Inference in the Presence of Intractable Normalizing Functions

Murali Haran

(joint with Jaewoo Park)

Department of Biostatistics, University of California, Los Angeles

March 2020

Outline

Intractable Normalizing Functions

An Attraction-Repulsion Point Process Model

Computational Challenges

A Framework for Normalizing Function Algorithms

A Novel Emulation-Based Algorithm

Theory and Applications

Outline

Computing with Intractable Normalizing Functions

- Framework, comparisons for current algorithms
  - Park and Haran (2018) "Bayesian Inference in the Presence of Intractable Normalizing Functions," *Journal of the American Statistical Association*
- New algorithm
  - Park and Haran (2019) "A Function Emulation Approach for Doubly Intractable Distributions," *Journal of Computational and Graphical Statistics*

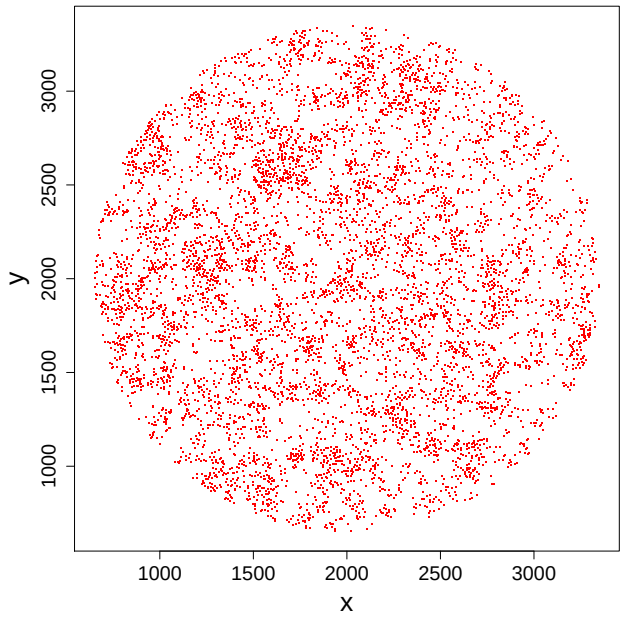
Outline

Models with Intractable Normalizing Functions

- Models with intractable normalizing functions
  - Data: $\mathbf{x} \in \chi$, parameter: $\theta \in \Theta$
  - Model: $h(\mathbf{x}|\theta)/Z(\theta)$, where $Z(\theta) = \int_{\chi} h(\mathbf{x}|\theta)d\mathbf{x}$ is intractable
- Popular examples
  - Social network models: exponential random graph models (Robins et al., 2002; Hunter et al., 2008)
  - Models for lattice data (Besag, 1972, 1974)
  - Spatial point process models: interaction models (Strauss, 1975, Goldstein, Haran et al., 2015)
- Challenge: likelihood-based inference with $Z(\theta)$

Interaction Point Process

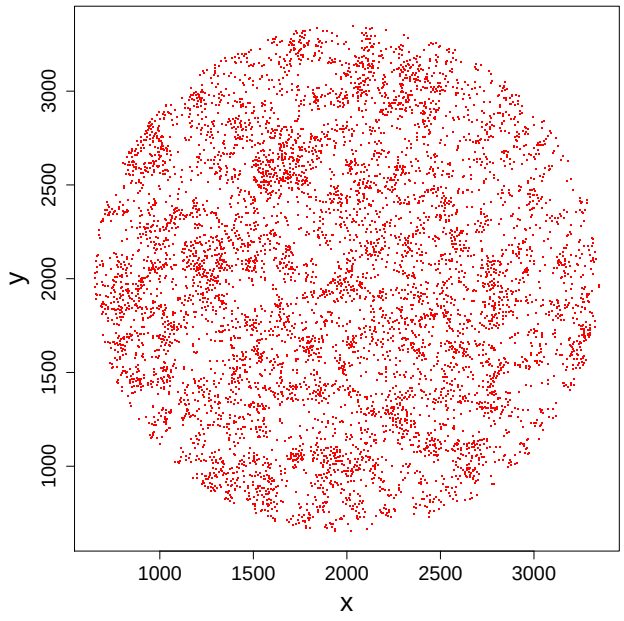
- Biologist's interest: study progression of viral infections
- Our goal: use data from imaging of cell cultures to study the spatial structure of an infection
- An *in vitro* cell culture study identifies and locates cells infected with two strains of the human respiratory syncytial virus (RSV-A and RSV-B)



Cells infected with RSV

Interaction Point Process

- Biologist's interest: study progression of viral infections
- Our goal: use data from imaging of cell cultures to study the spatial structure of an infection
- An *in vitro* cell culture study identifies and locates cells infected with two strains of the human respiratory syncytial virus (RSV-A and RSV-B)



Cells infected with RSV

Question: How does the presence of an infected cell impact infections in neighboring cells?

