UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di
Economia e Management

Corso di Laurea in
Statistica

Modelli markoviani latenti per serie temporali

Relatore
Alessandro Magrini

Candidato
Manuel Soffici

Anno Accademico 2022/2023

Hidden Markov Models for time series

Manuel Soffici

2023-07-13

Table of contents

Preface

To my brother, Mattia

It is with great pleasure and a sense of accomplishment that I present my Bachelor's thesis. It is largely based on W. Zucchini, I. MacDonald, R. Langrock, *Hidden Markov Models for Time Series: An Introduction Using R*, Second Edition, Chapman & Hall, 2016. And an important source of inspiration has been R. Langrock and J. Dyck's teaching material for the 2021 course on HMMs at the University of Bielefeld, Germany.

This work is the culmination of my studies in Statistics at the University of Florence, which I embarked upon almost three years ago. The field of Statistics has long fascinated me, and my insatiable curiosity for knowledge has driven me to delve deeper into the scientific art of learning from data. During my time in Florence, I had the privilege of refining my skills and knowledge under the guidance of dedicated professors, who allowed me to learn and grow.

I am particularly grateful for the support and mentorship provided by my thesis advisor, Alessandro Magrini, whose expertise and guidance have been invaluable throughout this endeavor. His patience, encouragement, and insightful feedback have significantly shaped the outcome of this thesis. Furthermore, I would like to extend my sincere appreciation to all the faculty members at DISIA for providing a great learning environment and fostering a spirit of academic excellence.

I would also like to acknowledge the invaluable support and guidance provided over the last four months by Cornelia U. Kunz and Julia Stoehr, my supervisors at Boehringer Ingelheim. Their help has not been instrumental for this thesis specifically, but rather for my overall professional and personal growth.

Finally, I am immensely grateful to my family and friends for their unwavering support, encouragement, and understanding throughout this journey. Their love and belief in my abilities have been constant sources of inspiration and motivation.

This Bachelor's thesis is not just the result of my individual efforts, but a culmination of the collective support provided by many. Thank you all for being part of this.

Manuel Soffici

July 13, 2023

1 Introduction

Hidden Markov Models serve as valuable statistical models for analyzing time series data. Developed in the 1960s initially to address speech recognition tasks, they have since found widespread application across various domains. They offer a framework that involves two stochastic processes: an observed response process, representing the time series data, and a latent state process, which operates as a Markov chain. In this framework, each observation is generated by one of N possible distributions. The latent state process determines which distribution is active at any given time, with the state at time t being dependent solely on the state at time $t-1$.

It is worth noting that the literature on Hidden Markov Models can be broadly categorized into two distinct branches:

- the engineering literature. This branch, from which they were originated, primarily focuses on recognition or classification problems. Typically, models are calibrated using training data, where the states are explicitly observed, employing a supervised learning approach. The objective is then to predict the latent states for new data
- the statistical literature. This branch is instead concerned with general inference regarding the studied phenomenon. Within fields such as economics and finance, Hidden Markov Models find utility in forecasting purposes and other forms of statistical analysis.

To delve deeper into the topic, we shall explore the theoretical foundations, statistical properties, and practical applications of these models in the following sections.

2 Preliminaries

In this section, we will lay the foundation for understanding the basic structure of Hidden Markov Models (HMMs). Indeed, it is crucial to familiarize ourselves with two fundamental concepts: independent mixture distributions and Markov chains. They serve as building blocks for comprehending the underlying principles of HMMs.

2.1 Independent mixtures

In an HMM with N states, each observation is generated by one of N component distributions, forming a dependent mixture model. To begin our exploration, we will first delve into independent mixture distributions: dependent mixtures will be explored later.

We define a general mixture model by considering densities $f_1, f_2, \ldots, f_K$ characterizing K component distributions. The mixing weights, denoted as $\pi_1, \pi_2, \ldots, \pi_K$, satisfy $\sum_{j=1}^{K} \pi_j = 1$ with $\pi_j \in [0, 1]$. The mixture density function can then be expressed as:

$$f(x) = \sum_{j=1}^{K} \pi_j f_j(x) = \pi_1 f_1(x) + \pi_2 f_2(x) + \ldots + \pi_K f_K(x)$$

In an independent mixture model, we assume that realizations from the mixture are independent of each other.

How do we fit a mixture model to the data? Suppose we have a set of n independent realizations $x_1, x_2, \ldots, x_n$ from the mixture distribution $f(x)$. We can define the likelihood function as:

$$\mathcal{L}(x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i) = \prod_{i=1}^{n} \sum_{j=1}^{K} \pi_j f_j(x_i)$$

and by taking the logarithm we obtain the log-likelihood:

$$\log \mathcal{L} = \sum_{i=1}^{n} \log(\sum_{j=1}^{K} \pi_j f_j(x_i))$$

We estimate the parameters by numerically maximizing the log-likelihood with respect to the model parameters, including the mixing weights and the parameters defining the component distributions. It is worth noting that alternative approaches to parameter estimation exist. Although the component distributions can belong to any suitable class, normal distributions are commonly employed in most cases.

Fitting mixture distributions enables us to effectively model complex and multimodal distributions. However, addressing serial correlation requires the use of Markov chains.

2.2 Markov chains

Independent mixture models assume observations to be independent. Nevertheless, time series data often exhibit strong serial correlation due to system inertia. Therefore, an appropriate mixture mechanism should favor the selection of the same component distribution at time $t+1$ as the one selected at time t. Markov chains provide a framework to model this principle.

A Markov chain is a stochastic process $\{S_t; t = 1, 2, ...\}$ that satisfies the following properties:

1. $S_t \in \{1, \ldots, N\}, \forall t$, where N denotes the number of states
2. $\Pr(S_{t+1} = s_{t+1} | S_t = s_t, S_{t-1} = s_{t-1}, \ldots, S_1 = s_1) = \Pr(S_{t+1} = s_{t+1} | S_t = s_t)$

The second point, known as the Markov property, implies that the distribution of S_{t+1} is solely determined by $S_t = s_t$. This dependence structure is both mathematically convenient and often plausible.

To characterize a Markov chain, we need to specify two key elements:

1. the initial state distribution $\delta^{(1)} = (\delta_1^{(1)}, \ldots, \delta_N^{(1)}) = (\Pr(S_1 = 1), \Pr(S_1 = 2), \ldots, \Pr(S_1 = N))$
2. the (one-step) state-transition probabilities $\gamma_{ij}^{(t)} = \Pr(S_{t+1} = j | S_t = i)$.

A Markov chain is said to be homogeneous if $\gamma_{ij}^{(t)} = \gamma_{ij}, \forall t$, indicating that the transition probabilities remain constant over time. In this case, we can represent the transition matrix as:

$$\Gamma = \begin{pmatrix} \gamma_{11} & \cdots & \gamma_{1N} \\ \vdots & \ddots & \vdots \\ \gamma_{N1} & \cdots & \gamma_{NN} \end{pmatrix}$$

For non-homogenous Markov chains, we would introduce superscripts (t) to indicate time-varying probabilities.

