O2O API Reference

Youness Diouane

Classes

class BivariateHawkesProcessSimulator(M, T, alpha = None, gamma = None, mu = None)

Simulator for a bivariate Hawkes process with two marks (1: offline, 2: online) over a time window $[0, T]$. The model is described by the coupled conditional intensity

$$\lambda_1^{\mathrm{g}} = \mu_1^{\mathrm{user}} + \sum_{k:t>t_k^1}^{N_{\mathrm{user}}^1} \alpha_{11}\gamma_{11} e^{-\gamma_{11}(t-t_k^1)} + \sum_{k:t>t_k^2}^{N_{\mathrm{user}}^2} \alpha_{12}\gamma_{12} e^{-\gamma_{12}(t-t_k^2)}$$

$$\lambda_2^{\mathrm{g}} = \mu_2^{\mathrm{user}} + \sum_{k:t>t_k^1}^{N_{\mathrm{user}}^1} \alpha_{21}\gamma_{21} e^{-\gamma_{21}(t-t_k^1)} + \sum_{k:t>t_k^2}^{N_{\mathrm{user}}^2} \alpha_{22}\gamma_{22} e^{-\gamma_{22}(t-t_k^2)},$$

where $\alpha_{ij},; i,j = 1,2$ represents the expected number of events of type j directly triggered by an initial event of type i; $\gamma_{ij},; i,j = 1,2$ denotes the decay rate from event type j to type i. The pair $(N_{\mathrm{user}}^1, N_{\mathrm{user}}^2)$ indicates the number of online events (e.g., negative comments on Facebook) and offline events (e.g., shootings) for a specific user, respectively. Finally, μ_1^{user} and μ_2^{user} denote the baseline intensities of online and offline activity for that user.

Parameters:

- `M (int)`: Number of independent users.
- `T (int)`: Time horizon of the simulation.
- `alpha (2x2 array, optional)`: Reproduction matrix. Defaults to a preset.
- `gamma (2x2 array, optional)`: Decay rates matrix. Defaults to a preset.
- `mu (Mx2 array, optional)`: Baseline rates. Defaults to fixed offline/online rates.

Methods

```
def simulate(self, save_online_data = False, save_offline_data = False)
```

Parameters:

- `save_online_data` (`bool`, optional): If `True`, saves the simulated online data (mark = 1) to a pickle file `online_data.pkl`.

- `save_offline_data` (`bool`, optional): If `True`, saves the simulated offline data (mark = 0) to a pickle file `offline_data.pkl`.

Returns

- `online_data` (`list of lists`): A list of M elements, where each sublist contains the timestamps (floats) of online events (mark = 1) for one user over the interval $[0, T]$.
- `offline_data` (`list of lists`): A list of M elements, where each sublist contains the timestamps of offline events (mark = 0) for one user over the interval $[0, T]$.

class ParameterEstimator(data_online, data_offline, M, T)

Estimates the model parameters of the bivariate Hawkes process and produces summary statistics. Specifically, it computes: (i) the spillover effect that each event type has on the other (α_{ij}), along with its 95% confidence interval (CI); (ii) the decay rate of cross- and self-excitation between event types (γ_{ij}) and its 95% CI; (iii) the baseline intensities of online and offline events for each user; and (iv) the percentage effect that each event type has on the other.

Note: The inputs `data_online` and `data_offline` can be either simulated using the class **BivariateHawkesProcessSimulator** or real-world event data. In both cases, the data must be structured as a list of length M, where each element corresponds to one user and contains a list (or array) of timestamps (floats) marking the times at which events occurred for that user. These timestamps must be expressed in consistent units—typically in days—and should represent actual event times relative to a common starting point (e.g., day 0).

Parameters

- `data_online` (`list of np.arrays`): Timestamps of online events per user.
- `data_offline` (`list of np.arrays`): Timestamps of offline events per user.
- `M` (`int`): Number of users.
- `T` (`int`): Observation period.

Methods

def `fit_model(data_online, data_offline, M, T, save_fit = False)`

Fits the data to the Bivariate Hawkes process and infers the model parameters α_{ij}, γ_{ij} and μ_i by maximizing the likelihood throufh a Bayesian approach by sampling from a posterior distribution given priors on the model parameters

using the probabilistic software Stan, which leverages an MCMC algorithm. The log-likelihood is approximated by

$$L(\theta) \simeq \sum_{\text{user}} \Bigg[\sum_{k=1}^{N^1_{\text{user}}} \log\left(\lambda_1(t^1_k)\right) + \sum_{k=1}^{N^2_{\text{user}}} \log\left(\lambda_2(t^2_k)\right) - \mu_1^{\text{user}} T - \mu_2^{\text{user}} T - (\alpha_{11} + \alpha_{21})\, N^1_{\text{user}} - (\alpha_{22} + \alpha_{12})\, N^2_{\text{user}} \Bigg]$$

In Stan we perform Hamiltonian Monte Carlo with 1000 samples. For the prior densities we use,

$\alpha_{11} \sim beta(1,1)\ ,\ \alpha_{12} \sim beta(1,1)\ ,\ \alpha_{21} \sim beta(1,1)\ ,\ \alpha_{22} \sim beta(1,1)$
$\gamma_{11} \sim cauchy(0,5)\ ,\ \gamma_{12} \sim cauchy(0,5)\ ,\ \gamma_{21} \sim cauchy(0,5)\ ,\ \gamma_{22} \sim cauchy(0,5)$
$\mu_1^{\text{user}} \sim cauchy(0,5)\ ,\ \mu_2^{\text{user}} \sim cauchy(0,5)$

Parameters:

- `data_online` (`list of arrays`): A list of length M, where each element is an array of timestamps (floats) corresponding to online events for one user.
- `data_offline` (`list of arrays`): A list of length M, where each element is an array of timestamps (floats) corresponding to offline events for one user.
- `M` (`int`): The number of users.
- `T` (`int or float`): The observation window length.
- `save_fit` (`bool`, optional): If `True`, the fitted model (posterior samples) will be saved as a pickle file to disk. Default is `False`.