Attraction-repulsion Model

- Previous models (e.g. Strauss process) do not allow for repulsion *and* attraction
- Our point process model (Goldstein, Haran, et al., 2015) allows for both
- Allows us to easily compare interaction behavior for different strains of RSV
- Our model (like Strauss and other popular models) has an intractable normalizing function
  - Motivation for (i) studying existing algorithms for this problem, and (ii) developing new algorithms

Outline

Maximum Likelihood (ML) Inference

$$\hat{\theta} = \arg\max_{\theta \in \Theta} h(\mathbf{x}|\theta)/Z(\theta)$$

- Pseudolikelihood approximation (Besag, 1975)
  - Often a poor approximation
  - Awkward in a hierarchical model (not compatible with a real probability model)
- Markov chain Monte Carlo Maximum Likelihood (Geyer and Thompson, 1994)
  - Sensitive to choice of importance function
  - Optimization can be unstable
  - For some models, obtaining standard errors is challenging
    E.g. Attraction-repulsion point process

Bayesian Inference

- Can address some of the challenges just discussed
- Bayesian inference
  - Prior : $p(\theta)$
  - Posterior: $\pi(\theta|\mathbf{x}) \propto p(\theta)h(\mathbf{x}|\theta)/Z(\theta)$
- Inference is based on $\pi(\theta|\mathbf{x})$
- Generally approximated via Markov chain Monte Carlo (MCMC)
- MCMC is challenging due to $Z(\theta)$

Markov chain Monte Carlo Basics

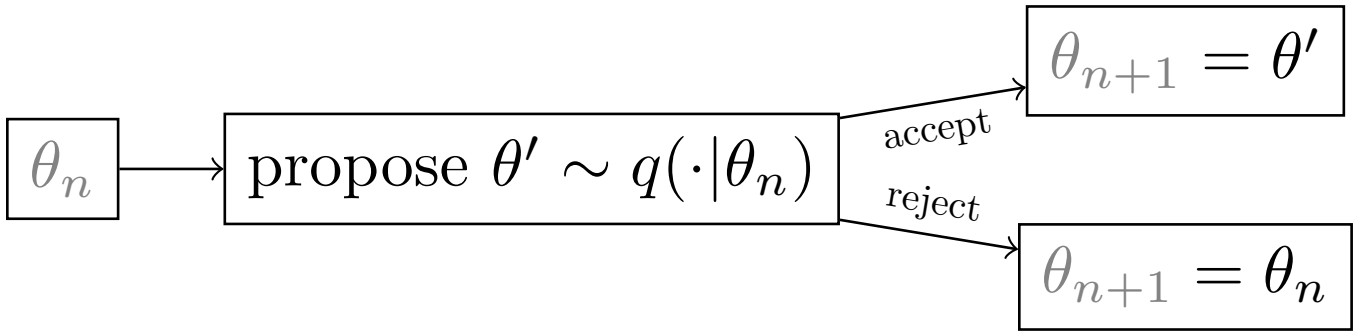
- Construct Harris-ergodic Markov chain $\theta_1, \theta_2, \ldots$ with stationary distribution $\pi(\theta \mid \mathbf{x})$
- Treat $\theta_1, \theta_2, \theta_3, \ldots.$ as samples from $\pi(\theta|\mathbf{x})$
- For any real-valued $g(\cdot)$, approximate $E_\pi(g(\theta))$ by

$$\hat{\mu}_n = \frac{\sum_{i=1}^{n} g(\theta_i)}{n}$$

- Under general conditions, $\hat{\mu}_n \to \mu$ as $n \to \infty$

The Metropolis-Hastings Algorithm

Recipe for constructing Markov chain: given θ_n, obtain θ_{n+1}



Accept-reject ratio

$$\frac{\pi(\theta'|\mathbf{x})q(\theta_n|\theta')}{\pi(\theta_n|\mathbf{x})q(\theta'|\theta_n)} = \frac{p(\theta')Z(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)Z(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

