Sampling and parameter fitting with **Hawkes Processes**

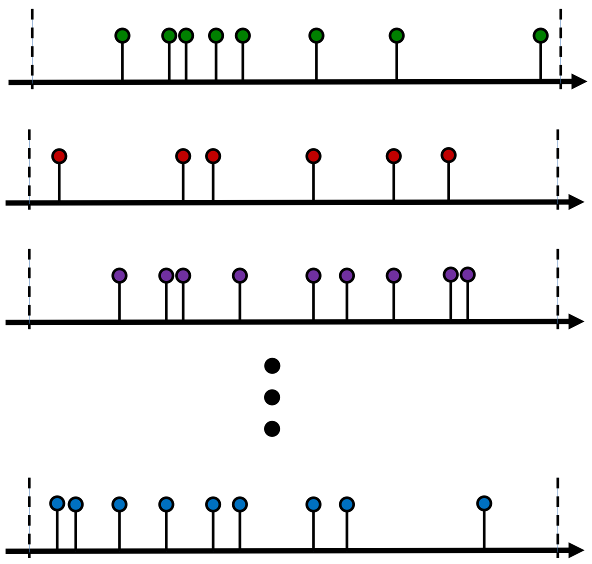
HUMAN-CENTERED MACHINE LEARNING

http://courses.mpi-sws.org/hcml-ws18/

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

Recap: How to fit and why sample?

Raw Data



Infer parameters (Learning)

- Parametrize intensity $\lambda^*_\theta(t)$
- Derive Log-likelihood:

$$\mathfrak{L}(\mathcal{H}(T);\theta) = \sum_{i=1}^{n} \log \lambda^*_\theta(t_i) - \int_0^T \lambda^*_\theta(\tau)\, d\tau,$$

- Maximum likelihood estimation:

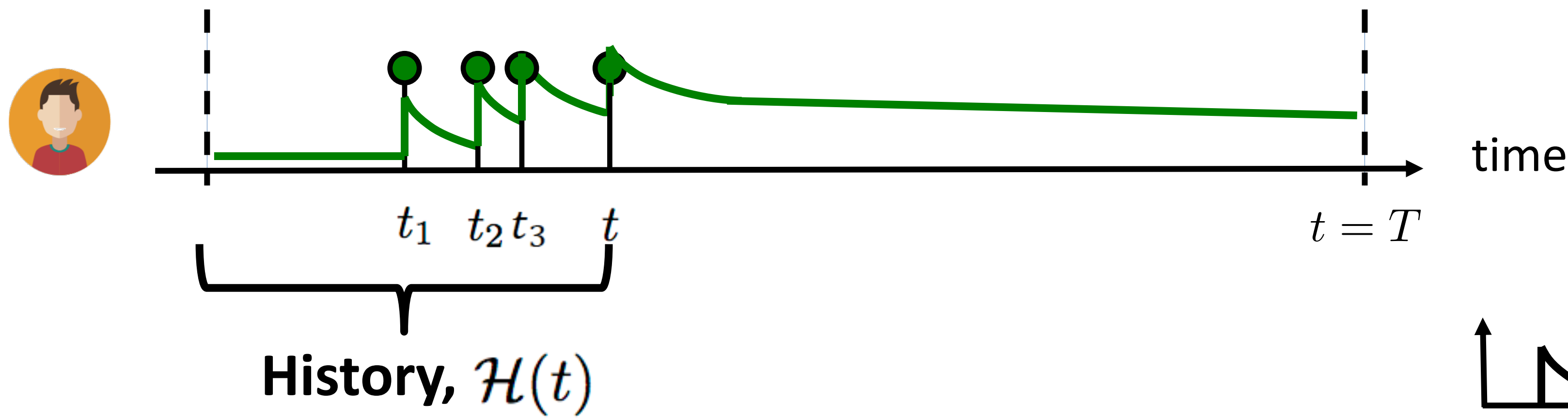
$$\hat{\theta} = \operatorname*{argmax}_{\theta} \mathfrak{L}(\mathcal{H}(T);\theta).$$

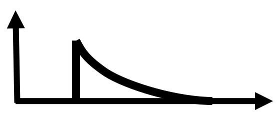
Sampling (Predicting)

- Event times drawn from: $\lambda^*_\theta(t) \quad \text{with } \theta = \hat{\theta}$
- Helps with:
  - Prediction
  - Model Checking
    - Sanity check
    - Gaining Intuition
  - **Simulator**
  - Summary statistics

We will first **sample** and then **fit**.

Recap: What are Hawkes processes?





