Probabilistic programming and Stan

mc-stan.org

Outline

- What is probabilistic programming
- Stan now
- Stan in the future
- A bit about other software

Probabilistic programming

- Probabilistic programming framework BUGS started revolution in statistical analysis in early 1990's
  - allowed easier use of elaborate models by domain experts
  - was widely adopted in many fields of science

Probabilistic programming

- Probabilistic programming framework BUGS started revolution in statistical analysis in early 1990's
  - allowed easier use of elaborate models by domain experts
  - was widely adopted in many fields of science
- The next wave of probabilistic programming
  - Stan
  - even wider adoption including wider use in companies
  - simulators (likelihood free)
  - autonomus agents (reinforcmenet learning)
  - deep learning

Stan

- Tens of thousands of users, 100+ contributors, 50+ R packages building on Stan
- Commercial and scientific users in e.g. ecology, pharmacometrics, physics, political science, finance and econometrics, professional sports, real estate
- Used in scientific breakthroughs: LIGO gravitational wave discovery (2017 Nobel Prize in Physics) and counting black holes
- Used in hundreds of companies: Facebook, Amazon, Google, Novartis, Astra Zeneca, Zalando, Smartly.io, Reaktor, ...
- StanCon 2018 organized in Helsinki in 29-31 August http://mc-stan.org/events/stancon2018Helsinki/

Probabilistic program

- “Probabilistic programs are usual functional or imperative programs with two added constructs:
  1) the ability to draw values at random from distributions, and
  2) the ability to condition values of variables in a program via observations.”

Gordon, Henzinger, Nori, and Rajamani “Probabilistic programming.” In Proceedings of On The Future of Software Engineering (2014) via Frank Wood

Probabilistic program

- Python with scipy, R with stats
  1) the ability to draw values at random from distributions, and
- Metropolis sampler is less than 10 lines of code
  2) the ability to condition values of variables in a program via observations.

Probabilistic programming language

- Wikipedia “A probabilistic programming language (PPL) is a programming language designed to describe probabilistic models and then perform inference in those models”

Probabilistic programming language

- Wikipedia “A probabilistic programming language (PPL) is a programming language designed to describe probabilistic models and then perform inference in those models”
- To make probabilistic programming useful
  - inference has to be as automatic as possible
  - diagnostics for telling if the automatic inference doesn’t work
  - easy workflow (to reduce manual work)
  - fast enough (manual work replaced with automation)

Probabilistic programming language

- Different types
  - models defined by probability distributions
    - graphical models, e.g. BUGS, WinBUGS, JAGS
    - program model with Turing complete language, e.g. Stan
  - models defined by simulation
    - exploration, e.g. Church/Anglican
    - approximative likelihood, e.g. ELFI

Logistic regression model in Stan

```
data {
  int<lower=0> n;                    // number of observations
  int<lower=1> d;                    // covariate dimension
  matrix[n, d] X;                    // covariates
  int<lower=0,upper=1> y[n];         // target variable
  real<lower=0> p_alpha_scale;       // prior alpha scale
  real<lower=0> p_beta_scale;        // prior beta scale
}
parameters {
  real alpha;                        // intercept
  vector[d] beta;                    // coefficients
}
model {
  // priors
  alpha ~ normal(0.0, p_alpha_scale);
  beta  ~ normal(0.0, p_beta_scale);
  // observation model
  y ~ bernoulli_logit(alpha + X * beta);
}
```

Stan

- Stan compiles (transplies) the model written in Stan language to C++
  - this makes the sampling for complex models and bigger data faster
  - also makes Stan models easily portable, you can use your own favorite interface

Stan

- Compilation (unless previously compiled model available)
- Adaptation (mass matrix, step length)
- Warm-up (required for MCMC)
- Sampling several chains using HMC+NUTS
- Generated quantities
- Save posterior draws
- Report divergences, tree-depth, n_{eff}, $\hat{R}$

Stan

- Stan is probabilistic programming language

Stan

- Stan is probabilistic programming language
- Stan is probabilistic programming software written in C++
  - transpile Stan language to C++, compile C++, run inference, return results
  - language module provides Stan language
  - math library provides math, distributions and automatic differentation through C++ code (can be used without Stan)
  - algorithms module provides inference algorithms
  - interfaces in command line, R, Python, Julia, Matlab, Stata, Mathematica

Stan

- Stan is probabilistic programming language
- Stan is probabilistic programming software written in C++
  - transpile Stan language to C++, compile C++, run inference, return results
  - language module provides Stan language
  - math library provides math, distributions and automatic differentation through C++ code (can be used without Stan)
  - algorithms module provides inference algorithms
  - interfaces in command line, R, Python, Julia, Matlab, Stata, Mathematica
- Stan is framework for good statistical practices and workflow
  - manual, case studies, tutorials, discussion forum, conferences, ...