Cannot evaluate because of $Z(\cdot)$

Outline

Previous Work

Two classes of algorithms (with some overlap)

1. Auxiliary variable algorithms (cf. Moller et al., 2006; Murray et al., 2007; Liang et al., 2010, 2016)
2. Likelihood approximation algorithms (Atchade et al., 2015; Andrieu and Roberts, 2009; Lyne et al., 2015; Alqueir et al., 2016)

I. Auxiliary Variable Algorithms

(Moller et al., 2016; Murray et al., 2017)

The acceptance ratio

$$\frac{\pi(\theta'|\mathbf{x})q(\theta_n|\theta')}{\pi(\theta_n|\mathbf{x})q(\theta'|\theta_n)} = \frac{p(\theta')Z(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)Z(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

- Generate an auxiliary random variate from model $h(\mathbf{x}|\theta')$
  - $X' \sim h(\mathbf{x}|\theta')$
  - M-H algorithm: accept/reject both X' and θ'
- New acceptance ratio: $Z(\theta)$ gets canceled

I. Auxiliary Variable Algorithms

(Moller et al., 2016; Murray et al., 2017)

The acceptance ratio

$$\frac{\pi(\theta'|\mathbf{x})q(\theta_n|\theta')}{\pi(\theta_n|\mathbf{x})q(\theta'|\theta_n)} = \frac{p(\theta')Z(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)Z(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

- Generate an auxiliary random variate from model $h(\mathbf{x}|\theta')$
  - $X' \sim h(\mathbf{x}|\theta')$
  - M-H algorithm: accept/reject both X' and θ'
- New acceptance ratio: $Z(\theta)$ gets canceled

Problem: $X \sim h(\mathbf{x}|\theta')$ **is difficult/expensive**

More practical and general: keep last draw from MCMC with stationary distribution $h(\mathbf{x}\theta)$ (double Metropolis, Liang, 2010)

This is *asymptotically inexact* and can still be slow

Double Metropolis-Hastings (DMH)

- Basic idea: DMH replaces exact sampling of $\mathbf{y}$ with MCMC sampling.
- Algorithm
  1. $\theta' \sim q(\cdot|\theta_n)$
  2. $\mathbf{y} \sim h(\cdot|\theta')/Z(\theta')$ via m-number of MCMC updates.
     - $\mathbf{y}_{new} \sim T(\mathbf{y}_{new}|\mathbf{y}_{old})$, where $T(\mathbf{y}_{new}|\mathbf{y}_{old})$ is proposal distribution.
     - Accept $\mathbf{y}_{new}$ with probability $\frac{h(\mathbf{y}_{new}|\theta')T(\mathbf{y}_{old}|\mathbf{y}_{new})}{h(\mathbf{y}_{old}|\theta')T(\mathbf{y}_{new}|\mathbf{y}_{old})}$.
     - Repeat this procedure m-times, and regard the last state of the resulting Markov chain as sample from $h(\mathbf{y}|\theta')/Z(\theta')$.
  3. Accept $\theta_{n+1} = \theta'$ with probability

$$\alpha = \frac{p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}|\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}|\theta')q(\theta'|\theta_n)}$$

The Adaptive Exchange Algorithm (AEX)

- Basic idea: AEX replaces exact sampling of $\mathbf{y}$ with re-sampling method.
- Algorithm (at $n+1$st iteration)

  1. **Auxiliary chain**:

  $\mathbf{x}_{n+1} \sim \{h(\mathbf{x}|\theta^{(1)})/Z(\theta^{(1)}), ..., h(x|\theta^{(d)})/Z(\theta^{(d)})\}$ via stochastic approximation Monte Carlo and keep sampled $\mathbf{x}_{n+1}$.