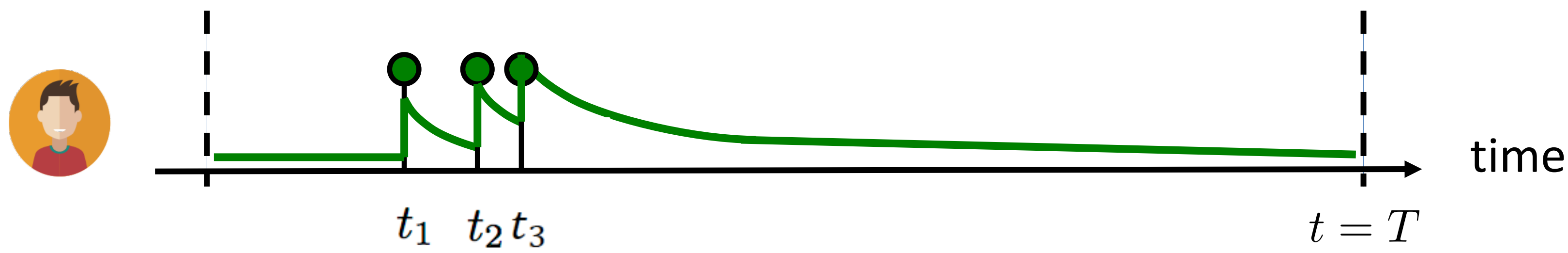
Intensity of self-exciting (or Hawkes) process:

$$\lambda^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}(t)} \kappa_\omega(t - t_i)$$

Observations:

1. **Clustered (or bursty) occurrence of events**
2. **Intensity is stochastic and history dependent**

Recap: How to fit a Hawkes process?



$$\mathfrak{L} = \lambda^*(t_1)\lambda^*(t_2)\,\lambda^*(t_3) \cdots\cdots \lambda^*(t_n)\ \exp\left(-\int_0^T \lambda^*(\tau)\,d\tau\right)$$

$$\lambda^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}(t)} \kappa_\omega(t - t_i)$$

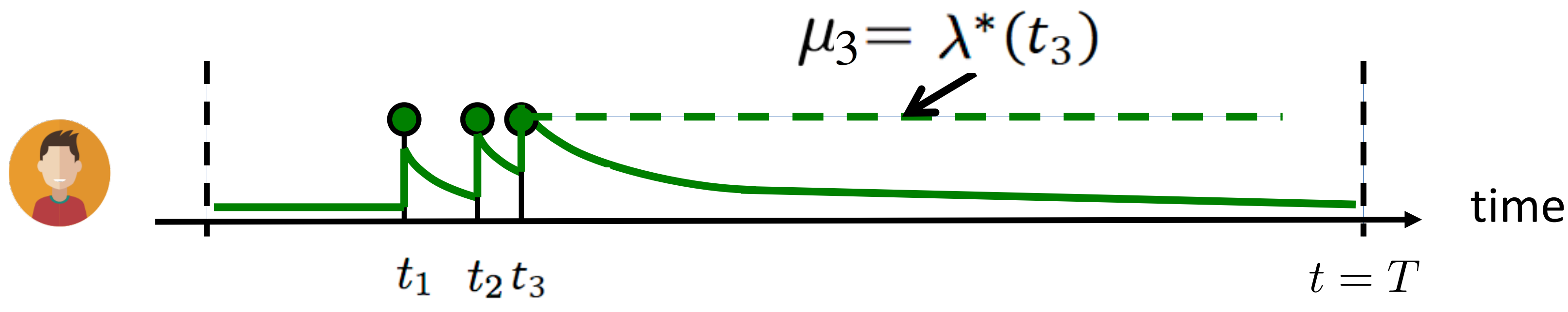
Maximum likelihood

$$\underset{\mu,\alpha}{\text{maximize}} \ \sum_{i=1}^{n} \log \lambda^*(t_i) - \int_0^T \lambda^*(\tau)\,d\tau$$

The max. likelihood is jointly convex in μ and α

(use CVX!)

Recap: How to sample from a Hawkes process



Thinning procedure (similar to rejection sampling):

