Hidden Markov Models and Application in Time Series Analysis*

Yiliu Cao

1 Introduction

In statistics, a Markov Chain is a stochastic process that models a sequence of random variables in which the distribution of the next state depends only on the current state or a limited number of past observations, and we call this structure a Markov process. This model was first introduced by Andrey Markov in the early 20th century (Rabiner, 1989). Building on this idea, Baum proposed the Hidden Markov Model (HMM) which has two sequences of random variables, one to be hidden and another to be observed, with hidden variables modeling as the Markov process (Baum & Petrie, 1966). This paper will introduce the HMM with some important algorithms, and will also apply them to real-world scenarios.

The Hidden Markov Model was first proposed by Baum through a series of papers between the 1960s and 1970s (Baum & Petrie, 1966). The initial settings of HMMs follows that, suppose $\{X_t\}$ with stochatic matrix $\{a_{ij}\}$ is a s-state Markov Process and define $\{Y_t\}$ as a probabilistic function of $\{X_t\}$ (Baum & Petrie, 1966). The conditional probability of Y_t given all the past values of $\{X_t\}$ and $\{Y_t\}$ up to time t is defined as b_{jk}. In their first paper, they proved the consistency of the likelihood of the $\{Y_t\}$ and the smoothness property of the expectation of the log-likelihood of $\{Y_t\}$ w.r.t to the parameter $\boldsymbol{\theta}$ from $\{a_{ij}\}$ and $\{b_{ij}\}$ (Baum & Petrie, 1966). In their later paper in the 1970s, they further introduced an iterative method for estimating the model parameters and proved that the algorithm converges to a local maximum (Baum & Petrie, 1966). This method later became known as the Baum-Welch algorithm, which is the EM Algorithm in HMMs (Zucchini et al., 2016). The parameters $\{a_{ij}\}$ and $\{b_{ij}\}$ are now commonly referred to as the transition distribution and emission distribution, respectively (Zucchini et al., 2016). The book "Hidden Markov Models for Time Series, An Introduction Using R" written by the Zucchini et al. (Zucchini et al., 2016) is one of the most classical textbook in HMMs, and this paper will heavily rely on this book.

One of the popular applications of HMMs in the Time Series Analysis is GARCH-HMMs or Regime-Switching GARCH (Cai, 1994). It captures the volatility process changes, which may depend on some unobserved regimes, and we assume the changes of hidden states following a Markov process. In this hybrid model, the HMMs divide the time series with different volatility levels and then use the GARCH model to model the time-varying variance within each regime (Zhuang & Chan, 2004). For example, in financial time series, this allows the model to shift between high and low volatility periods, such as market booms and crashes. We can, therefore, predict the state the next time and choose the corresponding local GARCH model to predict the volatility. This model is advantageous as it allows us to see how volatility changes between different levels and how GARCH works for each level.

*All codes and analyses can be found at: https://github.com/yiliuc/Hidden_Markov_Models

Another significant development in HMMs is Automatic Speech Recognition (ASR). It is based on HMMs and became well-known by IBM and Bell Lab in the 1980s (Bahl et al., 1983). It has been considered an important tool to enhance the performance of human-human and human-machine communications. A famous one is Apple's Siri, which can be acted as a virtual assistant. The basic algorithm for HMMs in ASR is to set hidden states to be the phonetic characters and use the properties of HMMs to identify the true sentence that the speaker is saying. It can even be used to predict what the speaker is trying to say, given what he has already said. Moreover, one of the most recent studies of ASR uses Deep learning methods such as Deep Neural Networks to construct multiple hidden layers to recognize speech, which are the Deep Neural Networks Markov Models for ASR (Yu & Deng, 2015).

This paper is structured as follows. Section 2 will briefly review the Markov Chain, along with some properties. Then, Section 3 will introduce the Hidden Markov Model, where we will discuss its definitions and properties. In addition, Section 4 will talk about the Baum-Welch Algorithm, which is the application of the EM algorithm in HMMs. The forecasting and decoind will be discussed at Section 5. After that, we will apply these algorithm to two real-world scenarios and compare their performance in Section 6. We will also provide discussions in Section 7.

2 Markov Chains

2.1 Basics

A Markov Chain MC models the structure of dependence for a sequence of variables. Let $\{X_t : t \in \mathbb{N}\}$ be a sequence of discrete random variables, where each X_t takes value in a finite state space $\mathbf{S} = \{1, \ldots, m\}$ with $|\mathbf{S}| = m$. The sequence $\{X_t\}$ is said to form a **Markov Chain (MC)** if, for all $t \in \mathbb{N}$, it satisfies the Markov Property such that

$$P(X_{t+1}|X_t, \ldots, X_1) = P(X_{t+1}|X_t)$$

The Markov Property says that the probability of a random variable in a MC only depends on the most recent one. With this property, the Markov Chain can be visualized as

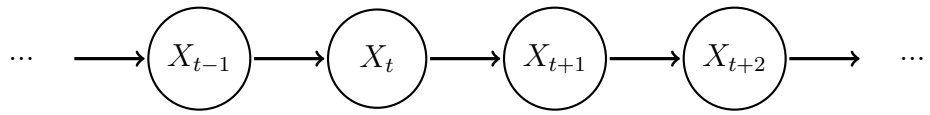

Figure 1: First Order Markov Chain

Since each variable is discrete, the probability of transitioning from state i to state j in one time step is called the **transition probability**, denoted by

$$\gamma_{ij} = p(X_{t+1} = j|X_t = i)$$

If γ_{ij} does not change across time, $p(X_{t+1} = j|X_t = i) = p(X_{t+2} = j|X_{t+1} = i)\ \forall t \in \mathbb{N}$, then such markov chains are said to be **time-homogeneous**. The matrix $\mathbf{\Gamma}(1) = \mathbf{\Gamma}$ is defined as the transition matrix with $(i, j)^{th}$ element as γ_{ij}. More generally, the transition probability of moving from state i to j over t time steps is denoted by $\gamma_{ij}(t) = P(X_{s+t} = j \mid X_s = i)$, with corresponding t step transition matrix $\mathbf{\Gamma}(t)$, which satisfies $\mathbf{\Gamma}(t + u) = \mathbf{\Gamma}(t)\mathbf{\Gamma}(u)$. This is so-called the **Champman-Kolmogorov equations**.

2.2 Stationary Distribution

In Markov Chains, the unconditional probabilities $u_j(t) = p(X_t = j)$ is the probability of X_t being in the state j at time t. The probabilities can be written as a row vector $\mathbf{u}(t) = (p(X_t = 1), \ldots, p(X_t = m)) = (u_1(t), \ldots, u_m(t)) \in \mathbb{R}^m$, which can be shown that $\mathbf{u}(t+1) = \mathbf{u}(t)\mathbf{\Gamma}$. Moreover, if we let the Markov Chain runs for a long time, then we would expect that, starting at some point, the Markov Chain will reach out to a steady state. This means that the subsequent time steps have the same distribution even after we apply the transition probabilities, and we call the distribution at the steady state as the **stationary distribution** of Markov Chain, denoted by $\boldsymbol{\delta}$. Mathematically, $\boldsymbol{\delta}$ can be computed by

$$\boldsymbol{\delta} = \boldsymbol{\delta}\mathbf{\Gamma}$$

That is, $\boldsymbol{\delta}$ is the eigenvector of transition matrix $\mathbf{\Gamma}$ with eigenvalue 1. This equation does not imply the uniqueness of stationary distribution. However, if a markov chain is irreducible (i.e., it is homogeneous, discrete-time, finite state space), then such markov chains has a unique, strictly positive stationary distribution.

Moreover, a Markov Chain is said to be **stationary** if its initial distribution is its stationary distribution and remains unchanged to all the subsequent time points, i.e., $\boldsymbol{\delta} = \mathbf{u}(1) = \cdots = \mathbf{u}(T)$. This is a stronger condition than time-homogeneity and is typically difficult to satisfy in practice, as it requires exact knowledge of the transition matrix and control over the initial distribution. On the other hand, the markov chain observed in real-world scenarios is usually already in progress, and we do not have the ability to set its initial distribution by $\boldsymbol{\delta}$. Furthermore, it is important to distinguish between time-homogeneous and stationary Markov Chain. The time-homogeneous Markov Chain only states that the conditional distribution of hidden variables is time invariant. The time-homogeneous Markov Chain can only be stationary if its initial distribution $\mathbf{u}(1)$ equals to the stationary distribution.

3 Hidden Markov Models

Hidden Markov Models HMMs is a special class of stochastic models that extends the Markov Chain by introducing a latent (hidden) process. In addition to the observed variables $\{X_t : t \in \mathbb{N}\}$, there is a sequence of hidden variables $\{Z_t : t \in \mathbb{N}\}$ that is modelled by a Markov Chain. Let $\mathbf{Z}^{(t)} = (Z_1, \ldots, Z_t)$ and $\mathbf{X}^{(t)} = (X_1, \ldots, X_t)$ denotes the histories for hidden and observed variables up to time t (Zucchini et al., 2016, p. 30). A basic Hidden Markov Model satisfies the following properties:

$$p\left(Z_t \mid \mathbf{Z}^{(t-1)}\right) = p\left(Z_t \mid Z_{t-1}\right), \quad t \in \{2, 3 \ldots\}$$
$$p\left(X_t \mid \mathbf{X}^{(t-1)}, \mathbf{Z}^{(t)}\right) = \Pr\left(X_t \mid Z_t\right), \quad t \in \mathbb{N}$$

These assumptions imply that each hidden state depends only on the previous hidden state (first-order Markov property), and each observed variable depends only on the hidden state at the same time t. A graphical representation of basic Hidden Markov Model is shown on Figure 2, where clearly shows the dependence structure.

