldsymbol{\Phi} = \{1, 2\}$ and two output labels $\Sigma = \{A, B\}$

[12] discussed below, we therefore stop the proposed DBHC algorithm when the number of clusters chosen minimises an information criterion for the entire mixture of clusters and sequences. This provides a clearer stopping criterion and prevents the algorithm's performance depending on tuning of such a threshold.

Later on, the maximum likelihood approach to HMM clustering was improved by others, for example in Smyth [20]. The author extends the method in two ways, by incorporating hierarchical clustering for initialisation of the algorithm and by adding a cross-validation approach for determining the number of clusters. He also proposes to estimate the parameters of the HMMs by using the *Baum–Welch* algorithm, which is a special case of the Expectation-Maximisation (EM) algorithm, and a common way to estimate ordinary HMMs [17]. His approach does assume the number of clusters is known, and the solution is to fit the model for different numbers of clusters and determine the optimal value with cross-validation afterwards. The author concluded in a simulation experiment that this search for the number of clusters works well. As a recommendation for further research he suggests incorporating the search for the number of clusters in the estimation procedure using Bayesian statistics. We follow the advice of Smyth [20] and cast the search for the number of clusters as a Bayesian model selection problem.

Bayesian statistics have also been introduced to estimating probabilities in an HMM, for example in Stolcke and Omohundro [21]. The authors use a Bayesian model-merging strategy for finding the number of states in an HMM. Nevertheless, they do not use the HMM for clustering and, therefore, assume a homogeneous population. They do consider an HMM with discrete output, for which they use a Dirichlet prior distribution, a multinomial extension of the Beta distribution. The authors do not extensively compare their prior distribution proposal to other alternatives, but they conclude that the prior type and parameters do not influence the search for the number of HMM states very much. They do admit, however, that including informative priors rather than flat (uninformative) priors will probably have an impact on the course of this search.

Li and Biswas [12] are the first to employ a Bayesian approach for sequence clustering using HMMs. They consider HMMs with continuous output and do not require the number of states and number of clusters known. Furthermore, they do not assume that each HMM has the same model size, i.e., the HMMs do not need to have the same number of states. They propose an algorithm that alternately searches for the optimal number of clusters, cluster assignment of objects, optimal number of states, and the parameters of each HMM. Furthermore, they propose a careful seed selection procedure for initialising cluster models. They do not use a full Bayesian approach, however, as they infer parameters using the frequentist Baum–Welch algorithm and do not use a Bayesian mixture for the clustering part of the algorithm, but rather cast the clustering problem as a Bayesian model selection problem. Besides a formal search, they also propose heuristics for finding the number of clusters and the sizes of the HMM models, which can overcome the computational complexity of an intensive search. In a simulation experiment the authors conclude that their heuristic methodology works well for clustering sequences with continuous output. In the current paper, we incorporate and extend their work to discrete sequences with the proposed DBHC algorithm.

3 Discrete-output HMM

This section describes the methodology of the model underlying the mixture used for sequence clustering in the DBHC algorithm: the HMM with discrete outputs. The HMM is a model for a time series with a finite number of latent states, sometimes also called regimes, which can capture hidden dynamic relations. In the model, emission probabilities can differ between distinct states, states which are assumed to develop according to a first-order Markov process. We describe HMMs where each hidden state defines a categorical distribution over a set of discrete output labels.

3.1 HMM definitions

Let $\mathbf{X} = \left(X_1, \ldots, X_T\right)'$ be a sequence of variables observed over a set of discrete time periods $t \in \{1, \ldots, T\}$. In the standard HMM, one considers an unobserved state variable $Z_t \in \Omega$, with $\Omega = \{1, \ldots, H\}$ being the set of states. These unobserved states are used to model the observed output variable X_t. Output variable X_t depends via an emission probability matrix $\boldsymbol{\Phi}$ directly on the corresponding unobserved state Z_t, which

is assumed to follow a first-order Markov process with some transition matrix $\boldsymbol{\Pi}$. Matrix $\boldsymbol{\Pi}$ is a stochastic matrix that is defined for all pairs of states in $\Omega \times \Omega$. $\boldsymbol{\Pi}$ is called a right stochastic matrix, i.e., the rows of $\boldsymbol{\Pi}$ sum to 1. We consider HMMs with a discrete output variable X_t, defined over the labels in Σ. Emission probabilities in stochastic matrix $\boldsymbol{\Phi}$ describe the relations between X_t and Z_t, defined for every pair of states and labels in $\Omega \times \Sigma$. Matrix $\boldsymbol{\Phi}$ is also a right stochastic matrix, i.e., the rows of $\boldsymbol{\Phi}$ sum to 1. The transition probabilities in $\boldsymbol{\Pi}$ describe the one-period state transitions $\pi_{z,y} = \Pr\left[Z_t = y | Z_{t-1} = z\right]$, where $\pi_{z,y}$ is the element in the zth row and the yth column of $\boldsymbol{\Pi}$. A vector $\boldsymbol{\pi}$ describes the probability distribution for the initial state Z_1. For every time period t, define an emission probability ϕ_{z,A_l} of observing label A_l given the current state Z_t, say z: $\phi_{z,A_l} = \Pr[X_t = A_l | Z_t = z]$. Here ϕ_{z,A_l} is the element in the zth row and the lth column of $\boldsymbol{\Phi}$. Figure 1 shows the dependencies of a two-state HMM with a set of two output labels. The total number of free parameters in an HMM is $(H-1) + H(H-1) + H(L-1)$. One should consider the number of free parameters for an HMM, as the probability vector and the rows of the probability matrices sum to 1 and therefore restrict some of the probabilities. For example, the two-state HMM with two labels in Fig. 1 has $(2-1) + 2(2-1) + 2(2-1) = 5$ free parameters.

Parameters $\boldsymbol{\Pi}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\pi}$ should be estimated from the data. This is a relatively difficult estimation procedure, as the path of hidden states $Z_1, \ldots, Z_T$ is not observed. Before describing the estimation procedure, a few extra probabilities need to be defined. Denote by $w_{z,y,t-1,x_t}, z, y \in \Omega$ the joint probability that the HMM moves from state z at time $t-1$ to state y at time t and emits x_t the weight of the transition. The weight then simply equals the product of a transition probability and an emission probability:

$$\begin{aligned} w_{z,y,t-1,x_t} &= \Pr\left[Z_t = y \,|\, Z_{t-1} = z\right] \cdot \Pr[X_t = x_t \,|\, Z_t = y] \\ &= \pi_{z,y} \cdot \phi_{y,x_t}. \end{aligned} \tag{1}$$

For the zero-th period, define the initial weight as

$$\begin{aligned} w_{0,z,0,x_1} &= \Pr\left[Z_1 = z\right] \cdot \Pr[X_1 = x_1 \,|\, Z_1 = z] \\ &= \pi_z \cdot \phi_{z,x_1}, \end{aligned} \tag{2}$$

where π_z is the zth element of the vector of initial probabilities $\boldsymbol{\pi}$ and where 0 denotes the non-existing preceding state. For the sake of completeness, also define a weight for the Tth period, called w_T, which equals 1 irrespective of the state at time T because there is no time period $T+1$. That is, the HMM ends with probability 1 at time T. Note that one can now write the joint probability for observations $\mathbf{x} = (x_1, \ldots, x_T)'$ of a sequence $\mathbf{X} = (X_1, \ldots, X_T)'$ and a path $Z_1, \ldots, Z_T$ as a product of weights:

$$\begin{aligned} &\Pr[\mathbf{X} = \mathbf{x}, Z_1 = z_1, \ldots, Z_T = z_T] \\ &\quad = \Pr[\mathbf{X} = \mathbf{x} \,|\, Z_1 = z_1, \ldots, Z_T = z_T] \\ &\qquad \cdot \Pr[Z_1 = z_1, \ldots, Z_T = z_T] \\ &\quad = \prod_{t=1}^{T} \Pr[X_t = x_t \,|\, Z_t = z_t] \cdot \Pr[Z_t = z_t \,|\, Z_{t-1} = z_{t-1}] \\ &\quad = \prod_{t=1}^{T} \phi_{z_t,x_t} \cdot \pi_{z_{t-1},z_t} \\ &\quad = \prod_{t=1}^{T} w_{z_{t-1},z_t,t-1,x_t}, \end{aligned} \tag{3}$$