It is evident that $\gamma_{ij} \in [0,1], \forall t$, since they represent probabilities. Additionally, the summation of transition probabilities over all states, $\sum_{j=1}^{N} \gamma_{ij} = 1$, guarantees that the N states are exhaustive.

Other useful quantities are the m-step transition probabilities $\gamma_{ij}(m) = \Pr\left(S_{t+m} = j \mid S_t = i\right)$. Let $\Gamma^{(m)}$ denote the corresponding m-step transition matrix. We can derive the Chapman-Kolmogorov equations (given without proof):

1. $\gamma_{ij}(m+n) = \sum_{k=1}^{N} \gamma_{ik}(m)\gamma_{kj}(n)$
2. $\Gamma^{(m+n)} = \Gamma^{(m)}\Gamma^{(n)}$
3. $\Gamma^{(k)} = \Gamma^{k}$.

These equations allow us to easily compute forecasts and conditional distributions with the one-step transition probabilities.

Furthermore, consider the unconditional distribution of the states at time t:

$$\delta^{(t)} = \left(\delta_1^{(t)}, \dots, \delta_N^{(t)}\right) = \left(\Pr\left(S_t = 1\right), \dots, \Pr\left(S_t = N\right)\right)$$

We observe the following relationships:

1. $\delta^{(t)} = \delta^{(t-1)}\Gamma$
2. $\delta^{(t)} = \delta^{(1)}\Gamma^{(t-1)} = \delta^{(1)}\Gamma^{t-1}$.

Finally, we introduce the stationary distribution of a Markov chain. The stationary distribution is the $\delta \in \mathbb{R}^N$ such that $\delta\Gamma = \delta$ with $\sum_{i=1}^{N} \delta_i = 1$. If $\delta^{(t)} = \delta$, then $\delta^{(t+1)} = \delta^{(t)}\Gamma = \delta\Gamma = \delta$. It is highly desirable for a Markov chain to converge over time to a unique stationary distribution. If a Markov chain exhibits this property, then δ gives the probabilities of encountering the process in any of the N states, so we can reasonably assume that a real process is in its stationary distribution when we start to observe it, since it will have been running for some time already.

A homogeneous Markov chain is called stationary if it starts in its stationary distribution, i.e., if $\delta^{(1)} = \delta$. In this case the process maintains the same unconditional distribution at all subsequent time points. Since a stationary Markov chain spends about $100\delta_j\%$ of the time in state j, the stationary distribution δ is a valuable summary statistic.

3 Model formulation and properties

Hidden Markov Models offer a high degree of flexibility and versatility, enabling the analysis of diverse data types with various dependence structures. In this section, we will explore the structure of HMMs and examine their key properties.

In an independent mixture we consider N component distributions, where one of them is selected independently at each time t. So $S_t \perp S_{t^*}, \forall t \neq t^*$, and in particular $\Pr(S_t = j \mid S_{t-1} = i) = \pi_j$ does not depend on i. In contrast, an HMM incorporates a Markov chain $\{S_t\}$ as a latent state process. Thus, the state S_t is not independent of the state at other time steps. Specifically, $\Pr(S_t = j \mid S_{t-1} = i) = \gamma_{ij}$, so the activation of i at time $t-1$ influences the distribution of states at time t.

An N-state Hidden Markov model is a (doubly) stochastic process in discrete time. It consists of an unobserved state process $S_1, S_2, \ldots, S_T$ taking values from $\{1, \ldots, N\}$, and an observed state-dependent process $X_1, X_2, \ldots, X_T$. The following properties characterize an HMM:

1. Markov property. The probability of state s_t, given the history of states $s_1, ..., s_{t-1}$, is equivalent to the probability of state s_t, given only the previous state s_{t-1}. This property is expressed as $f(s_t \mid s_1, \ldots, s_{t-1}) = f(s_t \mid s_{t-1})$
2. conditional independence. The probability of observation x_t, given the history of states $s_1, ..., s_t$, and observations $x_1, ..., x_{t-1}$, is equivalent to the probability of observation x_t, given the current state s_t. In other words, the observations are conditionally independent, given the current state. This property is expressed as $f(x_t \mid s_1, \ldots, s_t, x_1, \ldots, x_{t-1}) = f(x_t \mid s_t)$

The Markov property ensures that the state process $\{S_t\}$ is fully characterized by the initial distribution $\delta^{(1)}$ and the transition matrix Γ. On the other hand, the conditional independence assumption guarantees that the observed sequence $\{X_t\}$ is fully characterized by the state-dependent distributions $f(x_t \mid s_t)$. It is worth stressing that observations are independent only within states, as the Markov chain introduces dependence in the state-dependent process.

As anticipated, an HMM can be viewed as a dependent mixture model. If the state process $\{S_t\}$ is stationary, then the marginal distribution of X_t is:

$$f(x_t) = \sum_{j=1}^{N} \delta_j^{(t)} f_j\left(x_t\right) = \sum_{j=1}^{N} \delta_j f_j\left(x_t\right)$$

Lastly, it is important to acknowledge that the states in an HMM do not necessarily correspond to meaningful or interpretable entities in an unsupervised learning context. Instead, an HMM captures the most dominant patterns with respect to multimodality. Depending on the temporal resolution, certain patterns might not be inferred at all.

4 Parameter estimation

Various methods, like maximum likelihood estimation (MLE), the expectation-maximization (EM) algorithm, and Bayesian techniques can be employed to estimate the parameters of an HMM. In this section we will focus on the first without focusing extensively on technical details.

4.1 Naive likelihood evaluation

Given a sequence of observations $x_1, \dots, x_T$, the likelihood is defined as the joint density or probability of the observations conditioned on the parameter vector θ:

$$\mathcal{L}(\theta) = f_\theta\left(x_1, \dots, x_T\right) = f(x_1, \dots, x_T | \theta)$$

MLE seeks the parameter vector θ that maximizes $\mathcal{L}(\theta)$. In practice, it is often more convenient to (equivalently) maximize the log-likelihood function $\ell(\theta) = \log \mathcal{L}(\theta)$ to avoid numerical underflow issues, particularly in HMMs.

The likelihood can be (naively) computed summing over all possible state sequences:

$$\mathcal{L}(\theta) = \sum_{s_1=1}^{N} \sum_{s_2=1}^{N} \dots \sum_{s_T=1}^{N} \left(\prod_{t=1}^{T} f_{s_t}\left(x_t\right) \right) \left(\delta_{s_1}^{(1)} \prod_{t=2}^{T} \gamma_{s_{t-1}, s_t} \right)$$

However, this approach is computationally complex or even unfeasible for most cases due to the exponential number of summands N^T.