All the properties of Markov Chain introduced on the Section 2 are now applied to the hidden variables $\{Z_t : t \in \mathbb{N}\}$ rather than the observed variables. The observed process $\{X_t : t \in \mathbb{N}\}$ is referred as an m-state Hidden Markov Model if $|\mathbf{S}| = m$. In addition, since each observed variable X_t only depends on the corresponding hidden variable Z_t, the conditional distribution $p(X_t = x | Z_t = i)$

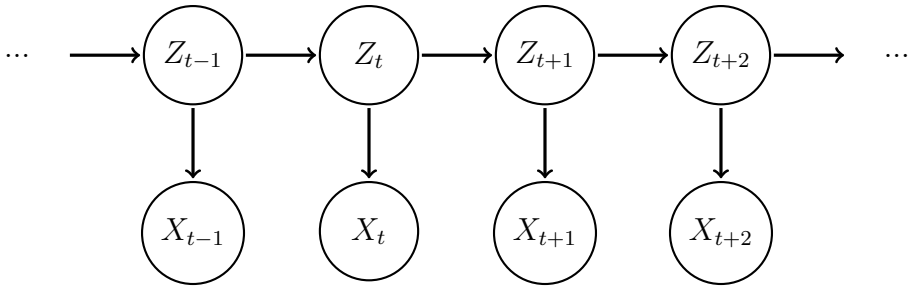

Figure 2: Basic Hidden Markov Model

is called the **Emission distribution** (or **state-dependent distribution**), and is denoted by $p_i(x) = p(X_t = x|Z_t = i)$. Intuitively, the emission distribution describes the distribution of the observed variable "emitted" by the hidden state. Each hidden state $i \in \{1, \ldots, m\}$ is associated with a distinct emission distribution $p_i(x)$, and these distributions are normally time-invariant.

In Hidden Markov Models, it is important to estimate the marginal distribution of the observed variables and the method marginalization over all hidden variable is preferred. Consider a binvariate setting ofX_t, X_{t+1}, its joint distribution of $p(X_t = a, X_{t+1} = b)$ can be expressed as

$$p(X_t = a, X_{t+1} = b) = \sum_i^m \sum_j^m p(X_t = a, X_{t+1} = b, Z_t = i, Z_{t+1} = j)$$

Using the dependence structure of HMMs, Z_{t+1} only depends on Z_t as we can marginalize the intermediate nodes out. Then the joint distribution of the four variables can be expressed as $p(Z_t = i)p(X_t = a|Z_t = i)p(Z_{t+1} = j|Z_t = i)p(X_{t+1} = b|Z_{t+1} = j)$. Therefore, the joint distribution of X_t and X_{t+1} can be expressed as

$$\begin{aligned} p(X_t = a, X_{t+1} = b) &= \sum_i^m \sum_j^m u_i(t) p_i(a) \gamma_{ij} p_j(b) \\ &= \mathbf{u}(t)\mathbf{P}(a)\mathbf{\Gamma}\mathbf{P}(b)\mathbf{1}' \end{aligned}$$

where $\mathbf{P}(x) = \text{diag}(p_1(x), p_2(x), \ldots, p_m(x))$ is the diagonal matrix of emission probabilities, $\mathbf{1}'$ is a column vector of ones. It can be further reduced to $\boldsymbol{\delta}\mathbf{P}(a)\mathbf{\Gamma}\mathbf{P}(b)\mathbf{1}'$ if the Markov Chain is stationary.

Using the idea on the bivariate case, it can be generalized to the joint likelihood of a full observation sequence $\{X_t : t \in \{1, \ldots, T\}\}$. Given the initial distribution $\boldsymbol{\delta}$ and emission probability $p_i(x)$, the likelihood function of observed variables is given by (Zucchini et al., 2016, pp. 32–33)

$$L_T = \boldsymbol{\delta}\mathbf{P}(x_1)\mathbf{\Gamma}\mathbf{P}(x_2)\mathbf{\Gamma}\mathbf{P}(x_3)\cdots\mathbf{\Gamma}\mathbf{P}(x_T)\mathbf{1}'$$

If the Markov Chain is stationary, we can also write the $\boldsymbol{\delta}$ as $\boldsymbol{\delta}\mathbf{\Gamma}$. However, as previously discussed, this substitution requires the exact knowledge of transition matrix. Therefore, in this context, the $\boldsymbol{\delta}$ here is interpreted as the distribution of the first hidden variable Z_1, and we will retain this interpretation in the remainder of this paper.

4 EM Algorithm in HMMs

The Expectation-Maximization (EM) algorithm is an iterative procedure for computing maximum likelihood estimates when some values are missing or unobserved. It is particularly advantageous

when the likelihood function is complex or intractable to maximize directly. The EM Algorithm is divided into Expectation (E) and Maximization (M) step. In E step, it computes the expected complete-data log-likelihood given the observed or incomplete data and the current estimate of the parameters. It then maximizes this expected log-likelihood derived in E step to update the parameter estimates, which is so-called the M step. These two steps are repeated until convergence, where it yields parameter estimates that are stationary point (typically a local maximum) of the observed-data likelihood (Bickel & Doksum, 2015).

The EM Algorithm are naturally applied in Hidden Markov Models as the hidden state sequence are unobserved and thus treated as missing data. The observed data are referred as the incomplete data. By applying EM in this context, we iteratively estimate the model parameters $\boldsymbol{\theta}$, which is so-called the **Baum-Welch Theorem** (Baum et al., 1970). In this section, we will discuss and prove the Expectation and Maximization steps in details.

4.1 Forward-Backward Algorithm

Before diving into EM algorithm, it is necessary to first discuss the Forward-Backward algorithm, as it plays a key role in the Expectation (E) step. The Forward-Backward algorithm is a fundamental procedure for computing the posterior distribution of the hidden states at each time point, given the entire sequence of observed data. Specifically, it evaluates $p\left(Z_t = z_t \mid \mathbf{X}^T = \mathbf{x}^T\right)$, which is crucial in Hidden Markov Models, as the hidden states are not directly observable. The key thing in Forward-Backward Algorithm are the **forward probabilities** and **backward probabilities**, denoted by $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$, respectively (Zucchini et al., 2016, pp. 65–68). Formally, they are defined as

$$\boldsymbol{\alpha}_t = \boldsymbol{\delta}\mathbf{P}\left(x_1\right)\boldsymbol{\Gamma}\mathbf{P}\left(x_2\right)\cdots\boldsymbol{\Gamma}\mathbf{P}\left(x_t\right) = \boldsymbol{\delta}\mathbf{P}\left(x_1\right)\prod_{s=2}^{t}\boldsymbol{\Gamma}\mathbf{P}\left(x_s\right)$$

$$\boldsymbol{\beta}_t' = \boldsymbol{\Gamma}\mathbf{P}\left(x_{t+1}\right)\boldsymbol{\Gamma}\mathbf{P}\left(x_{t+2}\right)\cdots\boldsymbol{\Gamma}\mathbf{P}\left(x_T\right)\mathbf{1}' = \left(\prod_{s=t+1}^{T}\boldsymbol{\Gamma}\mathbf{P}\left(x_s\right)\right)\mathbf{1}'$$

where the parameters are defined in the same way as previous. The terms forward and backward refer to the direction in which each quantity propagates information. The forward probabilities accumulate evidence from the beginning of the sequence, $t = 1$ up to current time t. In contrast, the backward probabilities aggregate information from the future ($t = T$) backward to the present. In the following, we will see how these quantities are computed recursively, and demonstrate that the posterior probability of the hidden state at time t is proportional to the product of the forward and backward probabilities.

Both $\alpha(i)$ and $\beta(i)$ are derived naturally from the factorization of the likelihood function L_T. The forward probabilities can be recursively expressed as $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t\boldsymbol{\Gamma}\mathbf{P}\left(x_{t+1}\right)$. Equivalently, each component of $\boldsymbol{\alpha}_{t+1}$ can be expressed as $\alpha_{t+1}(j) = \left(\sum_{i=1}^{m}\alpha_t(i)\gamma_{ij}\right)p_j\left(x_{t+1}\right)$. At $t = 1$, $\alpha_1(j) = \delta_j p_j\left(x_1\right) = p\left(Z_1 = j\right)p\left(X_1 = x_1 \mid Z_1 = j\right)$. Using induction (see Appendix 8.1), it can be further showed that

$$\alpha_t(j) = \Pr\left(\mathbf{X}^{(t)} = \mathbf{x}^{(t)}, Z_t = j\right)\ \forall t \in \{1,\ldots,T\},\ j \in \{1,\ldots,m\}$$

Similarly, the backward probabilities can be expressed as

$$\beta_t(i) = p(\mathbf{X}_{t+1}^T = \mathbf{x}_{t+1}^T | Z_t = i)$$

where $\mathbf{X}_a^b = (X_a, X_{a+1}, \ldots, X_b)$ (Zucchini et al., 2016, pp. pp67–68). Given these definitions, the joint probability of the full observed sequence and the hidden state being in state j at time t can be

expressed as the product of the two (Appendix 8.2), that is

$$\Pr\left(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, Z_t = j\right) = \alpha_t(j)\beta_t(j)$$

Using the results established above, we can now compute the posterior distribution of the hidden state at any time point $t \in \{1, \ldots, T\}$. The probability that the hidden state at time t is equal to j, given the entire observed sequence, is given by:

$$\Pr\left(Z_t = j \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \alpha_t(j)\beta_t(j)/L_T$$

Furthermore, the joint posterior probability of two consecutive hidden states $Z_{t-1} = i$ and $Z_t = j$ for $t \in \{2, \ldots, T\}$ is given by (see Appendix 8.3)

$$\Pr\left(Z_{t-1} = j, Z_t = k \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \alpha_{t-1}(j)\gamma_{jk}p_k\left(x_t\right)\beta_t(k)/L_T$$

These results provides us an effective way to estimate the hidden state for any time t conditional the entire observed data by combining the forward and backward probabilities. This can help us to find "beliefs" of the hidden state. As we will discuss in the next section, the Forward-Backward algorithm is intimately connected to the E-step of the EM algorithm. It is also important in local decoding strategies as well.

4.2 Expectation Step

In the Expectation-Maximization (EM) algorithm for Hidden Markov Models, the E-step involves computing the expectation of the complete-data log-likelihood given the incomplete data and the current parameter estimates (Bickel & Doksum, 2015). In the context of HMMs, the complete data combines both the observed data $\{x_t\}_{t=1}^T$ and the corresponding hidden states $\{z_t\}_{t=1}^T$. Since the hidden states are not directly observed, it naturally becomes the missing part. In this section, we will firstly express the complete-data likelihood, and then evaluate its expectation conditional on the observed data sequence.

