Grouped Hawkes Model Documentation

1 Model set-up and observed data

We assume an obervsation window of $[0, T]$, with observed data are the times of the policy violations $\{x_{upk}\}$ for $u = 1, \cdots, U$, $p = 1, \cdots, P$ and $k = 1, \cdots, M_{up}$. x_{upk} is the times of the k-th violation of policy p by user u. We interpret the observations $\{x_{upk}\}_{k=1}^{M_{up}}$ as realisations of a point process, with corresponding count process $N_{up}(t) = \sum_{k=1}^{M_{up}} \mathbb{I}_{[0,T]}(t_{upk})$. We will assume that counts on each edge are governed by a multivariate Hawkes process, independent between users.

Users are considered to have a latent group structure, with the group identity of user u, z_u, being such that $z_u \in \{1, \cdots, H\}$. The conditional intensity of the interactions of user u with process p is then

$$\lambda_{up}(t|\mathcal{H}_{ut}) = \sum_{h=1}^{H} \pi_h \left\{ \mu_{ph} + \sum_{j=1}^{P} \sum_{k=1}^{N_{uj}(t)} \beta_{jph} \exp\left\{-\theta_{jph}(t - t_{ujk})\right\} \right\}. \tag{1}$$

We write $x = \left(x_1^T, \cdots, x_U^T\right)^T$ to be our arrival time data and $z = (z_1, \cdots, z_U)^T$ the vector of group identities. Write $\psi = \{\pi_h, \varphi_h\}_{h=1}^{H}$ to store all parameters of interest, with $\varphi_h = \left(\mu_h^T, \beta_h^T, \theta_h^T\right)^T$ where $\mu_h = (\mu_{1h}, \cdots, \mu_{Ph})^T$ and $\beta_h = (\beta_{11h}, \cdots, \beta_{P1h}, \beta_{12h}, \cdots, \beta_{PPh})^T$, with θ_h taking an analogous form to β_h.

The log-likelihood of user u and process p, given that they belong to group h, is

$$\ell_{up}(\psi|x_u, z_u = h) = \sum_{k=1}^{M_{up}} \log \lambda_{up}\left(t_{upk}|\mathcal{H}_{ut_{upk}}, z_u = h\right) - \int_0^T \lambda_{up}\left(t|\mathcal{H}_{ut}, z_u = h\right) \mathrm{d}t. \tag{2}$$

We will state this loglikelihood explicitly, as it will be used within our code. After evaluating the integral, we find

$$\begin{aligned} \ell_{up}(\psi|x_u, z_u = h) = &\sum_{k=1}^{M_{up}} \log \left\{ \mu_{pz_u} + \sum_{j=1}^{P} \beta_{jph} A_{z_u}^{u,p,j}(N_{uj}(t_{upk})) \right\} - T\mu_{pz_u} \\ &- \sum_{j=1}^{P} \frac{\beta_{jpz_u}}{\theta_{jpz_u}} \sum_{t_{uj\ell} < T} \left(1 - \exp\left\{-\theta_{jpz_u}(T - t_{uj\ell})\right\}\right) \end{aligned} \tag{3}$$

where we define

$$A_h^{u,p,j}(k) = \sum_{t_{uj\ell} < t_{upk}} \exp\left\{-\theta_{jph}(t_{upk} - t_{uj\ell})\right\}$$

with $A_h^{u,p,j}(0) = 0$.

2 Model inference

We run an EM-style update, with gradient ascent steps taken at the M-step, which is why it's not a true EM procedure. We will use a superscript of (m) to indicate that we are at the m-th step of the procedure.

2.1 The E-step

Compute

$$\gamma_{uh}^{(m)} \propto \mathcal{L}_u\left(\varphi^{(m)}|z_u = h, x_u\right)\pi_h^{(m)} \tag{4}$$

where $\mathcal{L}_u\left(\varphi^{(m)}|z_u = h, x_u\right)$ denotes the likelihood of user u given that they are in group h.