4.2 The forward algorithm

Fortunately, a more efficient approach exists: the forward algorithm allows us to compute the likelihood in a recursive manner. The key idea is to use the following quantities:

$$\alpha_t(j) = f\left(x_1, \dots, x_t, S_t = j\right)$$

These joint densities carry information on the likelihood of the sequence of observations up to time t and on the probability of the state j being active at time t. We define $\alpha_t = (\alpha_t(1), \dots, \alpha_t(N))$ as the t-th forward variable.

The first forward variable can be computed as:

$$\alpha_1 = \delta^{(1)}\mathbf{P}\left(x_1\right)$$

$\delta^{(1)}$ is the initial state distribution, and $\mathbf{P}\left(x_t\right) = \operatorname{diag}\left(f_1\left(x_t\right), \dots, f_N\left(x_t\right)\right)$. Given that the diagonal entries of $\mathbf{P}\left(x_t\right)$ are the conditional densities $f_j(x_t) = f(x_t|S_t = j)$, then the j-th entry of the first forward variable is $\alpha_1(j) = f(x_1|S_1 = j)\Pr(S_1 = j) = f(x_1, S_1 = j)$, as defined. Then:

$$\alpha_2 = \alpha_1\Gamma\mathbf{P}\left(x_2\right), \alpha_3 = \alpha_2\Gamma\mathbf{P}\left(x_3\right), \dots, \alpha_T = \alpha_{T-1}\Gamma\mathbf{P}\left(x_T\right)$$

where Γ is the transition matrix.

Finally, by the law of total probability, the likelihood can be expressed as:

$$\mathcal{L}(\theta) = \sum_{j=1}^{N} f\left(x_1, \dots, x_T, S_T = j\right) = \sum_{j=1}^{N} \alpha_T(j)$$

4.3 Numerical maximization

Since there is no analytical solution for the MLE in HMMs, we resort to numerical maximization methods. Thanks to the linear computational complexity of the forward algorithm, maximizing the likelihood becomes feasible. One common approach is to use iterative algorithms such as the Newton-Raphson method.

The Newton-Raphson method approximates the roots of a function by iteratively finding the points where the tangent line to the curve intersects the x-axis. The iterations continue until a stable solution is reached. However, applying this method to HMMs requires addressing several challenges:

1. the transition probabilities and possibly other parameters are constrained
2. for large samples the likelihood tends to be very small, leading to numerical underflow
3. the algorithm might converge to local maxima instead of the global maximum.

To overcome these issues, various strategies have been proposed. However, exploring them is beyond the scope of this thesis.

5 Model selection and checking

In this section, we discuss how to choose the most adequate models among a set of alternatives and we define key quantities that enable us to assess how well a chosen model fits the observed data.

5.1 Model selection

A fundamental task is to identify the most suitable models from a set of candidate options. Two criteria are widely used for HMM selection: the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). These criteria offer valuable insights into model performance by assessing the trade-off between goodness of fit and model complexity.

The AIC quantifies the estimated discrepancy, measured through the Kullback-Leibler divergence, between a given candidate model and the hypothetical "true" model. It is calculated as $AIC = -2\log\mathcal{L} + 2p$, where $\mathcal{L}$ represents the likelihood and p denotes the number of parameters in the model. Conversely, the BIC takes into account the posterior probability of a model being the "true" model, given the observed data. It is defined as $BIC = -2\log\mathcal{L} + \log(n)\cdot p$, where n represents the sample size.

Although the formulations of AIC and BIC are similar, their complexity penalties are different. Since $\log(n) > 2$ for $n \geq 8$, the BIC has a (virtually always) stronger complexity penalty than the AIC. It is crucial to recognize that these criteria have serious limitations:

1. they provide only relative comparisons between alternative models and do not guarantee the adequacy of the chosen models
2. they are only asymptotically exact solutions to their respective optimization problems: inaccuracies may be large with small sample sizes
3. a strong assumption underlies both criteria: that the "true" model exists within the set of candidate models, which may often be an unreasonable assumption.

In the context of HMMs, it is worth noting that AIC and BIC tend to favor models with a higher number of states. In fact, the underlying assumptions regarding the Markov property, conditional independence, and the form of the state-dependent distributions are often only approximately valid. This means that HMMs are often convenient but not necessarily realistic representations of the phenomena of interest. Outliers can also favor states inflation, of course. Additional states tend to reflect these otherwise ignored patterns.

Models with a large number of states might provide accurate forecasts but may lack practical interpretability. To mitigate this issue, careful and precise model formulation is advised, although some level of discrepancy is inevitable. Consideration of prior information can also help inform the selection of the appropriate number of states, denoted as N.

An alternative approach to model selection is cross-validation, a widely used technique in machine learning. In its basic form, cross-validation involves splitting the sample into calibration and validation subsets. The candidate models are fitted using the calibration data and then evaluated based on their performance on the validation data. However, standard cross-validation is not well-suited for time series models and presents additional challenges specific to HMMs. Therefore, this approach will not be explored further in this context.

5.2 Model diagnostics

To assess the adequacy of a fitted HMM, various diagnostic methods can be employed:

1. comparing the marginal distribution obtained from the fitted model with the empirical distribution of the observed data
2. simulating data from the fitted model and comparing the patterns (like marginal distributions, autocorrelation functions, etc.) of the simulated data with those of the real data
3. running a formal residual analysis for the fitted model.

Since each X_t has a different distribution, it is hard to evaluate the residuals directly. But a common scale can be obtained (for continuous distributions) through a probability integral transform:

1. let X_t have a cumulative distribution function F_{X_t}. Then $F_{X_t}(X_t)$ follows a uniform distribution on the interval $[0,1]$

2. if $U \sim \text{Uniform}[0,1]$, then $\Phi^{-1}(U)$ follows a standard normal distribution $\mathcal{N}(0,1)$
3. consequently, $\Phi^{-1}\left(F_{X_t}\left(X_t\right)\right) \sim \mathcal{N}(0,1)$.

It is essential to note that the above result holds approximately for large sample sizes when the cumulative distribution function F_{X_t} is estimated.

By employing the forward and backward variables, a conditional distribution can be obtained:

$$F_{X_t}\left(x_t\right) = \Pr\left(X_t \leq x_t \mid X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_{t+1} = x_{t+1}, \ldots, X_T = x_T\right)$$

from which the (ordinary) pseudo-residuals $\Phi^{-1}\left(F_{X_t}\left(X_t\right)\right)$ can be derived by applying the probability integral transform. Note that the distribution F_{X_t} is conditional on all the other observations. For discrete data, pseudo-residuals need to be defined differently: not as points, but as intervals.

If the fitted model is correct, these pseudo-residuals will approximately follow a standard normal distribution. However, it is crucial to note that pseudo-residuals measure the deviation from the median rather than the expectation. Systematic departures from normality may indicate inadequacies in the model, and the presence of extreme observations could suggest differences in nature or origin compared to the rest of the data. Furthermore, it is essential to examine residual autocorrelation to ensure that the model accurately captures the correlation structure.

An important point must be highlighted: the crucial property of pseudo-residuals is their approximately identical distribution. Instead, the assumption of independence between pseudo-residuals may often be violated and is not of utmost importance for our analysis.