using the definitions of weights from (1) and (2). Furthermore, we define forward and backward probabilities to simplify notation, known from the *forward–backward algorithm* for HMMs [17]. These are recursive probabilities, which can be calculated either by going forward in the sequence of states $Z_1, \ldots, Z_T$ or by going backward in this sequence, after which they are named. Define $forward_{z,t}$ as the forward probability of being in state z at time point t and observing the sequence $x_1, \ldots, x_t$, given weights $w_{z,y,q,x_{q+1}}$, $q \in \{1, \ldots, t-1\}$. The forward probability $forward_{z,t}$ is calculated recursively as

$$\begin{aligned} &forward_{z,t}(x_1, \ldots, x_t) \\ &\quad = \sum_{y \in \Omega} forward_{y,t-1}(x_1, \ldots, x_{t-1}) \\ &\qquad \cdot w_{y,z,t-1,x_t}, \ \forall z \in \Omega, \ t = 2, \ldots, T, \end{aligned} \tag{4}$$

using the definitions of weights from (1) and (2). The recursion is initialised by initial weight $w_{0,z,0,x_1}$, i.e., $forward_{z,1}(x_1, \ldots, x_t) = w_{0,z,0,x_1} = \pi_z \cdot \phi_{z,x_1}, \forall z \in \Omega$. The recursion sums over all possible states for time periods before t. In a similar fashion, define $backward_{z,t}$ as the backward probability of being in state z at time point t and observing the sequence $x_{t+1}, \ldots, x_T$, given weights $w_{z,y,q,x_{q+1}}$, $q \in \{t, \ldots, T\}$. The backward probability $backward_{z,t}$ is then calculated as

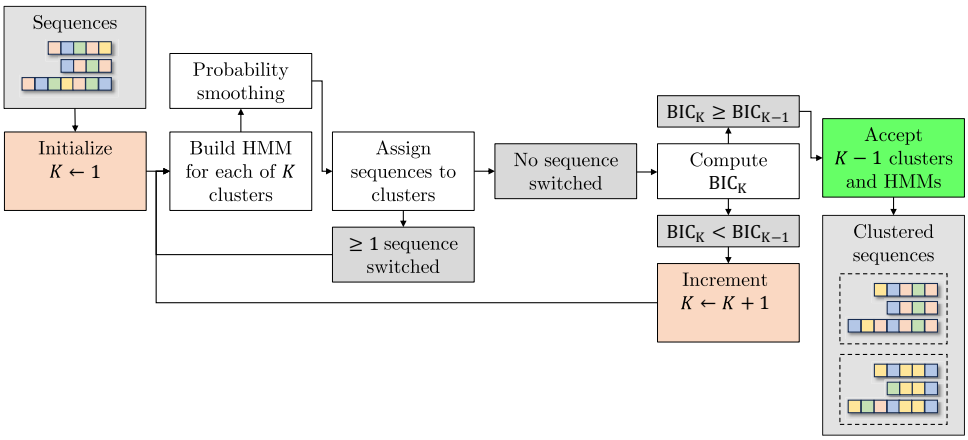
$$\begin{aligned} &backward_{z,t}(x_{t+1}, \ldots, x_T) \\ &\quad = \sum_{y \in \Omega} backward_{y,t+1}(x_{t+2}, \ldots, x_T) \\ &\qquad \cdot w_{z,y,t,x_{t+1}}, \ \forall z \in \Omega, \ t = 1, \ldots, T-1, \end{aligned} \tag{5}$$

again using the definitions of weights in (1) and (2). This recursion is initialised by the weight for the Tth period w_T, i.e., $backward_{z,T} = w_T = 1, \forall z \in \Omega$. One can break down the probability that the HMM passes through state z at time t while emitting $\mathbf{x}$ as

Fig. 2 Baum–Welch algorithm

Initialise HMM parameter estimates $\hat{\varphi}^{(0)} \leftarrow \varphi_0$
Initialise $m \leftarrow 1$
repeat
  E-step: Update responsibility profile in (9) and (10) given estimates in $\hat{\varphi}^{(m-1)}$
  M-step: Maximise $L^{(m)}(\varphi \mid \mathbf{X})$ in (14) by updating $\hat{\varphi}^{(m)}$ in (12) and (13) given current responsibility profile
  Calculate likelihood $L^{(m)}(\varphi \mid \mathbf{X})$ using current parameter estimates $\hat{\varphi}^{(m)}$
  Update $m \leftarrow m + 1$
until $\left|L^{(m)}(\varphi \mid \mathbf{X}) - L^{(m-1)}(\varphi \mid \mathbf{X})\right| < \varepsilon_1$
Accept estimates in $\hat{\varphi}^{(m)}$ as parameters of the HMM

Fig. 3 High-level flowchart of the DBHC algorithm



$$\begin{aligned}&\Pr\left[Z_t = z, \mathbf{X} = \mathbf{x}\right] \\ &\quad = forward_{z,t}(x_1, \ldots, x_t) \cdot backward_{z,t}(x_{t+1}, \ldots, x_T).\end{aligned} \tag{6}$$

Similarly, the probability of making a transition from state z to y from time t to $t+1$ and observing $\mathbf{x}$ can be broken down as

$$\begin{aligned}&\Pr\left[Z_t = z, Z_{t+1} = y, \mathbf{X} = \mathbf{x}\right] \\ &\quad = forward_{z,t}(x_1, \ldots, x_t) \\ &\qquad \cdot w_{z,y,t,x_t} \cdot backward_{y,t+1}(x_{t+2}, \ldots, x_T).\end{aligned} \tag{7}$$

Now, one can also calculate the probability of observing an entire sequence using forward probabilities, which is denoted $forward(\mathbf{x})$. Theexpression can then be evaluated as a product of terms as

$$forward(\mathbf{x}) = \Pr[\mathbf{X} = \mathbf{x}] = \prod_{t=1}^{T} \sum_{z \in \Omega} \sum_{y \in \Omega} w_{y,z,t-1,x_t}. \tag{8}$$

The forward and backward probabilities are used to calculate the so-called responsibility profile of the HMM. The responsibility profile consists of the probabilities of passing through a state z and of the probabilities of making the transition from a state y to a state z, all given the observed sequence $\mathbf{x} = (x_1, \ldots, x_T)'$. The first probability of passing through state z at time t equals

$$\begin{aligned}\Pr[Z_t = z \mid \mathbf{X} = \mathbf{x}] &= \frac{\Pr[Z_t = z, \mathbf{X} = \mathbf{x}]}{\Pr[\mathbf{X} = \mathbf{x}]} \\ &= \frac{forward_{z,t}(x_1, \ldots, x_t)}{forward(\mathbf{x})} \cdot backward_{z,t}(x_{t+1}, \ldots, x_T),\end{aligned} \tag{9}$$

where $forward(\mathbf{x})$ is the forward probability of observing the entire sequence $\mathbf{x}$. The second probability of making the transition from state y to state z between time points t and $t+1$ equals

$$\begin{aligned}\Pr[Z_t = z, Z_{t+1} = y \mid \mathbf{X} = \mathbf{x}] &= \frac{\Pr[Z_t = z, Z_{t+1} = y, \mathbf{X} = \mathbf{x}]}{\Pr[\mathbf{X} = \mathbf{x}]} \\ &= \frac{forward_{z,t}(x_1, \ldots, x_t)}{forward(\mathbf{x})} \cdot w_{z,y,t,x_{t+1}} \\ &\quad \cdot backward_{y,t+1}(x_{t+2}, \ldots, x_T),\end{aligned} \tag{10}$$

which is again defined in terms of forward and backward probabilities, combined with the forward probability of observing the entire sequence $\mathbf{x}$. The probabilities in (9) and (10) are used to estimate the HMM parameters with the *Baum–Welch* algorithm, which is described in the next paragraph.