Let the parameters be $\boldsymbol{\theta} = (\boldsymbol{\delta}, \boldsymbol{\Gamma}, \boldsymbol{\lambda})$, where $\boldsymbol{\delta}$ is the initial distribution, $\boldsymbol{\Gamma}$ is the transition matrix with $(i,j)^{th}$ element γ_{ij}, and $\boldsymbol{\lambda}$ be the emission distribution specific parameters. Let the complete data be $\mathcal{D}_{complete} = \left\{\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, \mathbf{Z}^{(T)} = \mathbf{z}^{(T)}\right\}$. Using the standard factorization of joint distribution of HMMs in Section 3, the log-likelihood function over $\mathcal{D}_{complete}$ can be written as

$$\begin{aligned}
\ell\left(\boldsymbol{\theta} \mid \mathcal{D}_{complete}\right) &= \log\left(p\left(z_1\right)\prod_{t=2}^{T} p\left(z_t \mid z_{t-1}\right)\prod_{t=1}^{T} p\left(x_t \mid z_t\right)\right) \\
&= \log\delta_{z_1} + \sum_{t=2}^{T}\log\gamma_{z_{t-1},z_t} + \sum_{t=1}^{T}\log p_{z_t}\left(x_t\right)
\end{aligned}$$

However, the log-likelihood expression above is written in terms of specific hidden states z_t, and does not explicitly leading to the parameters $\boldsymbol{\theta}$. To solve this, we introduce two new indicator variables $u_j(t) = \mathbf{1}\{Z_t = j\}$, $t \in \{1, \ldots, T\}$ and $v_{jk}(t) = \mathbf{1}\{Z_{t-1} = j, Z_t = k\}$, $t \in \{2, \ldots, T\}$ (Zucchini et al., 2016, pp. 70–71). By plugging in the indicators, the log-likelihood function becomes

$$\ell\left(\boldsymbol{\theta}|\mathcal{D}_{complete}\right) = \sum_{j=1}^{m} u_j(1)\log\delta_j + \sum_{j=1}^{m}\sum_{k=1}^{m}\left(\sum_{t=2}^{T} v_{jk}(t)\right)\log\gamma_{jk} + \sum_{j=1}^{m}\sum_{t=1}^{T} u_j(t)\log p_j\left(x_t\right)$$

The only random terms in the log-likelihood function are only the two indicator variables which depends only on the hidden states. The rest terms on the log-likelihood are all parameters which is not random. Therefore, in the E-step, we only need to compute the expectation of $u_j(1), v_{jk}(t), u_j(t)$ conditioning on the observed data $\mathbf{X}^{(T)} = \mathbf{x}^{(T)}$ and the current $\hat{\boldsymbol{\theta}}$ (Zucchini et al., 2016, pp. 71–72). Using the Forward-Backward Algorithm, it can be shown that the conditional expectations are given by

$$\begin{aligned}\mathbb{E}[u_j(t)|\mathbf{x}^{(T)}, \hat{\boldsymbol{\theta}}] &= p\left(Z_t = j \mid \mathbf{x}^{(T)}, \hat{\boldsymbol{\theta}}\right) \\ &= \hat{\alpha}_t(j)\hat{\beta}_t(j)/\hat{L}_T =: \hat{u}_j(t)\end{aligned}$$

and for $t \geq 2$

$$\begin{aligned}\mathbb{E}[v_{jk}(t)|\mathbf{x}^{(T)}, \hat{\boldsymbol{\theta}}] &= p\left(Z_{t-1} = j, Z_t = k \mid \mathbf{x}^{(T)}, \hat{\boldsymbol{\theta}}\right) \\ &= \hat{\alpha}_{t-1}(j)\hat{\gamma}_{jk}\hat{p}_k\left(x_t\right)\hat{\beta}_t(k)/\hat{L}_T =: \hat{v}_{jk}(t)\end{aligned}$$

where the estimated forward and backward probabilities are computed recursively as:

$$\hat{\alpha}_{t+1}(j) = \sum_{i=1}^{m} \hat{\alpha}_t(i)\hat{\gamma}_{ij}\hat{p}_j\left(x_{t+1}\right) \text{ s.t. } \hat{\alpha}_1(j) = \hat{\delta}_j\hat{p}_j\left(x_1\right)$$

$$\hat{\beta}_t(i) = \sum_{j=1}^{m} \hat{\gamma}_{ij}\hat{p}_j\left(x_{t+1}\right)\hat{\beta}_{t+1}(j) \text{ s.t. } \hat{\beta}_T(j) = 1$$

And the estimated likelihood is given by

$$\hat{L}_T = \hat{\boldsymbol{\alpha}}_T\hat{\boldsymbol{\beta}}'_T$$

Putting all together, the expected log-likelihood function is written as

$$\mathbf{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_{j=1}^{m} \hat{u}_j(1)\log\delta_j + \sum_{j=1}^{m}\sum_{k=1}^{m}\left(\sum_{t=2}^{T}\hat{v}_{jk}(t)\right)\log\gamma_{jk} + \sum_{j=1}^{m}\sum_{t=1}^{T}\hat{u}_j(t)\log p_j\left(x_t\right)$$

4.3 Maximization Step

In M step, we maximize the expected log-likelihood $\mathbf{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ wrt to $\boldsymbol{\theta}$ to update the estimates

$$\hat{\boldsymbol{\theta}}^{new} = \arg\max_{\theta} \mathbf{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$$

Conveniently, the subjective function separates to three terms with each depending on a component of $\boldsymbol{\theta}$: the initial distributions δ_j, transition matrix γ_{ij} and emission parameters $p_j(x_t)$. This allows us to maximize each set of independent parameters. Taking the derivatives for each time, we can obtain the closed-form updates for $\boldsymbol{\delta}$ and $\boldsymbol{\Gamma}$ (see Appendix 8.4) as

$$\delta_j = \frac{\widehat{u}_j(1)}{\sum_{j=1}^{m}\widehat{u}_j(1)} = \widehat{u}_j(1)$$

$$\gamma_{jk} = f_{jk}/\sum_{k=1}^{m} f_{jk}, \text{ where } f_{jk} = \sum_{t=2}^{T}\widehat{v}_{jk}(t)$$

However, the estimates for the emission parameters $\boldsymbol{\lambda}$ depends on the specific form of the emission model and we can not provide a general formula here.

We will run EM Algorithm recursively until convergence, though the rate and quality of convergence can vary depending on factors such as the choice of initial parameter estimates, the characteristics of the data etc. Similar to many other convergence algorithm the EM algorithm does not guarantee convergence to the global maximum of the likelihood function; rather, it may converge to a local maximum, especially in models with complex likelihood such as Hidden Markov Models (Bickel & Doksum, 2015; Zucchini et al., 2016, pp. 72–73). Another important thing is that we must state whether we assume the markov chain is stationary. This crucial because, recall in Section 3, we can replace the $\boldsymbol{\delta}$ to $\boldsymbol{\delta\Gamma}$ if the Markov Chain is stationary. This assumption has a direct impact on the parameter estimation process and must therefore be explicitly stated prior to performing the EM algorithm. Lastly, in this paper, we will use the R package `HiddenMarkov` (Harte, 2025) to run the EM instead of manually writing the iterative function. More practical applications of implementing EM Algorithm in HMMs will be presented in Section 6.

5 Forecast and Decoding

With the parameter estimates from the Baum-Welch Algorithm, we can use them to forecast and decoding the Hidden Markov Models, which can help us to understand the data better. In HMMs, the term "forecasting" usually infers to predict the future values of the observed variables X_t. In contrast, "decoding" means we use the observed values to compute the most likely hidden states (Murphy, 2023). In this section, we are going to discuss how to predict the observed variable in the future. More importantly, we are going to discuss two kinds of decoding process which are local and global decoding. The two decoding methods are similar but differed on the number of hidden states we estimate every time. We will discuss more details in the rest part of this section.

5.1 Forecasting

The forecasting in HMMs means we want to predict the values of the observed variables given the information we have. It describes the conditional distribution of the X_{T+h} given all the observed values we have up to time T, which are expressed as

$$\begin{aligned} p(X_{T+h} = x | \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) &= \frac{p(X_{T+h} = x, \mathbf{X}^{(T)} = \mathbf{x}^{(T)})}{p(\mathbf{X}^{(T)} = \mathbf{x}^{(T)})} \\ &= \frac{\boldsymbol{\alpha}_T \boldsymbol{\Gamma}^h \mathbf{P}(x) \mathbf{1}'}{\boldsymbol{\alpha}_T \mathbf{1}'} \end{aligned}$$

Moreover, let $\boldsymbol{\phi}_T = \frac{\boldsymbol{\alpha}_T}{\boldsymbol{\alpha}_T \mathbf{1}'}$, then it can also be written as $\phi_T \boldsymbol{\Gamma}^h \mathbf{P}(x) \mathbf{1}'$. It can be also written in terms of a mixture of the emission distribution $p\left(X_{T+h} = x \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \sum_{i=1}^{m} \xi_i(h) p_i(x)$, where $\xi_i(h)$ is ith entry of the vector $\phi_T \boldsymbol{\Gamma}^h$.

One important implication of the forecasted conditional distribution is that, as h increases, the forecast distribution converges to the marginal distribution of the stationary HMM (Zucchini et al., 2016, p. 85), that is

$$\lim_{h \to \infty} \Pr\left(X_{T+h} = x \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \lim_{h \to \infty} \boldsymbol{\phi}_T \boldsymbol{\Gamma}^h \mathbf{P}(x) \mathbf{1}' = \boldsymbol{\delta}^* \mathbf{P}(x) \mathbf{1}'$$

where $\boldsymbol{\delta}^*$ is the stationary distribution of the Markov Chain. It shows that for any non-negative row vectors with row sum 1 ($\boldsymbol{\delta}^*$ here) approaches the stationary distribution of the Markov Chain. However, this property can be only applied to irreducible and aperiodic Markov Chain (Zucchini et

al., 2016, p. 85). We call $\boldsymbol{\delta}^*\mathbf{P}(x)\mathbf{1}'$ as the limiting distribution of the Markov Chain. The speed of converging to the **limiting distribution** may various. We will discuss more about this in the application part.

5.2 Decoding

There are two types of decoding: local and global decoding. Their goals are both to estimate the hidden states given the observed variables but local decoding only estimated once a time but global decoding aims to estimate the sequence of hidden states simultaneously. The local decoding is fully based on the Forward-Backward Algorithm we discussed previously, which is simply the conditional distribution of hidden state at time t given all the observed variables. We will discuss them in more details in this section.