1. **Sample** t **from Poisson process with intensity** μ_3

$$t \sim -\frac{1}{\mu_3}\log(1-u)+t_3$$

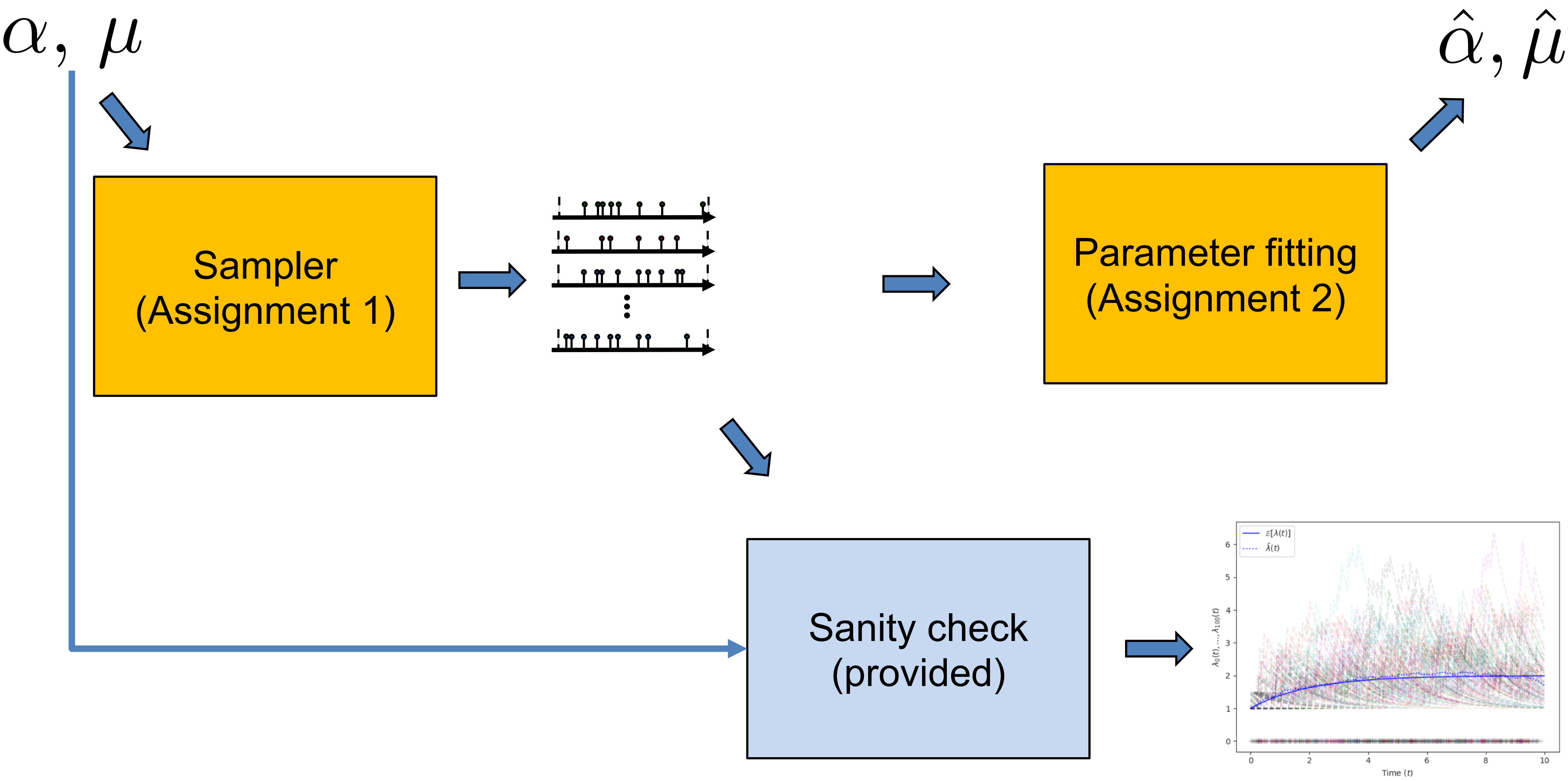
u: $Uniform(0,1)$

Inversion sampling

2. **Generate** $u_2 \sim Uniform(0,1)$
3. **Keep the sample if** $u_2 \leq \lambda^*(t) / \mu_3$

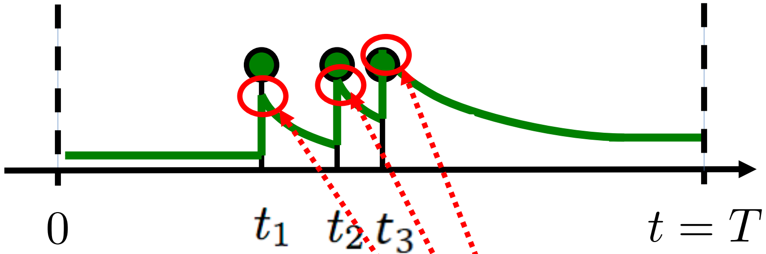
Keep sample with prob. $\lambda^*(t) / \mu_3$

Coding assignment overview



Sampling: Ogata's algorithm

Ogata's method of thinning



Algorithm 1: Hawkes Sampling using Thinning

Input: μ, α, ω (parameters), T (end time)
Output: $\{t_i\}$ (time of the events)
$i \leftarrow 1$; $t_0 \leftarrow 0$
repeat
  $\lambda_{\max} \leftarrow \lambda^*_{\mu,\alpha}(t^+_{i-1})$
  $t \leftarrow t_{i-1}$
  repeat
    $u_1 \leftarrow \mathrm{Unif}[0,1]$
    $t \leftarrow t - \log(1-u_1)/\lambda_{\max}$
    $u_2 \sim \mathrm{Unif}[0,1]$
  until $u_2 \leq \lambda^*_{\mu,\alpha}(t)/\lambda_{\max} \vee t > T$
  if $t < T$ **then**
    $t_i \leftarrow t$; $i \leftarrow i + 1$
  end if
until $t > T$
return $\{t_i\}_{[i \geq 1]}$