Hamiltonian Monte Carlo

- Hamiltonian Monte Carlo
  - Uses gradient information for more efficient sampling
- No U-Turn Sampling
  - adaptively selects number of steps to improve robustness and efficiency
  - can produce super-efficient antithetic Markov chains
- Adaptation in Stan
  - Step size adjustment (mass matrix) is estimated during initial adaptation phase

Functions

```
real gpareto_lpdf(vector y, real ymin, real k, real sigma) {
  // generalised Pareto log pdf
  int N = rows(y);
  real inv_k = inv(k);
  if (k<0 && max(y-ymin)/sigma > -inv_k)
    reject("k<0 and max(y-ymin)/sigma > -1/k; found k, sigma =", k, sig
  if (sigma<=0)
    reject("sigma<=0; found sigma =", sigma)
  if (fabs(k) > 1e-15)
    return -(1+inv_k)*sum(log1p((y-ymin) * (k/sigma))) -N*log(sigma);
  else
    return -sum(y-ymin)/sigma -N*log(sigma); // limit k->0
}
```

RStanarm + BRMS

- RstanARM: R style syntax for pre-compiled models
  e.g. logistic regression
  fit <- stan_glm (y $\sim$., data = diabetes, family = binomial)

RStanarm + BRMS

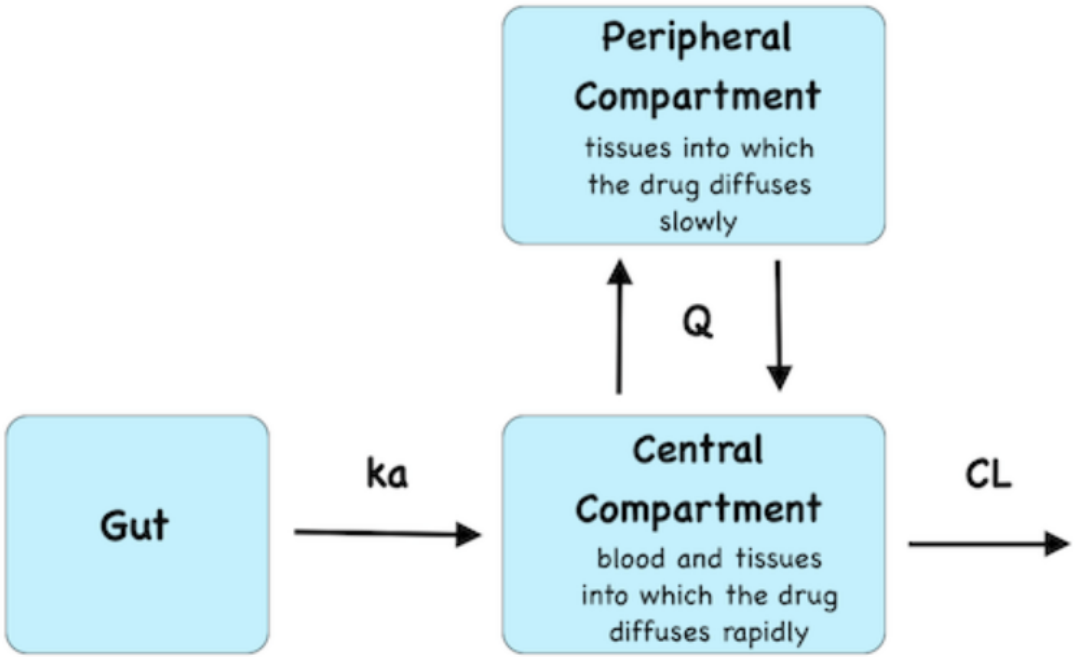
- RstanARM: R style syntax for pre-compiled models
  e.g. logistic regression
  fit <- stan_glm (y $\sim$., data = diabetes, family = binomial)
- BRMS: extended R style syntax with generation of Stan code

Additional packages

- loo: model assessment, comparison, selection, checking, averaging
- bayesplot: visualization, model checking
- shinystan: interactive visual and numerical diagnostics and posterior analysis
- rstantools: helps to make your own R package using Stan

ODE solvers

- Ordinal differential equation models used, e.g., in pharmacokinetic models



We describe the drug absorption with the following differential equations:

$$\frac{dy_{\text{gut}}}{dt} = -k_a y_{\text{gut}}$$

$$\frac{dy_{\text{central}}}{dt} = k_a y_{\text{gut}} - \left(\frac{CL}{V_{\text{central}}} + \frac{Q}{V_{\text{central}}}\right) y_{\text{central}} + \frac{Q}{V_{\text{peripheral}}} y_{\text{peripheral}}$$

$$\frac{dy_{\text{peripheral}}}{dt} = \frac{Q}{V_{\text{central}}} y_{\text{central}} - \frac{Q}{V_{\text{peripheral}}} y_{\text{peripheral}}$$

