Hidden Markov Chain Models

Group 11
Chen Taoyue 1155141543
DAI Qiyu 1155141616
JIANG Yunhui 1155141677
LI Linyuan 1155141569
MAK Man Fung 1155141893
QIN Zihao 1155124920

Contents

1 Introduction

Markov chain is widely used to describe a sequence of stochastic events satisfying the Markov property. However, for its variant called hidden Markov model (HMM), such a process of interest is unobservable. Instead, we only have the data whose distribution is determined by the corresponding hidden state [1]. In this report, we aim to investigate methodologies which retrieve information about HMMs by working on the observed data.

The rest of the report is organized as follows. In Section 2, we formulate two HMMs and discuss how the EM and Viterbi algorithms operate for parameters and hidden states estimation. Section 3 reports Monte Carlo simulation results. Lastly, our model is applied to fit three datasets online.

2 Theory

Notation. For the hidden Markov chain, we write the sequence as $\{\mathbf{C}_t : t = 1, , T\}$ and the history up to time t as $\mathbf{C}^{(t-1)}$; For the observable variables, the sequence and history are denoted as $\{\mathbf{X}_t : t = 1, , T\}$ and $\mathbf{X}^{(t-1)}$, respectively. Let m be the number of hidden states. Moreover, an HMM is uniquely characterized by three basic metrics, namely, transition kernel $\mathbf{\Gamma}$, state-dependent distributions $\mathbf{P}(x)$, and initial state occupation distribution δ. Θ denotes the parameters set for $\mathbf{P}(x)$. Notice that we will consider the state-dependent distributions to have Gaussian and Poisson structures in the following.

Several additional entities will be used in the further calculation. Define that $u_j(t) = \mathbb{1}(c_t = j)$ and $v_{jk}(t) = \mathbb{1}(c_{t-1} = j \text{ and } c_t = k)$, where $\mathbb{1}$ is an indicator function. The forward and backward probabilities are $\alpha_{\mathbf{t}} = \delta\mathbf{P}(x_1)\mathbf{\Gamma}\mathbf{P}(x_2)\cdots\mathbf{\Gamma}\mathbf{P}(x_t)$ and $\beta_t = \mathbf{\Gamma}\mathbf{P}(x_{t+1})\mathbf{\Gamma}\mathbf{P}(x_{t+2})\cdots\mathbf{\Gamma}\mathbf{P}(x_T)1'$, respectively. Finally, we use I_m to denote an m-dimensional identity matrix, and $\mathbf{1}$ stand for a vector of ones.

2.1 EM algorithm

We first employ the EM algorithm to infer the unknown parameters Θ. EM algorithm is widely adopted to handle mixture distribution in the presence of missing latent variables, which corresponds to the hidden state in HMMs.

The complete likelihood function $\log(P_1(X^{(T)}, C^{(T)})|\Theta)$ and observed likelihood function $\log(P_2(X^{(T)}, C^{(T)})|\Theta)$

are

$$\log(P_1(X^{(T)}, C^{(T)})|\Theta) = \sum_{j=1}^{m} u_j(1)\log\delta_j + \sum_{j=1}^{m}\sum_{k=1}^{m}(\sum_{t=2}^{T} v_{jk}(t))\log\gamma_{jk} + \underbrace{\sum_{j=1}^{m}\sum_{t=1}^{T} u_j(t)\log p_j(x_t)}_{f(\Theta)} \tag{1}$$

$$\log(P_2(X^{(T)})|\Theta) = \log(\alpha_{\mathbf{t}}\beta_{\mathbf{t}}) \tag{2}$$

