Bayesian Time Series Classification

December 2019

Bennaceur Mehdi
ENSAE & École Polytechnique
mehdi.bennaceur@ensae.fr

Delanoue Pierre
ENSAE & ENS Paris-Saclay
pierre.delanoue@ensae.fr

Abstract

This paper explores two Bayesian approaches to the classification of time series: a Bayesian k nearest neighbours (KNN) and a Bayesian hidden markov model (HMM). After implementing these algorithms in python we tested them on 6 datasets of univariate time series.

The Bayesian approach to KNN proved to be significantly faster and more accurate than a KNN whose optimal number of neighbours would have been found by cross validation. On the ECG200 data set, we go from 72% to 84% accuracy with the Bayesian KNN.

Less effective on real data sets, Bayesian HMM models present very interesting perspectives when combined with a priori knowledge on the studied data set.

All our codes can be found on the GitHub page https://github.com/DatenBiene/Bayesian_Time_Series_Classification

1. Introduction

Time series classification can be used for numerous applications in healthcare, finance, speech recognition or any field requiring decision-making based on the evolution of ordered data. In many situations and unlike the non-ordered case, a time-series forecasting method cannot be translated to a classification method. These two aspects make time series classifications the object of intense research on both machine learning [2] and deep learning approaches [5].

Here we study two Bayesian approaches to time series classifications with a focus on univariate time series. First we will present a Bayesian k nearest neighborhood method applied to time series then a Bayesian Hidden Markov Model.

2. Notations and Definitions

Definition 1. [Time Series] X is an univariate time series of length $T \in \mathbb{N}$ if $X = [x_1, x_2, ..., x_T]$ where $\forall i, x_i \in \mathbb{R}$ are ordered real values.

As we might consider time series of different length, we note $|X|$ the length of the time series X.

Definition 2. [Dataset] A dataset $D_N = \{(X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)\}$ is a collection of pairs where X_i a time series and Y_i is the corresponding label. In our classification situation, Y_i can only take a finite number of values, corresponding to the number of classes of our problem.

3. Bayesian K Nearest Neighborhood

The K Nearest Neighborhood is a well-known algorithm which, combined with an adapted distance, constitutes a strong baseline for the classification of time series [2]. One of the main drawbacks of this algorithm is the choice of the k number of neighbors. k is traditionally chosen by cross validation. We propose here a Bayesian approach to the choice of k (inspired by [10]) which is an application of a bayesian online changepoint detection method (from [1]). To our knowledge, there are no publications dealing with a Bayesian KNN applied to the time series classification framework.

Let use consider a data set $D_n = \{(x_1, y_1), ..., (x_n, y_n)\}$. For a new time series X_τ, we try to find the number k of neighbours that we will use to estimate the label of X_τ with a classic KNN.

The purpose of the bayesian KNN is to calculate $\mathbb{P}(k_\tau | \overleftarrow{y}_{0:n-1})$ for all $k_\tau \in \{0, n\}$, the probability of choosing k_τ neighbours knowing the labels of the n time series closest to X_τ. To facilitate the writing of the algorithm, we note y_0 the label of the most distant time series of X_τ, y_{n-1} the label of the closest time series of X_τ etc.

The idea of the Bayesian KNN is to consider that according to an appropriate neighbourhood, the data follow the same distribution. Having ordered our data $\overleftarrow{y}_{0:n-1}$ with respect to X_τ, we can therefore refer to a problem of bayesian online changepoint detection where the quantity we are trying to calculate corresponds to the a posteriori run length distribution, i.e. the distance covered before changing distribution.

We will choose the following a priori laws:

- $k_\tau \sim Geometric(p_\gamma)$. We model our situation so that p_γ is the probability that we have a point of change between two neighbors. In other words,

$$\mathbb{P}(k_t|k_{t-1}) = \begin{cases} p_\gamma & \text{if } k_t = 0 \\ 1 - p_\gamma & \text{if } k_t = k_{t-1} + 1 \\ 0 & \text{otherwise} \end{cases}$$

- $(Y_j)_{j \in [\![1:n]\!]}$ are i.i.d. such that $\mathbb{P}(Y = L_i) = q_i$ with $\forall i \in [\![1:m]\!]$, $0 \leq q_i$ and $\sum_{i=1}^{m} q_i = 1$.
- Thus, Dirichlet's law is naturally chosen as the prior conjugate for the parameters with $(q_1, ..., q_m) \sim Dir(\alpha_1, ..., \alpha_m)$.

In practice , we need to choose p_γ such that $\frac{1}{p_\gamma}$ is the number of neighbors that we think is a priori optimal for the set we are evaluating.

