Dynamic Processes over Information Networks

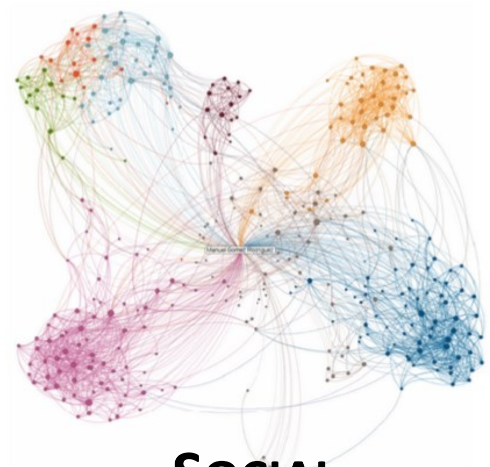
Representation, Modeling, Learning and Inference

Le Song

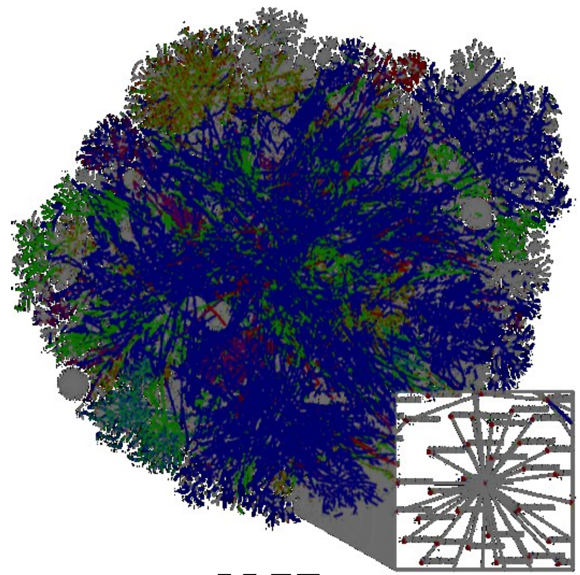
College of Computing

Georgia Institute of Technology

Network of things

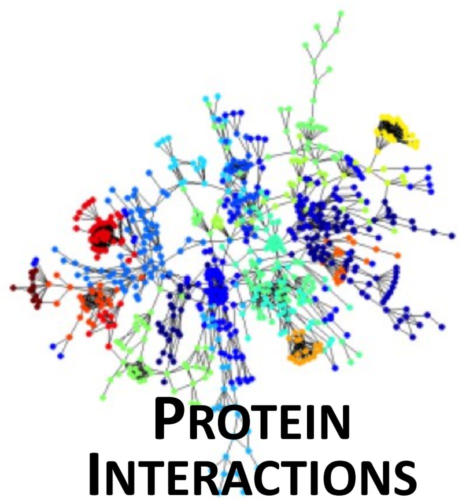


SOCIAL NETWORKS

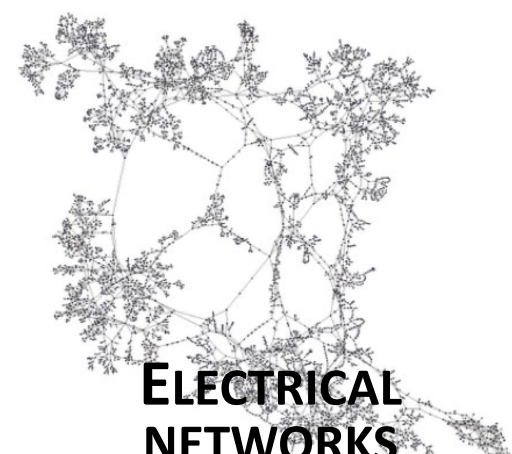


INFORMATION NETWORKS

TRANSPORTATION NETWORKS

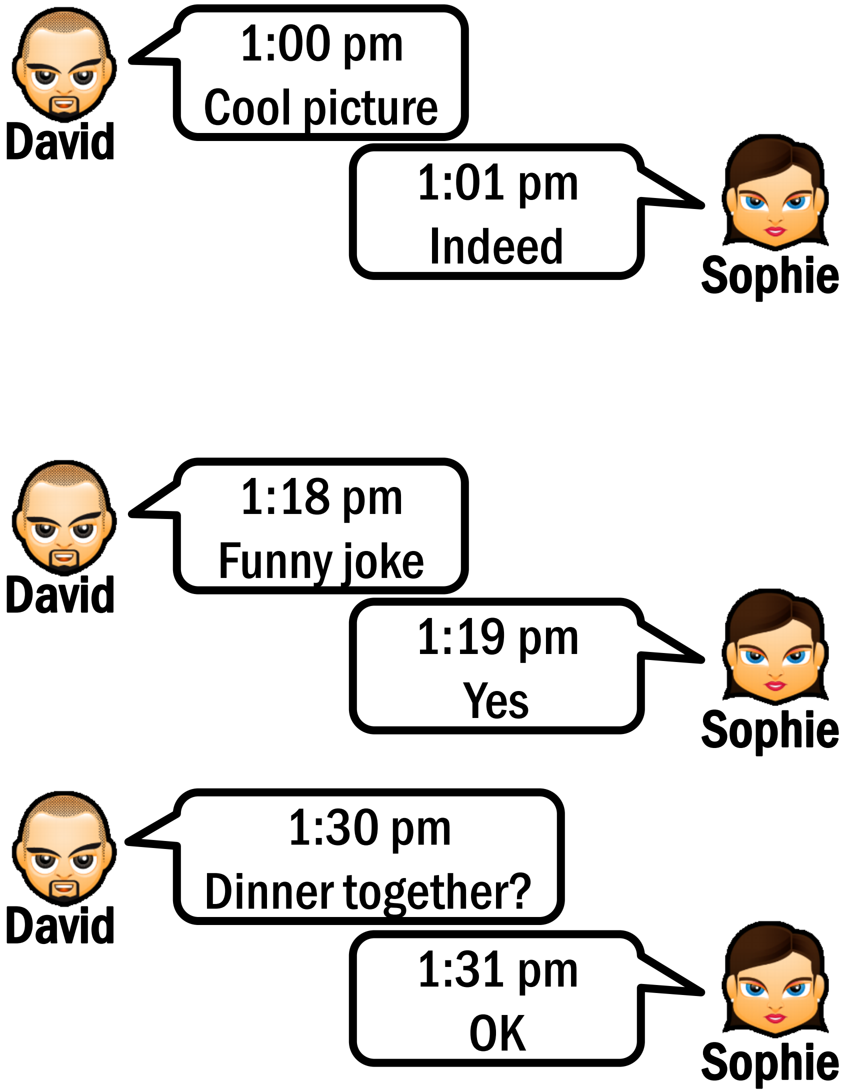


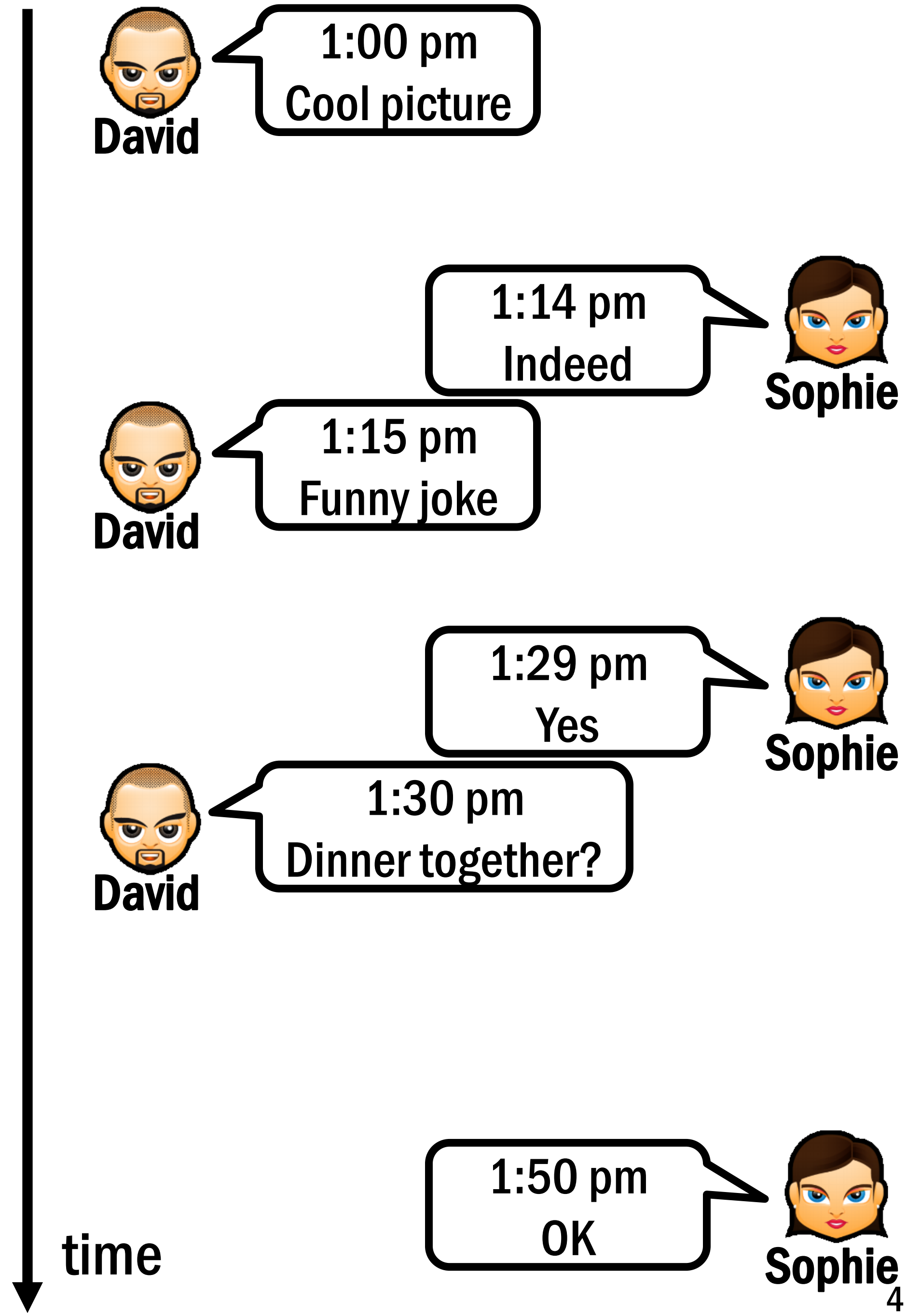
PROTEIN INTERACTIONS



ELECTRICAL NETWORKS

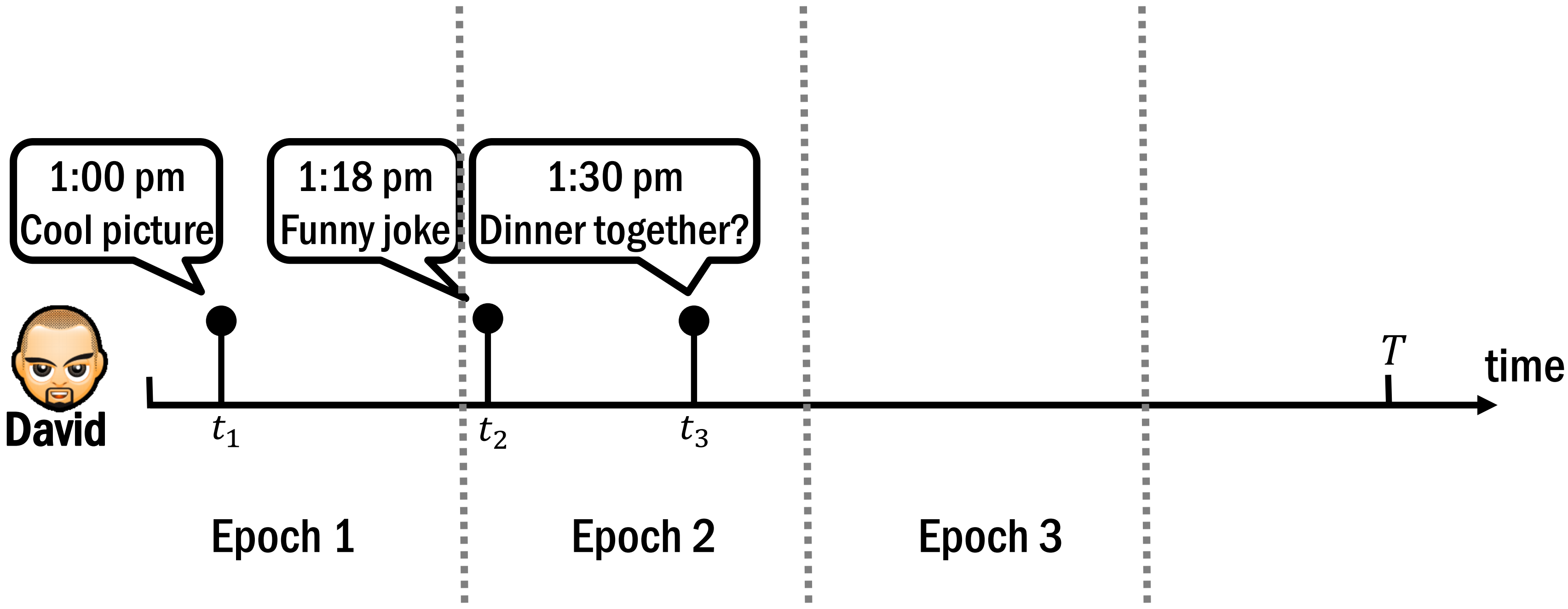
Mostly discrete events in continuous time





Timing is critically important for event data

Why not discrete the time axis for event data?

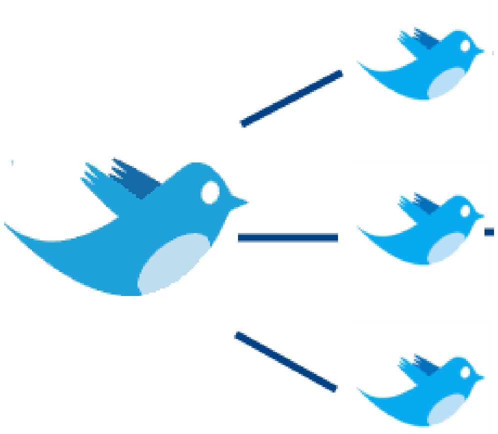


Discrete-time models artificially introduce epochs:

- How long is each epoch?
- How to aggregate events within epoch?
- What if no event within an epoch?
- Time is treated as index or conditioning variable, not easy to deal with time-related queries

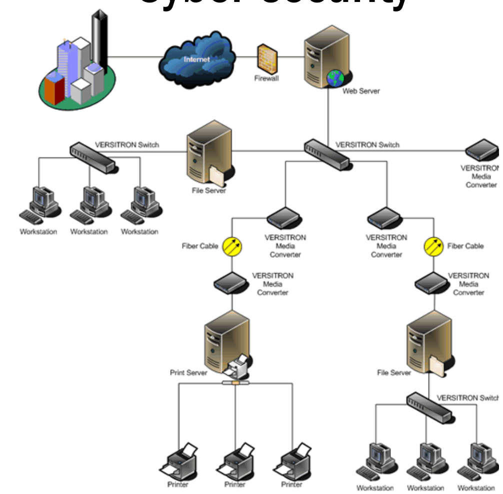
Dynamics are essential to many applications

Information spread



Epidemiology

Cyber-security

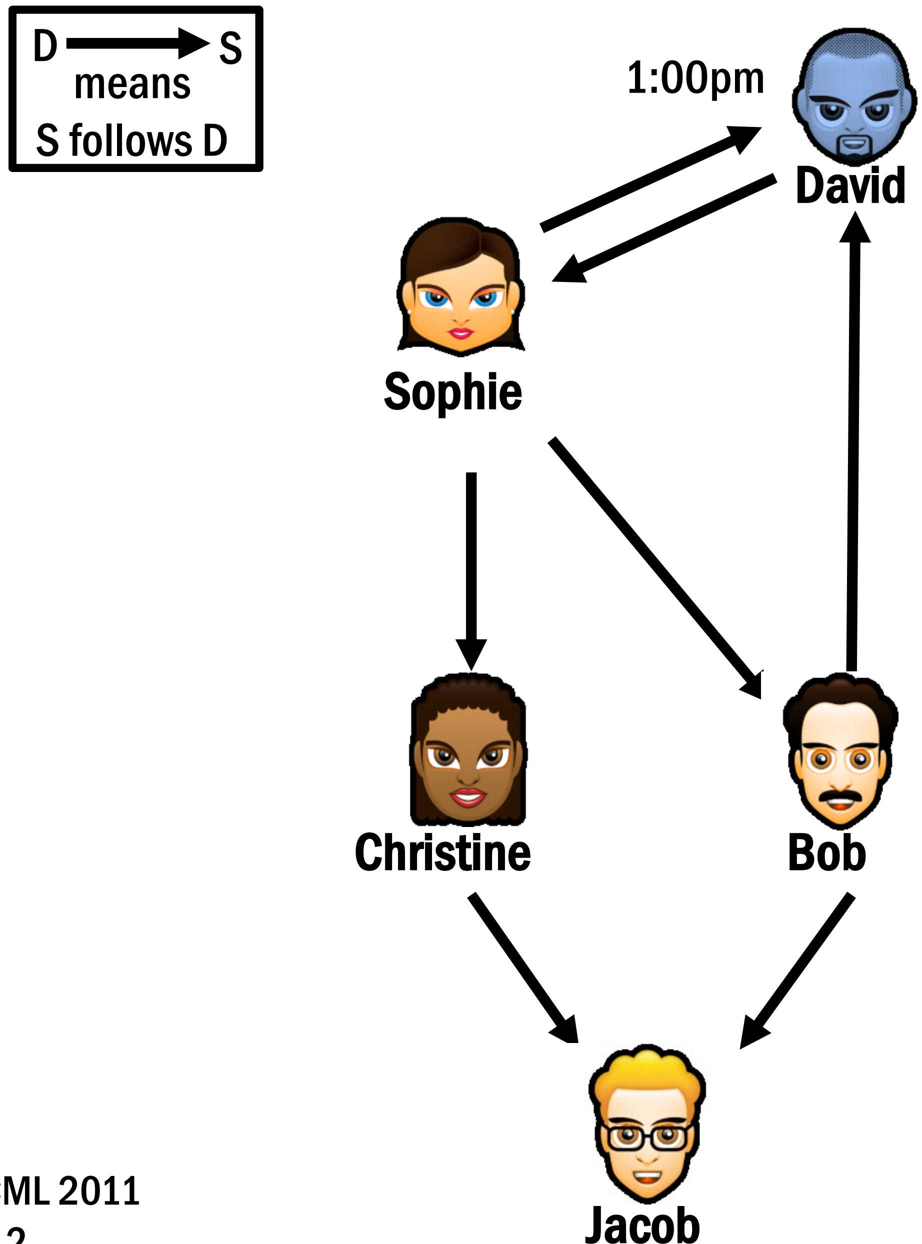


Healthcare analytics

Smart city

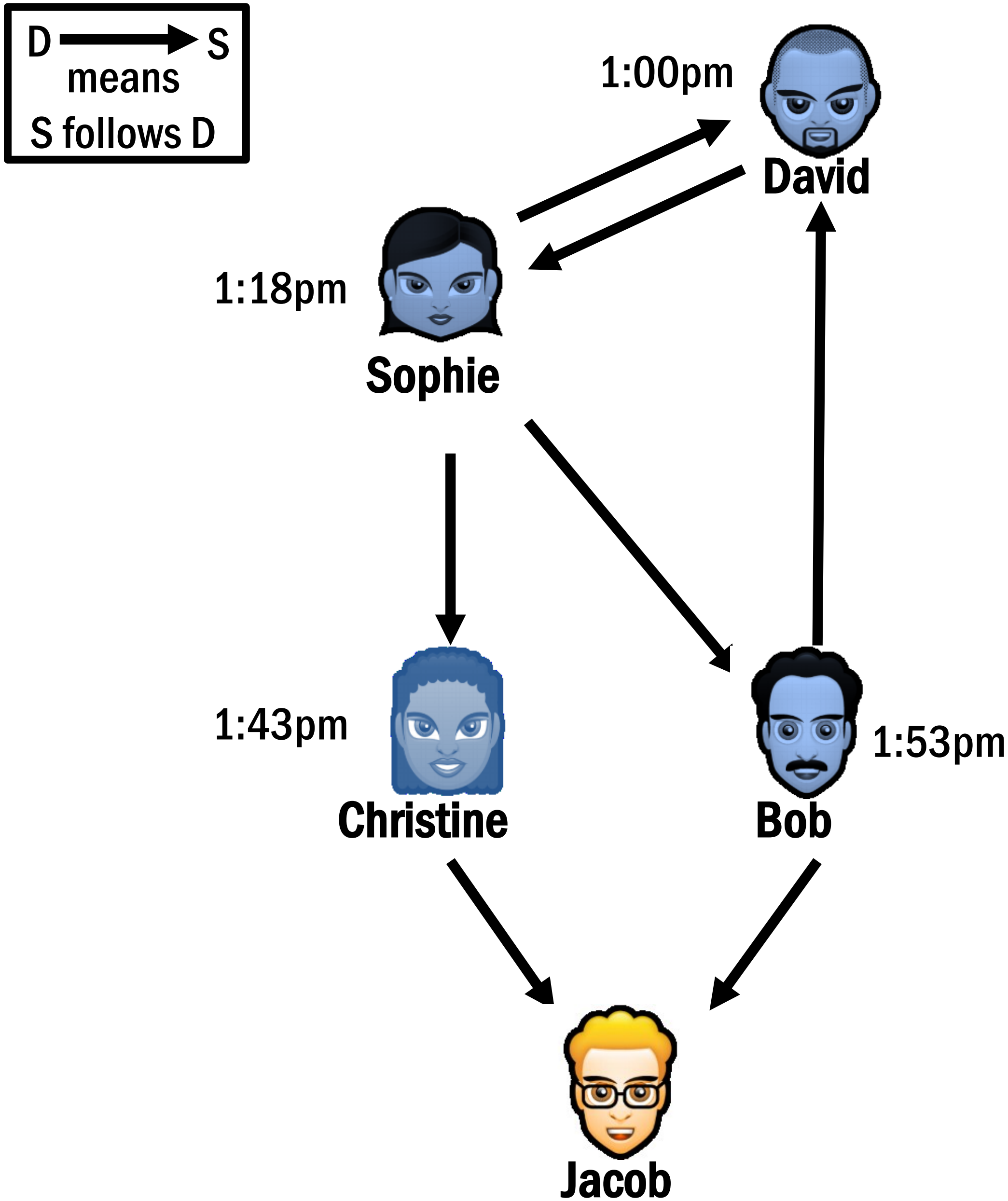
Wildlife conservation

Scenario I: Idea adoption/disease spread/viral marketing

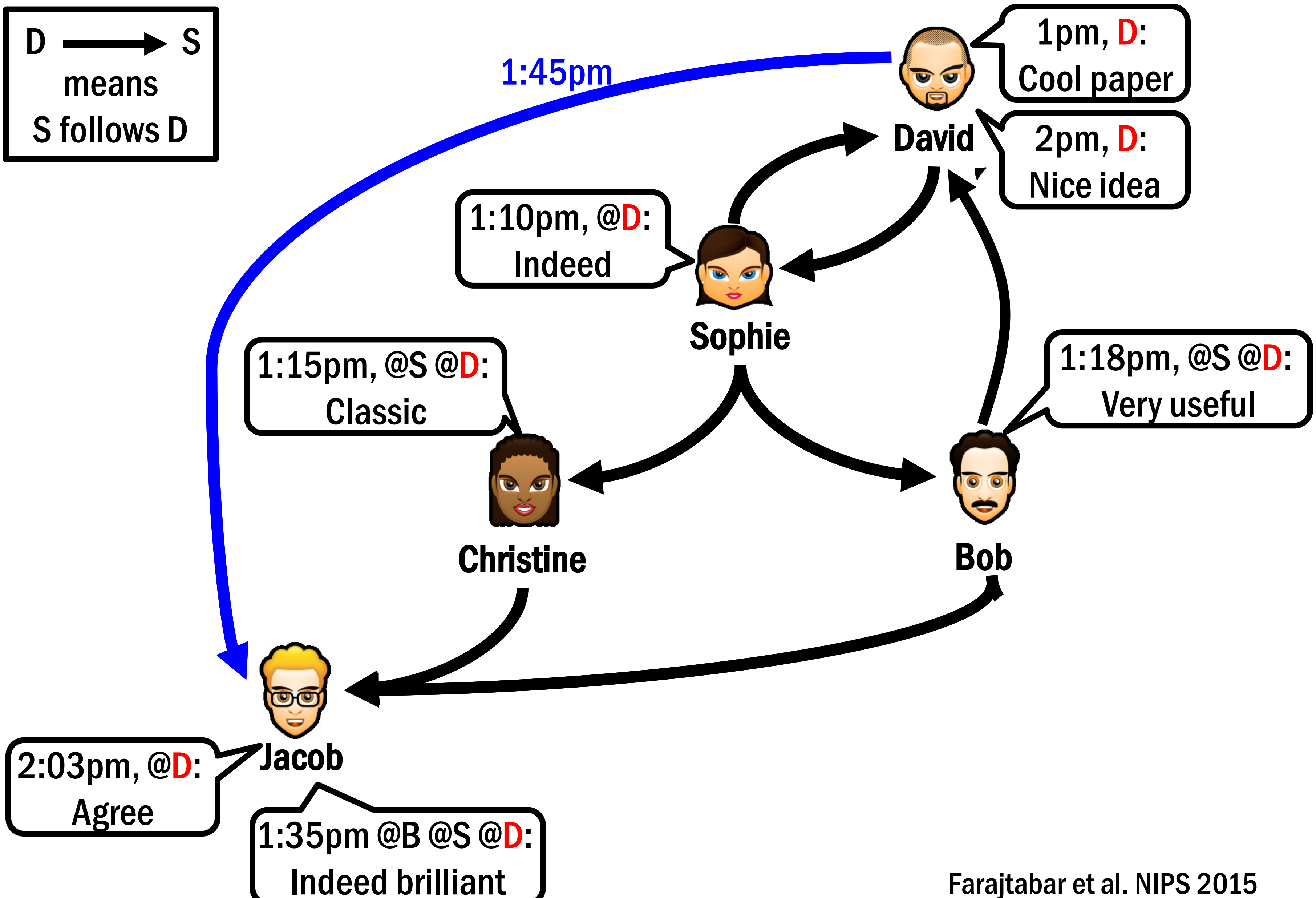


Rodriguez et al. ICML 2011
Du et al. NIPS 2012

Scenario I: Idea adoption/disease spread/viral marketing

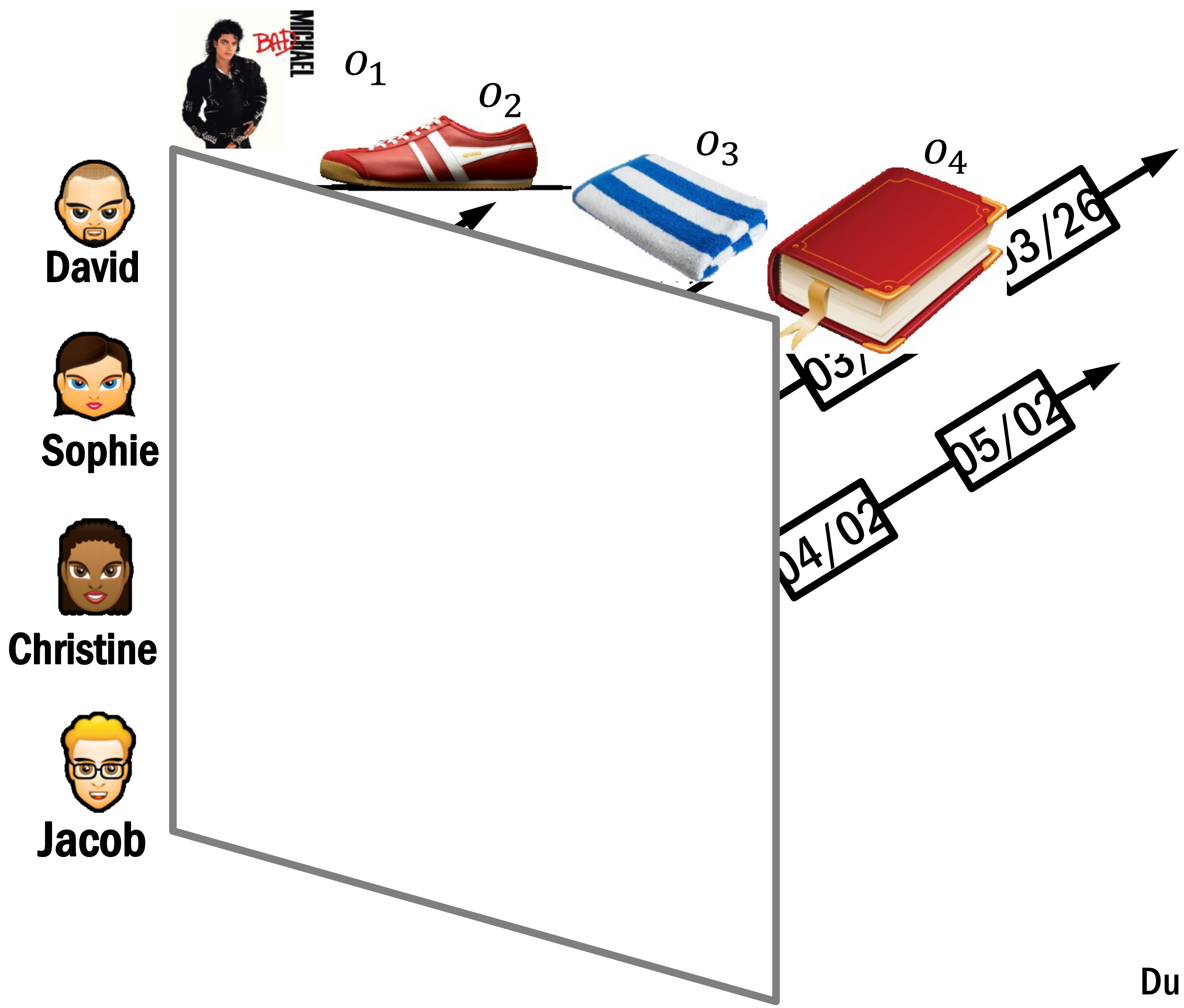


Scenario II: Information diffusion and network coevolution



Farajtabar et al. NIPS 2015

Scenario III: Collaborative dynamics



Du et al. NIPS 2015

A unified framework

Representation: introduce intensity

1. Intensity function
2. Basic building blocks
3. Superposition

Modeling: incorporate domain specifics

1. Idea adoption
2. Network coevolution
3. Collaborative dynamics

Learning : efficient algorithm

1. Sparse hidden diffusion networks
2. Low rank collaborative dynamics
3. Generic algorithm

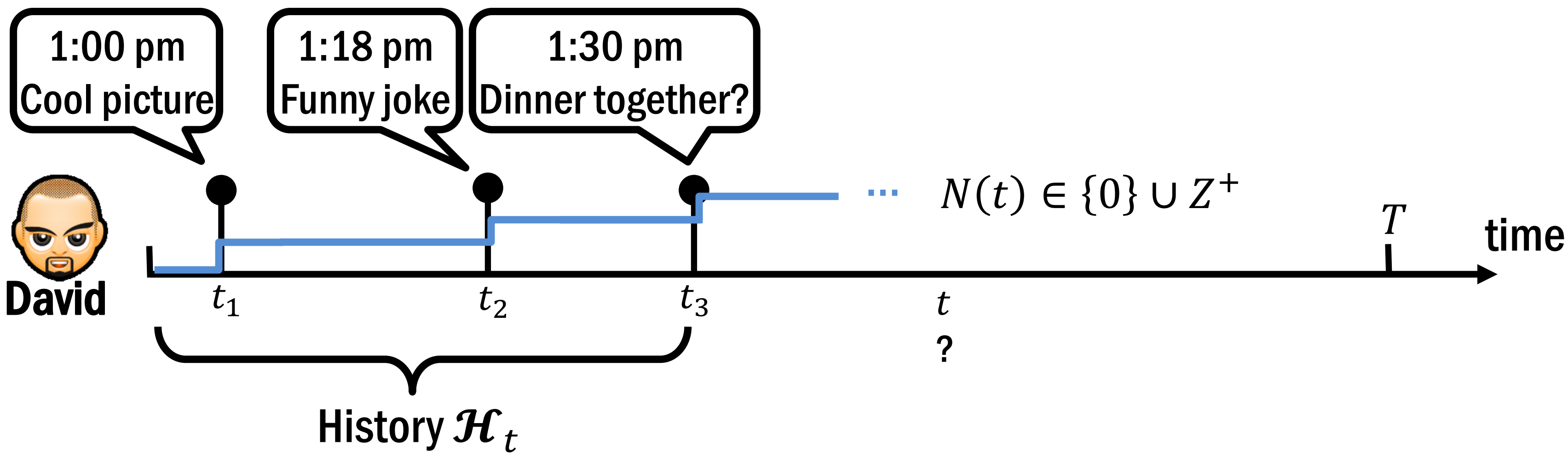
Inference: temporal queries

1. Time-sensitive recommendation
2. Scalable Influence estimation

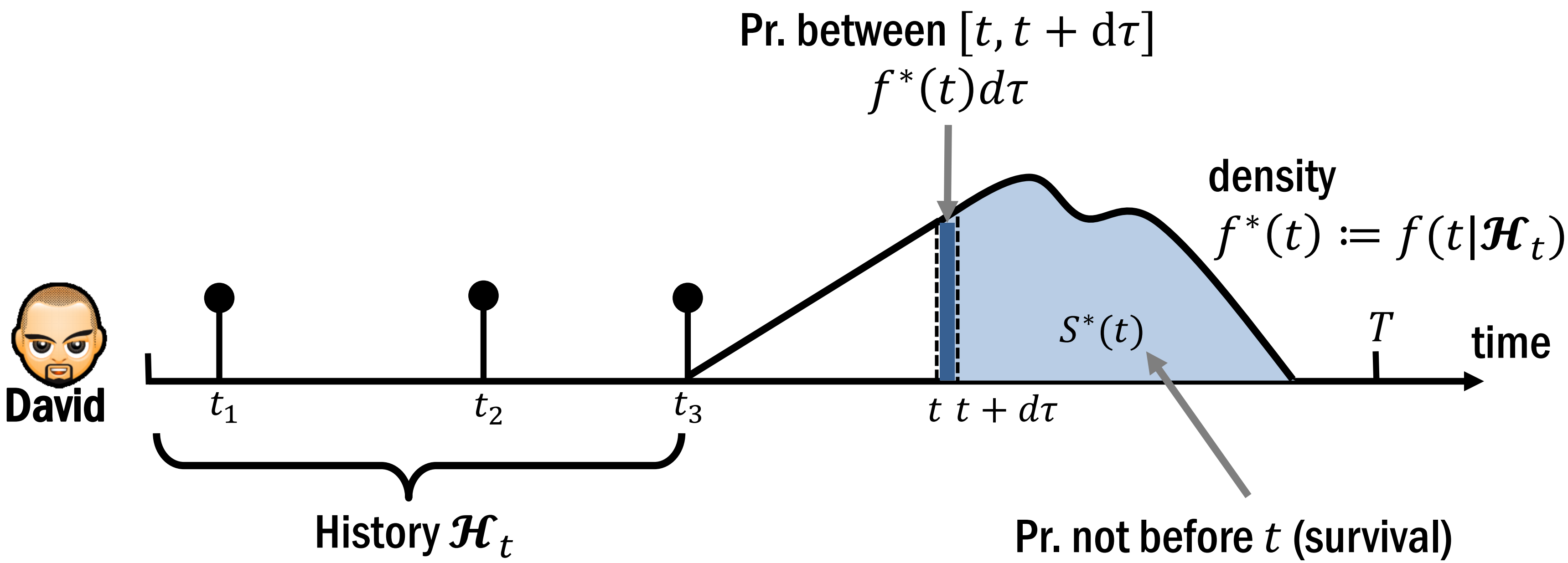
Representation:

Intensity Function

History is a sequence of past events

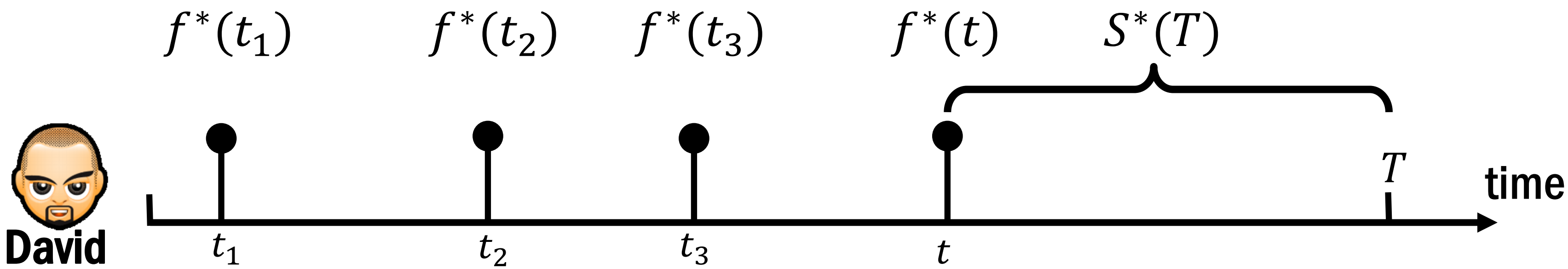


Model time as random variable



$$S^*(t) = 1 - \int_0^t f^*(\tau)d\tau$$

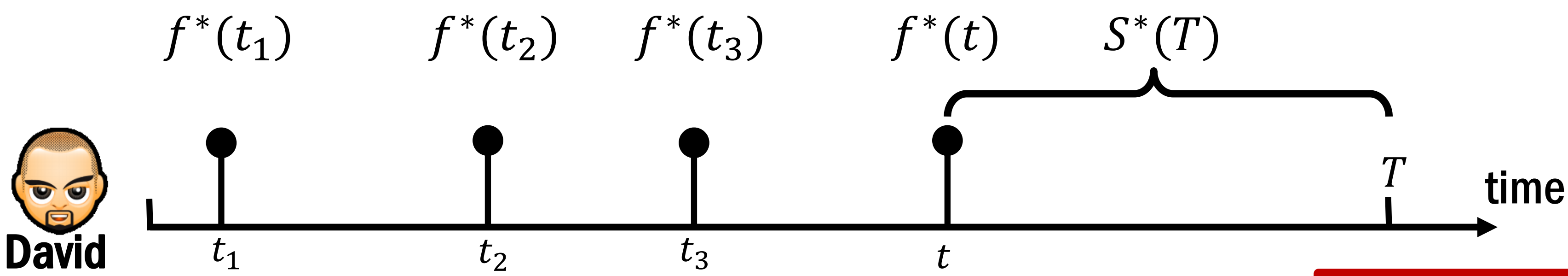
Likelihood of timeline



Likelihood:

$$f^*(t_1) \;\; f^*(t_2) \;\; f^*(t_3) \;\; f^*(t) \;\; S^*(T)$$

Problem of parametrizing density



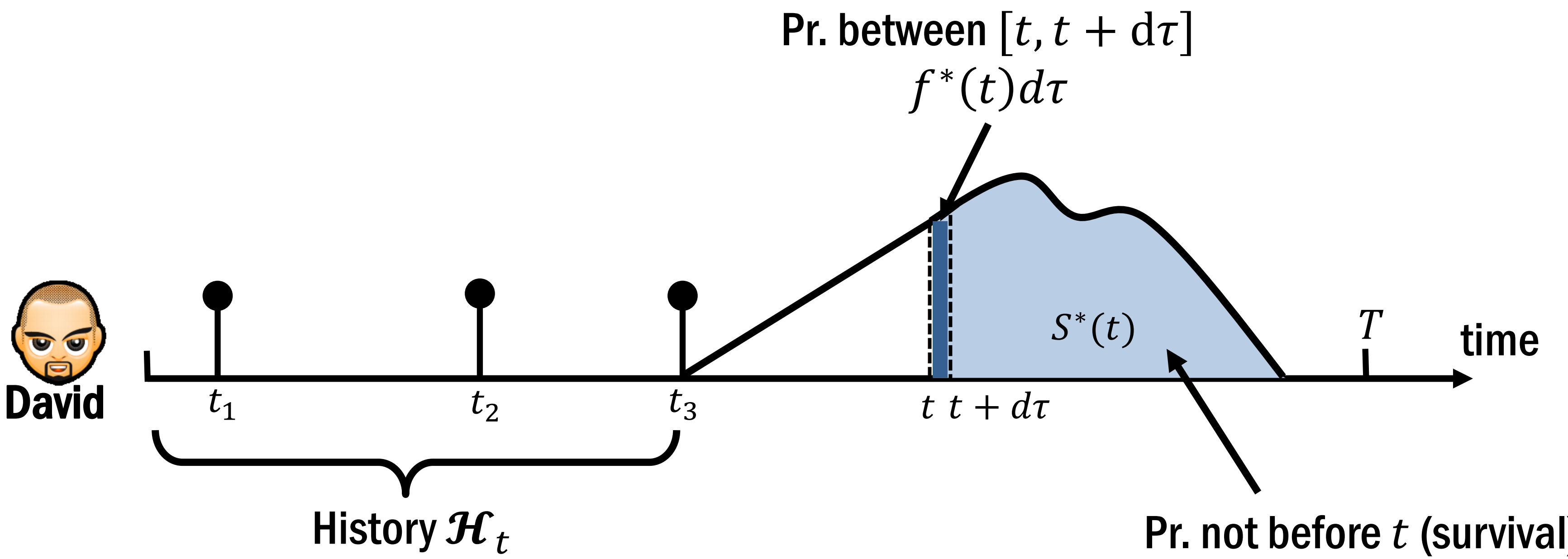
Not concave in w!