To get prepared for carrying out the EM algorithm, we first derive the likelihood maximizer for (1) given all other quantities. The Gaussian probability density function is $p_j(x) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}(\frac{x-\mu_j}{\sigma_j})^2}$. After plugging it into the third term of (1) and equating the first order derivatives to 0, we have

$$\begin{aligned} f(\Theta) &= \sum_{j=1}^{m}\sum_{t=1}^{T} u_j(t)[-\frac{1}{2}\log(\sigma_j^2) - \frac{1}{2}\log(2\pi) - \frac{1}{2}\frac{(x_t-\mu_j)^2}{\sigma_j^2}] \\ \text{let } \frac{\partial f(\Theta)}{\partial\sigma_j^2} &= \sum_{t=1}^{T}[-\frac{u_j(t)}{2\sigma_j^2} + \frac{(x_i-\mu_j)^2}{2\sigma_j^4}] = 0, \text{ and } \frac{\partial f(\Theta)}{\partial\mu_j} = \sum_{t=1}^{T} u_j(t)\frac{1}{\sigma_j^2}(x_t-\mu_j) = 0 \\ \hat{\sigma}_j^2 &= \frac{\sum_{t=1}^{T}(x_t-\hat{\mu}_j)^2 u_j(t)}{\sum_{t=1}^{T} u_j(t)}, \hat{\mu}_j = \frac{\sum_{t=1}^{T} x_t u_j(t)}{\sum_{t=1}^{T} u_j(t)} \end{aligned} \tag{3}$$

Similarly, for Poisson,

$$\begin{aligned} f(\Theta) &= \sum_{j=1}^{m}\sum_{t=1}^{T} u_j(t)[-x_t\log(\lambda_j) - \lambda_j - \log(x_t!)] \\ \text{let } \frac{\partial f(\Theta)}{\partial\lambda_j} &= \sum_{t=1}^{T} u_j(t)[-\frac{x_t}{\lambda_j} - \lambda_j] = 0 \\ \hat{\lambda}_j(t) &= \frac{\sum_{t=1}^{T} u_j(t)x_t}{\sum_{t=1}^{T} u_j(t)} \end{aligned} \tag{4}$$

Apart from Equation (3) and (4), the EM algorithm has already been explained in detail in the lecture notes. Yet, we provide a summary below to make the implementation more straightforward.

Algorithm 1: EM algorithm

Input: $\{X_t : t = 1, \cdots, T\}$, m, distribution type of $\mathbf{P}(x)$, N, ϵ

Initialize $\hat{\Theta}^{(0)}$ $\hat{\mathbf{\Gamma}}_{(\mathbf{0})}$, $\hat{\delta}^{(\mathbf{0})}$

for $n = 1$ *to* N **do**

  Forward Porbability Normalization:

  $\alpha_1 = \hat{\delta}^{(n-1)}\hat{\mathbf{P}}^{(n-1)}(x_1)$, $r_1 = \sum_{j=1}^{m} \alpha_1$, $\alpha_1 = \alpha_1 / \sum_{j=1}^{m} \alpha_1$

  for $t = 2$ *to* T **do**

    $\alpha_t = \alpha_{t-1}\hat{\mathbf{\Gamma}}^{(n-1)}\hat{\mathbf{P}}^{(n-1)}(x_t)$, $r_t = \sum_{j=1}^{m} \alpha_t$, $\alpha_t = \alpha_t / \sum_{j=1}^{m} \alpha_t$

  end

  Backward Porbability Normalization:

  $\beta_T = \mathbf{1}$,

  for $t = 1$ *to* $T - 1$ **do**

    for $tt = t$ *to* T **do**

      $\beta_t = \beta_t \hat{\mathbf{\Gamma}}^{(n-1)}\hat{\mathbf{P}}^{(n-1)}(x_{tt})$, $\beta_t = \beta_t / \sum_{i=1}^{m}\sum_{j=1}^{m} \beta_t(ij)$

    end

    $\beta_t = \beta_t \mathbf{1}'$

  end

  E-Step:

  $\hat{u}_j^{(n)}(t) = (C_t = j | \mathbf{x}^{(T)}, \hat{\Theta}^{(n-1)}) \leftarrow \alpha_t(j)\beta_t(j)/(\alpha_t\beta_t')$,

  $\hat{v}_{jk}^{(n)}(t) = P(C_{t-1} = j, C_t = k | \mathbf{x}^{(T)}) = \alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)/(\alpha_t\beta_t')/r_t$

  M-Step:

  $\forall j = 1, \cdots, m$,

  $\hat{\delta}_j^{(n)} = \hat{u}_j^{(n)}(1) / \sum_{j=1}^{m} \hat{u}_j^{(n)}(j)$, $\hat{\Gamma}_{jk}^{(n)} = \sum_{t=2}^{T} \hat{v}_{jk}^{(n)}(t) / \sum_{k=1}^{m}\sum_{t=2}^{T} \hat{v}_{jk}^{(n)}(t)$

  if distribution type is Gaussian, $\hat{\mu}_j^{(n)} \leftarrow \frac{\sum_{t=1}^{T} \hat{u}_j(t)x_t}{\sum_{t=1}^{T} \hat{u}_j(t)}$, $\hat{\sigma}_j^{2(t)} \leftarrow \frac{\sum_{t=1}^{T} \hat{u}_j(t)(x_t - \hat{\mu}_j^{(n)})^2}{\sum_{t=1}^{T} \hat{u}_j(t)}$,

  if distribution type is Poisson, $\hat{\lambda}_j^{(n)} \leftarrow \frac{\sum_{t=1}^{T} \hat{u}_j(t)x_t}{\sum_{t=1}^{T} \hat{u}_j(t)}$

  Checking Convergence

  if $|\log(P_2(X^T|\hat{\Theta}^{(n)}) - \log(P_2(X^T|\hat{\Theta}^{(n-1)})| < \epsilon$,

  $\hat{\Theta}^{(N)} \leftarrow \hat{\Theta}^{(n)}$, $\hat{\delta}^{(N)} \leftarrow \hat{\delta}^{(n-1)}$, $\hat{\Gamma}^{(N)} \leftarrow \hat{\Gamma}^{(n-1)}$, **break**

end

return $\hat{\Theta}^{(N)}$, $\hat{\delta}^{(N)}$,$\hat{\Gamma}^{(N)}$

2.2 Model Selection

Though acting as a key input in the EM algorithm, the number of hidden states m may not be directly known in practice. Therefore, we need to determine it through minimizing information criteria such as AIC and BIC, which are

$$\text{AIC} = -2\log(P_2(X^T|\hat{\Theta}_m) + 2(m^2 + pm - 1)$$

$$\text{BIC}= -2\log(P_2(X^T|\hat{\Theta}_m) + 2(m^2 + pm - 1)\log(T)$$

where $\hat{\Theta}_m$ is the parameter estimate given m, and p is the number of parameters in each state dependent distribution (it is 2 for Gaussian and 1 for Poisson). Intuitively, smaller AIC and BIC indicate better fitting to data and/or less complexity in the model.

2.3 Viterbi Algorithm

With estimated parameters in hand, we can identify the most likely state sequence from time 1 to T by applying the Viterbi algorithm. This procedure is also called "global decoding". We revisit its detailed implementation by referencing related reviews ([1], [2]).

Algorithm 2: Viterbi Algorithm

Input: $\{X_t, t = 1, \cdots, T\}$,$\hat{P}$, $\hat{\delta}$,$\hat{\Gamma}$, m

Initialize $s_1(j) = \hat{\delta}\hat{P}_j(x_1)$, $bt_1(j) = 0$, $\forall j = 1, \cdots, m$

for $t = 1$ *to* T **do**

  $s_t(j) = \max_{i=1}^{m} s_{t-1}(i)\hat{\Gamma}_{ij}P_j(x_t)$, $\forall j = 1, \cdots, m$,

  $s_t = s_t / \sum_{i=1}^{m} s_t(i)$

  $bt_t(j) = \arg\max_{i=1}^{m} v_{t-1}(i)\hat{\Gamma}_{ij}P_j(x_t)$, $\forall j = 1, \cdots, m$

end

$q_T = \arg\max_{i=1}^{m} s_T(i)$

back-tracing:

for $t = T - 1$ *to* 1 **do**

  $q_t = bt_{t+1}(q_{t+1})$

end

return $q_1, \cdots, q_T$

3 Simulation

3.1 Simulation Design

In this section, we evaluate the performance of the aforementioned methods via simulation. To demonstrate that our procedure can accommodate various settings, we considered different numbers of hidden states ($m = 2, 3, 4$). The number of replications is 1000. In each repetition, the sample size is set to be 60. Additionally, the true parameters of Gaussian and Poisson HMM can be seen in the first column of Table 1 to Table 6 in Appendix. We initialize the starting value of the EM algorithm by adding some small random noises to the truth. In this way, we are more likely to reach the global maximum.