We will also use the following notations to explicit the update of the law's parameters, here a Dirichlet:

- η_{prior} is the a priori parameters, i.e., $\eta_{prior} = (\alpha_1, ..., \alpha_m)$.
- Given a vector $Y_{0:t}$, $\eta(Y_{0:t})$ are the updated parameter of the a posteriori law given $Y_{0:t}$, i.e., $\eta(Y_{0:t}) = (\alpha_1 + \sum_{k=0}^{t} 1_{\{Y_k=1\}}, ..., \alpha_m + \sum_{k=0}^{t} 1_{\{Y_k=m\}})$.
  For the sake of clarity we will use the notation $\eta(k, \overleftarrow{y}_{0:t})$ instead of $\eta(\overleftarrow{y}_{t-k+1:t})$ to express the use of the vector of the k nearest values.
- Then we have :

$$\begin{aligned} \mathbb{P}(y_{target} = j|\eta(k, \overleftarrow{y}_{0:t})) &= \mathbb{E}[q_j|\eta_i(k_{t-1})] \\ &= \frac{\alpha_j + \sum_{l=0}^{k_{t-1}-1} 1_{y_{t-l}=j}}{k_{t-1} + \sum_{k=1}^{m} \alpha_k} \end{aligned} \quad (1)$$

We can now implement the algorithm 1 and choose k_{bayes} the Bayesian k such that:

$$\mathrm{k}_{bayes}(X_\tau) = E(k_\tau|\overleftarrow{y}_{0:n-1}) = \sum_{i=1}^{n} i \times \mathbb{P}(k_\tau = i|\overleftarrow{y}_{0:n-1})$$

Finaly, we can use k_{bayes} as number of neighbor in a KNN to predict the label of X_τ.

Algorithm 1: Efficient Bayesian KNN for TSC

```
Initialization:
$\overleftarrow{y}_{0:n-1} = (y_0, ..., y_{n-1}) \leftarrow$ ordered data label for target point $\tau$ using DTW
$\mathbb{P}(k_0 = 0) \leftarrow 1$
$\eta_0 \leftarrow \eta_{prior}$
for $t \leftarrow 0, n-1$ do
    Observe next variable $y_t$
    for $i \leftarrow 0, t$ do
        $k_{t-1} = i$
        Compute predictive probability
        $\pi_i = \mathbb{P}(y_t|\eta(k_{t-1}, \overleftarrow{y}_{0:t-1}))$
        Compute growth probability
        $\mathbb{P}(k_t = k_{t-1} + 1, \overleftarrow{y}_{0:t}) =$
          $\mathbb{P}(k_{t-1}, \overleftarrow{y}_{0:t-1})\pi_i(1 - p_\gamma)$
    end
    Compute change-point probability
    $\mathbb{P}(k_t = 0, \overleftarrow{y}_{0:t}) = \sum_{k_{t-1}} \mathbb{P}(k_{t-1}, \overleftarrow{y}_{0:t-1})\pi_0 p_\gamma$
    Compute evidence
    $\mathbb{P}(\overleftarrow{y}_{0:t}) = \sum_{k_t} \mathbb{P}(k_t, \overleftarrow{y}_{0:t})$
    for $i \leftarrow 0, t$ do
        Compute probability of $k$
        $\mathbb{P}(k_t = i|\overleftarrow{y}_{0:t}) = \frac{\mathbb{P}(k_t=i, \overleftarrow{y}_{0:t})}{\mathbb{P}(\overleftarrow{y}_{0:t})}$
    end
end
return $\mathbb{P}(k_\tau|\overleftarrow{y}_{0:n-1}) \; \forall k_\tau \in \{0, ..., n\}$
```

This Bayesian approach to the choice of k has two main advantages. It allows you to choose k without going through a procedure using cross validation, which is time-consuming in the case of KNN. In addition, unlike the traditional KNN where one must choose a k for the entire Test set, one uses a k_{bayes} for each element to be classified.

We now need to find a distance adapted to time series. The distance that comes naturally first to mind is the Euclidean distance. However, this one has several disadvantages. Firstly, this distance is not robust to time shifts or dilatation. For example, if we take two identical series but slightly shift one, then the Euclidean distance may consider that these series are very different from each other. Secondly, the Euclidean distance only works with series of the same length, which is not realistic in a large number of applications (duration of a surgical operation, etc.).

A correct approach will be to use Dynamic Time Warping (DTW). (More details in Chapter 4 of [8])

Definition 3. [Warping Path] With $(N, M) \in \mathbb{N}^2$, a (N, M)-wraping path is a sequence $w = (w_1, ..., w_L)$ with

$w_l = (n_l, m_l) \in [\![1 : N]\!] \times [\![1 : M]\!]$ for $l \in [\![1 : L]\!]$ satisfying:

- Boundary condition: $w_1 = (1,1)$ and $w_L = (N, M)$
- Step size condition: $w_{l+1} - w_l \in \{(1,0),(1,1),(0,1)\}$

We remark that $max(N, M) \leq L \leq N + M$

Given a distance d on $\mathbb{R}$, which we will choose to be the euclidean distance here, we then define the distance between two times series with respect to a wraping path.

Definition 4. For two univariate time series, $X = (x_1, ..., x_{|X|})$ and $Z = (z_1, ..., z_{|Z|})$, we note $D_{X,Z}$ the function defined on the space of all possible wrap path between X and Z such that

$$D_{X,Z}(w) = \sum_{l=1}^{L} d(x_{w_{l,1}}, z_{w_{l,2}}) = \sum_{l=1}^{L} |x_{w_{l,1}} - z_{w_{l,2}}|$$

where w is a given wrapping path between X and Y and L is the length of w.

We then define the optimal wrap path.

Definition 5. [Optimal Warp Path] The optimal warp w^* path between two univariate time series $X = (x_1, ..., x_{|X|})$ and $Z = (z_1, ..., z_{|Z|})$ is

$$w^*_{X,Z} = argminD_{X,Z}(w)$$

Definition 6. [DTW distance] The Dynamic Time Wrapping distance between X and Z is then now defined by

$$DTW(X, Z) = D_{X,Z}(w^*_{X,Z})$$

The dynamic time warping algorithm has an $O(N^2)$ time and space complexity, we will use the FastDTW algorithm [12] for implementation, which has a linear time and space complexity.