3.2 Baum–Welch Learning

The Baum–Welch algorithm is a special case of the Expectation-Maximisation (EM) algorithm of Dempster et al. [4], specifically designed for obtaining parameters in an HMM [17]. The algorithm alternates between estimating a responsibility profile given the current HMM parameters in the E-step, and maximising the likelihood by updating the HMM parameters given the current responsibility profile in the M-step. The steps are alternated until the likelihood of the HMM converges. In the E-step, the responsibility profile is calculated using Eqs. (9) and (10). The probabilities in the responsibility profile may be seen as expectations of the path of hidden states that has generated the observed data. For the M-step, we define expressions $\mathrm{T}^t_{z,y}$, $\mathrm{E}^t_z(A_l)$, and I_z based on the probabilities in the responsibility profile:

$$\begin{aligned}
&\mathrm{T}^t_{z,y} = \Pr\left[Z_t = z, Z_{t+1} = y \,|\, \mathbf{X} = \mathbf{x}\right], \\
&\quad t = 1, \ldots, T-1 \\
&\mathrm{E}^t_z(A_l) = \begin{cases} \Pr\left[Z_t = z \,|\, \mathbf{X} = \mathbf{x}\right] & \text{if } x_t = A_l \\ 0 & \text{otherwise} \end{cases}, \\
&\quad t = 1, \ldots, T \\
&\mathrm{I}_z = \Pr\left[Z_1 = z \,|\, \mathbf{X} = \mathbf{x}\right],
\end{aligned} \tag{11}$$

where $\mathrm{T}^t_{z,y}$ is a transition probability at time t, $\mathrm{E}^t_z(A_l)$ an emission probability at time t, and I_z an initial probability. These probabilities in (11) can be summarised into estimates of probabilities for an entire sequence by summing them over the relevant time periods and scaling this sum with respect to all other possibilities for either transition or emission. The parameter estimates of emission and transition probabilities are updated in the current iteration, say m, based on the responsibility profile as follows:

$$\begin{aligned}
\hat{\pi}^{(m)}_{z,y} &= \frac{\sum_{t=1}^{T-1} \mathrm{T}^t_{z,y}}{\sum_{t=1}^{T-1} \sum_{q \in \Omega} \mathrm{T}^t_{z,q}} \\
\hat{\phi}^{(m)}_{z,A_l} &= \frac{\sum_{t=1}^{T} \mathrm{E}^t_z(A_l)}{\sum_{t=1}^{T} \sum_{h \in \Sigma} \mathrm{E}^t_z(A_h)},
\end{aligned} \tag{12}$$

where the responsibility profile in iteration m is calculated using the parameter estimates from iteration $m-1$. Estimates for the initial state probabilities in $\boldsymbol{\pi}$ are simply:

$$\hat{\pi}^{(m)}_z = \frac{\mathrm{I}_z}{\sum_{q \in \Omega} \mathrm{I}_q}. \tag{13}$$

These parameter estimates maximise the likelihood of an HMM in (14), given the current expectations of the path of hidden states—i.e., the responsibility profile [17]. The likelihood is simply the probability that the HMM emitted the sequence $\mathbf{x}$ given the estimated parameters. Let φ denote the collection of the HMM parameters π, $\boldsymbol{\Pi}$, and $\boldsymbol{\Phi}$ for the sake of brevity, $\varphi = (\pi, \boldsymbol{\Pi}, \boldsymbol{\Phi})$. The likelihood of the mth iteration in the algorithm is evaluated as follows:

$$\begin{aligned}
L^{(m)}(\varphi \,|\, \mathbf{X}) &= \Pr\left[\mathbf{X} = \mathbf{x} \,\middle|\, \hat{\varphi}^{(m)}\right] \\
&= \prod_{t=1}^{T} \sum_{z \in \Omega} \sum_{y \in \Omega} w_{y,z,t-1,x_t} \\
&= forward(\mathbf{x}),
\end{aligned} \tag{14}$$

where $forward(\mathbf{x})$ is calculated using the estimated parameters of the mth iteration $\hat{\varphi}^{(m)}$. The algorithm converges when the likelihoods in two consecutive iterations differ less than some small number ε_1. The algorithm should be initialised with some initial guesses for the parameters in φ, $\varphi_0 = (\boldsymbol{\pi}_0, \boldsymbol{\Pi}_0, \boldsymbol{\Phi}_0)$. Usually, these guesses are obtained by random draws from Dirichlet distributions: $\boldsymbol{\pi}_0 \sim \mathrm{Dir}(1, \mathbf{1}_H)$, $\boldsymbol{\Pi}_0 \sim \mathrm{Dir}(H, \mathbf{1}_H)$, and $\boldsymbol{\Phi}_0 \sim \mathrm{Dir}(H, \mathbf{1}_L)$, where $\mathbf{1}_q$ is a $q \times 1$ vector of ones and $\mathrm{Dir}(n, \boldsymbol{\alpha})$ is a Dirichlet distribution with dimensionality n and hyperparameter vector $\boldsymbol{\alpha}$. Figure 2 denotes the pseudocode for the Baum–Welch algorithm.

4 Sequence clustering with the DBHC algorithm

This section describes clustering using HMMs with the DBHC algorithm. The DBHC algorithm is based on the Bayesian HMM Clustering algorithm of Li and Biswas [12], which is intended for sequences with continuous observations in discrete time, while the DBHC algorithm is intended for sequences with discrete observations in discrete time. To enhance the reader's understanding, a high-level flowchart of the DBHC algorithm is provided in Fig. 3.

The clustering of sequences using HMM representations was first mentioned in Rabiner et al. [18]. The idea behind HMM Clustering is to model a different HMM per cluster, that is, modelling a mixture of K HMMs. Let the observed output variable be $X_{i,t}$, gathered in a sequence $\mathbf{X}_i$ for individual i. Let the mixture distribution be described by a discrete variable S, where a value k of S represents a cluster. The probability that sequence i belongs to cluster k is denoted by $p_{i,k}$. The probability of observing sequence $\mathbf{x}_i$ is then modelled as follows:

$$\Pr\left[\mathbf{X}_i = \mathbf{x}_i \mid \varphi\right] = \sum_{k=1}^{K} p_{i,k} \Pr\left[\mathbf{X}_i = \mathbf{x}_i \mid S_i = k; \varphi_k\right], \tag{15}$$

where $\Pr\left[\mathbf{X}_i = \mathbf{x}_i \mid S_i = k; \varphi_k\right]$ is modelled with an HMM. Parameter φ_k denotes the HMM parameters of cluster k, and φ the collection of all φ_k, $\varphi = (\varphi_1, \ldots, \varphi_K)$.

4.1 Learning an HMM from multiple sequences

A cluster can contain multiple individuals, so one needs to be able to train a single HMM on multiple observed sequences. Training an HMM on multiple sequences is very similar to training an HMM on a single sequence when assuming the sequences are generated independently from each other by the HMM [17]. In the E-step of the Baum–Welch algorithm, the responsibility profile of each sequence in a cluster is updated using the current parameters of the HMM for that cluster. This E-step is performed for all sequences in all clusters. The M-step then adjusts the HMM parameters of each cluster using the responsibility profiles of all sequences in each cluster separately. That is, the HMM of a cluster is trained on all sequences in the cluster, i.e., the HMM is trained on multiple sequences. For this purpose, define the responsibility profile of sequence i in cluster k as follows:

$$\begin{aligned} &\Pr\left[Z_{i,t} = z \mid \mathbf{X}_i = \mathbf{x}_i; \hat{\varphi}_k\right] \\ &\Pr\left[Z_{i,t} = z, Z_{i,t+1} = y \mid \mathbf{X}_i = \mathbf{x}_i; \hat{\varphi}_k\right], \end{aligned} \tag{16}$$

which is calculated in the same way as (9) and (10) using the estimated HMM parameters in $\hat{\varphi}_k$. Define also the probabilities in (11) for each individual i in cluster k:

$$\begin{aligned} &\mathrm{T}^t_{i,z,y} = \Pr\left[Z_{i,t} = z, Z_{i,t+1} = y \mid \mathbf{X}_i = \mathbf{x}_i; \hat{\varphi}_k\right], \\ &\quad t = 1, \ldots, T-1 \\ &\mathrm{E}^t_{i,z}(A_l) = \begin{cases} \Pr\left[Z_{i,t} = z \mid \mathbf{X}_i = \mathbf{x}_i; \hat{\varphi}_k\right] & \text{if } x_{i,t} = A_l \\ 0 & \text{otherwise} \end{cases}, \\ &\quad t = 1, \ldots, T \\ &\mathrm{I}_{i,z} = \Pr\left[Z_{i,1} = z \mid \mathbf{X}_i = \mathbf{x}_i; \hat{\varphi}_k\right], \end{aligned} \tag{17}$$

