Social Network Analysis

ERGM Internals

Robin Burke

DePaul University

Chicago, IL

Thanks to Carter Butts for some slides and illustrations

Admin

- Milestone due
  - Visualization critique
- Next week
  - 5-7 visualizations
  - 2 from Gephi, 2 from ggplot
  - Others can be any source

ERGM Review

θ = [p1, p2, p3 ... pk]
A vector of parameters
What we are trying to fit

t(g) = [t1(g), t2(g), t3(g), ... tk(g)]
t(g) = a vector of computed properties of g
What we are measuring about g

$$P(g \mid \theta, t, \gamma) = \frac{e^{\boxed{\theta^T t(g)}}}{\sum_{g' \in \gamma} e^{\theta^T t(g')}}$$

Dot product of two vectors = scalar value

g = the graph we observed

γ = the set of all possible graphs of interest

Conditional Log-odds

$$\log\left[\frac{P(M = m_{i,j}^{+} \mid \theta, t, \mu)}{P(M = m_{i,j}^{-} \mid \theta, t, \mu)}\right] = \theta^{T}\left[t(m_{i,j}^{+}) - t(m_{i,j}^{-})\right] = \theta^{T}\Delta_{ij}$$

$$\Delta_{ij} = t(m_{ij}^{+}) - t(m_{ij}^{-})$$

- Useful implication
  - each unit change in the measurement t_k when (*i*,*j*) edge is present
  - increases the conditional log-odds of (*i*,*j*) by θ_k

What's under the hood?

- NOT like regression
- Stochastic sampling process
  - approximating maximum likelihood
  - many parameters of the fitting process
- Fitting can fail

Log-likelihood

$$\ell(\theta) = \theta^T t(g) - \log\left[\sum_{g' \in \gamma} e^{\theta^T t(g')}\right]$$

- Create a function κ to represent the ugly sum

$$\ell(\theta) = \theta^T t(g) - \log \kappa(\theta, \gamma)$$

Use a ratio with arbitrary vector

$$\ell(\theta)-\ell(\theta_0)=(\theta-\theta_0)^T t(y)-\log\left[\frac{\kappa(\theta,\gamma)}{\kappa(\theta_0,\gamma)}\right]$$

- An improvement?

- But, if Y has distribution governed by θ_0

$$\log\left[\frac{\kappa(\theta,\gamma)}{\kappa(\theta_0,\gamma)}\right]=E\left(e^{(\theta-\theta_0)^T t(Y)}\mid\theta_0\right)$$

By law of large numbers

$$E\left(e^{(\theta-\theta_0)^T s(Y)} \mid \theta_0\right) = \frac{1}{m}\sum_{i=1}^{m} e^{(\theta-\theta_0)^T s(Y_i)}$$

- So, generate a lot of networks Y_i based on a random θ_0
- Use them to estimate the log-likelihood of a given θ
- Try to find θ of maximum likelihood

Problem

- The farther away θ_0 is from θ
  - the more samples are needed
  - exponentially many
- Need a θ_0 that is "pretty good"
  - otherwise it doesn't work at all

Pseudolikelihood estimation

- Local approximation of likelihood
  - Pretend edges are independent
- Use logistic regression to estimate θ_0
  - Talked about this last time
- If your model is bad
  - your initial starting point will be bad
  - you start too far from the destination

Next problem: simulations

- How to generate the networks?
- We want to be able to sample the set of networks from the distribution given by θ_0
  - but we don't know the distribution
- We also want to know the statistical properties of this distribution
  - to calculate standard errors

Sampling

- Technique to approximate the expected value of a distribution
  - when the distribution is ugly / hard to integrate
  - idea: turn an integral into a sum
- Also can be used to get the variance, etc.