5.2.1 Local Decoding

The algorithm for local decoding is simple. Recall that the joint distribution $p\left(Z_t = i, \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right)$ can be expressed as the product of forward and backward probabilities, $\alpha_t(i)\beta_t(i)$ (Zucchini et al., 2016, pp. 87–88). Then the conditional distribution of being on state i at time t is

$$p\left(Z_t = i \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \frac{p\left(Z_t = i, \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right)}{p\left(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right)}$$

We have proved that $p\left(Z_t = i, \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \alpha_t(i)\beta_t(i)$ from the Forward-Backward Algorithm, then the joint distribution can be written as

$$p\left(Z_t = i \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \frac{\alpha_t(i)\beta_t(i)}{L_T}$$

Then for each $t \in \{1, \ldots, T\}$, we can compute the most probable hidden state $Z_t = i^*$ by

$$i_t^* = \underset{i=1,\ldots,m}{\operatorname{argmax}} p\left(Z_t = i \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right)$$

By this approach, we can find the most likely hidden state at any time $t \leq T$ once a time, and this approach is called the local decoding as we are estimating locally and not realted to other hidden states.

5.2.2 Global Decoding

In contrast to local decoding, global decoding allows us to compute a sequence of hidden states together instead of a single hidden state. It is noticeble that global decoding is not equivalent to run the local decoding for multiple times. The difference between the two is that running local decoding multiple times may give us unrealistic sequence such as impossible transition. However, we want to estimate the hidden states together to keep the joint distribution. This is motivation behind the global decoding, as we prefer global decoding as it will give us a probable and reasonable sequence of hidden states, while still keep the realistic transition probabilities.

In global decoding, our goal is to to maximize

$$\begin{aligned} p(\mathbf{Z}^{(t)} = \mathbf{z}^{(t)} | \mathbf{X}^{(t)} = \mathbf{x}^{(t)}) &\propto p(\mathbf{Z}^{(t)} = \mathbf{z}^{(t)}, \mathbf{X}^{(t)} = \mathbf{x}^{(t)}) \\ &= \delta_{z_1} \prod_{t=2}^{T} \gamma_{z_{t-1}, z_t} \prod_{t=1}^{T} p_{z_t}(x_t) \end{aligned}$$

The complete global decoding works as follows

1. We firstly define $\xi_{1,i} = p(Z_1 = i, X_1 = x_1) = \delta_i p_i(x_1)$, meaning that we are at state i at time 1.
2. For $t \in \{2, 3, \ldots, T\}$, we define

$$\xi_{ti} = \max_{z_1, z_2, \ldots, z_{t-1}} p\left(\mathbf{Z}^{(t-1)} = \mathbf{z}^{(t-1)}, Z_t = i, \mathbf{X}^{(t)} = \mathbf{x}^{(t)}\right)$$

   ξ_{ti} is the probability of the best sequence of $Z_1, \ldots, Z_t$ such that this sequence has to be ended on $Z_t = i$. It is computed by considering each possible path of $z_1, \ldots, z_t = i$ and return the highest probability. This equation is equivalent to

$$\xi_{tj} = \left(\max_i \left(\xi_{t-1,i}\gamma_{ij}\right)\right) p_j\left(x_t\right)$$

   By this step, it can give us a $T \times m$ matrix with each element storing the best score at in each time and state.
3. Given $t = T$, we have

$$i_T = \underset{i=1,\ldots,m}{\operatorname{argmax}}\, \xi_{Ti}$$

   This step helps us to find the most likely state at the final time T. It can be done simply by search the last row of the matrix we had and state corresponding to the highest score will be the choice of the value of the last hidden state.
4. For $t \in \{T-1, T-2, \ldots, 1\}$, find i_t such that

$$i_t = \underset{i=1,\ldots,m}{\operatorname{argmax}} \left(\xi_{ti}\gamma_{i,i_{t+1}}\right)$$

   We are moving backwards from the ending point and recursively find the path corresponding to the highest score. It is similar to step 3, and eventually we will have the most probable path of hidden states.

This algorithm is so-called the Viterbi Algorithm (Zucchini et al., 2016, p. 90).

5.3 State Prediction

Finally, we are also concerned about how to predict the value of hidden states in the future time $t > T$, $p\left(Z_t = i \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right)$. Since $\boldsymbol{\alpha}_T = p(Z_T, \mathbf{X}^{(T)} = \mathbf{x}^{(T)})$, then the joint distribution can be expressed as $p\left(Z_{T+1}, \mathbf{X}^{(T)}\right) = p\left(Z_{T+1} = k \mid Z_T = j\right) p\left(Z_T = j \mid \mathbf{X}^{(T)}\right) = \boldsymbol{\alpha}_T \boldsymbol{\Gamma}$. Similarly, we also have $p\left(Z_{T+h}, \mathbf{X}^{(T)}\right) = \boldsymbol{\alpha}_T \boldsymbol{\Gamma}^h$. To have $p\left(Z_{T+h} = i, \mathbf{X}^{(T)}\right)$, it is equivalent to the i-th component of $\boldsymbol{\alpha}_T \boldsymbol{\Gamma}^h$, which is equivalent to $\boldsymbol{\alpha}_T \boldsymbol{\Gamma}^h \mathbf{e}_i'$, where $\mathbf{e}_i$ is a row vector that only the ith element is 1 and all other are zeros (Zucchini et al., 2016, pp. 92–93). Therefore the conditional distribution can be written as

$$p\left(Z_t = i \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \frac{p\left(Z_t = i, \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right)}{\boldsymbol{\alpha}_T \mathbf{1}'} = \frac{\boldsymbol{\alpha}_T \boldsymbol{\Gamma}^h \mathbf{e}_i'}{\boldsymbol{\alpha}_T \mathbf{1}'} = \phi_T \boldsymbol{\Gamma}^h \mathbf{e}_i'$$

To sum up, recall that in the Forward-Backward Algorithm, we have derived that $p(Z_t = z_t | \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) = \alpha_t(z_t)\beta_t(z_t)$, the forecasting of hidden states at any time t can be summarized by

$$L_T \Pr(Z_t = i \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) = \begin{cases} \boldsymbol{\alpha}_T \boldsymbol{\Gamma}^{t-T} \mathbf{e}_i & \text{for } t > T \\ \alpha_T(i) & \text{for } t = T \\ \alpha_t(i)\beta_t(i) & \text{for } 1 \le t < T \end{cases}$$

6 Application

In this section, we present two real-world examples to evaluate the practical performance of Hidden Markov Models (HMMs): the Toronto bicycle thefts and NVIDIA stock prices. These two examples will show how to implement different types of emission distribution such as Poisson for Bicycle Thefts and Normal for volatility of stock, and compare their performances. When perform EM algorithm to estimate the parameters, although it is possible to implement the recursive algorithms for parameter estimation manually in R, we make use of existing R packages such as `HiddenMarkov` (Harte, 2025) to effectively perform the EM Algorithms in Hidden Markov Models. For all R functions, please see Appendix 8.5.

6.1 Toronto Bicycle Thefts

The data on Toronto bicycle thefts is obtained from the `opendatatoronto` package (Gelfand, 2022). It contains the records of bicycle thefts from January 2014 to December 2024. We aggregate this data to obtain the monthly counts of bicycle thefts over this eleven-year period, resulting in a total of 132 observations. The goal of this study is to see whether we can see the seasonal patterns using the HMMs. We will assume that the number of bicycle thefts per month follows a Poisson distribution with varying rate parameters. To model this time series, we fit Hidden Markov Models (HMMs) with different numbers of hidden states and select the optimal model based on the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).

Table 1: Poisson HMM Model Comparison (Non-Stationary)

States	AIC	BIC	LogLikelihood	Parameters	Iterations
2	4196.517	4210.931	-2093.258	5	7
3	2685.328	2717.039	-1331.664	11	9
4	2172.677	2227.450	-1067.339	19	15
5	1993.804	2077.405	-967.902	29	36
6	1753.180	1871.375	-835.590	41	30

Table 1 compares different HMMs under different criterias. We can see that even though the values of AIC and BIC continuously to decrease as the number of states increases, the number parameters and the number of iterations needed to reach convergence increases significantly. In addition, it seems that the decrease of adding one more hidden state after 4 is not as much as from 3 to 4 and the number of parameters and number of iterations under 4 hidden states are moderatre. Therefore we may conclude that the 4 hidden states are adequate for the HMMs. After fitting four-state poisson-HMM using EM Algorithm, the estimated initial distribution is $\hat{\boldsymbol{\delta}} = (1, 0, 0, 0)$, meaning that the chain starts at state 1. In addition, the estimated transition matrix

$$\begin{pmatrix} 0.76 & 0.218 & 0.022 & 0.000 \\ 0.407 & 0.148 & 0.446 & 0.000 \\ 0.028 & 0.308 & 0.411 & 0.252 \\ 0.000 & 0.000 & 0.362 & 0.638 \end{pmatrix}$$

This indicates that, given the system is currently in State 1, there is a 0.76 probability of remaining in the same state at time $t + 1$. Additionally, there is a 0.218 probability of transitioning to State 2.