where $\mathrm{T}^t_{i,z,y}$ is a transition probability at time t, $\mathrm{E}^t_{i,z}(A_l)$ an emission probability at time t, and $\mathrm{I}_{i,z}$ an initial probability, all for the sequence of individual i. Instead of maximising the likelihood of a single sequence, the parameter estimates in iteration m are now updated by maximising the joint likelihood of observing the sequences in cluster k given the responsibility profile in iteration m, i.e., the parameters from the previous iteration $m-1$:

$$\begin{aligned} &\Pr\left[\mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_{N_k} = \mathbf{x}_{N_k} \,\middle|\, \hat{\varphi}_k^{(m-1)}\right] \\ &= \prod_{i=1}^{N_k} \Pr\left[\mathbf{X}_i = \mathbf{x}_i \,\middle|\, \hat{\varphi}_k^{(m-1)}\right] \\ &= \prod_{i=1}^{N_k} \mathit{forward}(\mathbf{x}_i), \end{aligned} \tag{18}$$

based on the assumption that the sequences are independently generated to split up the likelihood and where N_k is the number of individuals in cluster k. This means that the likelihoods of the sequences can be maximised independently. Rabiner [17] then shows that the parameter estimates for cluster k in iteration m can be updated in a similar way as in (12) and (13), since the probabilities in the responsibility profile are scaled by the forward probabilities of observing the respective sequence:

$$\begin{aligned} \hat{\pi}^{(m)}_{k,z,y} &= \frac{\sum_{i=1}^{N_k} \sum_{t=1}^{T-1} \mathrm{T}^t_{i,z,y}}{\sum_{i=1}^{N_k} \sum_{t=1}^{T-1} \sum_{q \in \Omega_k} \mathrm{T}^t_{i,z,q}} \\ \hat{\phi}^{(m)}_{k,z,A_l} &= \frac{\sum_{i=1}^{N_k} \sum_{t=1}^{T} \mathrm{E}^t_{i,z}(A_l)}{\sum_{i=1}^{N_k} \sum_{t=1}^{T} \sum_{m \in \Sigma} \mathrm{E}^t_{i,z}(A_m)} \\ \hat{\pi}^{(m)}_{k,z} &= \frac{\sum_{i=1}^{N_k} \mathrm{I}_{i,z}}{\sum_{i=1}^{N_k} \sum_{q \in \Omega_k} \mathrm{I}_{i,q}}. \end{aligned} \tag{19}$$

The Baum–Welch algorithm is used for multiple sequences in the proposed clustering algorithm. Clustering with HMMs adds an extra dimension to the clustering problem, namely that for each cluster not only the number of clusters K for the traditional clustering problem needs to be determined, but also the number of states for each HMM. Following a Bayesian approach, Li and Biswas [12] incorporate this search in the parameter inference.

4.2 DBHC algorithm

Li and Biswas [12] propose a Bayesian estimation method that incorporates both the search for the optimal cluster partition and the search for the number of states per cluster. They call the latter search the model size selection problem. We adapt their strategy to a setting with discrete output. In the Bayesian framework, both the clustering problem and the model size selection problem can be seen as Bayesian model selection problems. That is, selecting the model M with the highest posterior probability: $\Pr[M \mid X]$. To compare two models M_1 and M_2, one would consider the ratio of their posterior probabilities:

Fig. 4 HMM model size search algorithm for a given cluster k

```
// Initialisation
Initialise m ← 1
Initialise H_k ← 1
Obtain HMM parameters for an HMM with size H_k using Baum-Welch algorithm
Set BIC^(1) ← BIC_1
// Model size expansion
repeat
   Update m ← m + 1
   Update H_k ← H_k + 1
   Obtain HMM parameters for an HMM with size H_k using Baum-Welch algorithm
   Set BIC^(m) ← BIC_{H_k}
until BIC^(m) ≥ BIC^(m−1)
// Accept final model
Accept model size of iteration m − 1
```

$$\frac{\Pr\left[M_2 \mid X\right]}{\Pr\left[M_1 \mid X\right]} = \frac{\frac{\Pr\left[M_2\right]\Pr\left[X \mid M_2\right]}{\Pr\left[X\right]}}{\frac{\Pr\left[M_1\right]\Pr\left[X \mid M_1\right]}{\Pr\left[X\right]}} = \frac{\Pr\left[M_2\right]\Pr\left[X \mid M_2\right]}{\Pr\left[M_1\right]\Pr\left[X \mid M_1\right]}, \tag{20}$$

where Bayes' rule is used to develop the posteriors. If there is no favour for any model in advance, that is, if there is no favour for any number of clusters or any model size for the HMMs, then $\Pr\left[M_1\right] = \Pr\left[M_2\right]$. The ratio of model posteriors in (20) would then simplify to the ratio of likelihoods:

$$\frac{\Pr\left[M_2 \mid X\right]}{\Pr\left[M_1 \mid X\right]} = \frac{\Pr\left[X \mid M_2\right]}{\Pr\left[X \mid M_1\right]}. \tag{21}$$

In conclusion, when choosing between different models, one can simply select the model with the largest likelihood. In case of unobserved variables, this should be the complete data likelihood. This strategy is used both for the search for the number of clusters and for the HMM model size. The complete data likelihoods can be obtained using Markov Chain Monte Carlo methods in case of latent variables, for example with Gibbs sampling [9].

When using Markov Chain Monte Carlo methods for Bayesian inference, the two search dimensions can turn out to be very costly in terms of computational complexity, namely exponential complexity [12]. The authors therefore propose to use approximations of the log complete data likelihood using the Bayesian Information Criterion (BIC). These approximations are used for both search dimensions. The Bayesian HMM Clustering algorithm uses hard assignment of clusters, that is, $p_{i,k} = 1$ for one value of k, and $p_{i,l} = 0$ for all $l \neq k$. In general the BIC of a model is defined as $\text{BIC} = -2 \log L + d \log N$, where L is the likelihood of the model without unobserved variables, d is the number of parameters, and N is the number of observations.

The number of clusters K is chosen such that the BIC for the entire mixture is minimised. Following the approach in Li and Biswas [12], the number of clusters K is added as a penalty to the BIC, similar to the penalty of the number of parameters. The BIC for a model M^K with K clusters can be calculated as follows:

$$\text{BIC}_K = -2 \sum_{i=1}^{N} \log \left[\sum_{k=1}^{K} p_{i,k} \Pr\left[\mathbf{X}_i \,\middle|\, S_i = k; \hat{\varphi}_k \right] \right] + \left(K + \sum_{k=1}^{K} d_k \right) \log N, \tag{22}$$

where d_k is equal to the number of free parameters in $\hat{\varphi}_k$ larger than some threshold ε_2, again following the original algorithm. In this way, the BIC only penalises the likelihood for probabilities that are set to a non-zero value larger than ε_2. Note that $p_{i,k} = 1$ if $S_i = k$ and 0 otherwise. Recall that the number of free parameters in an HMM for cluster k is $(H_k - 1) + H_k(H_k - 1) + H_k(L - 1)$. From the previous number, d_k is obtained by subtracting the number of parameters that are smaller than ε_2 from the total number of free parameters. Now d_k is a penalty for the number of non-zero parameters. Minimising BIC_K is approximately equivalent to maximising the log complete data likelihood [12]. Note that this is the complete data likelihood as it assumes the cluster memberships known. The number of clusters K is chosen such that its corresponding model minimises BIC_K. Starting the search with $K = 1$, K is iteratively increased with steps of 1, until BIC_K does not decrease anymore, as suggested by Li and Biswas [12].