Likelihood:

$$f^*(t_1) \quad f^*(t_2) \quad f^*(t_3) \quad f^*(t) \quad S^*(T)$$

$$\frac{\exp\langle w, \psi^*(t_1)\rangle}{Z} \quad \frac{\exp\langle w, \psi^*(t_2)\rangle}{Z} \quad \frac{\exp\langle w, \psi^*(t_3)\rangle}{Z} \quad \frac{\exp\langle w, \psi^*(t)\rangle}{Z} \quad 1 - \int_0^T \frac{\exp\langle w, \psi^*(\tau)\rangle}{Z} d\tau$$

Intensity function



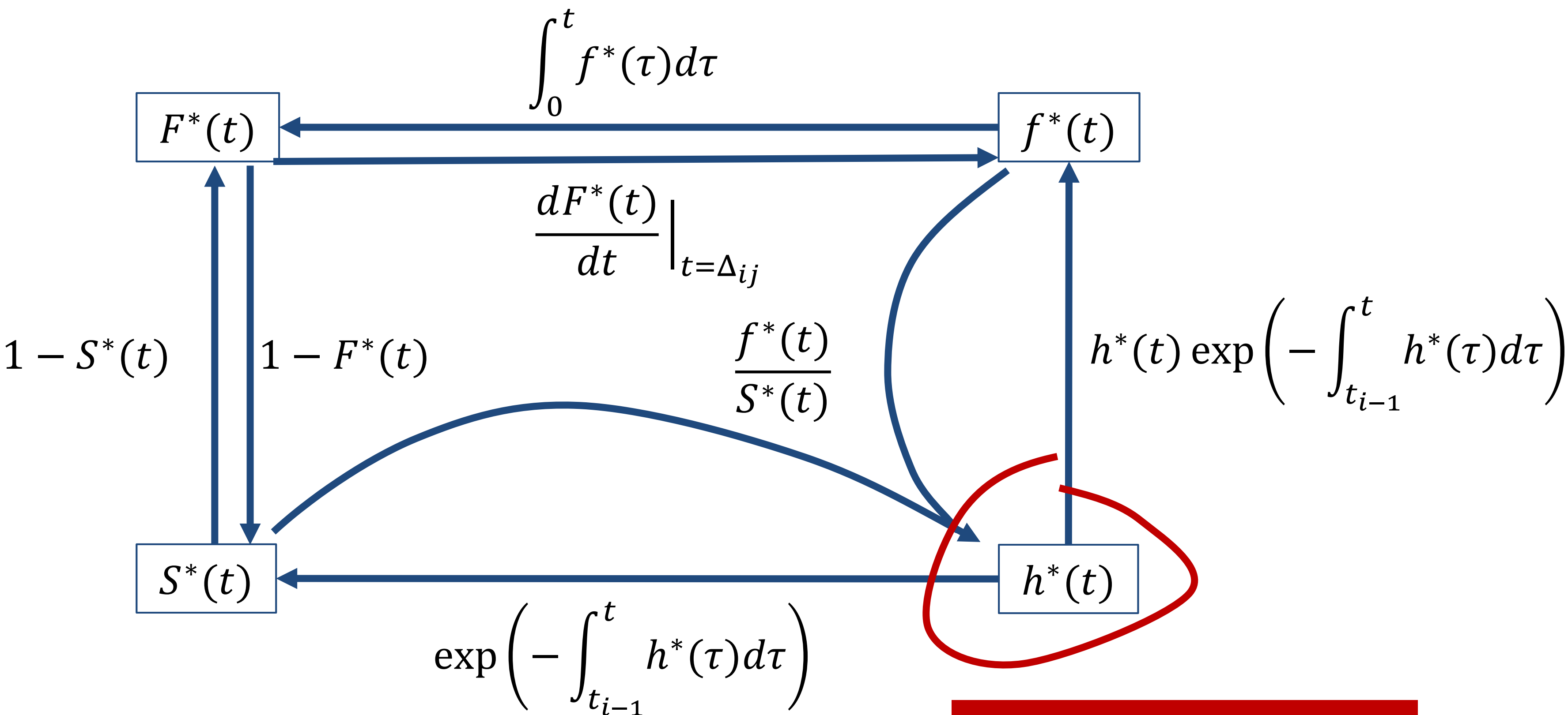
Intensity: Pr. between $[t, t + \mathrm{d}\tau]$ but not before t

$$h^*(t)d\tau = \frac{f^*(t)d\tau}{S^*(t)} > 0$$

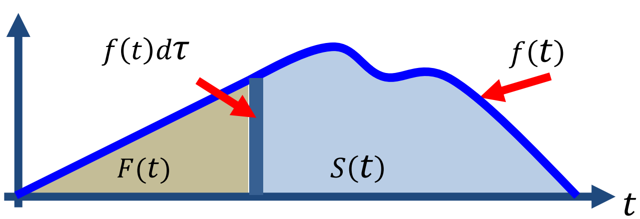
$$f^*(t) = h^*(t)\, S^*(t)$$

$$S^*(t) = \exp\left(-\int_{t_3}^{t} h^*(\tau)d\tau\right)$$

Relation between f^*, F^*, S^*, h^*



Central quantity to parameterize

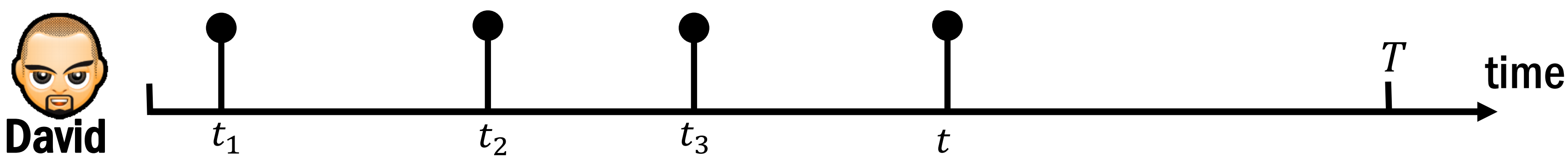


Advantage of parametrizing intensity

Log-likelihood

$$\sum_{i=1}^{m} \log\langle w, \phi^*(t_i)\rangle - \langle w, \Psi^*(T)\rangle$$

Concave in w!



Likelihood:

$$h^*(t_1) \quad h^*(t_2) \quad h^*(t_3) \quad h^*(t) \quad \exp\left(-\int_0^T h^*(\tau)d\tau\right)$$

$\langle w, \phi^*(t_1)\rangle$

$\langle w, \phi^*(t_2)\rangle$

$\langle w, \phi^*(t_3)\rangle$

$\langle w, \phi^*(t)\rangle$

$\exp\left(-\int_0^T \langle w, \phi^*(\tau)\rangle d\tau\right)$

Representation:

Basic Building Blocks

Poisson process

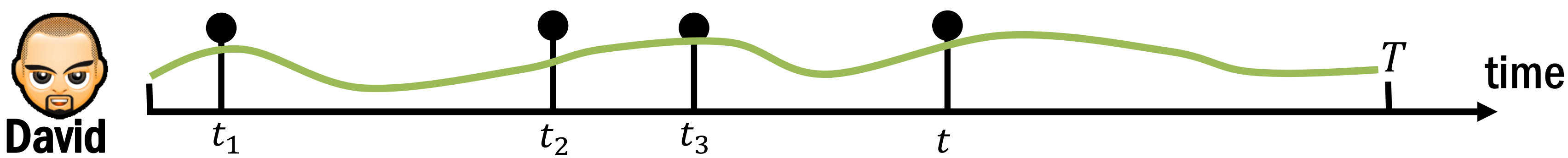
Uniformly random occurrence.

Time interval follows exponential distribution

$$h^*(t) = \mu$$

Inhomogeneous Poisson process

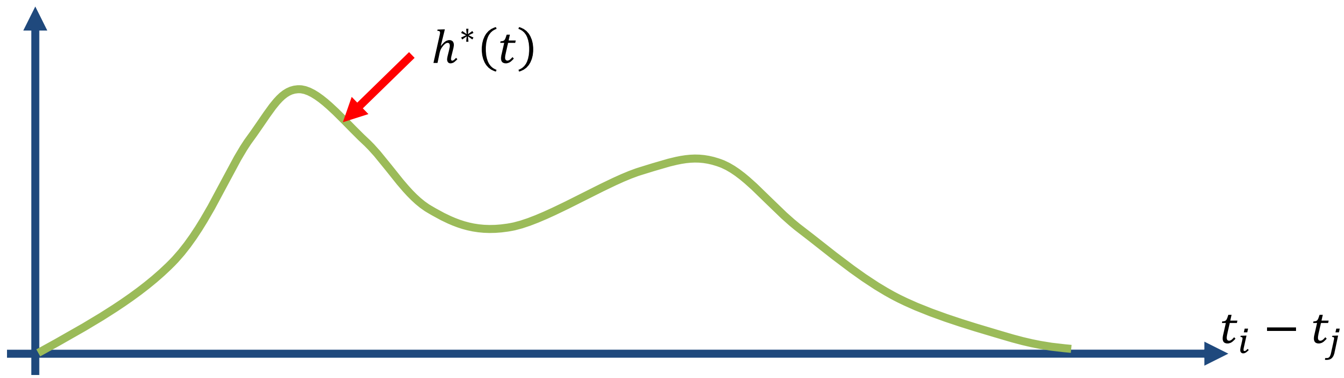
Intensity independent of history

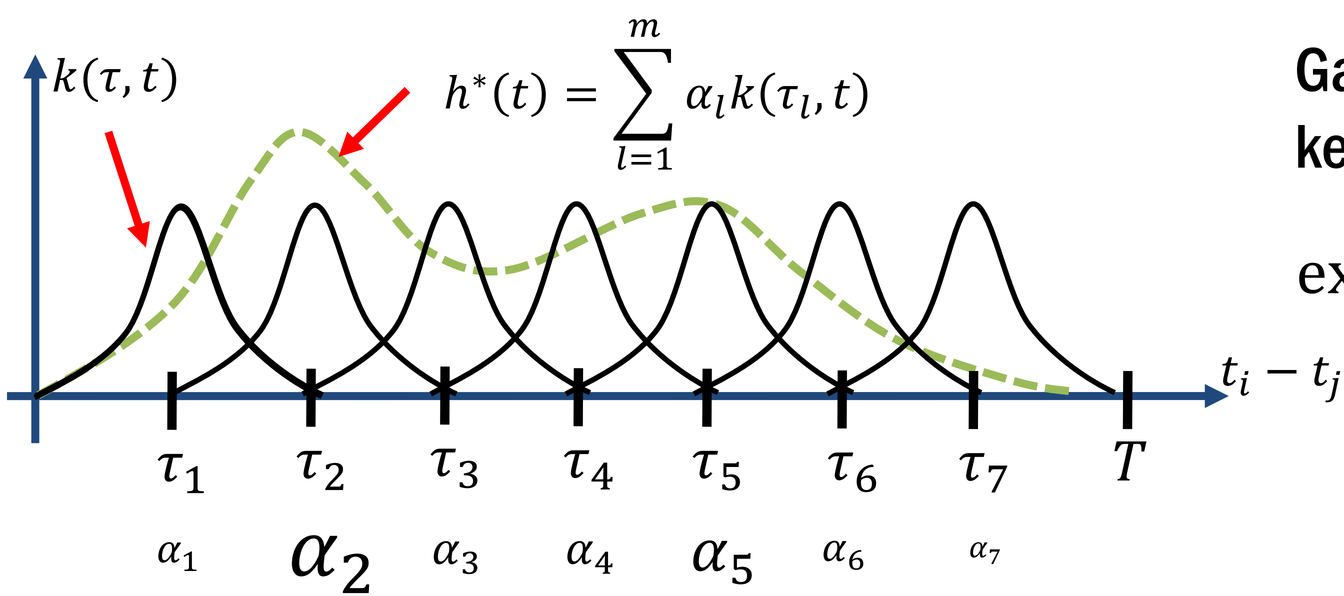


$$h^*(t) = g(t)$$

Nonparametric form of $h^*(t)$

Let $h^*(t)$ be positive combination of basis functions



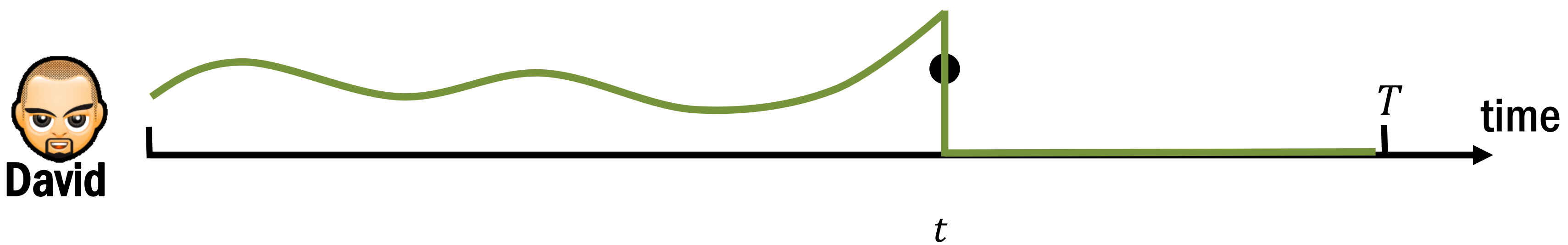


Gaussian RBF kernel $k(\tau, \mathrm{t})$:

$$\exp\left(-\frac{\|\tau - \mathrm{t}\|^2}{2\sigma^2}\right)$$

Terminating process

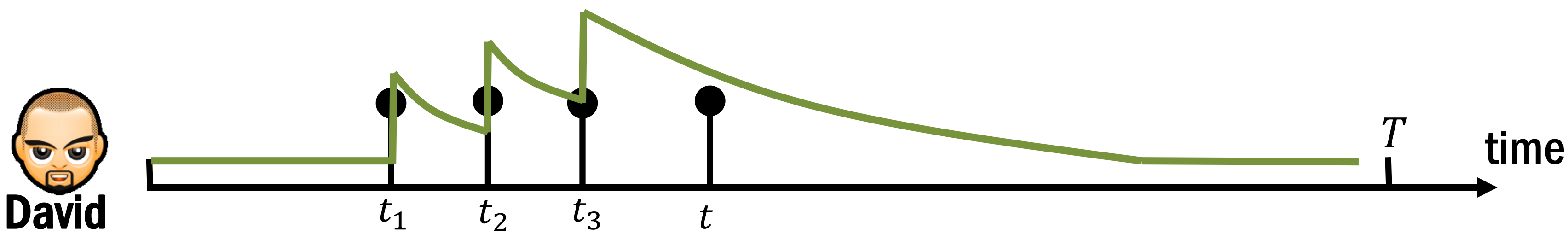
Limited number of occurrence



$$h^*(t) = \big(1 - N(t)\big) g^*(t)$$

Self-exciting or Hawkes process

Clustered occurrence

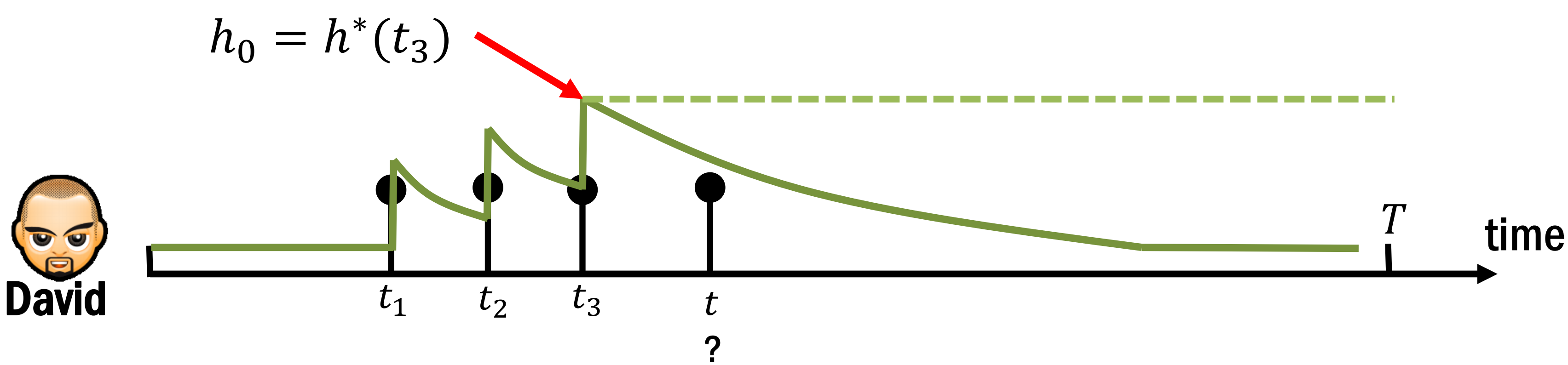


$$h^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}_t} \exp(-|t - t_i|)$$

$$= \mu + \alpha \underbrace{\exp(-|t|)}_{\text{Triggering kernel}} \star dN(t)$$

How to sample from intensity?

Thinning procedure (similar to rejection sampling)



$$h^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}_t} \exp(-|t - t_i|)$$

Sample t from homogeneous Poisson process with intensity h_0

$$t \sim -\frac{1}{h_0} \ln U[0,1]$$

Keep the sample with probability $h^*(t)/h_0$

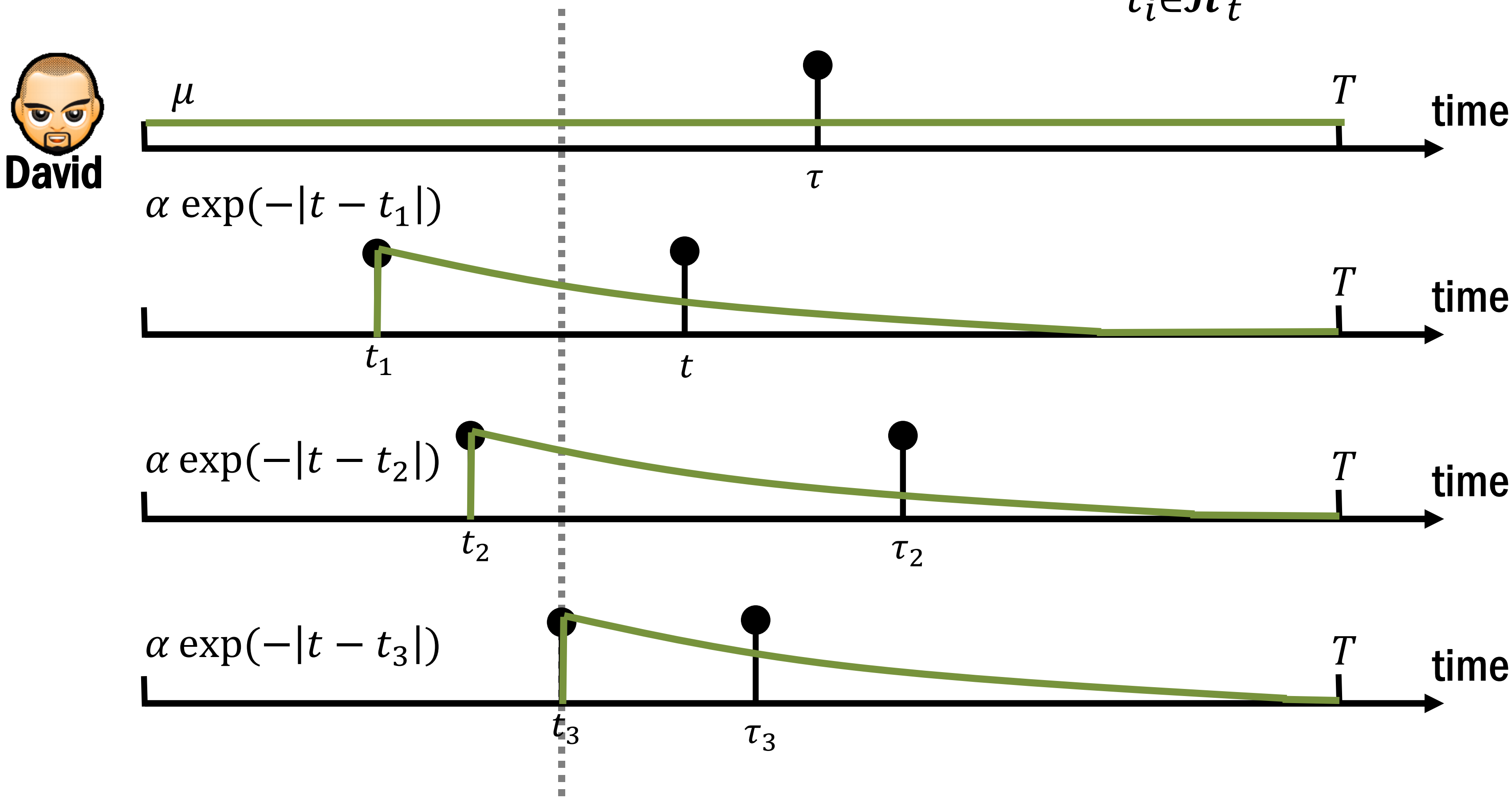
Dynamic Processes over Information Networks
Representation, Modeling, Learning and Inference

Representation:
Superposition

Supposition of processes

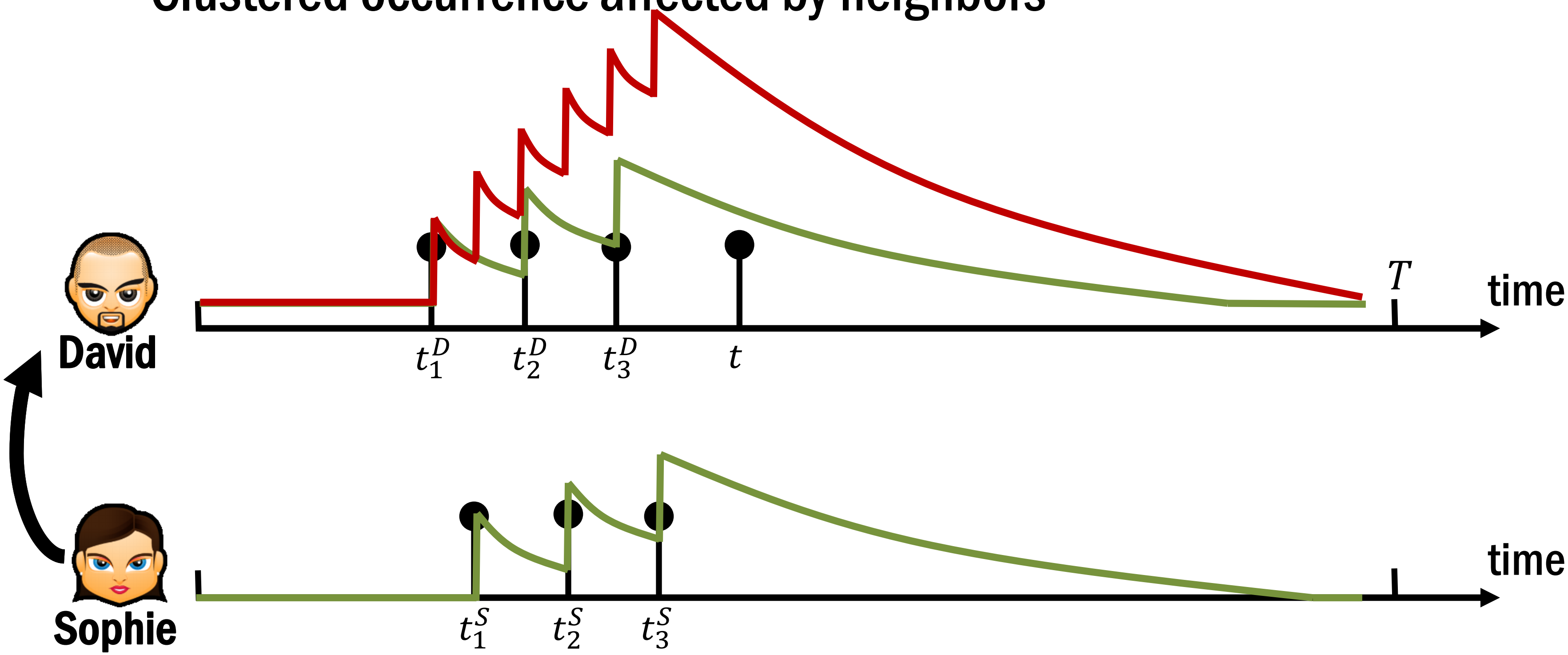
Sample each intensity + take minimum = Additive intensity

$$t = \min\{\tau,\ \tau_1,\ \tau_2,\ \tau_3\} \quad \Rightarrow \quad h^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}_t} \exp(-|t - t_i|)$$



Mutually-exciting process

Clustered occurrence affected by neighbors

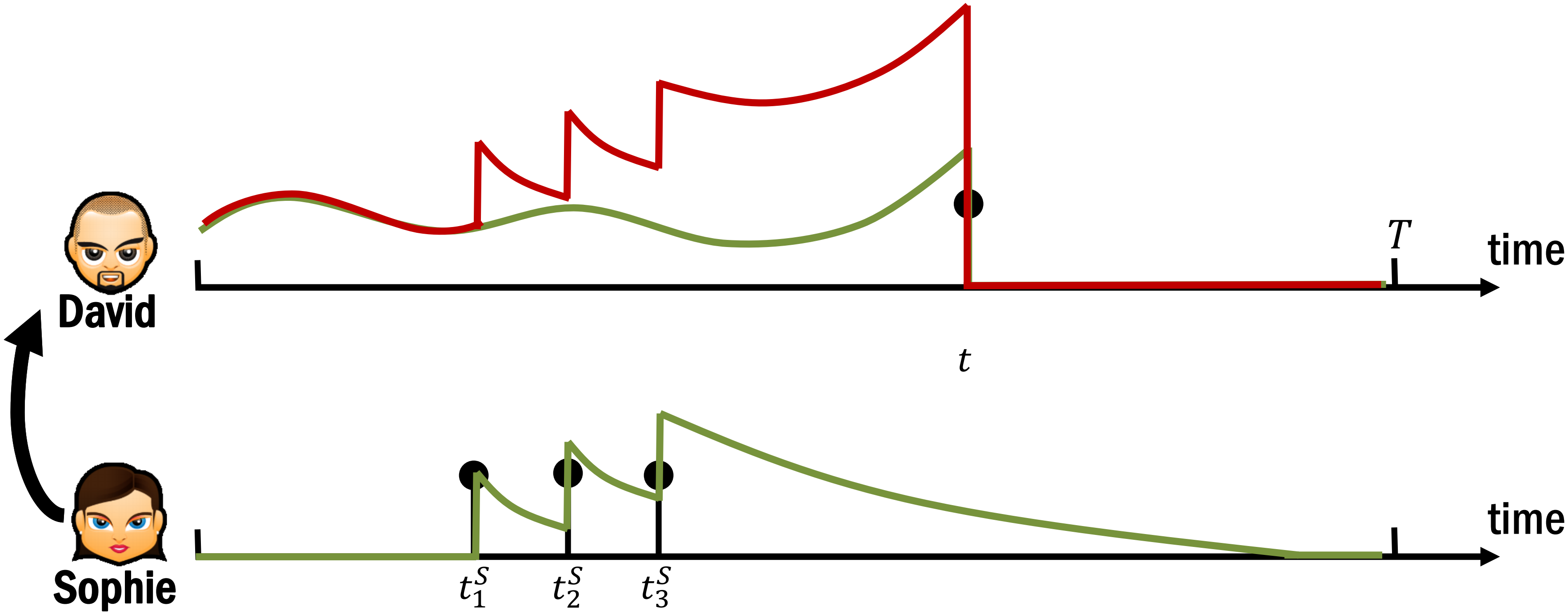


$$h^{D*}(t) = \mu + \alpha^D \sum_{t_i^D \in \mathcal{H}_t^D} \exp(-|t - t_i^D|)$$

$$+ \alpha^{DS} \sum_{t_i^S \in \mathcal{H}_t^S} \exp(-|t - t_i^S|)$$

Mutually-exciting terminating process

Limited number of occurrence affected by neighbors



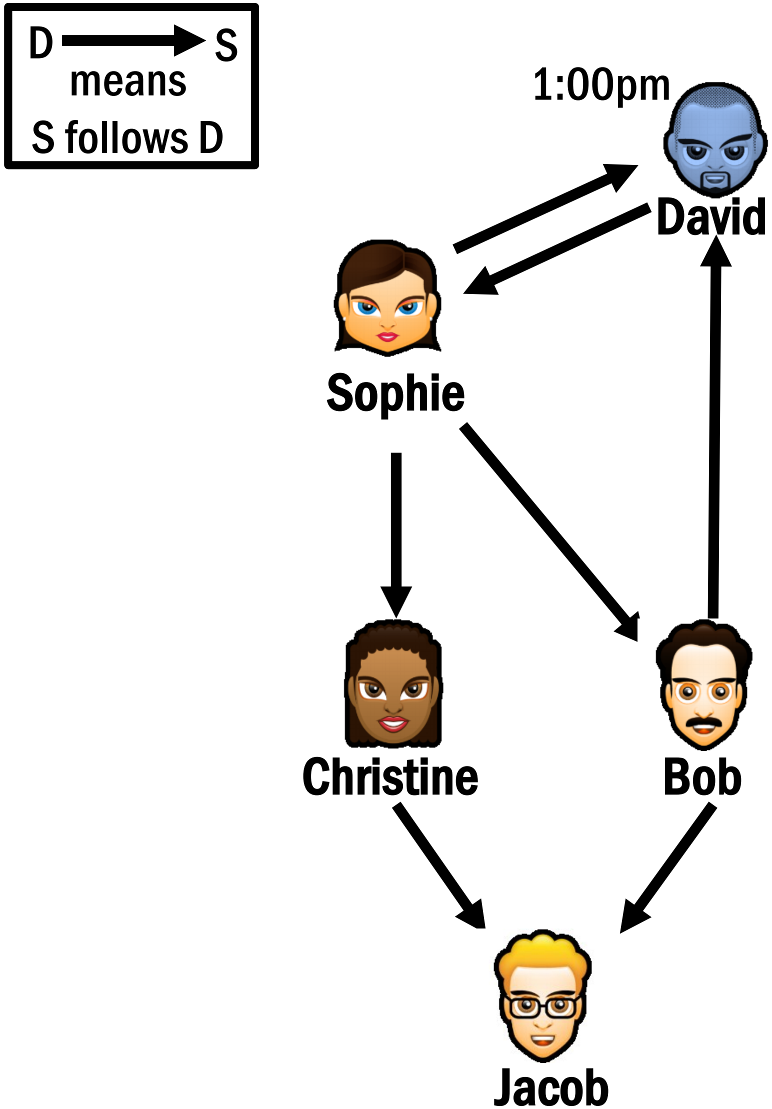
$$h^{D*}(t) = \left(1 - N^D(t)\right)\left(g^*(t) + \alpha^{DS} \sum_{t_i^S \in \mathcal{H}_t^S} \exp\left(-|t - t_i^S|\right)\right)$$

Georgia Tech

Dynamic Processes over Information Networks
Representation, Modeling, Learning and Inference

