# Self-Organized Hawkes Processes

Shen Yuan[1], **Hongteng Xu**[2]

[1]University of Electronic Science and Technology of China,
[2]Gaoling School of Artificial Intelligence, Renmin University of China

- Background
- Self-organized Hawkes Processes
- Reward-augmented Bandit Algorithm
- Experiments

# Event Sequences in real world

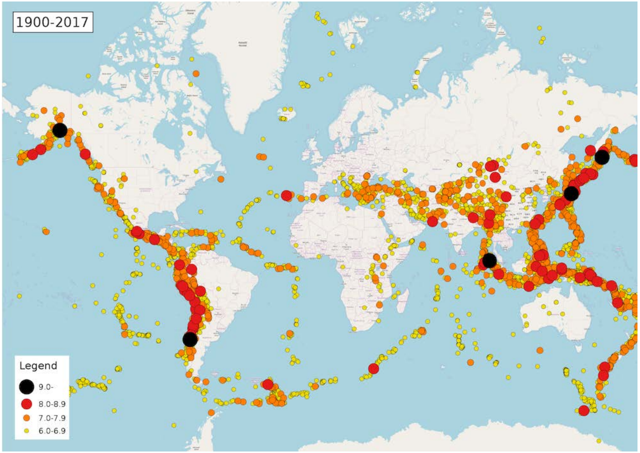



Figure 1: The locations and the intensities of the earthquakes from 1900 to 2017.

# Event Sequences in real world

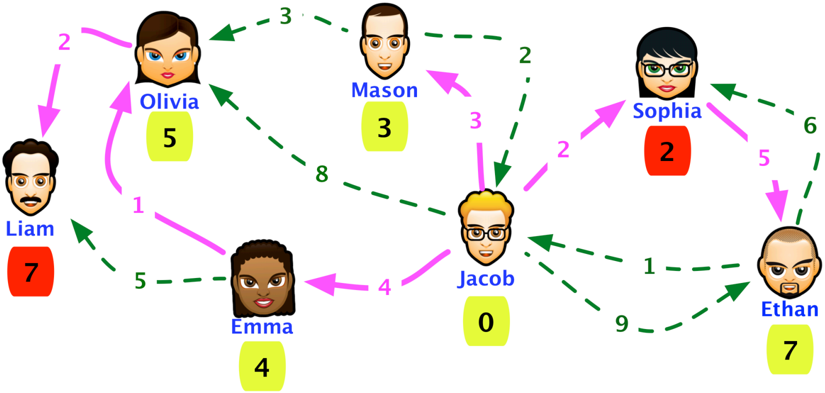



Figure 2: User behaviors on social networks.

# Event Sequences in real world

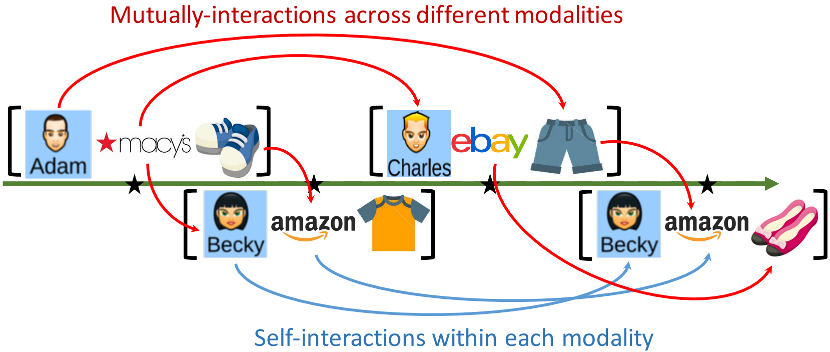
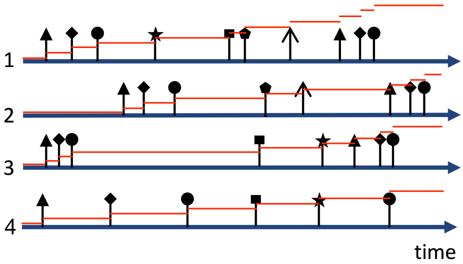



Figure 3: User behaviors on shopping websites.