Expected Values

- Want to know the expected value of a distribution.

$$\mathbb{E}[f] = \int f(x)p(x)\mathrm{dx}$$

- We can calculate p(**x**)
  - remember that the ERGM is a probabilistic model

$$\hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(x^{(l)})$$

- but integration is difficult

General method

- We have a representation of p(x) and f(x), but integration is intractable
  - E[f] is difficult as an integral, but easy as a sum.
- Randomly select points from distribution p(x) and use these as representative of the distribution of f(x).
- It turns out that if correctly sampled, only 10-20 points can be sufficient to estimate the mean and variance of a distribution.
  - Samples must be independently drawn
  - Expectation may be dominated by regions of high probability, or high function values

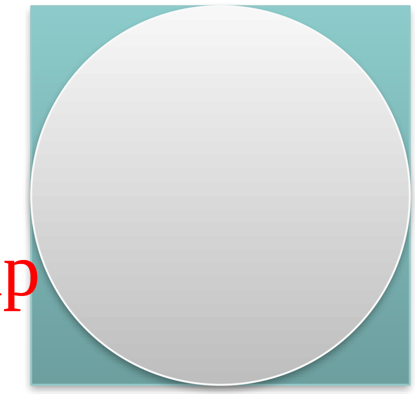
Monte Carlo Example

- Sampling techniques to solve difficult integration problems.

- What is the area of a circle with radius 1?

  - What if you don't know trigonometry?

Monte Carlo Estimation



- Take a random x and a random y between 1 and -1
  - Sample from x and sample from y.
- Determine if $x^2 + y^2 \leq 1$
- Repeat many times.
- Count the number of times that the inequality is true.
- Divide by the area of the square