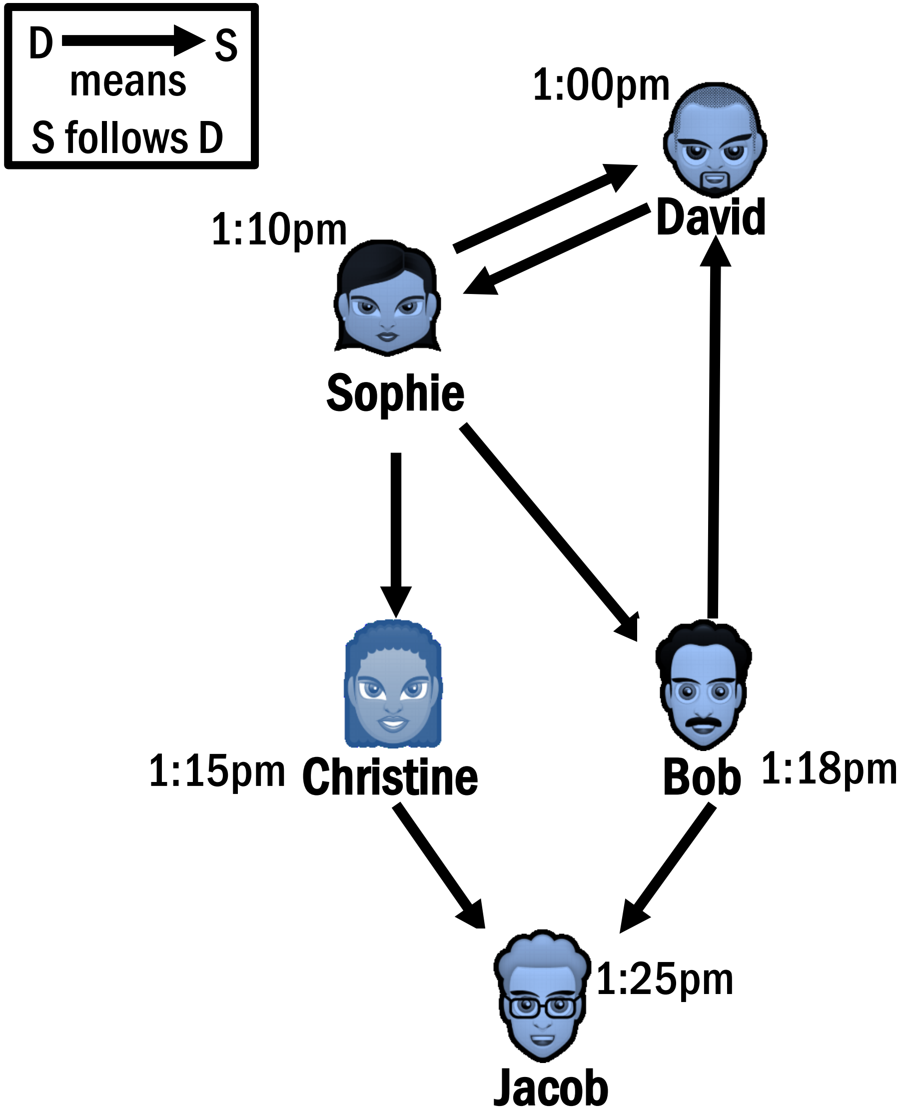
Modeling:
Idea Adoption

idea adoption/disease spread/viral marketing

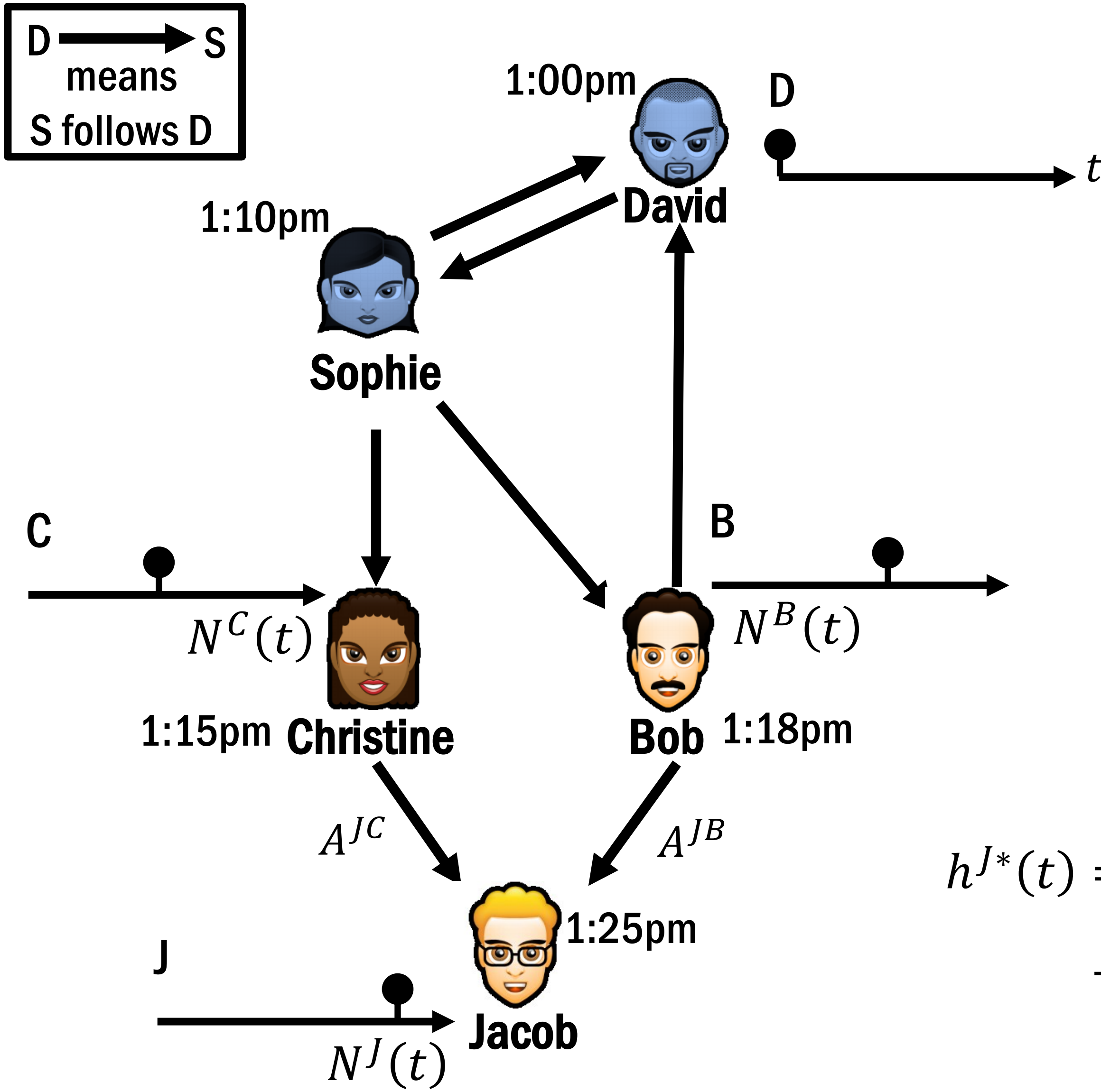


Rodriguez et al. ICML 2011
Du et al. NIPS 2012

idea adoption/disease spread/viral marketing



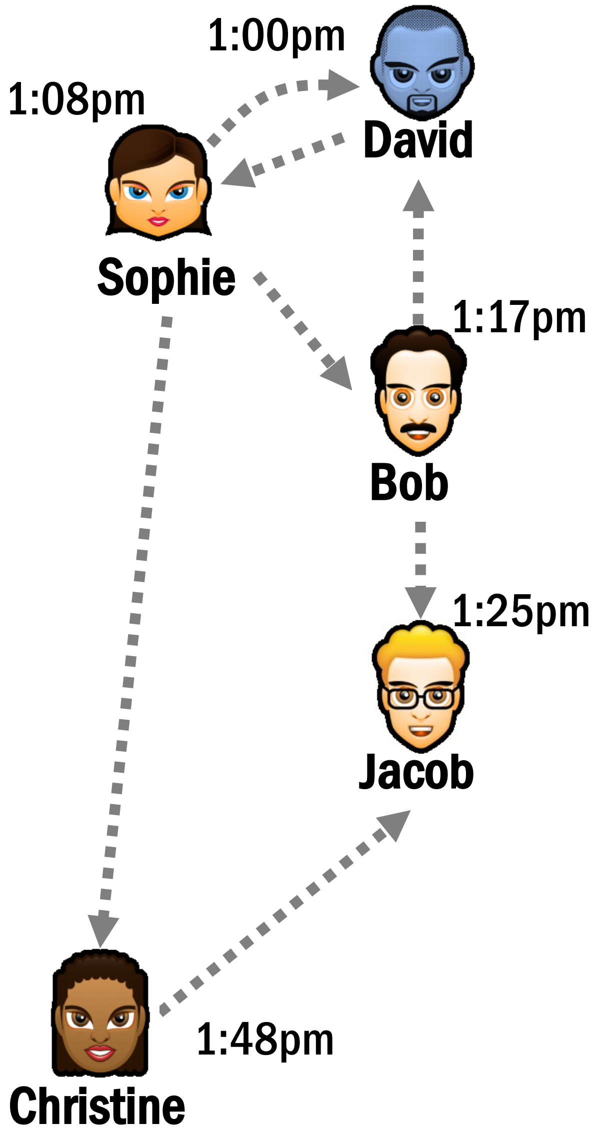
Scenario I: idea adoption



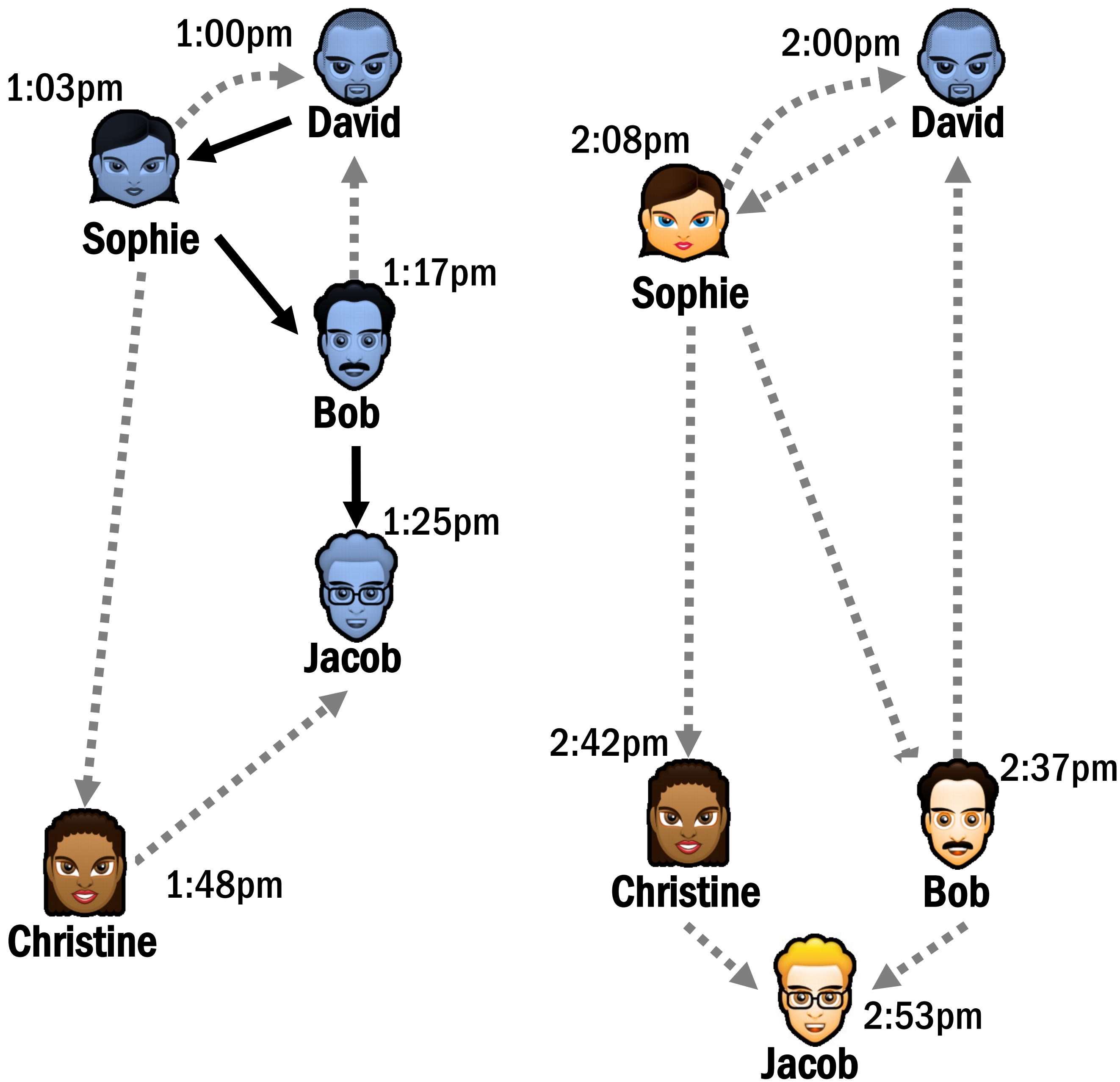
D is source
$N^D(t) = 1$

$$h^{J*}(t) = A^{JB}(t)\overbrace{\overbrace{\left(1 - N^J(t)\right)}^{\text{Not yet adopted}}\overbrace{N^B(t)}^{\text{Followee adopted}}}^{\text{Terminating process adopt product only once}} + A^{JC}(t)\left(1 - N^J(t)\right) N^C(t)$$

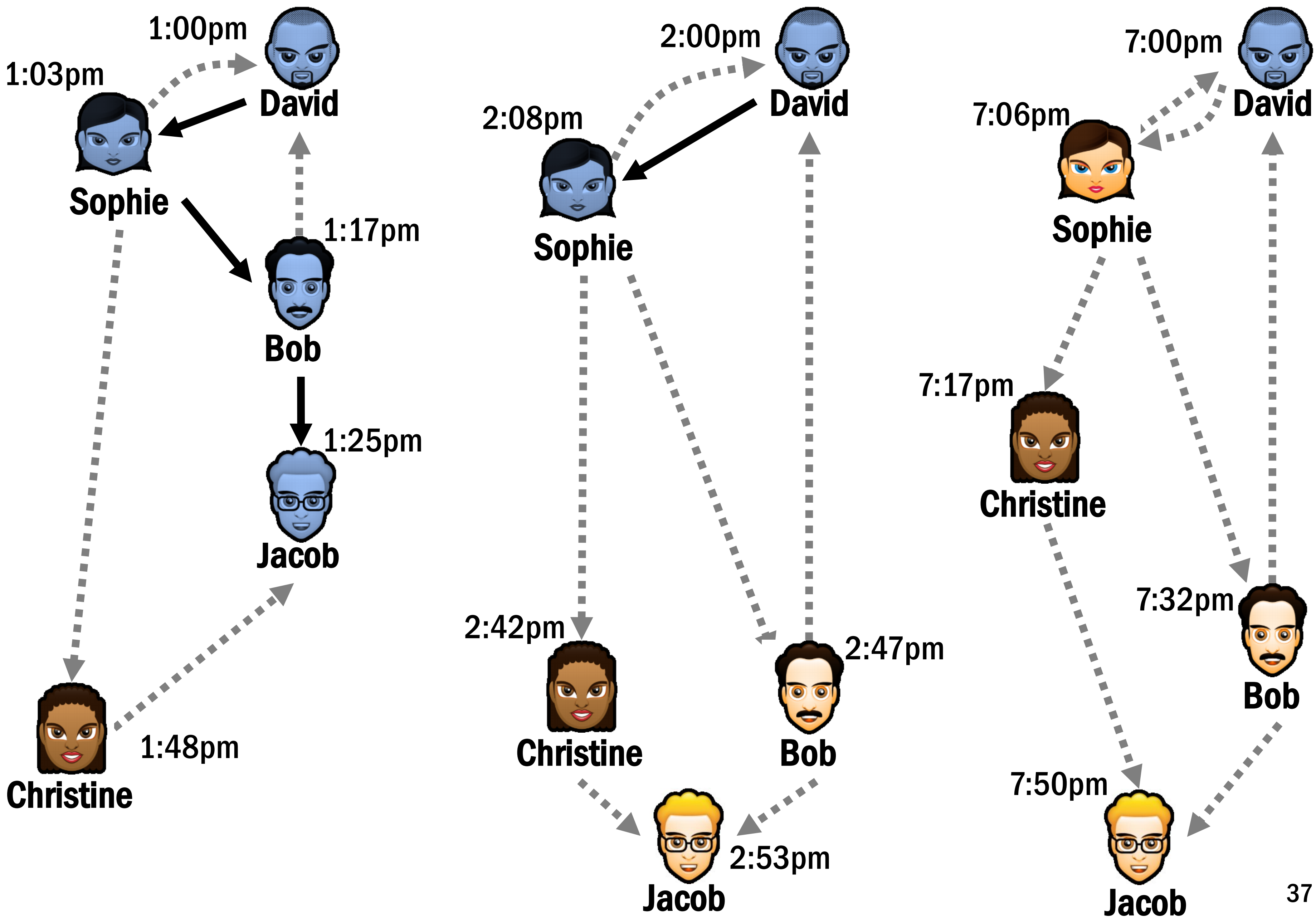
Cascades from D in 30 mins



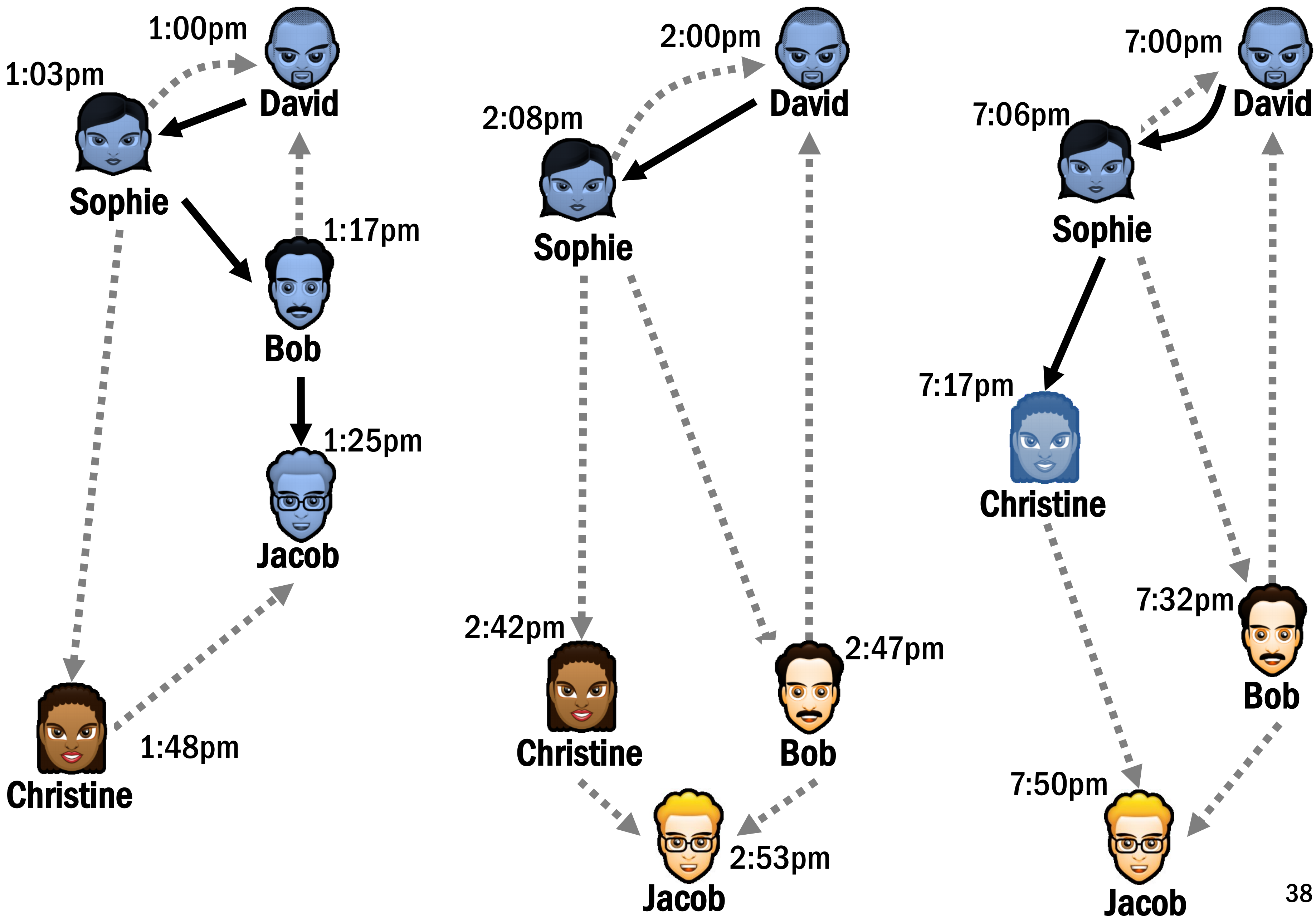
Cascades from D in 30 mins



Cascades from D in 30 mins



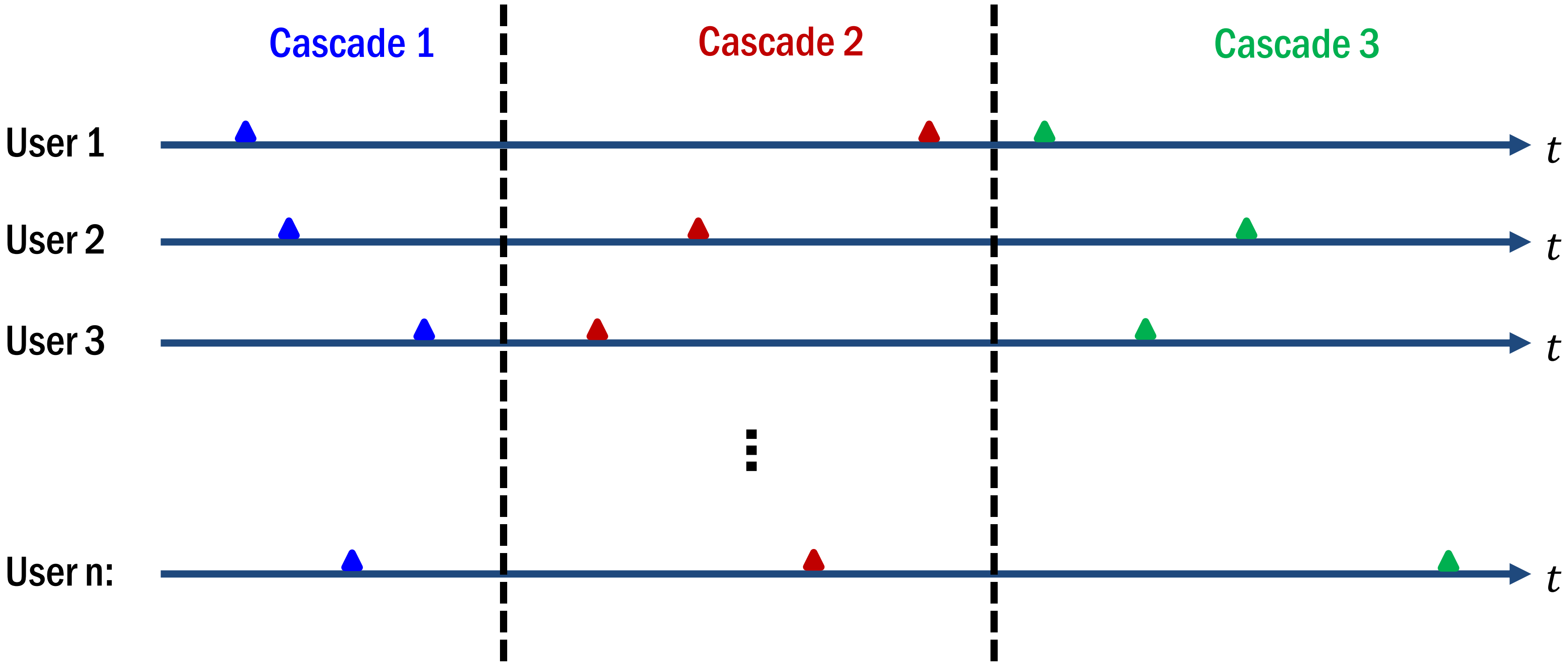
Cascades from D in 30 mins



Cascade Data

Cascade: a sequence of (node, time) pairs for a particular piece of news

Cascades can start from different sources



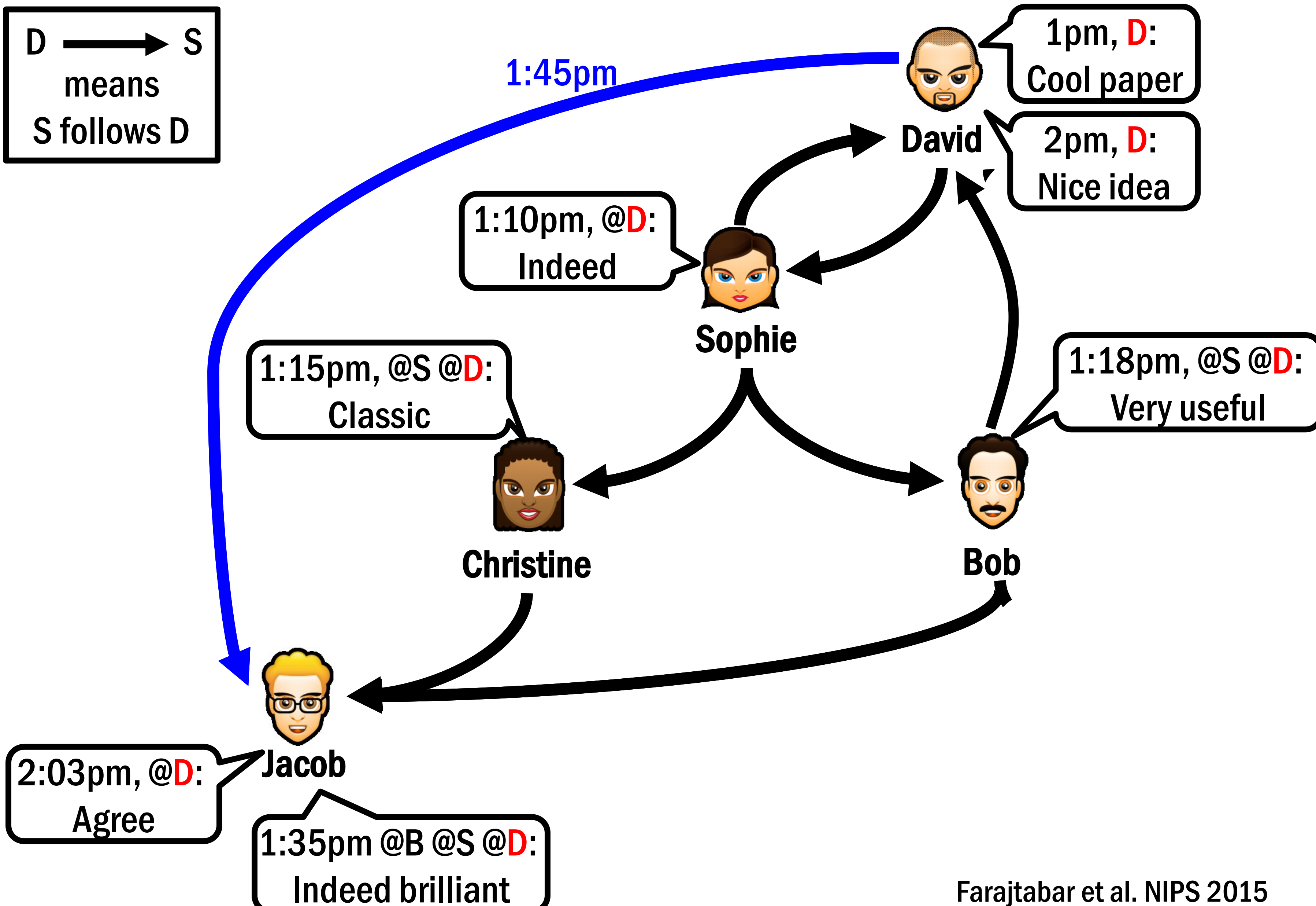
$(t_1, t_2, t_3, \ldots, t_n)$ $(t_1, t_2, t_3, \ldots, t_n)$ $(t_1, t_2, t_3, \ldots, t_n)$

Dynamic Processes over Information Networks
Representation, Modeling, Learning and Inference

Modeling:

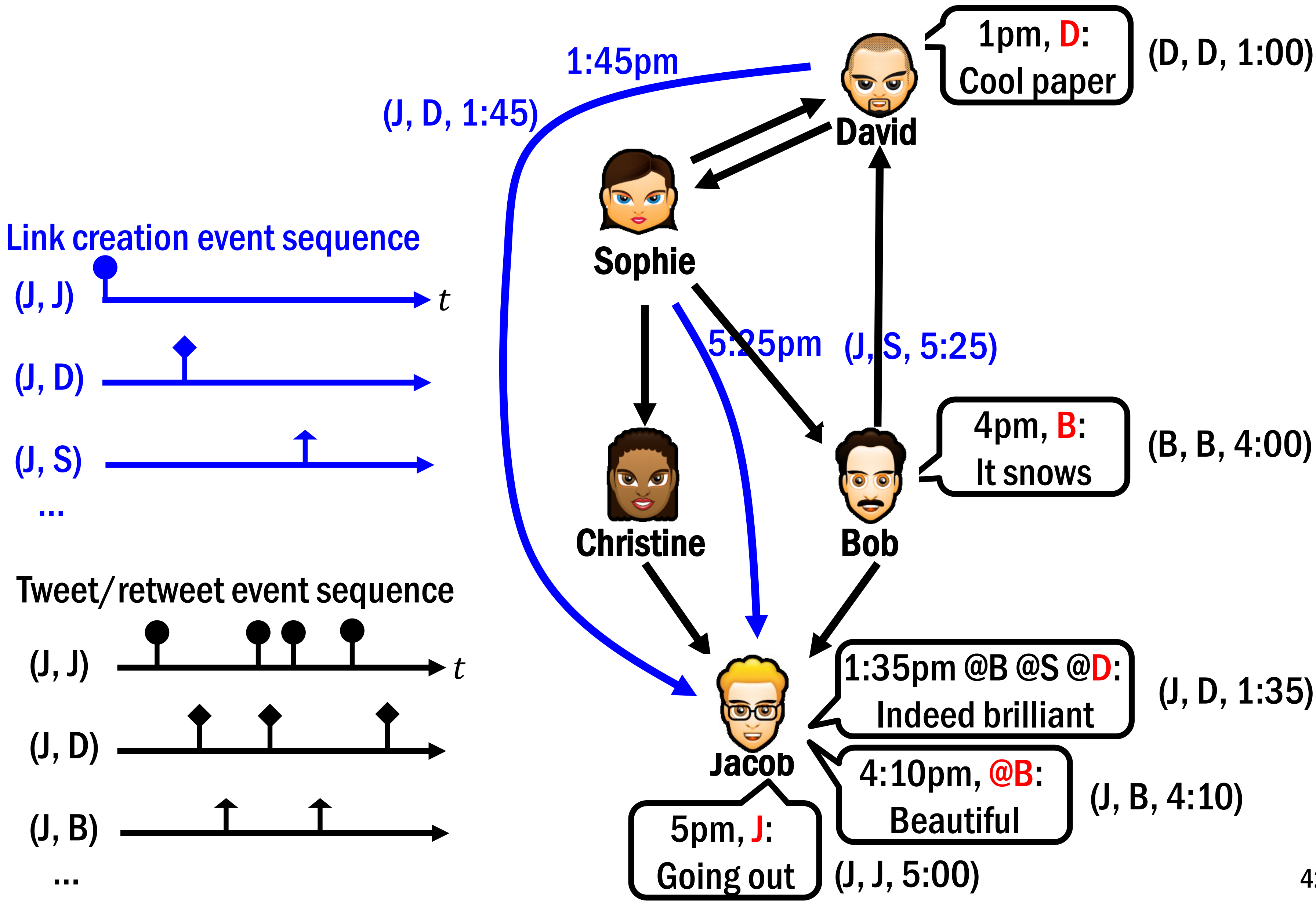
Coevolution

Information diffusion and network coevolution

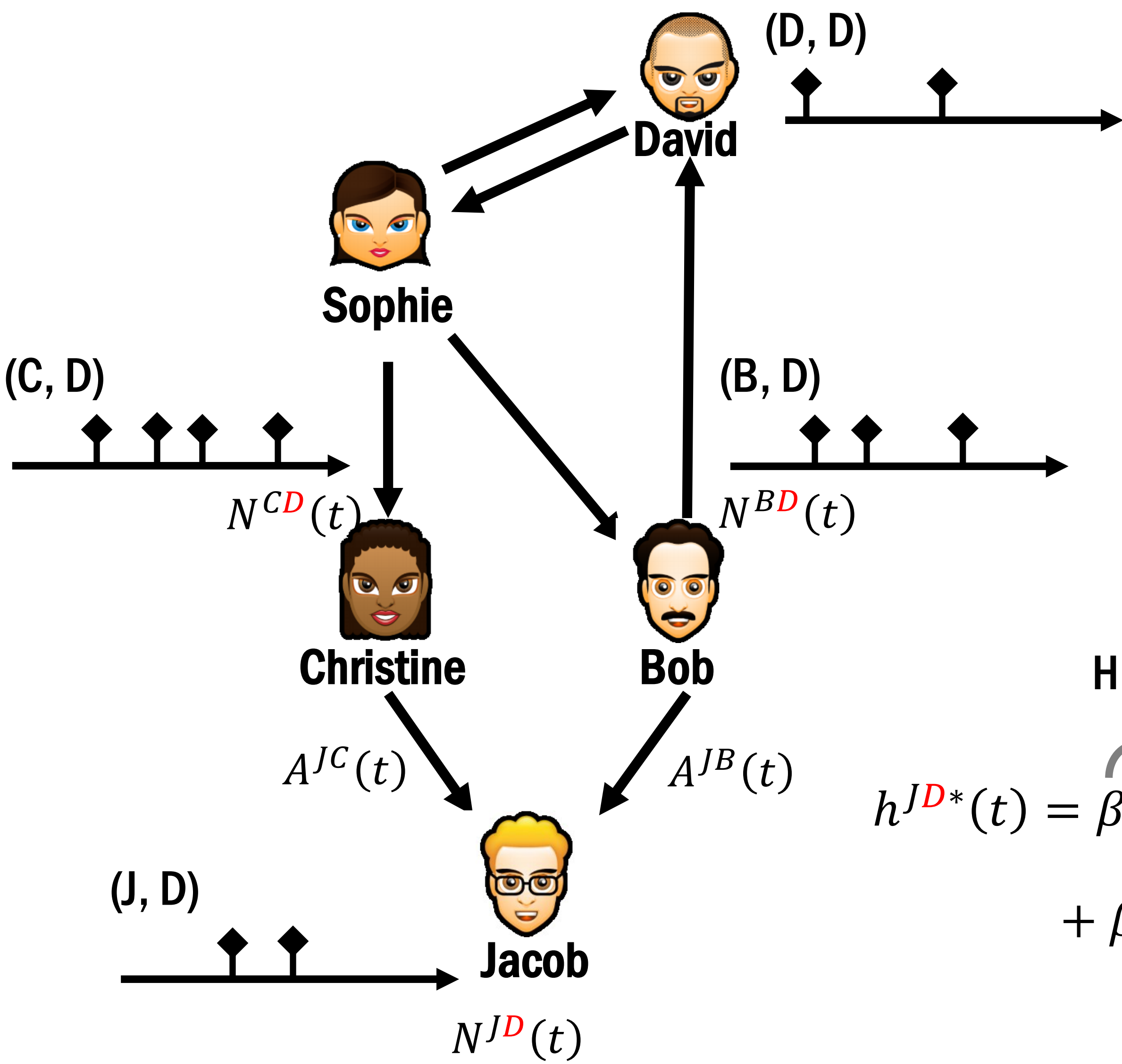


Farajtabar et al. NIPS 2015

Information diffusion and network coevolution



Targeted retweet



D's own initiative

$$h^{D*}(t) = \eta$$

Mutually-exciting process

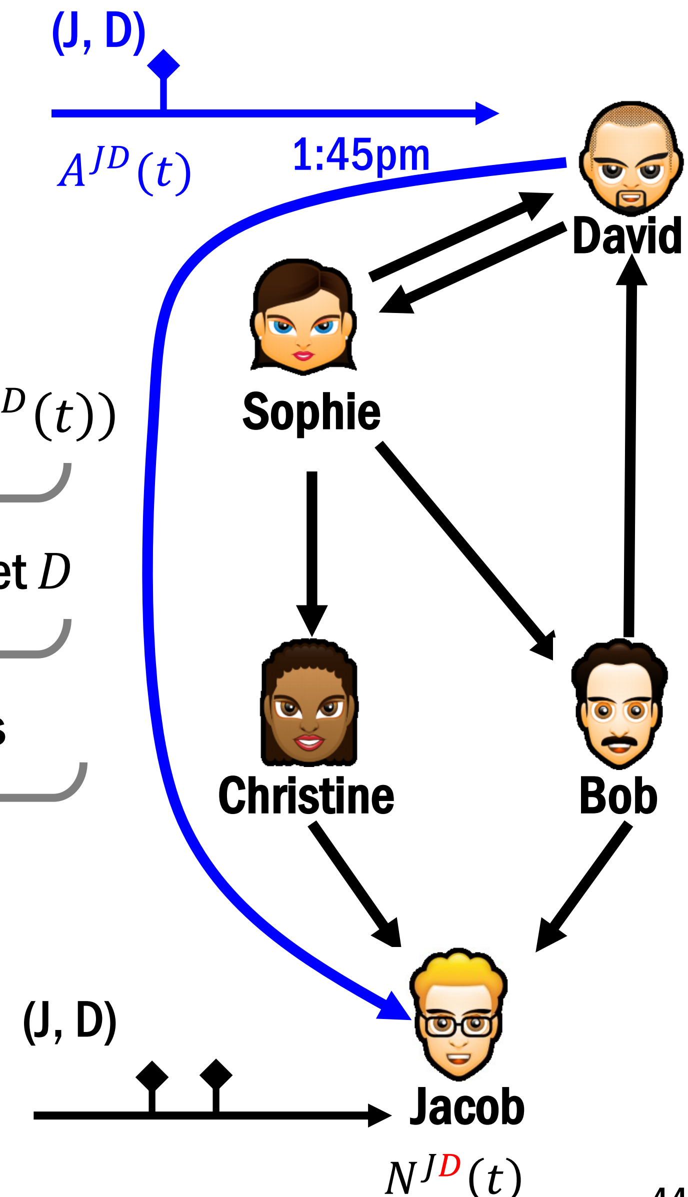
High if followees retweet frequently

$$h^{JD*}(t) = \underbrace{\beta^{D} A^{JB}(t) \exp(-|t|) \star dN^{BD}(t)}$$

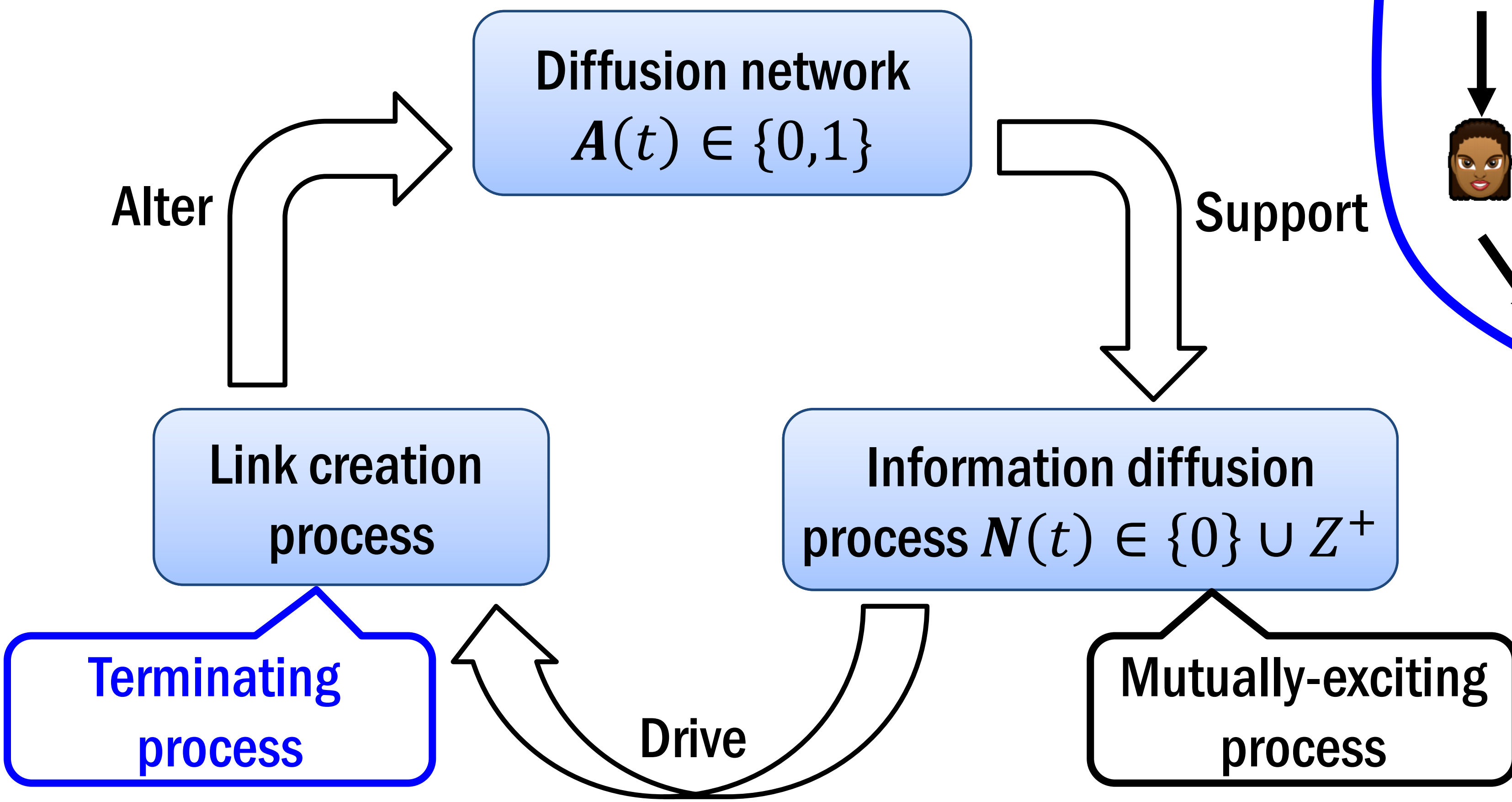
$$+ \beta^{D} A^{JC}(t) \exp(-|t|) \star dN^{CD}(t)$$