Returns

A fitted Stan object containing posterior samples of the Hawkes process model parameters

def `spillover_and_decays_values_and_CI(self, save_alpha = False, save_beta = False)`

Parameters:

- `save_alpha` (`bool`, optional): If `True`, saves the estimated spillover effect table (alpha values and their 95% confidence intervals) to a CSV file named `alpha_estimates.csv`. Default is `False`.
- `save_gamma` (`bool`, optional): If `True`, saves the estimated decay rate table (gamma values and their 95% confidence intervals) to a CSV file named `gamma_estimates.csv`. Default is `False`.

Returns

`alpha_df` (`pandas.DataFrame`): A table summarizing the estimated spillover effects (α_{ij}) between event types, along with their corresponding 95% confidence intervals. Each row describes the direction of influence (e.g., online → offline) and includes the point estimate and confidence interval.

`gamma_df` (`pandas.DataFrame`): A table summarizing the estimated decay rates (γ_{ij}) for self- and cross-excitation effects, with their 95% confidence intervals. It provides insight into how quickly the influence of past events decays over time.

def `base_and_CI(save_mu = False)`

Parameters:

- `save_mu` (`bool`, optional): If `True`, the resulting table of baseline intensities and their 95% confidence intervals will be saved to a CSV file named `"estimated_mu.csv"`. Default is `False`.

Returns

- `pd.DataFrame`: A table containing the estimated baseline intensities for both online and offline events per user, along with their 95% confidence intervals.

def `spillover_percentage(save_percentages = False)`

For each user, the method calculates the estimated percentage of online events attributed to offline events, and vice versa. It also includes aggregate percentages summarizing the overall cross-type influence across all users. The percentages are calculated using the formula

$$\%\text{Offline} \to \text{Online} = 100\left(1 - \frac{\bar{n}^0_{\text{Online}}}{\bar{n}_{\text{Online}}}\right)$$
$$\%\text{Online} \to \text{Offline} = 100\left(1 - \frac{\bar{n}^0_{\text{Offline}}}{\bar{n}_{\text{Offline}}}\right),$$

,
where $\bar{n}_{\text{Offline}}$ and $\bar{n}_{\text{Online}}$ represent the expected number of events per unit time and satisfy

$$\begin{pmatrix}\bar{n}_{\text{Online}} \\ \bar{n}_{\text{Offline}}\end{pmatrix} = (I - \alpha)^{-1}\begin{pmatrix}\mu_1 \\ \mu_2\end{pmatrix}, \tag{1}$$

whereas $\bar{n}^0_{\text{Online}}$ accounts for the expected number of online negative posts when $\alpha_{12} = 0$ (e.g. there is no contribution of offline attacks). Similarly, $\bar{n}^0_{\text{Offline}}$ corresponds to the expected number of offline attacks when $\alpha_{21} = 0$ (e.g. there is no contribution of online activities).

Parameters:

- `save_percentages` (`bool`, optional): If `True`, the resulting spillover percentage table is saved to a CSV file named `estimated_percentages_effects.csv`. Default is `False`.

Returns

- `pd.DataFrame`: A table containing the estimated percentage of online events attributed to offline events, and vice versa, for each user. Includes an additional row with aggregate percentages summarizing overall spillover effects across all users.

class O2OAnalyzer(data_online, data_offline, M, T, fit)

Analyzes online and offline event data for a group of users. It provides:

- A summary of the number of online and offline events per user.
- Time series plots of conditional event intensities.

Parameters:

- `data_online` (`list of np.arrays`): Timestamps of online events per user.
- `data_offline` (`list of np.arrays`): Timestamps of offline events per user.
- `M` (`int`): Number of users.
- `T` (`int`): Observation period.
- `fit` (`dict-like object`): The posterior samples from Stan containing parameter estimates such as baseline intensities (μ), spillover effects (α), and decay rates (γ).

Methods

`def sample_size(save_sizes = False)`

Parameters:

- `save_sizes` (`bool`, optional): If `True`, the method saves the resulting table of sample sizes for each user online and offline as a CSV file named `sizes.csv`. Defaults to `False`.

Returns

- `pd.DataFrame`: A table containing the number of online and offline events per user.

`def plot_intensity(save_figs = False)`

Parameters:

- `save_figs` (`bool`, optional): If set to `True`, saves each user's coupled online–offline intensity plot as an EPS file. Default is `False`.

Returns

- `None`: This method generates and optionally saves plots of the estimated online and offline intensities over time for each user. The figures are displayed using `matplotlib` and can be saved in EPS format if `save_figs` is set to `True`.