4. Bayesian Hidden Markov Model

Prior to the wide success of deep learning and its subsequent extended use in many classification tasks, Hidden Markov Models (HMM) were and still are a very popular tool for the modeling of time series and their classification [11]. It has been shown that probabilistic modeling of time series using HMM perform well in many classification tasks and can also help to significantly improve the accuracy of other classifiers [3]. More generally, HMM can be viewed as a special case of Dynamic Bayesian Networks [9] which allow to model and classify a number of time-varying sequences such as human-gesture trajectories [13].

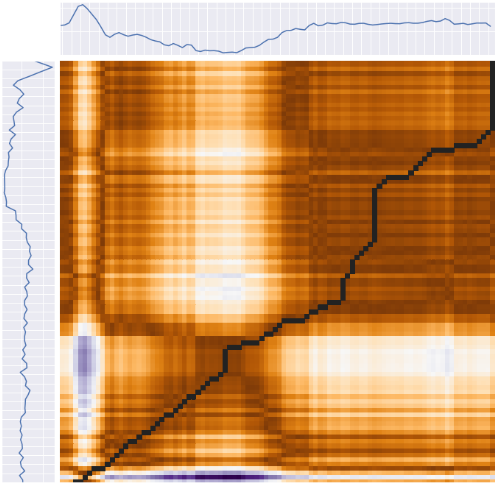

Figure 1. DTW path (in black) for first and second training examples of the ECG200 dataset.

In a HMM-like model, we assume that there are some underlying hidden states. These hidden states evolve over time and influence the generation of the observations. In a classification framework, these hidden states can be viewed as labels. In a Bayesian setting, we can then consider them as random variable with a given distribution with parameters to which we assign a prior. We discuss here how we can use these to perform classification on time series.

In this section we only need one series to explain our model, hence we will denote by X the time series studied and X_t the t^{th} value of X (in the previous section, x_i was the i^{th} series of our dataset).

Definition 7. [Hidden Markov Model (HMM)] Let H_t and X_t ($t \geq 1$) be discrete-time stochastic processes. The pair (H_t, X_t) is a Hidden Markov Model if :

- H_t is a Markov Process not observable (hidden).
- $\mathbb{P}(X_t \in A | H_1, \ldots, H_t) = \mathbb{P}(X_t \in A \, | H_t)$ for any $t \geq 1$ and a mesurable set A.

The process $H = (H_t)_{t=1,\ldots,|X|}$ will be called the hidden states. In our setting, H_t will take discrete values in $\{1, \ldots, K\}$ where K is fixed and known. A simple HMM model can be graphically represented as shown in figure 2.

Let's consider the following example to understand what could be the meaning of these hidden states. Assume we have a time series $X = (X_t)_{t \in [\![1,|X|]\!]}$ coming from an ECG (electrocardiogram) recorded during a two-hour period during which the subject slept half of the time. Here we can consider that the hidden states can take two

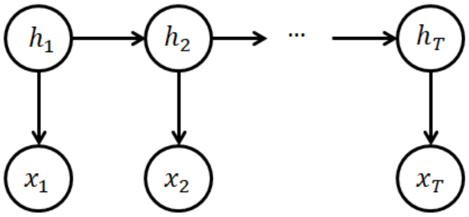

Figure 2. A simple HMM

values (e.g. $H_t \in \{0,1\}$) indicating whether the subject is sleeping at time t. We can directly see that the model assumes different behaviors (distributions) of X_t depending on H_t.

After having specified a conditional distribution on $X_t|H_t$, a first approach to estimate the parameters of such models is to maximize the likelihood. In our discrete-state setting, estimating the distribution of H_t is the same as estimating the transition matrix of the associated Markov Chains.

We assume the following Bayesian parametrization.

- $X_t|H_t = k \sim \mathcal{N}(\mu_k, \sigma_k^2)$ for every $(t,k) \in [\![1,\dots,|X|]\!] \times [\![1,\dots,K]\!]$. In plain words, we assume that for each class of the hidden states, our series is normally distributed with mean and variance depending on the class. In order to lighten the notations, we will also be noting $\theta_k := (\mu_k, \sigma_k^2)$.
- $(\mu_k, \tau_k) \sim NormalGamma(\mu_0, \lambda_0, \alpha_0, \beta_0)$ where $\tau_k := \sigma_k^{-2}$.
- For the transition matrix $P = (p_{ij})_{i,j\in[1,\dots,K]}$ of H we chose the following prior distribution :

  $$(p_{i1},\dots,p_{iK}) \sim Dirichlet(\alpha_{i1},\dots,\alpha_{iK}) \tag{2}$$

  This means that each row of the transition matrix follows a Dirichlet distribution assumed independent from each other. We set α the matrix containing all the $(\alpha_{ij})_{i,j=1,\dots K}$.

Our Bayesian framework is detailed on figure 3.