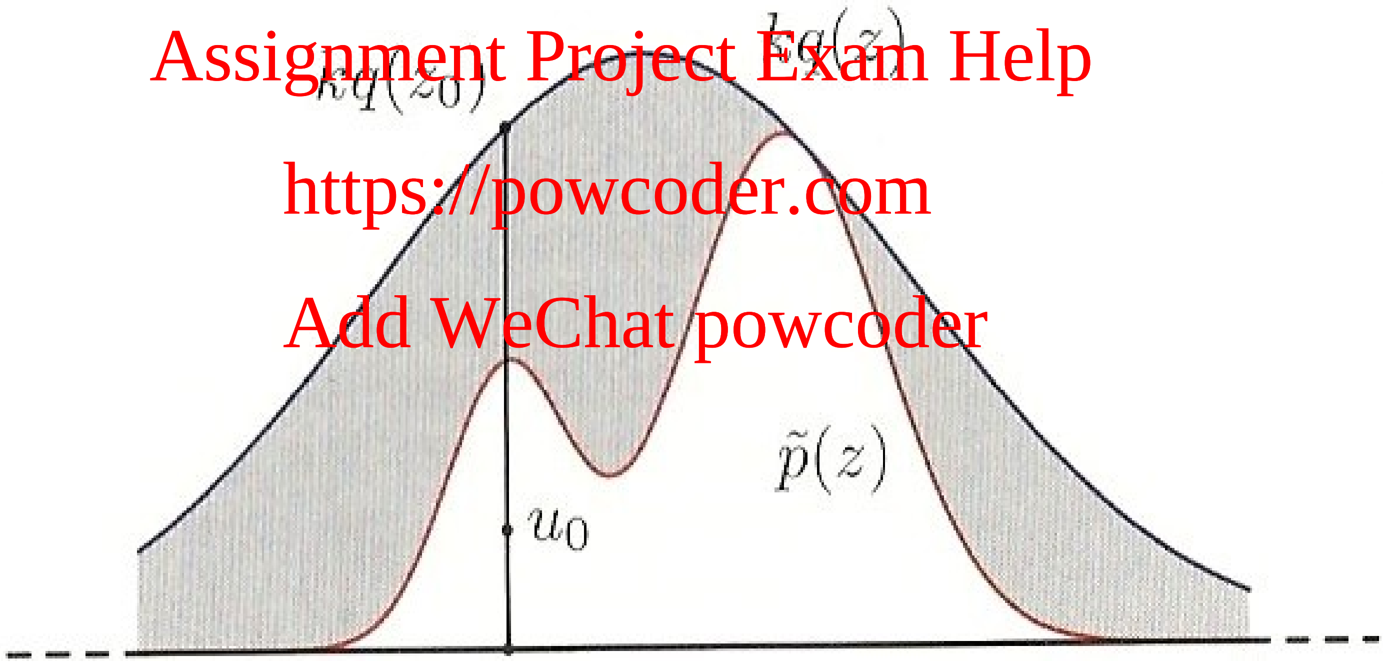
Rejection Sampling

- The distribution p(x) is easy to evaluate at point x
  - But difficult to integrate.
- Identify a simpler distribution, kq(x), which bounds p(x), and sample, x_0, from it.
  - This is called the **proposal distribution**.
- Generate another sample u from an even distribution between 0 and $kq(x_0)$.
  - If $u \leq p(x_0)$ **accept** the sample
    - E.g. use it in the calculation of an expectation of f
  - Otherwise **reject** the sample
    - E.g. omit from the calculation of an expectation of f

This is the square

This is the circle

Rejection Sampling Example



Importance Sampling

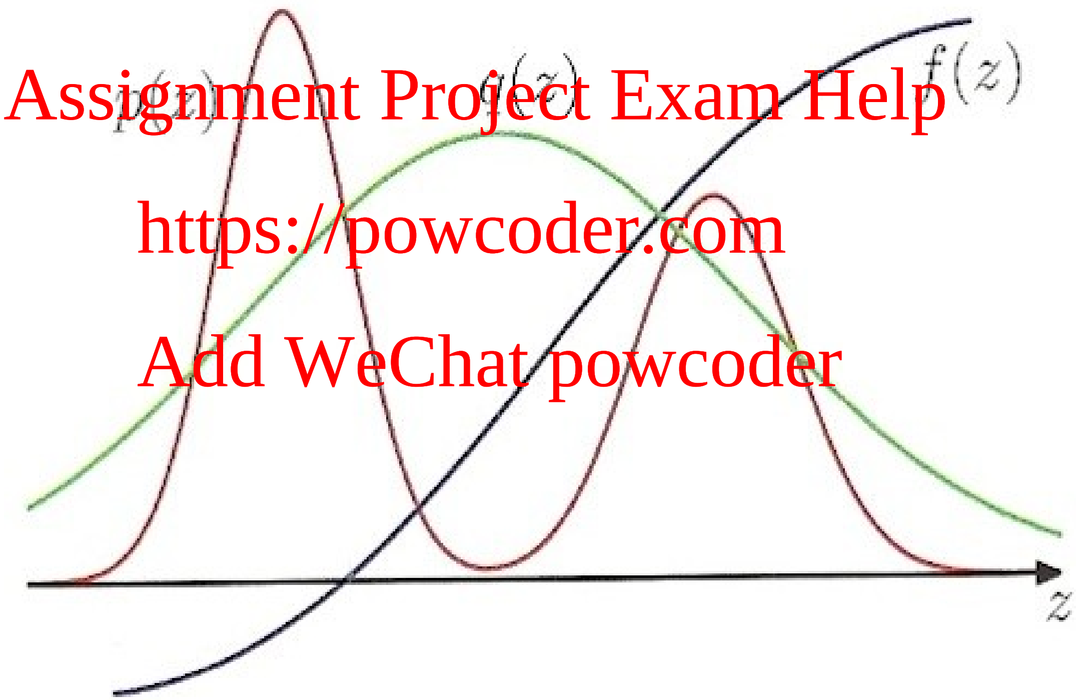
$$\mathbb{E}[f] = \int f(\vec{x}) p(\vec{x}) \mathrm{d}\vec{x}$$

$$= \int f(\vec{x}) \frac{p(\vec{x})}{q(\vec{x})} q(\vec{x}) \mathrm{d}\vec{x}$$

$$\approx \frac{1}{L} \sum_{l=1}^{L} \frac{p(\vec{x})}{q(\vec{x})} f(\vec{x}^{(l)})$$

- One problem with rejection sampling is that you lose information when throwing out samples.
- If we are **only** looking for the expected value of f(x), we can incorporate unlikely samples of x in the calculation.
- Again use a **proposal distribution** to approximate the expected value.
  - Weight each sample from q by the likelihood that it was also drawn from p.

