NME 2024

Network Modeling for Epidemics

Estimation

High level overview

ERGMs are an extension of generalized linear models

- Estimation for GLMs typically relies on computational methods

  Compared to, LMs, where there is a closed form solution for estimating the coefficients from the data directly

- For ERGM, the computational method depends on the terms in your model $g(y)$

  And there's the additional need for algorithms to compute these model terms

Before you can estimate a model

- You need to specify a model

- This requires selecting the model terms $g(y)$

- And for ERGMs, that involves an extra step

Calculating the model terms $g(y)$

- In most statistical models, the covariates, X, are directly observed in the data

- But for ERGMs, the covariates are instead network statistics, which are *functions* of the data: $g(y)$
  - So every term needs a different algorithm to calculate it
    - Some are simple – like the `edges` term
    - Some are not
  - These term algorithms are typically included in a network analysis package

Model terms $g(y)$ in ergm

- The ergm package has ~125 terms coded up
  - See the documentation for details

- But any configuration can be turned into a term
  - So the ones included in ERGM are not exhaustive

- You can code up your own terms if necessary
  - There's another statnet package for that
  - And online training materials

Moving on to estimation

- Different packages use different methods for estimation
- The `ergm` package uses Maximum Likelihood Estimation (MLE)
- So we'll start with a brief review of what that means in different contexts

Review: Maximum Likelihood Estimation (MLE)

- The likelihood equation represents the probability of the data under the model
  - $L = P(data\ |\theta)$
  - The MLE of θ is the value θ of that maximizes L -- the probability of the data under the model

- For traditional linear models
  - Observations are independent, so the likelihood function factors into a product: $L = \prod_i p(y_i|\theta)$
  - Maximization uses calculus to obtain a closed-form solution for the MLE

- For generalized linear models
  - Observations are still independent
  - But there is typically no closed form solution for the MLE
  - So computational methods are used (like iteratively re-weighted LS)

- For network models (generalized linear models for dependent data)
  - Observations may be dependent
  - Computational methods are always used for MLE

For dyad independent (DI) models

- The estimation algorithm is equivalent to that used for logistic regression
  - E.g., Iteratively reweighted least squares

- But you still can't just a standard stat package for these models
  - Because you still need to calculate the $g(y)$ statistics from your data
  - And you need a specialized network package for that

For dyad dependent (DD) models

- The observations (ties) are dependent
  - So L doesn't factor into the product of the individual probabilities
  - And we're stuck with an intractable expression

$$P(Y = y \,|\, \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}' \boldsymbol{g}(\boldsymbol{y}))}{k(\boldsymbol{\theta})}$$

Where the normalizing constant $k(\boldsymbol{\theta})$ can't be calculated

- Here, `ergm` uses Monte Carlo Markov Chain MLE
  - Effectively a network simulation algorithm that we use for estimation
  - … and later, also for model assessment and simulation

What is MCMC in this context?

1. Specify a model and set a starting value for your vector of coefficients, $\boldsymbol{\theta}$

2. Simulate networks from a model with this $\boldsymbol{\theta}$ vector
   - Select a dyad at random (possibly with weights)
   - Propose ties between nodes: “toggles”
     - Some are accepted, some not, based on the probability defined by this $\boldsymbol{\theta}$ vector
   - Every X toggles, grab the network and calculate the netstats $\boldsymbol{g}(\boldsymbol{y}_{\boldsymbol{sim(i)}})$
   - Repeat this step Y times

X and Y are typically > 1000

3. After Y sampled networks:
   - Compare the observed netstats $\boldsymbol{g}(\boldsymbol{y}_{\boldsymbol{obs}})$ to the mean of $\boldsymbol{g}(\boldsymbol{y}_{\boldsymbol{sim(i)}})$
   - Adjust the coefficients as indicated by the difference (higher, or lower)

4. Repeat step 2 & 3 until the netstats converge and the sampling uncertainty is low

Why it works

- We are getting another benefit of statistical theory here
  - Specifically, the theory of maximum likelihood estimation with exponential family models

- For (all) exponential family models:

  1. A defining property of the MLEs is that they will reproduce the observed sufficient statistics in expectation.
  2. The MLEs are unique

- For ERGMs this means
  - We can use the observed sufficient statistics, $g(y)$, to find the MLEs

    AND

  - Simulations from the fitted model will reproduce those $g(y)$ in expectation

It works … but slowly

- The larger your network
- The stronger the dependence in your model terms
- The longer this will take

- `ergm` has lots of control parameters for tweaking the MCMC process
  - In R type `?control.ergm` for more info

MCMC MLE is used a lot now

- In many different fields, not just network analysis
  - Foundation for most Bayesian estimation
  - And anytime you have dependent data

- Relatively recent development
  - The theory preceded the computational feasibility…
  - Nice review of the history: https://arxiv.org/pdf/0808.2902.pdf