(Last adjustments: December 6, 2017)

Workshop on statistical challenges in astronomy –Hierarchical models in Stan

Presenter

Dr. John T. Ormerod

School of Mathematics & Statistics F07

University of Sydney

(w) 02 9351 5883

(e) john.ormerod (at) sydney.edu.au

Why MCMC?

- Do you have data? ($\mathbf{x}$)
- Do you want to build a rich statistical model? ($p(\mathbf{x}|\boldsymbol{\theta})$)
- Perhaps you want to incorporate prior information? ($p(\boldsymbol{\theta})$)
- Are the integrals to get posterior densities are intractable?

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{x})} = \frac{p(\mathbf{x},\boldsymbol{\theta})}{\int p(\mathbf{x},\boldsymbol{\theta})d\boldsymbol{\theta}}$$

- Are point estimates not adequate? (If not Variational Bayes might be for you).

Review: MCMC

- Markov Chain Monte Carlo. The samples form a Markov Chain.
- Markov property:

$$p(\theta_{t+1}|\theta_t, \ldots, \theta_1) = p(\theta_{t+1}|\theta_t)$$

- Invariant distribution:

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$$

- Detailed balance: sufficient condition:

$$\mathbb{P}(\theta_{t+1}, \mathcal{A}) = \int_{\mathcal{A}} q(\theta_{t+1}, \theta_t) dy$$

$$\pi(\theta_{t+1}) q(\theta_{t+1}, \theta_t) = \pi(\theta_t) q(\theta_t, \theta_{t+1})$$

- A Markov chain satisfying the detailed balance will converge in distribution to $\pi(\theta)$.
- We want to design Markov chains to mimic posterior distributions.

Review: Random Walk Metropolis Hastings

- Want samples from posterior distribution: $p(\boldsymbol{\theta}|\mathbf{x}) \propto p(\mathbf{x}, \boldsymbol{\theta})$.
- Algorithm: (Suppose $\boldsymbol{\theta} \in \mathbb{R}^d$)
  - Suppose we have $\mathbf{x}$, $p(\mathbf{x}, \boldsymbol{\theta})$ and $\boldsymbol{\theta}_0$. Choose $\mathbf{B} \in \mathbb{R}^{d \times d}$
  - Loop $t = 1, \ldots, T$.
    - Sample $\boldsymbol{\theta}_{\text{prop}} \sim N(\boldsymbol{\theta}_{t-1}, \mathbf{B})$
    - With probability

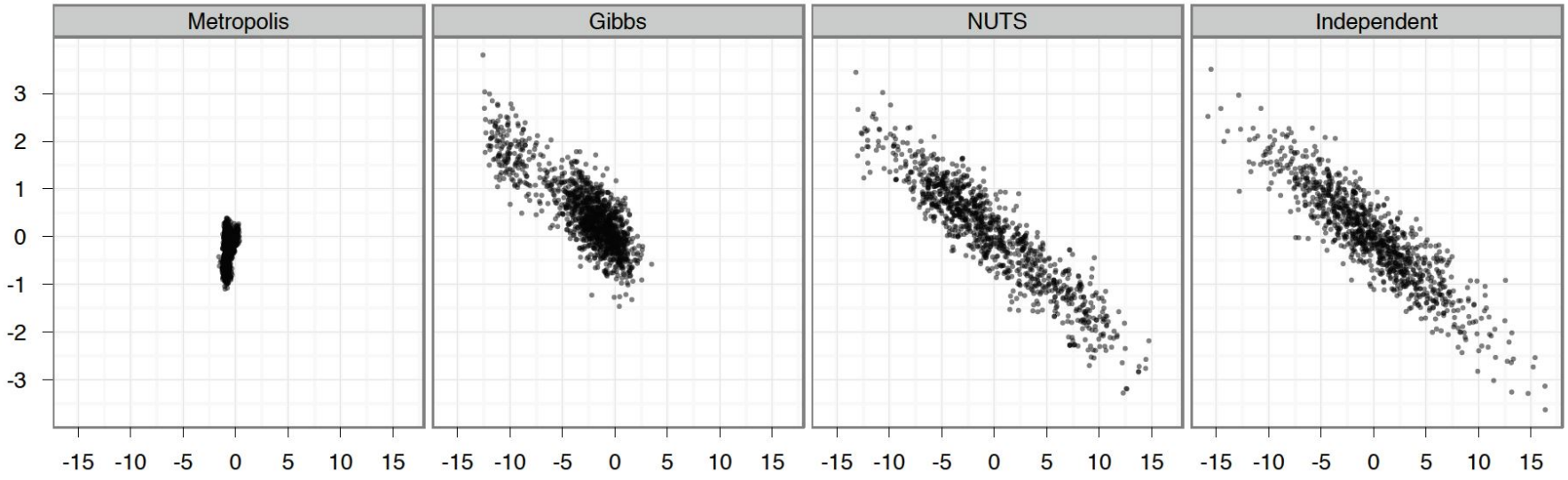
$$\alpha = \min \left[1, \frac{p(\mathbf{x}, \boldsymbol{\theta}_{\text{prop}})}{p(\mathbf{x}, \boldsymbol{\theta}_{t-1})} \right]$$

