Hidden Markov Models for Earthquake Prediction: GaussianHMM Analysis

Overall summary

Main Idea:

- split data into training (80%) and testing (20%) chronologically
- train Gausssian HMM based on the training data
- use best performing HMM parameter (based on log-likelihood, $\mathcal{L}$) to forecast values
  - the best X_{train} is based on the time series cross validation with `n_splits=3`
  - otherwise it was always choosing the highest number fo states and fitting to it
- Get the $\mathcal{L}, AIC, BIC$ of the best performing model on the testing data

$$\begin{aligned} \mathcal{L}: &\quad 9755.188755348854 \\ AIC: &\quad 8635.969558918778 \\ BIC: &\quad 12679.881862766959 \end{aligned}$$

- Use the HMM to fit states on the testing data based on feature thresholds and differing criteria
  - criteria that fits the states on the test features the best is chosen based on most accuracy
- Compare fitted value against forecasted values to get the following:

For reference, the training metrics were:

$$\begin{aligned} \mathcal{L}: &\quad -5590.1346817662425 \\ AIC: &\quad 11552.269363532485 \\ BIC: &\quad 12679.881862766959 \end{aligned}$$

Tried with univariate models on just the counts for Poisson (and also Gaussian):
Poisson:

$$\begin{aligned} \mathcal{L}: &\quad -2536.0145200794263 \\ AIC: &\quad 5434.029040158853 \\ BIC: &\quad 6403.977432758818 \end{aligned}$$

Gaussian:

$$\begin{aligned} \mathcal{L}: &\quad 5456.240959411002 \\ AIC: &\quad -10230.481918822004 \\ BIC: &\quad -8403.120582487261 \end{aligned}$$

HOWEVER, there are couple of issues with just using univariate data based on counts of above a threshold

1. Loss of risk-critical information: A univariate count model ignores magnitude, depth, location, and energy release patterns that are essential for assessing actual financial/infrastructure risk
   i. a single magnitude 8.0 earthquake has vastly different hazard than three magnitude 6.0 events, despite both scenarios having different "counts"
2. No actionable insights for risk mitigation: Risk modelling or portfolio optimisation models will need to understand what types of seismic conditions drive risk (shallow vs deep events, proximity to assets, energy buildup patterns) to make informed hedging or allocation decisions - a simple count prediction provides no guidance on which geographic regions, asset types, or time horizons require different risk premiums

> The HMM pipeline is given in much more detail below:

1. Gaussian Hidden Markov Models (GaussianHMM)

1.1 Mathematical Foundation

A Hidden Markov Model (HMM) is a statistical model that assumes the system being modeled is a Markov process with unobserved (hidden) states. For earthquake prediction, we use GaussianHMM where the observable features follow a multivariate Gaussian distribution.

Model Components

A GaussianHMM is characterized by:

- **Number of hidden states**: N
- **Initial state probabilities**: $\pi = \{\pi_i\}$ where $\pi_i = P(q_1 = S_i)$
- **State transition probabilities**: $A = \{a_{ij}\}$ where $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$
- **Emission probabilities**: Gaussian distributions for each state

Mathematical Formulation

For a sequence of observations $O = \{o_1, o_2, ..., o_T\}$ and hidden states $Q = \{q_1, q_2, ..., q_T\}$:

State Transition Probability:

$$P(q_{t+1} = j | q_t = i) = a_{ij}$$

Emission Probability (Gaussian): For state i, the observation o_t follows a multivariate Gaussian distribution:

$$b_i(o_t) = P(o_t | q_t = i) = \mathcal{N}(o_t; \mu_i, \Sigma_i)$$

Where:

$$\mathcal{N}(o_t; \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2} (o_t - \mu_i)^T \Sigma_i^{-1} (o_t - \mu_i) \right)$$

- μ_i: mean vector for state i
- Σ_i: covariance matrix for state i
- d: dimensionality of observations

Joint Probability

The joint probability of observations and hidden states:

$$P(O, Q | \lambda) = \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t)$$

Where $\lambda = \{\pi, A, B\}$ represents the model parameters.