3.2 Simulation Results

In the Appendix, Tables 1 to 3 display the bias, variance, and mean-squared error (MSE) of estimators in the EM algorithm. It is clear that the estimates of transition probability and state dependent distribution are accurate on average and generally concentrate on the truth, given the small scale of MSE. However, the empirical variance is relatively large when it comes to initial probability. The reason is that, in each replication, $\hat{\delta}$ usually converges to a point mass at a particular state.

We further observed that AIC and BIC are able to reveal the correct model only when the number of hidden states is 2. As shown in Table 4, the true model is rarely chosen in scenarios with $m = 3$ or $m = 4$. By scrutinizing the output, it is found that the likelihood (the first term in information criteria) is similar as m varies for a fixed replication. As a result, AIC and BIC tend to select simpler models with few states to lower the model complexity penalty.

As for the Viterbi algorithm, we computed the mean and variance of state classification accuracy in Table 5. It enjoys excellent performance when $m = 2$. On the other hand, as m becomes larger, which brings a more difficult grouping task, the accuracy decreases. This phenomenon might be due to the insignificant difference in parameters of emission probability.

4 Applications

To illustrate how HMM explores real datasets, this section presents two empirical analyses. Specifically, we first choose the number of hidden states based on AIC, and then guess the unknown parameters as well as the hidden path by employing EM and Viterbi algorithms. Note that we extracted the first 60 observations in each dataset, which makes the sample size consistent with that in the simulation.

4.1 Gaussian HMM Application: Electricity Consumption

We collected the daily energy consumption of a customer in a North American city from the *Kaggle* website[1]. Guided by common sense that changes in temperature and humidity can lead to fluctuations in electricity demand, we hypothesize that weather is the hidden factor of this data. More concretely, we require that whether on a certain day only depends on whether on the previous day. Additionally, given the weather, electricity consumption follows the Gaussian distribution.

According to AIC and BIC, an HMM with $m = 2$ is selected. The exact estimates of parameters are included in Table 7. The time series of observed data, along with identified hidden states, are shown in

[1] https://www.kaggle.com/datasets/ranja7/electricity-consumption

Figure 1. Moreover, in Figure 2, the densities of two datasets are compared. One is the collected data, and another is simulated data based on the output of the EM algorithm. Unfortunately, given the distance between the two curves, we conclude that Gaussian HMM with $m = 2$ may not be a sensible model to fit the electricity consumption data.

4.2 Poisson HMM Application: Traffic Accident

The data containing the number of accidents that happened in the UK is also gained from the Kaggle[2]. Again, we assume that weather is the hidden factor. This is intuitive: for example, road safety is low on foggy days.

With $m = 2$ chosen by AIC and BIC, we plotted the Markov chain path in Figure 3. Moreover, similar to before, the contrast between our data and data from the simulation in Figure 4 suggests that Poisson HMM is not enough to capture the data patterns for its over-simplicity.

5 Conclusion

In sum, we studied two hidden Markov models with emission probability distribution being Gaussian and Poisson. Theoretically, we derived the maximum likelihood estimators and provided detailed procedures to learn about the properties of the HMMs. After coding up related algorithms, the validity of these inference methods is verified by Monte-Carlo simulations. Regarding the empirical application, we successfully detected hidden patterns behind electricity consumption and traffic accidents, and provided informative visualizations.

[2]https://www.kaggle.com/datasets/sadeghjalalian/road-accident-in-uk

6 Appendix

6.1 Simulation results

6.1.1 EM algorithm

Notice that μ and σ^2 stand for the Gaussian distribution's mean and variance. λ is the parameter for Poisson distribution. On the other hand, δ denotes the initial probability, and P_{ij} is the element located in row i and column j of transition kernel.