  2. **Target chain**:

  - $\theta' \sim q(\theta'|\theta_n)$
  - $\mathbf{y} \sim \{\mathbf{x}_1, ..., \mathbf{x}_{n+1}\}$ w.r.t. $p(\mathbf{x}_i|\theta')$ for $i = 1, ..., n+1$.
  - Accept $\theta_{n+1} = \theta'$ with probability

$$\alpha = \frac{p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}|\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}|\theta')q(\theta'|\theta_n)}$$

2. Likelihood Approximation

The acceptance ratio

$$\frac{\pi(\theta'|\mathbf{x})q(\theta_n|\theta')}{\pi(\theta_n|\mathbf{x})q(\theta'|\theta_n)} = \frac{p(\theta')Z(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)Z(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

Likelihood approximation methods

1. Approximate $Z(\theta)$ using importance sampling
   - Requires its own MCMC algorithm
2. Use approximation $\widehat{Z}(\theta)$ in acceptance ratio

$$\frac{p(\theta')\widehat{Z}(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)\widehat{Z}(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

2. Likelihood Approximation

The acceptance ratio

$$\frac{\pi(\theta'|\mathbf{x})q(\theta_n|\theta')}{\pi(\theta_n|\mathbf{x})q(\theta'|\theta_n)} = \frac{p(\theta')Z(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)Z(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

Likelihood approximation methods

1. Approximate $Z(\theta)$ using importance sampling
   - Requires its own MCMC algorithm
2. Use approximation $\widehat{Z}(\theta)$ in acceptance ratio

$$\frac{p(\theta')\widehat{Z}(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)\widehat{Z}(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

Problem: Step 1 is computationally expensive

Outline

Intractable Normalizing Functions

An Attraction-Repulsion Point Process Model

Computational Challenges

A Framework for Normalizing Function Algorithms

A Novel Emulation-Based Algorithm

Theory and Applications

Emulation-Based Algorithm

Park and Haran (2019)

- Likelihood approximation approach with a two-step approximation
  1. Approximate $Z(\theta)$ using importance sampling on a set of θs
  2. Use Gaussian process "emulation" approach to interpolate this function at any new value
  3. Construct MCMC algorithm using this interpolation

Theory to justify this as number of design points and number of importance sampling draws increases

(Park and Haran, 2019)

Normalizing Function Emulation Algorithm

Part 1: Construct two-stage approximation

- Pre-MCMC

1. For each $\theta \in \{\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(d)}\}$, obtain importance sampling approximation $\widehat{Z}_{IMP}(\theta)$
2. Fit Gaussian process (GP) to $\{\widehat{Z}_{IMP}(\boldsymbol{\theta}^{(1)}), ..., \widehat{Z}_{IMP}(\boldsymbol{\theta}^{(d)})\}$
   Now for each θ obtain GP approximation, $\widehat{Z}_{GP}(\theta)$

Normalizing Function Emulation Algorithm

Part 1: Construct two-stage approximation

- Pre-MCMC

1 For each $\theta \in \{\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(d)}\}$, obtain importance sampling approximation $\widehat{Z}_{IMP}(\theta)$

2 Fit Gaussian process (GP) to $\{\widehat{Z}_{IMP}(\boldsymbol{\theta}^{(1)}), ..., \widehat{Z}_{IMP}(\boldsymbol{\theta}^{(d)})\}$
Now for each θ obtain GP approximation, $\widehat{Z}_{GP}(\theta)$

Part 2: MCMC algorithm with GP approximation

- Given $\theta_n \in \boldsymbol{\Theta}$ at nth iteration.

3 Propose $\boldsymbol{\theta}' \sim q(\cdot|\boldsymbol{\theta}_n)$

4 Obtain $\widehat{Z}_{GP}(\boldsymbol{\theta}')$, accept $\boldsymbol{\theta}'$ with

$$\alpha = \min\left\{\frac{p(\boldsymbol{\theta}')h(\mathbf{x}|\boldsymbol{\theta}')\widehat{Z}_{GP}(\boldsymbol{\theta})q(\boldsymbol{\theta}|\boldsymbol{\theta}')}{p(\boldsymbol{\theta})h(\mathbf{x}|\boldsymbol{\theta})\widehat{Z}_{GP}(\boldsymbol{\theta}')q(\boldsymbol{\theta}'|\boldsymbol{\theta})}, 1\right\}$$

Computational Benefits

- Can compute in parallel; much of this is done “offline”, before running the algorithm

Computational Benefits

- Can compute in parallel; much of this is done “offline”, before running the algorithm
- Two versions of our approach
  - (i) NormEm emulate $Z(\theta)$ with $\widehat{Z}_{GP}(\theta)$