Graphical Example of Importance Sampling



Markov Chain Monte Carlo

- Markov Chain:
  - $p(x_1|x_2,x_3,x_4,x_5,\ldots) = p(x_1|x_2)$
- For MCMC sampling start in a state $z^{(0)}$.
- At each step, draw a sample $z^{(m+1)}$ based on the previous state $z^{(m)}$
- **Accept** this step with some probability based on a **proposal distribution**.
  - If the step is accepted: state = $z^{(m+1)}$
  - Else: $z^{(m+1)} = z^{(m)}$
- Or only accept if the sample is consistent with an observed value

Markov Chain Monte Carlo

- Goal: $p(z^{(m)}) = p^*(z)$ as $m \to \infty$
  - MCMCs that have this property are **ergodic**.
  - Implies that the sampled distribution converges to the true distribution

- Need to define a transition function to move from one state to the next.
  - How do we draw a sample at state m+1 given state m?
  - Often, $z^{(m+1)}$ is drawn from a gaussian with $z^{(m)}$ mean and a constant variance.

$$T(z^{(m)}, z^{(m+1)}) = p(z^{(m+1)}|z^{(m)})$$

Metropolis-Hastings Algorithm

- Assume the current state is $z^{(m)}$.
- Draw a sample z^* from $q(z|z^{(m)})$
- Accept probability function

$$A(z^*, z^{(m)}) = \min\left(1, \frac{p(z^*)\, q(z^{(m)}|z^*)}{p(z^{(m)}), q(z^*|z^{(m)})}\right)$$

- Often use a normal distribution for q
  - Tradeoff between convergence and acceptance rate based on variance.

Application to simulation

- Start with a random network
- Modify the network by adding and removing an edge

- Use the MH criterion to accept or reject the network

- Markov chain converges (in the limit) to the set of networks defined by θ_0
- We sample this chain
  - to get networks that have high probability given θ_0