	Truth		**Bias**		**Variance**		**MSE**	
	Gaussian	**Poisson**	**Gaussian**	**Poisson**	**Gaussian**	**Poisson**	**Gaussian**	**Poisson**
μ_1	0	NA	0.0215	NA	0.0631	NA	0.0636	NA
μ_2	4	NA	0.0896	NA	0.3407	NA	0.3487	NA
σ_1^2	1	NA	-0.0191	NA	0.1218	NA	0.1222	NA
σ_2^2	4	NA	-0.4189	NA	2.2421	NA	2.4175	NA
λ_1	NA	1	NA	-0.0446	NA	0.1387	NA	0.1407
λ_2	NA	3	NA	0.1528	NA	0.5465	NA	0.5698
δ_1	0.2	0.2	0.0711	0.1371	0.1935	0.2184	0.1985	0.2372
δ_2	0.8	0.8	-0.0711	-0.1371	0.1935	0.2184	0.1985	0.2372
P_{11}	0.8	0.8	-0.0085	-0.0302	0.0086	0.0324	0.0086	0.0333
P_{21}	0.25	0.25	0.0344	0.0358	0.0165	0.0417	0.0177	0.0430
P_{12}	0.2	0.2	0.0085	0.0302	0.0086	0.0324	0.0086	0.0333
P_{22}	0.75	0.75	-0.0344	-0.0358	0.0165	0.0417	0.0177	0.0430

Table 1: Simulation result of m=2

	Truth		**Bias**		**Variance**		**MSE**	
	Gaussian	**Poisson**	**Gaussian**	**Poisson**	**Gaussian**	**Poisson**	**Gaussian**	**Poisson**
μ_1	0	NA	-0.1384	NA	0.2515	NA	0.2706	NA
μ_2	2	NA	0.0448	NA	0.7751	NA	0.7772	NA
μ_3	4	NA	0.3738	NA	0.3313	NA	0.4710	NA
σ_1^2	1	NA	-0.1475	NA	0.4118	NA	0.4336	NA
σ_2^2	4	NA	-1.9047	NA	4.2981	NA	7.9261	NA
σ_3^2	2	NA	-0.5581	NA	0.7308	NA	1.0423	NA
λ_1	NA	1	NA	-0.1361	NA	0.2669	NA	0.2855
λ_2	NA	3	NA	0.1844	NA	0.6417	NA	0.6757
λ_3	NA	5	NA	0.5614	NA	1.0577	NA	1.3728
δ_1	0.1	0.1	0.0978	0.0871	0.1561	0.1496	0.1657	0.1571
δ_2	0.3	0.3	-0.0022	0.0709	0.2056	0.2279	0.2056	0.2329
δ_3	0.6	0.6	-0.0956	-0.1580	0.2459	0.2411	0.2550	0.2661
P_{11}	0.1	0.1	0.0309	-0.0123	0.0231	0.0177	0.0240	0.0179
P_{21}	0.25	0.25	0.0782	0.0372	0.0930	0.0899	0.0991	0.0913
P_{31}	0.4	0.4	0.0033	0.0155	0.0545	0.0822	0.0545	0.0825
P_{12}	0.2	0.2	0.1110	0.1368	0.0798	0.1099	0.0921	0.1286
P_{22}	0.5	0.5	-0.0997	-0.0092	0.0952	0.0951	0.1051	0.0952
P_{32}	0.15	0.15	0.0774	0.1210	0.0571	0.0776	0.0631	0.0923
P_{13}	0.7	0.7	-0.1419	-0.1245	0.0866	0.1096	0.1067	0.1251
P_{23}	0.25	0.25	0.0215	-0.0280	0.0872	0.0703	0.0877	0.0711
P_{33}	0.45	0.45	-0.0807	-0.1365	0.0394	0.0559	0.0459	0.0745

