Bayesian data analysis – reading instructions 12

Aki Vehtari

Chapter 12

Outline of the chapter 12

- 12.1 Efficient Gibbs samplers (not part of the course)
- 12.2 Efficient Metropolis jump rules (not part of the course)
- 12.3 Further exetensions to Gibbs and Metropolis (not part of the course)
- 12.4 Hamiltonian Monte Carlo (used in Stan)
- 12.5 Hamiltonian dynamics for a simple hierarchical model (read through)
- 12.6 Stan: developing a computing environment (read through)

R and Python demos at https://avehtari.github.io/BDA_course_Aalto/demos.html

- See rstan_demo.Rmd, pystan_demo.py, pystan_demo.ipynb for demos how to use Stan from R/Python and several model examples

There is only 8 pages to read (sections 12.4-12.6) what is inside Stan.

MCMC animations

These don't include the specific version of dynamic HMC in Stan, but are useful illustrations anyway.

- Markov Chains: Why Walk When You Can Flow?
  http://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/
- MCMC animation site by Chi Feng https://chi-feng.github.io/mcmc-demo/

Hamiltonian Monte Carlo

An excellent review of static HMC (the number of steps in dynamic simulation are not adaptively selected) is

- Radford Neal (2011). MCMC using Hamiltonian dynamics. In Brooks et al (ed), *Handbook of Markov Chain Monte Carlo*, Chapman & Hall / CRC Press. Preprint https://arxiv.org/pdf/1206.1901.pdf.

Stan uses a variant of dynamic Hamiltonian Monte Carlo (using adaptive number of steps in the dynamic simulation), which has been further developed since BDA3 was published. The first dynamic HMC variant was

- Matthew D. Hoffman, Andrew Gelman (2014). The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *JMLR*, 15:1593–1623 http://jmlr.org/papers/v15/hoffman14a.html.

The No-U-Turn Sampler gave the name NUTS which you can see often associated with Stan, but the current dynamic HMC variant implemented in Stan has some further developments described (mostly) in

- Michael Betancourt (2018). A Conceptual Introduction to Hamiltonian Monte Carlo. arXiv preprint arXiv:1701.02434 https://arxiv.org/abs/1701.02434.

Instead of reading all above, you can also watch a video

- Scalable Bayesian Inference with Hamiltonian Monte Carlo by Michael Betancourt https://www.youtube.com/watch?v=jUSZboSq1zg

Divergences and BFMI

Divergences and Bayesian Fraction of Missing Information (BFMI) are HMC specific convergence diagnostics developed by Michael Betancourt after BDA3 was published.

- Divergence diagnostic checks whether the discretized dynamic simulation has problems due to fast varying density. See more in a case study http://mc-stan.org/users/documentation/case-studies/divergences_and_bias.html.
- BFMI checks whether momentum resampling in HMC is sufficiently efficient. See more in https://arxiv.org/abs/1604.00695
- Brief Guide to Stan's Warnings https://mc-stan.org/misc/warnings.html provides summary of available convergence diagnostics in Stan and how to interpret them.

Further information about Stan

- http://mc-stan.org/ & http://mc-stan.org/documentation/
  - I recommend to start with these
    - Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A. Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell (2015) In press for Journal of Statistical Software. Stan: A Probabilistic Programming Language. http://www.stat.columbia.edu/~gelman/research/published/stan-paper-revision-feb2015.pdf
    - Andrew Gelman, Daniel Lee, and Jiqiang Guo (2015) Stan: A probabilistic programming language for Bayesian inference and optimization. In press, Journal of Educational and Behavior Science. http://www.stat.columbia.edu/~gelman/research/published/stan_jebs_2.pdf
  - Basics of Bayesian inference and Stan, parts 1+2 Jonah Gabry & Lauren Kennedy https://www.youtube.com/playlist?list=PLuwyh42iHquU4hUBQs20hkBsKSMrp6H0J
  - Modeling Language User's Guide and Reference Manual (more complete reference with lot's of examples) https://github.com/stan-dev/stan/releases/download/v2.17.0/stan-reference-2.17.0.pdf

Compiler and transpiler

This is a minor comment on the terminology. As a shorthand it's common to see mentioned just Stan compiler, but sometimes the transpiler term is also mentioned as in the slides for this part.

Wikipedia (https://en.wikipedia.org/wiki/Source-to-source_compiler):

A source-to-source translator, source-to-source compiler (S2S compiler), transcompiler, or transpiler[1] is a type of translator that takes the source code of a program written in a programming language as

its input and produces an equivalent source code in the same or a different programming language. A source-to-source translator converts between programming languages that operate at approximately the same level of abstraction, while a traditional compiler translates from a higher level programming language to a lower level programming language.

So it is more accurate to say that the Stan model code is first transpiled to a C++ code, and then that C++ code is compiled to machine code to create an executable program. Cool thing about the new stanc3 transpiler (https://github.com/stan-dev/stanc3) is that it can create also, for example, LLVM IR or Tensorflow code.

Using transpiler and compiler allows to develop Stan language to be good for writing models, but get the benefit of speed and external libraries of C++, Tensorflow, and whatever comes in the future.