Find States in Time Series: HMM in Jump Model

Yuqi Nie
Department of Electrical and Computer Engineering
Princeton University
Princeton, NJ 08544, USA
ynie@princeton.edu

Abstract

This is a milestone report with goal of finding hidden states (or labels/modes) in time series data. I first conclude some popular models, and especially we see that hidden Markov model (HMM) can be unified within jump model framework. Then I have shown in experiment that HMM and simple jump model performs better respectively in the high and low self-transition probability cases. This inspires us to design a more sophisticated form of jump penalty adapted to general time series as well as develop corresponding theoretical analysis, which are the main goals of my next steps.

1 Introduction

Data clustering is essential in time series analysis, especially when data is considered to have properties based on latent states, which is widely use in many fields such as regime-switching model in stock market [1]. This can also be understood as labels/modes in classification tasks or hidden states in hidden Markov model. The main goal of this project is to develop a better algorithm that combining the advantages of current methods to find the states in time series data. Especially, I would like to focus on modifying the jump model penalty based on hidden Markov model.

Given a set of observed data $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_t}, ..., \mathbf{x_T})$ with the corresponding hidden states $\mathbf{S} = (s_1, s_2, ..., s_t, ..., s_T)$, where the states $s \in \{1, ..., k\}$ are parameterized by their centroid $\{\boldsymbol{\mu_1}, ..., \boldsymbol{\mu_k}\}$. I will introduce some popular algorithms to identify the states in time series.

1.1 K-means Clustering

K-means clustering is a widely used algorithm which clusters the observed vectors effectively with expectation–maximization (EM) algorithm. The objective is:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x_t} \in A_i} ||\mathbf{x_t} - \boldsymbol{\mu_i}||^2 \tag{1}$$

Here $A_i = \{\mathbf{x_t} : s_t = i\}$ is the data set that assigned to the state label i with centroid $\boldsymbol{\mu_i}$. The EM algorithm first fix the state label and tune $\boldsymbol{\mu_i}$ to minimize the objective function, then fix these centroids and re-assign the states.

1.2 Hidden Markov Model (HMM)

In a HMM, the hidden states can take can take discrete values, and the sequence $\mathbf{S}$ satisfies the Markov property:$p(s_t|s_{t-1}, ...s_0) = p(s_t|s_{t-1})$. Then $\mathbf{x_t}$ is drawn from a distribution parameterized by hidden states. Suppose that the HMM is parameterized by Θ, the maximum a posteriori (MAP) is:

$$\hat{\Theta} = \arg\max_{\Theta}[p(\Theta|\mathbf{X})] = \arg\max_{\Theta}[p(\mathbf{X}|\Theta)p(\Theta)] \tag{2}$$

If we don't have prior info about Θ, then $p(\Theta)$ is uniform distribution, and the MAP becomes the same as usual machine learning estimation, which can be done using EM algorithms [2]. In general, HMM are usually trained using Baum-Welch algorithm, which is a special case of the EM algorithm making use of the forward-backward algorithm [3].

1.3 Jump Model, Sparse Jump Model

Jump model is a simple but effective idea of learning latent variables and clustering time series data. Surprisingly, it was first proposed in 2018 to the best of my knowledge [3], which is fairly new. The key idea is to penalize the objective function when there is a change in hidden states [4]:

$$\min \sum_{t=1}^{T-1} [l(\mathbf{x_t}, \boldsymbol{\mu_{s_t}}) + \lambda \mathbb{I}_{s_t \neq s_{t+1}}] + l(\mathbf{x_T}, \boldsymbol{\mu_{s_T}}) \tag{3}$$

Course Project of COS 513. Spring 2022.

Intuitively, this objective tries to keep the states away from changing. Here the loss function is usually picked as $l(\mathbf{x_T}, \boldsymbol{\mu}_{\boldsymbol{s_T}}) = ||\mathbf{x_T} - \boldsymbol{\mu}_{\boldsymbol{s_T}}||^2$, but can also be chosen according to the tasks.

Sparse jump model is a modified version of jump model inspired from sparse k-means clustering [5]. It combines the purpose of selecting features and estimating states/parameters. The main idea is to change the minimization of intra-cluster distances to maximizing inter-cluster distances. Then we can weight the features and add limitation to their L_1 & L_2 norms. In this way we can do dimension reduction and feature selection effectively, especially when some features are dominated by noise in high dimensional data analysis. More details can be found in this paper [6].