      Set $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{\text{prop}}$ otherwise set $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$.
- The above Markov chain can be shown (under mild conditions) to satisfy the detailed balance condition and converges in distribution to $p(\boldsymbol{\theta}|\mathbf{x})$.
- In practice we can treat samples $\{\boldsymbol{\theta}_t\}_{t=1}^T$ as independent samples from $p(\boldsymbol{\theta}|\mathbf{x})$ for sufficiently large T.

Stan: Hamiltonian Monte Carlo

- In the previous example a multivariate normal distribution was used to generate proposal samples.
- Stan uses Hamiltonian dynamics (Hamiltonian Monte Carlo) to generate good proposal samples and then uses a accept-reject step to ensure that the Markov chain mimics the posterior distribution.
- See HMC notes for details.
- HMC samples are typically much higher quality and require less tuning than other samplers.

Stan: Hamiltonian Monte Carlo



Stan: Where to get help

In anticipation that we will not get through everything today I will begin with a list of resources on Stan

- Homepage `http://mc-stan.org/`
- User Guide
  `http://mc-stan.org/users/`
- Documentation, tutorials and case studies
  `http://mc-stan.org/users/documentation/index.html`
- Nice book
  "Bayesian Models for Astrophysical Data Using R, JAGS, Python, and Stan" by Joseph M. Hilbe, Rafael S. de Souza & Emille E. O. Ishida
  `https://www.bayesianmodelsforastrophysicaldata.com/`
- Code from above book
  `https://github.com/astrobayes/`

Motivation for Stan

- Fit rich Bayesian statistical models.
- The process
  1. Create a statistical model.
  2. Perform inference on the model.
  3. Evaluate.
- Difficulty with models of interest in existing tools.

Motivation (cont.)

- Usability
  - general purpose, clear modelling language, integration
- Scalability
  - model complexity, number of parameters, data size
- Efficiency
  - high effective sample sizes, fast iterations, low memory
- Robustness
  - model structure (i.e. posterior geometry), numerical routines

What is Stan?

- Statistical model specification language
  - high level, probabilistic programming language
  - user specifies statistical model
  - easy to create statistical models
- 4 cross-platform users interfaces
  - CmdStan - command line
  - RStan - R integration
  - PyStan - Python integration
  - MStan - Matlab integration (user contributed)

Inference

- Hamiltonian Monte Carlo (HMC)
  - Sample parameters on unconstrained space (transform & Jacobian adjustment)
  - Gradients of the model wrt parameters (automatic differentiation).
  - Sensitive to tuning parameters (No-U-Turn sampler).
- No-U-Turn Sampler (NUTS)
  - warmup: estimates mass matrix and step size
  - sampling: adapts number of steps
  - maintains detailed balance
- Optimization
  - BFGS, Newtons method
- Variational inference.