We note $N_k = Card(\{H_i \ s.t. \ H_{i-1} = k\})$. Computations detailed in Appendix A.2 let us obtain the following posterior:

$$\pi(P|H,\alpha) = \prod_{k=1}^{K} Dirichlet(\alpha_{1k}+N_{1k},\dots,\alpha_{Kk}+N_{Kk})$$

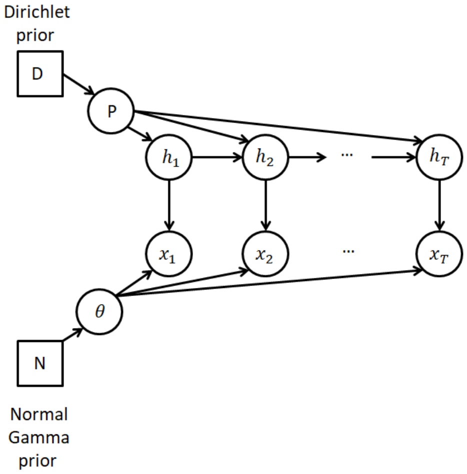

Figure 3. Bayesian priors to our Bayesian Hidden Markov Model

The posterior distribution of each row given the observations is then conjugated :

$$(p_{i1},\dots,p_{iK})|H,\alpha \sim Dirichlet(\alpha_{i1}+N_{i1},\dots,\alpha_{iK}+N_{iK})$$

We now focus on estimating our parameters. As only the time series X is known in our setting, we estimate the parameters by sampling from the full posterior distribution $\pi(P,\theta,H|X)$. To do so, we use a block Gibbs sampler to simulate the hidden states and a regular Gibbs sampler to estimates the parameters. We alternate between the following at each iteration i:

$$\begin{aligned} H^i &\sim \pi(H|\Theta^{i-1}, X) \\ \Theta^i &\sim \pi(\Theta|H^i, X) \end{aligned} \tag{3}$$

where Θ refers to all parameters to be estimated.

The block Gibbs sampler corresponds to the first sampling and refers to the fact that hidden states are sample all at once. To sample the hidden states we will need to use the *backward algorithms* [4]. We give the computations needed to obtain the marginal posterior distributions in Appendix A.1 and the backwrd algorithm in A.3. Following the two steps from (3), the sampling will go as described in the following paragraphs for each iteration.

First step of the sampling:

The markov chain structure of the hidden states implies that we can simulate a sequence H with :

$$\mathbb{P}(H_1 = j \mid \Theta) = \frac{1}{K}\phi(X_1 \mid \mu_j, \sigma_j)\pi(X_{2:|X|} \mid H_1 = j)$$

$$\mathbb{P}(H_t = j \mid H_{t-1} = i) = p_{ij}\phi(X_t \mid \mu_j, \sigma_j)\pi(X_{t+1:|X|} \mid H_t = j)$$

where:

- $B[t, j] := \pi(X_{t+1:|X|} \mid H_t = j)$ corresponds the *backward variables* and can be computed recursively in a efficient manner using the Backward algorithms.
- $\phi(x_t \mid \mu_j, \sigma_j)$ refers to the probability density function of a Gaussian distribution of mean μ_j and standard deviation σ_j evaluated at X_t.

Second step of the sampling:

Given the sequence H simulated in step one, we draw at each iteration from:

$$\begin{aligned} a_{i1}, \ldots, a_{iK} \mid H &\sim Dirichlet(n_{i1} + \alpha_{i1}, \ldots, n_{iK} + \alpha_{iK}) \\ \sigma_s^{-2} \mid x^{[s]}, H &\sim Gamma(c_s, d_s) \\ \mu_s \mid \sigma_s^{-2}, x^{[s]}, H &\sim \mathcal{N}(a_s, b_s^{-1}\sigma_s^2) \end{aligned}$$

where $n_{ik} = Card(\{2 \leq t \leq T,\ H_{t-1} = i,\ H_t = k\})$ is the number of times that the hidden states sequence goes from i to k and $x^{[s]}$ is the set $\{x_t,\ h_t = s\}$. The parameters $a_s,\ b_s,\ c_s$ and d_s correspond to the parameter we obtain when deriving the posterior densities involved here. The derivations are presented in Appendix A.2. The first draw is done once for each row and the distribution corresponds to the posterior density of row i given the previously simulated hidden sequence H. The second and the third draws correspond to the posterior of the parameters μ_s and σ_s^{-2}, the drawings are done for each state s. The dots stand for every other variable.

The two steps of the sampling we just detailed correspond to *one iteration* of (3).

This model can be used for classification and consists in two steps. First, the training data must be separated between each class. Then one HMM model is fitted to each subset. Given a new sequence to be classified and the fitted HMM models, we compute the likelihood that the series was generated by each model and select the class corresponding to the highest likelihood obtained doing so. We compute this likelihood using the *Forward algorithm* for which we provide details in Appendix A.4.

5. Comments on the HMM structure

This section tries to show in which cases the HMM model should perform well and in which it shouldn't.

We simulate a dataset according to HMM process defined earlier to visualize and interpret the convergence of the estimators. The structure we assumed for the data in the HMM model implies that y_t follows a mixture of Gaussian distributions. If these distributions are *far enough* from each other, we expect the model to provide good estimates of the parameters. If the opposite is true, then the estimates should have large variances.