Additionally, we introduce another type of pseudo-residuals, known as forecast pseudo-residuals. They can be derived similarly to ordinary pseudo-residuals, and they provide a valuable diagnostic tool by measuring the deviation of an observation from the median of the corresponding one-step-ahead forecast distribution.

Unlike ordinary pseudo-residuals, forecast pseudo-residuals depend only on preceding observations and not the entire dataset. Thus, extreme forecast pseudo-residuals indicate observations that deviate from the pattern of preceding observations and cannot be adequately explained by the model. They allow for continuous monitoring of the behavior of a time series.

6 Prediction and state decoding

In this section, we explore forecasting and state decoding, two fundamental applications of HMMs.

6.1 Forecasting future observations

To predict future observations, we can use the h-step ahead forecast distribution, which is given by the equation:

$$f\left(x_{T+h} \mid x_1, \ldots, x_T\right) = \phi_T \Gamma^h \mathbf{P}\left(x_{T+h}\right) 1^t$$

Here ϕ_T denotes the standardized forward variable, which provides probabilities of the Markov chain being in the different states at time T. As h increases, $\phi_T \Gamma^h$ converges to the stationary distribution of the Markov chain. Consequently, the asymptotic forecast distribution corresponds to the marginal distribution of the observations.

We can easily observe that since the Markov chain tends to settle quickly into its stationary distribution, which state is active at time t is relevant only for short horizons.

6.2 State decoding

Once a model is fitted, we often seek to decode the hidden states underlying the observed time series. There are two commonly employed approaches, which generally yield similar results: local decoding and global decoding. We will briefly describe them without delving into technical intricacies.

In local decoding, each time point is considered independently. At time t, the state selected is the one with the highest $\Pr(S_t = i | x_1, \ldots, x_T)$ among $i = 1, \ldots, N$. To evaluate these probabilities, we can use backward probabilities, which can be recursively computed akin to forward probabilities.

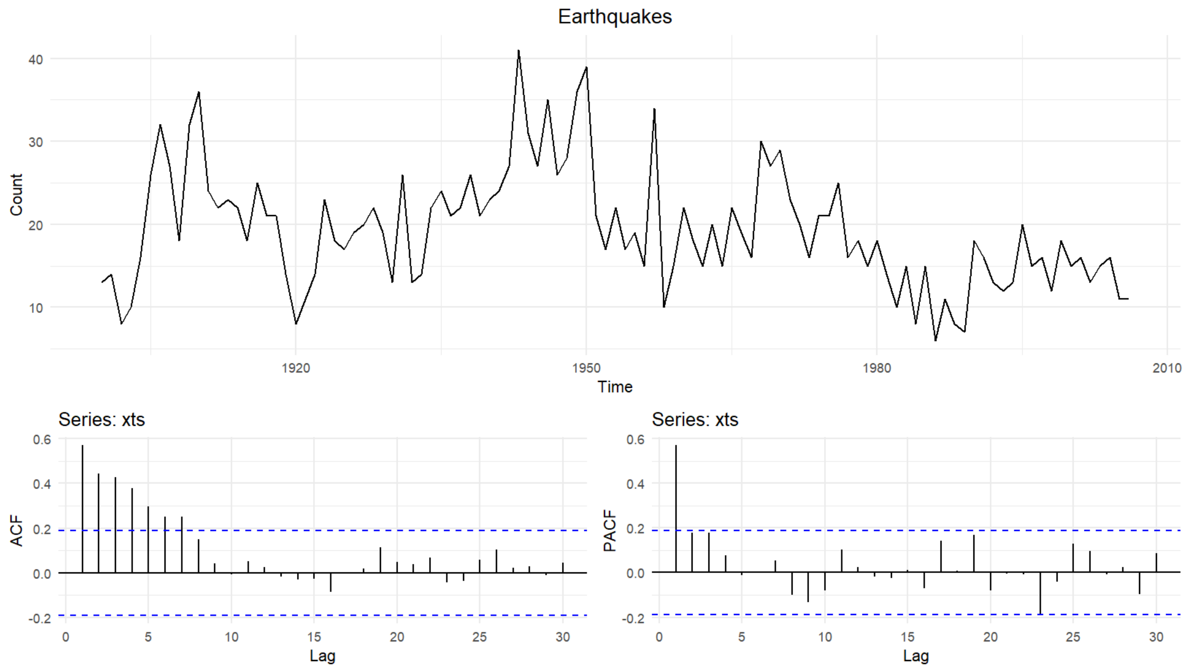
On the other hand, global decoding considers the entire time series jointly. The sequence of states chosen is the sequence $(i_1, \ldots, i_T) \in \{1, \ldots, N\}^T$ that maximizes $\Pr(S_1 = i_1, \ldots, S_T = i_T \mid x_1, \ldots, x_T)$. Due to the vast number of possible state sequences, a direct approach is impractical. Instead, we turn to the Viterbi algorithm, which provides a recursive solution.

While global decoding is more common and arguably more reasonable, local decoding offers a distinct advantage: it quantifies the uncertainty associated with state inference at each time point. This evaluation becomes crucial when the costs of a misclassification are high, such as in clinical settings.

As we have previously discussed, when the meaning of the states is unknown, state decoding in HMMs can be viewed as a form of unsupervised learning (cluster analysis). However, if we possess labeled data with known states and fit the model accordingly, state decoding for new data becomes a form of supervised learning (classification).

7 Application: earthquakes

The series we analyze is obtained from Zucchini et al., *Hidden Markov Models for Time Series: An Introduction Using R*, 2016, which has also been the main source of content for this thesis. It is a series of annual counts of earthquakes with a magnitude of 7 or higher that occurred worldwide from 1900 to 2006. Therefore, there are 107 observations, all of which are positive. Many of these observations fall between 15 and 20, approximately. From the sample autocorrelation function (ACF) graph, it is evident that there is a fairly strong autocorrelation up to a lag of 4 years, and it remains greater than zero up to 7 years.

