Siena Algorithms

A discussion of methods of fitting longitudinal network models

R.M. Ripley

Department of Statistics
University of Oxford

June 2010

Very brief outline of Siena model

- Longitudinal data of two or more waves of networks.
- Markov process of changes in the network.
- Model the process of changes conditional on the waves.
- Analytical methods not feasible so use simulation methods.

  Either:

  1. Simulate from each wave forwards for a fixed time or until a fixed number of events have occurred.

  or:

  2. Use MCMC to simulate draws from the distribution of all possible processes which share the same states at each wave.

Estimation methods

- Use Method of Moments to equate simulated values of specified statistics to observed values. (Simulation method 1.)
- Use Method of Moments to equate scores of simulated process to zero. (Simulation method 2.)
- Add the parameters to the MCMC process and explore parameter space in a Bayesian manner.(Simulation method 2.)
- Solve Method of moments equations by Robbins-Monro Algorithm

Solving the moment equation

- Not tractable analytically, so use stochastic optimisation method
- At step N, adjust the parameters

$$\boldsymbol{\theta}_{N+1} = \boldsymbol{\theta}_N - a_N D^{-1}(z_N - z)$$

- where D is a suitable matrix and a_N tends to zero
- Split the procedure into phases: within each phase hold a_N constant, at end of the phase, replace $\boldsymbol{\theta}$ by the average of the $\boldsymbol{\theta}_N$ within the phase.

The matrix **`D`**

- The derivative matrix $\frac{\partial Z_i}{\partial \boldsymbol{\theta}_j}$ of the statistics Z with respect to the parameter $\boldsymbol{\theta}$ is used (or its diagonal at least), as the matrix D.
- Estimate this by simulating the process multiple times with constant $\boldsymbol{\theta}$.
- Two estimation methods for simulation method 1:
  1. finite differences or
  2. score-based.
- In method 1, use **common random numbers** to reduce variance of the estimates.
- In method 2, use an extra term for **variance reduction**. (mean zero, but, you hope, very correlated with your estimate)

Maximum likelihood

- The times of the events and any events which had no effect on the appearance of the final network are treated as missing data.
- Create a minimal chain of events which lead from one wave to next.
- An event is one toggle of one link.
- Sort these events randomly.
- Perturb this chain in a MCMC procedure to obtain a “complete” chain, with some null events (toggling the non-existent link to one’s self) and other new links balanced by later deletions.

Bayesian approach

- Sample in turn from the posterior conditional distribution of the parameters given the process and from the distribution of the process given the parameters.
- **`rate ~ Γ(chainlength+1, 1/(number of actors))`**
- For **`betas`**, add **`N(0,sd)`** to each and accept if sum of choice probabilities using new parameter is high enough compared with the old parameter. Adjust **`sd`** to get reasonable acceptance rate.
- Simply collect any statistics of interest from the results.
- Conceptually simple, but slow?

Likelihood

The likelihood of a realisation which consists of R events within the time period, conditional on the starting position (and possibly end), is

$$\prod_{r=1}^{R}\left[\exp(-n\lambda)\frac{(n\lambda)^R}{R!}\right]\prod_{r=1}^{R}\frac{1}{n}\frac{\exp(\sum\beta_i s_{ij})}{\sum\exp(\sum\beta_i s_{ik})}$$

and the ratio of the likelihoods at $\tilde{\theta}$ and θ is

$$\frac{p_{\tilde{\theta}}}{p_\theta} = exp(-n\tilde{\lambda}+n\lambda)(\frac{\tilde{\lambda}}{\lambda})^R\prod\frac{\exp(\sum\tilde{\beta}_i s_{ij})}{\sum\exp(\sum\tilde{\beta}_i s_{ik})}\frac{\sum\exp(\sum\beta_i s_{ik})}{\exp(\sum\beta_i s_{ij})}$$

Importance sampling

$$E_{\tilde{\theta}}(Z) = E_\theta\left(\frac{p_{\tilde{\theta}}(Z)}{p_\theta(Z)}Z\right)$$

Since the relative likelihoods vary for each realisation of the simulation, we could estimate by using

$$\frac{1}{N}\sum_{1}^{N}\frac{p_{\tilde{\theta}_i}(Z_i)}{p_{\theta_i}(Z_i)}Z_i \tag{1}$$

In fact, we normalise the weights to sum to 1. (We divide by the sum of the weights: the expected value of this sum is 1. This normalisation introduces a (hopefully small) bias, but should reduce the variance.)