Stan to Scientists

- Flexible probabilistic language, language still growing
- Focus on science: the modelling and assumptions
  - access to multiple algorithms (default is pretty good)
  - faster and less error prone than implementing from scratch
  - efficient implementation
- Lots of (free) modelling help on users list
- Responsive developers, continued support for Stan
- Not just for inference
  - fast forward sampling; lots of distributions
  - gradients for arbitrary functions

The Stan Language

Data Types

- basic:

  `real, int, vector, row_vector, matrix`

- constrained:

  `simplex, unit_vector, ordered, positive_ordered, corr_matrix, cov_matrix`

- arrays

The Stan Language

Bounded variables

- applies to int, real, and matrix types
- lower example:

```
real<lower=0> sigma;
```

- upper example:

```
real<upper=100> x;
```

The Stan Language

Program Blocks

- data (optional)
- transformed data (optional)
- parameters (optional)
- transformed parameters (optional)
- model
- generated quantities (optional)

Stan Example: basic structure – Linear regression

```
data {
    int<lower=0> N;
    vector[N] y;
    vector[N] x;
}
parameters {
    real alpha;
    real beta;
    real<lower=0> sigma;
}
model {
    alpha ~ normal(0,10);
    beta ~ normal(0,10);
    sigma ~ cauchy(0,5);
    for (n in 1:N)
        y[n] ~ normal(alpha + beta * x[n], sigma);
}
```

Stan Example: vectorisation – Linear regression

```
data {
    int<lower=0> N;
    vector[N] y;
    vector[N] x;
}
parameters {
    real alpha;
    real beta;
    real<lower=0> sigma;
}
model {
    alpha ~ normal(0,10);
    beta ~ normal(0,10);
    sigma ~ cauchy(0,5);
    y ~ normal(alpha + beta*x, sigma);
}
```

Stan Example: Eight Schools: hierarchical example

- Educational Testing Service study to analyze effect of coaching
- SAT-V in eight high schools
- No prior reason to believe any program was:
  - more effective than the others
  - more similar to others

[see Rubin, 1981; Gelman et al., Bayesian Data Analysis, 2003]

Stan Example: Eight Schools: hierarchical example

School	Estimated Treatment Effect	Standard Error of Treatment Effect
A	28	15
B	8	10
C	−3	16
D	7	11
E	−1	9
F	1	11
G	18	10
H	12	18

Stan Example: Eight Schools: Model 0

Set up block stuctures so that data can be read.

```
data {
    int<lower=0> J;          // # of schools
    real y[J];               // estimated treatment
    real<lower=0> sigma[J]; // std err of effect
}
parameters {
    real<lower=0, upper=1> theta;
}
model {

}
```

Stan Example: Eight Schools: Model 1 – No pooling

Each school treated independently.

```
data {
    int<lower=0> J;          // # of schools
    real y[J];               // estimated treatment
    real<lower=0> sigma[J]; // std err of effect
}
parameters {
    real theta[J];           // school effect
}
model {
    y ~ normal(theta, sigma);
}
```

Stan Example: Eight Schools: Model 2 – Complete pooling

All schools lumped together.

```
data {
    int<lower=0> J;          // # of schools
    real y[J];               // estimated treatment
    real<lower=0> sigma[J]; // std err of effect
}
parameters {
    real theta;              // pooled school effect
}
model {
    y ~ normal(theta, sigma);
}
```

Stan Example: Eight Schools: Model 3 – Hierarchical Model

Estimate hyperparameters μ and σ^2

```
data {
    int<lower=0> J;          // # of schools
    real y[J];               // estimated treatment
    real<lower=0> sigma[J]; // std err of effect
}
parameters {
    real theta[J];           // school effect
    real mu;                 // mean for schools
    real<lower=0> tau;       // variance between schools
}
model {
    theta ~ normal(mu, tau);
    y ~ normal(theta, sigma);
}
```