2 HMM in Jump Model

It is proven [3] that HMM can be unified under the framework of general jump model. We extract some conclusion here.

As an extension of (3), the objective of general jump model is:

$$J(\mathbf{X}, \boldsymbol{\mu}, \mathbf{S}) = \sum_{t=1}^{T} l(\mathbf{x_t}, \boldsymbol{\mu}_{\boldsymbol{s_t}}) + \sum_{k=1}^{K} r(\boldsymbol{\mu_k}) + \mathcal{L}(\mathbf{S}) \quad (4)$$

Here $\mathbf{S} = (s_1, s_2, ..., s_t, ..., s_T)$ is the state sequence, $r(\boldsymbol{\mu_k})$ is the loss only depends on model parameters, and the last term defines the loss penalty related to the states and transitions:

$$\mathcal{L}(\mathbf{S}) = \mathcal{L}^{\text{init}}(s_0) + \sum_{i=1}^{T} \mathcal{L}^{\text{state}}(s_t) + \sum_{i=1}^{T} \mathcal{L}^{\text{trans}}(s_t, s_{t-1}) \quad (5)$$

The first term $\mathcal{L}^{\text{init}}(s_0)$ is the initial state cost, the second term $\mathcal{L}^{\text{state}}(s_t)$ is the cost in the later states, and the last term $\mathcal{L}^{\text{trans}}(s_t, s_{t-1})$ is then state transition cost. So easily we can see from (3) that in a simple jump model $r(\boldsymbol{\mu_k}) = \mathcal{L}^{\text{init}}(s_0) = \mathcal{L}^{\text{state}}(s_t) = 0$ and $\mathcal{L}^{\text{trans}}(s_t, s_{t-1}) = \lambda \mathbb{I}_{s_t \neq s_{t-1}}$.

In HMM, suppose the transition probabilities and the initial state distribution are:

$$p(s_{t+1} = i | s_t = j) = \pi_{i,j}, \quad i, j \in (1, ..., k) \quad (6a)$$

$$p(s_0) = \pi_{s_0} \quad (6b)$$

then the conclusion is that, if we set:

$$r(\boldsymbol{\mu_k}) = 0 \quad (7a)$$

$$\mathcal{L}^{\text{init}}(s_0) = -\log \pi_{s_0} \quad (7b)$$

$$\mathcal{L}^{\text{state}}(s_t) = 0 \quad (7c)$$

$$\mathcal{L}^{\text{trans}}(s_t, s_{t-1}) = -\log \pi_{s_t, s_{t-1}} \quad (7d)$$

then HMM model can be unified within the jump model framework. We conclude that jump model is more descriptive than HMM, and in HMM the jump penalty is $-\log \pi_{s_t, s_{t-1}}$, which indicates a lower penalty when the inter-state transition probability is high, i.e. a low self-transition probability. This phenomenon inspires our discussion on the jump penalty in the next section.

3 Mixed Penalty in Jump Model

We have seen in the previous sections that the penalty of jump model is simple and just depends on whether the state changes. This would work well when the transition probability is small, but may cause trouble in the opposite case. On the other hand, HMM provides a form of penalty which is proportional to the transition probability. It is natural to think about: can we combine those two to have a more subtle penalty for the jump model to deal with the general time series case?

Intuitively, we can write out the algorithm as follows:

Algorithm 1 Algorithm 1

Input: sequence of raw data $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_t}, ..., \mathbf{x_T})$ and penalty parameter λ

Output: model parameters $(\boldsymbol{\mu}, \Theta)$ (including centroid $\boldsymbol{\mu} = \{\boldsymbol{\mu_1}, ..., \boldsymbol{\mu_k}\}$ and other constraints-related parameters Θ) and state sequence $\mathbf{S} = (s_1, s_2, ..., s_t, ..., s_T)$

initialize Θ^0;

while $|\Theta^j - \Theta^{j-1}|/|\Theta^{j-1}| < \delta$ **do**

  initialize state sequence $\mathbf{S}^0$;

  while $\mathbf{S}^j \neq \mathbf{S}^{j-1}$ **do**

    $\boldsymbol{\mu^j} = \arg\min \sum_{t=1}^{T} [l(\mathbf{x_t}, \boldsymbol{\mu}_{\boldsymbol{s_t^{j-1}}})]$