Suggestions from Gelman (1995)

- Simulate several times, as in method 1, with the same θ.
- Alternatively, take multiple (well-enough spaced) samples from the MCMC process of method 2.
- Do an update step to give a new θ.
- Use ***importance sampling*** to estimate the value of the statistics/scores at this θ.
- Calculate the relative likelihood of *each* realisation of the process at θ and θ_0.
- Use these as weights in a weighted sum of the simulated outcomes to predict the values of the statistics/scores we would have obtained had we simulated with this θ.
- Do another update step.

Varying the ratio of types of steps

- In Siena, we use 1 observation of Z to estimate its mean, followed by 1 update step. (Except when using multiple processors, when we obtain 1 observation from each processor and average.)
- Could use importance sampling to perform more than one update step per simulation.
- i.e. calculate the new θ, guess the statistics/score with this one and calculate another.
- In general, simulation takes much longer than calculation, so this is faster.

Well, are we sure about that?

- ...simulation takes much longer than calculation....
- Calculation requires all the changes of all the effects at every event. Various possibilities:
  1. Store list of events, recalculate with the new θ using same code as for simulation. Quick to program. Competitive speed.
  2. Also store all the change contributions (a matrix for each event), return it all to R and use R matrix multiplication. Slower to program, but appealing as we get to 'see' the data. BUT very very slow to run.
  3. Store everything in C, and recalculate using the stored values in C. Even slower to program (maybe just for me!) but runs fast. But does not scale well. (Actually do rate calculations in R)

What is the best option?

- Fastest run time is to do it all in C++.
- Speed comparisons depend on size of problem, number and type of effects and simulation algorithm. (MC or MCMC). Copying, garbage collection...
- Slower C++ method is very low in memory requirements and may be only possible option in some problems.
- Using R an appealing idea, but multiple periods, groups, networks, actor sets... make it less so. Also changes to types of effects (evaluation, endowment, ...) would need more effort to incorporate.
- One optimisation (untried, but theoretically appealing), would be to store a (or a few) 3D array of contributions per chain rather than a matrix for each step. Only one unpack required.

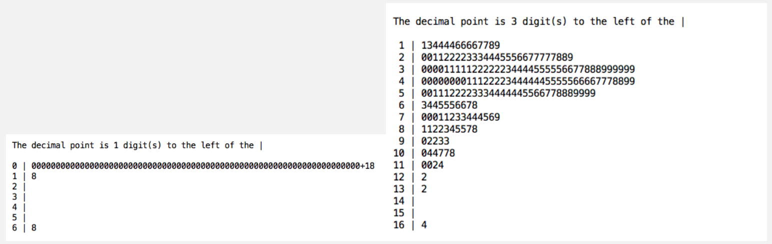
Specific proposal of Gelman

- Do a few loops of
  - a few simulations
  - a few update steps
- Do many simulations
- Do updates repeatedly to convergence.
- Monitor the stability of the importance sampling using the variance of the weights (see Kong (1992))
- Stop updating if the importance sampling is getting bad.

Experiments with this procedure

- Stem plots of the importance sampling weights very informative
- Use just diagonal of estimated derivative in update step until final run
- At start, variance of weights was high: few importance sampling steps taken.
- Once weights stable, take a large sample
- Use full derivative for the final updates.
- Need reasonably sized samples to get usable scale estimates.
- Not worth going very far, within "Monte Carlo" error.

Visual assessment of progress



```
The decimal point is 1 digit(s) to the left of the |

0 | 00000000000000000000000000000000000000000000000000000000000000000000+18
1 | 8
2 |
3 |
4 |
5 |
6 | 8
```

```
The decimal point is 3 digit(s) to the left of the |

 1 | 13444466667789
 2 | 001122223334445556677777889
 3 | 00001111122222234444555556677888999999
 4 | 00000000111222234444445555566667778899
 5 | 0011122223334444445566778889999
 6 | 3445556678
 7 | 00011233444569
 8 | 1122345578
 9 | 02233
10 | 044778
11 | 0024
12 | 2
13 | 2
14 |
15 |
16 | 4
```

Figure: Examples of importance sampling stem plots. Left one is not good, right one is useful.

Maximum Likelihood: EM

- An alternative approach is to use *Expectation-Maximisation*
- Alternate between sampling from process given parameters and maximising the expected value of the full log likelihood over the parameters
- We cannot do the expectation step exactly, so use Monte Carlo techiques.
- Once the process is in equilibrium(!), take samples of the chain at intervals (check correlations have subsided (interval will depend on the length of the chain)).
- Calculate full likelihood of each chain in the sample, and maximize mean log likelihood as a function of parameters.
- Optimisation by `optim()`, using **BFGS** method. Only need to run for one step, to improve the parameters a little.