# Event Sequences and Temporal Point Processes

**Event sequence:** $\{(t_i, c_i)\}_{i=1}^{I}$, $c_i \in \mathcal{C}$, or **Counting process:** $N(t) = \{N_c(t)\}_{c\in\mathcal{C}}$.

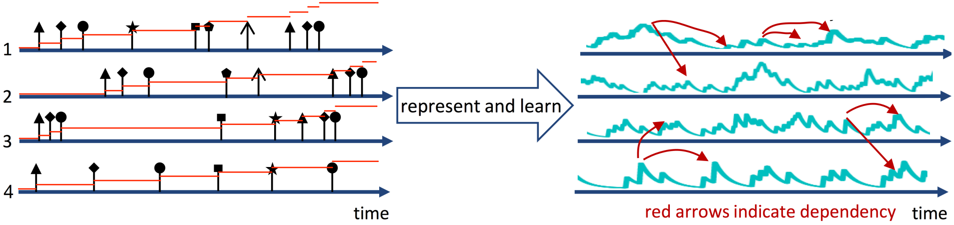



Where $t_i \in [0, T]$ mean time stamps and $c_i \in \mathcal{C} = \{1, \ldots, C\}$ mean event types.

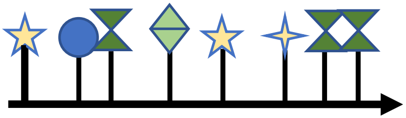
# Event Sequences and Temporal Point Processes

- **Intensity function:** The expected instantaneous happening rate of type-$c$ events given the history.

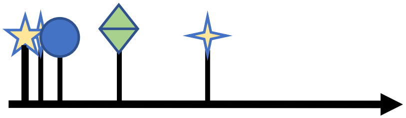
$$\lambda_c(t) = \frac{\mathbb{E}[dN_c(t)|\mathcal{H}_t]}{dt}, \quad \mathcal{H}_t = \{(t_i, c_i)|t_i < t, c_i \in \mathcal{C}\}.$$



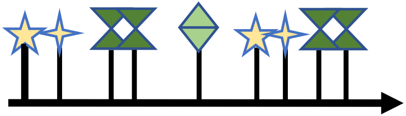
# Classical Models of Point Process



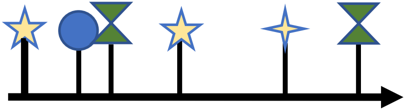
(a) **Poisson**: $\lambda^*(t) = \mu$



(b) **Terminating**: $\lambda^* = g^*(t)(1 - N(t))$



(c) **Hawkes**: $\lambda^*(t) = \mu + \alpha \sum \kappa(t - t_i)$



(d) **Self-correcting**:$\lambda^*(t) = e^{\mu t - \sum \alpha}$

# Hawkes Process

- **Homogeneous Poisson process:**

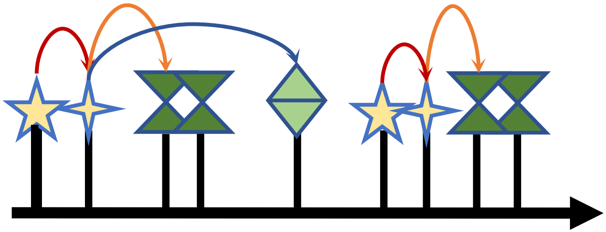
$$\lambda_c(t) = \mu_c$$

# Hawkes Process

- **Homogeneous Poisson process:**

$$\lambda_c(t) = \mu_c$$

Poisson process is too simple...
Is there a temporal point process that could model the self- and mutually-triggering patterns?