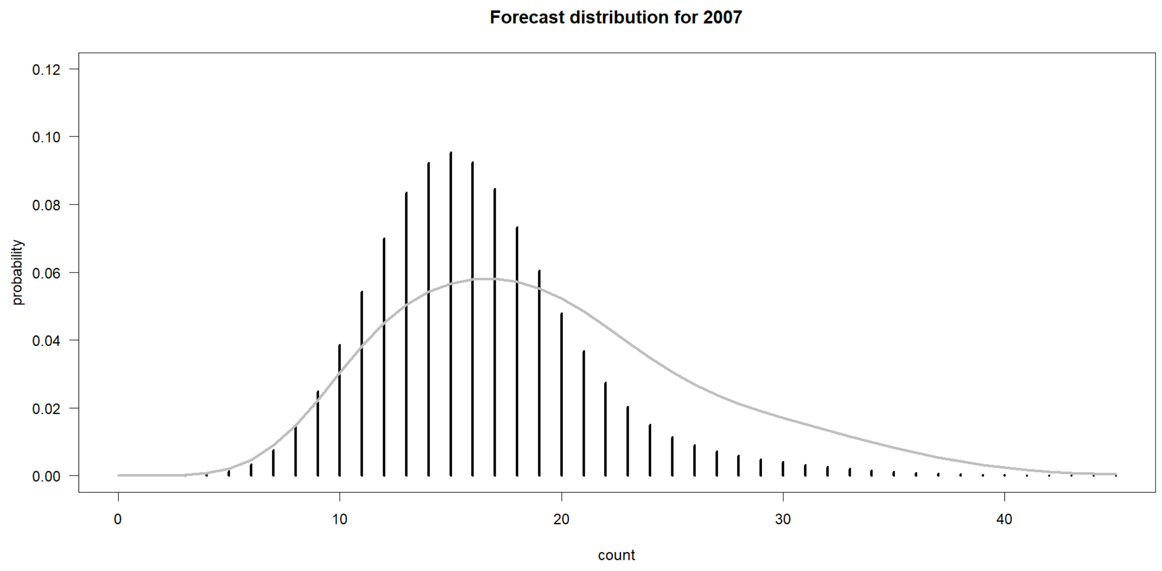


Now, for unbounded counts, the Poisson distribution is the standard. However, assuming that these observations are independent realizations of a single Poisson distribution (effectively an HMM with a single state) is not very realistic because, as we have seen, there is autocorrelation, but there is also significant overdispersion. In fact, the sample variance is approximately two and a half times the sample mean. Therefore, we fit HMM models with state-dependent Poisson distributions and an increasing number of states, while always enforcing chain homogeneity (but not necessarily stationarity).

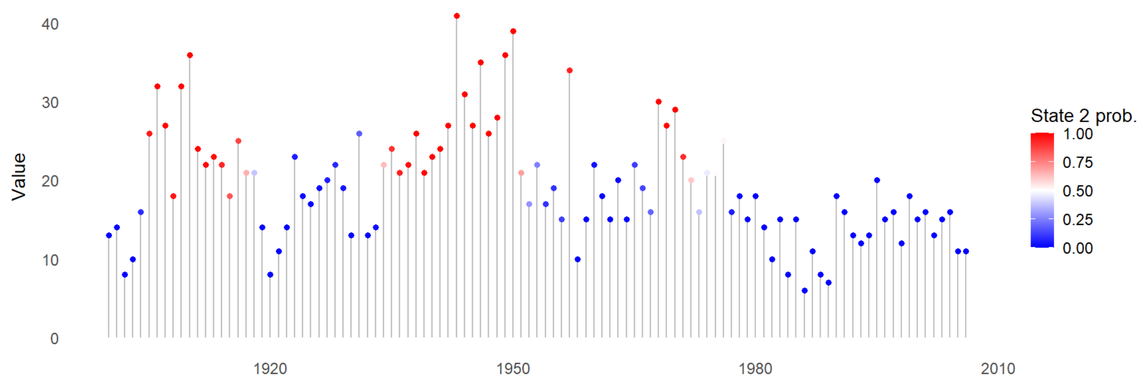
We observe that the AIC and BIC decrease with the 4-state models, so we immediately exclude them and focus on the 2- and 3-state models. As generative models, they perform quite well, but the non-stationary 3-state model tends to simulate samples with ACF plots that are sensibly different from the original series. However, the residuals are practically perfect for all models.

$$\Gamma = \begin{pmatrix} 0.934 & 0.066 \\ 0.129 & 0.871 \end{pmatrix}, \lambda_1 = 15.472, \lambda_2 = 26.125$$

Here are the estimated parameters and the asymptotic predictive distribution of the stationary 2-state model. In my opinion, this is the preferable model for two reasons. Firstly, the 3-state models have many transition probabilities that are not significantly different from zero (which is typical in series that are not very long). Secondly, with a very small N, stationarity is definitely not a heavy assumption.



And here we have the stationary distribution of the states and the series with local decoding. It can be observed that the two align fairly well: the first state, which generates the lowest numbers, appears to be much more frequently active than the second one.