Maximum Likelihood: Approximate EM

- Need "good enough" estimates of the log likelihood.
- Likelihood value decreases with increasing length of chain. (Multiply by terms that are less than 1, or consider there are more possibilities). So cannot compare between steps.
- In real EM, likelihood of data does not increase, but likelihood with missings filled in could.
- Straightforward with chains from MCMC
- Multiple chains from crude Monte Carlo vary too much in outcomes. (see plots on next screen). Maximum is always very close by.
- Since these simulations don't match our data we are not interested in the θ which maximises them anyway.

Likelihood surface

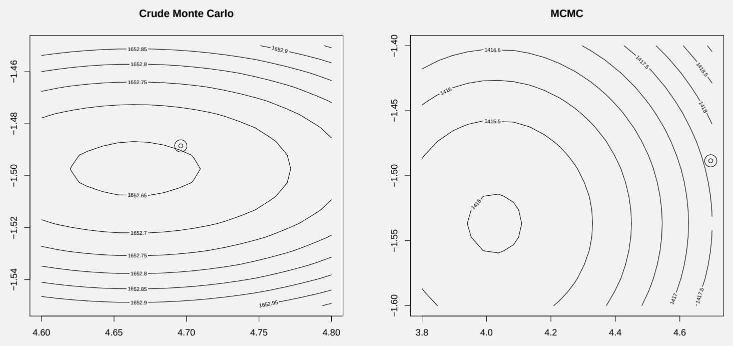


Figure: Likelihood surface with two effects, crude Monte Carlo (left) and MCMC (right), 100 samples

Maximum Likelihood: Approximate EM

- Here use Gelman's idea to reuse the same set of iterations more than once
- Weight the log-likelihoods by the ratio of the likelihood at the current $\hat{\boldsymbol{\theta}}$ to the sampled $\boldsymbol{\theta}_0$.
- Watch the simulations as for the method of moments.
- Resample when the variance of the sample weights gets large.
- Saving of time could be significant with MCMC.

Other suggestions from Cappé *et al.* (2005)

- Alter sampling rate:
  - SEM Stochastic Expectation-Maximisation: fixed sampling rate
  - MCEM Monte Carlo Expectation-Maximisation: altering (increasing) samplesize).
- Average the results... as in Robbins-Monro
- Re-use the history:
  - SAEM Stochastic Approximation Expectation-Maximisation: in maximise step, maximise a weighted sum of this samples' average log likelihood and the previous samples' optimand.

Geyer-Thompson

- Geyer-Thompson (1992) suggested approximating likelihoods directly using MonteCarlo integration.
  1. Take large set of samples at $\boldsymbol{\theta}_0$.
  2. Calculate the ratio of full likelihoods at $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_0$ *for each sample separately* and average.
  $$\frac{1}{n}\sum_1^n \frac{p_{full}(x_i|\boldsymbol{\theta})}{p_{full}(x_i|\boldsymbol{\theta}_0)} \sim \frac{p_{obs}(x|\boldsymbol{\theta})}{p_{obs}(x|\boldsymbol{\theta}_0)}$$
  3. Approximate difference in observed data log likelihoods at the two $\boldsymbol{\theta}$'s by log of this average, and maximise over $\boldsymbol{\theta}$.
  $$\max\left(\log\left(\frac{1}{n}\sum_1^n \frac{p(x_i|\boldsymbol{\theta})}{p(x_i|\boldsymbol{\theta}_0)}\right)\right) \sim \max\left(\log(p(x|\boldsymbol{\theta})) - \log(p(x|\boldsymbol{\theta}_0))\right)$$

Observed and full data likelihoods

Can consider the likelihood of interest as the normalising constant of the likelihood of the full data when sampling from the full data given the observed data:

$$p(\text{full data}|\text{observed}, \boldsymbol{\theta}) = \frac{p(\text{full data}|\boldsymbol{\theta})}{p(\text{observed}|\boldsymbol{\theta})} \equiv \frac{L(\boldsymbol{\theta}|\text{full data})}{L(\boldsymbol{\theta}|\text{observed })}$$

The ratios estimate the ratios of the normalizing constants:

$$E_{\boldsymbol{\theta}_0}\frac{p_{full}(x_i|\boldsymbol{\theta})}{p_{full}(x_i|\boldsymbol{\theta}_0)} = \int \frac{p_{full}(x_i|\boldsymbol{\theta})}{p_{full}(x_i|\boldsymbol{\theta}_0)} p_{full|obs}(x_i|\boldsymbol{\theta}_0)dx = \frac{p_{obs}(x_i|\boldsymbol{\theta})}{p_{obs}(x_i|\boldsymbol{\theta}_0)}$$