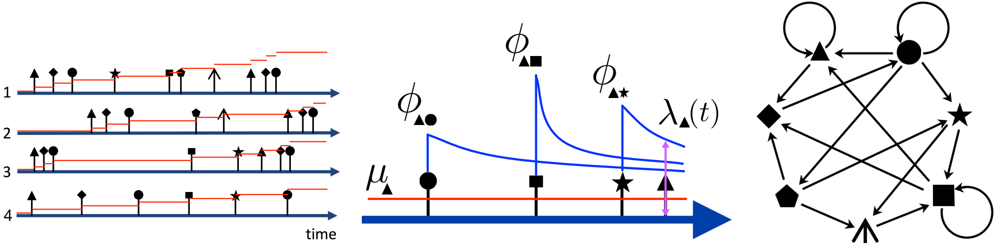


# Hawkes Processes

Hawkes Process $\mathrm{HP}_{\mathcal{C}}(\boldsymbol{\mu}, \boldsymbol{\Phi})$ models the triggering pattern between different events:

$$\lambda_c(t) = \underbrace{\mu_c}_{\text{base intensity}} + \sum_{(t_i,c_i)\in\mathcal{H}_t} \underbrace{\phi_{cc_i}(t - t_i)}_{\text{impact function}} \tag{1}$$

- $\boldsymbol{\mu} = [\mu_c]$: exogenous fluctuation of the system.
- $\boldsymbol{\Phi} = [\phi_{cc'}(t)]$: endogenous triggering pattern of type-$c'$ on type-$c$.




(e) Observations (f) Intensity per entity (g) Granger causality

# Hawkes Processes

- $\phi_{cc}(\cdot)$: the **self**-triggering pattern.
- $\phi_{cc'}(\cdot), c \neq c'$: the **mutually**-triggering pattern.

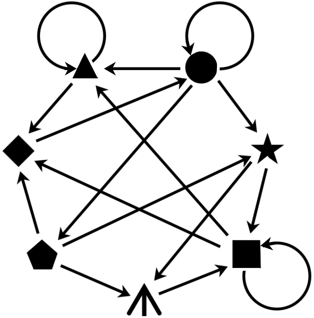


Figure 4: Granger causality

# Hawkes Processes

The **exponential impact function** is one of the widely-used impact function.

$$\phi_{cc'}(t - t') = a_{cc'} \exp(-w(t - t'))$$

where $\boldsymbol{A} = [a_{cc'}]$ is the trainable parameters of the impact functions and $w$ is the decay factor.

# Learning Method and Challenges

Learning **one** $\text{HP}_{\mathcal{C}}(\boldsymbol{\mu}, \boldsymbol{\Phi})$ from a set of sequences $\mathcal{N} = \{N^u\}_{u\in\mathcal{U}}$:

$$\begin{aligned}
&\min_{\boldsymbol{\theta}} -\underbrace{\log \mathcal{L}(\mathcal{N};\boldsymbol{\theta})}_{\text{log-likelihood}} + \underbrace{\gamma\mathcal{R}(\boldsymbol{\theta})}_{\text{regularizer}},\ \text{where } \boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Phi}\},\\
&\mathcal{L}(\mathcal{N};\lambda) = \prod\nolimits_{u\in\mathcal{U}}\Big(\prod\nolimits_{(c_i^u,t_i^u)\in N^u}\lambda_{c_i^u}(t_i^u)\times\exp\big(-\sum\nolimits_{c\in\mathcal{C}}\int_0^T\lambda_c^u(s)ds\big)\Big).
\end{aligned} \tag{2}$$

# Learning Method and Challenges

Learning **one** $\mathrm{HP}_{\mathcal{C}}(\boldsymbol{\mu}, \boldsymbol{\Phi})$ from a set of sequences $\mathcal{N} = \{N^u\}_{u\in\mathcal{U}}$:

$$\min_{\boldsymbol{\theta}} - \underbrace{\log \mathcal{L}(\mathcal{N}; \boldsymbol{\theta})}_{\text{log-likelihood}} + \underbrace{\gamma\mathcal{R}(\boldsymbol{\theta})}_{\text{regularizer}}, \text{ where } \boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Phi}\},$$
$$\mathcal{L}(\mathcal{N}; \lambda) = \prod_{u\in\mathcal{U}}\Big(\prod_{(c_i^u, t_i^u)\in N^u} \lambda_{c_i^u}(t_i^u) \times \exp\big(-\sum_{c\in\mathcal{C}} \int_0^T \lambda_c^u(s)ds\big)\Big). \tag{2}$$