For each cluster, the number of HMM states is determined in a likewise fashion. In a similar way as shown in (20) and (21), we show that the optimal size H_k for the model of cluster k can be determined by maximising the likelihood over a set of possible values for H_k. Again, the likelihood is approximated with the BIC:

Fig. 5 DBHC algorithm

// Initialisation
Initialise $m \leftarrow 1$ // iteration
Initialise $K \leftarrow 1$ // number of clusters
Set seed size r // seed size
// First iteration
Select random first observation from $1, \ldots, N$
Build temporary HMM with 2 states based on first observation
Smooth emission probabilities of temporary HMM
Evaluate sequence-to-HMM likelihood based on temporary HMM for all sequences not yet selected
Add $r - 1$ observations to seed with highest sequence-to-HMM likelihood
Apply HMM model size search in Figure 4 to first cluster
Smooth emission probabilities of HMM
Add all sequences to first cluster
Set $\text{BIC}^{(1)} \leftarrow \text{BIC}_1$ first clustering
// Partition expansion
repeat
  Update $m \leftarrow m + 1$
  Update $K \leftarrow K + 1$
  // Seed selection
  for $i \leftarrow 1$, K **do**
    if i = 1 **then**
      Select random first observation from $1, \ldots, N$
    else
      Evaluate sequence-to-HMM likelihood for every seed model in iteration i for all sequences not yet selected
      Consider for each sequence the HMM for which it has the highest sequence-to-HMM likelihood
      Select first observation as the one with lowest such sequence-to-HMM likelihood: the sequence worst represented by the current seed models
    end if
    Build temporary HMM with 2 states based on first observation
    Smooth emission probabilities of temporary HMM
    Evaluate sequence-to-HMM likelihood based on temporary HMM for sequences not yet selected
    Add $r - 1$ observations to seed with highest sequence-to-HMM likelihood
    Apply HMM model size search in Figure 4 to the seed
    Smooth emission probabilities of HMM
  end for
  Set seed models as the cluster models
  // Object redistribution
  Assign each sequence to the cluster model for which it has highest sequence-to-HMM likelihood
  repeat
    Update HMM parameters with Baum-Welch algorithm in Figure 2 for all clusters
    Smooth emission probabilities of each HMM
    Assign each sequence to the cluster model for which it has highest sequence-to-HMM likelihood
  until No sequence switches between clusters
  Accept current sequence distribution
  Set $\text{BIC}^{(m)} \leftarrow \text{BIC}_K$ current clustering
until $\text{BIC}^{(m)} \geq \text{BIC}^{(m-1)}$
// Accept final model
Accept clustering in iteration $m - 1$ as the model

$$\text{BIC}_{H_k} = -2 \sum_{i=1}^{N_k} \log \Pr \left[\mathbf{X}_i \,\middle|\, H_k, \hat{\varphi}_k \right] + d_k \log N_k, \tag{23}$$

where d_k is equal to the number of free parameters in $\hat{\varphi}_k$ larger than some threshold ε_2. Each HMM is initialised with $H_k = 1$, and H_k is iteratively increased with steps of 1 until BIC_{H_k} does not decrease anymore. This set-up is similar to the search for the optimal number of clusters. Figure 4 denotes the pseudocode for the HMM model size search algorithm for cluster k.

For a given model size, model parameters for each cluster k are obtained with the Baum–Welch algorithm for multiple sequences. For assigning sequences to clusters, the sequence-to-HMM likelihood measure is used [17]. An observed sequence $\mathbf{x}_i$ is assigned to the cluster k that has the highest

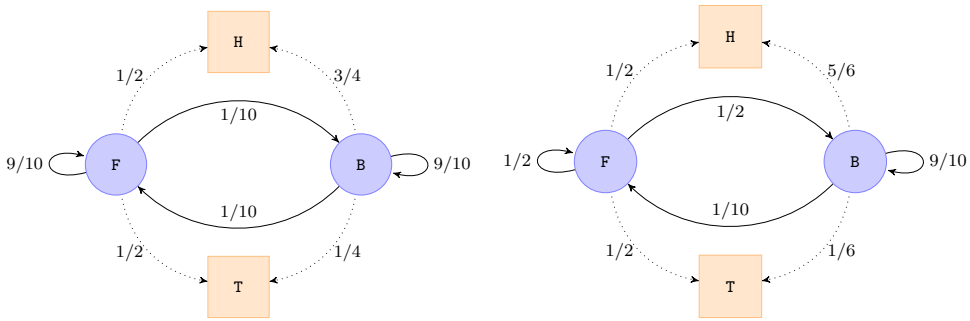


Fig. 6 Visual representations of two dishonest casino HMMs, each with two states $\boldsymbol{\Phi} = \{\mathtt{F}, \mathtt{B}\}$ and two output labels $\Sigma = \{\mathtt{H}, \mathtt{T}\}$

likelihood: $\Pr\left[\mathbf{x}_i \mid \hat{\varphi}_k\right]$. The sequence-to-HMM likelihood is simply the forward probability of observing that sequence in the HMM of cluster k, i.e., $\Pr\left[\mathbf{x}_i \mid \hat{\varphi}_k\right] = forward(\mathbf{x}_i)$. If after all assignments at least one of the sequences switched clusters, the HMM of each cluster is re-estimated without changing the model size, and the observations are distributed over the new clusters. The process is repeated until no sequence changes clusters after redistribution. Finally, the size search algorithm is invoked for all clusters to make the models best reflect the data in the clusters.

At the start of every iteration in the search for the optimal clustering, all models in the partition must be initialised with a set of observations, a seed. The seed is used to determine the number of HMM states for a cluster before other observations are added. When the size search algorithm is invoked for heterogeneous data—i.e., it contains observations from different true clusters—additional states might be added to reflect the heterogeneity, even though this heterogeneity is ideally incorporated in the cluster memberships. The seed selection procedure aims at preventing this. In each iteration, the model for each cluster is rebuilt to reflect the data best according to the number of clusters at that point in the algorithm. A seed consists of r observations, with r usually an arbitrarily chosen small number. For the first seed in every iteration, a sequence is randomly selected from the set of all sequences. Then, an HMM is estimated for this observation—with 2 states, as this is just a temporary HMM—and the remaining $r-1$ observations that are best represented by this HMM in terms of sequence-to-HMM likelihood, are added to the seed. For all the other $K-1$ cluster seeds, the first sequence is selected as the one that is worst represented by the current set of seed models. The remaining $r-1$ sequences are found in the same way as for the first seed. In total, r observations have now been included in each of the K seeds. Li and Biswas [12] use $r = 3$ for initialising the models throughout their paper. After the observations for the seed have been selected, the seed model is determined using the size search algorithm in Fig. 4.

In the seed selection procedure the likelihood of an HMM is many times developed for sequences the HMM has not been trained on. It is therefore possible that these sequences contain labels the training set for the HMM did not contain—these emissions were assigned probability zero in the Baum–Welch algorithm. The new sequences potentially also require emission-state combinations with zero probability in the HMM. All these cases will result in a likelihood value of 0, corresponding to a log likelihood value of $-\infty$. The step that selects the worst represented sequence will then be inconclusive among all sequences with log likelihood $-\infty$, and proceed with a random draw from these sequences. This inconclusiveness is overcome by adding a probability smoothing step to the emission probability matrix after parameter estimation.

Probability smoothing is replacing zero probabilities with very small values to prevent the likelihood of unseen data to be zero, while preserving the condition that all probabilities sum to 1. After parameter estimation with the Baum–Welch algorithm, the probability smoothing technique of absolute discounting is used, which has been shown to work well for HMMs in word recognition [22]. Absolute discounting is subtracting a small amount of probability from all non-zero probabilities, and dividing this portion equally over the zero probabilities. The smoothing is applied row-wise for the emission probability matrix after parameter estimation.

For a certain state z in the HMM of a cluster k, let the number of non-zero emission probabilities be v. Recall that L is the total number of labels in the alphabet, thus $L-v$ is the number of zero emission probabilities. A small amount q is subtracted from each non-zero emission probability and then divided equally over the $L-v$ non-zero probabilities. The smoothed probability of ϕ_{k,z,A_l}, say $\widetilde{\phi}_{k,z,A_l}$, then equals:

$$\widetilde{\phi}_{k,z,A_l} = \begin{cases} \phi_{k,z,A_l} - q & \text{if } \phi_{k,z,A_l} > 0 \\ \frac{vq}{L-v} & \text{otherwise} \end{cases}, \quad l = 1, \ldots, L. \tag{24}$$

There is no standard way for determining the smoothing parameter q, but for the proposed algorithm it is important to pick it such that $\frac{vq}{L-v}$ is smaller than ε_2. In this way the smoothed zero probabilities do not count towards the total of number of non-zero parameters d_k.

