Automatically generating Reversible Jump Markov chain Monte Carlo samplers from user-provided target and proposal probabilistic programs

Reversible Jump Markov chain Monte Carlo (RJMC)

- As with Metropolis–Hastings (M–H), RJMC generates candidate states via a proposal distribution and accepts/rejects according to an acceptance ratio:

\[
\text{M–H: } \alpha_{x \rightarrow x'} = \frac{f(x')g(u(x'))}{f(x)g(u(x))}
\]

\[
\text{RJMC: } \alpha_{x \rightarrow x'} = \frac{f(x')g(u(x'))f(x)g(u(x))}{f(x)g(u(x'))f(x')g(u(x))}
\]

- Application to probabilistic programming requires automatic derivation of \(\alpha\)

- Jacobian depends on structure of problem-specific proposal distribution

Automatic derivation of RJMC inference program

- Recursively apply rewrite rules to invert outermost operation:

\[
x^* \leftarrow \text{inserAt}(i, \text{exp}(u^*)) = x \text{ if final}(x^* \neq x_i)
\]

- Solving case expressions, i.e. \(h(u^*; x^*) = x\) where

\[
h(u^*; x^*) = \begin{cases} 
  e_1 & \text{if } e_0 = 1 \\
  e_2 & \text{if } e_0 = 2 \\
  \vdots & \\
  e_n & \text{if } e_0 = n
\end{cases}
\]

- First solve component expressions, e.g.:

\[
\begin{array}{c|c|c}
  j & x_i = x_{i+1} & x_i = x_i \\
  \hline
  1 & \gamma & \beta \\
  2 & \delta & \\
  3 & \gamma & \\
  4 & \delta & \\
\end{array}
\]

- Combine to form final solutions:

\[
k \leftarrow \begin{cases} 
  0 & \text{if } x_i = x_{i+1} \\
  1 & \text{if } x_i = x_i - 1 \\
  2 & \text{if } x_i = x_i - 1 \\
  3 & \text{if } x_i = x_i - 1
\end{cases}
\]

\[
w_i = \begin{cases} 
  \alpha & \text{if } k = 0 \\
  b & \text{if } k = 1 \\
  0 & \text{if } k = 2 \\
  \alpha & \text{if } k = 3
\end{cases}
\]

Automatic transformation inversion

Stochaskell intermediate representation

Inversion by recursive graph rewriting

Stochaskell example: coal mining disasters (Green, 1995)

User-supplied code (high-level)

Output

Marginal plots provide density estimates for elements of \(x\) (change point locations) and \(g\) (step heights) for \(w = 1, 2, 3\)

Future work

- Improve efficiency of generated code
  - General-purpose code optimisation
  - Algebraic expression simplification

- Support inversion of more general transformations than

\[
h(u; x) = (h_1(u_1; x), h_2(u_2; x), \ldots, h_u(u_i; x))
\]

References