Table 2: Simulation result of m=3

	Truth		Bias		Variance		MSE	
	Gaussian	**Poisson**	**Gaussian**	**Poisson**	**Gaussian**	**Poisson**	**Gaussian**	**Poisson**
μ_1	0	NA	-0.2735	NA	0.4664	NA	0.5412	NA
μ_2	2	NA	-0.4082	NA	0.4750	NA	0.6416	NA
μ_3	3	NA	0.2043	NA	0.3448	NA	0.3865	NA
μ_4	4	NA	0.9178	NA	0.5597	NA	1.4020	NA
σ_1^2	1	NA	-0.1783	NA	0.9368	NA	0.9686	NA
σ_2^2	4	NA	-2.5005	NA	3.2548	NA	9.5071	NA
σ_3^2	2	NA	-0.8790	NA	2.9737	NA	3.7463	NA
σ_4^2	3	NA	-1.5401	NA	1.2797	NA	3.6516	NA
λ_1	NA	1	NA	-0.0598	NA	0.4683	NA	0.4719
λ_2	NA	3	NA	0.0687	NA	0.6301	NA	0.6348
λ_3	NA	5	NA	0.1156	NA	1.1473	NA	1.1607
λ_4	NA	7	NA	0.6666	NA	2.4387	NA	2.8831
δ_1	0.1	0.1	0.0986	0.1000	0.1567	0.1575	0.1664	0.1675
δ_2	0.3	0.3	-0.0782	-0.0337	0.1699	0.1916	0.1760	0.1928
δ_3	0.2	0.2	0.0843	0.0222	0.2002	0.1690	0.2073	0.1695
δ_4	0.4	0.4	-0.1047	-0.0884	0.2048	0.2107	0.2158	0.2185
P_{11}	0.1	0.1	0.0529	0.0137	0.0359	0.0318	0.0387	0.0320
P_{21}	0.15	0.15	0.0616	0.0660	0.0636	0.0809	0.0674	0.0852
P_{31}	0.22	0.22	0.0306	0.0772	0.0686	0.1083	0.0696	0.1142
P_{41}	0.08	0.08	0.0830	0.0287	0.0491	0.0431	0.0560	0.0439
P_{12}	0.2	0.2	0.0315	0.0168	0.0776	0.0983	0.0786	0.0986
P_{22}	0.4	0.4	-0.1214	-0.0753	0.0739	0.1001	0.0886	0.1057
P_{32}	0.32	0.32	-0.0552	-0.0321	0.0830	0.1083	0.0860	0.1093
P_{42}	0.42	0.42	-0.1080	-0.0897	0.0859	0.1204	0.0976	0.1284
P_{13}	0.25	0.25	0.0527	0.0322	0.1048	0.1267	0.1076	0.1278
P_{23}	0.15	0.15	0.1020	0.0189	0.0809	0.0707	0.0913	0.0711
P_{33}	0.2	0.2	0.0230	0.0177	0.0643	0.0802	0.0648	0.0805
P_{43}	0.3	0.3	0.0537	0.1417	0.0918	0.1247	0.0947	0.1448
P_{14}	0.45	0.45	-0.1370	-0.0627	0.0937	0.1403	0.1125	0.1442
P_{24}	0.3	0.3	-0.0422	-0.0096	0.0801	0.1028	0.0819	0.1029
P_{34}	0.26	0.26	0.0016	-0.0628	0.0772	0.0778	0.0772	0.0818
P_{44}	0.2	0.2	-0.0286	-0.0807	0.0384	0.0359	0.0393	0.0424

Table 3: Simulation result of m=4

6.1.2 Model selection

model	Gaussian			Poisson		
number of hidden states	$m = 2$	$m = 3$	$m = 4$	$m = 2$	$m = 3$	$m = 4$
AIC	87.77%	10.85%	2.44%	99.80%	3%	0%
BIC	100%	0.20%	0%	100%	0.1%	0%

Table 4: Model selection accuracy based on AIC and BIC

6.1.3 Viterbi Algoirthm Decoding

model	Gaussian			Poisson		
number of hidden states	$m = 2$	$m = 3$	$m = 4$	$m = 2$	$m = 3$	$m = 4$
mean	0.9226	0.6298	0.4143	0.7713	0.5698	0.4604
variance	0.0022	0.0103	0.0048	0.0101	0.0093	0.0061

Table 5: State classification accuracy by Viterbi algoirthm

6.2 Application

		m=2	**m=3**	**m=4**	**Selected m**
Energy	AIC	873.1664	887.1664	905.1664	2
	BIC	887.8268	916.4872	953.3363	2
Accident	AIC	814.6022	826.6022	842.6022	2
	BIC	825.0739	849.6400	882.3947	2