Table 1 Resulting parameter estimates from the DBHC algorithm for two clusters in the working example for HMM Clustering, where parameter estimates are rounded up to two decimal points

$\hat{\boldsymbol{\pi}}_1$					
S1	0				
S2	0				
S3	1				
$\hat{\Pi}_1$	S1	S2	S3		
S1	0	1	0		
S2	0	0	1		
S3	1	0	0		
$\hat{\boldsymbol{\Phi}}_1$	H	T			
S1	0.67	0.33			
S2	1	0			
S3	1	0			
$\hat{\boldsymbol{\pi}}_2$					
S1	0.85				
S2	0				
S3	0				
S4	0.15				
S5	0				
$\hat{\Pi}_2$	S1	S2	S3	S4	S5
S1	0	0.20	0	0.80	0
S2	0.90	0	0.10	0	0
S3	0	0.81	0.19	0	0
S4	0	0	0	0	1
S5	0.08	0	0.75	0.10	0.07
$\hat{\boldsymbol{\Phi}}_2$	H	T			
S1	1	0			
S2	1	0			
S3	0	1			
S4	1	0			
S5	0.12	0.88			

Now we present the entire DBHC algorithm with the order in which we execute the search and estimation steps, including the probability smoothing step for discrete sequence data. Figure 5 denotes the pseudocode for the algorithm. As described in Sect. 3.2, the Baum–Welch algorithm requires starting values for the HMM parameters. The HMM size search algorithm builds the HMMs with random starting values for the parameters as there is no information on the sequences in a cluster yet. For building temporary HMMs—used for finding either the most similar or the most dissimilar sequences in the seed selection procedure—random starting values are used for the same reason. However, during the updating of HMM parameters in the object redistribution phase at the end of each iteration, the HMM for a cluster is initialised with the parameters that were estimated for the cluster before the most recent redistribution. This is done to facilitate convergence during the alternation between object redistribution and parameter updating. Whenever we do initialise an HMM with random starting values for the parameters, we use 10 of these random starts and select the model that achieves the highest likelihood among these, following Helske and Helske [10]. We use only 10 random starts as this estimation procedure is often repeated in the algorithm and higher numbers of random starts would put an extra burden of computational complexity on the algorithm.

Finally, the DBHC algorithm is implemented in the **DBHC** package of the R statistical software [1]. Default hyperparameter values in the software package correspond to the values suggested in this paragraph. For example, the default number of hidden states in an initial HMM is set equal to 2 and the default number of observations in a seed is set equal to 3. These and other hyperparameters can be controlled by the user of the software package, see Budel and Frasincar [1]. The analysis in Sect. 6.2 utilises this R implementation of the algorithm.

5 Working example

We briefly illustrate the estimation procedure for HMM Clustering with the DBHC algorithm. As the algorithm requires multiple runs of the Baum–Welch algorithm for multiple sequences, it would be infeasible to work through every step of the DBHC algorithm. Therefore, we illustrate the DBHC algorithm at a high level by showing the course of the algorithm for a fictional scenario based on the work of [6], who provide an example of a discrete HMM using the narrative of an occasionally dishonest casino. The reader may further assume that the steps described hereafter are representative of a real run of the DBHC algorithm; the specific parameter settings are unimportant.

Suppose there are two clusters, for each of which the true model is given by either the left or the right HMM seen in Fig. 6; denote these HMM.1 and HMM.2, respectively. In both HMMs, the initial distribution is assumed to be identical for the two hidden states: both states have an initial probability of 1/2. In the narrative of the dishonest casino, the two HMMs can be seen as two different casinos, where the second casino uses an even more biased coin and, compared to the first casino, is even more inclined to use that biased coin instead of a fair coin. Following the narrative, customers do not know in which casino they are and therefore they do not know the casino's tendency to use the biased coin—in fact, in the DBHC algorithm, even the number of casinos is unknown, but this is harder to map to the narrative. The seed selection procedure of the DBHC algorithm requires

at least 3 sequences per cluster. Therefore, for each cluster 5 sequences of length 10 are randomly generated according to the probabilities in the corresponding HMM, in order to have a few more sequences than the minimum for each cluster—in this way, it is also possible to perform the iteration in the algorithm where $K = 3$. By manually generating 5 sequences for each cluster, there is no randomness in the cluster assignments, which is not required for illustrating the estimation procedure.

The sequences of the first cluster are drawn using HMM.1:

1 : H − H − T − T − H − H − H − H − T − H
2 : H − T − H − H − T − H − H − H − H − H
3 : H − T − T − H − H − H − T − T − T − H
4 : H − H − H − H − T − T − H − H − H − T
5 : H − H − T − H − T − T − H − H − H − H,

and the sequences of the second cluster using HMM.2:

6 : H − H − T − T − T − H − T − H − H − H
7 : H − H − H − H − T − H − H − H − H − H
8 : H − H − H − H − T − T − T − H − H − H
9 : H − H − H − H − H − H − H − H − H − H
10 : H − H − T − H − H − T − T − H − H − H.

It is obvious that in sequences 6 to 10, H is drawn more frequently than in sequences 1 to 5, as this probability is higher for HMM.2.

In the first iteration ($K = 1, m = 1$), observation 7 is randomly selected for the first seed. We estimate a temporary 2-state HMM for this observation and evaluate log likelihoods for all other observations. Observations 9 ($\log L = -1.20$) and 2 ($\log L = -4.83$) appear to be the most similar and are added to the first seed, which seems reasonable as all three sequences contain a lot of H-observations. The size search algorithm finds BICs of 21.34, 14.75, and 18.93 for the number of HMM states equal to 2, 3, and 4, respectively. Therefore, 3 is accepted as the number of hidden states for this cluster. Now all other sequences are added to this cluster and the HMM is smoothed, resulting in a BIC of 339.04.

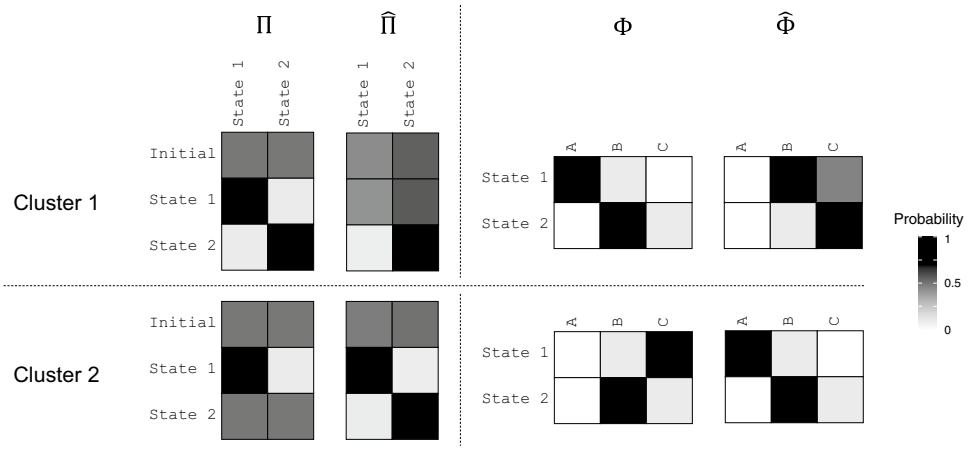
In the second iteration ($K = 2, m = 2$), observation 2 is selected for the first seed, after which observations 9 ($\log L = -2.95$) and 7 ($\log L = -3.61$) are added as being most similar. Note that this is the exact same selection that occurred in iteration 1. Therefore, the size search algorithm finds the exact same models with the exact same BICs as in iteration 1 and again returns the number of HMM states equal to 3. For the second seed, observation 3 is selected as being worst represented by the current set of seed models ($\log L = -29.95$). The reason that observation 3 is represented badly by the current model is probably because observation 3 contains the most T-observations in the sample: 5. Next, a temporary 2-state HMM is estimated based on observation 3, and observations 6 ($\log L = -6.67$) and 4 ($\log L = -6.86$) are added as they are the most similar among the remaining observations. All sequences are assigned to clusters, and the re-estimation of parameters and re-assignment of clusters is iterated until no observation changes clusters anymore. In the resulting assignment observations 2, 7, and 9 are assigned to the first cluster with 3 hidden states; the remaining observations are assigned to cluster 2 with 5 hidden states. The resulting model achieves a BIC of 107.69, therefore the algorithm run continues.