- The real-world event sequences are often driven by multiple *local heterogeneous Hawkes processes* $\{\mathrm{HP}_{\mathcal{C}_u}(\boldsymbol{\mu}_u, \boldsymbol{\Phi}_u)\}_{u\in\mathcal{U}}$.
  - **Local:** $\mathcal{C}_u \subset \mathcal{C}$ and $|\mathcal{C}_u| \ll |\mathcal{C}|$.
  - **Heterogeneous:** $\mathcal{C}_u \neq \mathcal{C}_{u'}$ for $u \neq u'$.

# Learning Method and Challenges

Learning **one** $\mathrm{HP}_{\mathcal{C}}(\boldsymbol{\mu}, \boldsymbol{\Phi})$ from a set of sequences $\mathcal{N} = \{N^u\}_{u\in\mathcal{U}}$:

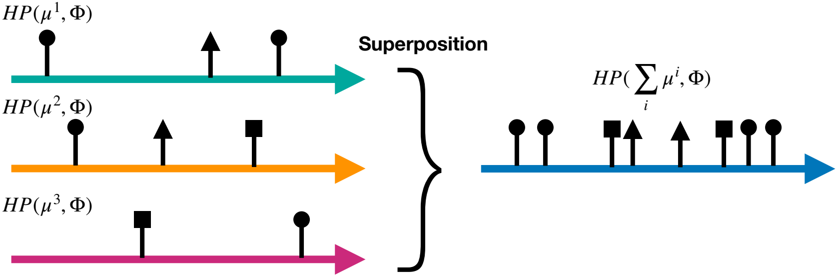
$$\min_{\boldsymbol{\theta}} -\underbrace{\log \mathcal{L}(\mathcal{N};\boldsymbol{\theta})}_{\text{log-likelihood}} + \underbrace{\gamma\mathcal{R}(\boldsymbol{\theta})}_{\text{regularizer}}, \text{ where } \boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Phi}\},$$
$$\mathcal{L}(\mathcal{N};\lambda) = \prod_{u\in\mathcal{U}}\Big(\prod_{(c_i^u,t_i^u)\in N^u} \lambda_{c_i^u}(t_i^u) \times \exp\big(-\sum_{c\in\mathcal{C}} \int_0^T \lambda_c^u(s)ds\big)\Big). \tag{2}$$

- The real-world event sequences are often driven by multiple *local heterogeneous Hawkes processes* $\{\mathrm{HP}_{\mathcal{C}_u}(\boldsymbol{\mu}_u, \boldsymbol{\Phi}_u)\}_{u\in\mathcal{U}}$.
  - **Local:** $\mathcal{C}_u \subset \mathcal{C}$ and $|\mathcal{C}_u| \ll |\mathcal{C}|$.
  - **Heterogeneous:** $\mathcal{C}_u \neq \mathcal{C}_{u'}$ for $u \neq u'$.
- However, the huge number of event types and extremely few observations make the learning of the HPs over-fitting even intractable.

- Background
- Self-organized Hawkes Processes
- Reward-augmented Bandit Algorithm
- Experiments

# Proposed Learning Task

**Superposition Property**. For a set of independent Hawkes processes with a shared infectivity matrix, i:e:, $\{N^u \sim \mathrm{HP}(\boldsymbol{\mu}^u, \boldsymbol{\Phi})\}_{u\in\mathcal{U}}$, the superposition of their sequences satisfies $\sum_{u\in\mathcal{U}} N^u \sim \mathrm{HP}(\sum_{u\in\mathcal{U}} \boldsymbol{\mu}^u, \boldsymbol{\Phi})$.[Xu et al. AISTATS'18]



# Proposed Learning Task

**Learning multiple local heterogeneous Hawkes processes from few short sequences.**

- **Key 1:** Leverage the **Superposition property** to increase the event density of each individual sequence.