Computational Benefits

- Can compute in parallel; much of this is done "offline", before running the algorithm
- Two versions of our approach
  - (i) NormEm emulate $Z(\theta)$ with $\widehat{Z}_{GP}(\theta)$
  - (ii) LikEm emulate $\mathcal{L}(\theta) = h(\mathbf{x}|\theta)/Z(\theta)$ with $\widehat{\mathcal{L}}_{GP}(\theta)$

Outline

Intractable Normalizing Functions

An Attraction-Repulsion Point Process Model

Computational Challenges

A Framework for Normalizing Function Algorithms

A Novel Emulation-Based Algorithm

Theory and Applications

Theory

The Markov chain constructed by the function-emulation algorithm, with $n-$step transition kernel $P^n_{GP}(x, \cdot)$, converges in total variational distance to the target distribution π

$$\lim_{n\to\infty} \| P^n_{GP}(x, \cdot) - \pi(\cdot) \|_{TV} = 0, \forall x \in \Omega$$

- Key assumptions satisfied for all our examples
- Results as # samples for $\widehat{Z}_{IMP}$ and number of design points for $\widehat{Z}_{GP}$ both go to infinity. Hence, in practice asymptotically inexact (like Double Metropolis-Hastings)

Park and Haran (2019)

Also see Mitrophanov (2005); Alquier et al. (2016)

Example: Interaction Point Process

(1) Interaction point process model (Goldstein et al., 2015)

- Real data set, n =3,000 points
- Comparing fastest existing algorithm DMH (Double Metropolis-Hastings) with our two new algorithms
- HPD=highest posterior density region

θ_1	Mean	95%HPD	Time(hour)
DMH	1.34	(1.30,1.39)	18.99
NormEm	1.34	(1.30,1.39)	3.60
LikEm	1.34	(1.29, 1.39)	2.53

Two More Examples

(2) Exponential random graph model (ERGM): 2,000 dimensional ERGM for a network (Hunter et al, 2006)

- Reliable results from NormEm, LikEm within 2 hours
- All other algorithms are computationally infeasible

(3) Susceptible-infected-recovered (SIR) model for rotavirus Park, Haran et al., 2018

- Likelihood evaluation: involve solving SIR dynamics at each iteration
- LikEm emulates entire likelihood
- 10 times faster than regular approach, similar results

The Last Slide

- This methodology is widely applicable, can carry out full inference for some problems when previously infeasible
- LikeEm algorithm is useful for intractable likelihood problems (not just intractable normalizing function problems): e.g. applied to disease model
- Open problems and caveats
  - Relies heavily on design points selected initially
    - Slow: use Double Metropolis-Hastings or Approximate Bayesian Computing (ABC)
  - Practical for low-dimensional (< 7) parameter space only
  - Automated tuning and stopping rules?
  - **How to evaluate quality of approximation? Compare approximate algorithms?**

References

All papers on `arxiv.org`

1. Framework, comparisons for current algorithms
   - Park and Haran (2018) "Bayesian Inference in the Presence of Intractable Normalizing Functions," *Journal of the American Statistical Association*
2. New algorithm
   - Park and Haran (2019) "A Function Emulation Approach for Doubly Intractable Distributions," *Journal of Computational and Graphical Statistics*

References

- Atchade, Y., Lartillot, N. and Robert, C. (2013) Bayesian computation for statistical models with intractable normalizing constants. *Brazilian Journal of Probability and Statistics*
- Liang, F., Jin, I. H., Song, Q., and Liu, J. S. (2015) An adaptive exchange algorithm for sampling from distributions with intractable normalising constants. *Journal of the American Statistical Association*
- Goldstein, J., Haran, M., Simeonov, I., Fricks, J., and Chiaromonte, F. (2015) An attraction-repulsion point process model for respiratory syncytial virus infections. *Biometrics*

Exponential Random Graph Models

2-star	Mean	95%HPD	ESS	Time(second)
DMH	1.224	(-0.102,2.587)	986.796	6.620
AEX	1.245	(0.169,2.579)	991.874	126.460
ALR	1.247	(0.160,2.518)	1456.573	2500.370
Gold standard	1.265	(0.084,2.498)	9655.902	

Table: 30,000 MCMC samples are used for all algorithms.