7.1 Executable R code (with selected outputs)

Here is the R code that was used for this section, based on the script provided by the authors of the reference textbook, cited in the following bibliography section.

```r
library(ggplot2)
library(forecast)
library(lmtest)
library(ggthemes)
library(gridExtra)

set.seed(496)
par(mar=c(1,1,1,1))

### A.1.1 Transforming natural parameters to working
pois.HMM.pn2pw <- function(m,lambda,gamma,delta=NULL,stationary=TRUE) {
  tlambda <- log(lambda)
  if(m==1) return(tlambda)
  foo <- log(gamma/diag(gamma))
  tgamma <- as.vector(foo[!diag(m)])
  if(stationary) {
    tdelta <- NULL
  } else {
    tdelta <- log(delta[-1]/delta[1])
  }
  parvect <- c(tlambda, tgamma, tdelta)
  return(parvect)
```

```r
}

### A.1.2 Transforming working parameters to natural
pois.HMM.pw2pn <- function(m,parvect,stationary=TRUE) {
  lambda <- exp(parvect[1:m])
  gamma <- diag(m)
  if (m==1) return(list(lambda=lambda, gamma=gamma, delta=1))
  gamma[!gamma] <- exp(parvect[(m+1):(m*m)])
  gamma <- gamma/apply(gamma, 1, sum)
  if(stationary) {
    delta <- solve(t(diag(m)-gamma+1), rep(1, m))
  } else {
    foo <- c(1, exp(parvect[(m*m+1):(m*m+m-1)]))
    delta <- foo/sum(foo)
  }
  return(list(lambda=lambda, gamma=gamma, delta=delta))
}

### A.1.3 Computing minus the log-likelihood from the working parameters
pois.HMM.mllk <- function(parvect, x, m, stationary=TRUE, ...) {
  if(m==1) return(-sum(dpois(x, exp(parvect), log=TRUE)))
  n <- length(x)
  pn <- pois.HMM.pw2pn(m, parvect, stationary=stationary)
  foo <- pn$delta * dpois(x[1], pn$lambda)
  sumfoo <- sum(foo)
  lscale <- log(sumfoo)
  foo <- foo/sumfoo
  for (i in 2:n) {
    if(!is.na(x[i])) {
      P <- dpois(x[i], pn$lambda)
    } else {
      P <- rep(1, m)
    }
    foo <- foo %*% pn$gamma * P
    sumfoo <- sum(foo)
    lscale <- lscale + log(sumfoo)
    foo <- foo/sumfoo
  }
  mllk <- -lscale
  return(mllk)
```

```
}

### A.1.4 Computing the MLEs, given starting values for the natural parameters
pois.HMM.mle <- function(x,m,lambda0,gamma0,delta0=NULL,stationary=TRUE,...) {
  parvect0 <- pois.HMM.pn2pw(m, lambda0, gamma0, delta0, stationary=stationary)
  mod <- nlm(pois.HMM.mllk, parvect0, x=x, m=m, stationary=stationary)
  pn <- pois.HMM.pw2pn(m=m, parvect=mod$estimate, stationary=stationary)
  mllk <- mod$minimum
  np <- length(parvect0)
  AIC <- 2*(mllk + np)
  n <- sum(!is.na(x))
  BIC <- 2*mllk + np*log(n)
  list(m=m, lambda=pn$lambda, gamma=pn$gamma, delta=pn$delta, code=mod$code,
       mllk=mllk, AIC=AIC, BIC=BIC)
}

### A.1.5 Generating a sample
pois.HMM.generate_sample <- function(ns, mod) {
  mvect <- 1:mod$m
  state <- numeric(ns)
  state[1] <- sample(mvect, 1, prob=mod$delta)
  for (i in 2:ns) {
    state[i] <- sample(mvect, 1, prob=mod$gamma[state[i-1], ])
  }
  x <- rpois(ns, lambda=mod$lambda[state])
  return(x)
}

### A.1.6 Global decoding by the Viterbi algorithm
pois.HMM.viterbi <- function(x, mod) {
  n <- length(x)
  xi <- matrix(0, n, mod$m)
  foo <- mod$delta * dpois(x[1], mod$lambda)
  xi[1, ] <- foo/sum(foo)
  for (i in 2:n) {
    foo <- apply(xi[i-1, ] * mod$gamma, 2, max) * dpois(x[i], mod$lambda)
    xi[i, ] <- foo/sum(foo)
  }
  iv <- numeric(n)
  iv[n] <- which.max(xi[n, ])
  for (i in (n-1):1) {
    iv[i] <- which.max(mod$gamma[, iv[i+1]] * xi[i, ])
  }
  return(iv)
}

### A.1.7 Computing log(forward probabilities)
pois.HMM.lforward <- function(x, mod) {
  n <- length(x)
  lalpha <- matrix(NA, mod$m, n)
  foo <- mod$delta * dpois(x[1], mod$lambda)
  sumfoo <- sum(foo)
  lscale <- log(sumfoo)
  foo <- foo/sumfoo
  lalpha[, 1] <- lscale + log(foo)
  for (i in 2:n) {
    foo <- foo %*% mod$gamma * dpois(x[i], mod$lambda)
    sumfoo <- sum(foo)
    lscale <- lscale + log(sumfoo)
    foo <- foo/sumfoo
    lalpha[, i] <- log(foo) + lscale
  }
  return(lalpha)
}

### A.1.8 Computing log(backward probabilities)
pois.HMM.lbackward <- function(x, mod) {
  n <- length(x)
  m <- mod$m
  lbeta <- matrix(NA, m, n)
  lbeta[, n] <- rep(0, m)
  foo <- rep(1/m, m)
  lscale <- log(m)
  for (i in (n-1):1) {
    foo <- mod$gamma %*% (dpois(x[i+1], mod$lambda) * foo)
    lbeta[, i] <- log(foo) + lscale
    sumfoo <- sum(foo)
    foo <- foo/sumfoo
```

```r
    lscale <- lscale + log(sumfoo)
  }
  return(lbeta)
}

### A.1.9 Conditional probabilities
#==Conditional probability that observation at time t equals
#  xc, given all observations other than that at time t.
#  Note: xc is a vector and the result (dxc) is a matrix.
pois.HMM.conditional <- function(xc, x, mod) {
  n <- length(x)
  m <- mod$m
  nxc <- length(xc)
  dxc <- matrix(NA, nrow=nxc, ncol=n)
  Px <- matrix(NA, nrow=m, ncol=nxc)
  for (j in 1:nxc) Px[, j] <- dpois(xc[j], mod$lambda)
  la <- pois.HMM.lforward(x, mod)
  lb <- pois.HMM.lbackward(x, mod)
  la <- cbind(log(mod$delta), la)
  lafact <- apply(la, 2, max)
  lbfact <- apply(lb, 2, max)
  for (i in 1:n) {
    foo <- (exp(la[, i] - lafact[i]) %*% mod$gamma) * exp(lb[, i] - lbfact[i])
    foo <- foo/sum(foo)
    dxc[, i] <- foo %*% Px
  }
  return(dxc)
  }

### A.1.10 Pseudo-residuals
pois.HMM.pseudo_residuals <- function(x, mod) {
  n <- length(x)
  cdists <- pois.HMM.conditional(xc=0:max(x), x, mod)
  cumdists <- rbind(rep(0, n), apply(cdists, 2, cumsum))
  ulo <- uhi <- rep(NA, n)
  for (i in 1:n) {
    ulo[i] <- cumdists[x[i]+1, i]
    uhi[i] <- cumdists[x[i]+2, i]
  }
  umi <- 0.5 * (ulo + uhi)
  npsr <- qnorm(rbind(ulo, umi, uhi))
  return(npsr)
}

### A.1.11 State probabilities
pois.HMM.state_probs <- function(x, mod) {
  n <- length(x)
  la <- pois.HMM.lforward(x, mod)
  lb <- pois.HMM.lbackward(x, mod)
  c <- max(la[, n])
  llk <- c + log(sum(exp(la[, n] - c)))
  stateprobs <- matrix(NA, ncol=n, nrow=mod$m)
  for (i in 1:n) stateprobs[, i] <- exp(la[, i] + lb[, i] - llk)
  return(stateprobs)
}

### A.1.12 State prediction
pois.HMM.state_prediction <- function(h=1, x, mod) {
  n <- length(x)
  la <- pois.HMM.lforward(x, mod)
  c <- max(la[, n])
  llk <- c + log(sum(exp(la[, n] - c)))
  statepreds <- matrix(NA, ncol=h, nrow=mod$m)
  foo <- exp(la[, n] - llk)
  for (i in 1:h) {
    foo <- foo %*% mod$gamma
    statepreds[, i] <- foo
  }
  return(statepreds)
}

### A.1.13 Local decoding
pois.HMM.local_decoding <- function(x, mod) {
  n <- length(x)
  stateprobs <- pois.HMM.state_probs(x, mod)
  ild <- rep(NA, n)
  for (i in 1:n) ild[i] <- which.max(stateprobs[, i])
  return(ild)
}
```

```
### A.1.14 Forecast probabilities
pois.HMM.forecast <- function(xf, h=1, x, mod) {
  n <- length(x)
  nxf <- length(xf)
  dxf <- matrix(0, nrow=h, ncol=nxf)
  foo <- mod$delta * dpois(x[1], mod$lambda)
  sumfoo <- sum(foo)
  lscale <- log(sumfoo)
  foo <- foo/sumfoo
  for (i in 2:n) {
    foo <- foo %*% mod$gamma * dpois(x[i], mod$lambda)
    sumfoo <- sum(foo)
    lscale <- lscale + log(sumfoo)
    foo <- foo/sumfoo
  }
  for (i in 1:h) {
    foo <- foo %*% mod$gamma
    for (j in 1:mod$m) dxf[i, ] <- dxf[i, ] + foo[j] * dpois(xf, mod$lambda[j])
  }
  return(dxf)
}


### Fitting Poisson HMMs to the earthquakes series

dat <- read.table("C:\\Users\\soffi\\Desktop\\earthquakes.txt")
x <- dat[, 2]
d <- dat[, 1]
n <- length(x)

#======================================= fit 2-state HMM
m <- 2
lambda0 <- c(15, 25)
gamma0 <- matrix(
  c(
    0.9, 0.1,
    0.1, 0.9
  ), m, m, byrow=TRUE)
mod2s <- pois.HMM.mle(x, m, lambda0, gamma0, stationary=TRUE)
delta0 <- c(1, 1) / 2
```

```r
mod2h <- pois.HMM.mle(x, m, lambda0, gamma0, delta=delta0, stationary=FALSE)
mod2s
```

```
$m
[1] 2

$lambda
[1] 15.47223 26.12535

$gamma
          [,1]       [,2]
[1,] 0.9340391 0.06596091
[2,] 0.1285104 0.87148957

$delta
[1] 0.6608194 0.3391806

$code
[1] 1

$mllk
[1] 342.3183

$AIC
[1] 692.6365

$BIC
[1] 703.3278
```

```r
mod2h
```

```
$m
[1] 2

$lambda
[1] 15.42071 26.01812

$gamma
          [,1]       [,2]
```

```
[1,] 0.9283743 0.07162567
[2,] 0.1190295 0.88097054

$delta
[1] 9.999948e-01 5.237225e-06

$code
[1] 1

$mllk
[1] 341.8787

$AIC
[1] 693.7574