# Proposed Learning Task

**Learning multiple local heterogeneous Hawkes processes from few short sequences.**

- **Key 1:** Leverage the **Superposition property** to increase the event density of each individual sequence.

The problem becomes **learning each individual HP from selective subset of sequences.**

$$\min_{\{\boldsymbol{\theta}^u\}_{u\in\mathcal{U}}} \min_{\mathcal{N}^u\subset\mathcal{N}} -\sum_{u\in\mathcal{U}} \frac{1}{|N^u \cup \mathcal{N}^u|} \log \mathcal{L}(N^u \cup \mathcal{N}^u; \boldsymbol{\theta}^u), \tag{3}$$

# Proposed Learning Task

**Learning multiple local heterogeneous Hawkes processes from few short sequences.**

- **Key 1:** Leverage the **Superposition property** to increase the event density of each individual sequence.

The problem becomes **learning each individual HP from selective subset of sequences.**

$$\min_{\{\boldsymbol{\theta}^u\}_{u\in\mathcal{U}}} \min_{\mathcal{N}^u \subset \mathcal{N}} -\sum_{u\in\mathcal{U}} \frac{1}{|N^u \cup \mathcal{N}^u|} \log \mathcal{L}(N^u \cup \mathcal{N}^u; \boldsymbol{\theta}^u), \tag{3}$$

Therefore, we need to design an algorithm to select a suitable subset for each HP.

- Background
- Self-organized Hawkes Processes
- Reward-augmented Bandit Algorithm
- Experiments

# Select Neighbors via A Bandit Algorithm

Intuitively, we hope that

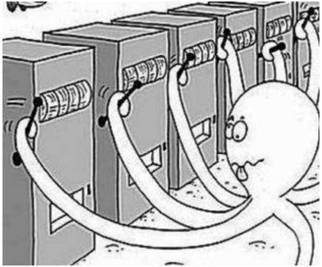
- the selected sequences are **similar** to the target sequence;
- the selected sequences own some **randomness** to avoid the over-fitting problem.

# Select Neighbors via A Bandit Algorithm

Intuitively, we hope that

- the selected sequences are **similar** to the target sequence;
- the selected sequences own some **randomness** to avoid the over-fitting problem.

we treat the selection of target sequence's neighbors as a **multi-armed bandit problem**.



# Select Neighbors via A Bandit Algorithm

- **Key 2:** Design a Reward-augmented Bandit Algorithm to **select neighbors for each sequence in the training phase.**

# Select Neighbors via A Bandit Algorithm

- **Key 2:** Design a Reward-augmented Bandit Algorithm to **select neighbors for each sequence in the training phase.**
- Formulate a benefit matrix $\boldsymbol{B} = [b_{u,v}] \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|}$, where $b_{u,v}$ represents the benefit from selecting the $v$-th sequence for the $u$-th Hawkes process.

# Select Neighbors via A Bandit Algorithm

- **Key 2:** Design a Reward-augmented Bandit Algorithm to **select neighbors for each sequence in the training phase.**
- Formulate a benefit matrix $\boldsymbol{B} = [b_{u,v}] \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|}$, where $b_{u,v}$ represents the benefit from selecting the $v$-th sequence for the $u$-th Hawkes process.

1. Initialize $b_{u,v} = \max_k d(N^u, N^k) - d(N^u, N^v)$ for $v \in \mathcal{U}$, where $d(N^u, N^k)$ is the optimal transport distance between different event sequences.

# Select Neighbors via A Bandit Algorithm

- **Key 2:** Design a Reward-augmented Bandit Algorithm to **select neighbors for each sequence in the training phase.**
- Formulate a benefit matrix $\boldsymbol{B} = [b_{u,v}] \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|}$, where $b_{u,v}$ represents the benefit from selecting the $v$-th sequence for the $u$-th Hawkes process.