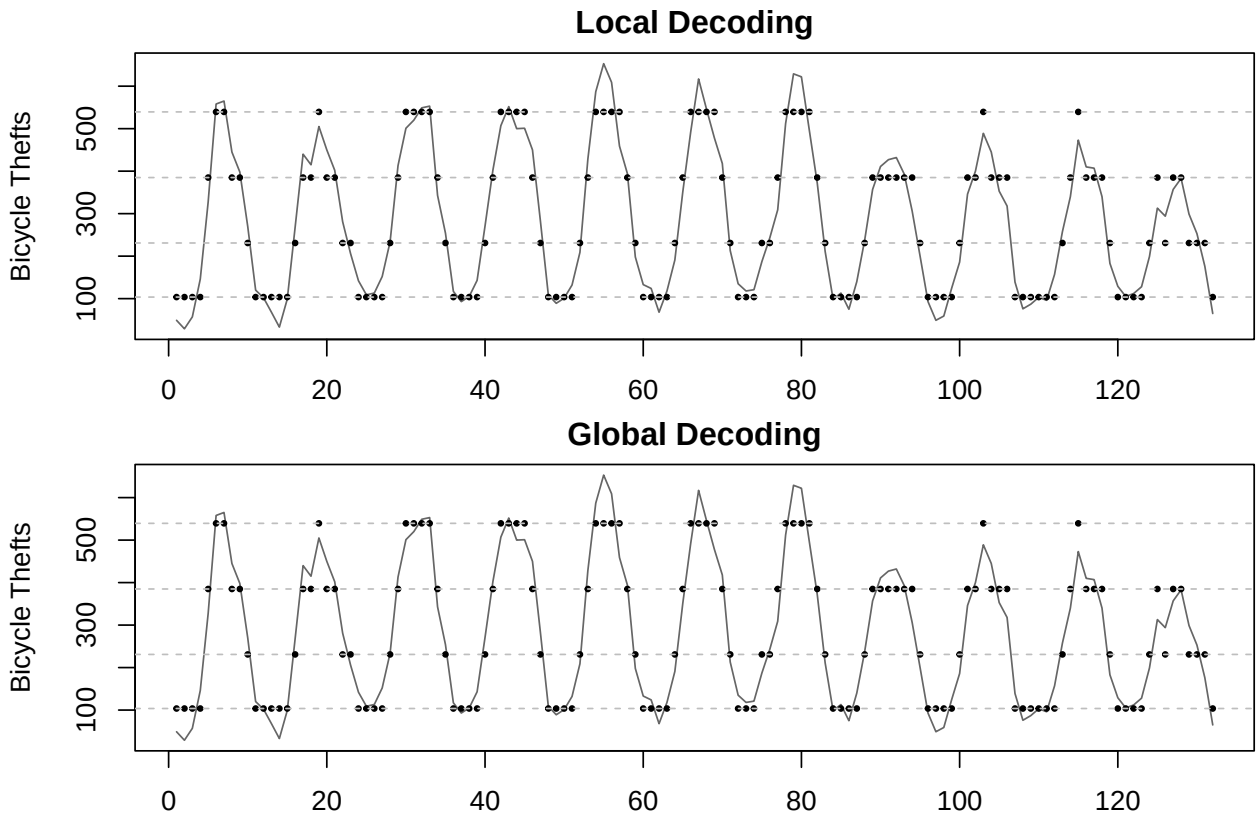

Figure 3: *Local decoding (top) and glocal decoding (bottom) for the number of monthly bicycle thefts at Toronto since 2014 January (4 hidden states)*

The remaining transition probabilities can be interpreted in a similar way. Moreover, the estimated emission distribution parameters are $(104, 231, 385, 549)$ with rounded to the nearest integer.

After estimating all model parameters, we apply the local and global decoding methods introduced in Section 5 to identify the hidden states over time. Figure 3 presents the local (top) and global (bottom) decoding results. The three dashed lines represent the estimated values of lambdas for each hidden states, while the black dots indicate the decoded state at each time point. Both decoding methods perform exceptionally well, which we can observe a clear separation between hidden states that closely aligns with the observed time series. Moreover, the decoded states appear to capture meaningful seasonal trends. For example, lower rates of bicycle thefts are observed during the winter months as people may not ride bicycles. This can be seen in the first few observations (January to April 2014), which are mostly assigned to the lowest-rate state. In contrast, higher theft rates are observed during the summer months (e.g., time 5 to 10), which also makes sense as people may increase bicycle usage during warmer weather. Furthermore, the estimated hidden state sequences from local and global decoding are exactly identical. This suggests that the posterior probabilities at each time point are highly concentrated on a single state, possibly due to a strong seasonal pattern in the data. The agreement between the two decoding approaches implies that the most probable path and the most probable individual states coincide.

In addition to local and global decoding, we can also forecast the distribution of future observations. Figure 4 presents the forecast densities for the number of bicycle thefts over the next nine months. In each panel, the blue-shaded area represents the forecast density, while the black line denotes the density of the limiting (stationary) distribution. Each forecast exhibits a clear tri-modal structure, reflecting the underlying three hidden states in the fitted model. This suggests that the number of bicycle thefts at each future time point is drawn from a mixture of three Poisson distributions, each corresponding to a different hidden state. The presence of multiple modes in the forecast densities indicates that the system can evolve into several plausible future regimes, each associated with a distinct distribution of X_t. Nevertheless, the dominant mode near 100 has the highest density, suggesting that a monthly count around 100 is the most likely outcome. This is reasonable as Toronto experiences long winters, leading to fewer thefts during much of the year.

Table 2: *5 Steps Ahead Forecast Summary with State Probabilities for Toronto Bicycle Thefts*

Day	Forecast	State Probabilities
1	137.49 (84.84, 264.07)	S1: 0.76, S2: 0.22, S3: 0.02, S4: 0.00
2	164.57 (85.37, 400.87)	S1: 0.67, S2: 0.20, S3: 0.12, S4: 0.01
3	178.33 (85.77, 405.43)	S1: 0.59, S2: 0.21, S3: 0.16, S4: 0.03
4	188.62 (86.06, 408.45)	S1: 0.54, S2: 0.21, S3: 0.19, S4: 0.06
5	196.46 (86.28, 410.52)	S1: 0.50, S2: 0.21, S3: 0.20, S4: 0.09
6	202.80 (86.47, 412.10)	S1: 0.47, S2: 0.20, S3: 0.22, S4: 0.11

Lastly, Table 2 presents the five-step-ahead forecasts for the number of bicycle thefts along with the predicted probabilities of the hidden states. Since the available data ends in December 2024, we expect the number of bicycle thefts to gradually increase in the subsequent months, which we can observe this pattern from the forecasted values. Moreover, as previously discussed, the four hidden states likely correspond to the four seasons. Therefore, it is reasonable to see the decrease in the probability of State 1 (winter). The concurrent increase in the probabilities of the other states align

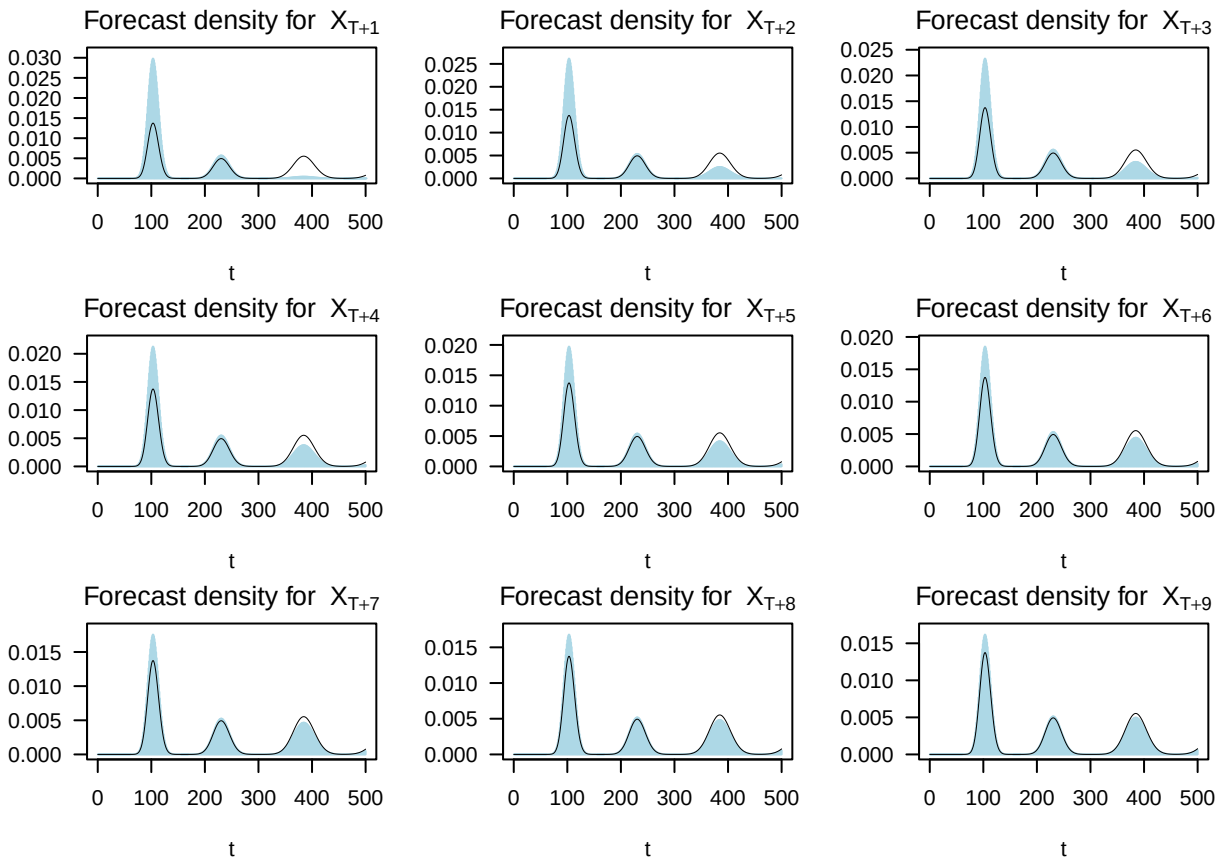

Figure 4: *Forecasted distribution of monthly bicycle theft counts in Toronto for the next nine months*

with seasonal transitions and provide further support for this interpretation. Overall, the HMMs in Toronto bicycle thefts perform excellently.

6.2 NVIDIA stock price

The previous analysis of bicycle thefts demonstrated that Hidden Markov Models (HMMs) perform well when the emission distribution is Poisson. To further evaluate the effectiveness of HMMs, we now apply them to model NVIDIA stock prices. The data are obtained using the R package `quantmod` (Ryan & Ulrich, 2023) to access the NVIDIA stock price from January 2nd, 2024 to April 11st, 2025. The objective of this analysis is to use an HMM to estimate the latent volatility states underlying the observed price movements and we will quantify the volatility using the absolute log returns of the stock price. We will assume the emission distribution is normal; that is, the distribution of absolute log return given the volatility state follows normal distribution. As usual, we begin by fitting HMMs with different numbers of hidden states and selecting the best model based on information criteria such as AIC and BIC.