Consider a first example where we have a sample of times series generated as follows. $H_1 \sim U(\{0, 1, 2\})$ and $\forall t \geq 2,\ \mathbb{P}(H_t = j | H_{t-1} = i) = a_{i,j}$ with

$$A := (a_{i,j})_{i,j} = \begin{pmatrix} 0.83 & 0.06 & 0.11 \\ 0.14 & 0.75 & 0.11 \\ 0.15 & 0.20 & 0.65 \end{pmatrix}$$

Given this we generate two time series. In order to do this we generate a sequence H and then for each t, we simulate $X_t | H_t = k \sim \mathcal{N}(\mu_k, \sigma_k)$ for $k \in \{0, 1, 2\}$.

For the first sequence we use : $(\mu_0, \mu_1, \mu_2) = (-2, 0, 2)$ and $\sigma_k = 0.5$. For the second sequence $(\mu_0, \mu_1, \mu_2) = (-1, 3, 1)$ and $(\sigma_0, \sigma_1, \sigma_2) = (2, 1.5, 1.5)$.

Figure 4 shows the density of both these simulated sequence, and we can see that the one generated under small variances (i.e. chosen so that there is little overlapping) seems much more separable, in some sense, that the high variance (chosen so that there is large overlapping) one in which we cannot distinguish between the different gaussians.

Using our algorithms to estimate the parameters, we expect very good performances on the first sequence and a very poor one on the second. Figures 5 and 6 illustrate this phenomenon.

The same results can be observed for the variances estimators. In the high variance setting, the model is not able to distinguish between the different components of the Gaussian mixture and the estimates are very poor. Therefore, a good thing to do before using this model would be to check the density of the data.

6. Implementation

For the implementation of the Bayesian KNN, we have adapted in python a function of the open-source package sktime [7] which implements a k-nearest neighbors algorithm for time series. In addition to implementing the algorithm 1, many code adaptations had to be made to allow a different number of neighbors to be used for each element of the Test set. We optimized our calculations with the Numba package [6].

The bayesian Hidden Markov Model described earlier was implemented from scratch by ourselves under the form of a python class.

Dataset	Train	Test	Length	Classes	Type
ECG200	100	100	96	2	ECG
CricketX	390	390	300	12	Motion
Adiac	390	391	176	37	IMAGE
Epilepsy	137	138	207	4	HAR
Sim1	87	43	100	2	Simulated
Sim2	117	58	100	3	Simulated

Table 1. Characteristics of the 6 selected univariate time series data sets

Dataset	KNN	Bayesian KNN	Bayesian HMM
ECG200	72%	84%	64%
CricketX	73%	74%	10%
Adiac	56%	56%	8%
Epilepsy	95%	96%	27%
Sim1	67%	65%	58%
Sim2	59%	55%	46%

Table 2. Results of our different approaches on the 6 datasets

All our codes and results can be found on the GitHub dedicated to this project: https://github.com/DatenBiene/Bayesian_Time_Series_Classification

7. Results

To evaluate the performance of our algorithms we decided to test them on 6 datasets: 2 data sets are simulated and 4 from the database https://timeseriesclassification.com [2]. The characteristics of the choosen datasets are reported in Table 1.

The performances of the algorithms are reported in Table 2. The Bayesian KNN proved to be very efficient. Compared to the traditional KNN where the number of neighbours k is chosen by cross validation, the Bayesian KNN is significantly faster and always superior or equivalent in terms of performance.

The Bayesian HMM seem to give poor results on most dataset we tested here, especially when dealing with real data. In addition to generally having significantly lower accuracy than Bayesian KNN, our Bayesian HMM code is on average 60 times slower. For each data set, the number of hidden states in the HMM model was chosen by plotting the empirical density of the concatenated vector of each label of the train set and counting the number of picks. An example is given in Figure 7.

8. Difficulties and Areas of Improvement

The Bayesian KNN code could be written even more efficiently. Other distance measures could be tested.

The HMM technique presented here is not able to perform decently on most real dataset we experimented. This comes from the assumptions made by the structure of the model. The HMM model we studied here is a simple one. A first approach to complexify the model would be to consider the number of hidden states K as random and estimate it as well. Further work may explore more complex structure to treat time series classification.

Both methods could be improved to handle multivariate time series.

References

[1] R. P. Adams and D. J. C. MacKay. Bayesian online changepoint detection, 2007.

[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2016.

[3] R. K. F. G. T. Bilal Esmael, Arghad Arnaout. Improving time series classification using hidden markov models. May 2014.

[4] S. Chib. Calculating posteriors distributions and modal estimates in markov mixture models. *Journal of Econometrics*, 1996.

[5] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, Feb 2019.

[6] S. K. Lam, A. Pitrou, and S. Seibert. Numba. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM 15*, 2015.

[7] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király. sktime: A unified interface for machine learning with time series, 2019.

[8] M. Meinard. *Information retrieval for music and motion*. Springer, 2010.

[9] K. P. Murphy. Dynamic bayesian networks: Representation, inference and learning. *UNIVERSITY OF CALIFORNIA, BERKELEY*, 2002.

[10] G. Nuti. An efficient algorithm for bayesian nearest neighbours. *Methodology and Computing in Applied Probability*, 21(4):1251–1258, 2018.

[11] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *IEEE*, page 257–286, 2019.

[12] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, Oct 2007.

[13] T. S. H. Vladimir Pavlovi, Brendan J. Freyz. Time-series classification using mixed-state dynamic bayesian networks. *ECE Department and Beckman Institute University of Illinois*.

