Hierarchical models in Stan

Daniel Lee
Columbia University, Statistics Department
bearlee@alum.mit.edu

Stan: Help

- User Guide:
  http://mc-stan.org/manual.html
- Homepage:
  http://mc-stan.org
- Stan Users group:
  https://groups.google.com/d/forum/stan-users

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 modeling 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, Newton's method

Stan to Scientists

- Flexible probabilistic language, language still growing
- Focus on science: the modeling and assumptions
  - access to multiple algorithms (default is pretty good)
  - faster and less error prone than implementing from scratch
  - efficient implementation
- Lots of (free) modeling 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
- 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

```
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: vectorization

```
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);
}
```

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

[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 schools
}
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 schools
}
model {
  theta ~ normal(mu, tau);
  y ~ normal(theta, sigma);
}
```

Eight Schools: Summary

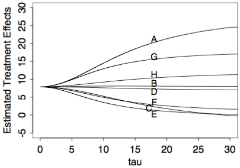


Figure 5.6 *Conditional posterior means of treatment effects,* $E(\theta_j|\tau, y)$*, as functions of the between-school standard deviation* τ*, for the educational testing example. The line for school* C *crosses the lines for* E *and* F *because* C *has a higher measurement error (see Table 5.2) and its estimate is therefore shrunk more strongly toward the overall mean in the Bayesian analysis.*

Stan's Near Future (2014)

- L-BFGS optimization
- User specified functions
- Differential equations
- Approximate inference
  - maximum marginal likelihood
  - expectation propagation
- More efficient automatic differentiation
- Refactoring, testing, adding distributions, etc.

Stan’s Future

- Riemann Manifold HMC
  - needs efficient implementation of Hessians
  - needs rewrite of current auto-diff implementation
- Variational Bayesian Inference
  - non-conjugate
  - black-box
- Stocastic Variational Bayes
  - needs stats / machine learning research
  - data partitioning

Limitations

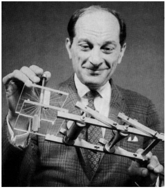
- no discrete parameters (can marginalize)
- no implicit missing data (code as parameters)
- not parallelized within chains
- language limited relative to black boxes (cf., emcee)
- limited data types and constraints
- C++ template code is complex for user extension
- sampling slow, nonscalable; optimization brittle or approx

How Stan Got its Name

- "Stan" is *not* an acronym; Gelman mashed up

    1. Eminem song about a stalker fan, and

    2. Stanislaw Ulam (1909–1984), co-inventor of Monte Carlo method (and hydrogen bomb).



Ulam holding the Fermiac, Enrico Fermi's physical Monte Carlo simulator for random neutron diffusion