Drawing 1 sample with intensity $\lambda_{\max}$ (lines 8–9)

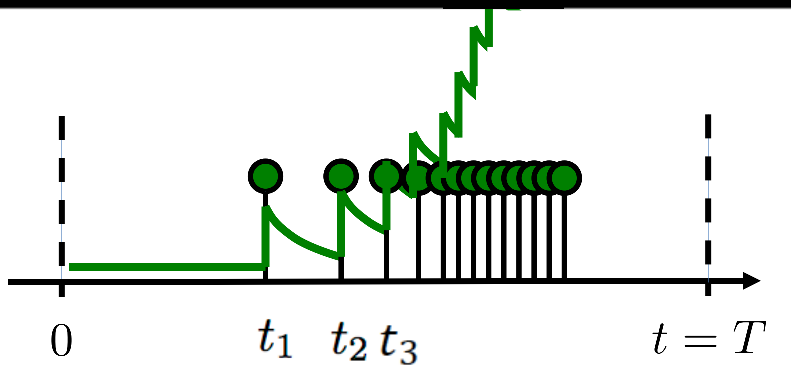
Accepting it with prob $\lambda^*_{\mu,\alpha}(t)/\lambda_{\max}$ (lines 10–11)

Sampling: Achtung I

Ogata's method of thinning

Algorithm 1: Hawkes Sampling using Thinning

Input: μ, α, ω (parameters), T (end time)
Output: $\{t_i\}$ (time of the events)
$i \leftarrow 1$; $t_0 \leftarrow 0$
repeat
  $\lambda_{\max} \leftarrow \lambda^*_{\mu,\alpha}(t^+_{i-1})$
  $t \leftarrow t_{i-1}$
  repeat
    $u_1 \leftarrow \text{Unif}[0,1]$
    $t \leftarrow t - \log(1 - u_1)/\lambda_{\max}$
    $u_2 \sim \text{Unif}[0,1]$
  until $u_2 \leq \lambda^*_{\mu,\alpha}(t)/\lambda_{\max} \;\vee\; t > T$
  if $t < T$ **then**
    $t_i \leftarrow t$; $i \leftarrow i + 1$
  end if
until $t > T$
return $\{t_i\}_{[i \geq 1]}$



Careful about exploding intensities!

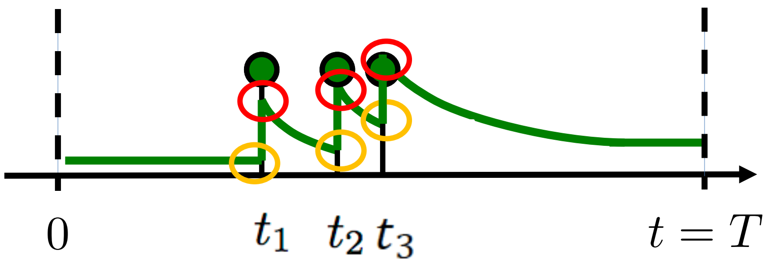
- Include a check on *i*.
- Ensure $\frac{\alpha}{\omega} < 1$

Sampling: Achtung II

Ogata's method of thinning

Algorithm 1: Hawkes Sampling using Thinning

1: **Input:** $\mu,\ \alpha,\ \omega$ (parameters), T (end time)
2: **Output:** $\{t_i\}$ (time of the events)
3: $i \leftarrow 1;\ t_0 \leftarrow 0$
4: **repeat**
5: $\quad \lambda_{\max} \leftarrow \lambda^*_{\mu,\alpha}(t^+_{i-1})$
6: $\quad t \leftarrow t_{i-1}$
7: $\quad$ **repeat**
8: $\quad\quad u_1 \leftarrow \text{Unif}[0,1]$
9: $\quad\quad t \leftarrow t - \log(1-u_1)/\lambda_{\max}$
10: $\quad\quad u_2 \sim \text{Unif}[0,1]$
11: $\quad$ **until** $u_2 \le \lambda^*_{\mu,\alpha}(t)/\lambda_{\max} \ \vee\ t > T$
12: $\quad$ **if** $t < T$ **then**
13: $\quad\quad t_i \leftarrow t;\ i \leftarrow i+1$
14: $\quad$ **end if**
15: **until** $t > T$
16: **return** $\{t_i\}_{[i \ge 1]}$

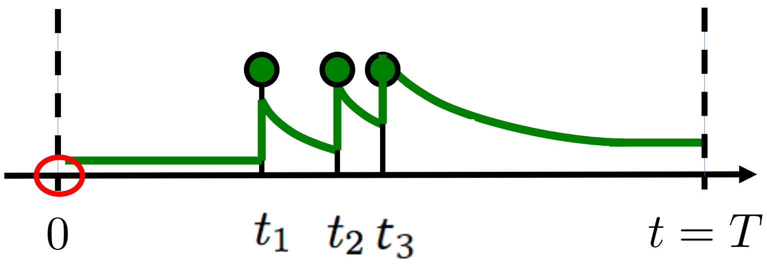


$$\lambda^*_{\mu,\alpha}(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}(t)} e^{-\omega(t-t_i)}$$

History up to
but not including
t.

