PoPPy: A **Po**int **P**rocess **Py**Torch Toolbox

Hongteng Xu

Infinia ML, Inc. and Duke University

Oct. 18, 2018

Temporal Point Processes

- **Event sequence:** $S = \{(t_i, c_i)\}_{i=1}^{I}$, $c_i \in \mathcal{C}$.
- **Counting processes:** $N = \{N_c(t)\}_{c=1}^{C}$. $N_c(t)$ is the number of type-c events occurring till time t.
- **Intensity function:** The expected instantaneous happening rate of type-c events given the history.

$$\lambda_c(t) = \frac{\mathbb{E}[dN_c(t)|\mathcal{H}_t]}{dt},\ \mathcal{H}_t = \{(t_i, c_i)|t_i < t, c_i \in \mathcal{C}\}.$$

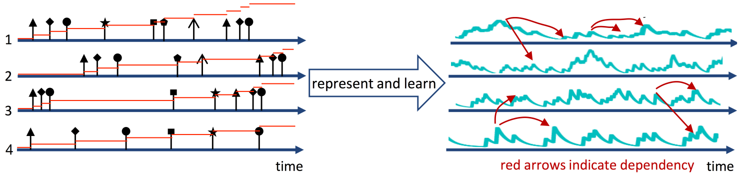


Figure 1: Event sequences and intensity functions.

Hawkes Processes

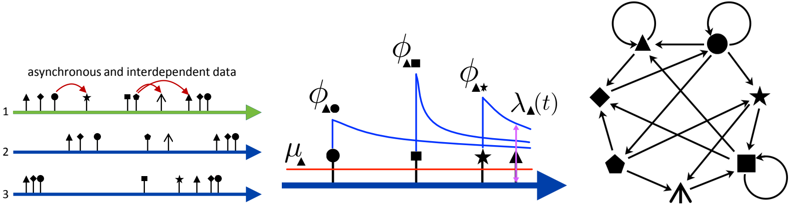
Hawkes Process has a particular form of intensity:

$$\begin{aligned}\lambda_c(t) &= \mu_c + \sum_{c'=1}^{C} \int_0^t \phi_{cc'}(t-s)dN_{c'}(s) \\ &= \mu_c + \sum_{(t_i,c_i)\in\mathcal{H}_t} \alpha_{cc_i}\kappa(-\delta_{cc_i}(t-t_i)).\end{aligned} \quad (1)$$

- $\boldsymbol{\mu} = [\mu_c]$: the exogenous intensity of the system.
- $\mathbf{\Phi} = [\phi_{cc'}(t)]$: the endogenous impact of the type-c events on the type-c' events.
- $\sum_{(t_i,c_i)\in\mathcal{H}_t} \phi_{cc_i}(t-t_i)$: the accumulated endogenous intensity caused by history.
- $\alpha_{cc'}$ is infectivity, and $\kappa(t)$ is decay kernel.

Hawkes Processes

It has good interpretability, and is highly correlated to Granger causality graph.



(a) Observations (b) Intensity per entity (c) Granger causality

Scene	Entities	Sequences	Task
Patient admission	Diseases	Patients' admission records	Disease network
Job hopping	Companies	LinkedIn users' job history	Talent flow
Online shopping	Items	Buying/rating behaviors	Recommendation

Motivations

Build a powerful point process toolbox...

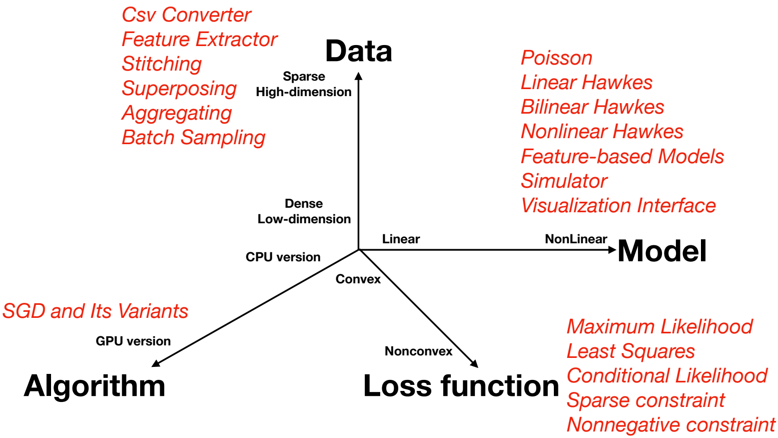
- Generalize Hawkes process to enhance the capacity of the model.
- Overcome the drawbacks of existing works.
- Fill the gap between stochastic process models and practical applications.
- Summarize my work in past 4 years.

Goals

The proposed toolbox should have the following features:

- **Rich Functionality:** data operations, learning, prediction, simulation, visualization, ...
- **Explicit Intrepretability:** Granger causality analysis, interpretable parameters, ...
- **High Flexibility:** modular design of model, multiple loss functions, regularizers, support numerical and categorical features, ...
- **High Speed:** low computational complexity, support GPU
- **High Scalability:** provide a general framework to design arbitrary PP models.

Current Stage: What can PoPPy do?



Data Structure

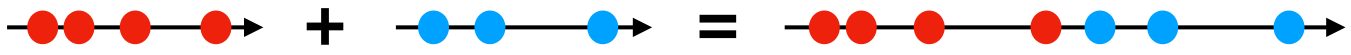
- Construct event sequences from csv files.
- Extract and vectorize numerical and/or categorical features from csv files.
- Each event contains $\{t,\ c,\ \boldsymbol{f}_c,\ \boldsymbol{f}_s\}$.

```
database = {'event_features': None or (C, De) float array of event's static features,
                          C is the number of event types.
            'type2idx': a Dict = {'event_name': event_index}
            'idx2type': a Dict = {event_index: 'event_name'}
            'seq2idx': a Dict = {'seq_name': seq_index}
            'idx2seq': a Dict = {seq_index: 'seq_name'}
            'sequences': a List  = {seq_1, seq_2, ..., seq_N}.
            }

For the i-th sequence:
seq_i = {'times': (N,) float array of timestamps, N is the number of events.
         'events': (N,) int array of event types.
         'seq_feature': None or (Ds,) float array of sequence's static feature.
         't_start': a float number indicating the start timestamp of the sequence.
         't_stop': a float number indicating the stop timestamp of the sequence.
         'label': None or int/float number indicating the labels of the sequence
         }
```

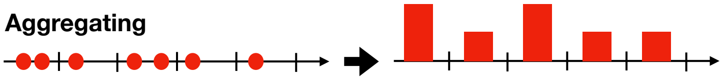
Special Data Operations

Stitching (random or feature-based)

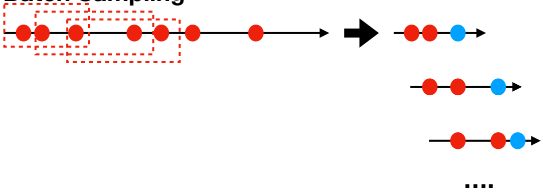


Superposing (random or feature-based)

Aggregating



Batch Sampling



Model

Intensity function:

$$\lambda_c(t) = g_\lambda\left(\mu(c, \boldsymbol{f}_c, \boldsymbol{f}_s) + \sum_{t_i < t} \phi(t, t_i, c, c_i, \boldsymbol{f}_c, \boldsymbol{f}_{c_i})\right) \tag{2}$$

Exogenous Intensity:

$$\mu(c, \boldsymbol{f}_c, \boldsymbol{f}_s) = \begin{cases} g_\mu(\mu_c), \\ g_\mu(\boldsymbol{w}_c^\top \boldsymbol{f}_s), \\ g_\mu(\boldsymbol{f}_c^\top \boldsymbol{W} \boldsymbol{f}_s), \\ NN(c, \boldsymbol{f}_c, \boldsymbol{f}_s). \end{cases} \tag{3}$$

Model

Endogenous Impact

$$\phi(t, t_i, c, c_i, \boldsymbol{f}_c, \boldsymbol{f}_{c_i}) = \sum_{m=1}^{M} \alpha_m(c, c_i, \boldsymbol{f}_c, \boldsymbol{f}_{c_i}) \kappa_m(t - t_i). \tag{4}$$

Infectivity

$$\alpha_m(c, c_i, \boldsymbol{f}_c, \boldsymbol{f}_{c_i}) = \begin{cases} g_\alpha(\alpha_{cc_i m}), \\ g_\alpha(\boldsymbol{u}_{c,m}^\top \boldsymbol{v}_{c_i,m}), \\ g_\alpha(\boldsymbol{w}_{c,m}^\top \boldsymbol{f}_{c_i}), \\ g_\alpha(\boldsymbol{f}_c^\top \boldsymbol{W}_m \boldsymbol{f}_{c_i}), \\ NN(c, c_i, \boldsymbol{f}_c, \boldsymbol{f}_{c_i}). \end{cases} \tag{5}$$

Model

Decay Kernel

$$\kappa_m(t) = \begin{cases} \mathit{Exp}, \\ \mathit{Rayleigh}, \\ \mathit{PowerLaw}, \\ \mathit{Gate}, \\ \mathit{Gaussian}, \\ \mathit{GMM}. \end{cases} \tag{6}$$

Activation functions $g_\lambda, g_\mu, g_\alpha$ can be identity, ReLU, or softplus functions.

Loss Function

Maximum Likelihood Estimation

$$L(\theta) = -\sum_{i \in \mathcal{D}} \left(\log \lambda_{c_i}(t_i) - \sum_{c \in \mathcal{C}} \int_{t_{i-1}}^{t_i} \lambda_c(s) ds \right). \tag{7}$$

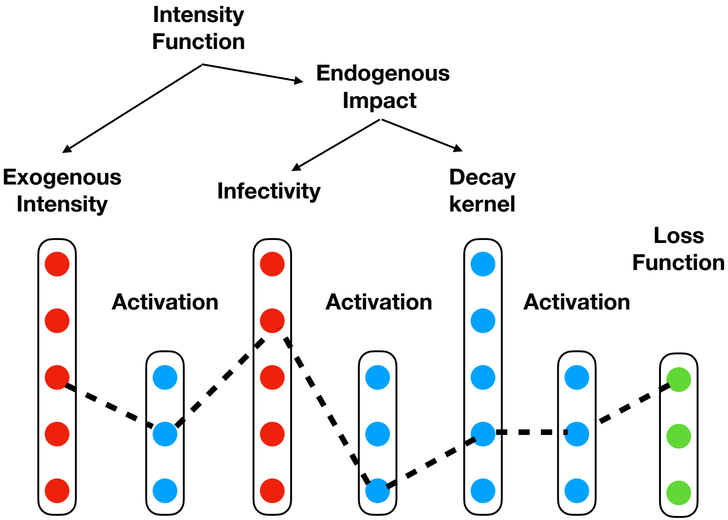
Least-Square Estimation

$$L(\theta) = \sum_{i \in \mathcal{D}} \left\| \int_{t_{i-1}}^{t_i} \boldsymbol{\lambda}(s) ds - \mathbf{1}_{c_i} \right\|_2^2. \tag{8}$$

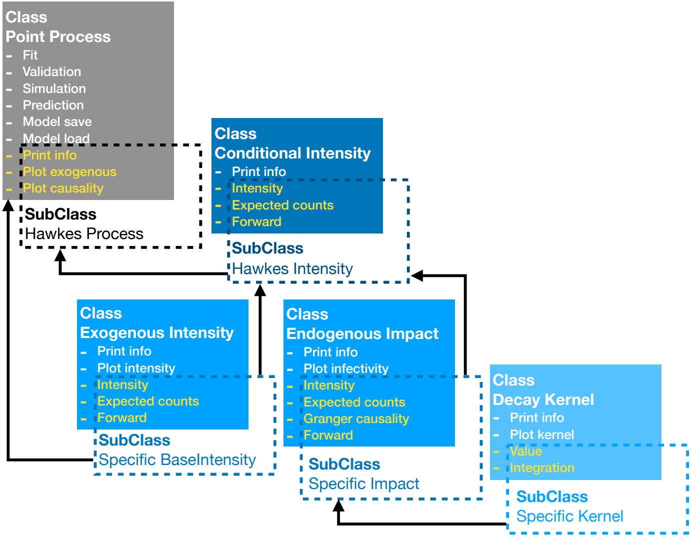
Conditional Likelihood Estimation

$$L(\theta) = -\sum_{i \in \mathcal{D}} \log p(c_i | t_i, \mathcal{H}_i) = -\sum_{i \in \mathcal{D}} \log softmax \left(\int_{t_{i-1}}^{t_i} \boldsymbol{\lambda}(s) ds \right). \tag{9}$$

Possible Models



Code Framework



- Besides the models above, you can define your own models as long as the IO satisfies required format.

Algorithm

With the help of PyTorch...

- Most of SGD algorithms like Adam are applicable to learn the models defined above.
- Both CPU- and GPU-based computations are applicable
- Nonnegative and sparsity constraints can be imposed to the models.
- Tricks: although all optimizers are applicable, but generally Adam provides the most stable performance.

Future work

Short-term tasks in next 3 months

- Build attention-based point process model.
- Learn point process from aggregate observations.
- Integrate more my previous work, e.g., learning mixture model of point processes.
- Continue to do unit test.
- Documentation.
- Add more application examples, e.g., finance, healthcare, social network ...

Long-term tasks

- AutoML for Point Process: configure hyperparameters (i.e., the bandwith and the number of decay kernel) automatically?
- Extend to spatial-temporal point process model?