In the third iteration, the seed selection procedure is performed in an identical way as in the first two iterations. This results in cluster models with 7, 3, and 6 hidden states for cluster 1, 2, and 3, respectively. After iterating between (re-)assignment and (re-)estimation, this clustering partition achieves a BIC of 113.37. As this is an increase in BIC compared to the previous iteration, the number of clusters is determined to be 2; the model in iteration 2 is accepted.

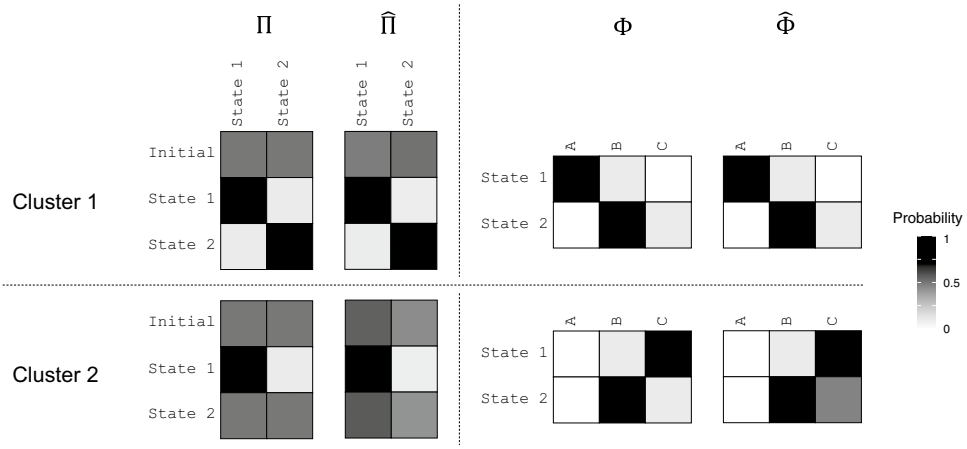
Table 1 shows the parameter estimates of the resulting model from the DBHC algorithm. Note that the nature nor the ordering of the hidden states are known, therefore the states are labelled arbitrarily as S1, S2, S3 and S1, S2, S3, S4, S5 for clusters 1 and 2, respectively. Besides the fact that the algorithm found two clusters, the estimates look very different from the true parameters. As there are a lot of 0's and 1's in the estimates, it again looks like the parameters are a little overfitted. This might be due to the small sequence length or the low number of observations. In case of the former, this problem is also present in our data study. Please note, however, that this working example is merely meant for illustrating the estimation procedure with the DBHC algorithm and is not representative of a data set for an actual clustering problem.

6 Simulation and case study

This section details the evaluation of the proposed DBHC algorithm. First, it provides a simulation study with controlled parameters to which the parameters estimated by the DBHC algorithm can be compared. Second, it provides a case study on life sequence clustering, with data taken from the Swiss Household Database. The aim of the first study is to showcase the capabilities of the DBHC algorithm, as we can directly compare the values of its parameter estimates to their ground truth values. The second



(a) Raw results of the simulation study.



(b) Relabelled results of the simulation study.

Fig. 7 Heatmaps of the simulation study results. From **a**, **b**, the estimated parameters $\widehat{\Pi}$ and $\widehat{\Phi}$ are relabelled to correspond to the ground truth initial and transition probabilities Π and emission probabilities Φ, respectively. These ground truths are the same in both figures

study further illustrates how one can use the DBHC algorithm to perform sequence clustering of real-life data.

6.1 Simple simulation study

We showcase the capabilities of the DBHC algorithm in a controlled environment where the true values of the parameters to be estimated can be fixed. This is done via a simple simulation study, so that the estimated parameters from the DBHC algorithm can be readily compared to the ground truth values. In the simulation, two clusters are generated with 500 sequences each, where each sequence comprises 20 observations. The observations are labelled with the letters {A, B, C}. The simulated clusters each have a state with a high probability of generating label *B*, while only cluster 1 has a high probability of generating label *A*, and only cluster 2 has a high probability of generating label *C*. Furthermore, cluster 1 is likely to stay in whichever state it is already in, while cluster 2 is only likely to stay in state 1.

Again, the algorithm is run with the default hyperparameter values described in Sect. 4.2. The results of this simulation study are shown via heatmaps in Fig. 7. The initial, transition, and emission probability matrices used to simulate the sequences are denoted by Π (initial and transition) and Φ (emission), respectively; the parameters estimated by the DBHC algorithm are denoted by $\widehat{\Pi}$ and $\widehat{\Phi}$. Before estimation, the sequences were randomly shuffled such that the cluster they belong to could not be deduced anymore from the order in which they were generated.

Note that from Fig. 7a and b, the estimated parameters have been relabelled from cluster 1 to cluster 2, as well as (within each cluster) from state 1 to state 2. This is done for visual purposes only and illustrates the growing complexity of validating the estimation results of the DBHC algorithm, or any HMM Clustering algorithm in general, when increasing the number of clusters and/or

Table 2 Sequence labels appearing in swiss household data

Original	Label	Family status
0	P	Living with parents
1	L	Left home
2	M	Married
3	LM	Left home, married
4	C	Having children
5	LC	Left home, having children
6	LMC	Left home, married, having children
7	D	Divorced

Table 3 Clustering results swiss household data

Cluster	# Individuals	# Hidden states
1	157	2
2	66	3
3	211	3
4	425	6
5	298	4
6	843	5

hidden states. Namely, as far as the DBHC algorithm is concerned, there is no distinction between the numbering of the clusters; it does not matter whether cluster 1 is cluster 2, and vice versa. Additionally, within each cluster, the numbering of states also does not matter as long as the transition probabilities remain conformative; if states 1 and 2 are relabelled to states 2 and 1, it is merely required to exchange transition probability $P_{1,2}$ with $P_{2,1}$. This applies to any number of clusters and states.

As can be seen from the relabelled estimated parameters in Fig. 7b, the results very closely match the ground truths of the initial, transition, and emission probabilities. Only for the emission probability of label *C* from state 2 in the second cluster is there a visible discrepancy between the ground truth parameter and the estimated parameter. This may be explained by the fact that the generated sequences were limited in both length and number, making accurate estimation more challenging. All other probabilities were estimated with very high accuracy. We conclude that the DBHC algorithm performs very well on this simulation study with 2 sequence clusters.

6.2 Case study: clustering life sequences

To further illustrate how one can perform sequence clustering using the DBHC algorithm, we perform an analysis on the `biofam` data set from R package **TraMineR**.[1] These data consist of family life sequences built from a biographical survey carried out by the Swiss Household Panel (SHP) in 2002 [8]. The data contains 16-year-long sequences with yearly observations of family status for 2000 individuals who were at least 30 years old at the time of the survey. Table 2 shows the labels appearing in the data set, that is, the statuses recorded in the survey for the 2000 individuals, along with the original labels attached to these statuses in `biofam`. The non-intuitive labels 0 to 7—those that appear in the original data set—are replaced by the more intuitive ones in Table 2.

Table 3 shows descriptives of the results found by the DBHC algorithm for the Swiss household data. The algorithm is run with the default hyperparameter values described in Sect. 4.2, e.g., the seed size is set to 3. The algorithm finds 6 clusters for the `biofam` data set, with the number of hidden states ranging from 2 to 6. The cluster sizes range from 66 individuals in cluster 2 to roughly 800 individuals in cluster 6. Here, an 'individual' of course corresponds to a sequence belonging to an individual.