Sampling: Achtung III

Ogata's method of thinning



Algorithm 1: Hawkes Sampling using Thinning

Input: μ, α, ω (parameters), T (end time)
Output: $\{t_i\}$ (time of the events)
$i \leftarrow 1$; $t_0 \leftarrow 0$
repeat
  $\lambda_{\max} \leftarrow \lambda^*_{\mu,\alpha}(t^+_{i-1})$
  $t \leftarrow t_{i-1}$
  repeat
    $u_1 \leftarrow \text{Unif}[0,1]$
    $t \leftarrow t - \log(1 - u_1)/\lambda_{\max}$
    $u_2 \sim \text{Unif}[0,1]$
  until $u_2 \leq \lambda^*_{\mu,\alpha}(t)/\lambda_{\max} \ \vee \ t > T$
  if $t < T$ **then**
    $t_i \leftarrow t$; $i \leftarrow i + 1$
  end if
until $t > T$
return $\{t_i\}_{[i \geq 1]}$

$t_0 = 0$ is not an actual sample!

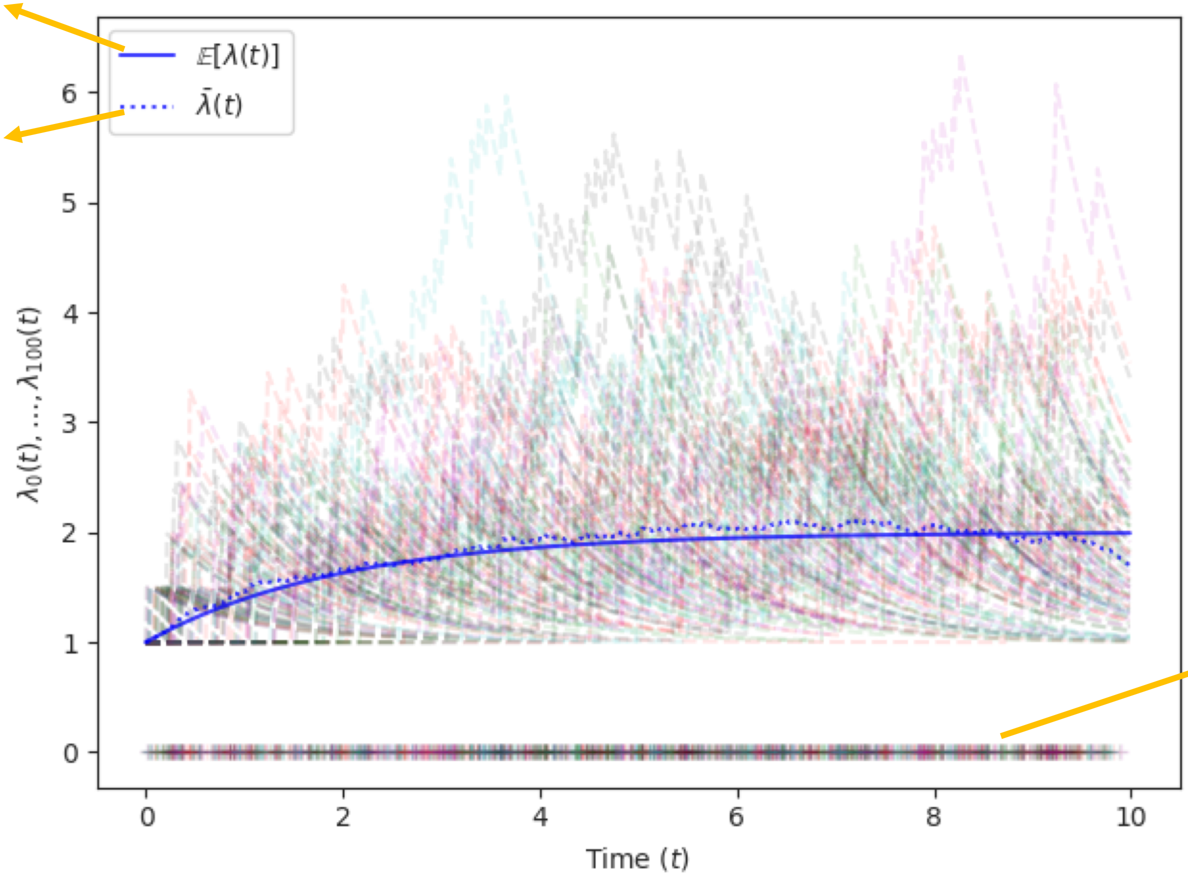
Sampling: Evaluation

```
python plot_hawkes.py 1.0 0.5 1.0 10 sampled-sequences.txt output-plot.png
```