Take logs and ignore the terms that do not involve $\boldsymbol{\theta}$, and we have estimate of $L(\boldsymbol{\theta}|\text{observed})$

Compare with MCEM

- Geyer-Thompson (Sometimes referred to as Simulated Maximum Likelihood

$$\log\left(\frac{1}{n}\sum_{1}^{n}\frac{p_{full}(x_i|\boldsymbol{\theta})}{p_{full}(x_i|\boldsymbol{\theta}_0)}\right)\sim\log(p_{obs}(x|\boldsymbol{\theta}))-\log(p_{obs}(x|\boldsymbol{\theta}_0)) \quad (2)$$

- MCEM

$$\left(\frac{1}{n}\sum_{1}^{N}\log\left(p_{full}(x_i|\boldsymbol{\theta})\right)\right)\sim\log(p(_{full}x|\boldsymbol{\theta}))$$

- We have seen something like the lhs of equation (2) before: importance sampling.

MCEM or Geyer-Thompson

- Geyer-Thompson suggest their method will only be reasonable if importance sampling would be OK. i.e. near the answer.
- So iterate a few times.
- EM is notorious for slow convergence.
- Do you need a much bigger sample to get Geyer-Thompson to work to compensate for the reduced number of iterations?
- My experience is that large samples at the start are not useful: the parameters don't move far.

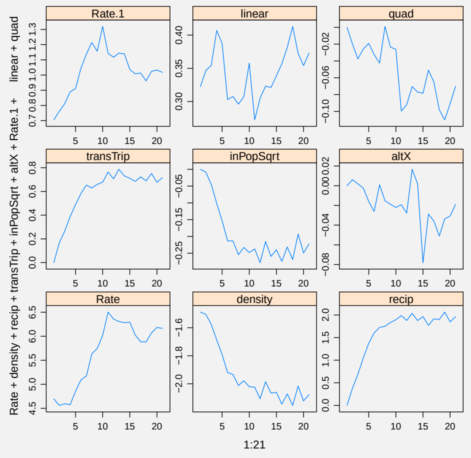
Other possibilities

- Other possibilities which I have coded and tried:
  1. Fit a linear response surface to deviations near $\boldsymbol{\theta}$, using importance sampling, and solve for zero deviations.
  2. Fit a quadratic surface to deviations and minimise sum of squared deviations. Never positive definite in my experience!
  3. Collect all the samples used in each subphase of phase2 of siena07 and use regression to find the best value of $\boldsymbol{\theta}$. This gives slightly different answers in early subphases, virtually the same by the last sub phase. May allow fewer iterations to be used.
  4. Importance sampling steps can be used within phase 2. Again may allow fewer iterations.

Summary

- Can (soon!) treat RSiena as a simulation tool box for longitudinal analysis of social networks.
- Two basic approaches considered here:
  1. Reduce a deviation (statistics versus targets or scores) by a step in an appropriate direction
  2. Maximize a log likelihood (with two competing methods)
- Using scores in the first is essentially equivalent to the second, but may behave differently in the Monte Carlo framework.

Graphical assessment of progress



Profile likelihoods

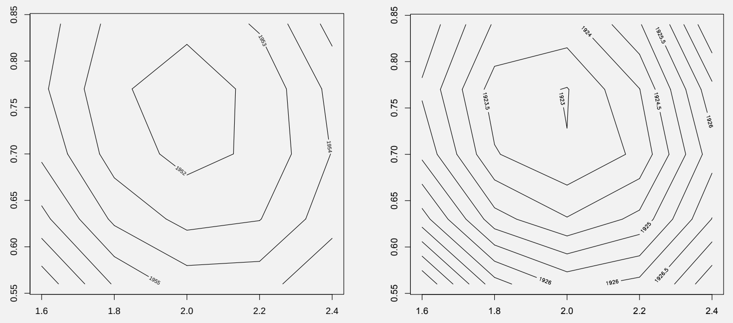


Figure: Profile likelihood, left based on 100 samples, right on 1000 (4hrs!)

Cappé, O., Moulines, E. and Rydén, T. (2005) *Inference in Hidden Markov Models.* Springer.

Gelman, A. (1995) Method of moments using Monte Carlo simulation. *Journal of Computational and Graphical Statistics*, **4**(1), 36–54.

Geyer C. J. and Thompson E. A. (1992) Constrained Monte Carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society, Series B*, **54**, 657–699.

Kong, A. (1992) A note on importance sampling using standardised weights. Technical Report 348, Department of Statistics, University of Chicago.