1. Initialize $b_{u,v} = \max_k d(N^u, N^k) - d(N^u, N^v)$ for $v \in \mathcal{U}$, where $d(N^u, N^k)$ is the optimal transport distance between different event sequences.

2. Select neighbor sequences $\{s_v\}$ for each sequence $u$ by a bandit algorithm (*e.g.*, the Upper Confidence Bound method).

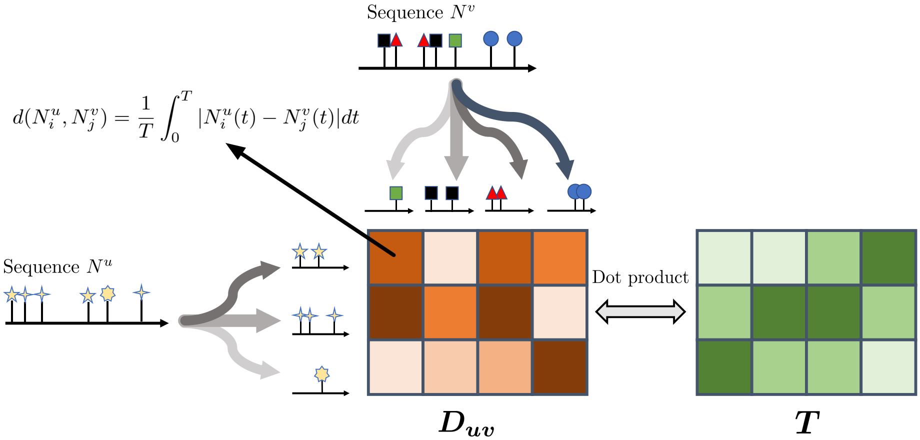
# Select Neighbors via A Bandit Algorithm

- **Key 2:** Design a Reward-augmented Bandit Algorithm to **select neighbors for each sequence in the training phase.**
- Formulate a benefit matrix $\boldsymbol{B} = [b_{u,v}] \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|}$, where $b_{u,v}$ represents the benefit from selecting the $v$-th sequence for the $u$-th Hawkes process.

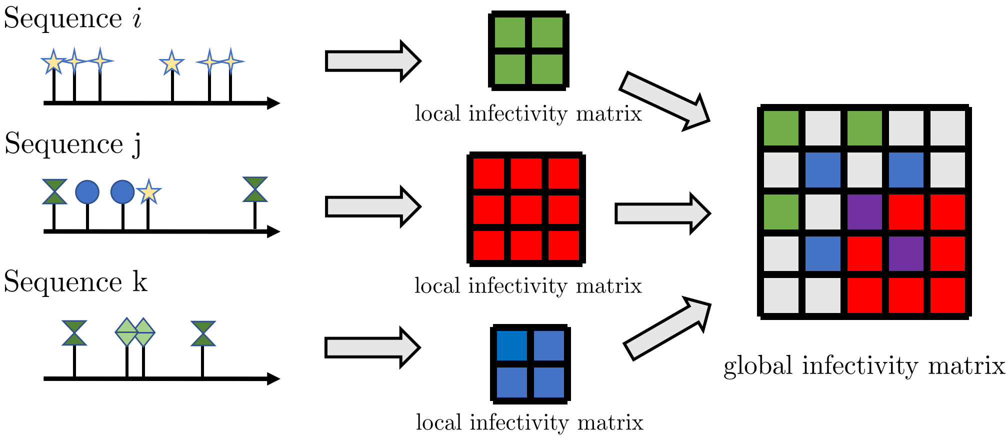
1. Initialize $b_{u,v} = \max_k d(N^u, N^k) - d(N^u, N^v)$ for $v \in \mathcal{U}$, where $d(N^u, N^k)$ is the optimal transport distance between different event sequences.
2. Select neighbor sequences $\{s_v\}$ for each sequence $u$ by a bandit algorithm (*e.g.*, the Upper Confidence Bound method).
3. Update $b_{u,v} = b_{u,v} + \alpha\mathcal{L}(N^{s_v}; \boldsymbol{\theta}^u)$.