$BIC
[1] 707.1216
```

```
#======================================== fit 3-state HMM
m <- 3
lambda0 <- c(10, 20, 30)
gamma0 <- matrix(
  c(
    0.8, 0.1, 0.1,
    0.1, 0.8, 0.1,
    0.1, 0.1, 0.8
  ), m, m, byrow=TRUE)
mod3s <- pois.HMM.mle(x, m, lambda0, gamma0, stationary=TRUE)
delta0 <- c(1, 1, 1) / 3
mod3h <- pois.HMM.mle(x, m, lambda0, gamma0, delta=delta0, stationary=FALSE)
mod3s
```

```
$m
[1] 3

$lambda
[1] 13.14573 19.72102 29.71438

$gamma
             [,1]       [,2]       [,3]
```

```
[1,] 9.546238e-01 0.02444335 0.02093284
[2,] 4.976687e-02 0.89936661 0.05086652
[3,] 4.235237e-08 0.19664334 0.80335661

$delta
[1] 0.4436404 0.4045001 0.1518595

$code
[1] 1

$mllk
[1] 329.4603

$AIC
[1] 676.9206

$BIC
[1] 700.976
```

```
mod3h
```

```
$m
[1] 3

$lambda
[1] 13.13374 19.71312 29.70964

$gamma
             [,1]       [,2]       [,3]
[1,] 9.392936e-01 0.03209738 0.02860898
[2,] 4.040127e-02 0.90643712 0.05316160
[3,] 1.849487e-12 0.19025321 0.80974679

$delta
[1] 9.999999e-01 3.171305e-08 2.970722e-08

$code
[1] 1

$mllk
```

```
[1] 328.5275

$AIC
[1] 679.055

$BIC
[1] 708.4561
```

```
#======================================= fit 4-state HMM
m <- 4
lambda0 <- c(10, 15, 20, 30)
gamma0 <- matrix(
  c(
    0.85, 0.05, 0.05, 0.05,
    0.05, 0.85, 0.05, 0.05,
    0.05, 0.05, 0.85, 0.05,
    0.05, 0.05, 0.05, 0.85
  ), m, m, byrow=TRUE)
mod4s <- pois.HMM.mle(x, m, lambda0, gamma0, stationary=TRUE)
delta0 <- c(1, 1, 1, 1) / 4
mod4h <- pois.HMM.mle(x, m, lambda0, gamma0, delta=delta0, stationary=FALSE)
mod4s
```

```
$m
[1] 4

$lambda
[1] 11.28288 13.85317 19.69535 29.69979

$gamma
              [,1]      [,2]       [,3]         [,4]
[1,]  8.048715e-01 0.1018756 0.09325287 1.401597e-08
[2,] 3.078409e-267 0.9760653 0.00000000 2.393467e-02
[3,]  5.012403e-02 0.0000000 0.90172597 4.815000e-02
[4,]  0.000000e+00 0.0000000 0.18811711 8.118829e-01

$delta
[1] 0.09356906 0.39826774 0.36425630 0.14390690

$code
```

```
[1] 1

$mllk
[1] 327.8316

$AIC
[1] 687.6632

$BIC
[1] 730.4284
```

```
mod4h
```

```
$m
[1] 4

$lambda
[1] 11.26322 14.34032 19.66679 29.97209

$gamma
             [,1]         [,2]         [,3]         [,4]
[1,] 7.947941e-01 6.259660e-02 1.426093e-01 2.217113e-09
[2,] 1.185153e-19 1.000000e+00 5.878746e-45 2.037222e-18
[3,] 4.175459e-02 9.748651e-27 8.842467e-01 7.399866e-02
[4,] 2.603597e-42 4.030638e-48 2.129640e-01 7.870360e-01

$delta
[1] 1.000000e+00 4.628494e-08 6.857124e-33 2.034844e-33

$code
[1] 1

$mllk
[1] 326.6749

$AIC
[1] 691.3499

$BIC
[1] 742.1336
```

```
#### Ts plot, acf, pacf of the series

name = "Earthquakes"
unit = "1900-2006"
xts = ts( data = x, start = 1900, frequency = 1)

name <- "Earthquakes"
unit <- "1900-2006"
xts <- ts(data = x, start = 1900, frequency = 1)

df <- data.frame(Time = time(xts), Count = as.numeric(xts))

#plot_ts <- ggplot(df, aes(x = Time, y = Count)) +
#  geom_line() +
#  labs(title = name, y = "Count") +
#  theme_minimal() +
#  theme(plot.title = element_text(size = 14, hjust = 0.5))

#plot_acf <- ggAcf(xts, lag.max = 30) +
#  theme_minimal()

#plot_pacf <- ggPacf(xts, lag.max = 30) +
#  theme_minimal()

#grid.arrange(
#  plot_ts,
#  arrangeGrob(plot_acf, plot_pacf, ncol = 2),
#  nrow = 2,
#  heights = c(3, 2)
#)


#### Simulations

sim_mod2s <- pois.HMM.generate_sample(ns = 107, mod = mod2s)

#par(mfrow = c(3,1))
#plot(sim_mod2s, type = "l", main = name, ylab = unit)
#Acf(x = sim_mod2s, type = "correlation", na.action = na.pass,
#    lag.max = 60, main = name)
#Acf(x = sim_mod2s, type = "partial",     na.action = na.pass,
#    lag.max = 60, main = name)

sim_mod2h <- pois.HMM.generate_sample(ns = 107, mod = mod2h)

#par(mfrow = c(3,1))
#plot(sim_mod2h, type = "l", main = name, ylab = unit)
#Acf(x = sim_mod2h, type = "correlation", na.action = na.pass,
#lag.max = 60, main = name)
#Acf(x = sim_mod2h, type = "partial",     na.action = na.pass,
#lag.max = 60, main = name)

sim_mod3s <- pois.HMM.generate_sample(ns = 107, mod = mod3s)

#par(mfrow = c(3,1))
#plot(sim_mod3s, type = "l", main = name, ylab = unit)
#Acf(x = sim_mod3s, type = "correlation", na.action = na.pass,
#    lag.max = 60, main = name)
#Acf(x = sim_mod3s, type = "partial",     na.action = na.pass,
#    lag.max = 60, main = name)

sim_mod3h <- pois.HMM.generate_sample(ns = 107, mod = mod3h)

#par(mfrow = c(3,1))
#plot(sim_mod3h, type = "l", main = name, ylab = unit)
#Acf(x = sim_mod3h, type = "correlation", na.action = na.pass,
#lag.max = 60, main = name)
#Acf(x = sim_mod3h, type = "partial",     na.action = na.pass,
#lag.max = 60, main = name)


#### Residuals

res_mod2s <- pois.HMM.pseudo_residuals(x, mod2s)[2,]

#par(mfrow = c(3,1))
#plot(res_mod2s, type = "l", main = name, ylab = unit)
#Acf(x = res_mod2s, type = "correlation", na.action = na.pass,
```

```r
#    lag.max = 60, main = name)
#Acf(x = res_mod2s, type = "partial",     na.action = na.pass,
#    lag.max = 60, main = name)

shapiro.test(res_mod2s)
```

```

	Shapiro-Wilk normality test

data:  res_mod2s
W = 0.99175, p-value = 0.7667
```

```r
#res_mod2h <- pois.HMM.pseudo_residuals(x, mod2h)[2,]

#par(mfrow = c(3,1))
#plot(res_mod2h, type = "l", main = name, ylab = unit)
#Acf(x = res_mod2h, type = "correlation", na.action = na.pass,
#lag.max = 60, main = name)
#Acf(x = res_mod2h, type = "partial",     na.action = na.pass,
#lag.max = 60, main = name)

#shapiro.test(res_mod2h)

res_mod3s <- pois.HMM.pseudo_residuals(x, mod3s)[2,]

#par(mfrow = c(3,1))
#plot(res_mod3s, type = "l", main = name, ylab = unit)
#Acf(x = res_mod3s, type = "correlation", na.action = na.pass,
#    lag.max = 60, main = name)
#Acf(x = res_mod3s, type = "partial",     na.action = na.pass,
#    lag.max = 60, main = name)

shapiro.test(res_mod3s)
```

```

	Shapiro-Wilk normality test

data:  res_mod3s
W = 0.99164, p-value = 0.7577
```

```r
res_mod3h <- pois.HMM.pseudo_residuals(x, mod3h)[2,]

#par(mfrow = c(3,1))
#plot(res_mod3h, type = "l", main = name, ylab = unit)
#Acf(x = res_mod3h, type = "correlation", na.action = na.pass,
#lag.max = 60, main = name)
#Acf(x = res_mod3h, type = "partial",     na.action = na.pass,
#lag.max = 60, main = name)

shapiro.test(res_mod3h)
```

```

	Shapiro-Wilk normality test

data:  res_mod3h
W = 0.99187, p-value = 0.7772
```

```r
#### Winner

## Forecasts

h <- 1
xf <- 0:45
dstat <- numeric(length(xf))
forecasts <- pois.HMM.forecast(xf, h, x, mod2s)
fc <- forecasts[h, ]
#par(mfrow=c(1, 1), las=1)
#plot(xf, fc, type="h",
#     main=paste("Forecast distribution for", d[n]+h),
#     xlim=c(0, max(xf)), ylim=c(0, 0.12), xlab="count", ylab="probability",
#lwd=3)
#lines(xf, dstat, col="gray", lwd=3)

## Decoding

local <- pois.HMM.local_decoding(x, mod2s)
state_probs <- pois.HMM.state_probs(x, mod2s)[2,]

df1 <- data.frame(Date = index(xts), Value = coredata(xts), state_probs)
```

```r
red_color <- colorRampPalette(c("white", "red"))(100)
blue_color <- colorRampPalette(c("blue", "white"))(100)
dot_shape <- 21

#ggplot(df1, aes(x = Date, y = Value)) +
#  geom_segment(aes(xend = Date, yend = 0), color = "#C0C0C0", size = 0.5) +
#  geom_point(aes(color = state_probs), size = 1.8) +
#  scale_color_gradientn(colours = c(blue_color, red_color), limits = c(0, 1),
#                        guide = guide_colorbar(title = "State 2 prob.")) +
#  theme_minimal() +
#  theme(plot.title = element_text(size = 18, hjust = 0.5,
#                                  margin = margin(b = 20)),
#        axis.title.x = element_blank(),
#        axis.title.y = element_text(size = 14, margin = margin(r = 10)),
#        axis.text.x = element_text(size = 12, angle = 0, hjust = 0.5,
#                                   vjust = 0.5),
#        axis.text.y = element_text(size = 12),
#        legend.title = element_text(size = 14),
#        legend.text = element_text(size = 12),
#        panel.grid.major = element_blank(),
#        panel.grid.minor = element_blank(),
#        panel.border = element_blank(),
#        plot.margin = margin(20, 20, 20, 20))
```