Table 3: *Gaussian HMM Model Comparison (Non-Stationary)*

States	AIC	BIC	LogLikelihood	Parameters	Iterations
2	-1583.110	-1556.797	798.555	7	34
3	-1616.910	-1564.285	822.455	14	247
4	-1622.727	-1536.272	834.364	23	220

Table 3 compares HMMs with hidden states two to four. We can see that the AIC and BIC do not decreases significantly as the number of hidden states increases. However, as hidden states increases, the number of parameters needs to be estimated and number of iterations untill convergence increases dramatically. Therefore, it seems that the two-state Gaussian-HMM is adequate for this data. By performing the EM Algorithm with two states, the estimated initial distribution $\hat{\boldsymbol{\delta}} = (1, 0)$, indicating the Markov Chain starts at state 1. The estimated transition matrix is

$$\hat{\mathbf{\Gamma}} = \begin{pmatrix} 0.84 & 0.16 \\ 0.738 & 0.262 \end{pmatrix}$$

The estimated transition matrix shows that, given we are at the state 1, there is a 0.84 probability of remaining at state 1 at $t + 1$ and 0.16 probable to move to the second state.In contrast, given we are at state 2, the probability of stay at the same state is only 0.262, while there are 0.738 probability to move to state 1. In addition, the estimated parameters of the emission distribution are $\mu_1 = 0.019, \sigma_1 = 0.013$ and $\mu_2 = 0.059, \sigma_2 = 0.033$, respectively.

Using the estimated parameters, we perform both local and global decoding to estimate the hidden states. Figure 5 presents the estimated hidden states over time for the absolute log returns of NVIDIA stock prices, with local decoding results shown at the top and global decoding at the bottom. As in the previous example, the two dashed lines represent the estimated mean volatilities for each hidden state, while the black dots indicate the estimated hidden state at each time point. Both decoding approaches yield reasonable results, which they all successfully identifying periods of high volatility corresponding to sharp spikes in the data. Unlike the bicycle theft example, however, the state sequences derived from local and global decoding has slight differences. This discrepancy reflects the ambiguity in state assignments for continuous data with overlapping distributions, where local and global approaches may yield different solutions.

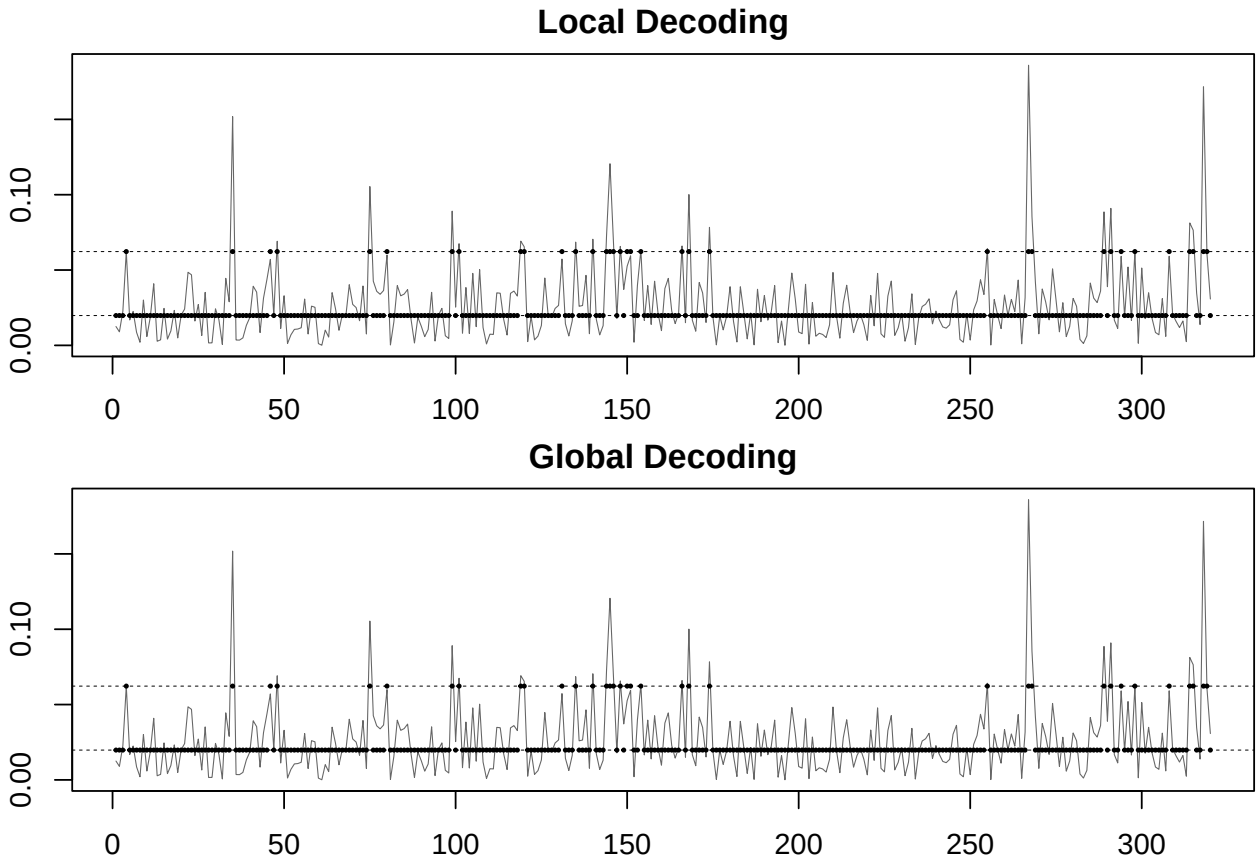

Figure 5: *Local decoding (top) and glocal decoding (bottom) for the volatility of daily NVIDIA stock (the most recent 150 observations due to visualbility) (2 hidden states)*

Figure 6 presents the forecast densities for the future absolute log returns over the next nine time points. As before, the blue-shaded regions represent the predictive densities, while the black curves denote the stationary distribution. In contrast to the bicycle theft example where convergence to the limiting distribution occurred gradually, the forecast densities for the absolute log returns match the stationary distribution almost immediately, with only minor deviations at X_{t+1}. One potential reason for this rapid convergence is the structure of the estimated transition matrix, where each row is dominated by a single large probability. This structure makes transitions between states infrequent, particularly from State 1 to State 2, and keeping the process in the same state for longer durations. This behavior is also reflected in Figure 5, where the estimated high-volatility state appears much less frequently than the low-volatility state.

Another possible reason is that the Markov chain may have already reached its stationary distribution by the end of the observed data. This would result in the hidden state predictions for future time points remaining nearly constant. We can verify this conclusion again from Table 4, which displays the predicted state probabilities over the next five days and they are nearly identical. This suggests the markov chain is already reached the steady state. In fact, we can compute the stationary distribution by calculating the eigenvector of the estimated transition matrix, which is $\boldsymbol{\delta}^{\star} = (0.823, 0.177)$. This result closely alignes with Table 4.

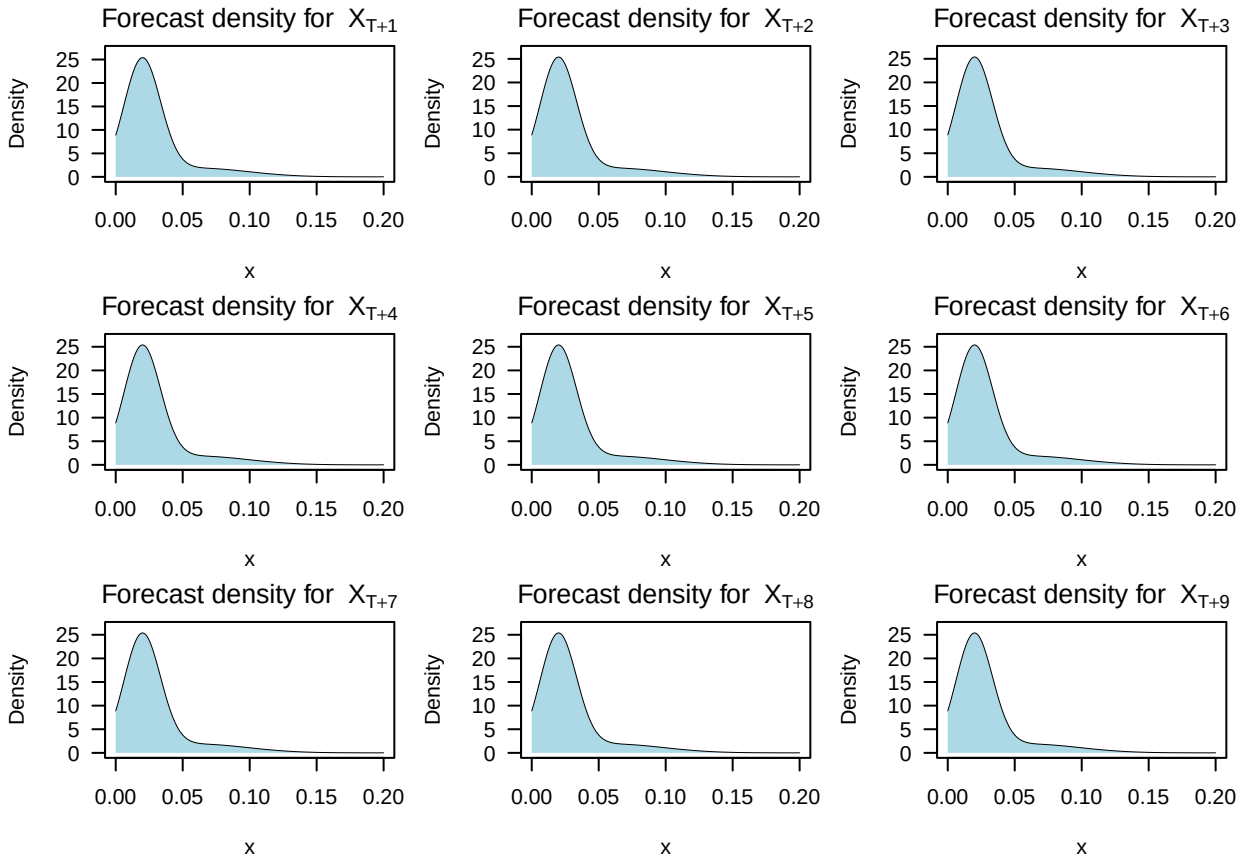

Figure 6: *Forecasted distribution of daily volatility (by absolute log returns) of NVIDIA stock for the next nine days*

Table 4: *5 Steps Ahead Forecast Summary with State Probabilities for Volatility of NVIDIA Stock Price*

Day	Forecast	State Probabilities
1	0.03 (0.00, 0.10)	S1: 0.84, S2: 0.16
2	0.03 (0.00, 0.10)	S1: 0.83, S2: 0.17
3	0.03 (0.00, 0.10)	S1: 0.83, S2: 0.17
4	0.03 (0.00, 0.10)	S1: 0.83, S2: 0.17
5	0.03 (0.00, 0.10)	S1: 0.83, S2: 0.17
6	0.03 (0.00, 0.10)	S1: 0.83, S2: 0.17

7 Discussion

The hidden Markov Model is a very powerful stochastic model and has wide applications in fields such as Time Series, Finance, speech recognition, and bioinformatics. It shows us a unique dependence structure between the observed and hidden variables by the two Markov properties. This paper summarizes the basic concepts of Hidden Markov Models such as the Forward-Backward Algorithm, Baum-Welch Algorithm. In addition, this paper further applies HMMs to two practical examples to demonstrate its performance.