To show how the clusters can be interpreted using the hidden states of the cluster model, we study the heatmaps of the HMM of one of the clusters, say cluster 5, which contains roughly 300 individuals and for which the algorithm finds 4 hidden states. Figure 8 shows heatmaps of the initial, transition, and emission probability matrices of this cluster. People in this cluster initially start in state 1 and—looking at the emission probability matrix—they are living with their parents in this state. After being in this state 1, there is a small probability to move to state 2, where they leave home—although the majority stays in state 1. Observe that after state 2, part of the individuals moves to state 4, where they remain in the status 'left home'. Individuals in state 4 tend to stay in state 4. This leaves us with state 3, for which it is not visible by which state it is preceded in the heatmap, i.e., apparently the probability of moving to state 3 is marginal. Note that the corresponding emissions of state 3 therefore also rarely occur for this cluster. We conclude that cluster 5 consists of individuals who start out living with their parents, after which there is a probability of transitioning into the state of leaving home, where they will stay until the end once this has occurred, without getting married or having children. A random sample of 5 sequences drawn from the ones contained in this cluster confirms this intuition:

[1] The data set is available from **TraMineR** (see https://search.r-project.org/CRAN/refmans/TraMineR/html/biofam.html) and sourced from the SWISSUbase (see https://www.swissubase.ch/en/catalogue/studies/6097/19347/datasets).

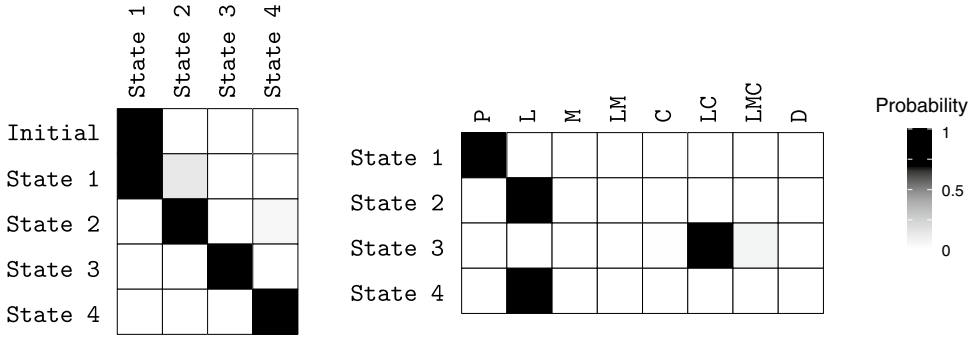


Fig. 8 Heatmaps of initial and transition probability matrix (left) and emission probability matrix (right) of cluster 5

1 : P – P – P – P – P – P – L – L – L – L – L – L – L – L – L
2 : P – P – P – P – P – P – P – P – P – P – L – L – L – L – L – L
3 : P – P – L – L – L – L – L – L – L – L – L – L – L – L – L – L
4 : P – P – P – P – P – P – L – L – L – L – L – L – L – L – L – L
5 : P – P – L – L – L – L – L – L – L – L – L – L – L – L – L – L.

Similar analyses can be carried out for the remaining cluster models, which one would use to obtain a complete view of the partition found by the DBHC algorithm. Doing so provides descriptions of the other clusters, too. Individuals in cluster 1 live with their parents and stay there for the entire observation period. In cluster 2, individuals start out already having left home and marry eventually, combined with either directly having children or having children at some later point in time. Individuals in cluster 3 start out living with their parents and go into marriage without leaving their paternal home or having children. In cluster 4, individuals start out living with their parents, after which they leave home and later get married with a potential for having children, a divorce, or a combination of those two. Cluster 6 is the biggest cluster with 843 individuals, for which the algorithm finds 5 hidden states. Individuals in cluster 6 start out living with their parents and then follow all kinds of different paths before they eventually end up having left home, being married, and having children. With this cluster, we may conclude that it is the most common path to start out living with parents and eventually end up moving out, being married and having children. Note that the path identified for cluster 4 is the only one that involves the potential for a divorce. In total, the algorithm identifies roughly 6 different life paths, each path corresponding to one of the clusters. In this way, we have illustrated how hidden states can help interpreting cluster memberships in sequence data sets.

7 Conclusion

This paper presented the DBHC algorithm for sequence clustering using HMMs and showed how the hidden states in a mixture of HMMs can aid the interpretation task of a cluster analysis for sequence data. The DBHC algorithm extends the existing Bayesian HMM Clustering algorithm to be applicable to discrete sequence data by adding a probability smoothing step with absolute discounting to its seed selection procedure. The DBHC algorithm is completely self-contained as it incorporates both the search for the number of clusters and the search for the number of hidden states in each cluster model, next to the parameter estimation procedure with the Baum–Welch algorithm for each cluster model. The algorithm can be used to obtain a mixture of HMMs for discrete sequence data and is implemented in the **DBHC** package of the R statistical software. We have provided a working example, simulation study, and case study to both further explain the algorithm and highlight its capabilities. We may conclude that the algorithm works well as it provides well-interpretable clusters for the considered application. In future work, we would like to explore whether a full Bayesian approach—as opposed to the current semi-Bayesian approach—improves the fit of the mixture model. We expect that this will most likely put an extra computational burden on the algorithm complexity and research into the feasibility is required.

Data availability For the simulation study, the data is given in the paper. For the case study, we have used the biofam data set and the link to it is given in a footnote at the end of the paper.

References

1. Budel G, Frasincar F (2022) DBHC: sequence clustering with Discrete-Output HMMs. https://CRAN.R-project.org/web/packages/DBHC, R package version 0.0.3

2. Burke J, Davison D, Hide W (1999) d2_cluster: a validated method for clustering EST and full-length cDNA sequences. Genome Res 9(11):1135–1142
3. Cadez I, Heckerman D, Meek C, Smyth P, White S (2003) Model-based clustering and visualization of navigation patterns on a web site. Data Min Knowl Discov 7(4):399–424
4. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B (Methodol) 39(1):1–22
5. Dong G, Pei J (2007) Sequence data mining. Springer Science & Business Media, Berlin
6. Durbin R, Eddy SR, Krogh A, Mitchison G (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge
7. Fan W, Hou W (2022) Unsupervised modeling and feature selection of sequential spherical data through nonparametric hidden Markov models. Int J Mach Learn Cybern 13(10):3019–3029
8. Gabadinho A, Ritschard G, Mueller NS, Studer M (2011) Analyzing and visualizing state sequences in R with TraMineR. J Stat Softw 40(4):1–37
9. Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans Pattern Anal Mach Intell PAMI–6(6):721–741
10. Helske J, Helske S (2019) Mixture hidden Markov models for sequence data: the seqHMM Package in R. J Stat Softw 88(3):1–32
11. Lagona F, Jdanov D, Shkolnikova M (2014) Latent time-varying factors in longitudinal analysis: a linear mixed hidden Markov model for heart rates. Stat Med 33(23):4116–4134
12. Li C, Biswas G (2000) A Bayesian approach to temporal data clustering using hidden Markov models. In: Proceedings of the 17th international conference on machine learning (ICML 2000). Morgan Kaufmann Publishers Inc., pp 543–550
13. Liao TW (2005) Clustering of time series data—a survey. Pattern Recognit 38(11):1857–1874
14. MacKay RJ (2002) Estimating the order of a Hidden Markov model. Can J Stat 30(4):573–589
15. Mirkin B (1996) Mathematical classification and clustering. Kluwer Academic Publishers, Norwell
16. R Core Team (2017) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/
17. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286
18. Rabiner LR, Lee CH, Juang BH, Wilpon JG (1989) HMM clustering for connected word recognition. In: Proceedings of the international conference on acoustics, speech, and signal processing (ICASSP 1989). IEEE, pp 405–408
19. Rabiner LR, Juang B (1986) An introduction to hidden Markov models. IEEE ASSP Mag 3(1):4–16
20. Smyth P (1996) Clustering sequences with hidden Markov models. In: Proceedings of the 10th international conference on neural information processing systems (NIPS 1996). MIT Press, pp 648–654
21. Stolcke A, Omohundro SM (1994) Best-first model merging for hidden Markov model induction. ICSI Technical Report TR-94-003
22. Taghva K, Coombs JS, Pereda R, Nartker TA (2005) Address extraction using hidden Markov models. In: Proceedings of the 12th document recognition and retrieval conference (DRR 2005). SPIE, pp 119–126
23. Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2(2):165–193