Future of Stan

- Better adaptations and diagnostics
- 2nd and 3rd order autodiff
- Riemannian Hamiltonian Monte Carlo
- Better variational inference
- MPI parallelism (multi-core, cluster)
- GPU parallelism
- OpenMP parallelism (multiple threads)
- Sparse matrix arithmetic
- Differential algebraic equation (DAE) solver
- Partial differential equation (PDE) solvers
- Definite integrators, bounded and unbounded
- Approximate GLVM (a la INLA)
- Conditional marginalisation, INLA
- Better Gaussian process and Gaussian Markov random field support
- User-defined functions with analytic gradients
- ...

Some other probabilistic programming frameworks

- JAGS
  - WinBUGS like, but seems to have replaced WinBUGS
  - many users are moving to Stan
- Pymc3
  - written in Python using Theano, looking for a new autodiff library
  - some Python users prefer instead of Stan
- Edward
  - used to be algorithm development and testing framwork in Python, is now being integrated to Google's Tensorflow
  - more machine learning flavored than Stan (e.g. more variational inference), and likely to have big impact in deep learning
- Pyro: Uber's probabilistic programming for deep learning
  - influenced by Edward
  - focus in scalability (e.g. logistic regression) and deep learning

Some of my contributions to Stan

- Algorithm diagnostics
  - MCMC convergence diagnostics (Gelman et al, 2013; work in progress)
  - importance sampling diagnostics (Vehtari et al, 2017; loo package)
  - variational inference diagnostics (Yao et al, 2018)
  - any inference diagnostics (Talts et al, 2018)
- Model checking, assessment, comparison and selection
  - Pareto smoothed importance-sampling leave-one-out (Vehtari et al, 2017ab; loo package)
  - Bayesian stacking and Pseudo-BMA+ (Yao et al, 2018; loo package)
  - Variable selection (Piironen and Vehtari, 2017a; projpred package)
- Approximative inference
  - Parallelization with EP (Vehtari et al, 2018)
  - Meta-analysis using non-linear models (Weber et al, 2017)
  - Distributional approximations, work in progress
- Priors
  - Regularized horseshoe prior (Piironen and Vehtari, 2017bc)
  - Priors for Gaussian processes (Trangucci et al, work in progress)
  - Prior choice recommendations wiki github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations
- Workflow
  - Visualization (Gabry et al, 2018)

References

- Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., and Gelman, A. (2018). Visualization in Bayesian workflow. Journal of the Royal Statistical Society Series A, accepted fo publication.
- Gelman, A., Hwang, J. and Vehtari, A. (2014). Understanding predictive information criteria for Bayesian models. Statistics and Computing, 24(6):997-1016.
- Piironen, J. and Vehtari, A. (2017). Comparison of Bayesian predictive methods for model selection, Statistics and Computing 27(3), 711–735.
- Piironen, J., and Vehtari, A. (2017). On the Hyperprior Choice for the Global Shrinkage Parameter in the Horseshoe Prior. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR 54:905–913.
- Piironen, J., and Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. Electronic Journal of Statistics, 11(2):5018-5051.
- Vehtari, A., Gelman, A. and Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Statistics and Computing, 27(5):1413–1432.
- Vehtari, A., Gelman, A. and Gabry, J. (2017). Pareto smoothed importance sampling. arXiv preprint arXiv:1507.02646.
- Vehtari, A., Gelman, A., Sivula, T., Jylänki, P., Tran, D., Sahai, S., Blomstedt, P., Cunningham, J. P., Schiminovich, D. and Robert, C. (2018). Expectation propagation as a way of life: A framework for Bayesian inference on partitioned data. arXiv preprint arXiv:1412.4869.
- Weber, S., Gelman, A., Lee, D., Betancourt, M., Vehtari, A., and Racine-Poon, A. (2017). Bayesian aggregation of average data: An application in drug development. In Annals of Applied Statistics, accepted for publication.
- Williams, D. R., Piironen, J., Vehtari, A., and Rast, P. (2018). Bayesian estimation of Gaussian graphical models with projection predictive selection. arXiv preprint arXiv:1801.05725.
- Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2017). Using stacking to average Bayesian predictive distributions. In Bayesian Analysis, doi:10.1214/17-BA1091.
- Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2018). Yes, but Did It Work?: Evaluating Variational Inference. arXiv preprint arXiv:1802.02538.