Overall, HMMs have excellent performance in predicting the hidden states behind the observed data, especially for the example of Toronto Bicycle Thefts, where the predicted hidden states correspond to four seasons in a year. In addition, its predictions on the observed and hidden states also reasonable when combined with common sense. For example, the predictions of the number of bicycle thefts in the next six months, shown in Table 2, tell us the number of thefts may increase in the summertime. This makes sense as people are more likely to ride bicycles during the summer. However, even though the predictions in either observed variable or hidden state on NVIDIA stock volatility on future days are less informative, HMMs successfully classify the high and low volatility levels from the previous day, as shown in Figure 5, where each peak has a prediction of high volatility level. Its bad performance on prediction is related to the data structure, where the stock stays at low volatility most of the time. This example also does not imply the low efficiency of HMMs in continuous emission distribution.

However, it is true that the HMMs still have limitations. The first one is that, just like other convergence problems, the parameter estimated from the EM Algorithm does not guarantee to reach the global maximum (Bickel & Doksum, 2015). This means that the final estimates may depend heavily on the choice of initial parameter values and may converge to local optima, particularly in models with complex structures or multiple hidden states. Besides, the choice of the initial parameter values may also affect the convergence speed. Just like what we presented on Table 1 and 3, a practical way to solve this problem is to run the HMMs multiple times with different initial values and compare the criteria like AIC and BIC to choose the optimal one. On the other hand, the number of parameters will also increase dramatically as the number of proposed hidden states increases. For example, the number of parameters of transition probabilities increases $2m - 1$ every time we increase one hidden state. This will also make the estimation of parameters more complicated and has a low convergence rate.

Another limitation is the assumption of time homogeneity, which states that the transition probabilities between hidden states remain constant over time. While this assumption simplifies the

modeling process, it may not hold in dynamic real-world contexts. For example, transitions between volatility regimes in financial markets may evolve over time due to external influences such as economic policy, political events, or macroeconomic shocks, especially on these days that President Trump impose tariffs. In such cases, more flexible models such as Hidden Semi-Markov Models (HSMMs), which explicitly model the duration that the system stays in each hidden state using a user-specified or learned duration distribution (Zucchini et al., 2016). Alternatively, models that incorporate GARCH dynamics into the HMM framework, referred to as HMM-GARCH, can also be applied to financial time series to better capture changing volatility patterns (Cai, 1994; Zhuang & Chan, 2004).

In conclusion, Hidden Markov Models are powerful tools for time series analysis, which provide us valuable insights into the underlying structure of sequential data through the estimation and prediction of latent states. While HMMs perform well in many applications, their practical use in real-world settings may require more advanced or hybrid models.

8 Appendix

8.1 Forward probabilities $\alpha_t(j)$

WTS: $\alpha_t(j) = \Pr\left(\mathbf{X}^{(t)} = \mathbf{x}^{(t)}, Z_t = j\right)$

proof: Based on the factorization of $\boldsymbol{\alpha}_t$, we know that

$$\alpha_1(j) = \delta_j p_j\left(x_1\right) = \Pr\left(Z_1 = j\right) \Pr\left(X_1 = x_1 \mid Z_1 = j\right)$$

Prove by induction. Assume that $\alpha_1(j) = \Pr\left(X_1 = x_1, Z_1 = j\right)$ is true at time t. At $t+1$, we have

$$\begin{aligned}
\alpha_{t+1}(j) &= \sum_{i=1}^{m} \alpha_t(i)\gamma_{ij} p_j\left(x_{t+1}\right) \\
&= \sum_i \Pr\left(\mathbf{X}^{(t)} = \mathbf{x}^{(t)}, Z_t = i\right) \Pr\left(Z_{t+1} = j \mid Z_t = i\right) \times \Pr\left(X_{t+1} = x_{t+1} \mid Z_{t+1} = j\right) \\
&= \sum_i \Pr\left(\mathbf{X}^{(t+1)} = \mathbf{x}^{(t+1)}, Z_t = i, Z_{t+1} = j\right) \quad \text{by structure of the model} \\
&= \Pr\left(\mathbf{X}^{(t+1)} = \mathbf{x}^{(t+1)}, Z_{t+1} = j\right),
\end{aligned}$$

■

8.2 Product of Forward and Backward probabilities

We can write $\alpha_t(i)\beta_t(i)$ as

$$\begin{aligned}
\alpha_t(i)\beta_t(i) &= \Pr\left(\mathbf{X}_1^t, Z_t = i\right) \Pr\left(\mathbf{X}_{t+1}^T \mid Z_t = i\right) \\
&= \Pr\left(Z_t = i\right) \Pr\left(\mathbf{X}_1^t \mid Z_t = i\right) \Pr\left(\mathbf{X}_{t+1}^T \mid Z_t = i\right)
\end{aligned}$$

since $\mathbf{X}_1^t$ and $\mathbf{X}_{t+1}^T$ are conditionally independent given Z_t, therefore

$$\alpha_t(i)\beta_t(i) = \Pr\left(Z_t = i\right) \Pr\left(\mathbf{X}_1^t, \mathbf{X}_{t+1}^T \mid Z_t = i\right) = \Pr\left(\mathbf{X}^{(T)}, Z_t = i\right)$$

■

8.3 Conditional distribution of two hidden variable

We firstly have

$$\Pr\left(Z_{t-1} = j, Z_t = k \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) = \Pr\left(\mathbf{X}^{(T)}, Z_{t-1} = j, Z_t = k\right) / L_T$$

By the dependence structure of HMM, we know that it can be further expressed as

$$\Pr\left(\mathbf{X}^{(t-1)}, Z_{t-1} = j\right) \Pr\left(Z_t = k \mid Z_{t-1} = j\right) \Pr\left(\mathbf{X}_t^T \mid Z_t = k\right) / L_T$$

Hence, the conditional distribution can be written as

$$\begin{aligned}
\Pr\left(Z_{t-1} = j, Z_t = k \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}\right) &= \alpha_{t-1}(j)\gamma_{jk}\left(\Pr\left(X_t \mid Z_t = k\right) \Pr\left(\mathbf{X}_{t+1}^T \mid Z_t = k\right)\right) / L_T \\
&= \alpha_{t-1}(j)\gamma_{jk} p_k\left(x_t\right) \beta_t(k) / L_T
\end{aligned}$$

■

8.4 Maximization Step in EM Algorithm

For δ_j:

Objective Function:

$$\sum_{j=1}^{m} \hat{u}_j(1) \log \delta_j$$

such that $\sum_{j=1}^{m} \delta_j = 1$

We introduce the Lagrange multiplier λ to enforce the constraint:

$$\mathcal{L} = \sum_{j=1}^{m} \hat{u}_j(1) \log \delta_j + \lambda \left(1 - \sum_{j=1}^{m} \delta_j\right)$$

Take Derivative and Set to Zero:

$$\frac{\partial \mathcal{L}}{\partial \delta_j} = \frac{\hat{u}_j(1)}{\delta_j} - \lambda = 0 \quad \Rightarrow \quad \hat{u}_j(1) = \lambda \delta_j$$

Sum over j to find λ:

$$\sum_{j=1}^{m} \hat{u}_j(1) = \lambda \sum_{j=1}^{m} \delta_j \quad \Rightarrow \quad \lambda = \sum_{j=1}^{m} \hat{u}_j(1)$$

Since $\sum_{j=1}^{m} \hat{u}_j(1) = 1$, we get $\lambda = 1$. Thus:

$$\delta_j = \hat{u}_j(1)$$

For γ_{jk}:

Objective Function:

$$\sum_{j=1}^{m} \sum_{k=1}^{m} \left(\sum_{t=2}^{T} \hat{v}_{jk}(t)\right) \log \gamma_{jk}$$

such that for each j, $\sum_{k=1}^{m} \gamma_{jk} = 1$

We use the Lagrange multiplier λ again to enforce the constraint:

$$\mathcal{L}_j = \sum_{k=1}^{m} f_{jk} \log \gamma_{jk} + \lambda_j \left(1 - \sum_{k=1}^{m} \gamma_{jk}\right)$$

where $f_{jk} = \sum_{t=2}^{T} \hat{v}_{jk}(t)$

Take Derivative and Set to Zero:

$$\frac{\partial \mathcal{L}_j}{\partial \gamma_{jk}} = \frac{f_{jk}}{\gamma_{jk}} - \lambda_j = 0 \quad \Rightarrow \quad \gamma_{jk} = \frac{f_{jk}}{\lambda_j}$$

Sum over k to find λ_j:

$$\sum_{k=1}^{m} f_{jk} = \lambda_j \sum_{k=1}^{m} \gamma_{jk} \quad \Rightarrow \quad \lambda_j = \sum_{k=1}^{m} f_{jk}$$

$$\Rightarrow \gamma_{jk} = \frac{f_{jk}}{\sum_{k=1}^{m} f_{jk}}$$

■

8.5 R codes

Below are the codes for the HMM for poisson and gaussian emission distribution. If you want to see how I implement it with the real-world examples, please check my GitHub page (here). I hide the codes in the text as I want to make this as an academic paper.