2.2 The M-step

We look to compute

$$\psi^{(m+1)} = \arg\max_{\psi} \sum_{u=1}^{U}\sum_{h=1}^{H} \gamma_{uh}^{(m)} \left\{\log \pi_h + \ell_u(\varphi_h|z_u = h, x_u)\right\}$$

with $\ell_u(\varphi_h|z_u = h, x_u) = \sum_{p=1}^{P} \ell_{up}(\varphi_h|z_u = h, x_u)$ being the loglikelihood of user u. We update π trivially as

$$\pi_h^{(m+1)} = \frac{1}{N}\sum_{i=1}^{N} \gamma_{ih}^{(m)} \tag{5}$$

The necessary derivatives for the gradient ascent update can be shown to be

$$\frac{\partial}{\partial \mu_{p'h'}}\ell_{up}(\phi|z_u = h) = \delta_{pp'}\delta_{hh'}\left(\sum_{k=1}^{M_{up}} \frac{1}{\mu_{ph} + \sum_{j=1}^{P}\beta_{jph}A_h^{u,p,j}(N_{uj}(t_{upk}))} - T\right) \tag{6}$$

$$\begin{aligned}\frac{\partial}{\partial \beta_{j'p'h'}}\ell_{up}(\phi|z_u = h) = \delta_{pp'}\delta_{hh'}\Bigg(&\sum_{k=1}^{M_{up}} \frac{A_h^{u,p,j'}(N_{uj'}(t_{upk}))}{\mu_{ph} + \sum_{j=1}^{P}\beta_{jph}A_h^{u,p,j}(N_{uj}(t_{upk}))} \\ &- \theta_{j'ph}^{-1}\sum_{t_{uj'\ell}<T}\left(1 - \exp\left\{-\theta_{j'ph}(T - t_{uj'\ell})\right\}\right)\Bigg)\end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\partial}{\partial \theta_{j'p'h'}}\ell_{up}(\phi|z_u = h) = \delta_{pp'}\delta_{hh'}\Bigg(&\beta_{j'ph}\sum_{k=1}^{M_{up}} \frac{\tilde{A}_h^{u,p,j'}(N_{uj'}(t_{upk})) - t_{upk}A_h^{u,p,j'}(N_{uj'}(t_{upk}))}{\mu_{ph} + \sum_{j=1}^{P}\beta_{jph}A_h^{u,p,j}(N_{uj}(t_{upk}))} \\ &+ \beta_{j'ph}\theta_{j'ph}^{-2}\sum_{t_{uj'\ell}<T}\left(1 - \exp\left\{-\theta_{j'ph}(T - t_{uj'\ell})\right\}\right) \\ &- \beta_{j'ph}\theta_{j'ph}^{-1}\sum_{t_{uj'\ell}<T}(T - t_{uj'\ell})\exp\left\{-\theta_{j'ph}(T - t_{uj'\ell})\right\}\Bigg)\end{aligned} \tag{8}$$

where

$$\tilde{A}_h^{u,p,j}(k) = \sum_{t_{uj\ell}<t_{upk}} t_{uj\ell}\exp\left\{-\theta_{jph}(t_{upk} - t_{uj\ell})\right\}$$

with $\tilde{A}_h^{u,p,j}(0) = 0$. We have the following recursions:

$$A_h^{u,p,j}(k) = \exp\left\{-\theta_{jph}(t_{upk} - t_{up(k-1)})\right\}A_h^{u,p,j}(k-1) + \sum_{t_{up(k-1)}\le t_{uj\ell}<t_{upk}} \exp\left\{-\theta_{jph}(t_{upk} - t_{uj\ell})\right\} \tag{9}$$

$$\tilde{A}_h^{u,p,j}(k) = \exp\left\{-\theta_{jph}(t_{upk} - t_{up(k-1)})\right\}\tilde{A}_h^{u,p,j}(k-1) + \sum_{t_{up(k-1)}\le t_{uj\ell}<t_{upk}} t_{uj\ell}\exp\left\{-\theta_{jph}(t_{upk} - t_{uj\ell})\right\} \tag{10}$$

which is what allows us to run our algorithm in a computationally feasible time.

2.3 Reparameterisation

We need to put some constraints on our gradient ascent scheme to ensure that our parameter conditions are satisfied. We have to constrain $\mu_{ph}, \beta_{jph}, \theta_{jph} > 0$ and further that $\theta_{jph} > \beta_{jph}$ for all j, p, h combinations. To ensure these are satisfied, we make the following transformation

$$\begin{aligned}
\mu_{ph} &= \exp \tilde{\mu}_{ph} \\
\beta_{jph} &= \exp \tilde{\beta}_{jph} \\
\theta_{jph} &= \exp \tilde{\beta}_{jph} + \exp \tilde{\theta}_{jph}
\end{aligned}$$

This gives the following relation between our derivatives with repsect to the original and transformed parameters:

$$\begin{aligned}
\frac{\partial}{\partial \tilde{\mu}_{ph}} &= \exp \tilde{\mu}_{ph} \frac{\partial}{\partial \mu_{ph}} \\
\frac{\partial}{\partial \tilde{\beta}_{jph}} &= \exp \tilde{\beta}_{jph} \left(\frac{\partial}{\partial \beta_{jph}} + \frac{\partial}{\partial \theta_{jph}} \right) \\
\frac{\partial}{\partial \tilde{\theta}_{jph}} &= \exp \tilde{\theta}_{jph} \frac{\partial}{\partial \theta_{jph}}
\end{aligned}$$

which is needed for the update. We can then use the forms of the derivatives given in Equations (6) - (8) to compute the derivatives under the reparameterisation. It is these forms that we will use to run the EM algorithm. Parameters are inputted to each step of the algorithm in the reparameterised form, but the output is in the original parameterisation.

3 The Code

3.1 The `src` folder

Within the `src` folder are three python files (plus an empty `__init__.py` file). These function in the following way:

- `data_processor.py`: this file contains a basic class for processing the inputted data. This processing is run during the implementation and the user need not interact with this file.
- `EM_update_reparam.py`: this file contains all code pertaining to the actual implementation of the EM-update. The class `EM` has the functionality to run the full update, but there are a series of helper functions below that compute the values described in the previous section, such as the loglikelihood, derivatives etc. Again, this is run in the `main.py` file and the user does not need to interact with this file.
- `grouped_graphical_hawkes.py`: this script is for producing simulated output, and is not relevant to the inference procedure.

3.2 The `main.py` file

This is the entry point for running the EM-algorithm. The user runs the `main.py` function and must pass in a series of arguments:

- `data_file_path` is a string giving the relative directory of the dataset.
- `N_EM_its` is the number of runs of the EM algorithm you would like to run. The user is advised to start at 100 and assess parameter convergence.

- `N_grad_steps` is the number of gradient ascent steps to take on each run of the M-step of the algorithm. A suitable value to start with is between 5 and 10.
- `P, U, H` are the number of processes, users and assumed groups. Note that the algorithm currently does not have functionality to select the number of groups itself.
- `T_max` is the upper time limit of the observation window, where we assum data is observed on $[0, \texttt{T\_max}]$.

The data set that the user passes in is should be a `csv`. This should contain three columns: `Arrival_Times`, `Arrival_Process` and `User`, which should be of types `float`, `int` and `int`, respectively. Processes should be enumerated from 0 to $\texttt{P} - 1$, and users from 0 to $\texttt{U} - 1$. If there are no arrivals for a user to a process, the data processing step will handle this.

The output of the function is `psi` and `gammas`. `psi` is a `numpy.array` of shape `(H,1+P+2*P**2)`. We have

$$\begin{aligned}
\texttt{psi[h,0]} &= \pi_h \\
\texttt{psi[1:(P+1)]} &= (\mu_{1h}, \cdots, \mu_{Ph}) \\
\texttt{psi[(P+1):(P+1+P**2)]} &= (\beta_{11h}, \cdots, \beta_{1Ph}, \beta_{21h}, \cdots, \beta_{2Ph}, \cdots, \beta_{PPh}) \\
\texttt{psi[(P+1+P**2):]} &= (\theta_{11h}, \cdots, \theta_{1Ph}, \theta_{21h}, \cdots, \theta_{2Ph}, \cdots, \theta_{PPh}) \\
\texttt{gammas[u,h]} &= \gamma_{uh}
\end{aligned}$$

Finally, the user is advised to add a method for saving `psi` and `gammas` to the end of the `main.py` script.