Table 6: Model selection

	μ_1	μ_2	σ_1^2	σ_2^2	λ_1	λ_2	δ_1	δ_2	P_{11}	P_{21}	P_{12}	P_{22}
Energy (Gaussian)	1114.2440	1634.8100	206572.2900	6851.9200	NA	NA	2.8184e-12	1	0.7082	00.3370	0.2918	0.6630
Accident (Poisson)	NA	NA	NA	NA	425.9425	283.4546	0	1	0.8355	0.5315	0.1645	0.4685

Table 7: Parameters estimation

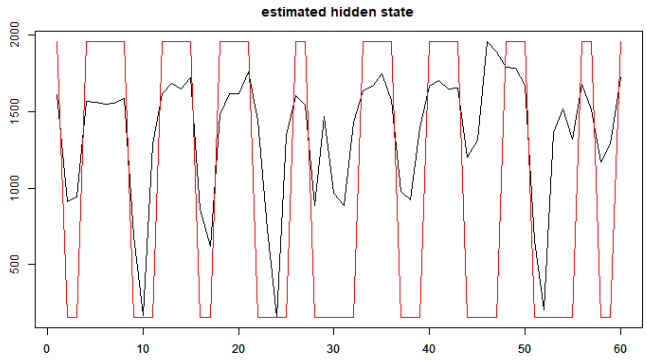

Figure 1: Gaussian Application: Comparison

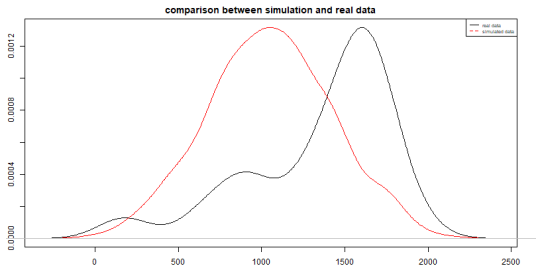

Figure 2: Gaussian Application: Comparison

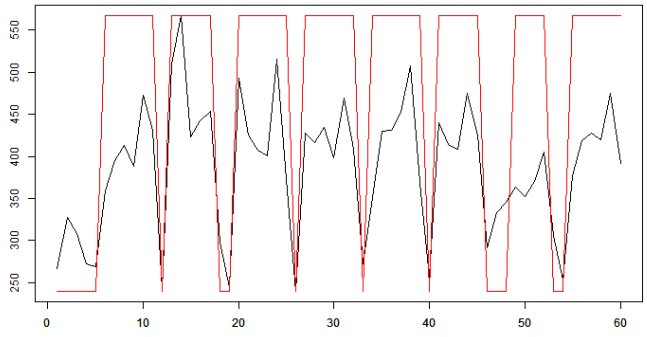

Figure 3: Poisson Application: Accident

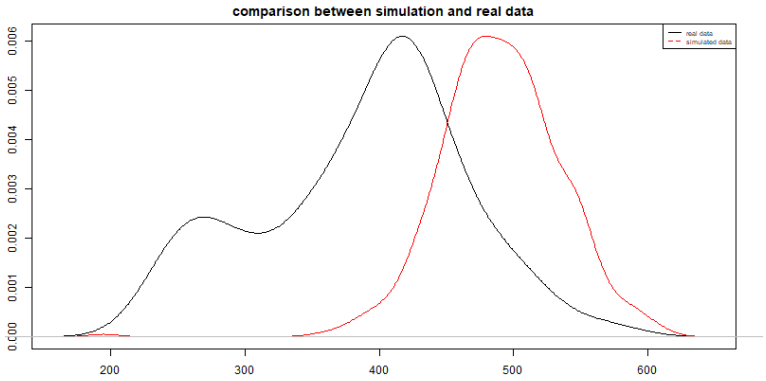

Figure 4: Poisson Application: Comparison

6.3 Code Explanation

- Firstly, to play with our code, you may put all codes in a folder called "code" and all raw data into another folder called "data". Also, please create a folder named "output" to store results.
- For simulation, the main script is *simulation.r*. By changing the working directory in Line 8 and setting the number of hidden states in Lines 20 and 221, you are able to produce all related tables and figures.
- For empirical application, the main script is *application.r*. You only need to set the working directory in Line 3 to replicate all results.

References

[1] André Inge. Hidden markov models. *Sweden: Stockholm University*, 2013.

[2] Constituency Parsing. Speech and language processing. 2009.