- Stan: http://mc-stan.org
- bearlee@alum.mit.edu
- @djsyclik

Stan for Bayesian Inference

Daniel Lee

with Bob Carpenter, Andrew Gelman,
Matt Hoffman, Jiqiang Guo, and Ben Goodrich

Columbia University, Department of Statistics

and others

CDSS Talk: Nov 2013

Stan: What Is It?

- Open-source software package for Bayesian inference
- Language for specifying statistical models
- Written in templated C++
- Multiple interfaces:
  - RStan
  - CmdStan
  - PyStan

Stan: Why?

- Want to fit complex statistical models (relative to size of data)
- Existing tools too slow / crashes
  - WinBUGS and OpenBUGS
  - JAGS

Bayesian Data Analysis

- "By Bayesian data analysis, we mean practical methods for making inferences from data using probability models for quantities we observe and about which we wish to learn."
- "The essential characteristic of Bayesian methods is their **explict use of probability for quantifying uncertainty** in inferences based on statistical analysis."

[Gelman et al., *Bayesian Data Analysis*, 2003]

Definitions

- Basic definitions
  - y: observed data
  - θ: parameters (and other unobserved quantities)
- Distributions (use of p is overloaded in stats)
  - Joint: $p(y, \theta)$
  - Sampling / Likelihood: $p(y \,|\, \theta)$
  - Prior: $p(\theta)$
  - Data Marginal: $p(y)$
  - Posterior: $p(\theta \,|\, y)$

Inference

- non-Bayesian
  - Model y as random variable, but not θ
  - MLE: $\hat{\theta} = \arg\max_{\theta \in \Theta} p(y \,|\, \theta)$
- Bayesian
  - Model y and θ as random variables
  - Posterior distribution: $p(\theta \,|\, y)$
  - "explict use of probability for quantifying uncertainty"

Posterior Distribution

- Suppose the data y is fixed (i.e., observed). Then

$$\begin{aligned} p(\theta|y) &= \frac{p(y,\theta)}{p(y)} = \frac{p(y|\theta)\,p(\theta)}{p(y)} \\ &= \frac{p(y|\theta)\,p(\theta)}{\int p(y,\theta)\;d\theta} \\ &= \frac{p(y|\theta)\,p(\theta)}{\int p(y|\theta)\,p(\theta)\;d\theta} \\ &\propto p(y|\theta)\,p(\theta) = p(y,\theta) \end{aligned}$$

- Posterior proportional to likelihood times prior (i.e., joint)

Difficulties with Bayesian Inference

- For arbitrary joint model, $p(y, \theta)$
  - Normalizing constant is hard:
    $\int p(y|\theta)\, p(\theta)\; d\theta$
- Markov Chain Monte Carlo (MCMC)
  - For integrals that can't be solved analytically
  - But sampling and evaluation are tractable
  - Algorithms can be slow.
    (high auto-correlation in samples)

Gibbs Sampling

- Samples a parameter given data and other parameters
- Requires conditional posterior $p(\theta_n|y, \theta_{-n})$
- Conditional posterior easy in Bayesian networks
- Conditional sampling and general unidimensional sampler can both lead to slow convergence and mixing

(Geman and Geman 1984)

Metropolis-Hastings Sampling

- Proposes new point by changing all parameters randomly
- Computes accept probability of new point based on ratio of new to old log probability (and proposal density)
- Only requires evaluation of $p(\theta|y)$
- Requires good proposal mechanism to be effective
- Acceptance requires small changes in log probability
- But small step sizes lead to random walks and slow convergence and mixing

(Metropolis et al. 1953; Hastings 1970)

Hamiltonian Monte Carlo

- Converges faster and explores posterior faster when posterior is complex
- Function of interest is log posterior (up to proportion)

$$\log p(\theta|y) \propto \log p(y|\theta) + \log p(\theta)$$

- HMC exploits its gradient

$$\begin{aligned} g &= \nabla_\theta \log p(\theta|y) \\ &= \left(\frac{d}{d\theta_1} \log p(\theta|y), \ldots \frac{d}{d\theta_K} \log p(\theta|y) \right) \end{aligned}$$

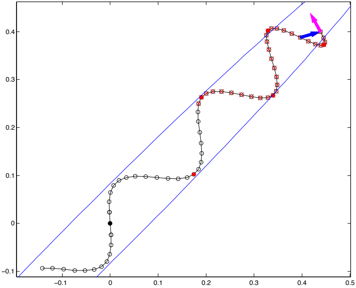
(Duane et al. 1987; Neal 1994)

HMC's Physical Analogy

1. Negative log posterior $-\log p(\theta|y)$ is potential energy
2. Start point mass at current parameter position θ
3. Add random kinetic energy (momentum)
4. Simulate trajectory of the point mass over time t
5. Return new parameter position*

$*$ In practice, Metropolis adjust for imprecision in trajectory simulation due to discretizing Hamiltonian dynamics

HMC Example Trajectory



- Blue ellipse is contour of target distribution
- Initial position at black solid circle
- Arrows indicate a U-turn in momentum

No-U-Turn Sampler (NUTS)

- HMC highly sensitive to tuning parameters
  - discretization step size ϵ
  - discretization number of steps L
- NUTS sets ϵ during burn-in by stochastic optimization (Nesterov-style dual averaging)
- NUTS chooses L online per-sample using no-U-turn idea:

  keep simulating as long as position gets further away from initial position

- Number of steps just a bit of bookkeeping on top of HMC

(Hoffman and Gelman, 2011)

Stan: Under the Hood

- Default sampler: NUTS
- Reverse-mode algorithmic differentiation for calculating gradients
- Code generation for variable transformations with Jacobian determinants
- Efficiently drop additive constants using template metaprogramming

Stan: Steps

0. Build Stan translator and library files
1. Create Bayesian model using Stan language
2. Translate Stan model to C++ code; compile
3. Sample parameters of model given data
4. Perform posterior analysis

Stan: Language

- Blocks:
  - data / transformed data
  - parameters / transformed data
  - model
  - generated quantities
- Data Types:
  - Basic: real, int, vector, row_vector, matrix
  - Constrained: simplex, ordered, cov_matrix, corr_matrix, ...

Stan: Some Examples in the Manual

- Linear Regression (12.1)
- LDA (11.1)
- Clustering (15)
- Gaussian Processes (16)

Stan: Linear Regression

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

Stan: Eight Schools

- 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

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

Stan: Eight Schools Data

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

Eight Schools: Model 0

- Make sure data can be read

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

Eight Schools: No Pooling

- Each school treated independently

```
data {
  int<lower=0> J;             // # 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);
}
```

Eight Schools: Complete Pooling

- All schools lumped together

```
data {
  int<lower=0> J;            // # 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);
}
```

Eight Schools: Partial Pooling

- Fit hyperparameter μ, but set $\tau = 25$

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

Eight Schools: Hierarchical Model

- Estimate hyperparameters μ and σ

```
data {
  int<lower=0> J;              // # 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 school
model {
  theta ~ normal(mu, tau);
  y ~ normal(theta, sigma);  }
```

Stan: Future

- ODE integrators
- More functionality
- Faster
- Riemmanian Manifold HMC
- Variational Bayes (VB)
- Stochastic VB for “big data”

The End