- An exponential random graph model: $\theta \in R^4$

  $h(\theta|\mathbf{x})/Z(\theta) = exp\{\theta' S(\mathbf{x})\}/Z(\theta)$
- Business networks among 16 Florentine families (Padgett, 1994)
  - Data $\mathbf{x} \in R^{16\times 16}$ representing connections among 16 families (0-1).
  - For AEX and ALR, we can only store 4-dimensional sufficient statistics for a sampled $\mathbf{x}_{n+1}$ with each iteration.

Exchange Algorithm

- Joint distribution: $\pi(\theta_n, \theta'_n, \mathbf{y}_n|\mathbf{x})$

  $= \pi(\theta_n|\mathbf{x})\pi(\theta'_n|\theta_n)\pi(\mathbf{y}_n|\theta'_n) \propto p(\theta_n)\frac{h(\mathbf{x}|\theta_n)}{Z(\theta_n)}q(\theta'_n|\theta_n)\frac{h(\mathbf{y}_n|\theta'_n)}{Z(\theta'_n)}$

- Algorithm

  1. Update $[\theta'_n, \mathbf{y}_n]$:
     - Propose $\theta'_{n+1} \sim q(\cdot|\theta_n)$
     - Propose $\mathbf{y}_{n+1} \sim h(\cdot|\theta'_{n+1})/Z(\theta'_{n+1})$ independently
  2. Update $[\theta_n]$ through swapping proposal:
     - $S : (\mathbf{x}, \theta_n), (\mathbf{y}_{n+1}, \theta'_{n+1}) \Rightarrow (\mathbf{x}, \theta'_{n+1}), (\mathbf{y}_{n+1}, \theta_n)$
     - Accept $\theta_{n+1} = \theta'_{n+1}$ with probability

$$\alpha = \frac{\cancel{S(\theta_n|\theta'_{n+1})}\,\pi(\theta'_{n+1}, \theta_n, \mathbf{y_{n+1}}|\mathbf{x})}{\cancel{S(\theta'_{n+1}|\theta_n)}\,\pi(\theta_n, \theta'_{n+1}, \mathbf{y}_{n+1}|\mathbf{x})}$$

$$\text{xc}= \frac{p(\theta'_{n+1})h(\mathbf{x}|\theta'_{n+1})\cancel{Z(\theta_n)}h(\mathbf{y}|\theta_n)\cancel{Z(\theta'_{n+1})}q(\theta_n|\theta'_{n+1})}{p(\theta_n)h(\mathbf{x}|\theta_n)\cancel{Z(\theta'_{n+1})}h(\mathbf{y}|\theta'_{n+1})\cancel{Z(\theta_n)}q(\theta'_{n+1}|\theta_n)}$$

Attraction-repulsion Model

Previous models did not allow for repulsion *and* attraction

New point process model (Goldstein, Haran, et al., 2015):

The likelihood can be written as

$$\mathcal{L}(X|\Theta) = \frac{f(X|\Theta)}{Z(\Theta)}, f(X|\Theta) = \lambda^n \Big[\prod_{i=1}^{n} e^{\min\left[\sum_{i \neq j} log(\phi(x_i, x_j)), k\right]} \Big]$$

Model parameters:

- λ is the intensity of the process
- $\theta_1, \theta_2, \theta_3$ control the shape of $\phi(r)$.
- R is the minimum distance allowed between points
- k is a truncation constant necessary to prevent “clumping" behavior

Important: $Z(\Theta)$ is intractable

Appendix: Double Metropolis-Hastings (DMH)

- Basic idea: DMH replaces exact sampling of $\mathbf{y}$ with MCMC sampling.
- Algorithm
  1. $\theta' \sim q(\cdot|\theta_n)$
  2. $\mathbf{y} \sim h(\cdot|\theta')/Z(\theta')$ via m-number of MCMC updates.
     - $\mathbf{y}_{new} \sim T(\mathbf{y}_{new}|\mathbf{y}_{old})$, where $T(\mathbf{y}_{new}|\mathbf{y}_{old})$ is proposal distribution.
     - Accept $\mathbf{y}_{new}$ with probability $\frac{h(\mathbf{y}_{new}|\theta')T(\mathbf{y}_{old}|\mathbf{y}_{new})}{h(\mathbf{y}_{old}|\theta')T(\mathbf{y}_{new}|\mathbf{y}_{old})}$.
     - Repeat this procedure m-times, and regard the last state of the resulting Markov chain as sample from $h(\mathbf{y}|\theta')/Z(\theta')$.
  3. Accept $\theta_{n+1} = \theta'$ with probability

$$\alpha = \frac{p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}|\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}|\theta')q(\theta'|\theta_n)}$$