Estimation

- Being able to sample networks that match our distribution

- Means we can calculate the expected values and variance of the target metrics

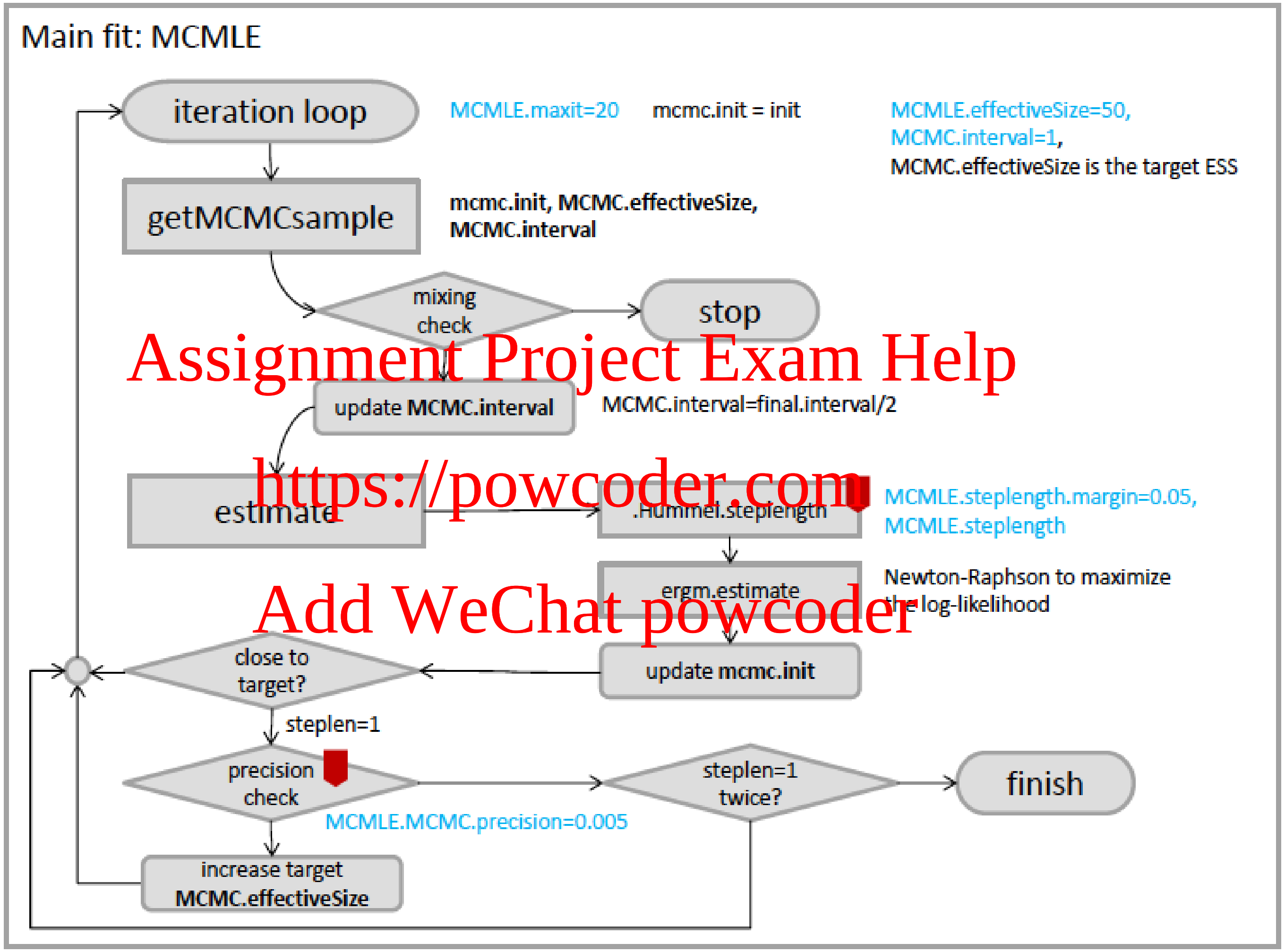
  - t(g) terms

Convergence

- We need enough steps in the chain
  - so that the distribution has "mixed"
  - not contaminated by choice of θ_0
- We need enough samples from the mixed chain
  - so that we get good statistical properties
  - "snapshot"

Summing up

- Little gnomes make an initial guess at θ_0 using the MPLE
  - parameters of the model
- More gnomes simulate $y_1,\ldots,y_n$ based on the initial guess
  - graphs that are likely given the parameters
- The simulated sample is used to find θ using MLE
- Possibly, the previous two steps are iterated a few times for good measure
  - since initial estimate may be incorrect


Main fit: MCMLE
iteration loop
MCMLE.maxit=20
mcmc.init = init
MCMLE.effectiveSize=50,
MCMC.interval=1,
MCMC.effectiveSize is the target ESS
getMCMCsample
mcmc.init, MCMC.effectiveSize,
MCMC.interval
mixing
check
stop
update MCMC.interval
MCMC.interval=final.interval/2
estimate
.Hummel.steplength
MCMLE.steplength.margin=0.05,
MCMLE.steplength
ergm.estimate
Newton-Raphson to maximize
the log-likelihood
update mcmc.init
close to
target?
steplen=1
precision
check
MCMLE.MCMC.precision=0.005
steplen=1
twice?
finish
increase target
MCMC.effectiveSize

Simulation can fail

- Insufficient burn-in
  - starting point still affects results
- Insufficient post-burn samples
  - sample hasn't converged
- May be degenerate
  - almost all graphs are same
  - usually complete/empty
- Sample does not cover observed graph
  - problematic for inference
  - bad θ_0 due to bad model