Poisson HMM notes

March 19, 2025

1 Motivation for $PHMM$

1.1 Why a Poisson model?

The reason for choosing a Poisson process for Earthquales is because upon reading a lot of earthquale modelling papers to see how we can use the probability transitions, several of them relied on poisson processes (ex literature link).

Poisson processes are really good for modelling rare, discrete events over time, such as earthquakes. It assumes that events occur independently at a constant average rate (λ).

> As oppposed to Gaussian emission which is standard for normal HMM and what I said in the previous meetings, using such Poisson emissions might perform better.

1.2 Why the hidden states

Trying to build up on different "states" for the HMM, The "hidden" aspect of the HMM allows for the inclusion of unobserved (latent) states that influence the earthquake occurrence rate.

For example, the Earth's crust can be in different states of stress accumulation or release, which are not directly observable but affect the likelihood of earthquakes. Each hidden state can have its own Poisson rate parameter (λ), representing different levels of seismic activity (e.g., low, medium, high).

Earthquakes often exhibit temporal **clustering** (e.g., aftershocks following a mainshock) or periods of quiescence. A $PHMM$ can capture these dependencies by allowing transitions between hidden states over time. For example: - A "high activity" state might correspond to aftershock sequences. - A "low activity" state might correspond to periods of stress accumulation.

So, $PHMMs$ can model both the randomness of earthquake occurrences and the underlying hidden states driving them.

Interpretability: The hidden states can provide insights into the seismic cycle (e.g., stress buildup and release).

Adaptability: The model can be updated as new data arrives, improving its predictions over time.

2 How does it work

A useful intro site for $PHMM$: 1. $PHMM$ basics 2. $PHMM$ implementation

The main motivation behind the development of $PHMM$ is to enable a Poisson model to account for autocorrelation in the time series data. Especially, with our high, and low intensity markov model.

The Poisson Hidden Markov Model goes one step further by mixing in a discrete k-state Markov model that is 'hidden' in that at each time step in the time series, one does not know for sure which Markov regime the Markov process is in. Instead, at each time step, one estimates the effect of the possible existence of each regime on the mean value that is predicted by the Poisson model.

2.1 Key components:

1. **Hidden states:** a finite set of unobserved (latent) states, $S = \{S_1, S_2, \dots, S_n\}$
2. **Transition probabilities:** a matrix, $\widetilde{P}$, where $\widetilde{P}_{ij}$ is $\mathbb{P}(S_{n+1} = j | S_n = i)$
3. **Observation model:** The observations are counts (non-negative integers) modeled using a Poisson distribution. Each state S_i has an associated Poisson rate parameter i i, which determines the expected number of events in that state.
4. **Initial State probabilities:** A vector $\pi = \{\pi_i\}$ where $\pi_i = \mathbb{P}(S_1 i)$.

2.2 Mathematical formulation

At each time step t, the system is in one of the hidden states

Given the state $S_t = i$, the observation y_t (count) is generated from the following distribution:

$$y_t \sim \text{Poisson}(\lambda_i) \implies \mathbb{P}(y_t | S_t = i) = \frac{e^{-\lambda_i} \lambda_i^{y_t}}{y_i!}$$

We have the following joint likelihood of the hidden states and observations:

$$\mathbb{P}(S, y) = \mathbb{P}(S_1) \prod_{t=2}^{T} \mathbb{P}(S_t | S_{t-1}) \prod_{t=1}^{T} \mathbb{P}(y_t | S_t)$$

where: - $\mathbb{P}(S_1) = \pi_{S_1}$ (initial state probability), - $\mathbb{P}(S_t \mid S_{t-1}) = \widetilde{P}_i j$ (transition probability), - $\mathbb{P}(y_t \mid S_t)$ is the Poisson likelihood.

Mathematically, the formulation of it is very close to Poisson regression:

3 Fitting an HMM Poisson model

I think the default `hmmlearn` uses EM for the parameter estimation: 1. E : Compute the expected values of the hidden states given the current parameter estimates 2. M : Update the parameters numerically to maximise the expected likelihood

If we assume ϕ as a vector of all the estimation parameters and L^c refers to the joint likelihood

$$\phi = (\widetilde{P}_{1,2}, \widetilde{P}_{1,3}, \dots, \widetilde{P}_{n-1,n}, \lambda_1, \lambda_n)$$

$$Q(\phi, \phi^k) = \mathbb{E}_{\phi^k}[\ln L_T^c(\phi) | y]$$

Our observations are denoted with y_t and therefore the state dependent probability have the same distribution as above:

$$\pi_{y_t,i} = \frac{e^{-\lambda_i}\lambda_i^{y_t}}{y_i!}$$

We then maximise in the M step and search ϕ^k that maximise $Q(\phi, \phi^k)$ such that

$$Q(\phi^{k+1}; \phi^k) > Q(\phi, \phi^k) \qquad \forall \phi \in \Phi$$

- Φ is the parameter space $\in \mathbb{R}^{n^2+n}$

We can do inference via viterbi or via forward-backward - I think `PoissonHMM` in python I used uses viterbi and it was relatively fast, plus, it is faster when doing prediction and inference in the same step with it.

I ran a simple fitting algorithm for `PoissonHMM` from `hmmlearn` library on monthly and yearly aggregrated data for states being from `1 to 5` and choose the best state model: - just as a proof of concept and see if it is computationally intensive

We can see the implementation of the forward-backward probabilities in the following paper: paper but I think we can treat it like black box model as the inference is implemented in most *hmm* models in python (furthermore, I am not sure if we will can use just forward-backward or use a combination of that and viterbi algorithm to implement it).

The main idea for forward-backward is use the following α_t, β_t and substitute in $Q(\phi, \phi^k)$, then run EM to maximise it:

- Forward probabilities:

$$\alpha_t(j) = \mathbb{P}(Y_1 = y, ..., Y_t = y_t, S_t = j) = \left(\sum_{i \in S_X} \alpha_{t-1}(i)\widetilde{P}_i j\right) \pi_{y_t,j}$$

- Bakward probabilities:

$$\beta_t(j) = \sum_{i \in S_X} \pi_{y_{t+1},i}\beta_{t+1}(i)\widetilde{P}_{j,i}$$

Might have to double check for our model assumptions, but the MLE for $\widetilde{P}_i j$ for the $k+1$ iteration is given by:

$$\lambda_i^{k+1} = \frac{\sum_{t \in T} \alpha_t^k(i)\beta_t^k(i)y_t}{\sum_{t \in T} \alpha_t^k(i)\beta_t^k(i)}$$

4 Trying $PHMM$ and evaluating fitting

So, I have used the `earthquake.csv` data to fit the model on the training data to visualise the "time series" well. But we can use the same idea to fit the $PHMM$ on the most recent n data and then do inference via the two methods and predict the probabilities of the next earthquake.

Notes: - I treated the counts of the earthquakes as a time series and then using $PHMM$ to determine the "high" and "low sensitivity" events and then aggregrate on it. - We could use magnitude and location in our observation set and then look at HMM using higher dimensional data but I am not sure which emission probabilities would work the best and I have certain reservations

about the model fit being well, the reason for using Poisson process in this case is because of several abundance of papers using some form of Poisson Process model to do so. - The model was aggregrated on yearly and monthly data with a small number of parameters for a proof of concept approach and seeing if the model converged and the estimated parameters made sense (check `poisson_HMM_research.ipynb` for relevant graphs), The monthly data made sense but yearly data did not as much. - we can change the granuality of the time units to have a more precise model and also allow for more possible states, more data and larger search should result in a better $\widetilde{P}$