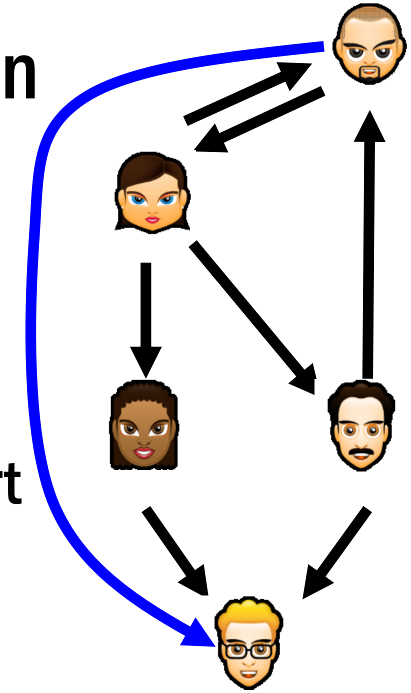
Information driven link creation

$$\gamma^{JD*}(t) = \underbrace{\underbrace{\left(1 - A^{JD}(t)\right)}_{\text{Check whether the link already there}} \cdot \left(\overbrace{\mu^J}^{J\text{'s random exploration}} + \underbrace{\alpha^D \exp(-|t|) \star \underbrace{dN^{JD}(t)}_{\text{Retweet } D}}_{\text{Self-exciting process}}\right)}_{\text{Terminating process no link and retweet often}}$$



Joint model of retweet + link creation



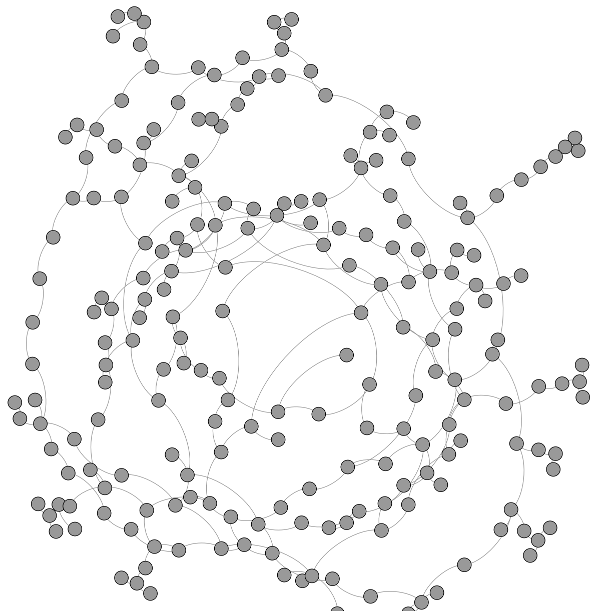


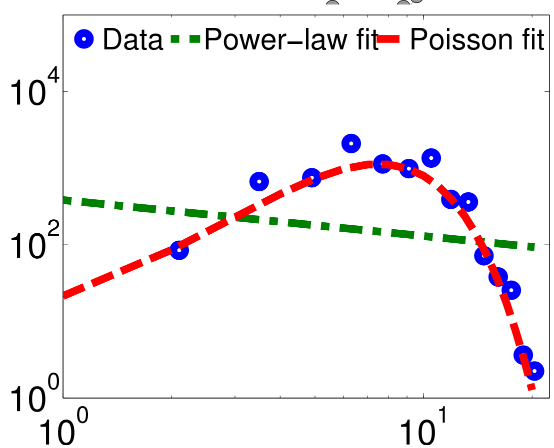
Simulation

Link creation parameter controls network type

$\alpha^D = 0$

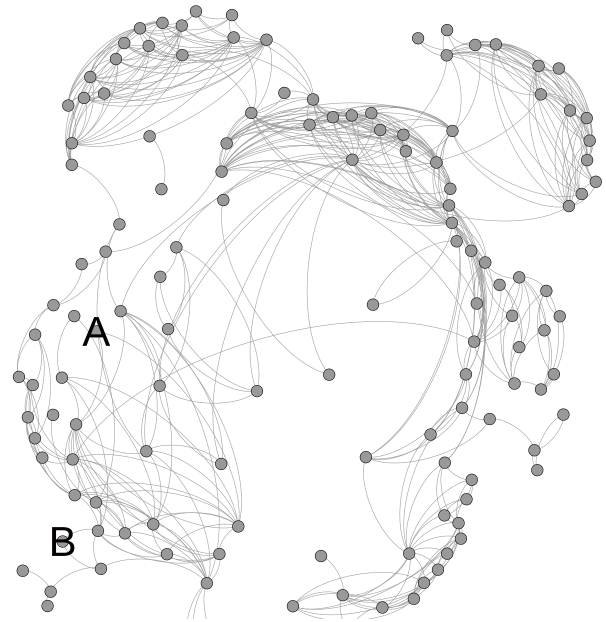
Erdos-Renyi random networks

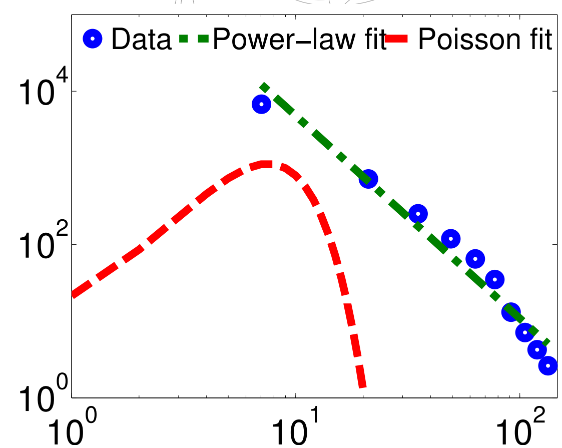




α^D large

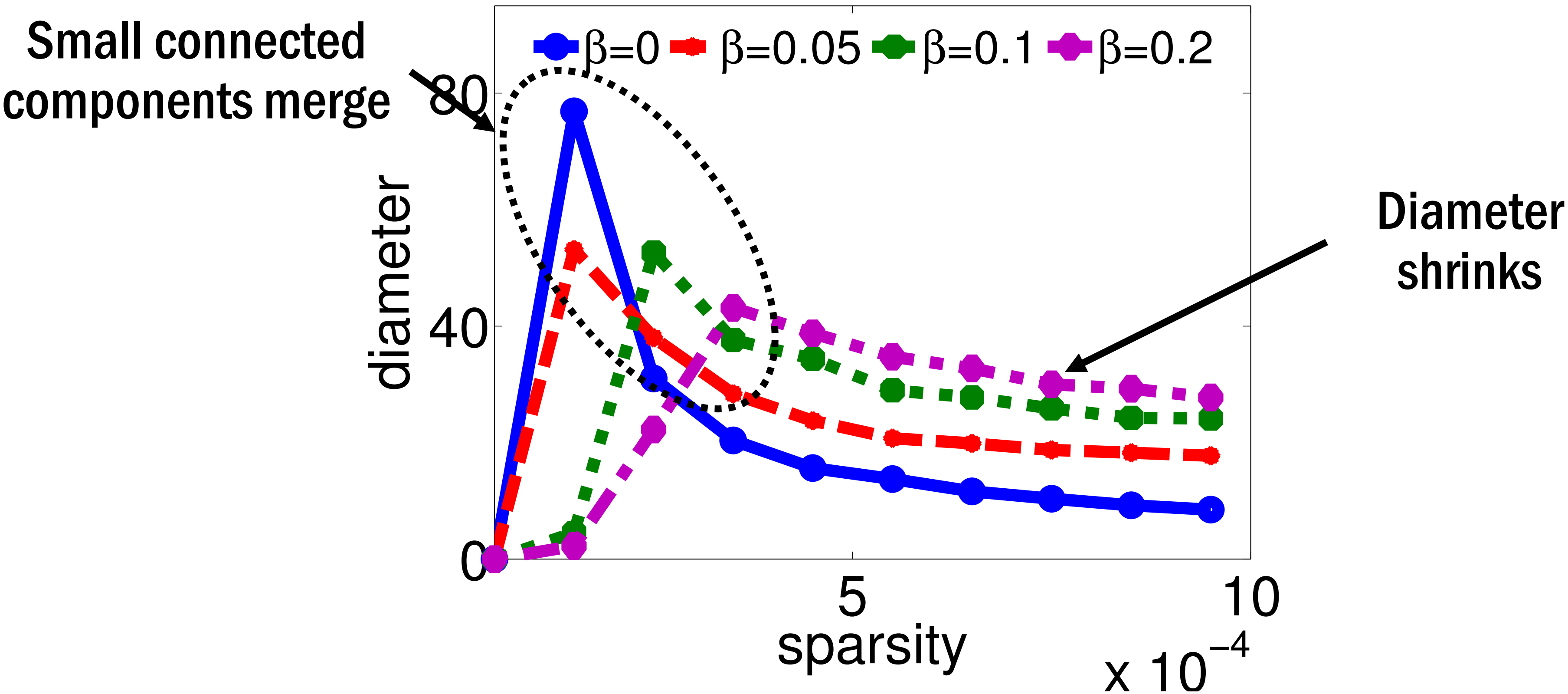
Scale-free networks





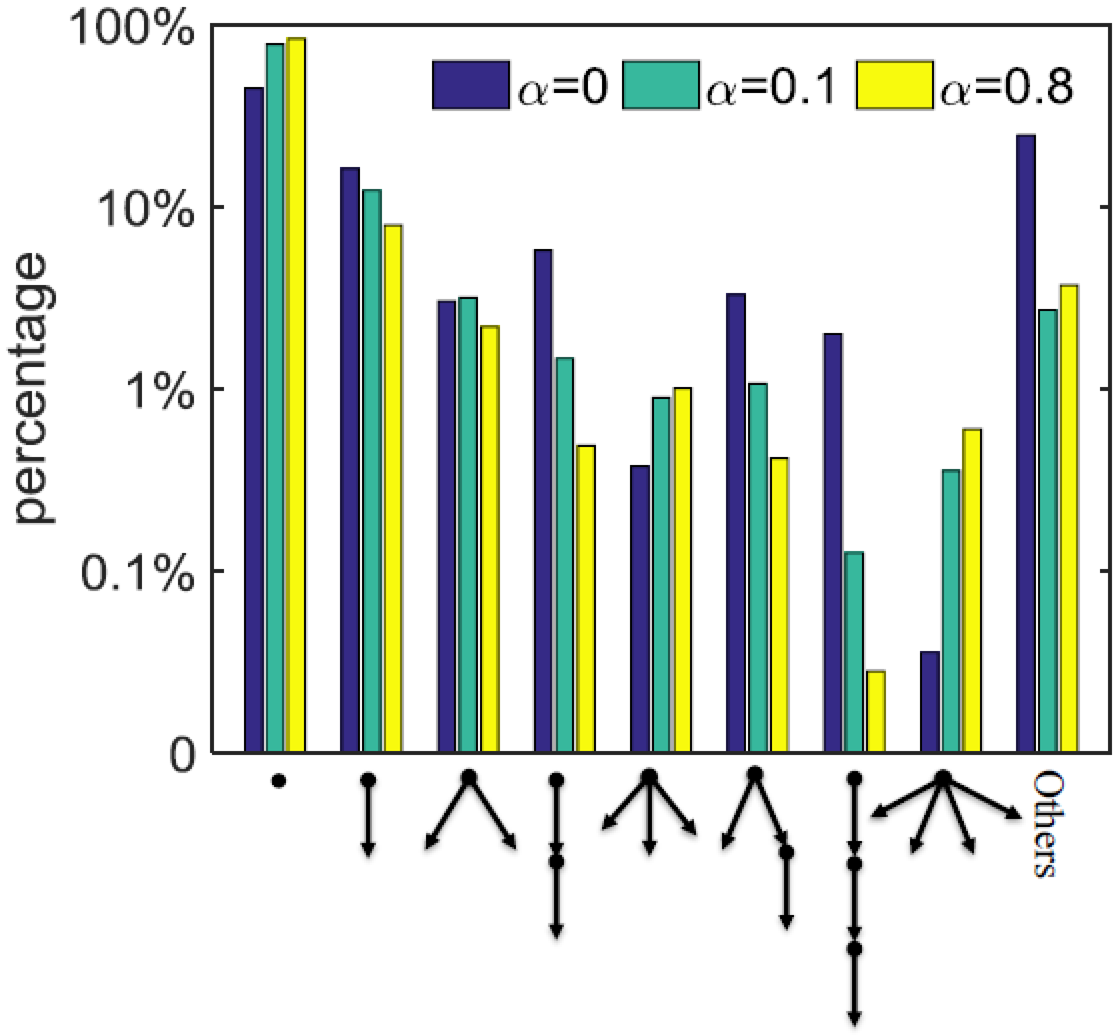
Shrinking network diameters

Generate networks with small shrinking diameter



Cascade patterns: structure

Generate short and fat cascades as α increases

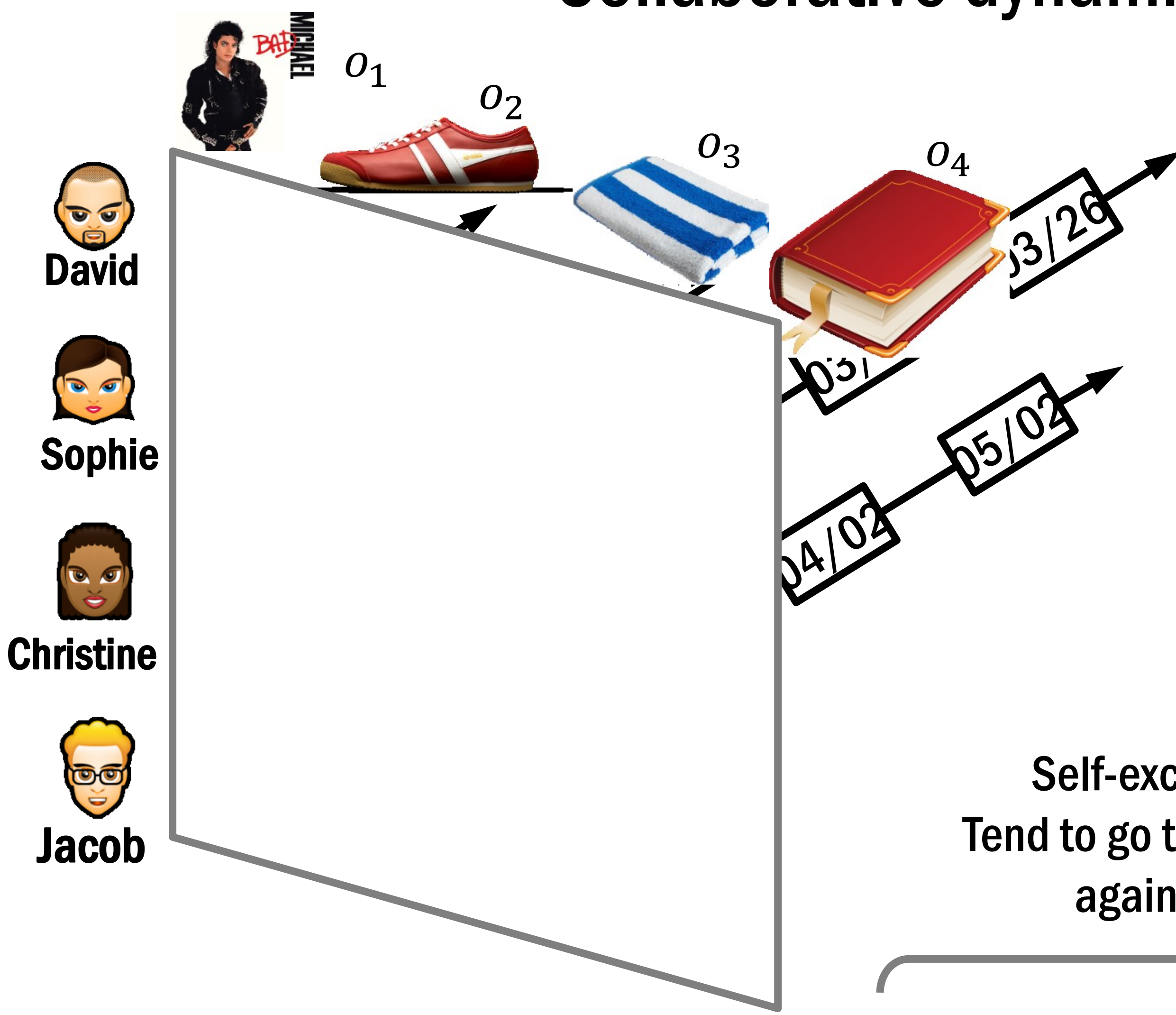


$\beta = 0.2$

Modeling:

Collaborative Dynamics

Collaborative dynamics



Low rank

$$\begin{pmatrix} \mu^{Do_1} & \cdots & \mu^{Do_4} \\ \vdots & \ddots & \vdots \\ \mu^{Jo_1} & \cdots & \mu^{Jo_4} \end{pmatrix}$$

$$\begin{pmatrix} \alpha^{Do_1} & \cdots & \alpha^{Do_4} \\ \vdots & \ddots & \vdots \\ \alpha^{Jo_1} & \cdots & \alpha^{Jo_4} \end{pmatrix}$$

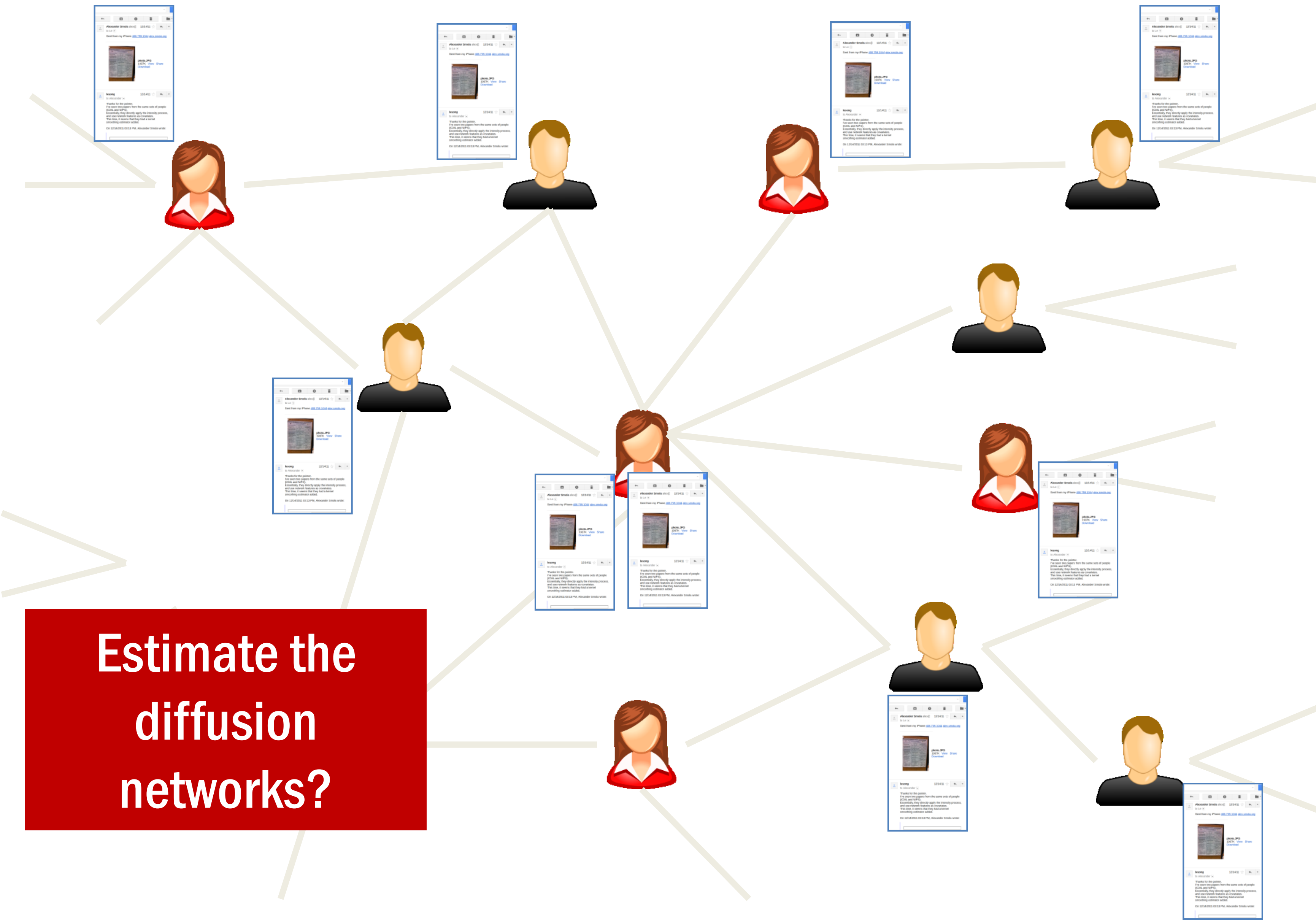
Self-exciting process
Tend to go to the same store
again and again

$$h^{Do_1*}(t) = \mu^{Do_1} + \alpha^{Do_1} \underbrace{\sum_{t_i^{Do_1} \in \mathcal{H}_t^{Do_1}} \exp\left(-|t - t_i^{Do_1}|\right)}$$

Learning:

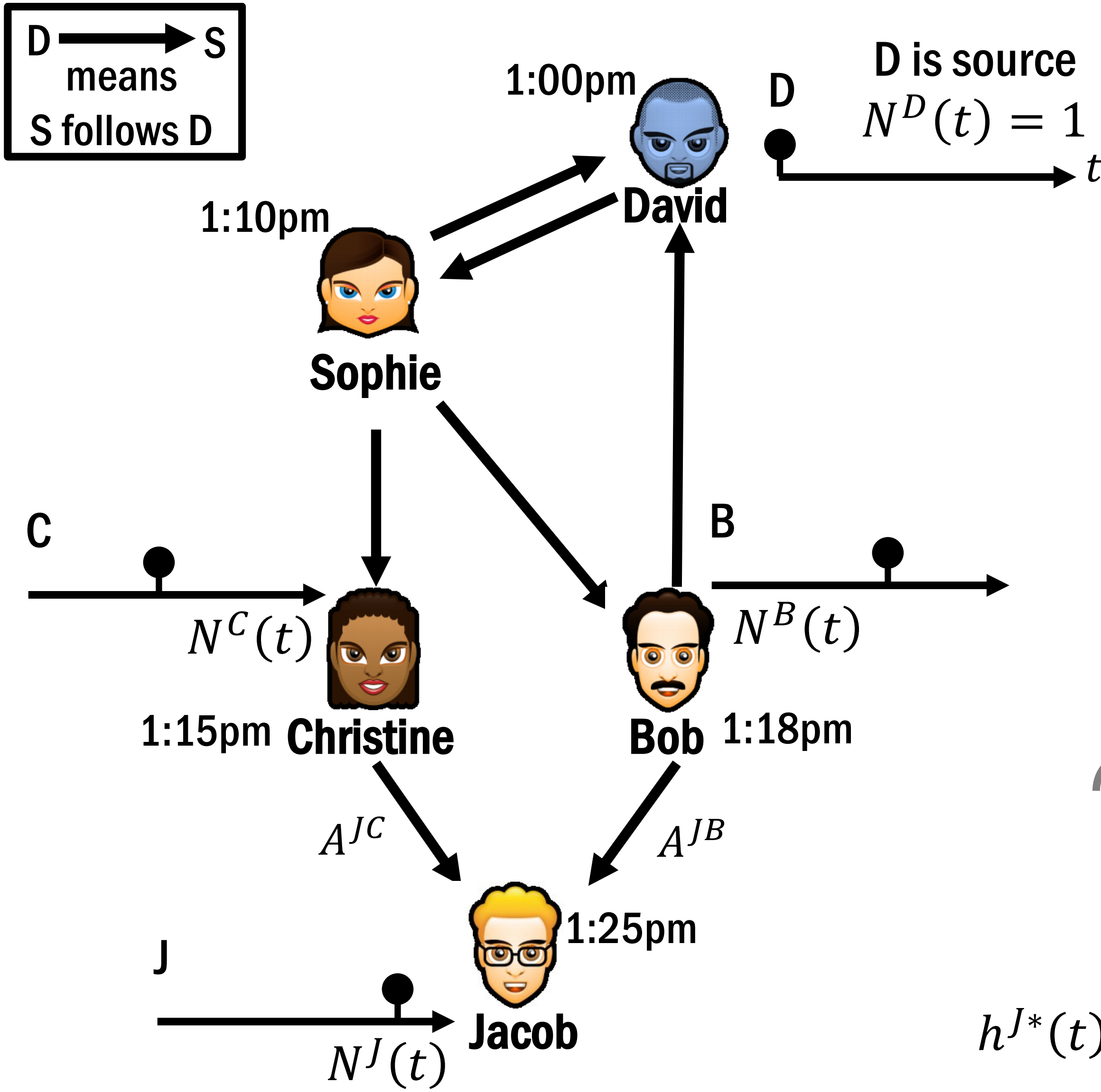
Sparse Networks

Hidden diffusion networks



Estimate the diffusion networks?

Parametrization of idea adoption model



Parametrization

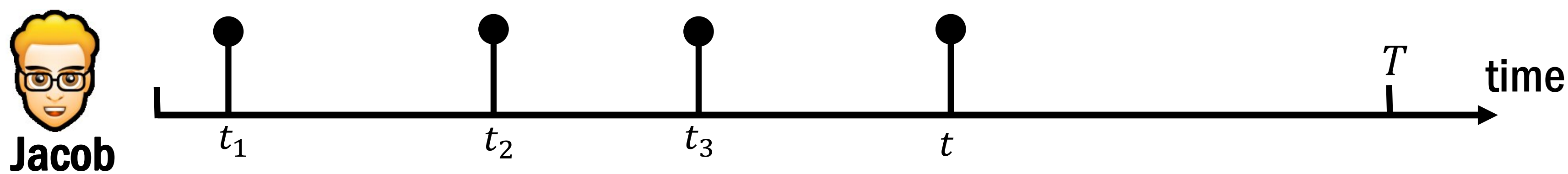
$$w = \begin{pmatrix} A^{JD} \\ A^{JS} \\ A^{JB} \\ A^{JC} \end{pmatrix}$$

$$h^{J*}(t) = \overbrace{\color{red}{A^{JB}} \overbrace{\left(1 - N^J(t)\right)}^{\text{Not yet adopted}} \overbrace{N^B(t)}^{\text{Followee adopted}}}^{\text{Terminating process adopt product only once}}$$

$$+ \color{red}{A^{JC}} \left(1 - N^J(t)\right) N^C(t)$$

ℓ_1 Regularized log-likelihood

$$L(w) + \lambda\|w\|_1 = \sum_{i=1}^{m} \log\langle w, \phi^*(t_i)\rangle - \langle w, \Psi^*(T)\rangle - \lambda\|w\|_1$$



Likelihood:

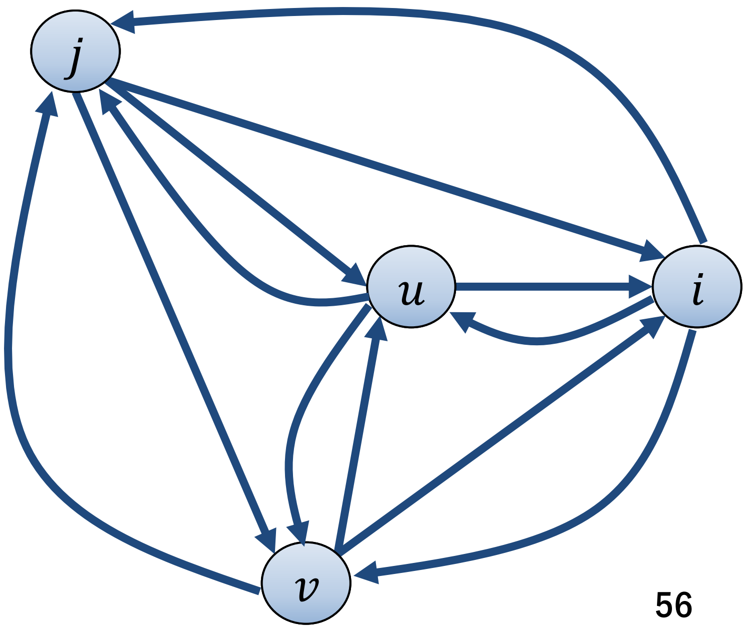
$$h^*(t_1) \quad h^*(t_2) \quad h^*(t_3) \quad h^*(t) \quad \exp\left(-\int_0^T h^*(\tau)d\tau\right)$$

$\langle w, \phi^*(t_1)\rangle$ $\quad$ $\langle w, \phi^*(t_2)\rangle$ $\quad$ $\langle w, \phi^*(t_3)\rangle$ $\quad$ $\langle w, \phi^*(t)\rangle$ $\quad$ $\exp\left(-\int_0^T \langle w, \phi^*(\tau)\rangle d\tau\right)$

Soft-thresholding algorithm

ℓ_1-reguarlized likelihood estimation problem. Solve one such problem for each node.

- Set learning rate β
- $k = 0$
- Initialize w
- While $k \le K$, do
  - $w^{k+1} = \left(w^k - \beta \cdot \nabla_w L\left(w^k\right) - \lambda \cdot \beta\right)_+$
  - $k = k + 1$
- End while

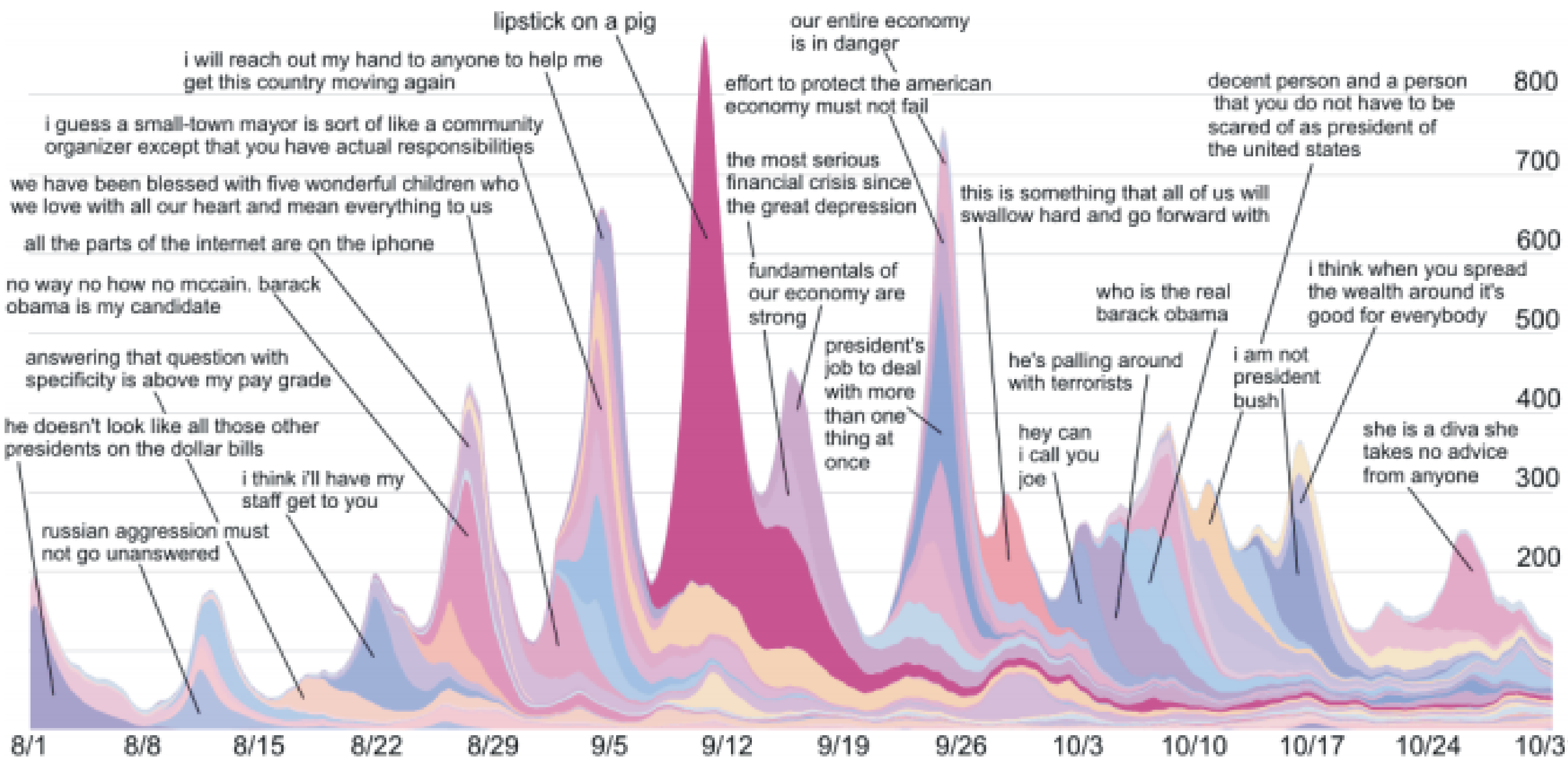


Statistical guarantees

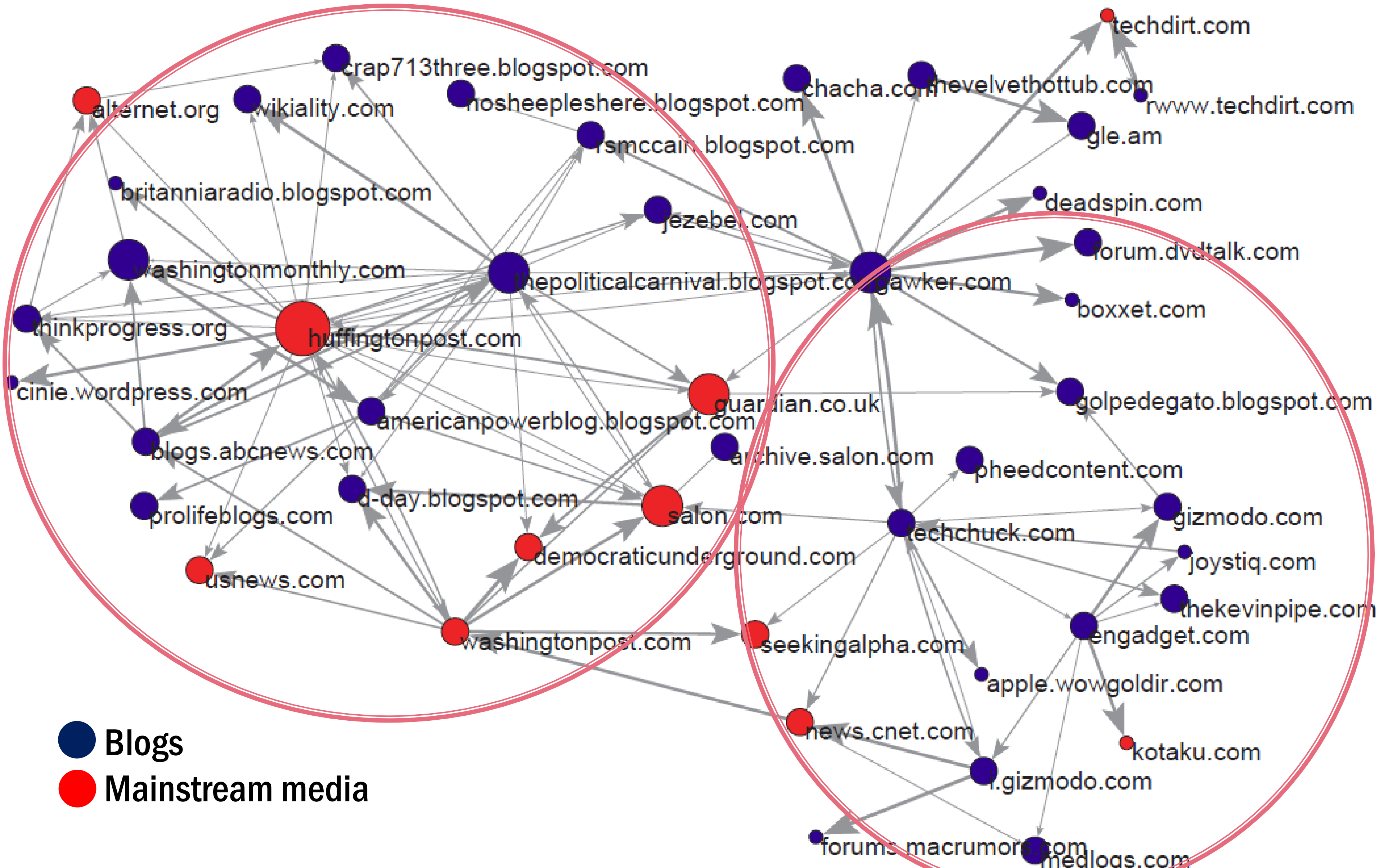
- Recovery conditions:
  - Eigenvalue of the Hessian, $Q = \nabla_w^2 L$, is bounded $[C_{min}, C_{max}]$
  - Gradient is upper bounded, $\|\nabla_w L\|_\infty \leq C_1$
  - Hazard is lower bounded, $\min w_j \geq C_2$
  - Incoherence condition: $\|Q_{S^c S}(Q_{SS})^{-1}\|_\infty \leq 1 - \varepsilon$
    - network structure
    - parameter value
    - observation window
    - source node distribution