A. Appendix

A.1. Posterior distribution for the transition matrix

We first give a justification for the prior choice. Let us consider a sample $H_1, \dots, H_T$ of discrete random variables in $\{1, \dots, K\}$ satisfying the markov property. Using the Markov property we have that $\pi(H_1, \dots, H_T) = \pi(H_1) \cdot \prod_{t=2} \pi(H_t|H_{t-1})$. This probability can be written in a more convenient way. Considering the fact that each random variable can take one of K different values, if we denote $H = (h_1, \dots, h_T)$, we can write $H = \{h_1\} \bigcup H^1 \bigcup \dots \bigcup H^K$ where we define the set H^k as the set of all values succeeding the value k in the sequence. Formally, we have $H^k = \{h_i \ s.t. \ h_{i-1} = k\}$. We also introduce $N_k = |H^k|$. We can now say that each element of H^k was independently drawn from the multinomial distribution $\pi(h_i|h_{i-1} = k, P)$ which is parametrized by K numbers corresponding to the row k^{th} row of P, we call this row p_k. In other words, we have $\pi(h_i = j|h_{i-1} = k, P) = p_{kj}$ by definition of the transition matrix. Using this we can write the following :

$$\begin{aligned} \pi(H|P) &= \pi(h_1) \cdot \prod_{k=1}^{K} \pi(h^k|p_k) \\ &= \pi(h_1) \cdot \prod_{k=1}^{K} \prod_{j=1}^{K} p_{jk}^{N_{jk}} \end{aligned} \tag{4}$$

where $N_{jk} = \sum_{i=2}^{T} 1_{h_{i-1}=j} \cdot 1_{h_i=k}$ which counts the number of transition from state j to state k. We can now show that the prior given in (2) is conjugate. Using previous notation we can infer the prior for the whole matrix P:

$$\begin{aligned} \pi(P) &= \prod_k p_k \\ &= \prod_k \frac{\Gamma(\sum_j \alpha_{jk})}{\Gamma(\alpha_{jk})} \cdot \prod_j p_{kj}^{\alpha_{jk}-1} \end{aligned}$$

By combining this result with (4) we get the joint distribution :

$$\begin{aligned} \pi(H, P|\alpha) &= \pi(h_1) \cdot \prod_k \pi(H^k, p_k|\alpha) \\ &= \pi(h_1) \cdot \prod_k \frac{\Gamma(\sum_j \alpha_{jk})}{\Gamma(\alpha_{jk})} \cdot \prod_j p_{kj}^{N_{jk}+\alpha_{jk}-1} \end{aligned}$$

Where α denotes the matrix containing all the $(\alpha_{ij})_{i,j=1,\dots K}$. This results in the posterior distribution:

$$\begin{aligned} \pi(P|H, \alpha) &= \prod_k \pi(p_k|H^k, \alpha) \\ &= \prod_k Dirichlet(N_{1k} + \alpha_{1k}, \dots, N_{Kk} + \alpha_{Kk}) \end{aligned}$$

Finally we have that the posterior distribution of each row given the observations is : $(p_{i1}, \dots, p_{iK}) \sim Dirichlet(\alpha_{i1} + N_{i1}, \dots, \alpha_{iK} + N_{iK})$. We will use the form of this posterior distribution to update our parameters in the first step of the Gibbs sampling.

A.2. Posterior distributions for the Gibbs sampler

We first focus on obtaining the posterior densities involved in the second sampling step as the sampling of the hidden state sequence is straightforward as explained later on.

The Hidden Markov Model structure implies the following :

$$\begin{aligned} &H_{t+1}|H_t \perp\!\!\!\perp (X_{1:t}, H_{1:t-1}) \\ &X_t|H_t \perp\!\!\!\perp (X_{1:t-1}, X_{t+1:T}, H_{1:t-1}, H_{t+1:T}) \end{aligned}$$

Which implies that we have :

$$\begin{aligned} \pi(H_{1:T}, X_{1:T}|\Theta) = \pi(H_1|\Theta) \cdot \prod_{t=1}^{T-1} \pi(H_{t+1}|H_t, \Theta) \\ \cdot \prod_{t=1}^{T} \pi(X_t|H_t, \Theta) \end{aligned}$$

Therefore, using the Bayes rules we have that :

$$\begin{aligned} \pi(\Theta|H_{1:T}, X_{1:T}) = \pi(P|H_{1:T}) \cdot \prod_{s \in C} \prod_{t \in s} \pi(\Theta|X_t) \\ \propto \pi(H_{1:T}|P) \cdot \pi(P) \cdot \prod_{s \in C} \prod_{t \in s} (\pi(X_t|\theta)\pi(\mu_s|\sigma_s)\pi(\sigma_s)) \end{aligned} \tag{5}$$

Where C is the partition of the set $\{1, \dots, T\}$ such that $|C| = K$ and $\forall s \in C, \ \forall t, t' \in s, \ h_t = h_{t'}$.