    $\mathbf{S}^{j-1} = \arg\min\{\sum_{t=1}^{T-1} [l(\mathbf{x_t}, \boldsymbol{\mu}_{\boldsymbol{s_t^{j-1}}}) + \mathcal{P}(\lambda, s_t^{j-1}, s_{t+1}^{j-1})] + l(\mathbf{x_T}, \boldsymbol{\mu}_{\boldsymbol{s_T^{j-1}}})\}$

  end

  use constraints to update Θ^j.

end

Here the $l(\mathbf{x_t}, \boldsymbol{\mu}_{\boldsymbol{s_t^{j-1}}})$ denotes the loss function and $\mathcal{P}(\lambda, s_t^{j-1}, s_{t+1}^{j-1})$ is the penalty function for state transition. In jump model this is simple $\mathcal{P}(\lambda, s_t^{j-1}, s_{t+1}^{j-1}) = \lambda \mathbb{I}_{s_t \neq s_{t+1}}$. In HMM, this would be in the form of $\mathcal{P}(\lambda, s_t^{j-1}, s_{t+1}^{j-1}) \propto -\lambda \log \pi_{s_t^{j-1}, s_{t+1}^{j-1}}$. Our main goal is to discover a more balanced version of penalty which could combine the advantage from both sides.

4 Experiments

In this section I will show some preliminary results about the content in the previous sections. Since there is no open-source package on the web, I need to write it from scratch.

The first goal is to reproduce the models mentioned in the section 1, and the second goal is to compare the pros and cons of all the model candidates.

There are four models tested here: Gaussian HMM, Jump Model, Sparse Jump Model and K-means clustering. Our test data is generated from a three-state multivariate Gaussian HMM:

$$\mathbf{x_t}|s_t \sim \mathcal{N}(\mu_{s_t}, \mathbf{I_D}) \tag{8}$$

where $\mathbf{I_D}$ is an identity matrix, D is the dimension of feature. We set two cases with different transition probabilities of hidden states: case 1 has high self-transition probability, while case 2 has lower self-transition probability. The corresponding transition matrices are:

$$P_1 = \begin{pmatrix} 0.99 & 0.002 & 0.008 \\ 0.02 & 0.97 & 0.01 \\ 0.025 & 0.035 & 0.94 \end{pmatrix} \tag{9}$$

$$P_2 = \begin{pmatrix} 0.9 & 0.02 & 0.08 \\ 0.08 & 0.8 & 0.12 \\ 0.1 & 0.2 & 0.7 \end{pmatrix} \tag{10}$$

Besides all the other settings of these 2 cases are the same. The details of experiments can be found in Appendix A.

We here use balanced accuracy (BAC) as a metric following the paper [7]. It is defined as:

$$\text{BAC} = \frac{1}{k}\sum_{i=1}^{k}\frac{tp_i}{tp_i + fn_i} \tag{11}$$

Here tp_i, fn_i are the numbers of true positives and false negatives in state i respectively. The results are shown in the table 1.

This proves our intuition in section 3. When the self-transition probability is high, the jump model performs well because it set a high threshold for state transition, which is usually the case in stock market [6]. However when the self-transition probability is low, simply penalizing the transition would be less effective, which is the case in some biological models [8].

We further plot them intuitively in Figure 1. The plot clearly shows that, when the self-transition probability is high, jump model is better than Gaussian HMM. Otherwise the Gaussian HMM fit well with the growth truth because it is a natural maximization of the likelihood. Thus we may need a more sophisticated penalty in more general cases.

Currently I haven't write the code about this modified penalty, because it need to estimate the HMM parameter and state sequence simultaneously. This would be an essential point for the next step.

5 Conclusion and Next Steps

In this paper, I have shown the potential of jump models in learning states in time series data. Especially, the simple jump model work well in stock market and other cases with high self-transition probability in latent states. However, we have shown that its effectiveness will be influenced be the value of self-transition probability, which is usually ignored. The main goal of this project is to develop a better way of generalize jump penalty to general time series data and learn their state labels.