- Given $n > C_3 \cdot d^3 \log p$ cascades, set regularization parameter $\lambda \geq C_4 \cdot \frac{2-\varepsilon}{\varepsilon} \sqrt{\frac{\log p}{n}}$, the network structure can be recovered with probability at least $1 - 2\exp(-C'' \lambda^2 n)$

Memetracker

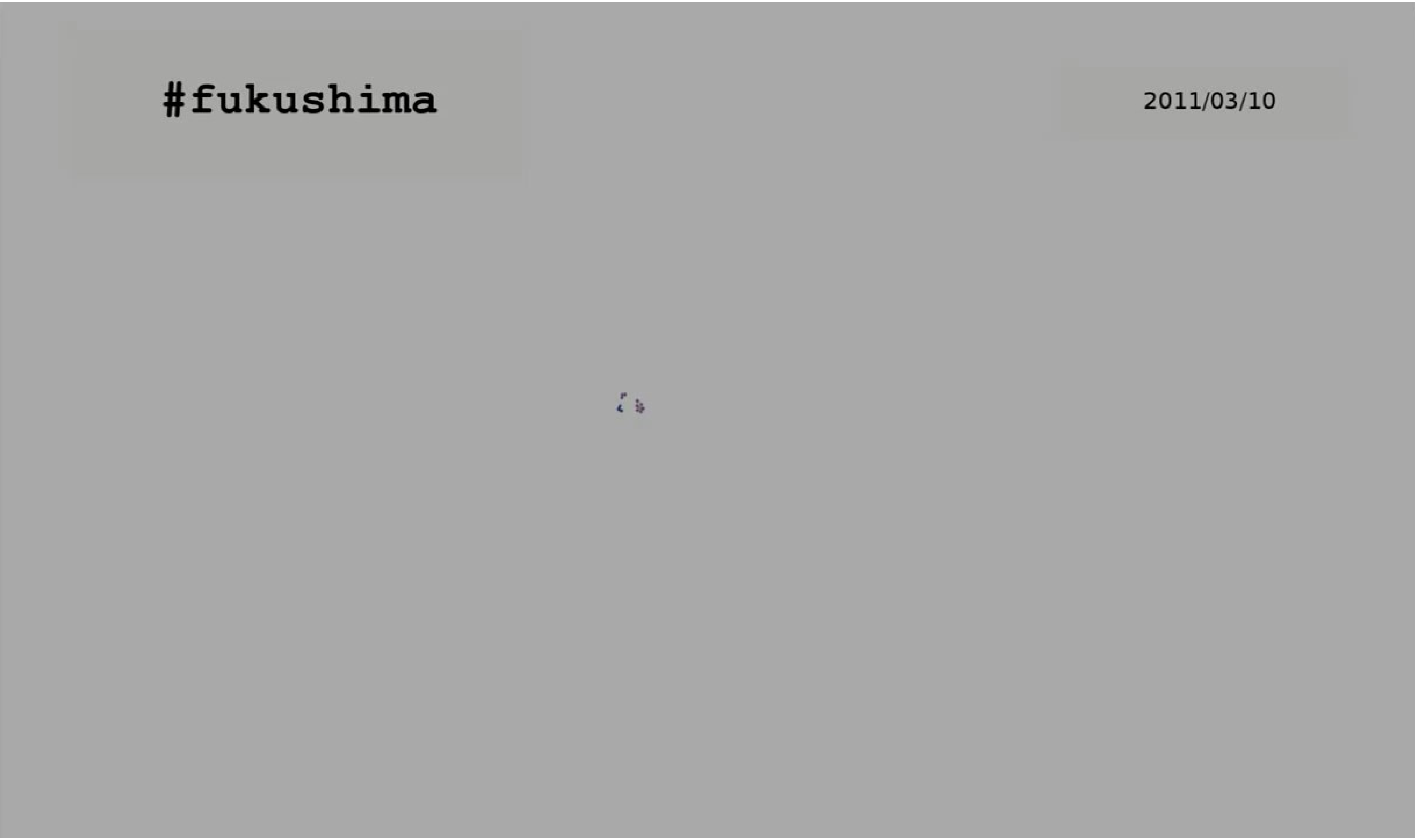


Estimated diffusion network



Nan et al. NIPS 2012

Tracking diffusion networks

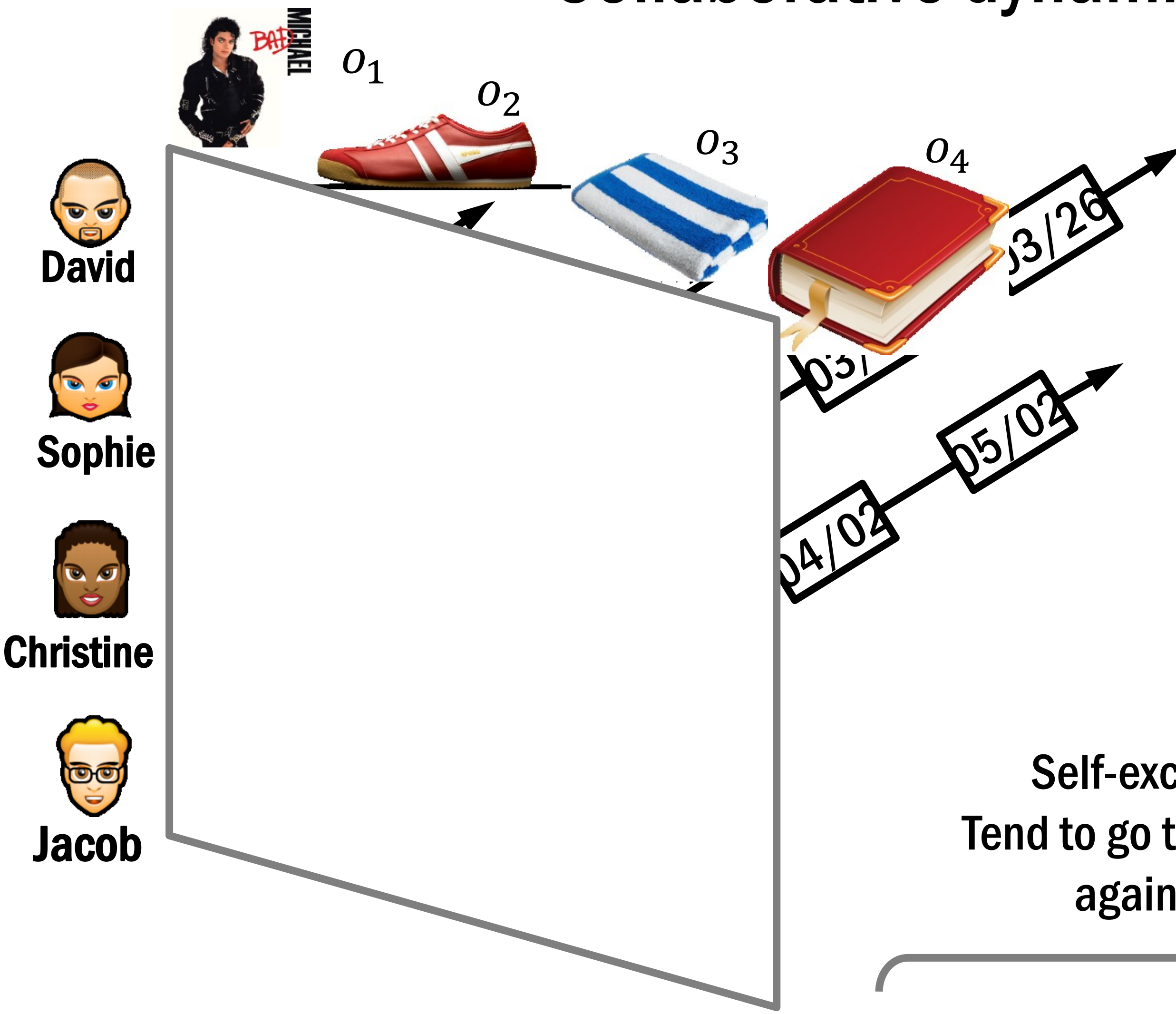


Dynamic Processes over Information Networks
Representation, Modeling, Learning and Inference

Learning:

Low Rank Collaborative Dynamics

Collaborative dynamics



Regularization

$$\left\| \begin{pmatrix} \mu^{Do_1} & \cdots & \mu^{Do_4} \\ \vdots & \ddots & \vdots \\ \mu^{Jo_1} & \cdots & \mu^{Jo_4} \end{pmatrix} \right\|_*$$

$$\left\| \begin{pmatrix} \alpha^{Do_1} & \cdots & \alpha^{Do_4} \\ \vdots & \ddots & \vdots \\ \alpha^{Jo_1} & \cdots & \alpha^{Jo_4} \end{pmatrix} \right\|_*$$

Self-exciting process
Tend to go to the same store
again and again

$$h^{Do_1*}(t) = \mu^{Do_1} + \alpha^{Do_1} \overbrace{\sum_{t_i^{Do_1} \in \mathcal{H}_t^{Do_1}} \exp(-|t - t_i^{Do_1}|)}$$

Dynamic Processes over Information Networks
Representation, Modeling, Learning and Inference

Learning:

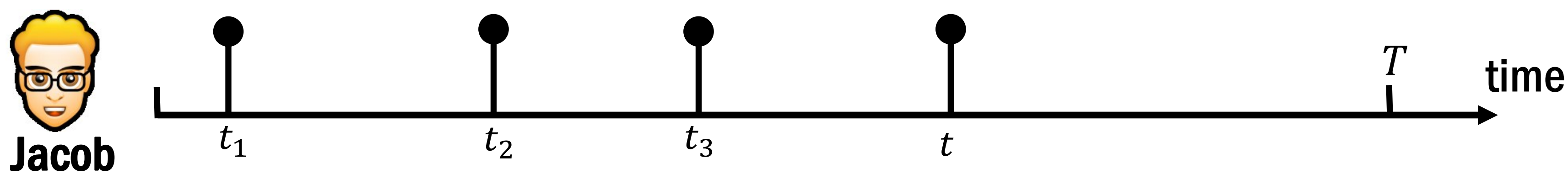
Generic Algorithm

Concave log-likelihood of event sequence

Log-likelihood

$$L(w) = \sum_{i=1}^{m} \log\langle w, \phi^*(t_i)\rangle - \langle w, \Psi^*(T)\rangle$$

Concave in w!



Likelihood:

$$h^*(t_1) \quad h^*(t_2) \quad h^*(t_3) \quad h^*(t) \quad \exp\left(-\int_0^T h^*(\tau)d\tau\right)$$

$\langle w, \phi^*(t_1)\rangle$

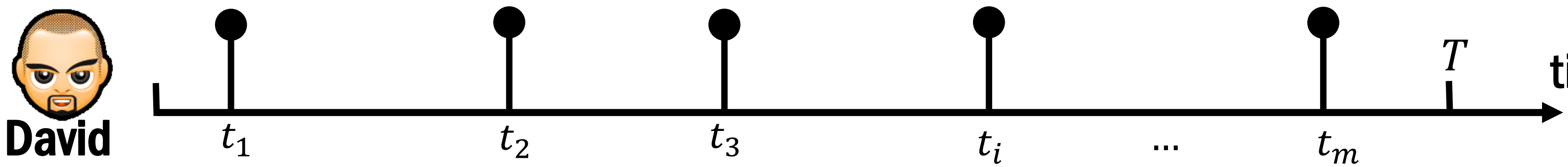
$\langle w, \phi^*(t_2)\rangle$

$\langle w, \phi^*(t_3)\rangle$

$\langle w, \phi^*(t)\rangle$

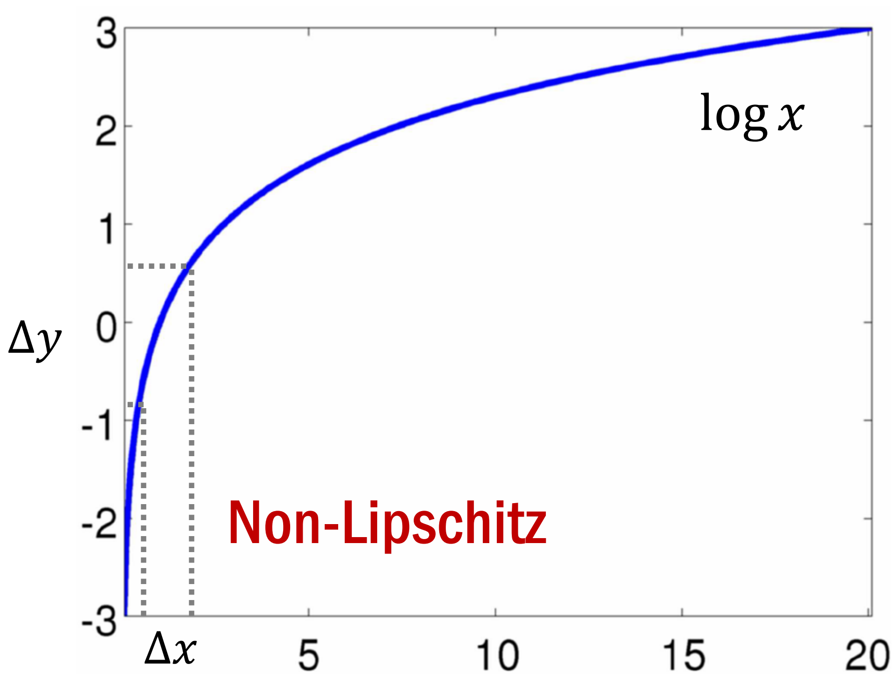
$$\exp\left(-\int_0^T \langle w, \phi^*(\tau)\rangle d\tau\right)$$

Challenge in optimization problem



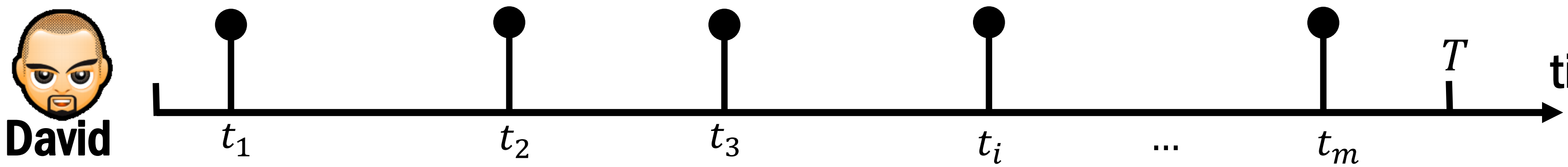
Negative log-likelihood

$$\min_{w \in \mathbb{R}_+^n} \langle w, \Psi^*(T) \rangle - \sum_{i=1}^{m} \log \langle w, \phi^*(t_i) \rangle + \lambda \, \|w\|_1$$



Existing first order methods
$O\left(\frac{1}{\epsilon^2}\right)$ iterations

Saddle point reformulation



Negative log-likelihood

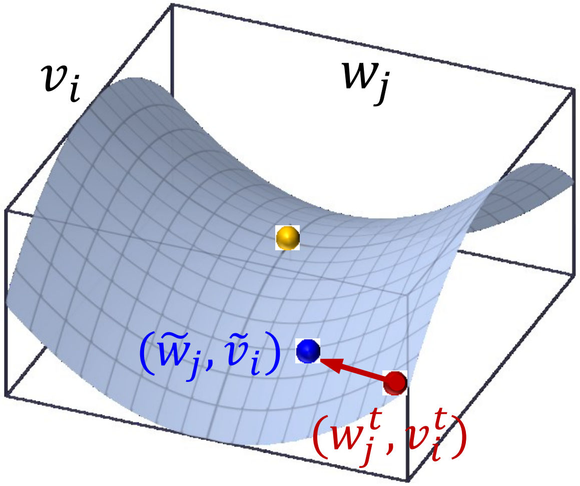
$$\min_{w \in \mathbb{R}_+^n} \langle w, \Psi^*(T) \rangle - \sum_{i=1}^{m} \log \langle w, \phi^*(t_i) \rangle + \lambda \, \|w\|_1$$

Fenchel dual

$$\max_{v_i > 0} v_i \langle w, \phi^*(t_i) \rangle - \log v_i - 1$$

$$\min_{w \in \mathbb{R}_+^n} \max_{\{v_i > 0\}_{i=1}^m} \langle w, \Psi^*(T) \rangle - \sum_{i=1}^{m} v_i \langle w, \phi^*(t_i) \rangle + \sum_{i=1}^{m} \log v_i + \lambda \, \|w\|_1$$

He et al. Arxiv 2016



Proximal gradient

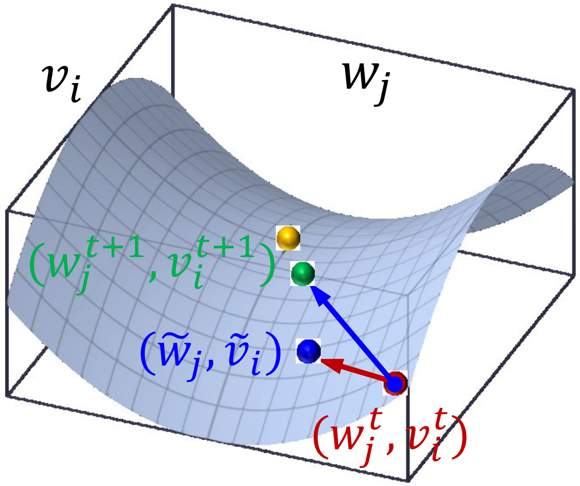
$$\bar{w}_j = w_j^t - \gamma \, \nabla_{w_j} L\left(w^t, \{v_i^t\}\right)$$
$$\bar{v}_i = v_i^t + \gamma \, \nabla_{v_i} L\left(w^t, \{v_i^t\}\right)$$

$$\tilde{w}_j = \left(\bar{w}_j - \lambda\gamma\right)_+$$
$$\tilde{v}_i = \frac{\bar{v}_i + \left(\bar{v}_i^2 + 4\gamma\right)^{1/2}}{2}$$

Given current $w^t, \{v_i^t\}$

$$\min_{w \in \mathbb{R}_+^n} \max_{\{v_i > 0\}_{i=1}^m} \underbrace{\langle w, \Psi^*(T) \rangle - \sum_{i=1}^m v_i \langle w, \phi^*(t_i) \rangle}_{\text{Bilinear form } L(w, \{v_i\})} + \sum_{i=1}^m \log v_i + \lambda \, \|w\|_1$$

celerated proximal gradient



$$\bar{w}_j = w_j^t - \gamma \nabla_{w_j} L(\tilde{w}, \{\tilde{v}_i\})$$

$$\bar{v}_i = v_i^t + \gamma \nabla_{v_i} L(\tilde{w}, \{\tilde{v}_i\})$$

$$w^{t+1} = (\bar{w}_j - \lambda\gamma)_+$$

$$v_i^{t+1} = \frac{\bar{v}_i + (\bar{v}_i^2 + 4\gamma)^{1/2}}{2}$$

$$\bar{w}_j = w_j^t - \gamma \nabla_{w_j} L(w^t, \{v_i^t\})$$

$$\bar{v}_i = v_i^t + \gamma \nabla_{v_i} L(w^t, \{v_i^t\})$$

$$\tilde{w}_j = (\bar{w}_j - \lambda\gamma)_+$$

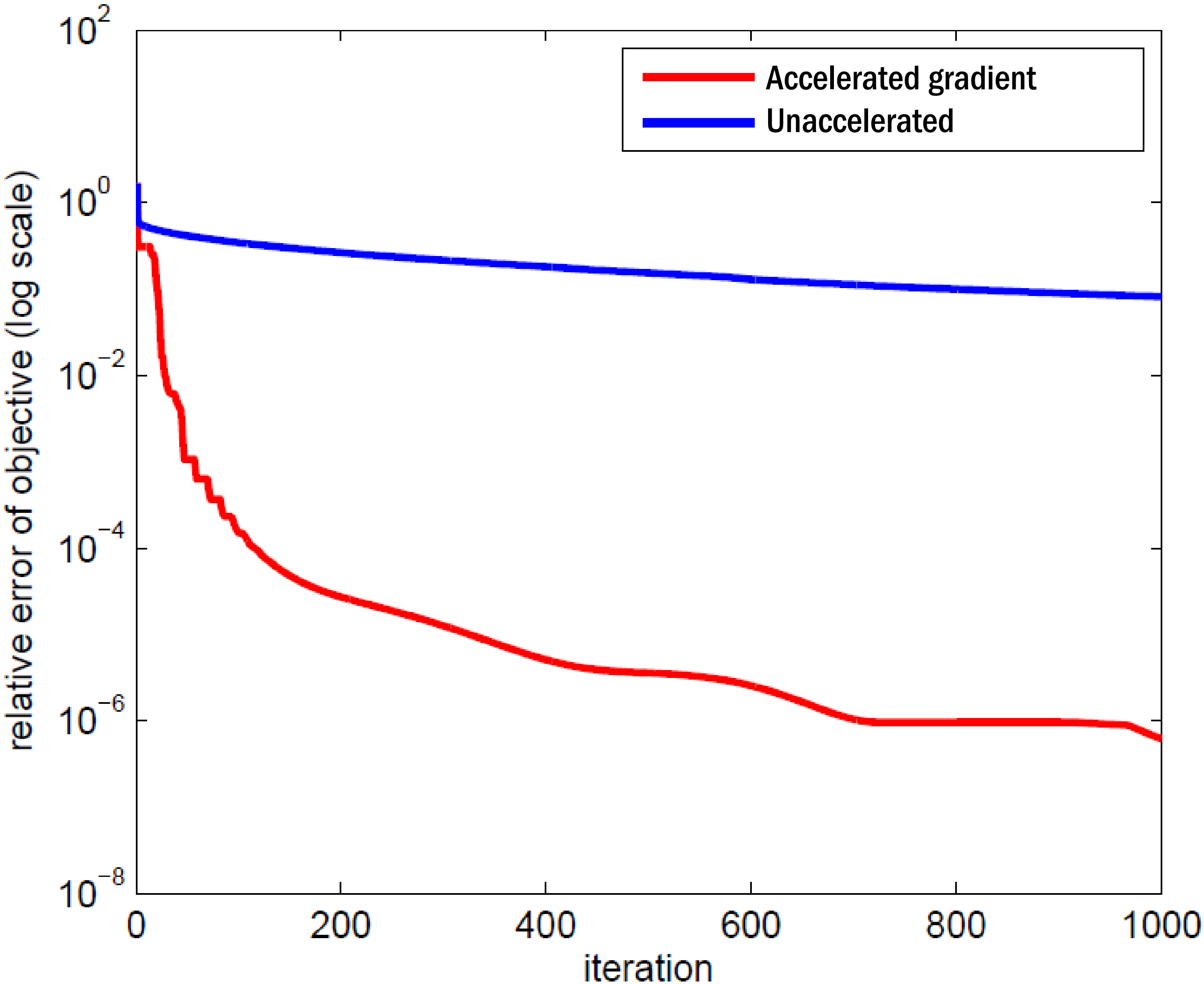
$$\tilde{v}_i = \frac{\bar{v}_i + (\bar{v}_i^2 + 4\gamma)^{1/2}}{2}$$

Given current $w^t, \{v_i^t\}$

$O\left(\frac{1}{\epsilon}\right)$ iterations

$$\min_{w \in \mathbb{R}_+^n} \max_{\{v_i > 0\}_{i=1}^m} \underbrace{\langle w, \Psi^*(T)\rangle - \sum_{i=1}^m v_i \langle w, \phi^*(t_i)\rangle}_{\text{Bilinear form } L(w, \{v_i\})} + \sum_{i=1}^m \log v_i + \lambda \|w\|_1$$

Converge much faster

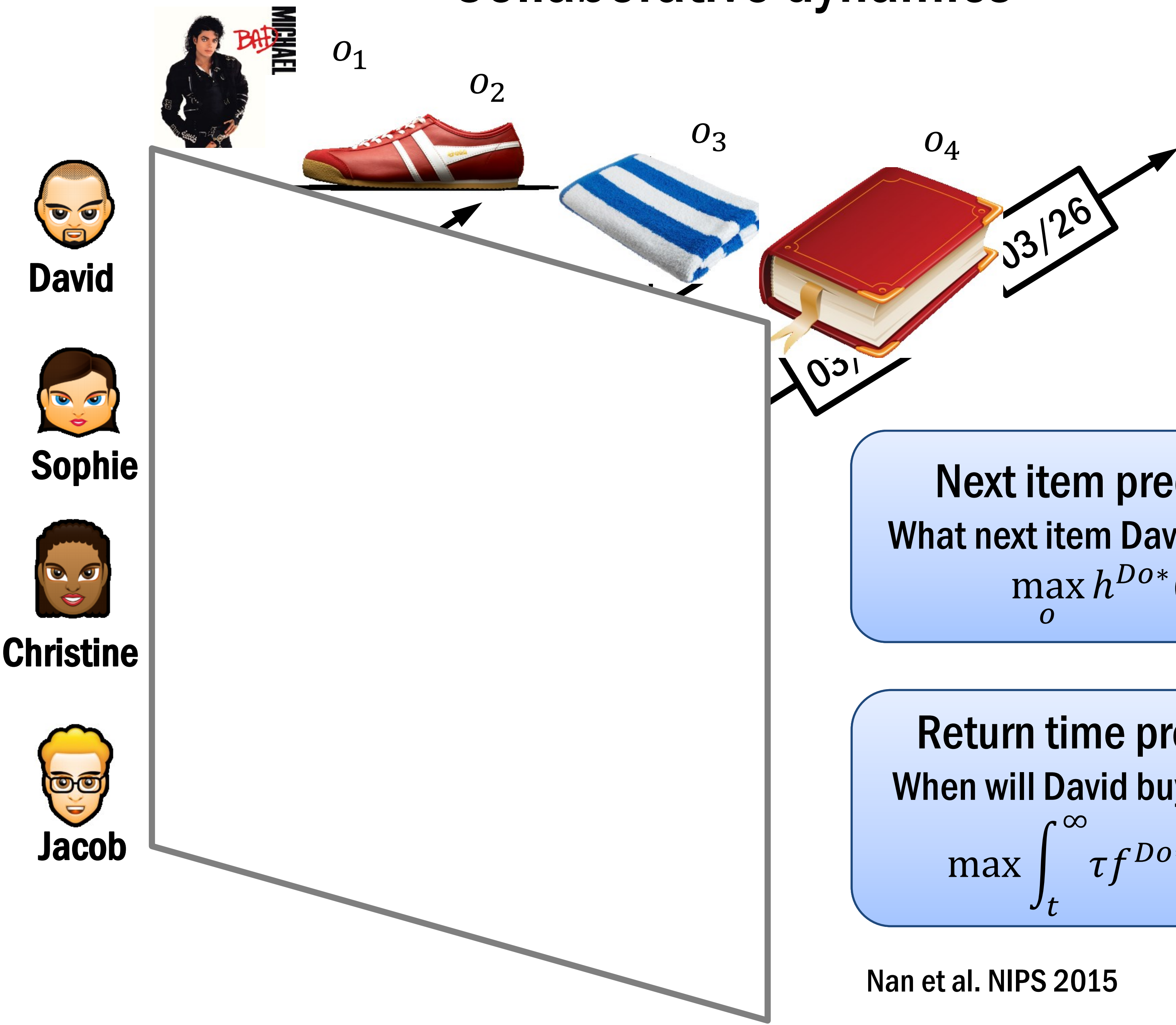


Dynamic Processes over Information Networks
Representation, Modeling, Learning and Inference

Inference:

Time-Sensitive Recommendation

Collaborative dynamics



Next item prediction

What next item David will buy?

$$\max_{o} h^{Do*}(t)$$

Return time prediction

When will David buy the item?

$$\max \int_{t}^{\infty} \tau f^{Do*}(\tau) d\tau$$

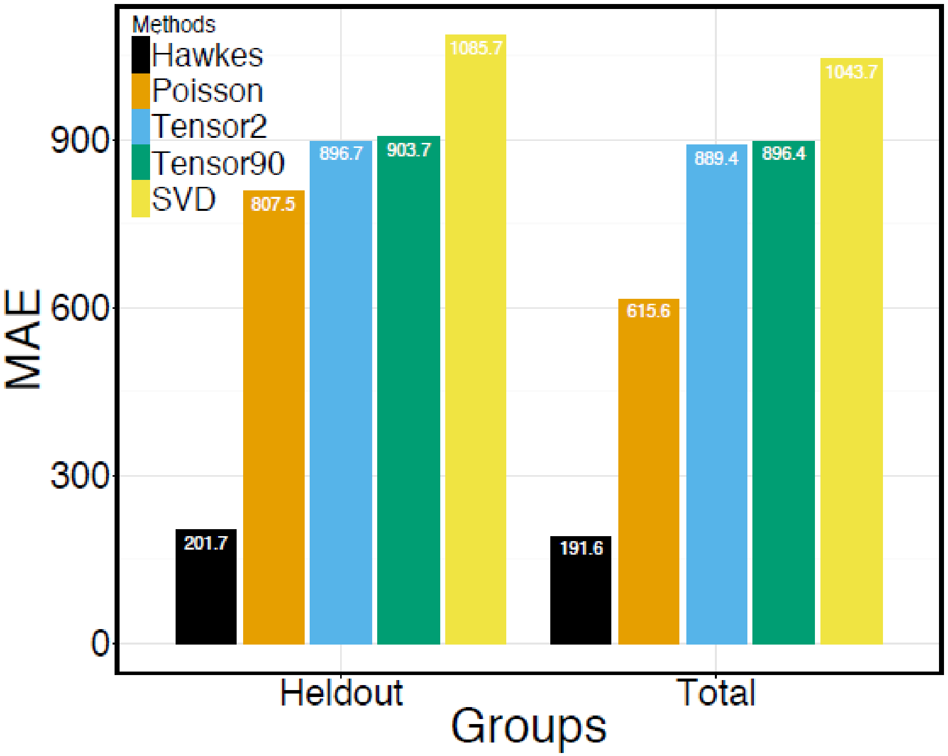
Nan et al. NIPS 2015

Music recommendation for Last.fm

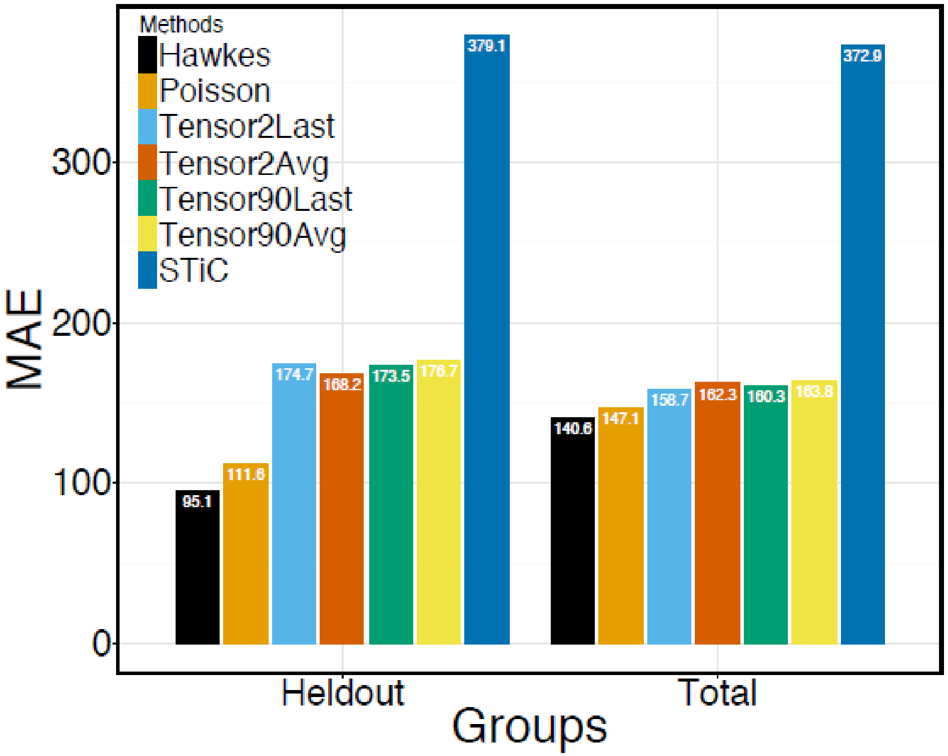
Online records of music listening. The time unit is hour

1000 users, 3000 albums

20,000 observed pairs, more than 1 million events



Album prediction



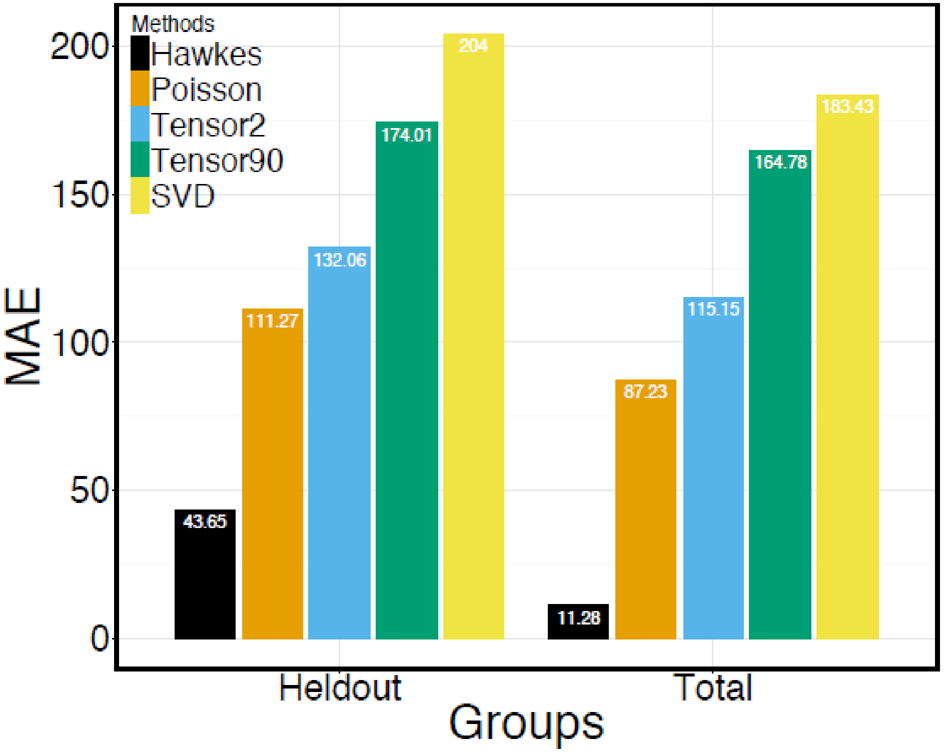
Returning time prediction

Electronic healthcare records

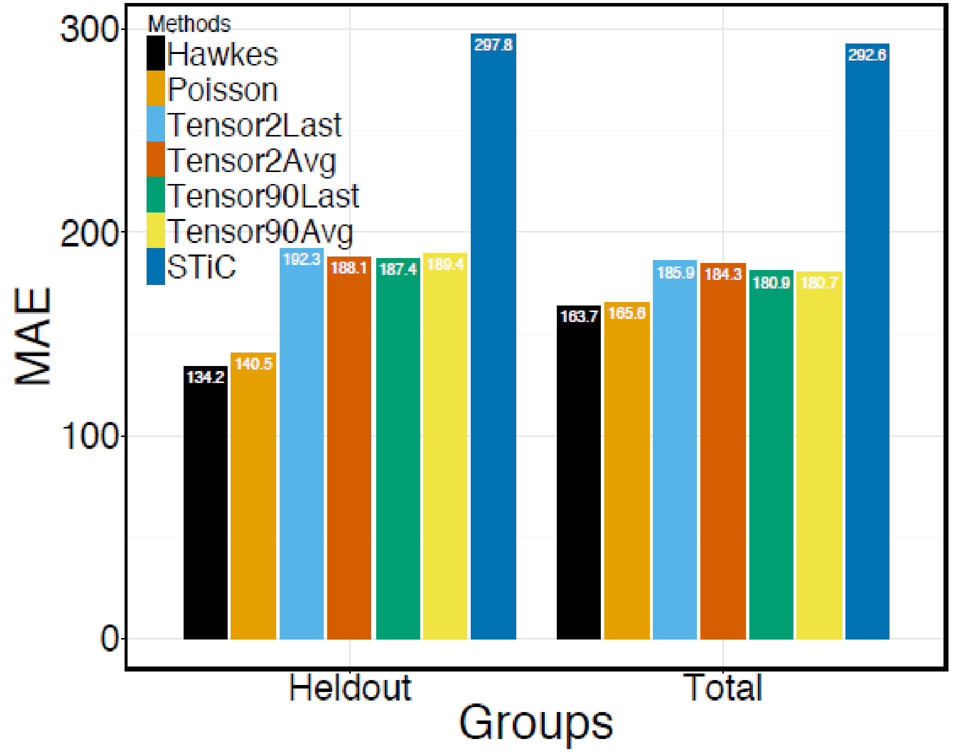
MIMIC II dataset: a collection of de-identified clinical visit records

The time unit is week

650 patients and 204 disease codes



Diagnosis code prediction



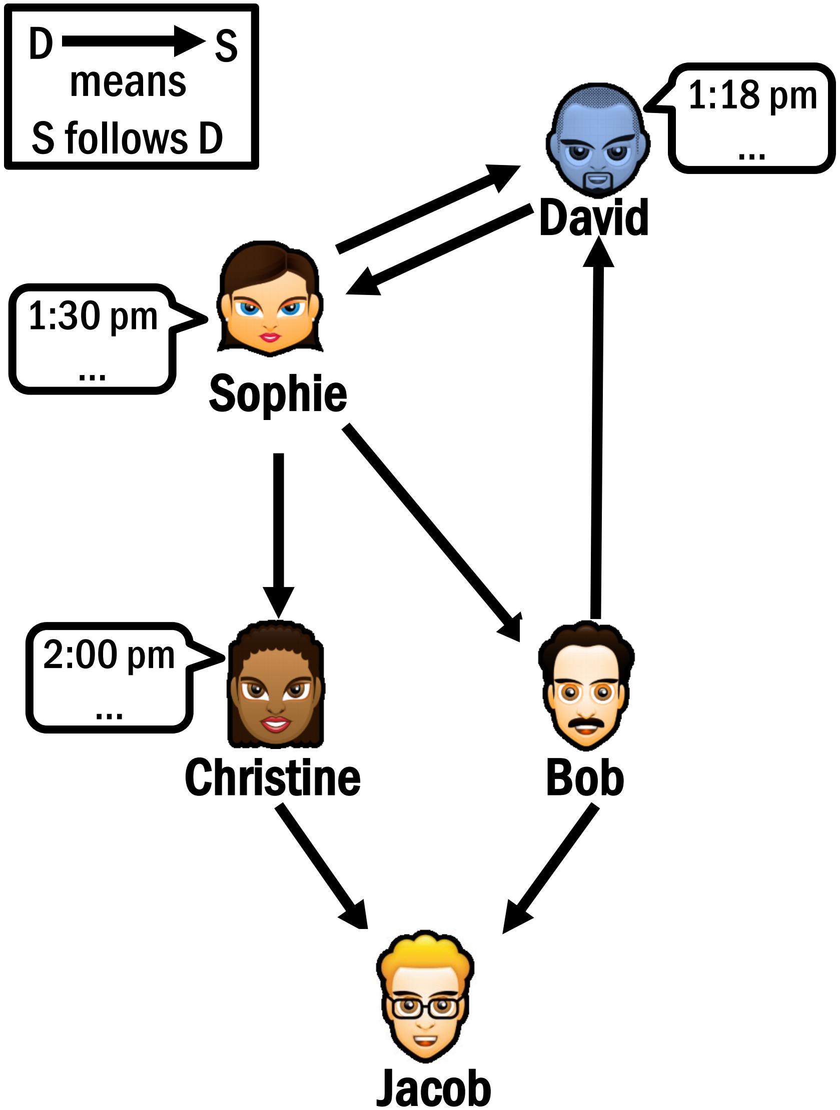
Returning time prediction

Dynamic Processes over Information Networks
Representation, Modeling, Learning and Inference

Inference:

Influence Maximization

Inference in dea adoption



Influence estimation

Can a piece of news spread, in 1 month, to a million user?

$$\sigma(s,t) := \mathbb{E}\left[\sum_{i \in V} N^i(t)\right]$$

Influence maximization

Who is the most influential user?

$$\max_{s \in V} \sigma(s,t)$$

Source localization

Where is the origin of information?

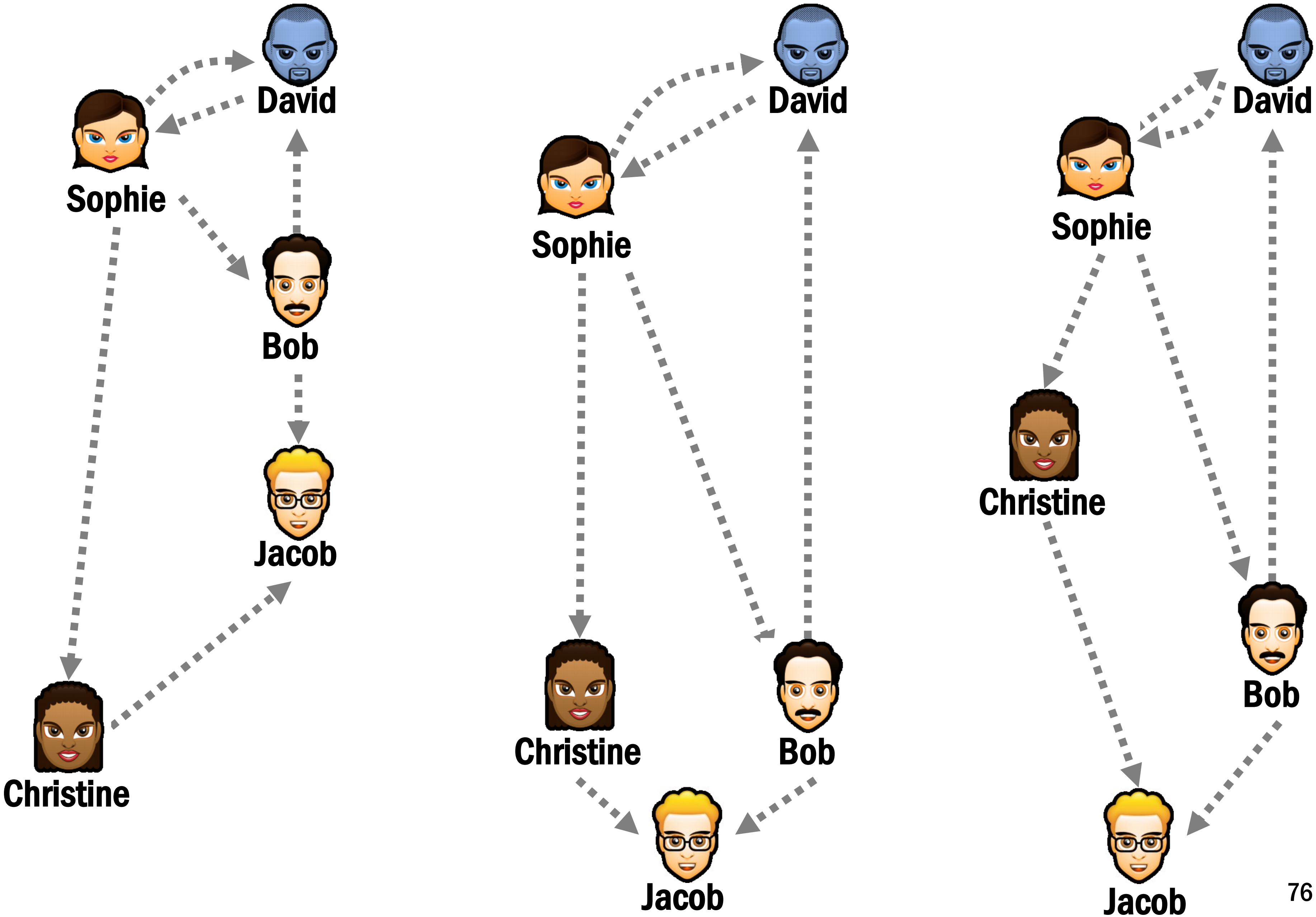
$$\max_{s \in V, t \in [0,T]} \text{Likelihood(partial cascade)}$$

Rodriguez et al. ICML 2012

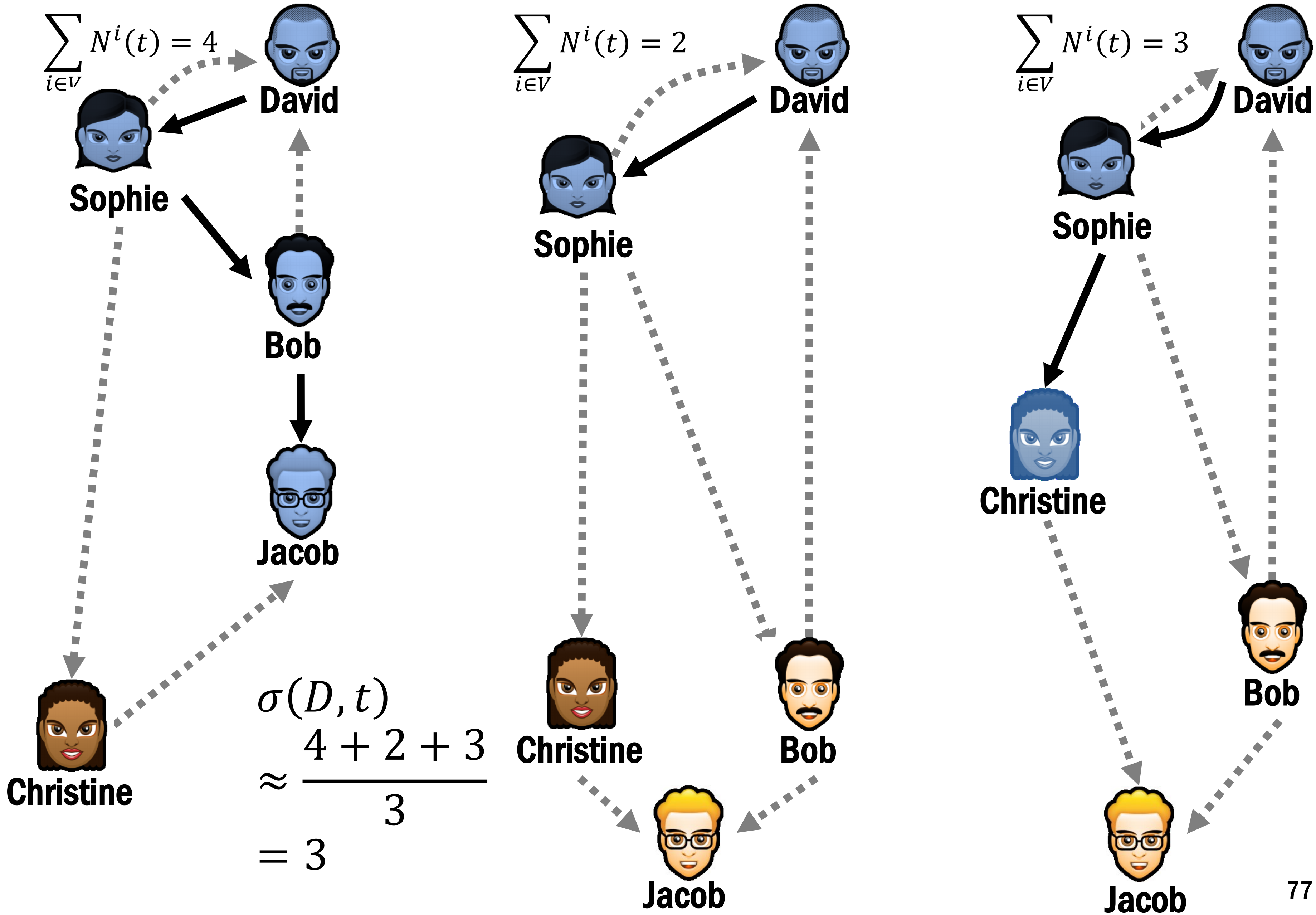
Nan et al. NIPS 2013

Farajtabar et al. AISTATS 2015

Cascades from D in 1 month

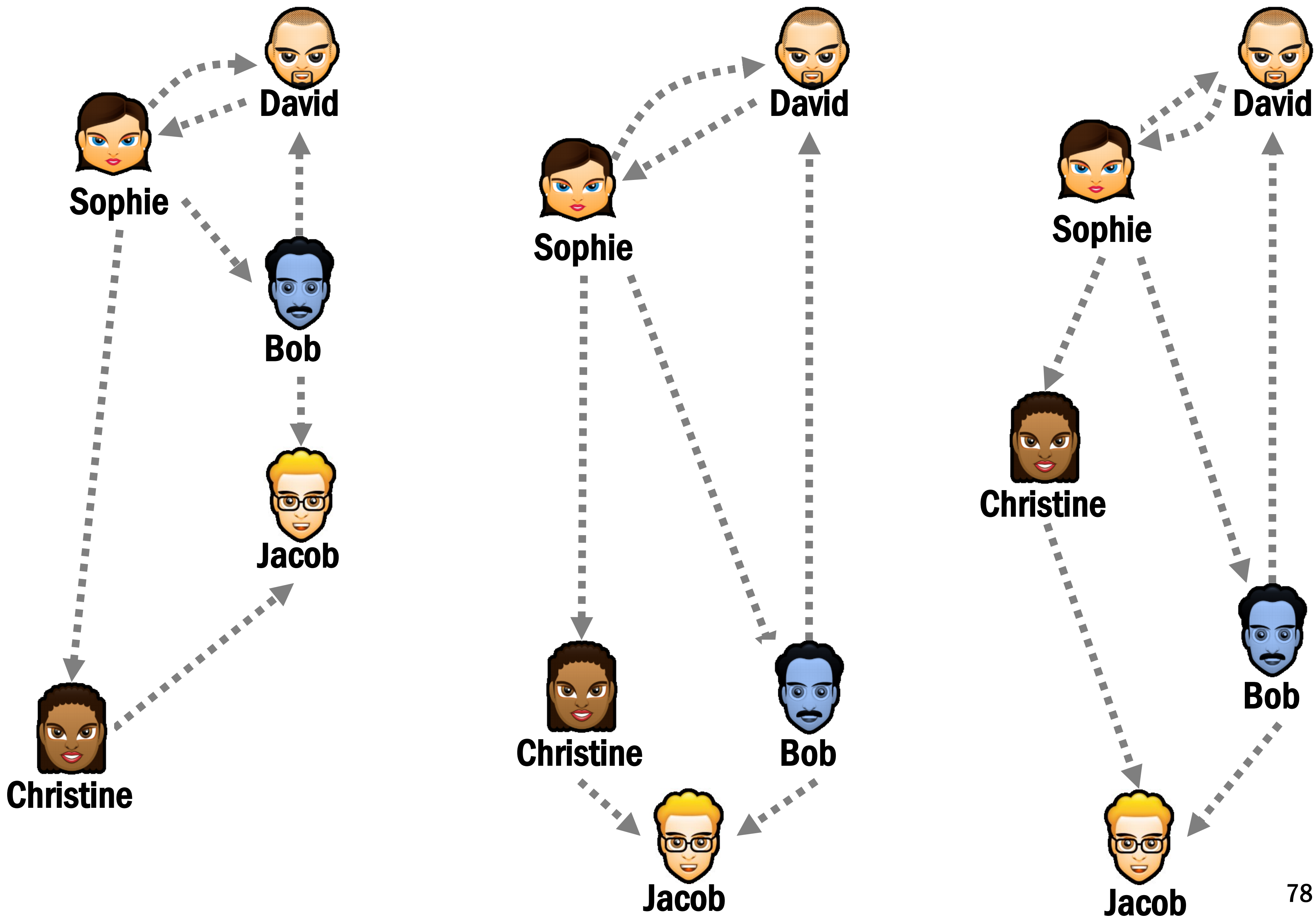


Cascades from D in 1 month

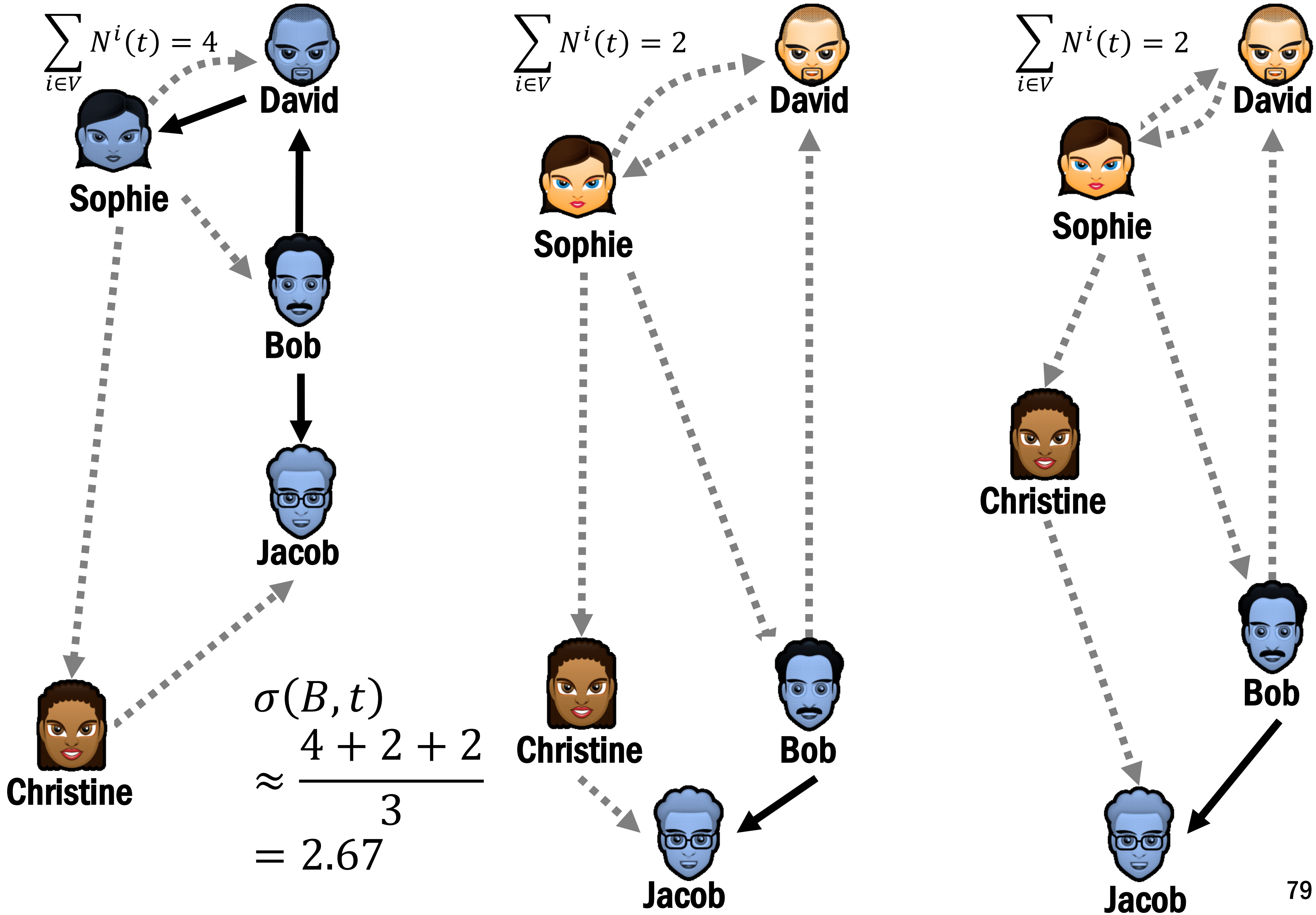


$$\sigma(D,t) \approx \frac{4+2+3}{3} = 3$$

Cascades from B in 1 month



Cascades from B in 1 month



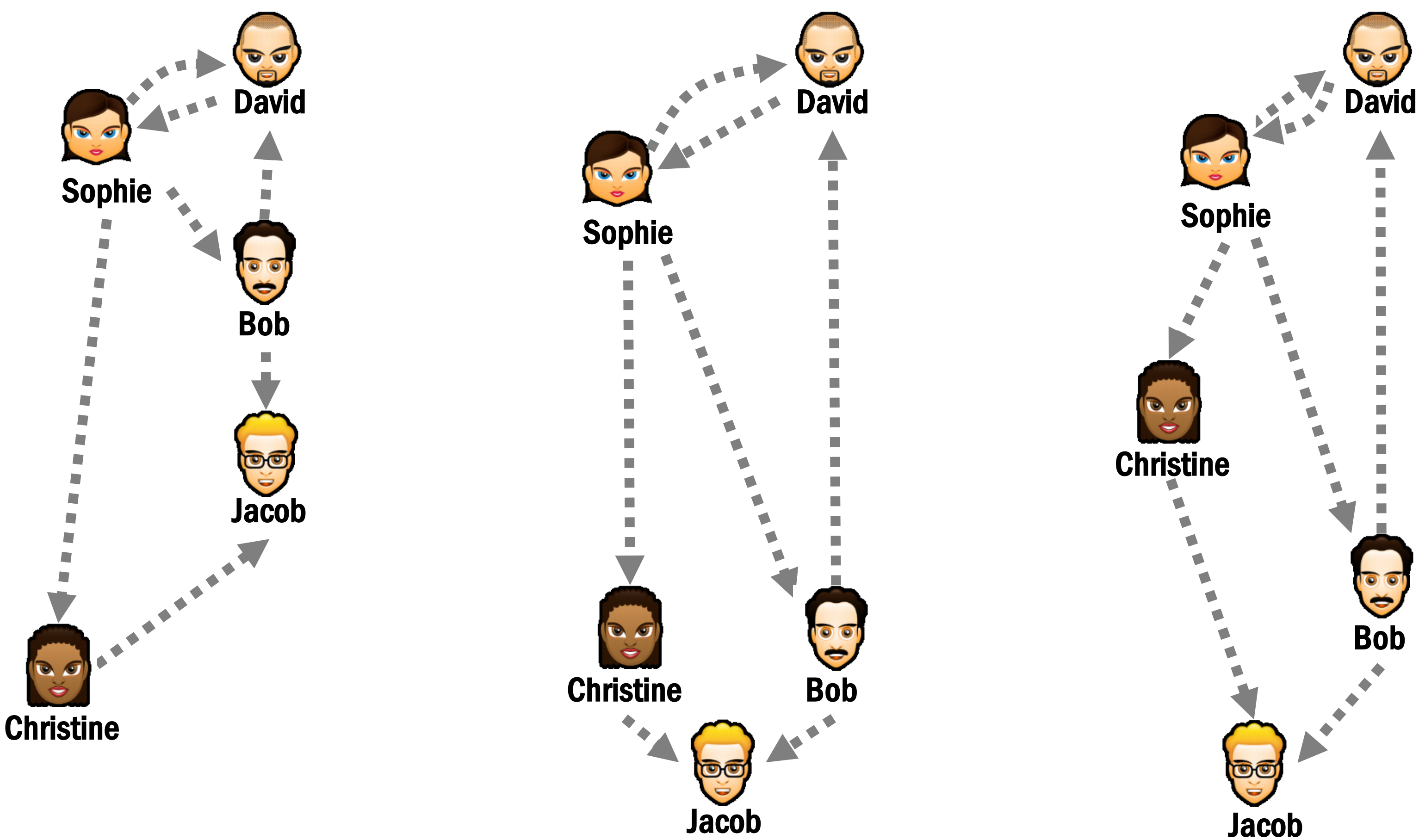
$$\sigma(B,t) \approx \frac{4+2+2}{3} = 2.67$$

Find most influential user

$$\max_{s \in V} \sigma(s,t)$$

$$O\big(p\,|V|\,(|V|+|E|)\big)$$

Each graph

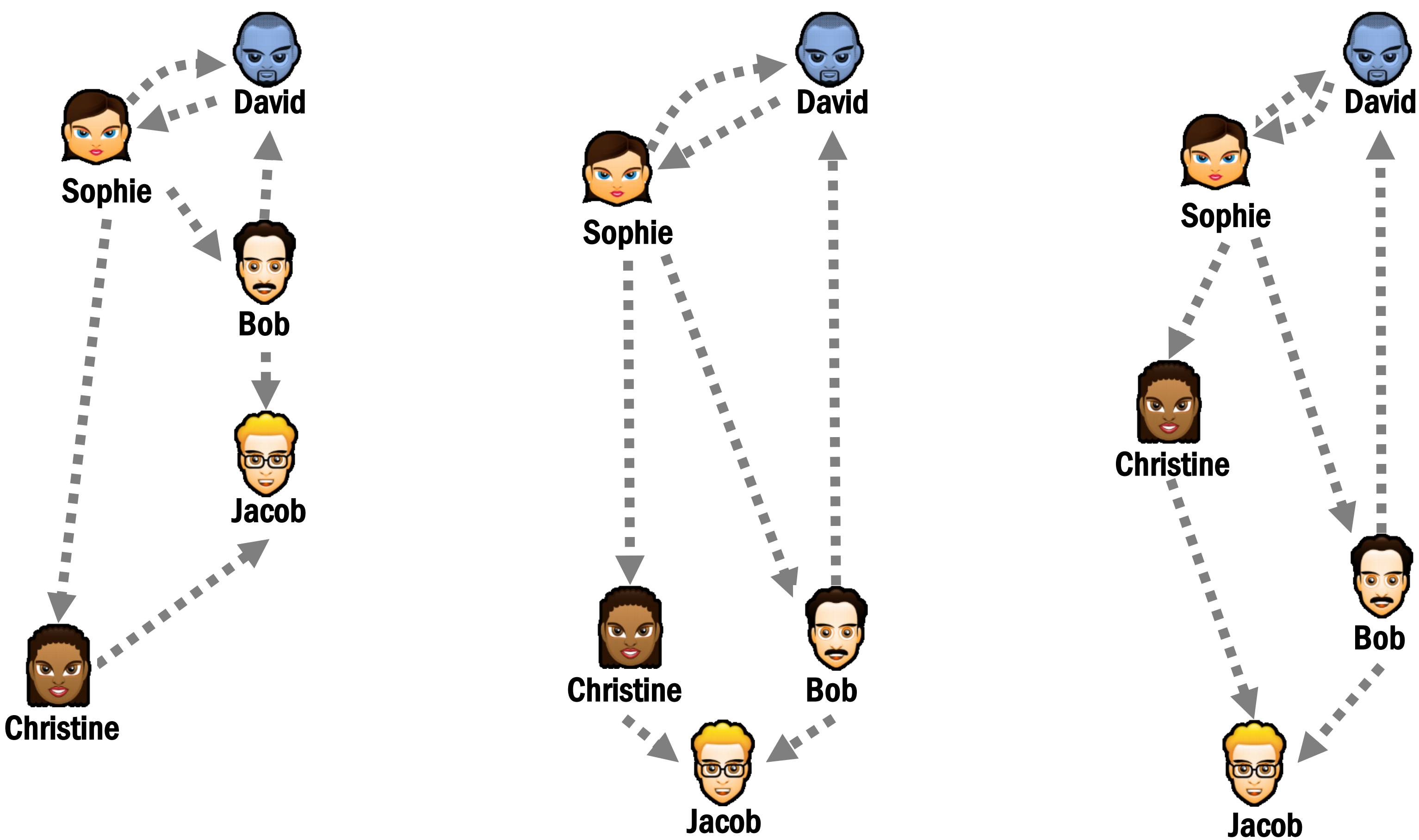


Find most influential user

$$\max_{s \in V} \sigma(s, t)$$

$$O\big(p\,|V|\,(|V| + |E|)\big)$$

Each graph   Each node



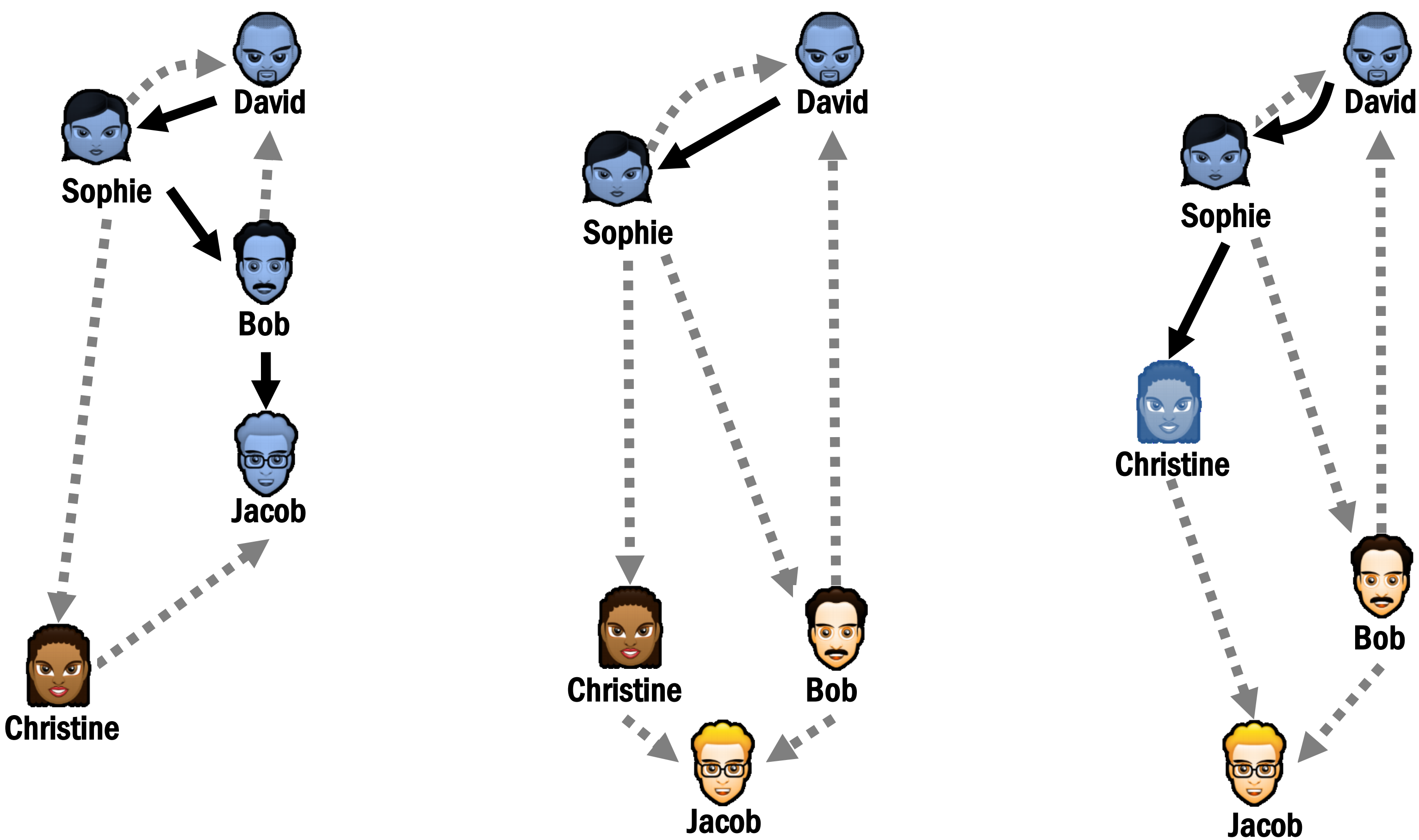
Find most influential user

$$\max_{s \in V} \sigma(s, t)$$

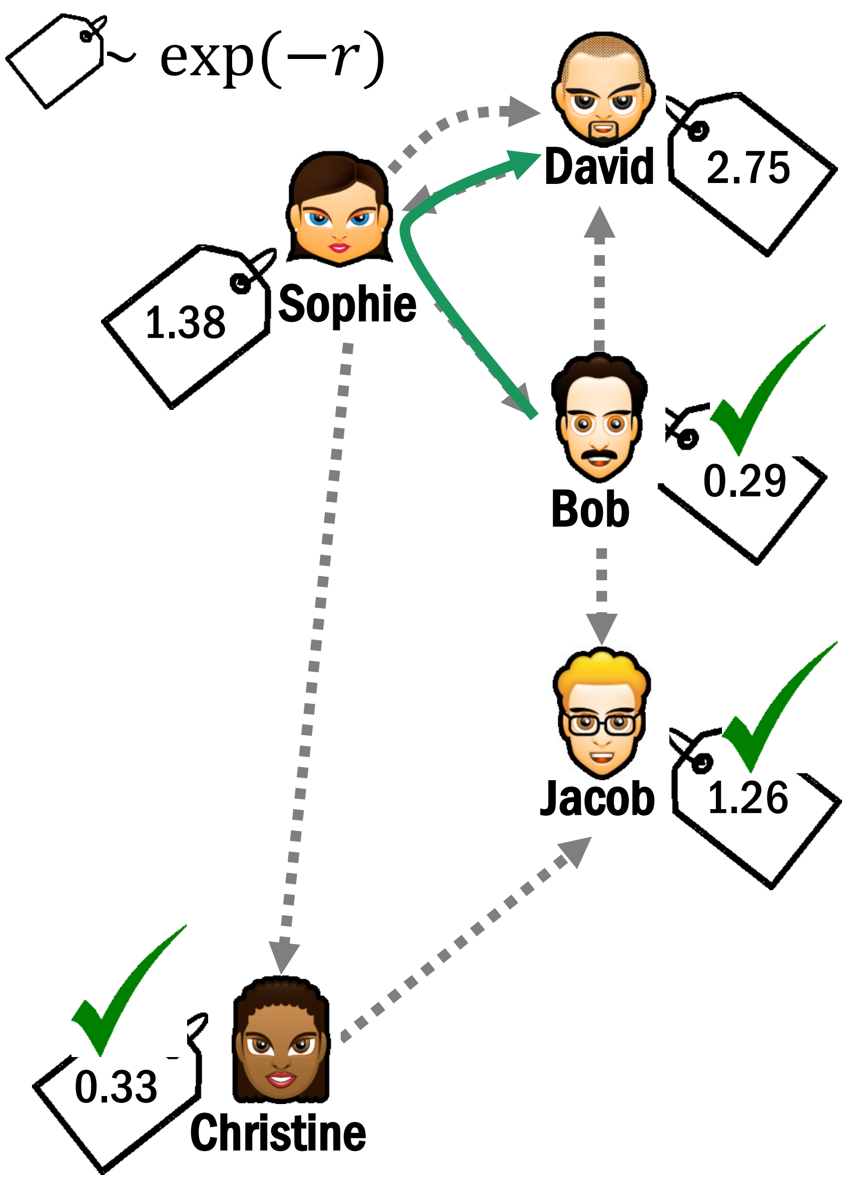
$$O\big(p\,|V|\,\underbrace{(|V| + |E|)}\big)$$

Each graph   Each node   Single source shortest path

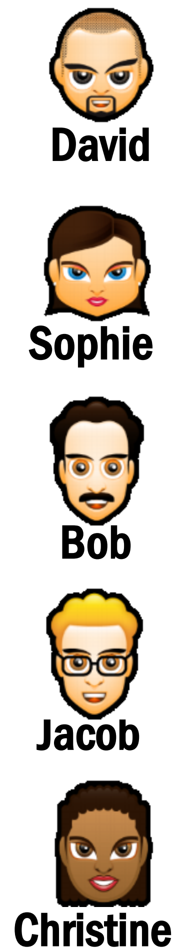
Quadratic in $|V|$ not scalable!



Randomized neighborhood estimation



Linear in # of nodes and edges



$R^D = \{ \quad \}$

$R^S = \{ \quad \}$

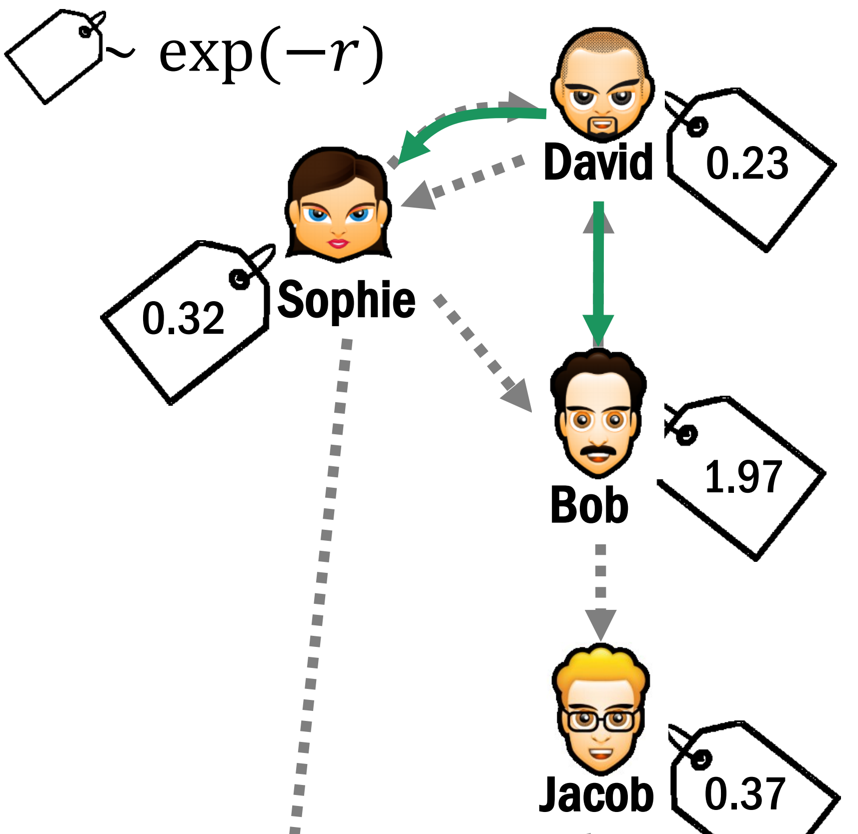
$R^B = \{ \quad \}$

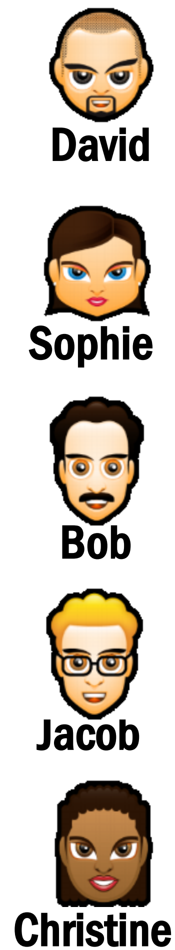
$R^J = \{ \quad \}$

$R^C = \{ \quad \}$

Randomized neighborhood estimation

$$\sigma(s,t) \approx \frac{m-1}{\sum_{i=1}^{m} R^s(i)}$$





$R^D = \{\ 0.29,\quad \}$

$R^S = \{\ 0.29,\quad \}$

$R^B = \{\ 0.29,\quad \}$

$R^J = \{\ 1.26,\quad \}$

$R^C = \{\ 0.33,\quad \}$

Given m iid samples, $r \sim e^{-r}$, their minimum r_* is distributed as

$$r_* \sim me^{-mr}$$

Computational complexity

$$\sigma(s,t) \approx \frac{1}{p}\sum_{j=1}^{p} \frac{m-1}{\sum_{i=1}^{m} R_j^s(i)}$$

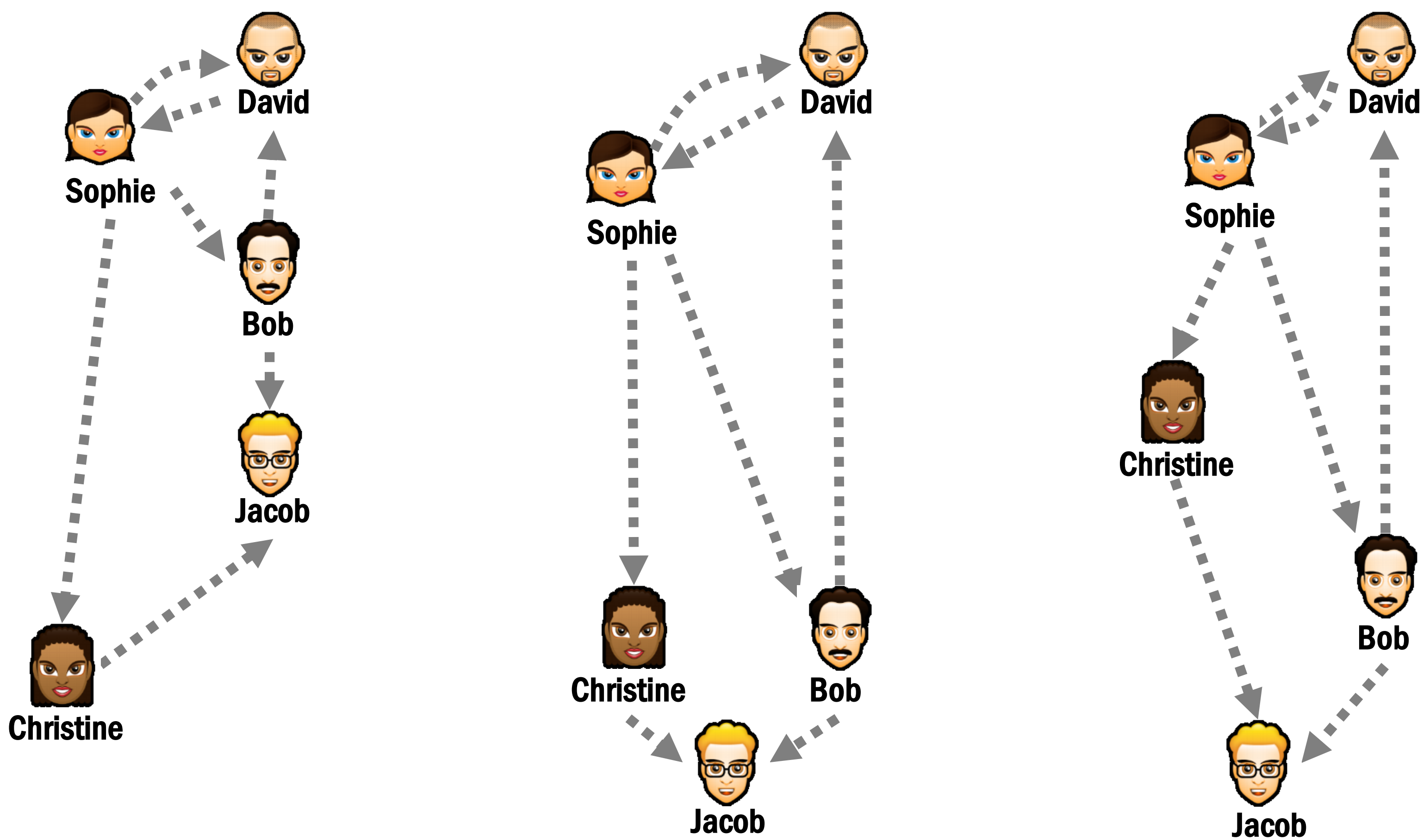
$$O\left(p\ m\left(|V| + \underbrace{(|V| + |E|)}\right)\right)$$

Each graph

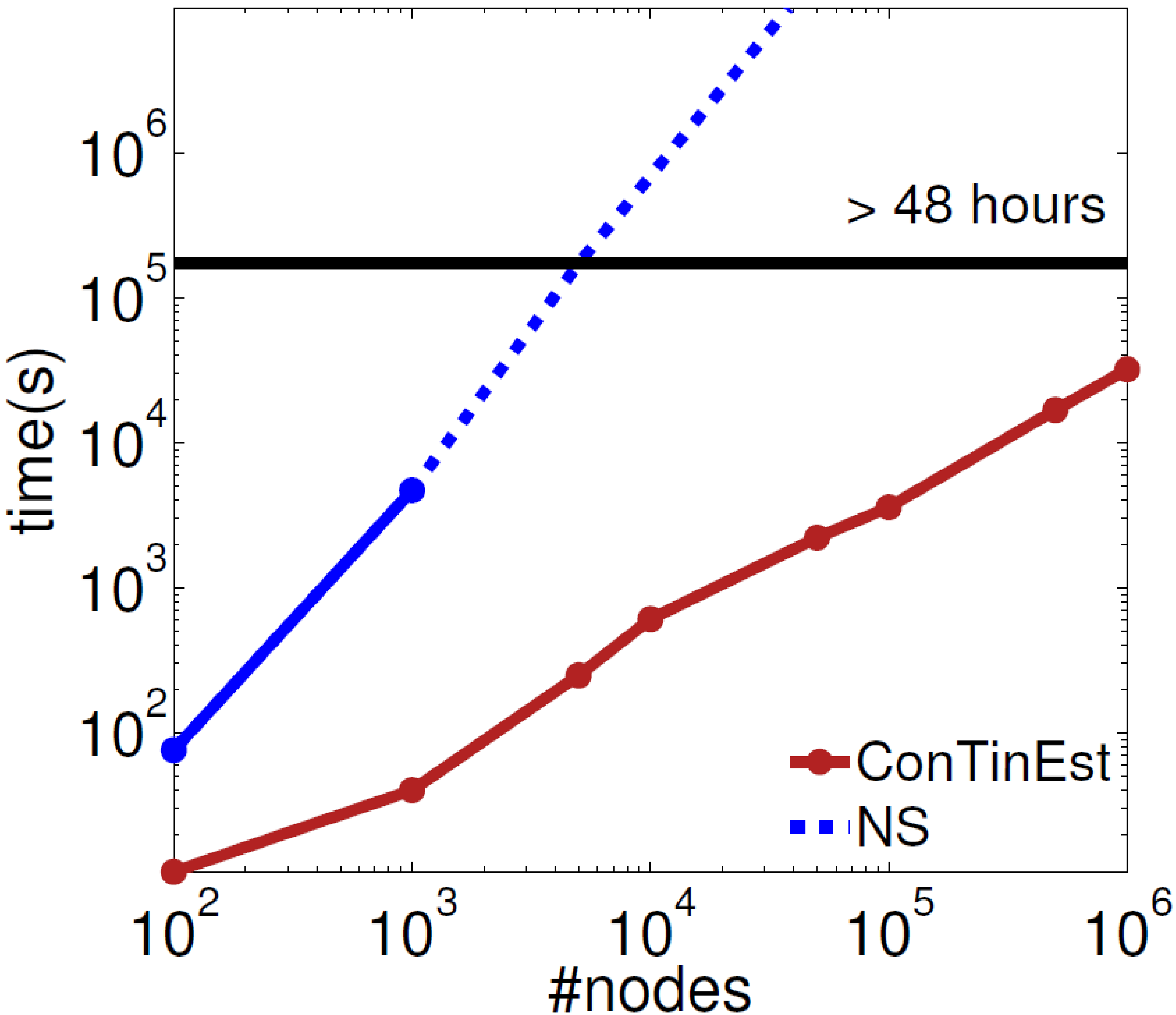
Each random label set

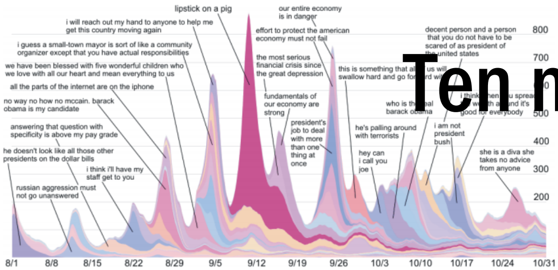
Each node

Breadth first search



Scalability





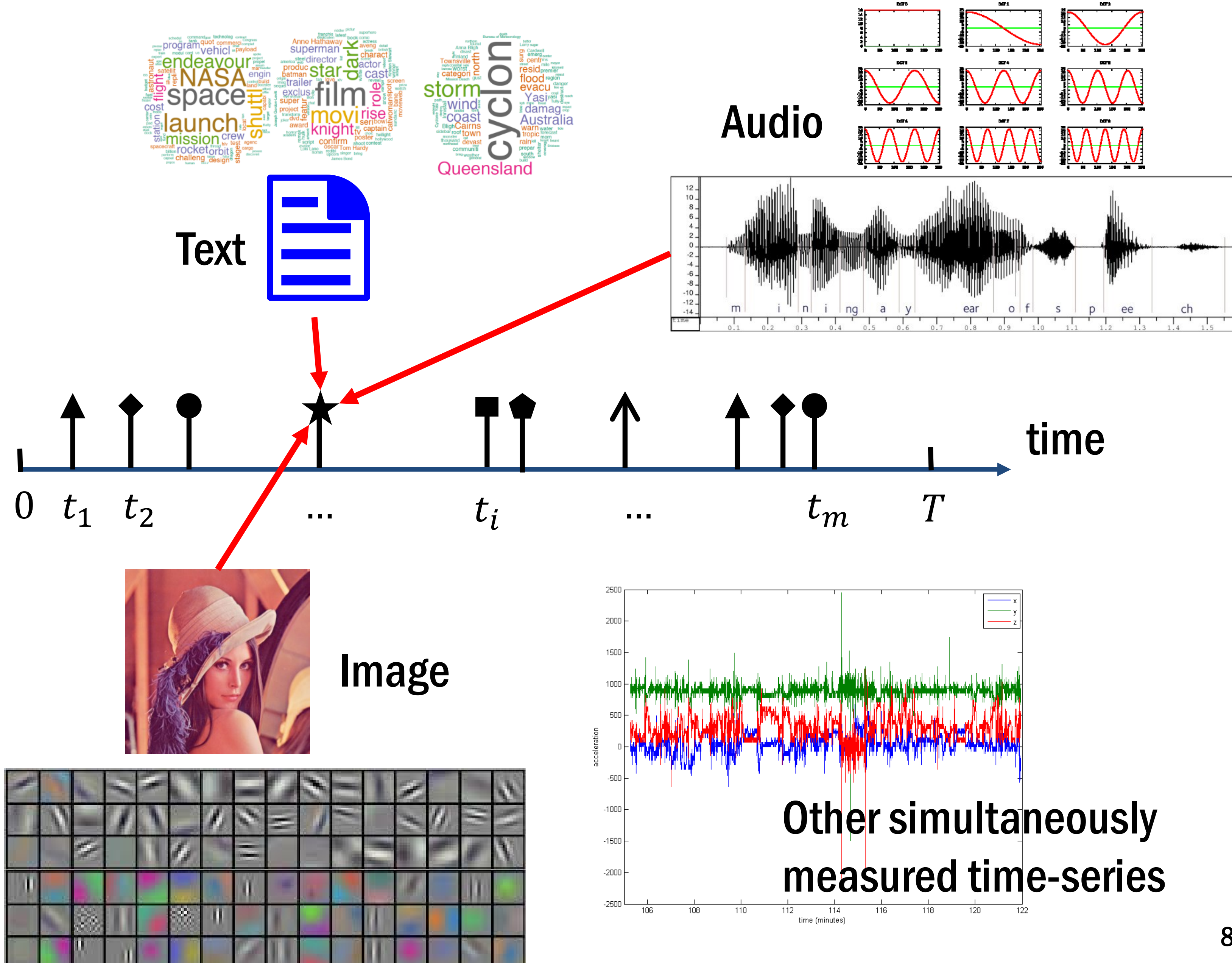
Ten most influential sites in a month

Site	Type of site
digg.com	popular news site
lxer.com	linux and open source news
exopolitics.blogs.com	political blog
mac.softpedia.com	mac news and rumors
gettheflick.blogspot.com	pictures blog
urbanplanet.org	urban enthusiasts
givemeaning.blogspot.com	political blog
talkgreen.ca	environmental protection blog
curriki.org	educational site
pcworld.com	technology news

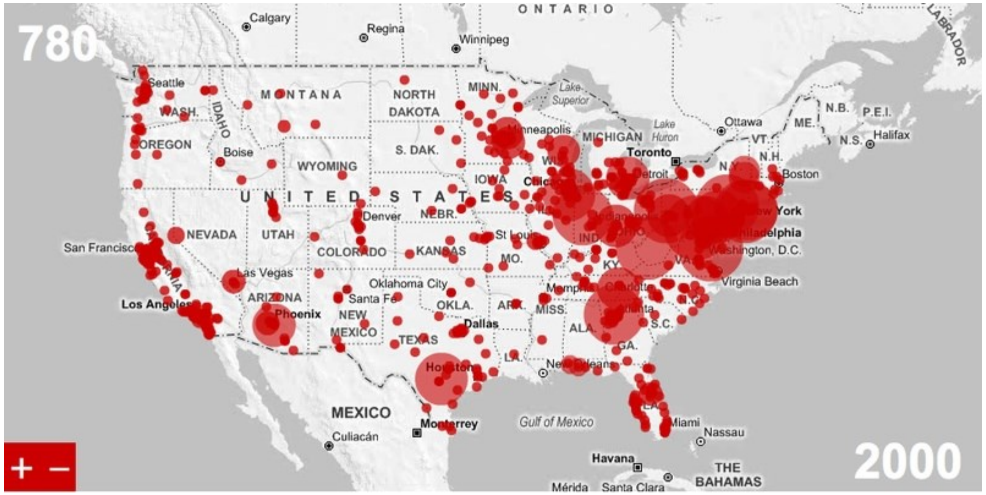
More Advanced Models

Nan et al. AISTATS 203
Nan et al. KDD 2015

Joint models with rich context

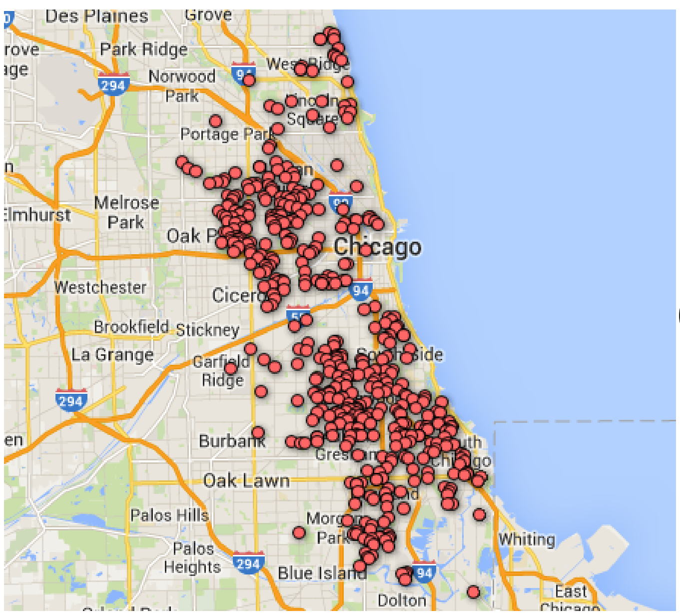


Spatial temporal processes



influenza spread

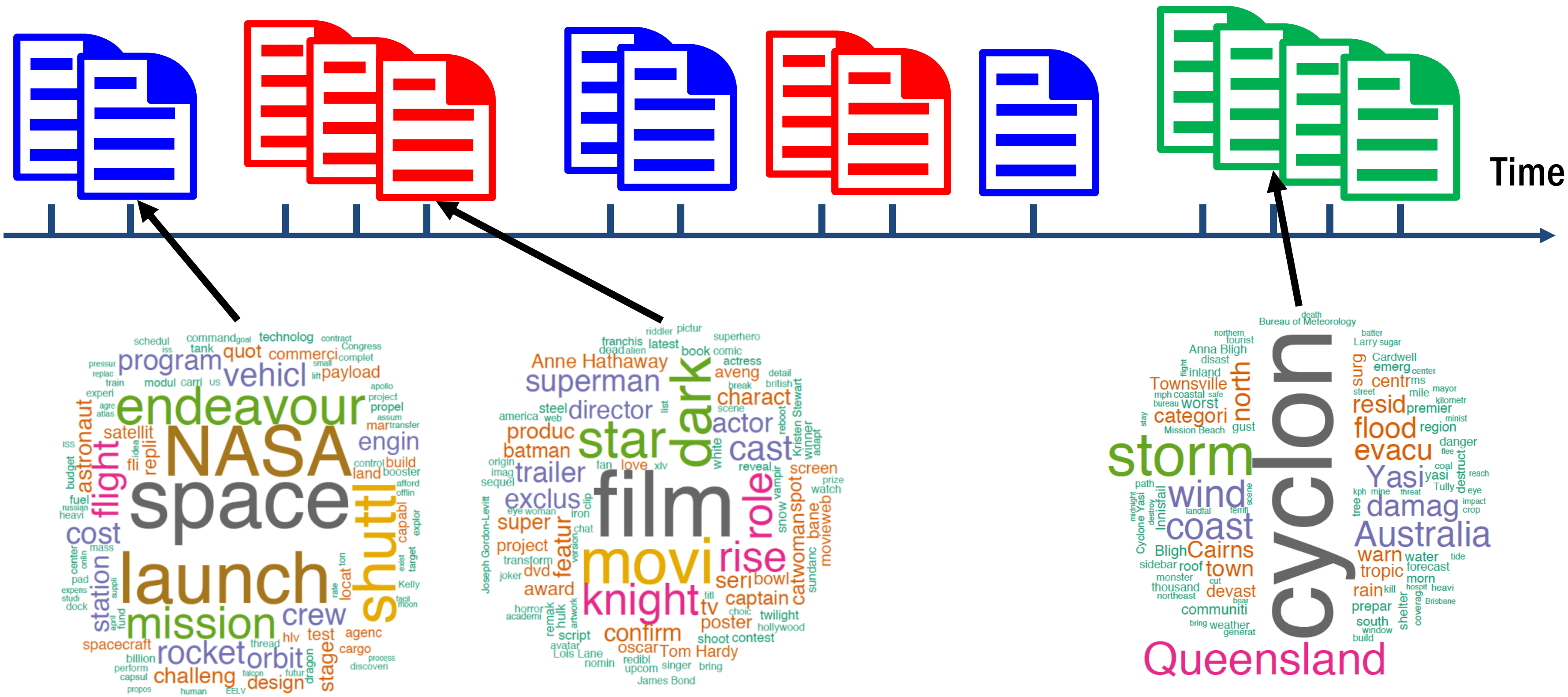
bird migration



Crime

Smart city

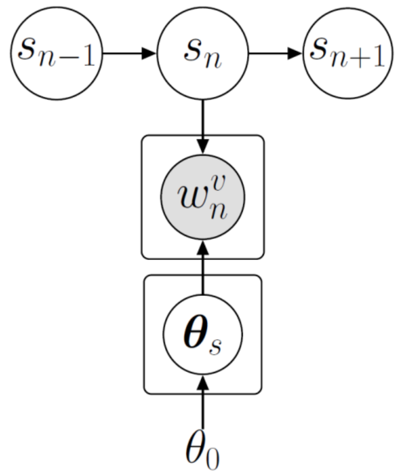
Continuous-time document streams



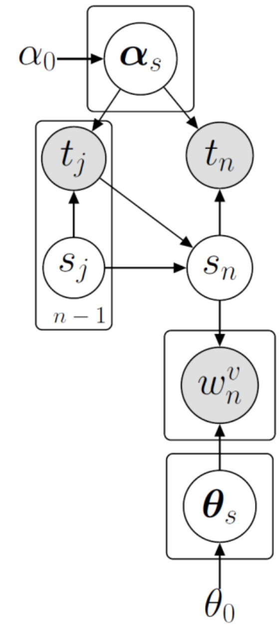
Nan et al. KDD 2015

Dirichlet-Hawkes processes

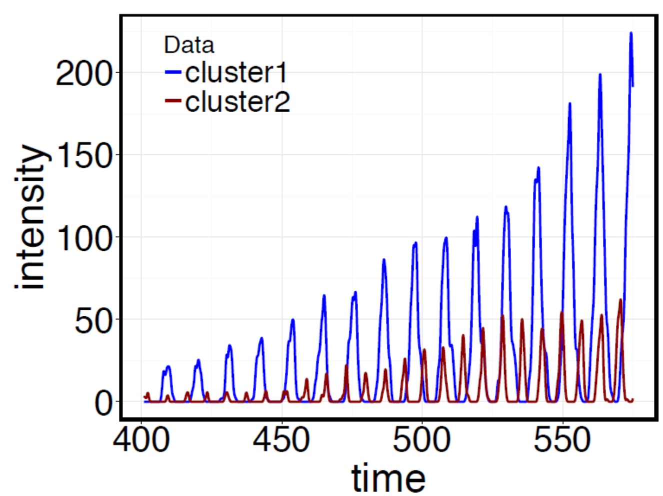
Recurrent Chinese Restaurant Process

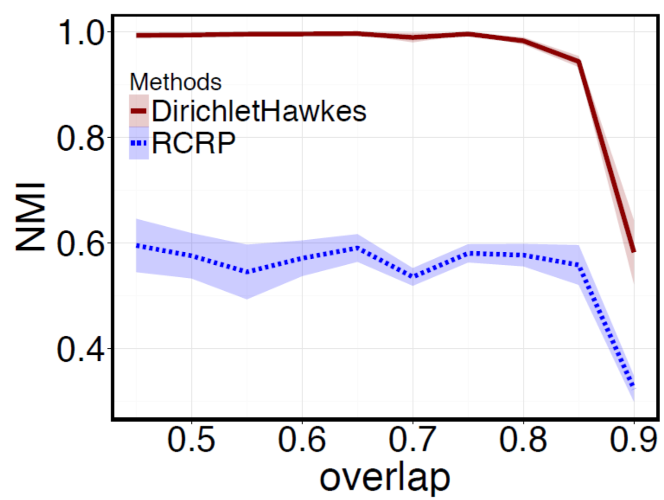


Dirichlet Hawkes Process



$$\theta_n | \theta_{1:n-1} \sim \sum_k \frac{h_k(t_n)}{\sum_{k'} h_{k'}(t_n) + \alpha} \delta(\theta_k) + \frac{\alpha}{\sum_{k'} h_{k'}(t_n) + \alpha} G_0(\theta)$$

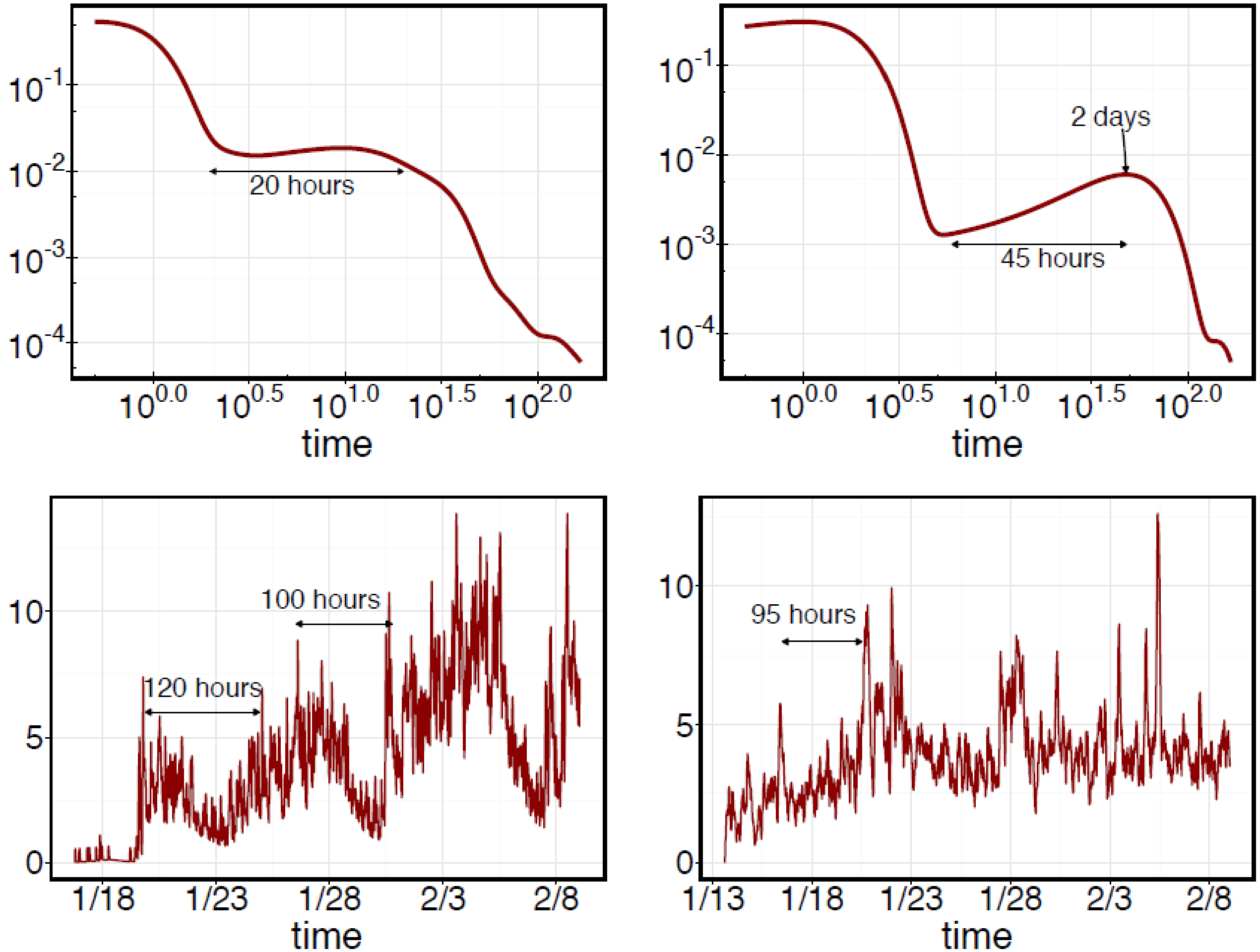




Dark Knight vs. Endeavour

Triggering Kernel

Temporal Dynamics



(b) Dark Knight Rise

(c) Endeavour

Previous models are parametric

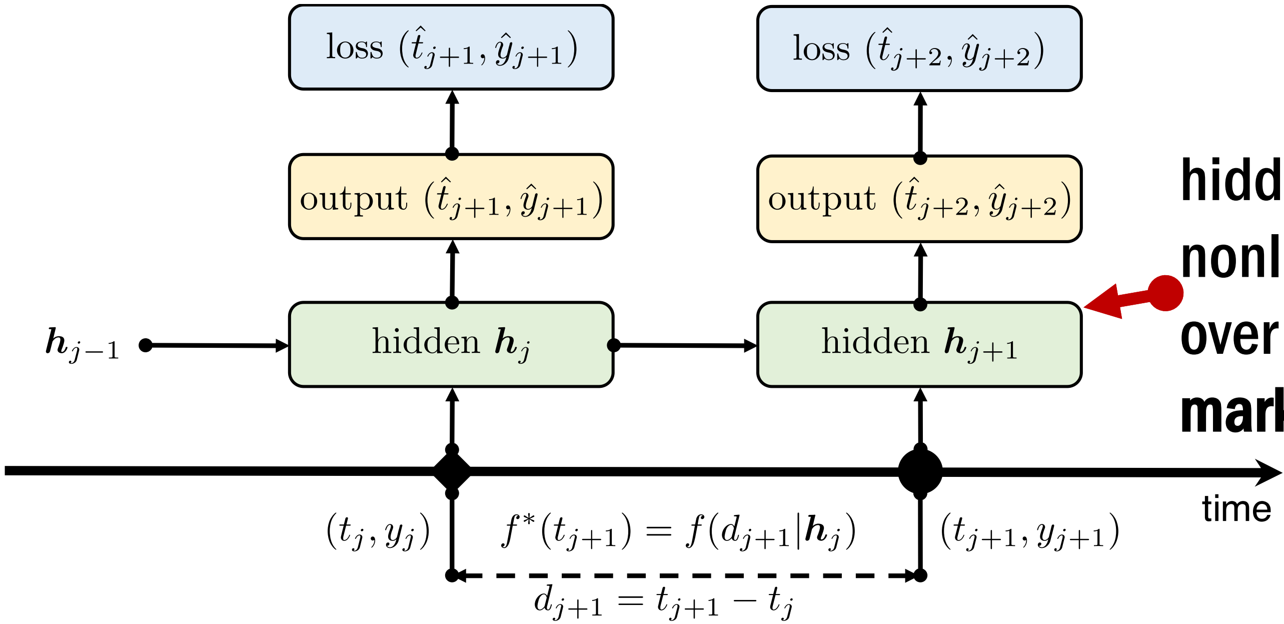
- Each parametric form encodes our prior knowledge
  - Poisson Process
  - Hawkes Process
  - Self-Correcting Process
  - Autoregressive Conditional Duration Process

- Limitations
  - Model may be misspecified
  - Hard to encode complex features or markers
  - Hard to encode dependence structure

Can we learn a more expressive model of marked temporal point processes ?

Recurrent Marked Temporal Point Processes

Recurrent neural network + Marked temporal point processes



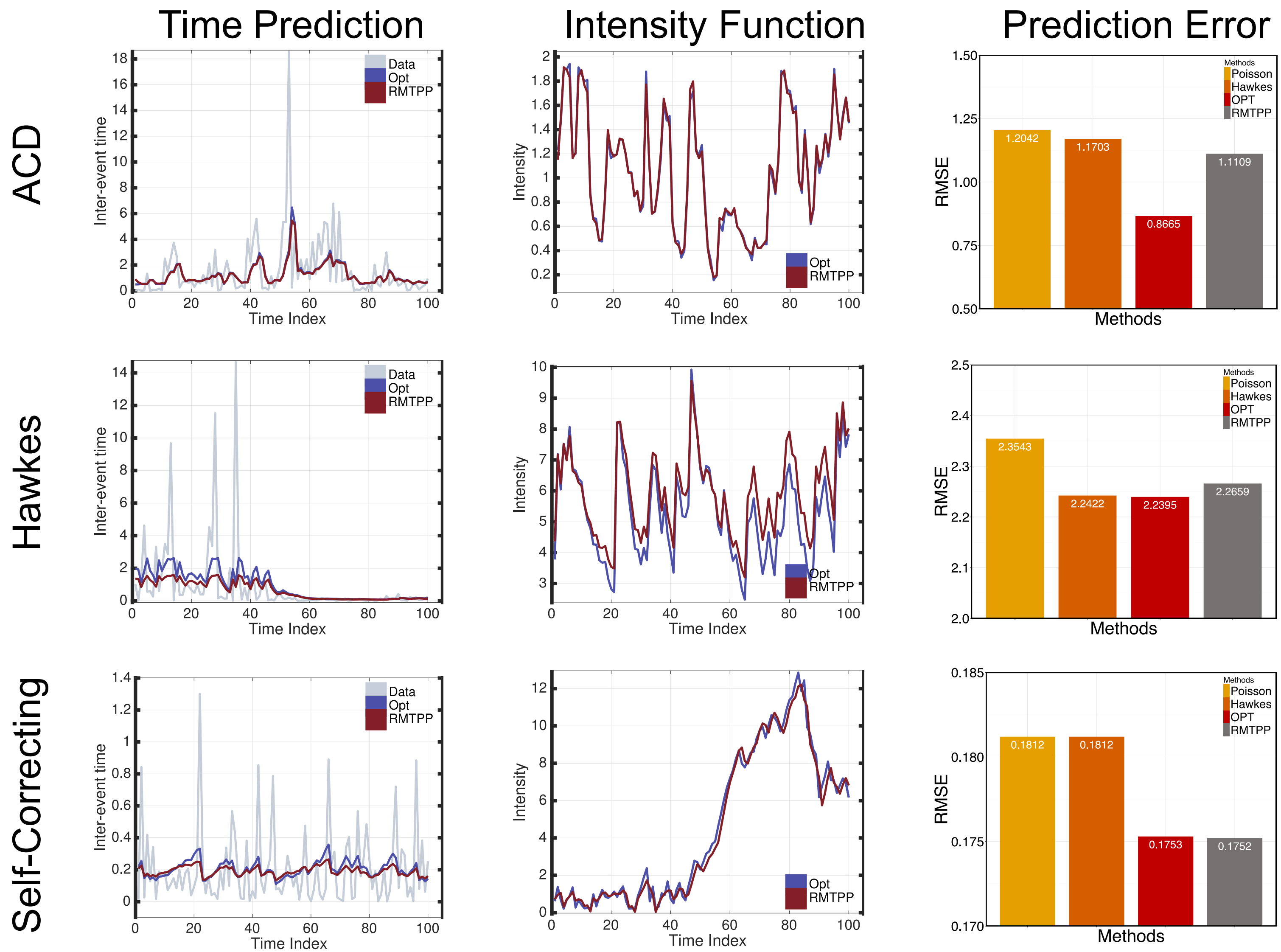
hidden vector of RNN learns a nonlinear dependency over both past **time** and **markers**

$$f^*(t_{j+1}, y_{j+1}) = f(t_{j+1}|\boldsymbol{h}_j) f(y_{j+1}|\boldsymbol{h}_j)$$

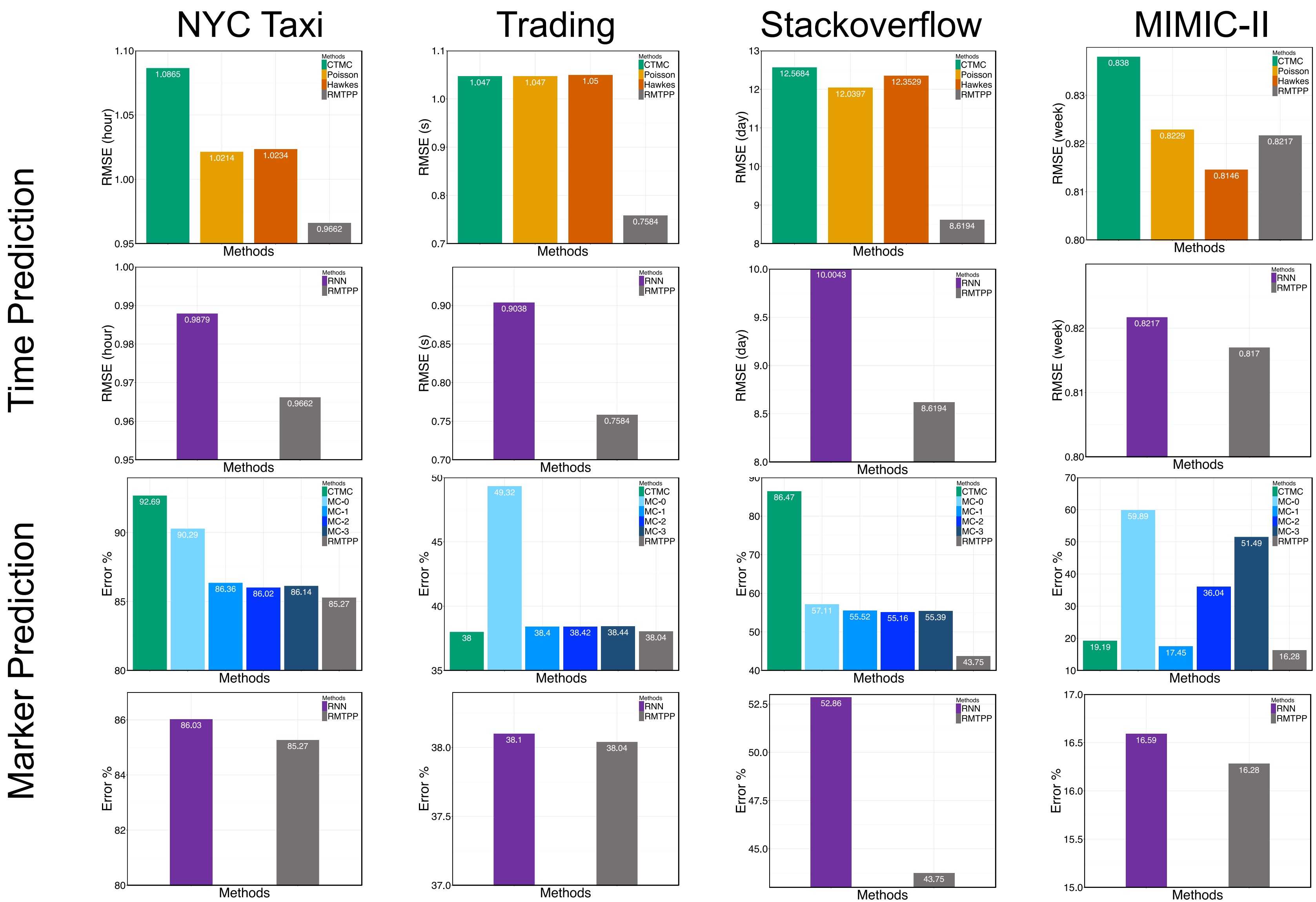
general conditional density of the next timing

multinomial distribution for the markers

Experiments: synthetic



Experiments: real world data



A unified framework

Representation

1. Intensity function
2. Basic building blocks
3. Superposition

Modeling

1. Idea adoption
2. Network coevolution
3. Collaborative dynamics

Learning

1. Sparse hidden diffusion networks
2. Low rank collaborative dynamics
3. Generic algorithm

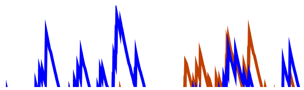
Inference

1. Time-sensitive recommendation
2. Scalable Influence estimation

PROBABILISTIC MODELS
and
LEARNING METHODS
to

$$\left\{ \begin{array}{c} \textbf{understand} \\ \textbf{predict} \\ \textbf{control} \end{array} \right\}$$

PROCESSES & ACTIVITY
over
SOCIAL & INFORMATION NETWORKS



Introduction to PtPack

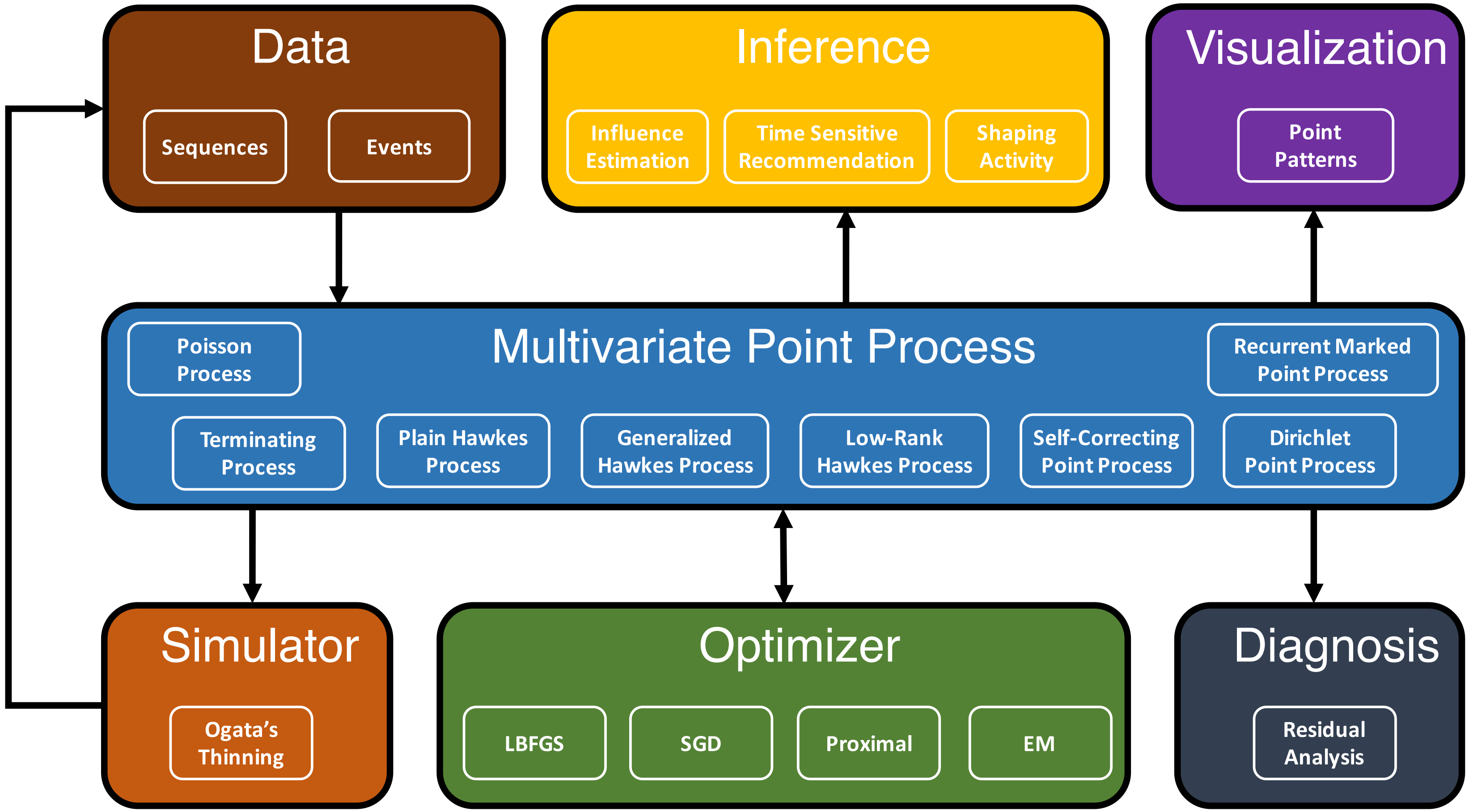
A C++ Multivariate T

Point Process Package

Features

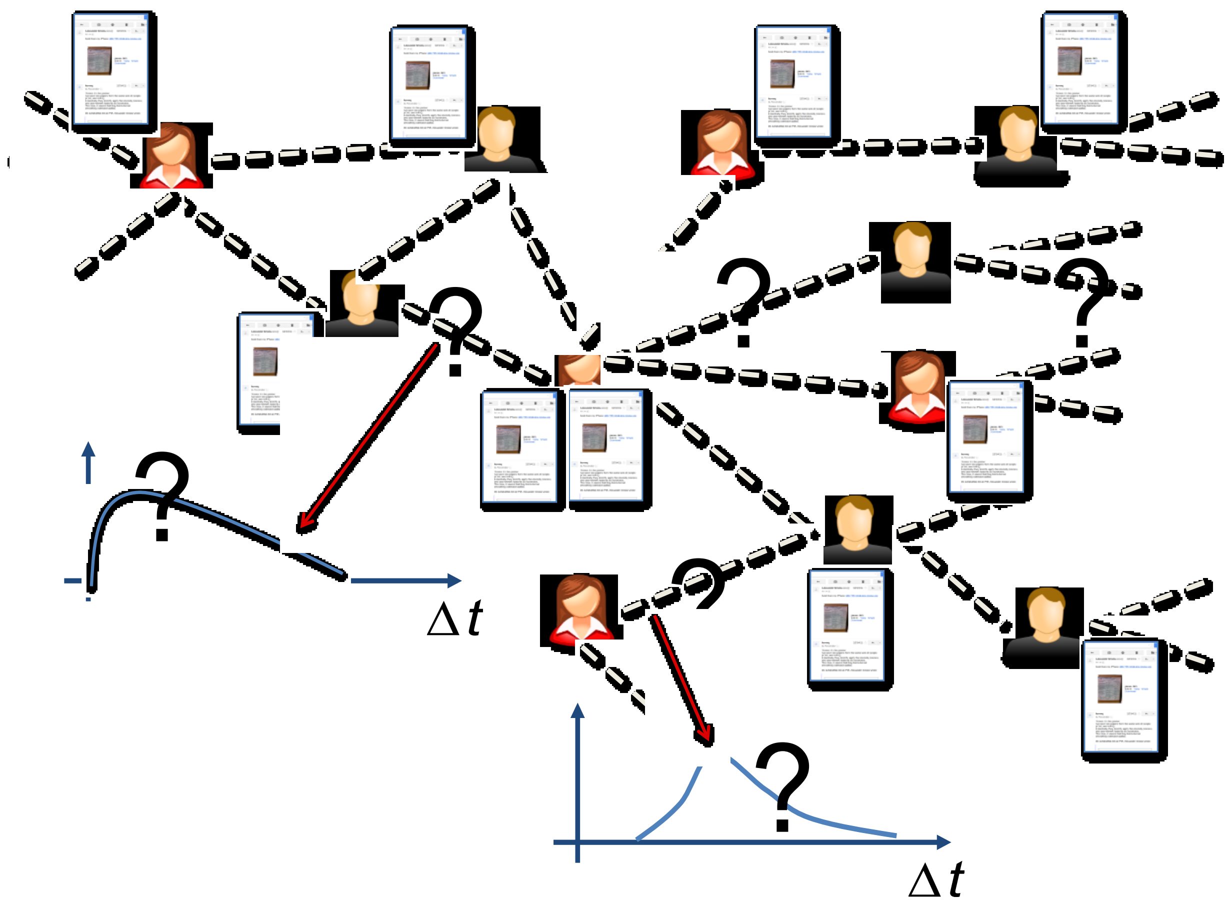
- Learning sparse interdependency structure of continuous-time information diffusions
- Scalable continuous-time influence estimation and maximization
- Learning multivariate Hawkes processes with different structural constraints, like: sparse, low-rank, customized triggering kernels
- Learning low-rank Hawkes processes for time-sensitive recommendations
- Efficient simulation of standard multivariate Hawkes processes
- Learning multivariate self-correcting processes
- Simulation of customized general temporal point processes
- Basic residual analysis and model checking of customized temporal point processes
- Visualization of triggering kernels, intensity functions, and simulated events

Overview

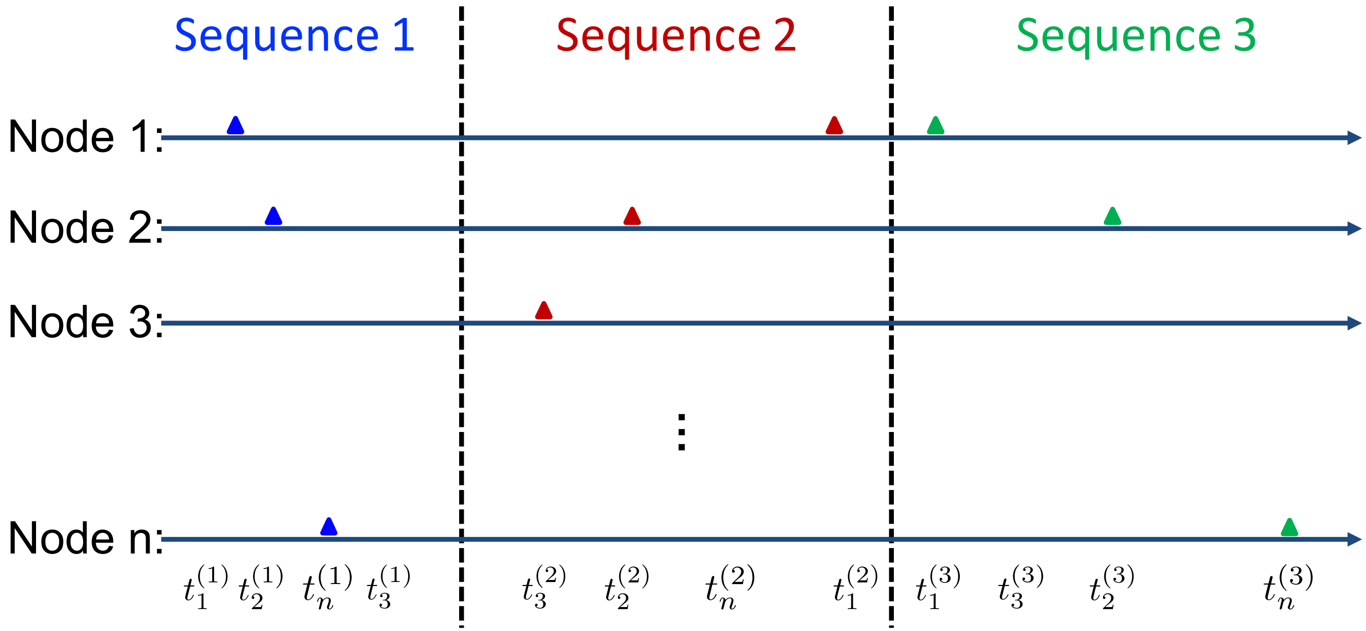


https://github.com/dunan/MultiVariatePointProcess

Demo: learning network structure



Input: sequences of infection times



node1, time1, node2, time2, node3, time3, ...

Sequence 1 **1,0, 3,0.280236, 2,2.02846, 5,2.80793, ...**

Sequence 2 **0,0, 2,0.386698, 1,0.387333, 5,0.454235, ...**

Sequence 3 **4,0, 5,2.70542**

Load sequences

```
// data stores all input sequences
std::vector<Sequence> data;

// suppose we have six nodes without knowing their interdependency
   structure
unsigned N = 6;

// initialize the observation window
double T = 0;

// load sequences from file
ImportFromExistingCascades("data/example_cascade_exp_1000", N, T, data);
std::cout << "1. Loaded " << data.size() << " sequences" << std::endl;
```

Setting options

```
unsigned dim = N, num_params = dim * dim;
// initialize the standard multivariate terminating process
PlainTerminating terminating(num_params, dim);

// set different fitting options
PlainTerminating::OPTION options;

// use projected-LBFGS optimization algorithm
options.method = PlainTerminating::PLBFGS;

// use L1-norm for sparsity
options.excitation_regularizer = PlainTerminating::L1;

// set the regularization coefficient
options.coefficients[PlainTerminating::LAMBDA] = 1e-3;

// fit the parameters
std::cout << "2. Fitting parameters" << std::endl << std::endl;
terminating.fit(data, options);
```

Retrieving results

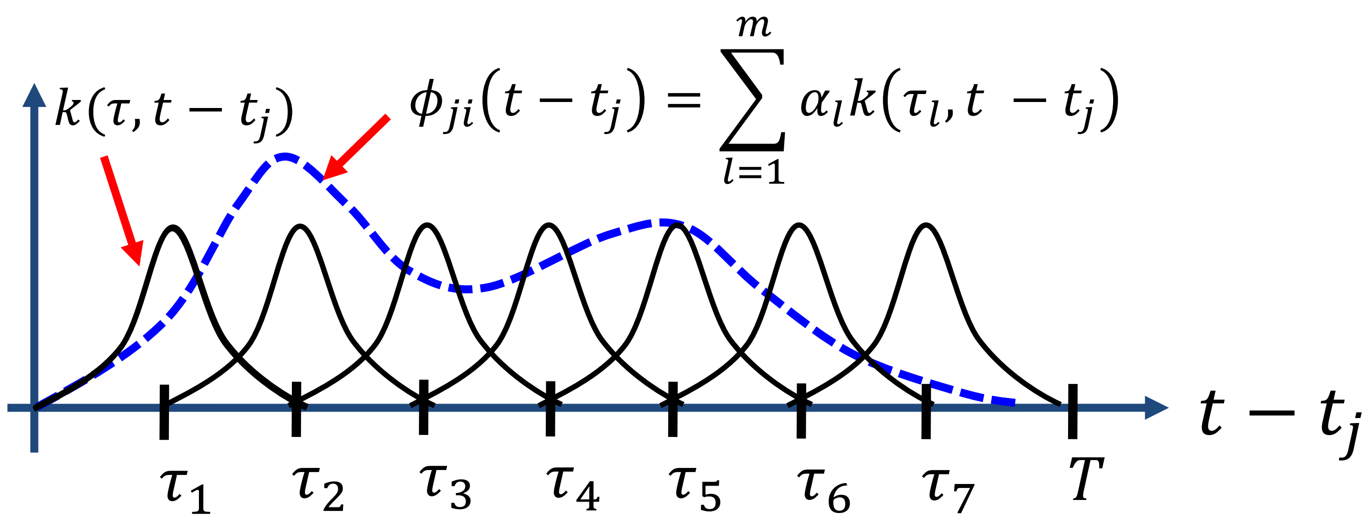
```
// Get fitted parameters
Eigen::VectorXd result = terminating.GetParameters();

// 6-by-6 interdependency matrix
Eigen::Map<Eigen::MatrixXd> alpha_matrix = Eigen::Map<Eigen::MatrixXd>(
    result.data(), dim, dim);

std::cout << std::endl << "Estimated Parameters : " << std::endl <<
    alpha_matrix << std::endl;
```

https://github.com/dunan/MultiVariatePointProcess/blob/master/example/learning_network_structure_exp_kernel.cc

Running

```
lawn-143-215-206-69:example nandu$ build/learning_network_structure_exp_kernel
1. Loaded 1000 sequences
2. Fitting parameters

 Iteration        FunEvals      Step Length     Function Val      Opt Cond
         1               2         0.013487          31.8054        68.2172
         2               3                1          2.67816        1.40804
         3               4                1          2.62518        1.38426
         4               6              0.1           2.4393       0.990146
         5               8              0.1          2.31951       0.654871
         6              10              0.1          2.26407       0.592488
         7              11                1          2.23483       0.376114
         8              14             0.01          2.22692       0.340944
         9              15                1          2.19597       0.112433
        10              16                1          2.19311      0.0596917
        11              17                1          2.19158      0.0349702
        12              18                1          2.19154      0.0275834
        13              19                1          2.19132       0.027687
        14              21              0.1          2.19129      0.0244095
        15              22                1          2.19122      0.0123253
        16              23                1          2.19119     0.00924419
        17              24                1          2.19116     0.00868183
        18              25                1          2.19086      0.0239286
        19              26                1          2.19049      0.0348729
        20              27                1          2.19004      0.0622901
        21              28                1          2.18917      0.0227247
        22              29                1          2.18904      0.0181591
        23              30                1          2.18868      0.0334986
        24              31                1          2.18825     0.00359063
        25              32                1          2.18825    0.000825038
        26              33                1          2.18825    9.32674e-05
        27              34                1          2.18825    3.64155e-05
        28              35                1          2.18825    3.26591e-06
Directional Derivative below optTol


Estimated Parameters :
       0 0.985479  1.02109        0        0        0
       0        0 0.958809 0.924004        0        0
       0        0        0        0        0  1.02045
       0        0        0        0        0        0
       0        0        0        0        0   1.0601
       0        0        0        0        0        0
```

Learn general infection risks




```
// Use 100 RBF basis functions
unsigned dim = N, num_basis = 100, num_params = num_basis * dim * dim;

// Set the grid point from 0 to the observation window T
Eigen::VectorXd tau = Eigen::VectorXd::LinSpaced(num_basis, 0, T);

// Set the bandwidth of the RBF basis function
Eigen::VectorXd sigma = Eigen::VectorXd::Constant(tau.size(), 1.0);
```

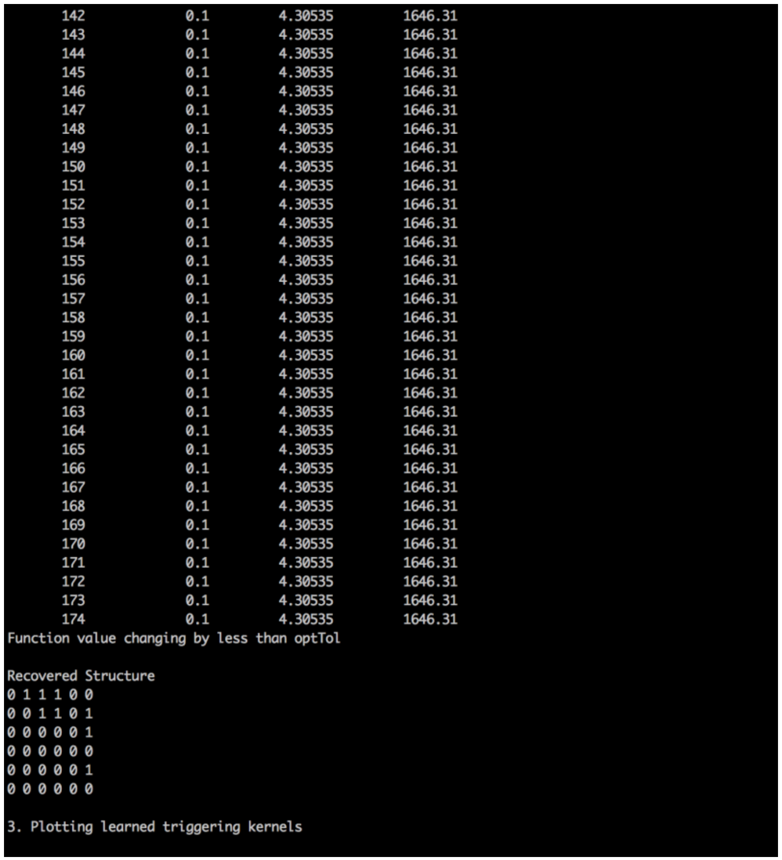
Setting options

```
// initialize the multivariate terminating process with general triggering
    kernels
TerminatingProcessLearningTriggeringKernel terminating(num_params, dim,
   tau, sigma);

TerminatingProcessLearningTriggeringKernel::OPTION options;
// use group-lasso type of regularization
options.excitation_regularizer =
   TerminatingProcessLearningTriggeringKernel::GROUP;

// set the regularization coefficient
options.coefficients[TerminatingProcessLearningTriggeringKernel::LAMBDA] =
    1;

std::cout << "2. Fitting parameters" << std::endl << std::endl;
terminating.fit(data, options);
```

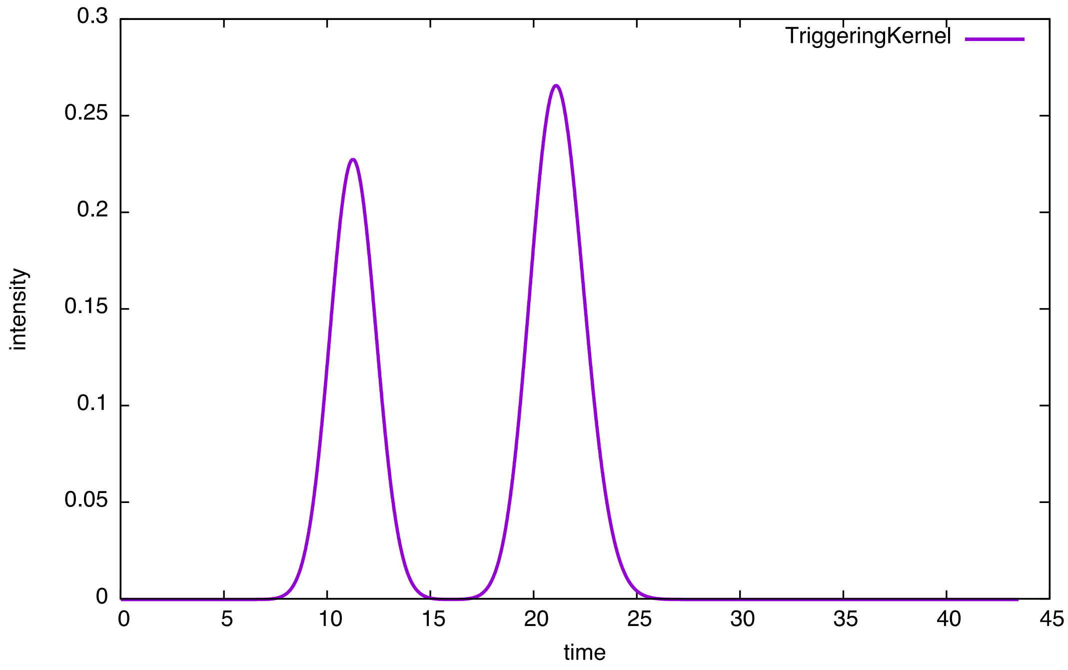
https://github.com/dunan/MultiVariatePointProcess/blob/master/example/learning_network_structure_general_kernel.cc

Plotting the learned functions

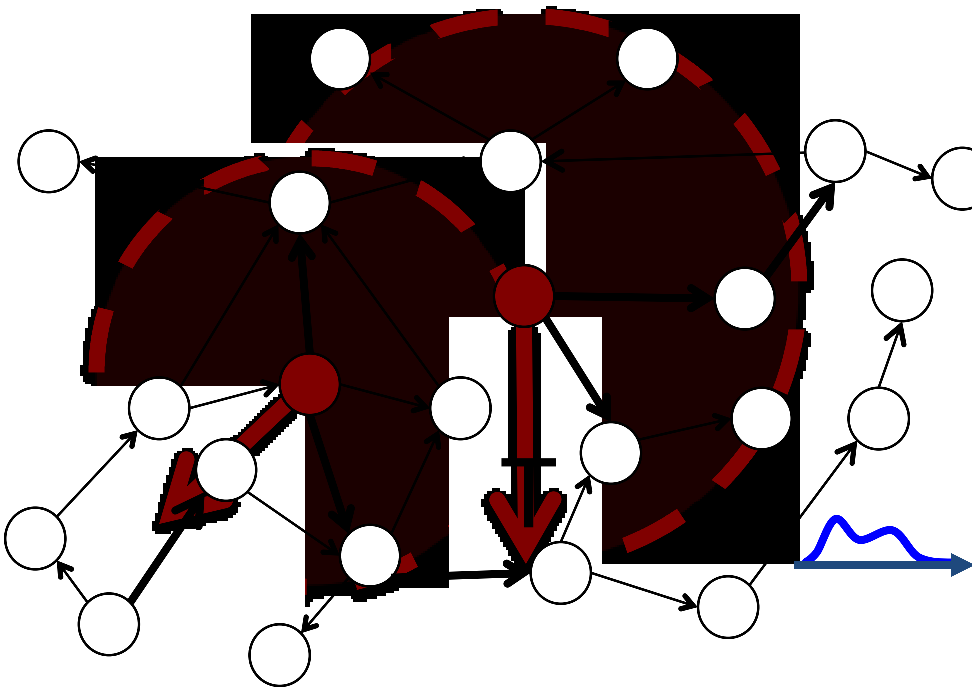
```cpp
std::cout << "3. Plotting learned triggering kernels" << std::endl << std
    ::endl;

// plot the infection risk function between node 0 and 1
terminating.PlotTriggeringKernel(0, 1, T, 0.01);
```



```
142    0.1    4.30535    1646.31
143    0.1    4.30535    1646.31
144    0.1    4.30535    1646.31
145    0.1    4.30535    1646.31
146    0.1    4.30535    1646.31
147    0.1    4.30535    1646.31
148    0.1    4.30535    1646.31
149    0.1    4.30535    1646.31
150    0.1    4.30535    1646.31
151    0.1    4.30535    1646.31
152    0.1    4.30535    1646.31
153    0.1    4.30535    1646.31
154    0.1    4.30535    1646.31
155    0.1    4.30535    1646.31
156    0.1    4.30535    1646.31
157    0.1    4.30535    1646.31
158    0.1    4.30535    1646.31
159    0.1    4.30535    1646.31
160    0.1    4.30535    1646.31
161    0.1    4.30535    1646.31
162    0.1    4.30535    1646.31
163    0.1    4.30535    1646.31
164    0.1    4.30535    1646.31
165    0.1    4.30535    1646.31
166    0.1    4.30535    1646.31
167    0.1    4.30535    1646.31
168    0.1    4.30535    1646.31
169    0.1    4.30535    1646.31
170    0.1    4.30535    1646.31
171    0.1    4.30535    1646.31
172    0.1    4.30535    1646.31
173    0.1    4.30535    1646.31
174    0.1    4.30535    1646.31
Function value changing by less than optTol

Recovered Structure
0 1 1 1 0 0
0 0 1 1 0 1
0 0 0 0 0 1
0 0 0 0 0 0
0 0 0 0 0 1
0 0 0 0 0 0

3. Plotting learned triggering kernels
```



Demo: influence maximization



$$f_{ji}(t_i|t_j) = f_{ji}(t_i - t_j)$$

https://github.com/dunan/MultiVariatePointProcess/blob/master/example/influence_maximization.cc

Load diffusion networks

- For the demo, we assume pairwise Weibull distribution

$$f_{ji}(\Delta t; \alpha, \beta) = \frac{\beta}{\alpha}\left(\frac{\Delta t}{\alpha}\right)^{\beta-1} e^{-(\Delta t/\alpha)^{\beta}}$$

scale parameter

shape parameter

- For each edge, we have:

$$\text{node } j, \text{ node } i, \beta_{ji}, \alpha_{ji}$$

Load diffusion networks

```
// suppose 1024 nodes
unsigned N = 1024;

// load network with edge reversed
Graph G("data/std_weibull_DAG_core-1024-1-network", N, true);
// load the original network for comparing with the monte-carlo
    simulations
Graph G1("data/std_weibull_DAG_core-1024-1-network", N, false);

// use 5000 sampled networks, each of which has 5 sets of node labels
unsigned num_samples = 5000, num_labels = 5;

// initialize influence estimation
ConTinEst continest(&G, &G1, num_samples, num_labels);
```

Influence estimation

```
std::cout <<"Get all least-label lists : " << num_samples << " sets of
   transmission times; " << num_labels << " sets of random labels; ";

// calculate least-neighbor lists for each node
contin est.GetLeastElementLists();

std::cout <<"done" << std::endl << std::endl;

// use 10000 monte carlo simulations for comparison
unsigned C = 10000;

// increase observation window from 1 to 10
for(unsigned T = 1; T <= 10; ++ T)
{
  std::cout << "Estimate Influence T = " << T;
  double estimated_influence = continest.EstimateNeighborhood(sources, T);
  std::cout << " done" << std::endl << std::endl;

  std::cout << "Simulation Check T = " << T << " " << " with " << C << "
   samples" << std::endl;

  std::cout << "ConTinEst : " << estimated_influence << std::endl << "
   Simulation : " << continest.RandomSimulation(T, sources, C) << std::
   endl << std::endl;
}
```

Influence maximization

```
std::vector<double> set_T;
std::vector<unsigned> set_K;

// set observation window to 10
set_T.push_back(10);
// select at most 10 sources
set_K.push_back(10);

std::cout <<"Influence Maximization by selecting 10 nodes with T = 10 ";

// influence maximization
std::vector<std::set<unsigned> > tables = continest.Optimize(set_T, set_K)
    ;

std::cout << "done" << std::endl;
```

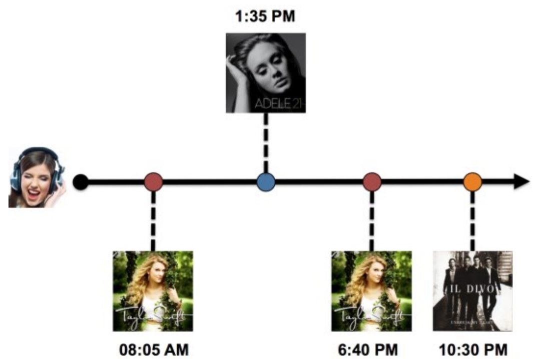
Running

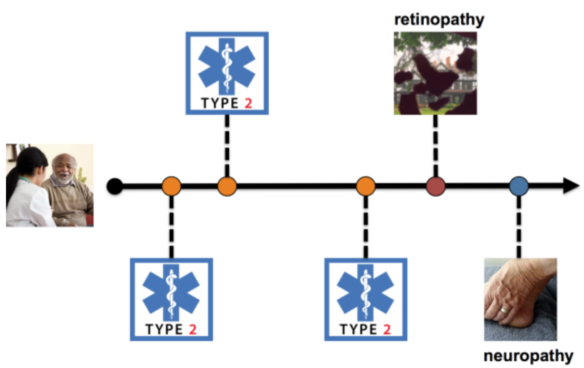
```
Get all least-label lists : 5000 sets of transmission times; 5 sets of random labels; done

node 0 has the largest out-degree 24
Estimate Influence T = 1 done

Simulation Check T = 1  with 10000 samples
ConTinEst : 7.65846
Simulation : 7.7011

Estimate Influence T = 2 done

Simulation Check T = 2  with 10000 samples
ConTinEst : 14.0149
Simulation : 13.974

Estimate Influence T = 3 done

Simulation Check T = 3  with 10000 samples
ConTinEst : 24.0831
Simulation : 24.239

Estimate Influence T = 4 done

Simulation Check T = 4  with 10000 samples
ConTinEst : 38.965
Simulation : 39.1839

Estimate Influence T = 5 done

Simulation Check T = 5  with 10000 samples
ConTinEst : 57.268
Simulation : 58.0492

Estimate Influence T = 6 done

Simulation Check T = 6  with 10000 samples
ConTinEst : 79.4595
Simulation : 79.5592

Estimate Influence T = 7 done

Simulation Check T = 7  with 10000 samples
ConTinEst : 104.509
Simulation : 104.387
```

```
Estimate Influence T = 8 done

Simulation Check T = 8  with 10000 samples
ConTinEst : 133.747
Simulation : 133.88

Estimate Influence T = 9 done

Simulation Check T = 9  with 10000 samples
ConTinEst : 163.572
Simulation : 163.398

Estimate Influence T = 10 done

Simulation Check T = 10  with 10000 samples
ConTinEst : 192.547
Simulation : 191.941

Influence Maximization by selecting 10 nodes with T = 10 done
selected sources : 0;1;2;4;10;16;17;21;524;890;
lawn-143-215-206-69:example nandu$
```

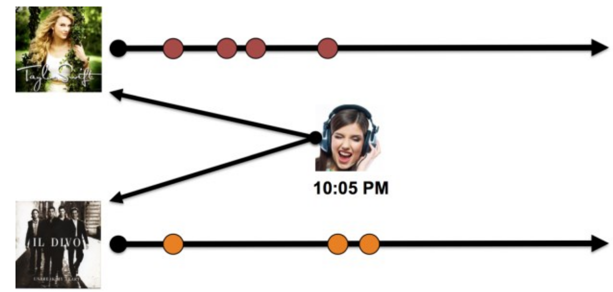
Demo: time-sensitive recommendation

From

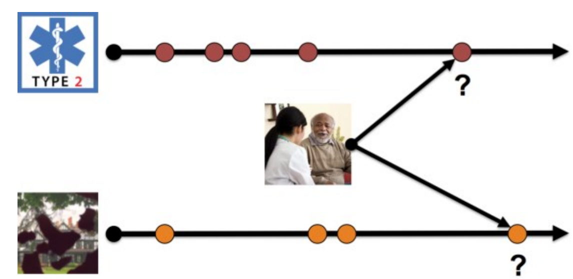




to predict



Which one to recommend at time T ?

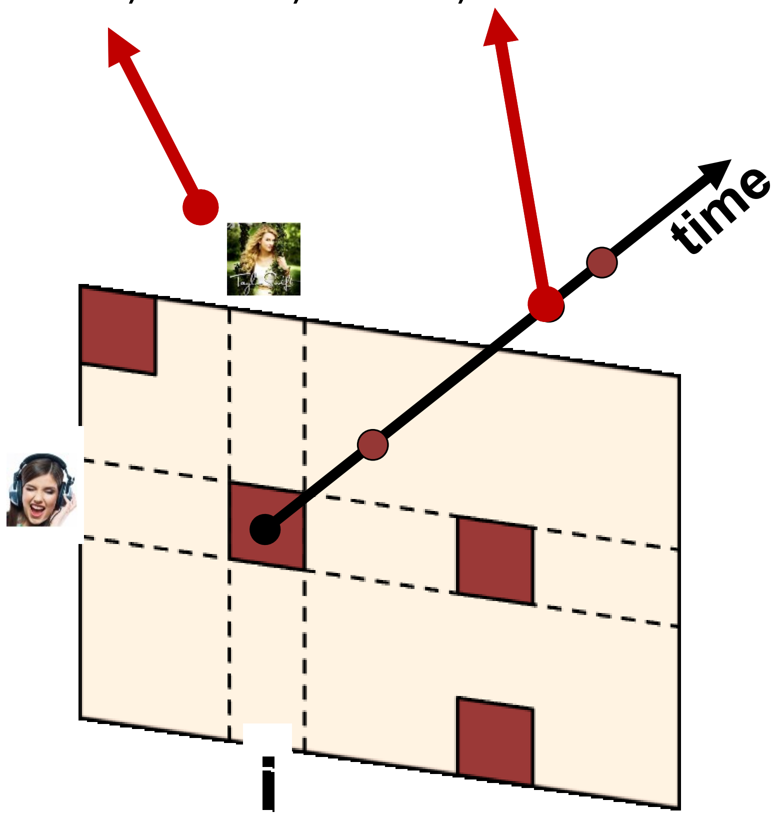


Which disease will progress at the next visit ?

https://github.com/dunan/MultiVariatePointProcess/blob/master/example/learning_lowrank_hawkes.cc

Input: sequences of activities

- user-id u, item-id i, time1, time2, time3, ...



Load sequences of activities

```
// suppose 64 users and 64 items
unsigned num_users = 64, num_items = 64;
unsigned dim = num_users * num_items;

std::vector<Sequence> data;
std::cout << "1. Loading " << num_users << " users " << num_items << " items " << " with 1000 events each" << std::endl;

// load sequences from user-item pairs
ImportFromExistingUserItemSequences("data/low_rank_hawkes_sampled_entries_events", num_users, num_items, data);
```

Set options

```
// initialize low-rank Hawkes process
LowRankHawkesProcess low_rank_hawkes(num_users, num_items, beta);

LowRankHawkesProcess::OPTION options;
// regularization coefficients of the base and the excitation matrix
options.coefficients[LowRankHawkesProcess::LAMBDA0] = 1;
options.coefficients[LowRankHawkesProcess::LAMBDA] = 1;

// learning rate
options.ini_learning_rate = 1e-2;

// estimation of the nuclear norm upper bound
options.ub_nuclear_lambda0 = 25;
options.ub_nuclear_alpha = 25;

// estimation error penalization
options.rho = 1e1;

options.ini_max_iter = 1000;
```

Learning

```
// pass true parameters for comparison
Eigen::MatrixXd TrueLambda0, TrueAlpha;
LoadEigenMatrixFromTxt("data/truth-syn-Lambda0", num_users, num_items,
    TrueLambda0);
LoadEigenMatrixFromTxt("data/truth-syn-Alpha", num_users, num_items,
    TrueAlpha);

// start to fit
std::cout << "2. Fitting Parameters " << std::endl;
low_rank_hawkes.fit(data, options, TrueLambda0, TrueAlpha);
```

Time sensitive recommendation

```
// select testing user
unsigned test_userID = 0;

// select given testing time
double t = 100;

// recommend item
std::cout << "3. Predicted Item for User " << test_userID <<": " <<
    low_rank_hawkes.PredictNextItem(test_userID, t, data) << std::endl;
```

```
// select testing user
test_userID = 24;

// select testing item
unsigned test_itemID = 6;

// set maximum observation window
double observation_window = 2000;

// predict returning time
std::cout << "4. Predicted next event for user " << test_userID <<" and
    item " << test_itemID <<": " << low_rank_hawkes.PredictNextEventTime(
    test_userID, test_itemID, observation_window, data) << std::endl;
```

Running

```
1. Loading 64 users 64 items  with 1000 events each
2. Fitting Parameters
 Iteration      Step Length     Function Val     Base intensity MAE    Excitation matrix MAE
         1                1          2225.05               0.292325                 0.298426
         2         0.666667          2173.08               0.216273                 0.225375
         3              0.5          1964.49               0.172853                 0.183434
         4              0.4          1886.79               0.140363                 0.155322
         5         0.333333          1831.16                0.12192                 0.133455
         6         0.285714          1815.87               0.116931                 0.121319
         7             0.25          1811.53               0.108687                 0.117834
         8         0.222222          1806.45               0.113883                 0.112743
         9              0.2          1815.31               0.107994                 0.115142
        10         0.181818          1810.84               0.116839                 0.111769
        11         0.166667          1823.03               0.112816                 0.118249
        12         0.153846          1818.99               0.125171                 0.116117
        13         0.142857           1838.8               0.121897                 0.122979
        14         0.133333          1828.94               0.133864                 0.122622
        15            0.125          1852.08                0.12985                 0.127855
        16         0.117647          1835.26               0.140701                 0.128251
        17         0.111111          1861.15               0.135881                 0.132231

       980       0.00203874          1756.68              0.0194458                0.0585401
       981       0.00203666           1756.7              0.0194354                0.0584736
       982       0.00203459          1756.68              0.0194194                0.0585626
       983       0.00203252          1756.71              0.0194103                0.0584961
       984       0.00203046          1756.68              0.0193923                0.0585848
       985        0.0020284          1756.71              0.0193844                0.0585183
       986       0.00202634          1756.68              0.0193654                0.0586068
       987       0.00202429          1756.71              0.0193587                0.0585402
       988       0.00202224          1756.68              0.0193389                0.0586285
       989        0.0020202          1756.71               0.019333                0.0585618
       990       0.00201816          1756.68              0.0193136                0.0586499
       991       0.00201613          1756.71              0.0193087                0.0585833
       992        0.0020141          1756.68              0.0192895                0.0586712
       993       0.00201207          1756.71              0.0192854                0.0586046
       994       0.00201005          1756.68               0.019267                0.0586923
       995       0.00200803          1756.71              0.0192635                0.0586256
       996       0.00200602          1756.68              0.0192468                0.0587132
       997       0.00200401          1756.71              0.0192434                0.0586465
       998         0.002002          1756.68              0.0192277                0.0587339
       999            0.002          1756.71              0.0192248                0.0586672
      1000         0.001998          1756.68              0.0192105                0.0587544
3. Predicted Item for User 0: 46
4. Predicted next event for user 24 and item 6: 1983.19
```

More examples

- Learning standard Hawkes processes
- Support customized triggering kernels for Hawkes
- Learning standard self-correcting processes
- Support customized point processes
- Basic residual analysis
- Efficient simualtions
- ……
- Check out the project website

http://www.cc.gatech.edu/%7Endu8/ptpack/html/index.html