Now, looking at this posterior distribution, we can see that the first two components of this product will give the marginal posterior distribution of the transition matrix P which we already have computed. We can now look at the last term of the product which is a sum over C in which term k will give the marginal posterior distribution of μ_k, σ_k. We can then focus on one term of this product over C and the others will have the same posterior distribution but with different parameters. More precisely, we are looking at the distribution, for a given hidden state s

$$\prod_{t\in s}\pi(X_t|\theta)\pi(\mu_s|\sigma_s)\pi(\sigma_s)$$

If we note $x^{[s]}$ the set $\{x_t,\ t\in s\}$, n_s its cardinal and $c_s := \frac{1}{n_s}\sum_{t\in s}(x_t - \overline{x^{[s]}})^2$ then we have the following.

$$\begin{aligned}\pi(x^{[s]} \mid \tau_s,\mu_s) &\propto \prod_{t\in s}\tau_s^{1/2}\exp\left[\frac{-\tau_s}{2}(x_t-\mu_s)^2\right]\\ &\propto \tau^{n_s/2}\exp\left[\frac{-\tau_s}{2}\sum_{t\in s}(x_t-\mu_s)^2\right]\\ &\propto \tau^{n_s/2}\exp\left[\frac{-\tau_s}{2}\sum_{t\in s}(x_t-x^{\bar{[s]}}+x^{\bar{[s]}}-\mu)^2\right]\\ &\propto \tau^{n_s/2}\exp\left[\frac{-\tau_s}{2}\left(n_s\cdot c_s+n_s\cdot(x^{\bar{[s]}}-\mu_s)^2\right)\right]\end{aligned}$$

Thus,

$$\begin{aligned}\pi(\tau_s,\mu_s \mid x^{[s]}) &\propto \pi(x^{[s]} \mid \tau_s,\mu_s)\pi(\tau_s,\mu_s)\\ &\propto \tau^{n_s/2}\exp\left[\frac{-\tau_s}{2}\left(n_s\cdot c_s+n_s(x^{\bar{[s]}}-\mu_s)^2\right)\right]\\ &\cdot\ \tau_s^{\alpha_0-\frac{1}{2}}\ \exp[-\beta_0\tau_s]\ \exp\left[-\frac{\lambda_0\tau_s(\mu_s-\mu_{s0})^2}{2}\right]\\ &\propto \tau_s^{\frac{n_s}{2}+\alpha_0-\frac{1}{2}}\exp\left[-\tau_s\left(\frac{1}{2}n_s\dot{c}_s+\beta_0\right)\right]\\ &\cdot\exp\left[-\frac{\tau_s}{2}\left(\lambda_0(\mu_s-\mu_{s0})^2+n_s(x^{\bar{[s]}}-\mu_s)^2\right)\right]\end{aligned}$$

Remark that

$$\begin{aligned}&\lambda_0(\mu_s-\mu_{s0})^2\\ &+n_s(x^{\bar{[s]}}-\mu_s)^2\\ &=(\lambda_0+n_s)\left(\mu_s-\frac{\lambda_0\mu_{s0}+n_sx^{\bar{[s]}}}{\lambda_0+n_s}\right)^2\\ &+\frac{\lambda_0n_s(x^{\bar{[s]}}-\mu_{s0})^2}{\lambda_0+n_s}\end{aligned}$$

And we get that :

$$\begin{aligned}&\pi(\tau_s,\mu_s \mid x^{[s]})\\ &\propto \tau_s^{\frac{n_s}{2}+\alpha_0-\frac{1}{2}}\exp\left[-\tau_s\left(\tfrac{1}{2}n_s\cdot c_s+\beta_0+\tfrac{\lambda_0n_s(x^{\bar{[s]}}-\mu_0)^2}{2(\lambda_0+n_s)}\right)\right]\\ &\cdot\exp\left[-\tfrac{\tau_s}{2}(\lambda_0+n_s)\left(\mu_s-\tfrac{\lambda_0\mu_{s0}+n_sx^{\bar{[s]}}}{\lambda_0+n_s}\right)^2\right]\end{aligned}$$

Therefore,

$$\pi(\tau_s,\mu_s \mid x^{[s]}) = \text{NormalGamma}\left(a_s,b_s,c_s,d_s\right)$$

with

$$\begin{aligned}a_s &= \frac{\lambda_0\mu_0+n_sx^{\bar{[s]}}}{\lambda_0+n_s}\\ b_s &= \lambda_0+n_s\\ c_s &= \alpha_0+\frac{n_s}{2}\\ d_s &= \beta_0+\frac{1}{2}\left(n_s\cdot c_s+\frac{\lambda_0n_s(x^{\bar{[s]}}-\mu_0)^2}{\lambda_0+n_s}\right)\end{aligned}$$

A.3. The Backward algorithm

Definition 8. [Backward probabilities] Backward probabilities give the probability of observing the remaining observations given any starting point $\beta_t := P(X_{t+1:T} \mid H_t)$.

In section 4, we used these probabilities to simulate a sequence of hidden states.

- 1. Initialization:
  $\beta_1(j) = 1\ 1 \le j \le K$
- 2. Recursion :
  $\beta_t(j) = \sum_{i=1}^{K}\beta_{t+1}(j)\cdot p_{ij}\cdot P(X_{t+1} \mid H_{t+1}=j)$
- 3. Output :
  $\beta_{t,j}\forall t,j$

This algorithm can output $P(X_{1:T} \mid \Theta)$ similarly to the next one.