8.5.1 Codes for poisson emission distribution

The codes for poisson emission distribution is based on the book from Zucchini et al. (Zucchini et al., 2016).

Transfer the natural parameter to working parameter.

```
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)
}
```

Transfer working parameter back to natural parameter.

```
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))
}
```

Compute the negative log-likelihood

```
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)
}
```

Direct MLE for Poisson

```
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, 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
  )
}
```

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)
}
```

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
    lscale <- lscale + log(sumfoo)
  }

  return(lbeta)
}
```

Estimate the 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)
}
```

Hidden states predictions

```
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)
}
```

Local decoding and global decoding (Viterbi Algorithm)

```
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])
  }
```

```
  ild
}

pois.HMM.viterbi <- function(x, mod) {
  n <- length(x)
  xi <- matrix(0, n, mod$m)

  # Initialization
  foo <- mod$delta * dpois(x[1], mod$lambda)
  xi[1, ] <- foo / sum(foo)

  # Forward recursion
  for (i in 2:n) {
    foo <- apply(xi[i - 1, ] * mod$gamma, 2, max) * dpois(x[i], mod$lambda)
    xi[i, ] <- foo / sum(foo)
  }

  # Backtracking
  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)
}
```

Forecasting of observed variables

```
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)
}
```

8.5.2 Codes for poisson emission distribution

Below are the R codes for gaussian emission distribution. These are NOT provided by Zucchini et al., and I wrote them on my own. They follow the same order to the poisson, and hence I will write them together.

```
##############################################################################
# Gaussian HMM functions
##############################################################################
normal.HMM.pn2pw <- function(m, mu, sigma, gamma, delta = NULL, stationary = TRUE) {
  tmu <- mu                    # Means are unconstrained
  tsigma <- log(sigma)         # Log-transform standard deviations

  # Transition matrix transformation (same as Poisson HMM)
  foo <- log(gamma / diag(gamma))
  tgamma <- as.vector(foo[!diag(m)])

  if (stationary) {
    tdelta <- NULL
  } else {
    tdelta <- log(delta[-1] / delta[1])
  }

  parvect <- c(tmu, tsigma, tgamma, tdelta)
  return(parvect)
}

normal.HMM.pw2pn <- function(m, parvect, stationary = TRUE) {
  mu <- parvect[1:m]
  sigma <- exp(parvect[(m + 1):(2 * m)])

  gamma <- diag(m)
  gamma[!gamma] <- exp(parvect[(2 * m + 1):(2 * m + m * (m - 1))])
  gamma <- gamma / rowSums(gamma)

  if (stationary) {
    delta <- solve(t(diag(m) - gamma + 1), rep(1, m))
  } else if (stationary == FALSE) {
```

```r
    foo <- c(1, exp(parvect[(2 * m + m * (m - 1) + 1):(2 * m + m * (m - 1) + m - 1)]))
    delta <- foo / sum(foo)
  }

  return(list(mu = mu, sigma = sigma, gamma = gamma, delta = delta))
}

normal.HMM.mllk <- function(parvect, x, m, stationary = TRUE, ...) {
  if (m == 1) return(-sum(dnorm(x, mean = parvect[1], sd = exp(parvect[2]), log = TRUE)))

  n <- length(x)
  pn <- normal.HMM.pw2pn(m, parvect, stationary = stationary)

  # Initial probabilities
  alpha <- pn$delta * dnorm(x[1], mean = pn$mu, sd = pn$sigma)
  sum_alpha <- sum(alpha)
  lscale <- log(sum_alpha)
  alpha <- alpha / sum_alpha # Initial probability

  for (i in 2:n) {
    if (!is.na(x[i])) {
      P <- dnorm(x[i], mean = pn$mu, sd = pn$sigma)
    } else {
      P <- rep(1, m)
    }
    if (any(!is.finite(P))) {
      cat("Invalid P at t =", i, "; x[i] =", x[i], "\n")}

    alpha <- alpha %*% pn$gamma * P
    sum_alpha <- sum(alpha)

    lscale <- lscale + log(sum_alpha)
    alpha <- alpha / sum_alpha
  }

  neg_log_likelihood <- -lscale
  return(neg_log_likelihood)
}

normal.HMM.mle <- function(x, m, mu0, sigma0, gamma0, delta0 = NULL, stationary = TRUE, ...) {
  parvect0 <- normal.HMM.pn2pw(m, mu0, sigma0, gamma0, delta0, stationary = stationary)
  mod <- nlm(normal.HMM.mllk, parvect0, x = x, m = m, stationary = stationary)
  pn <- normal.HMM.pw2pn(m = m, parvect = mod$estimate, stationary = stationary)

  # Step 4: model diagnostics
  mllk <- mod$minimum
  np <- length(parvect0)
```

```r
  n <- sum(!is.na(x))

  AIC <- 2 * (mllk + np)
  BIC <- 2 * mllk + np * log(n)

  # Step 5: return results
  return(list(
    m = m,
    mu = pn$mu,
    sigma = pn$sigma,
    gamma = pn$gamma,
    delta = pn$delta,
    code = mod$code,
    mllk = mllk,
    AIC = AIC,
    BIC = BIC
  ))
}

normal.HMM.lforward <- function(x, mod) {
  n <- length(x)
  lalpha <- matrix(NA, mod$m, n)

  # Initial step
  foo <- mod$delta * dnorm(x[1], mean = mod$mean, sd = mod$sd)
  sumfoo <- sum(foo)
  lscale <- log(sumfoo)
  foo <- foo / sumfoo
  lalpha[, 1] <- lscale + log(foo)

  # Forward loop
  for (i in 2:n) {
    foo <- foo %*% mod$gamma * dnorm(x[i], mean = mod$mean, sd = mod$sd)
    sumfoo <- sum(foo)
    lscale <- lscale + log(sumfoo)
    foo <- foo / sumfoo
    lalpha[, i] <- log(foo) + lscale
  }

  return(lalpha)
}

normal.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$gamma %*% (dnorm(x[i + 1], mean = mod$mean, sd = mod$sd) * foo)
    lbeta[, i] <- log(foo) + lscale
    sumfoo <- sum(foo)
    foo <- foo / sumfoo
    lscale <- lscale + log(sumfoo)
  }

  return(lbeta)
}

normal.HMM.state_probs <- function(x, mod) {
  n <- length(x)
  la <- normal.HMM.lforward(x, mod)
  lb <- normal.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)
}

# Note that state output 'statepreds' is a matrix even if h=1.
normal.HMM.state_prediction <- function(h = 1, x, mod) {
  n <- length(x)
  la <- normal.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)
}

normal.HMM.local_decoding <- function(x, mod) {
```

```r
  n <- length(x)
  stateprobs <- normal.HMM.state_probs(x, mod)
  ild <- rep(NA, n)
  for (i in 1:n) {
    ild[i] <- which.max(stateprobs[, i])
  }
  ild
}

normal.HMM.viterbi <- function(x, mod) {
  n <- length(x)
  xi <- matrix(0, n, mod$m)

  # Initialization (t = 1)
  foo <- mod$delta * dnorm(x[1], mean = mod$mean, sd = mod$sd)
  xi[1, ] <- foo / sum(foo)

  # Forward recursion (t = 2 to n)
  for (i in 2:n) {
    foo <- apply(xi[i - 1, ] * mod$gamma, 2, max) * dnorm(x[i], mean = mod$mean, sd = mod$sd)
    xi[i, ] <- foo / sum(foo)
  }

  # Backtracking
  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)
}

normal.HMM.forecast <- function(xf, h = 1, x, mod) {
  n    <- length(x)
  nxf  <- length(xf)
  dxf  <- matrix(0, nrow = h, ncol = nxf)

  # Initial state probabilities after first observation
  foo <- mod$delta * dnorm(x[1], mean = mod$mean, sd = mod$sd)
  sumfoo <- sum(foo)
  lscale <- log(sumfoo)
  foo <- foo / sumfoo
  for (i in 2:n) {
    foo <- foo %*% mod$gamma * dnorm(x[i], mean = mod$mean, sd = mod$sd)
    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] * dnorm(xf, mean = mod$mean[j], sd = mod$sd[j])
    }
  }
  return(dxf)
}
```

References

Bahl, L. R., Jelinek, F., & Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-5*(2), 179–190. https://doi.org/10.1109/TPAMI.1983.4767370

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, *37*(6), 1554–1563. https://doi.org/10.1214/aoms/1177699147

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, *41*(1), 164–171. https://doi.org/10.1214/aoms/1177697196

Bickel, P. J., & Doksum, K. A. (2015). *Mathematical statistics: Basic ideas and selected topics, volumes i-II package* (1st ed.). Chapman; Hall/CRC. https://doi.org/10.1201/9781315369266

Cai, J. (1994). A markov model of switching-regime ARCH. *Journal of Business & Economic Statistics*, *12*(3), 309–316. https://doi.org/10.2307/1392087

Gelfand, S. (2022). *Opendatatoronto: Access the city of toronto open data portal*. https://CRAN.R-project.org/package=opendatatoronto

Harte, D. S. (2025). *HiddenMarkov: Hidden markov models*. https://www.statsresearch.co.nz/dsh/sslib/

Murphy, K. P. (2023). *Probabilistic machine learning: Advanced topics*. MIT Press. http://probml.github.io/book2

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286. https://doi.org/10.1109/5.18626

Ryan, J. A., & Ulrich, J. M. (2023). *Quantmod: Quantitative financial modelling framework*. https://CRAN.R-project.org/package=quantmod

Yu, D., & Deng, L. (2015). *Automatic speech recognition: A deep learning approach* (1st ed., pp. XXVI, 321). Springer London. https://doi.org/10.1007/978-1-4471-5779-3

Zhuang, X. F., & Chan, L. W. (2004). Volatility forecasts in financial time series with HMM-GARCH models. In Z. R. Yang, H. Yin, & R. M. Everson (Eds.), *Intelligent data engineering and automated learning – IDEAL 2004* (Vol. 3177, pp. 995–1001). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-28651-6_120

Zucchini, W., MacDonald, I. L., & Langrock, R. (2016). *Hidden markov models for time series: An introduction using r* (2nd ed.). Chapman; Hall/CRC. https://doi.org/10.1201/b20790