$\mu, \ \alpha, \ \omega, \ T$

Theoretical value

Empirical average

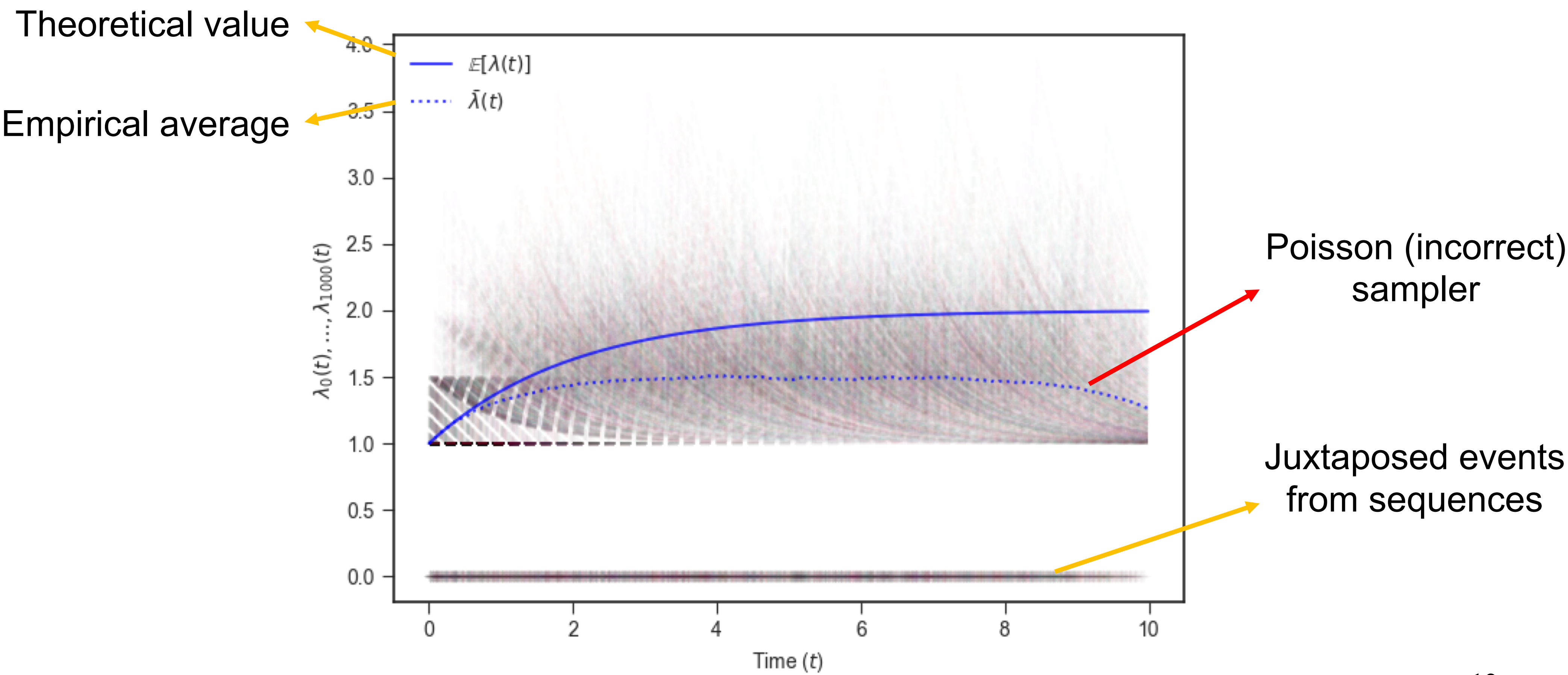


Juxtaposed events from sequences

Sampling: Evaluation

```
python plot_hawkes.py 1.0 0.5 1.0 10 sampled-sequences.txt output-plot.png
```