A.4. The Forward algorithm

In this section we explain how we computed the likelihood of a time series given a HMM models with known parameters. Formally, we want to compute $P(X_{1:T}|\Theta)$. If we knew the sequence of hidden states, then computing this probability would be straightforward : $P(Y_{1:T}|H_{1:T}) = \prod_{t=1}^{T}P(X_t|H_t)$. Since we don't know the states, computing the probability requires requires summing over all possible sequences of hidden states. So the probability of observing a sequence $X_{1:T}$ given a model would be :

$$P(X_{1:T} \mid \Theta) = \sum_{H_{1:T}\in S}P(X_{1:T},H_{1:T} \mid \Theta) =$$

$$\sum_{H_{1:T}\in S}P(X_{1:T} \mid H_{1:T},\Theta)P(H_{1:T}|\Theta)$$

Where is the set of all possible sequence of hidden states of length T which is of cardinal K^T resulting in an huge computational cost and therefore inefficiency. The forward algorithme solve this issue by providing a method to compute this probability with a $O(K^2 \cdot T)$ complexity.

- 1. Initialization:
  $\alpha_1(j) = P(H_1 = j) \cdot P(X_1 \mid H_1 = j)\ 1 \leq j \leq K$
- 2. Recursion :
  $\alpha_t(j) = \sum_{i=1}^{K} \alpha_{t-1}(j) \cdot p_{ij} \cdot P(X_t \mid H_t = j)$
- 3. Output :
  $P(X_{1:T} \mid \Theta) = \sum_{i=1}^{K} \alpha_T(i)$

These steps are implemented in our python package *Bayesian hmm*.

A.5. Figures

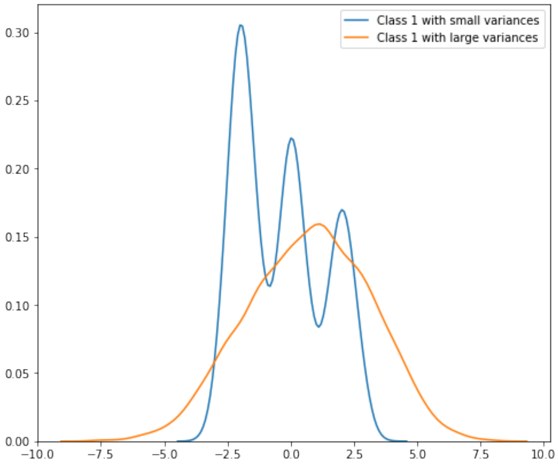

Figure 4. Density of observation from class with little and large overlapping.

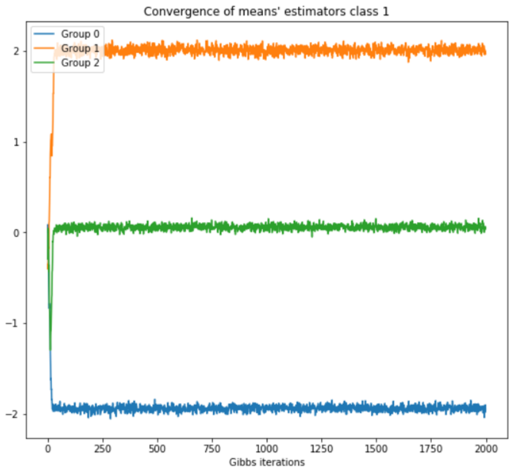

Figure 5. Estimator of the means in the little overlapping case

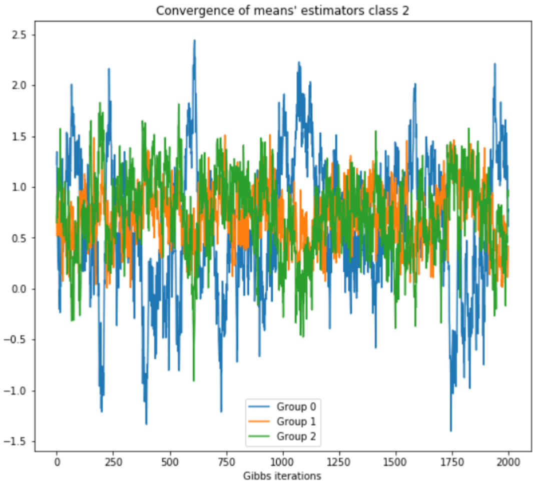

Figure 6. Estimator of the means in the large overlapping case

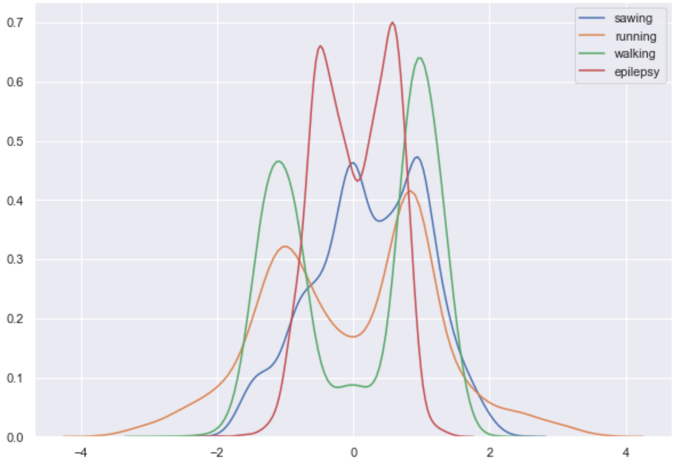

Figure 7. Empirical density for each classes of the Epilepsy dataset