The next step is direct: inspired from the penalty in HMM and jump model, we can design a balanced penalty for general cases. This penalty can be adaptive, and we would also like to analyze it theoretically to ensure its generalization. Also, we would like to test it over more time series tasks with high and low self-transition probabilities. The coding part could be complex, since we will need to estimate more variables in one loop, so the convergence is also a potential problem that needs to be solved in further steps.

Table

Models	Gaussian HMM	Jump Model	Sparse Jump Model	K-means Clustering
Case 1	0.9816	**0.9949**	0.9819	0.8962
Case 2	**0.9594**	0.9486	0.7655	0.9220

Table 1: BAC in case 1: high self-transition probability; case 2: low self-transition probability.

Figure

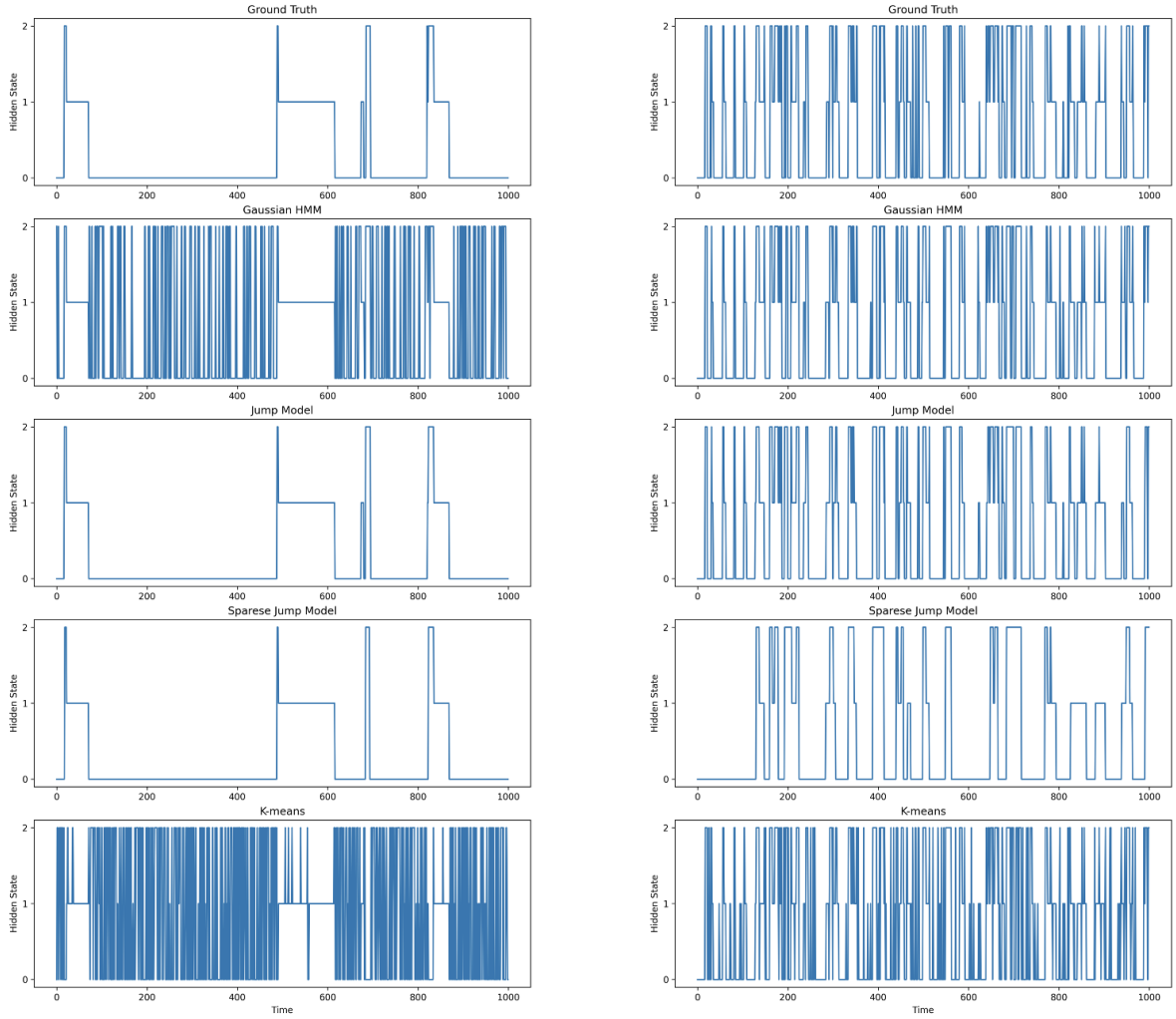

Figure 1: An example of the true state sequence and estimated state sequences for Gaussian HMM, jump model, sparse jump model and k-means clustering. There are 3 states and $T = 1000$. **Left column**: Case 1: high self-transition probability. **Right column**: Case 2: low self-transition probability.

Acknowledgement

I appreciate the helpful discussion with Onat Aydinhan, a PhD candidate in Princeton ORFE department. Also I would like to thank Prof. Adji Bousso Dieng's instruction and assistant instructors' help in this course.

References

[1] F.I.A. Mary R. Hardy A.S.A. A regime-switching model of long-term stock returns. *North American Actuarial Journal*, 5(2):41–53, 2001. doi: 10.1080/10920277.2001.10595984. URL https://doi.org/10.1080/10920277.2001.10595984.

[2] Hsiao-Wuen Hon Xuedong Huang, Alex Acero and Raj Reddy. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001. URL https://dl.acm.org/doi/abs/10.5555/560905.

[3] Alberto Bemporad, Valentina Breschi, Dario Piga, and Stephen P. Boyd. Fitting jump models. *Automatica*, 96: 11–21, 2018. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2018.06.022. URL https://www.sciencedirect.com/science/article/pii/S0005109818303145.

[4] Peter Nystrup, Petter N. Kolm, and Erik Lindström. Greedy online classification of persistent market states using realized intraday volatility features. *The Journal of Financial Data Science*, 2(3):25–39, 2020. ISSN 2640-3943. doi: 10.3905/jfds.2020.2.3.025. URL https://jfds.pm-research.com/content/2/3/25.

[5] Daniela M. Witten and Robert Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, 2010. doi: 10.1198/jasa.2010.tm09415. URL https://doi.org/10.1198/jasa.2010.tm09415. PMID: 20811510.

[6] Petter N. Kolm Peter Nystrup and Erik Lindström. Feature selection in jump models. *Expert Systems with Applications*, 184:115558, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115558. URL https://www.sciencedirect.com/science/article/pii/S0957417421009647.

[7] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, 2010. doi: 10.1109/ICPR.2010.764.

[8] Byung-Jun Yoon. Hidden markov models and their applications in biological sequence analysis. *Current Genomics*, 10(6):402–415, 2009. URL https://dx.doi.org/10.2174/138920209789177575.

A Experiment Settings

Data Generation. The data is generated with a Gaussian HMM described in (8) using the $hmmlearn$ package. The length is $T = 1000$, and the dimension of feature is $D = 20$. There are 3 hidden states, and the mean in each state is:

$$(\boldsymbol{\mu_1})_d = \mu\mathbb{I}_{d\leq 10}, \quad (\boldsymbol{\mu_2})_d = 0, \quad (\boldsymbol{\mu_3})_d = \mu\mathbb{I}_{d\leq 10}$$

That is to say, there are 10 features which are purely white noise. This setting could help us to evaluate the robustness and feature selection ability of model in future. The covariance matrix are identity matrixes as described in (8). The initial state distribution is $\pi_0 = (0.4, 0.3, 0.3)$, and the state transition probability is given by (9) - (10).

Gaussian HMM. The Gaussian HMM is implemented by using the $hmmlearn$ package. The transition matrix prior is set as 1, and the maximum iteration is 100.

Jump Model & Sparse Jump Model. There are some references on the web. However I haven't found a open library so far. With the help of Onat Aydinhan, I wrote it myself and it seems to work. The penalty parameter is set as $\lambda = 10$, and for sparse jump model the maximum norm of feature weight is set is 1.

The jump model seems to work good in our test. However sparse jump model has poorer performance than jump model in both case 1 & 2, which may due to the lack of tuning in its maximum norm parameter.

K-means Clustering. K-meaning clustering is a classical algorithm, and we use the $cluster.KMeans$ from $sklearn$ package. The random state is set as 0.

Evaluation. The evaluation metric is BAC in (11). We generate the data randomly for 5 times and evaluate the four models, and the results in table 1 is the average of 5 experiments. The figure 1 shows the visualization of one experiment.