$\mu,\ \alpha,\ \omega,\ T$



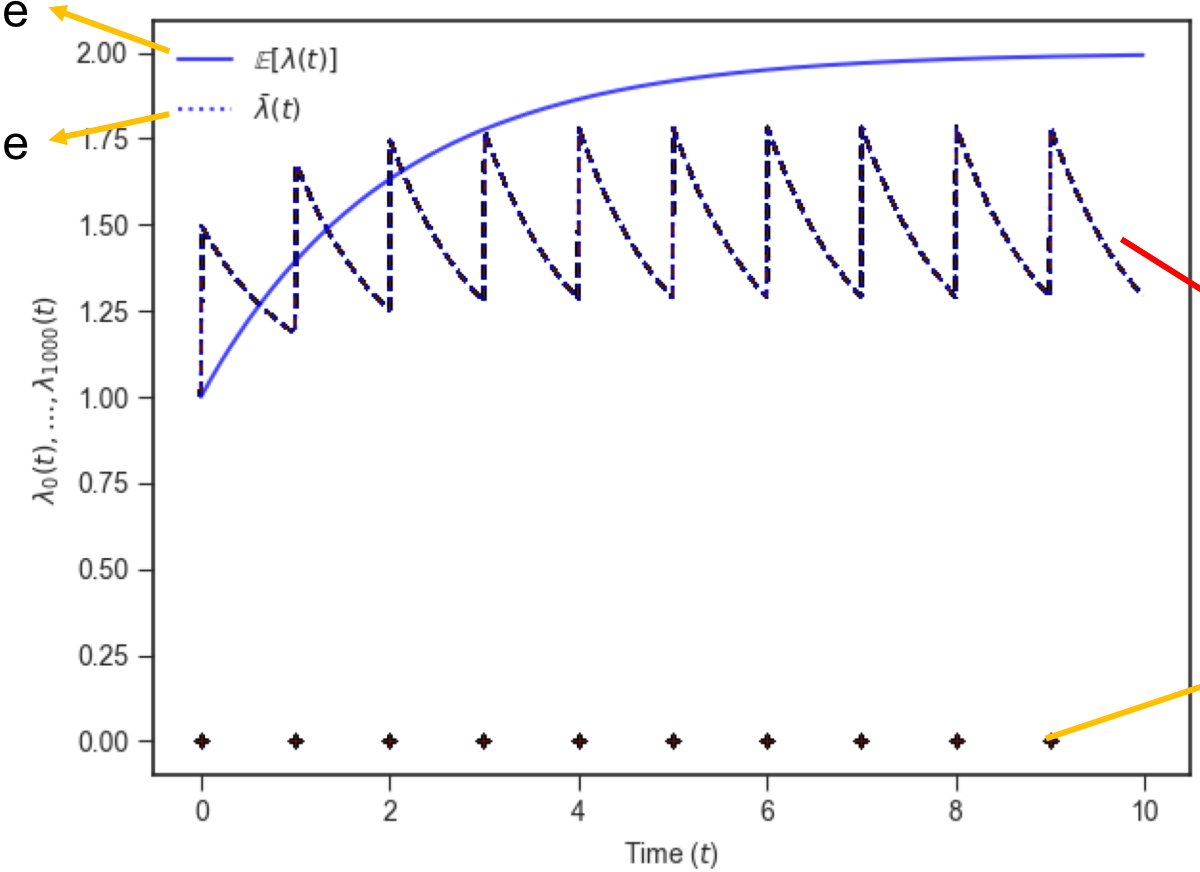
Sampling: Evaluation

```
python plot_hawkes.py 1.0 0.5 1.0 10 sampled-sequences.txt output-plot.png
```