Forward-Backward Algorithm

Forward variable:

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, q_t = i | \lambda)$$

Recursive computation:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$$

Backward variable:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T | q_t = i, \lambda)$$

Likelihood computation:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

1.2 Feature Engineering for Earthquake Data

Our GaussianHMM uses the following earthquake features:

1. **Inter-event Time**: Time between consecutive earthquakes (days)
2. **Depth**: Earthquake depth (km)
3. **Magnitude**: Earthquake magnitude (Richter scale)
4. **Time Since Magnitude 6.5+**: Days since last major earthquake
5. **Energy**: Seismic energy release: $E = 10^{1.5M+4.8}$ Joules
6. **Distance from Reference**: Haversine distance from reference point (24.785°N, 121.006°E)

Energy Budget Computation

Energy budget analysis using rolling window approach:

$$\text{Energy Budget Ratio} = \frac{\sum_{i=t-w}^{t} E_i}{\bar{E} \times w}$$

Where:

- E_i: Energy of earthquake i
- $\bar{E}$: Average daily energy release
- w: Window size (30 days)

2. Validation and Testing Methodology

2.1 Model Evaluation Metrics

2.1.1 Log-Likelihood and Information Criteria

Log-Likelihood:

$$\mathcal{L}(\lambda) = \log P(O|\lambda) = \log \sum_{Q} P(O, Q|\lambda)$$

Akaike Information Criterion (AIC):

$$\text{AIC} = 2k - 2\mathcal{L}(\lambda)$$

Bayesian Information Criterion (BIC):

$$\text{BIC} = k \log(n) - 2\mathcal{L}(\lambda)$$

Where:

- k: Number of model parameters
- n: Number of observations
- For GaussianHMM: $k = N(N-1) + (N-1) + Nd + Nd = N^2 + 2Nd - 1$

2.1.2 Mean Squared Error (MSE)

For forecasting evaluation:

$$\text{MSE} = \frac{1}{T} \sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2$$

Where x_t is actual observation and $\hat{x}_t$ is forecasted value.

2.2 State Validation Methods

2.2.1 Threshold-Based State Categorization

States are categorized using statistical bounds:

$$\text{Lower Threshold}_i = \mu_i - 2\sigma_i$$

$$\text{Upper Threshold}_i = \mu_i + 2\sigma_i$$

Categorization Criterion: An observation is assigned to state j if at least $\eta \times 100\%$ of its features fall within the thresholds:

$$\text{Assigned State} = \arg\max_j \left\{ \frac{\sum_{d=1}^{D} \mathbb{I}[\mu_{j,d} - 2\sigma_{j,d} \leq x_d \leq \mu_{j,d} + 2\sigma_{j,d}]}{D} \geq \eta \right\}$$

Where $\mathbb{I}[\cdot]$ is the indicator function and $\eta \in [0.5, 1.0]$ is the criteria threshold.

2.2.2 Classification Performance Metrics

Accuracy: Percentage of correctly classified states **Precision, Recall, F1-score:** Standard classification metrics **Confusion Matrix:** Cross-tabulation of predicted vs actual states

2.3 Forecasting Validation

2.3.1 State-Based Forecasting

For each hidden state i, the forecasted observation is:

$$\hat{x}_t = \mu_i \text{ where } i = \arg\max_j P(q_t = j | O_{1:t-1})$$

2.3.2 Temporal Analysis

- **Hidden State Time Series**: Analysis of state transitions over time
- **State Persistence**: Duration analysis of consecutive state occurrences
- **Transition Patterns**: Identification of common state transition sequences

2.4 Severe Event Prediction

Severe Event Definition:

$$\text{Severe Event} = \begin{cases} 1 & \text{if } (M \geq 6 \text{ and } D \leq 100) \text{ or } (M \geq 5.5 \text{ and } D \leq 10) \\ 0 & \text{otherwise} \end{cases}$$

Where M is magnitude and D is depth in kilometers.

2.5 Cross-Validation Strategy

1. **Temporal Split**: Training on historical data, testing on recent events
2. **Walk-Forward Validation**: Incremental training with rolling prediction windows
3. **State Consistency Check**: Verification that similar conditions yield similar states

3. Implementation Details

3.1 Model Training Process

1. **Data Preprocessing**: Feature scaling and normalization
2. **Model Selection**: Grid search over number of states (2-10)
3. **Parameter Optimization**: Expectation-Maximization algorithm
4. **Convergence Criteria**: Log-likelihood improvement threshold
5. **Model Selection**: Best model based on BIC/AIC scores

3.2 Validation Pipeline

1. **Load trained model**: Best performing GaussianHMM from training phase
2. **Feature computation**: Extract same features from test data
3. **State fitting**: Fit the state on the test data based on the extracted features and thresholds of the given features
4. **State comparison**: Compare the fitted states against the predicted states (using ViterBi algorithm)
5. **Confusion matrix**: Compute confusion matrix based on the plot