# Optimal Transport Distance between Sequences

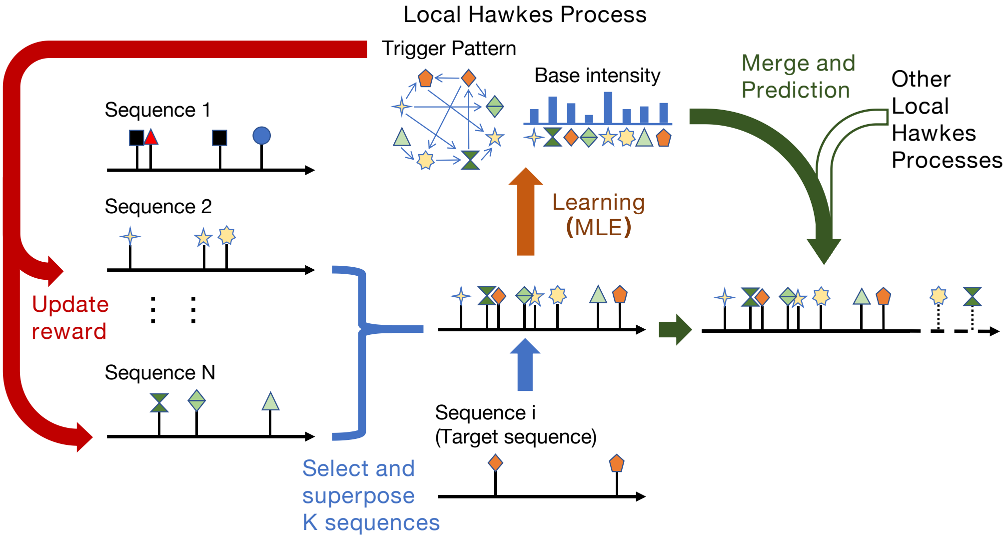
$$d(N^u, N^v) := \min_{\boldsymbol{T}\in\Pi(\frac{1}{|\mathcal{C}_u|}\mathbf{1}_{|\mathcal{C}_u|}, \frac{1}{|\mathcal{C}_v|}\mathbf{1}_{|\mathcal{C}_v|})} \sum_{i\in\mathcal{C}_u, j\in\mathcal{C}_v} T_{ij} d(N_i^u, N_j^v) = \min_{\boldsymbol{T}\in\Pi(\frac{1}{|\mathcal{C}_u|}\mathbf{1}_{|\mathcal{C}_u|}, \frac{1}{|\mathcal{C}_v|}\mathbf{1}_{|\mathcal{C}_v|})} \langle \boldsymbol{D}_{uv}, \boldsymbol{T} \rangle$$



# Merging learned Hawkes processes for exploration



# Self-organized Hawkes Processes



- Background
- Self-organized Hawkes Processes
- Reward-augmented Bandit Algorithm
- Experiments

# Experiments

We experiment on the **Amazon review dataset** containing product reviews from Amazon spanning May 1996 - July 2014.

# Experiments

We experiment on the **Amazon review dataset** containing product reviews from Amazon spanning May 1996 - July 2014.
We select those items with **more than** 40 reviews. Then, their users need to satisfy three conditions:

- the ratings they gave to these items are bigger than 4;
- there are **at most** 3 reviews spanning January 2014 - April 2014;
- there are **at least** 1 review from April 2014 to July 2014.

# Experiments

We experiment on the **Amazon review dataset** containing product reviews from Amazon spanning May 1996 - July 2014.
We select those items with **more than** 40 reviews. Then, their users need to satisfy three conditions:

- the ratings they gave to these items are bigger than 4;
- there are **at most** 3 reviews spanning January 2014 - April 2014;
- there are **at least** 1 review from April 2014 to July 2014.