8 Summary

This is a theoretical review thesis with the aim of introducing the basic hidden Markov models. As we have seen, HMMs are very versatile and often effective, but they have their biggest limitations in their (relative) lack of interpretability and estimation issues, especially with many states.

But, if the latter are properly addressed and the former is not desired nor strictly necessary, they offer many additional opportunities:

- serial dependence between observations can be modelled through an autoregressive structure (Markov-switching models)
- memory can be allowed in the Markov chain (Semi-HMMs)
- covariates can easily be included
- states can be modelled as continuous-valued,

and more.

However, exploring these extensions in detail is beyond the scope of this thesis.

Bibliography

- W. Zucchini, I. MacDonald, R. Langrock, *Hidden Markov Models for Time Series: An Introduction Using R*, Second Edition, Chapman & Hall, 2016. https://www.crcpress.com/p/book/9781482253832
- R. Langrock, R. King, J. Matthiopoulos, L. Thomas, D. Fortin, J.M. Morales (2012), *Flexible and practical modeling of animal telemetry data: hidden Markov models and extensions.* Ecology, 93: 2336-2342. https://doi.org/10.1890/11-2241.1
- L.R. Rabiner (1989), *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, 77, 257-286. http://dx.doi.org/10.1109/5.18626
- M. Stamp (2021), *A revealing introduction to Hidden Markov Models.* Department of Computer Science, San Jose State University. http://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf