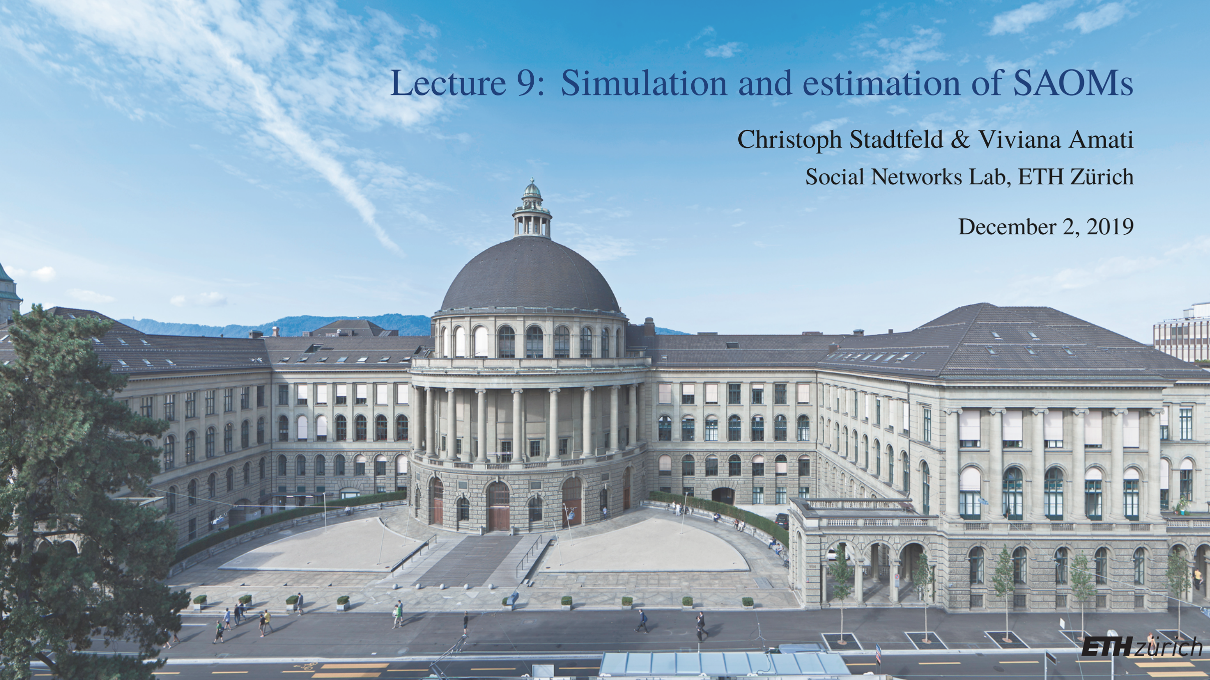
Lecture 9: Simulation and estimation of SAOMs

Christoph Stadtfeld & Viviana Amati

Social Networks Lab, ETH Zürich

December 2, 2019



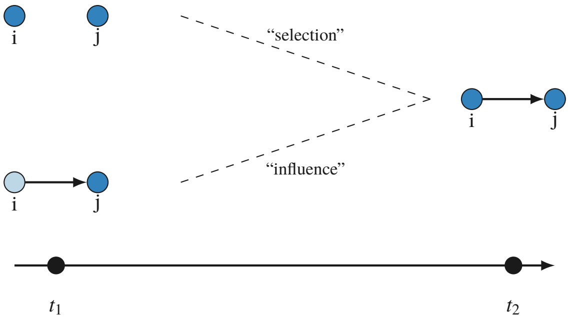
ETH zürich

Back to stochastic actor-oriented models

- Model for network panel data
  observation of a network at discrete time points
- Developed to explain the micro-mechanism that led to the network evolution
- Extended to explain the micro-mechanism that led to the co-evolution of networks and behaviours
  The only model enabling to distinguish selection from influence mechanisms

In the following, we denote by X the network and by Z the behaviour

Competing explanatory stories



- Study of influence requires the consideration of selection and vice versa
- Only longitudinal data allows distinguishing between selection and influence

Model formulation: network dynamics

- Rate function $\tau_m^{[X]}$: waiting time between opportunities for a network change
  rate describing the average number of opportunities of change between $x(t_{m-1})$ and $x(t_m)$
- Evaluation function: choice of actor i

$$f_i^{[X]}(\beta, x', z) = \sum_{k=1}^{K} \beta_k^{[X]} \, s_{ik}^{[X]}(x', z)$$

Conditional on i

$$p_{(x',z)}^{[X]} = \frac{\exp\left(f_i^{[X]}(\beta, x', z)\right)}{\sum_{x''} \exp\left(f_i^{[X]}(\beta, x'', z)\right)}$$

with $x' = x(i \rightsquigarrow j)$

Model formulation: behavioural dynamics

- Rate function $\tau_m^{[Z]}$: waiting time between opportunities for a behavioural change
  rate describing the average number of opportunities of change between $z(t_{m-1})$ and $z(t_m)$

- Evaluation function: choice of actor i

$$f_i^{[Z]}(\beta, x, z') = \sum_{k=1}^{K} \beta_k^{[Z]} \, s_{ik}^{[Z]}(x, z')$$

Conditional on i

$$p_{(x,z')}^{[Z]} = \frac{\exp\left(f_i^{[Z]}(\beta, x, z')\right)}{\sum_{z''} \exp\left(f_i^{[Z]}(\beta, x, z'')\right)}$$

with $z' = \{z, z+1, z-1\}$

For convenience, we omit v and w from the arguments of s_{ik}

Learning objectives of today

- Learn how to simulate from SAOMs
- Understand the Method of Moments (MoM) estimation
- Understand the Maximum likelihood estimation
- Learn about the stochastic approximation algorithm in SAOMs

Simulating network evolution

Simulate the longitudinal panel data $x(t_1), \ldots, x(t_M)$

For simplicity, we assume $M = 2$ (extension to more than two waves is straightforward)

Input

$x(t_1)$ = network at time t_1

τ = rate parameter

$\beta = (\beta_1, \ldots, \beta_k)$ = vector of evaluation function parameters

Output

$x^{\text{sim}}(t_2)$ = network at time t_2

Simulating network evolution

Algorithm: Network evolution

Input: $x(t_1), \tau, \beta, n$
Output: $x^{\text{sim}}(t_2)$

$t \leftarrow 0$

$x \leftarrow x(t_1)$

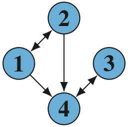
while $condition = TRUE$ **do**

- $\mathrm{dt} \sim \mathrm{Exp}(\mathrm{n}\tau)$
- $i \sim \mathrm{Uniform}(1, \ldots, \mathrm{n})$
- $j \sim \mathrm{Multinomial}(\mathrm{p_{i1}}, \ldots, \mathrm{p_{in}})$
- **if** $i \neq j$ **then**
  - $x \leftarrow x(i \rightsquigarrow j)$
- **else**
  - $x \leftarrow x$
- $t \leftarrow t + dt$

$x^{\text{sim}}(t_2) \leftarrow x$

return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from



$n = 4$

$\tau = 1.5$

$\beta = (\beta_{out}, \beta_{rec}, \beta_{trans})$
=(-1,0.5,-0.25)

Simulating network evolution

Algorithm: Network evolution

Input: $x(t_1)$, τ, β, n
Output: $x^{\text{sim}}(t_2)$

$t \leftarrow 0$
$x \leftarrow x(t_1)$
while $condition = TRUE$ **do**
  $\text{dt} \sim \text{Exp}(\text{n}\tau)$
  $i \sim \text{Uniform}(1, \ldots, \text{n})$
  $j \sim \text{Multinomial}(\text{p}_{\text{i}1}, \ldots, \text{p}_{\text{in}})$
  if $i \neq j$ **then**
    $x \leftarrow x(i \rightsquigarrow j)$
  else
    $x \leftarrow x$
  $t \leftarrow t + dt$
$x^{\text{sim}}(t_2) \leftarrow x$
return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from

Generate the time elapsed between t_1 and the first opportunity to make a change

The more intuitive way to generate dt is:

- generate the waiting time for each actor i

$$t_i \sim \text{Exp}(\tau)$$

- $dt = \min_{1 \leq i \leq n} \{t_i\}$

But this requires the generation of n numbers...

Simulating network evolution

Algorithm: Network evolution

Input: $x(t_1), \tau, \beta, n$

Output: $x^{\text{sim}}(t_2)$

$t \leftarrow 0$

$x \leftarrow x(t_1)$

while $condition = TRUE$ **do**

  $\mathrm{dt} \sim \mathrm{Exp}(\mathrm{n}\tau)$

  $i \sim \mathrm{Uniform}(1, \ldots, \mathrm{n})$

  $j \sim \mathrm{Multinomial}(\mathrm{p}_{\mathrm{i}1}, \ldots, \mathrm{p}_{\mathrm{in}})$

  if $i \neq j$ **then**

    $x \leftarrow x(i \rightsquigarrow j)$

  else

    $x \leftarrow x$

  $t \leftarrow t + dt$

$x^{\text{sim}}(t_2) \leftarrow x$

return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from

Generate the time elapsed between t_1 and the first opportunity for a change

To avoid the generation of n numbers, we use the following result:

If

$$T_i \sim \mathrm{Exp}(\tau_{\mathrm{i}}), \quad 1 \leq \mathrm{i} \leq \mathrm{n}$$

and $T_1, \ldots, T_n$ are mutually independent, then

$$dt = \min\{T_1, \ldots, T_n\} \sim \mathrm{Exp}(\sum_{\mathrm{i}=1}^{\mathrm{n}} \tau_{\mathrm{i}})$$

e.g. $dt = 0.0027$

Simulating network evolution

Algorithm: Network evolution

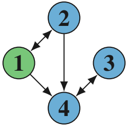
Input: $x(t_1)$, τ, β, n
Output: $x^{\text{sim}}(t_2)$

$t \leftarrow 0$
$x \leftarrow x(t_1)$
while $condition = TRUE$ **do**
  $\mathrm{dt} \sim \mathrm{Exp}(\mathrm{n}\tau)$
  $i \sim \mathrm{Uniform}(1, \ldots, \mathrm{n})$
  $j \sim \mathrm{Multinomial}(\mathrm{p}_{\mathrm{i}1}, \ldots, \mathrm{p}_{\mathrm{in}})$
  if $i \neq j$ **then**
    $x \leftarrow x(i \rightsquigarrow j)$
  else
    $x \leftarrow x$
  $t \leftarrow t + dt$
$x^{\text{sim}}(t_2) \leftarrow x$
return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from

Select the actor i who has the opportunity to change

e.g. $i = 1$



ETH zürich

Simulating network evolution

Algorithm: Network evolution

Input: $x(t_1)$, τ, β, n
Output: $x^{\text{sim}}(t_2)$
$t \leftarrow 0$
$x \leftarrow x(t_1)$
while $condition = TRUE$ **do**
  $\mathrm{dt} \sim \mathrm{Exp}(\mathrm{n}\tau)$
  $i \sim \mathrm{Uniform}(1, \ldots, \mathrm{n})$
  $j \sim \mathrm{Multinomial}(\mathrm{p}_{\mathrm{i}1}, \ldots, \mathrm{p}_{\mathrm{in}})$
  if $i \neq j$ **then**
    $x \leftarrow x(i \rightsquigarrow j)$
  else
    $x \leftarrow x$
  $t \leftarrow t + dt$
$x^{\text{sim}}(t_2) \leftarrow x$
return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from

Select j

$i \rightarrow j$	f_i	p_{ij}
$1 \rightarrow 1$	-1.75	0.15
$1 \rightarrow 2$	-1.00	0.31
$1 \rightarrow 3$	-3.25	0.03
$1 \rightarrow 4$	-0.5	0.51

Simulating network evolution

Algorithm: Network evolution

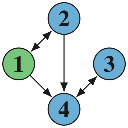
Input: $x(t_1)$, τ, β, n
Output: $x^{\text{sim}}(t_2)$

```
t ← 0
x ← x(t_1
while condition = TRUE do
    dt ~ Exp(nτ)
    i ~ Uniform(1, ..., n)
    j ~ Multinomial(p_i1, ..., p_in)
    if i ≠ j then
        x ← x(i ⇝ j)
    else
        x ← x
    t ← t + dt
x^sim(t_2) ← x
return x^sim(t_2)
```

t = time
dt = holding time between consecutive opportunities to change
~ = generated from

Select j

e.g. $j = 4$



Simulating network evolution

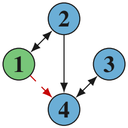
Algorithm: Network evolution

Input: $x(t_1)$, τ, β, n
Output: $x^{\text{sim}}(t_2)$
$t \leftarrow 0$
$x \leftarrow x(t_1$
while $condition = TRUE$ **do**
    $\text{dt} \sim \text{Exp}(\text{n}\tau)$
    $i \sim \text{Uniform}(1, \ldots, \text{n})$
    $j \sim \text{Multinomial}(\text{p}_{\text{i1}}, \ldots, \text{p}_{\text{in}})$
    if $i \neq j$ **then**
        $x \leftarrow x(i \rightsquigarrow j)$
    else
        $x \leftarrow x$
    $t \leftarrow t + dt$
$x^{\text{sim}}(t_2) \leftarrow x$
return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from

Select j

e.g. $j = 4$



Simulating network evolution

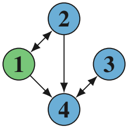
Algorithm: Network evolution

Input: $x(t_1), \tau, \beta, n$
Output: $x^{\text{sim}}(t_2)$
$t \leftarrow 0$
$x \leftarrow x(t_1)$
while $condition = TRUE$ **do**
  $\text{dt} \sim \text{Exp}(\text{n}\tau)$
  $i \sim \text{Uniform}(1, \ldots, \text{n})$
  $j \sim \text{Multinomial}(\text{p}_{\text{i1}}, \ldots, \text{p}_{\text{in}})$
  if $i \neq j$ **then**
    $x \leftarrow x(i \rightsquigarrow j)$
  else
    $x \leftarrow x$
  $t \leftarrow t + dt$
$x^{\text{sim}}(t_2) \leftarrow x$
return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from

Select j

e.g. $j = 1$



Simulating network evolution

Algorithm: Network evolution

Input: $x(t_1)$, τ, β, n
Output: $x^{\text{sim}}(t_2)$

$t \leftarrow 0$
$x \leftarrow x(t_1)$
while $condition = TRUE$ **do**
    $\text{dt} \sim \text{Exp}(\text{n}\tau)$
    $i \sim \text{Uniform}(1, \ldots, \text{n})$
    $j \sim \text{Multinomial}(\text{p}_{\text{i1}}, \ldots, \text{p}_{\text{in}})$
    if $i \neq j$ **then**
        $x \leftarrow x(i \rightsquigarrow j)$
    else
        $x \leftarrow x$
    $t \leftarrow t + dt$
$x^{\text{sim}}(t_2) \leftarrow x$
return $x^{\text{sim}}(t_2)$

t = time
dt = holding time between consecutive opportunities to change
$\sim$ = generated from

e.g. $t = 0 + 0.0027$

Simulating network evolution

Two different stopping rules:

1. *Unconditional* simulation:

   the simulation of the network evolution carries on until a predetermined time length has elapsed (usually until $t = 1$)

Simulating network evolution

Two different stopping rules:

1. *Unconditional* simulation:

   the simulation of the network evolution carries on until a predetermined time length has elapsed (usually until $t = 1$)

2. *Conditional* simulation on the observed number of changes:

   the simulation runs on until

$$\sum_{\substack{i,j=1\\ \text{ß}\neq j}}^{n} \left| x_{ij}^{obs}(t_2) - x_{ij}(t_1) \right| = \sum_{\substack{i,j=1\\ \text{ß}\neq j}}^{n} \left| x_{ij}^{\text{sim}}(t_2) - x_{ij}(t_1) \right|$$

Simulating the co-evolution of networks and behaviour

Simulate the longitudinal panel data $(x, z)_{t_1}, \ldots, (x, z)_{t_M}$

For simplicity, we assume $M = 2$ (extension to more than two waves is straightforward)

Input

n: number of actors (given)

$\tau^{[X]}, \tau^{[Z]}$: network and behaviour rate parameters (given)

$\beta^{[X]} = (\beta_1^{[X]}, \ldots, \beta_K^{[X]})$: network evaluation function parameters (given)

$\beta^{[Z]} = (\beta_1^{[Z]}, \ldots, \beta_K^{[Z]})$: behavioural evaluation function parameters (given)

$(x, z)(t_1)$: network and behaviour at time t_1 (given)

Output

$(x, z)^{\mathrm{sim}}(t_2)$: network and behaviour at time t_2

Simulating the co-evolution of networks and behavior

Algorithm: Network-behavior co-evolution

Input: $x(t_1), z(t_1), \tau^{[X]}, \tau^{[Z]}, \beta^{[X]}, \beta^{[Z]}, n$
Output: $x^{\text{sim}}(t_2), z^{\text{sim}}(t_2)$

$t \leftarrow 0$; $x \leftarrow x(t_1)$; $z \leftarrow z(t_1)$
while *condition=TRUE* **do**
    $\mathrm{dt}^{[X]} \sim \mathrm{Exp}(n\tau^{[X]})$; $\mathrm{dt}^{[Z]} \sim \mathrm{Exp}(\mathrm{n}\tau^{[Z]})$
    if $\min\{\mathrm{dt}^{[X]}, \mathrm{dt}^{[Z]}\} = \mathrm{dt}^{[X]}$ **then**
        $i \sim \mathrm{Uniform}(1, \ldots, n)$
        $j \sim \mathrm{Multinomial}(\mathrm{p}_{\mathrm{i1}}, \ldots, \mathrm{p}_{\mathrm{in}})$
        if $i \neq j$ **then**
            $x \leftarrow x(i \rightsquigarrow j)$
        $t \leftarrow t + dt^{[X]}$
    else
        $i \sim \mathrm{Uniform}(1, \ldots, \mathrm{n})$
        $z' \sim \mathrm{Multinomial}(\mathrm{p}_{\mathrm{z(z-1)}}, \mathrm{p}_{\mathrm{zz}}, \mathrm{p}_{\mathrm{z(z+1)}})$
        if $z \neq z'$ **then**
            $z \leftarrow z'$
        $t \leftarrow t + dt^{[Z]}$

$x^{\text{sim}}(t_2) \leftarrow x$; $z^{\text{sim}}(t_2) \leftarrow z$
return $x^{\text{sim}}(t_2), \; z^{\text{sim}}(t_2)$

Parameter estimation

Given the longitudinal data

$$(x, z)_{t_1}, \ldots, (x, z)_{t_M}$$

we need to estimate 2(M-1)+K+W parameters

- $M - 1$ rate parameters for the network rate function

$$\tau_1^{[X]}, \ \ldots, \ \tau_{M-1}^{[X]}$$

- $M - 1$ rate parameters for the behaviour rate function

$$\tau_1^{[Z]}, \ \ldots, \ \tau_{M-1}^{[Z]}$$

- K and W parameters for the network and behavioural evaluation function, respectively

$$f_i^{[X]}(\beta^{[X]}, x', z) = \sum_{k=1}^{K} \beta_k^{[X]} s_{ik}^{[X]}(x', z)$$

$$f_i^{[Z]}(\beta^{[Z]}, x, z') = \sum_{w=1}^{W} \beta_w^{[Z]} s_{iw}^{[Z]}(x, z')$$

Estimating the parameters of SAOMs

We would like to estimate

$$\theta = (\tau^{[X]}, \tau^{[Z]}, \beta^{[X]}, \beta^{[Z]}), \quad \theta \in \mathbb{R}^P, \quad P = 2(M-1) + K + W$$

1. Method of Moments (MoM) and its generalization (GMoM)
2. Maximum-likelihood estimation
3. Bayesian estimation

Estimating the parameters of SAOMs

We would like to estimate

$$\theta = (\tau^{[X]}, \tau^{[Z]}, \beta^{[X]}, \beta^{[Z]}), \quad \theta \in \mathbb{R}^P, \quad P = 2(M-1) + K + W$$

1. **Method of Moments (MoM)** and its generalization (GMoM)
2. Maximum-likelihood estimation
3. Bayesian estimation

Estimating the parameters of SAOMs: MoM

$$\theta = (\tau^{[X]}, \tau^{[Z]}, \beta_1^{[X]}, \dots, \beta_K^{[X]}, \beta_1^{[Z]}, \dots, \beta_W^{[Z]}), \quad \theta \in \mathbb{R}^P, \quad P = 2(M-1) + K + W$$

1. Find P statistics $s(X) = (s_1(X, Z), \dots, s_p(X, Z))$
   i.e. P variables that can be calculated from the network
2. Set the expected value of $s(X, Z)$ equal to its sample counterpart $s(x, z)$
   $$E_\theta[s(X, Z)] = s(x, z)$$
3. Solve the resulting system of equations with respect to θ.

1. Defining the statistics

The statistics $s(X)$ must be sensitive to the parameter θ in the sense that

$$\frac{\partial E_\theta(s_p(x,z))}{\partial \theta_p} > 0$$

- Rate function:
  - τ models the frequency at which actors get opportunities for a change
  - The higher τ_m, the higher the number of changes between t_{m-1} and t_m
  - Relevant statistics for τ are

$$s_\tau^{[X]}(X(t_m), X(t_{m-1})) = \sum_{ij} \left| X_{ij}(t_m) - X_{ij}(t_{m-1}) \right|$$

$$s_\tau^{[Z]}(Z(t_m), Z(t_{m-1})) = \sum_{i} \left| Z_i(t_m) - Z_i(t_{m-1}) \right|$$

Why do we sum over m?

1. Defining the statistics

- Evaluation function:

  for the parameter β_k in

$$f_i(\beta^{[X]}, x, z) \quad \text{and} \quad \mathrm{f_i}(\beta^{[Z]}, \mathrm{x}, \mathrm{z})$$

  - the higher β_k, the more strongly all actors strive after a high value of $s_{ik}(x, z)$
  - this leads to the statistics

$$s_k^{[X]}(X(t_m), Z(t_{m-1})) = \sum_i s_{ik}^{[X]}(X(t_m), Z(t_{m-1}))$$

$$s_k^{[Z]}(X(t_{m-1}), Z(t_m)) = \sum_i s_{ik}^{[Z]}(X(t_{m-1}), Z(t_m))$$

2. Setting the moment equations

- Rate function

$$E_\theta\left[s_\tau^{[X]}(X(t_m), X(t_{m-1})) \mid (X,Z)_{t_{m-1}}\right] = \sum_{ij} \left|x_{ij}(t_m) - x_{ij}(t_{m-1})\right| \quad m = 2, \ \ldots, \ M$$

$$E_\theta\left[s_\tau^{[Z]}(Z(t_m), Z(t_{m-1})) \mid (X,Z)_{t_{m-1}}\right] = \sum_{i} |z_i(t_m) - z_i(t_{m-1})| \qquad m = 2, \ \ldots, \ M$$

- Evaluation function

$$\sum_{m=2}^{M} E_\theta\left[s_k^{[X]}(X(t_m), Z(t_{m-1}) \mid (X,Z)_{t_{m-1}}\right] = \sum_{m=2}^{M} s_k^{[X]}(x(t_m), z(t_{m-1}))$$

$$\sum_{m=2}^{M} E_\theta\left[s_k^{[Z]}(X(t_{m-1}), Z(t_m)) \mid (X,Z)_{t_{m-1}}\right] = \sum_{m=2}^{M} s_k^{[Z]}(x(t_{m-1}), z(t_m))$$

3. Solving the moment equation

The system of moment equation

$$E_\theta[s(X,Z)] = s(x,z)$$

cannot be solved by analytical or the usual numerical procedures, because

$$E_\theta[s(X,Z)]$$

cannot be calculated explicitly.

However, the solution can be approximated by the
Robbins-Monro (1951) method for stochastic approximation

an iterative stochastic algorithm that attempts to find zeros of functions which cannot be analytically computed

Stochastic approximation algorithm

- Iterative algorithm with step

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \alpha_i \, D^{-1} \left[s^{(i)}(x,z) - s(x,z) \right]$$

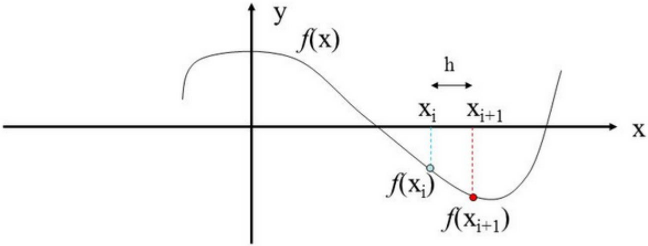
- The algorithm has three phases (Snijders (2001)):
  1. Phase 1: given an initial value θ_0, n_1 simulations of the network evolution is used to approximate the matrix of the first order derivatives D and update θ
  2. Phase 2: divided into a few sub-phases characterized by a decreasing value of α_i
     - A step in each sub-phase consists in generating one network evolution and updating the value of $\hat{\theta}_i$ via the Robbins-Monro step
     - At the end of the sub-phase α_i is halved and the estimate for θ is the average of the values $\hat{\theta}_i$
     - This estimate is used as the initial value of θ for the next sub-phase
  3. Phase 3: is used to check the convergence of the algorithm and compute the standard errors of the estimates based on a large number of network trajectories simulated from the value of θ obtained at the end of phase 2

Stochastic approximation algorithm

Phase 1: Approximation of $D = \frac{\partial E_\theta[S]}{\partial \theta}$

- Finite difference method:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{h}$$



Stochastic approximation algorithm

Phase 1: Approximation of $D = \frac{\partial E_\theta[S]}{\partial \theta}$

- Finite difference method: at each step j, $j = 1, \ldots, n_1$,
  - simulate one network evolution from a SAOM with $\theta = \theta_0$ and compute the vector of statistics S_{j0}
  - define the vectors $\theta_{kj} = \theta_0 + \varepsilon_k e_k$
    e_k the k-th unit vector in P dimensions, ε_k suitable constants
  - simulate from θ_{kj}, compute the the statistic S_{jk} and the ratio

$$d_{jk} = \frac{S_{jk} - S_{j0}}{\varepsilon_k}$$

    for each element of θ

  D is approximated by

$$\hat{D} = \frac{1}{n_1} \sum_{j=1}^{n_1} d_j$$

  with $d_j = (d_{j1}, \ldots, d_{jp})$

Stochastic approximation algorithm

Phase 1: Approximation of $D = \frac{\partial E_\theta[S]}{\partial \theta}$

- Score function method
  - Score function: first order derivative of the log-likelihood

$$SF = \frac{\partial \ell(\theta)}{\partial \theta}$$

  - We can prove that

$$\frac{\partial E_\theta[(S - s)]}{\partial \theta} = E_\theta[(S - s) \cdot SF]$$

    and approximate D using the score function method: (Schweinberger and Snijders (2007))

$$\hat{D} = \frac{1}{n_1} \sum_{j=1}^{n_1} (S_j - s) SF_j$$

  - The use of the score function method provide a better approximation

Stochastic approximation algorithm

Phase 1: Update θ

- Newton Raphson step

$$\hat{\theta}_1 = \theta_0 - \hat{D}^{-1} \left(\overline{s} - s \right)$$

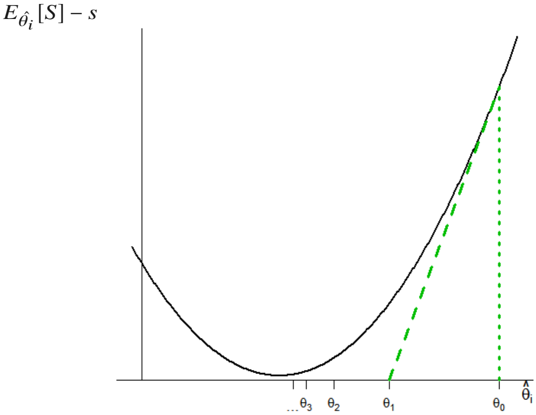
with

$$\overline{s} = \frac{1}{n_1} \sum_{j=1}^{n_1} S_{j0}$$

Stochastic approximation algorithm

Phase 2: the Robbins-Monro step

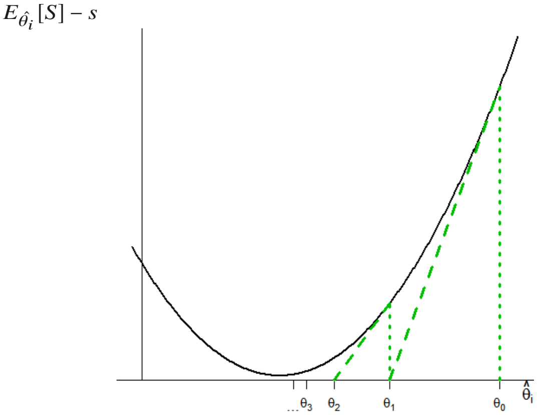
(Intuitively, in one dimension)



Stochastic approximation algorithm

Phase 2: the Robbins-Monro step

(Intuitively, in one dimension)

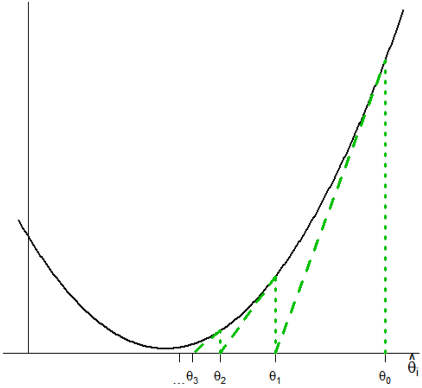


Stochastic approximation algorithm

Phase 2: the Robbins-Monro step

(Intuitively, in one dimension)

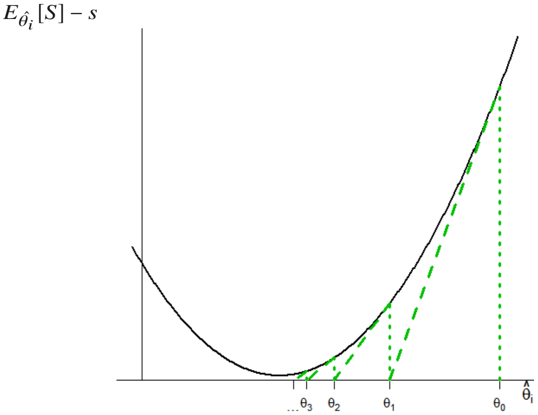
$E_{\hat{\theta}_i}[S] - s$



Stochastic approximation algorithm

Phase 2: the Robbins-Monro step

(Intuitively, in one dimension)



Stochastic approximation algorithm

Phase 2: Sub-phase h

Each sub-phase consists of the following steps:

- Generate one network evolution from the current value of $\hat{\theta}$
- Update θ

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \alpha_h \, \tilde{D}^{-1} \left[\, s^{(i)}(x,z) - s(x,z) \,\right], \quad \tilde{D} = \mathrm{diag}(\hat{\mathrm{D}})$$

The steps are repeated until the maximum number of steps n^*_{2h} is reached or

$$\sum_i (s_k^{(i-1)}(x,z) - s_k(x,z))(s_k^{(i)}(x,z) - s_k(x,z)) < 0, \ \forall k$$

At the end of the sub-phase:

$$\hat{\theta} = \frac{1}{n_{2h}} \sum_i^{n_{2h}} \hat{\theta}_i \quad \alpha_{h+1} = \alpha_h / 2$$

with n_{2h} the number of steps performed in sub-phase h

Stochastic approximation algorithm

Phase 3: Checking convergence

- The value $\hat{\theta}$ returned by the last sub-phase in phase 2 is the estimate for θ
- A large number n_3 of network evolution is simulated given $\hat{\theta}$
- For each simulation, the vector of statistics S_j is computed

$$\text{t-conv}_k = \frac{\overline{s}_k - s_k}{\text{s.d}(s_k)}$$

$$\text{maximum t-ratio} = (\overline{s} - s)' \Sigma^{-1} (\overline{s} - s)$$

with

$$\overline{s} = \frac{1}{n_3} \sum_{j=1}^{n_3} s_j, \quad j = 1, \ldots, n_3$$

and

$$\Sigma = \text{Cov}[S] = \frac{1}{n_3} s s' - \overline{s}' \overline{s}, \quad j = 1, \ldots, n_3$$

Stochastic approximation algorithm

Phase 3: Approximate the covariance matrix of the MoM estimator

We approximate the covariance matrix of the MoM estimator using the delta method

$$\mathrm{Cov}[\hat{\theta}] \approx \hat{\mathrm{D}}^{-1}\Sigma\hat{\mathrm{D}}^{-1}$$

with

- $\hat{D}$ the matrix of first order derivatives approximated as in phase 1
- $\Sigma = \mathrm{Cov}[S] = \frac{1}{n_3} s s' - \overline{s}'\overline{s}, \quad j = 1, \ldots, n_3$

ETH zürich

Estimating the parameters of SAOMs

We would like to estimate

$$\theta = (\tau^{[X]}, \tau^{[Z]}, \beta^{[X]}, \beta^{[Z]}), \quad \theta \in \mathbb{R}^P, \quad P = 2(M-1) + K + W$$

1. Method of Moments (MoM) and its generalization (GMoM)
2. **Maximum-likelihood estimation**
3. Bayesian estimation

Computing the (log-)likelihood of the evolution process

(Snijders et al., 2010)

The model assumptions allow to decompose the process in a series of micro-steps:

$$\{(T_r, i_r, j_r), r = 1, \ldots, R\}$$

- T_r: time point for an opportunity for change,
- i_r: actor who has the opportunity to change
- j_r: actor towards whom the tie is changed

Given the sequence $\{(T_r, i_r, j_r),\ r = 1, \ldots, R\}$, the likelihood of the evolution process

$$\log[p(x(t_2)|x(t_1);\theta)] = \log\left[\prod_{r=1}^{R} P_\theta((T_r, i_r, j_r))\right] \propto \log\left[\underbrace{\frac{(N\tau)^R}{R!}e^{-N\tau}}_{\text{Prob. R steps}} \prod_{r=1}^{R} \underbrace{\frac{1}{N} p_{i_r j_r}(\beta, x(T_r))}_{\text{Prob. }(i_r, j_r)}\right]$$

Maximizing the (log-)likelihood

- Assuming M=2 for simplicity, we look for the value $\hat{\theta}$ such that

$$\hat{\theta} = \max_{\theta \in \Theta} \log[p(x(t_2)|x(t_1);\theta)]$$

  or equivalently

$$\hat{\theta} : \underbrace{\frac{\partial}{\partial \theta} \log[p(x(t_2)|x(t_1);\theta)]}_{\text{Score function}} = 0$$

- We cannot observe the complete data, i.e., the complete series of mini-steps that lead from $x(t_1)$ to $x(t_2)$, and thus we cannot compute the likelihood of the observed data
- A stochastic approximation method must be applied

Maximizing the (log-)likelihood

Augmented data method

The *augmented data* (or *sample path*) consists of the sequence of tie changes that brings the network from $x(t_1)$ to $x(t_2)$

$$(i_1, j_1), \ldots, (i_R, j_R)$$

Formally:

$$\underline{v} = \{(i_1, j_1), \ldots, (i_R, j_R)\} \in \mathcal{V}$$

where $\mathcal{V}$ is the set of all sample paths connecting $x(t_1)$ and $x(t_2)$.

Maximizing the (log-)likelihood

Stochastic approximation method: approximation

Augmented data method

The *augmented data* (or *sample path*) consists of the sequence of tie changes that brings the network from $x(t_1)$ to $x(t_2)$

$$(i_1, j_1), \dots, (i_R, j_R)$$

Formally:

$$\underline{v} = \{(i_1, j_1), \dots, (i_R, j_R)\} \in \mathcal{V}$$

where $\mathcal{V}$ is the set of all sample paths connecting $x(t_1)$ and $x(t_2)$.

We can approximate the (log-)likelihood function of the observed data using the probability of $\underline{v}$

$$\log[p(\underline{v}|x(t_1), x(t_2))] \propto \log\left[\frac{(N\tau)^R}{R!} e^{-N\tau} \prod_{r=1}^{R} \frac{1}{N} p_{i_r j_r}(\beta, x(T_r))\right]$$

Questions you should be able to answer now

- How could you simulate from SAOMs?
- How are the parameters of SAOMs estimated?
- Why we need a stochastic approximation algorithm to compute the MoM estimate?
- Why MLE cannot be computed in SAOMs?

References

Schweinberger, M. and Snijders, T. A. (2007). Markov models for digraph panel data: Monte carlo-based derivative estimation. *Computational statistics & data analysis*, 51(9):4465–4483.

Snijders, T. A. (2001). The statistical evaluation of social network dynamics. *Sociological methodology*, 31(1):361–395.

Snijders, T. A., Koskinen, J., and Schweinberger, M. (2010). Maximum likelihood estimation for social network dynamics. *The Annals of Applied Statistics*, 4(2):567.