Table 1: Statistics of our datasets

| Categories | Musical Instruments | Baby | Video Games | Garden | Instant Video |
|---|---|---|---|---|---|
| #Users | 471 | 1979 | 2142 | 1812 | 5948 |
| #Items | 678 | 2134 | 2104 | 2064 | 1344 |
| #Ratings | 1218 | 6070 | 6126 | 4976 | 15470 |

# Experiments

After learning the behaviors of users in 3 months, our model predicts their behaviors in the next 3 months. For each user, 10 items are recommended.

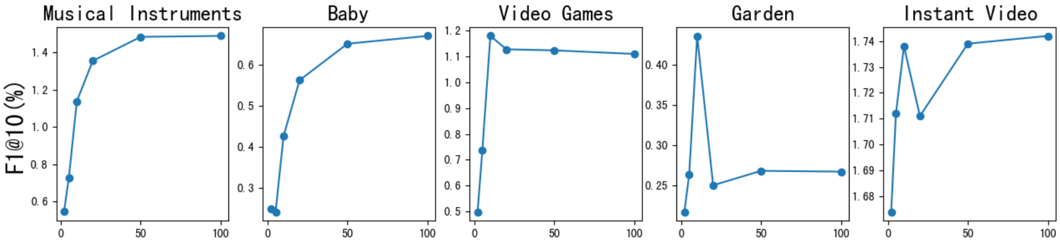
$$p(c|t+\Delta t, \mathcal{H}_t^{\mathcal{C}}) = \frac{\lambda_c(t+\Delta t)}{\sum_{c' \in \mathcal{C}} \lambda_{c'}(t+\Delta t)}. \tag{4}$$

# Experiments: Continuous-Time Sequential Recommendation

To prove the effect of the reward-augmented bandit algorithm, we test the **SHP** model that learn a single Hawkes process by randomly superpose event sequences.

| Data | Musical Instruments | | | Baby | | | Video Games | | | Garden | | | Instant Video | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| @10(%) | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| SVD | 0.106 | 1.061 | 0.193 | 0.121 | 0.735 | 0.207 | 0.065 | 0.498 | 0.113 | 0.055 | 0.395 | 0.094 | 0.126 | 1.052 | 0.221 |
| kNN | 0.382 | 2.671 | 0.649 | 0.389 | 2.513 | 0.638 | 0.661 | 4.915 | 1.163 | 0.237 | 1.751 | 0.405 | 0.817 | 6.901 | 1.435 |
| BPR | 0.467 | 3.750 | 0.811 | 0.389 | 2.469 | 0.635 | 0.658 | 4.864 | 1.112 | 0.110 | 0.762 | 0.185 | 0.859 | 7.049 | 1.503 |
| SLIM | 0.212 | 1.351 | 0.347 | 0.111 | 0.712 | 0.180 | 0.499 | 3.595 | 0.835 | 0.242 | 1.544 | 0.401 | **1.333** | **11.428** | **2.351** |
| FPMC | 0.594 | 4.193 | 1.006 | 0.283 | 1.912 | 0.470 | 0.556 | 3.799 | 0.927 | 0.171 | 1.117 | 0.285 | 0.931 | 7.413 | 1.622 |
| SHP | 0.361 | 2.406 | 0.604 | 0.258 | 1.734 | 0.432 | 0.317 | 2.037 | 0.525 | 0.199 | 1.350 | 0.331 | 0.933 | 7.406 | 1.623 |
| Ours | **0.658** | **5.149** | **1.138** | **0.389** | **2.640** | **0.651** | **0.700** | **5.108** | **1.180** | **0.248** | **1.801** | **0.436** | 0.999 | 7.911 | 1.738 |

# Experiments: Influence of the number of neighbors $K$



# Summary

- The self-organized Hawkes process (SOHP) model learns heterogeneous local Hawkes processes based on selective subsets of event sequences.
- A learning algorithm combining bandit algorithm and optimal transport is proposed.
- Apply SOHP model into sequential recommendation system, and achieve higher f1 scores than state-of-the-art methods.
- The code is available at `https://github.com/DaShenZi721/MHP`.