Appendix: The Adaptive Exchange Algorithm (AEX)

- Basic idea: AEX replaces exact sampling of $\mathbf{y}$ with re-sampling method.
- Algorithm (at $n+1$st iteration)

  1. **Auxiliary chain**:

  $\mathbf{x}_{n+1} \sim \{h(\mathbf{x}|\theta^{(1)})/Z(\theta^{(1)}), ..., h(x|\theta^{(d)})/Z(\theta^{(d)})\}$ via stochastic approximation Monte Carlo and keep sampled $\mathbf{x}_{n+1}$.

  2. **Target chain**:

  - $\theta' \sim q(\theta'|\theta_n)$
  - $\mathbf{y} \sim \{\mathbf{x}_1, ..., \mathbf{x}_{n+1}\}$ w.r.t. $p(\mathbf{x}_i|\theta')$ for $i = 1, ..., n+1$.
  - Accept $\theta_{n+1} = \theta'$ with probability

$$\alpha = \frac{p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}|\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}|\theta')q(\theta'|\theta_n)}$$

Appendix: ALR Algorithm

- Basic idea: approximate $Z(\theta)$ adaptively through weighted importance sampling.
- $\widehat{Z}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{h(\mathbf{x}_i|\theta)}{h(\mathbf{x}_i|\theta^{(1)})}$, where $\mathbf{x}_i \sim h(\mathbf{x}|\theta^{(1)})/Z(\theta^{(1)})$ might be poor if $\theta^{(1)}$ is far from θ.
- Algorithm

  1. $\mathbf{x}_{n+1} \sim \{h(\mathbf{x}|\theta^{(1)})/Z(\theta^{(1)}), ..., h(\mathbf{x}|\theta^{(d)})/Z(\theta^{(d)})\}$ via stochastic approximation and keep sampled $\mathbf{x}_{n+1}$.
  2. $\theta' \sim q(\theta'|\theta_n)$.
  3. $\widehat{Z}(\theta) = \sum_{j=1}^{d} w_j \frac{1}{N_j}\sum_{i=1}^{N_j} \frac{h(\mathbf{x}_i|\theta)}{h(\mathbf{x}_i|\theta^{(j)})}$ using $\{\mathbf{x}_1, ..., \mathbf{x}_{n+1}\}$, where $\sum_{j=1}^{d} N_j = n+1$ and $\sum_{j=1}^{d} w_j = 1$.
  4. Accept $\theta_{n+1} = \theta'$ with probability

$$\alpha = \frac{p(\theta')\widehat{Z}(\theta_n)h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)\widehat{Z}(\theta')h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)}$$

Appendix: Computational Complexity and Memory

DMH	Exponential family	Point process
Complexity	$\mathbf{G(n)} + \mathbf{L(n)}$	$\mathbf{G(n^2)} + 3\mathbf{L(n^2)}$
$*$AEX	Exponential family	Point process
Complexity	$(1-\beta)[\mathbf{G(n)} + \mathbf{L(n)}]$	$(1-\beta)\mathbf{G(n^2)} + [N_1/b + m + 5]\mathbf{L}$
Memory	$\mathbf{p} + 2$	$\mathbf{n^2} + 3$
$*$ALR	Exponential family	Point process
Complexity	$\mathbf{G(n)} + \mathbf{L(n)}$	$\mathbf{G(n^2)} + [(d + 2m + 2)/c + 1]\mathbf{L}$
Memory	$\mathbf{p} + 1$	$\mathbf{n^2} + d + 1$

- **n**: size of data, **p**: dimension of θ,

  G(n): Complexity of sampling $\mathbf{y}$ (DMH), $\mathbf{x}_{n+1}$ (AEX, ALR),

  L(n): complexity of $h(\mathbf{x}|\theta)$
- In addition, there are fixed costs for AEX and ALR (preliminary step)