$\mu, \ \alpha, \ \omega, \ T$

Theoretical value

Empirical average



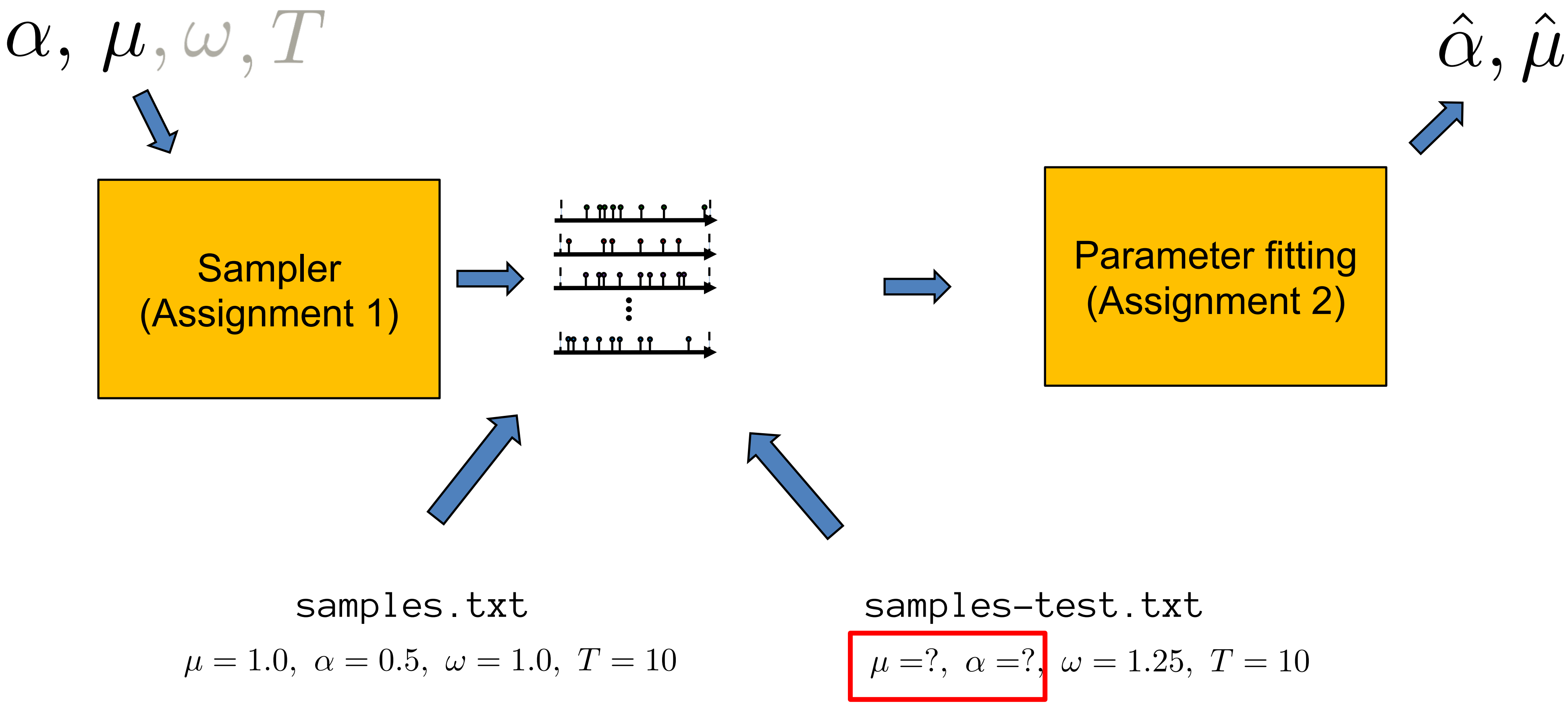
Deterministic (incorrect) sampler

Juxtaposed events from sequences

Live coding

- Show baseline + sanity check
- Show Poisson + sanity check

Parameter fitting: Problem setting



Parameter fitting: Method

Maximum likelihood estimation: $$\hat{\mu}, \hat{\alpha} = \underset{\mu,\alpha}{\operatorname{argmax}}\, \mathfrak{L}(\mathcal{H}(T))$$

$$\mathfrak{L}(\mathcal{H}(T)) = \sum_{t_i \in \mathcal{H}(T)} \log \lambda^*_{\mu,\alpha}(t_i) - \int_0^T \lambda^*_{\mu,\alpha}(\tau)\, d\tau$$

$$= \sum_{t_i \in \mathcal{H}(T)} \left[\log \left(\mu + \alpha \sum_{t_j \in \mathcal{H}(t_i)} e^{-\omega(t_i - t_j)} \right) - \alpha \int_{t_i}^T e^{-\omega(t - t_i)} dt \right] - \mu T$$

Nested sum

$\mathcal{O}(n^2)$

Can we do it faster for exponential kernels?

Parameter fitting: Using cvxpy

```
import cvxpy as cp
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A*x - b))
constraints = [0 <= x, x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by `prob.solve()`.
result = prob.solve()
# The optimal value for x is stored in `x.value`.
print(x.value)
```

$$\boldsymbol{A} = [a_{ij} \sim \mathcal{N}(0,1)]$$
$$\boldsymbol{b} = [b_i \sim \mathcal{N}(0,1)]$$

$$\underset{\boldsymbol{x}}{\text{minimize}} \; ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||^2$$
$$\text{s.t. } \forall i.\, 0 \leq x_i \leq 1$$

Parameter fitting: Using cvxpy

```
import cvxpy as cp
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
b = np.random.randn(m)

# Construct the problem.
x = cp.Variable(n)
objective = cp.Minimize(cp.sum_squares(A*x - b))
constraints = [0 <= x, x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by `prob.solve()`.
result = prob.solve()
# The optimal value for x is stored in `x.value`.
print(x.value)
```

$$\boldsymbol{A} = [a_{ij} \sim \mathcal{N}(0,1)]$$

$$\boldsymbol{b} = [b_i \sim \mathcal{N}(0,1)]$$

$$\underset{\boldsymbol{x}}{\text{minimize}} \; ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||^2$$

$$\text{s.t. } \forall i.\, 0 \leq x_i \leq 1$$

Change this to maximize the likelihood $\mathfrak{L}(\mathcal{H}(T))$

Live coding

- Show baseline + desired output
- Evaluation via sampling

Happy coding and holidays!

Questions?

- Drop me an e-mail at utkarshu@mpi-sws.org
- Skype: utkarsh.upadhyay