Introduction

Opportunity Higher-order probabilistic programming languages such as Venture make it easy to represent models that combine:

- 1. discrete latent variables,
- 2. stochastic control flow, and
- 3. combinatorial stochastic processes

High-dimensional continuous latent spaces.

Challenge These kinds of probabilistic programs present new inference challenges that existing probabilistic programming systems have not addressed:

1. the energy function’s curvature can vary unpredictably,
2. the number of continuous dimensions can grow or shrink dynamically,
3. gradients may not be derivable analytically, and
4. the conditional dependence and conditional existence of variables caused by stochastic control flow.

Solution We describe adaptive hybrid Monte Carlo transition operators and gradient-based MAP schemes for inference in higher-order probabilistic programs that overcome these limitations.

Context

Venture induces a Markov chain on the space of possible probabilistic program execution histories [1]. The current state of the chain is stored in an explicit graph structure that tracks conditional dependence and conditional existence of variables. Local transitions We realize adaptive hybrid Monte Carlo and gradient-based MAP as local transition operators that change part of the stored execution history. These operators can be mixed with other proposal kernels to compose higher-order inference schemes for various statistical models.

Math

Local Inference on Scaffold The local inference problem is to sample from

\[ \Pr(P | A) \propto \Pr(P | A) \sum C \Pr(c | A, P) \Pr(A, P | c) \]

Hamiltonian Monte Carlo The Hamiltonian Monte Carlo methods exploit gradient information to make better transitions through the space of real values \( V \) of a particle in the dynamic system:

\[ H(V) = V(\dot{V}) + K(V) \]

where \( U(P) \) is the log density, \( \dot{V} \) and \( V \) are the kinetic and the potential energy.

Conditioning on Regeneration We cannot evaluate or compute the gradient of the natural energy function

\[ U(P) = -\log(\Pr(P | A, d)) \]

at any given \( P \). Conditional on \( K \), the randomness of sampling \( c \) given \( P \) and \( par \), we can evaluate

\[ U(P) = -\log(\Pr(P | A, R)) \]

\[ -\log(\Pr(P | A, R)) - \log(\Pr(\hat{A}, P, R)) \]

We effectively treat the regeneration graph as auxiliary variables and sample them together with the principal node. The implementation accomplishes “sampling” the randomness \( R \) controlling the state of the random number generator available to the probabilistic program.

Reverse-Mode Autodifferentiation We compute the needed gradient of the energy function \( U(P) \) by reverse-mode automatic differentiation. See [3] for more details. In the terminology of automatic differentiation, the scaffold functions as the tape.

Code

(defvar gradient-of-regen-computer (object:))
(defvar self scaffold)
(defvar self._rng_state (randn-returns))
(defvar self._rng_values (values))
(defvar regen (self scaffold))
def _call (self, values):
  return _regen (self, rng_state: regen(self scaffold), values)
  return _regen (self, scaffold: scaffold, compute_gradents:True)

(defvar gradient-of-regen-computer (object:))
(defvar self scaffold)
(defvar self._rng_state (randn-returns))
(defvar self._rng_values (values))
(defvar regen (self scaffold))
def _call (self, values):
  return _regen (self, rng_state: regen(self scaffold), values)
  return _regen (self, scaffold: scaffold, compute_gradents:True)

In probabilistic programs, the notion of a Markov blanket needs to be generalized to handle the conditional independence and conditional existence of variables caused by stochastic control flow. In Venture, we call the generalization a Scaffold, which is shown in Figure 1.

Inference with Automatic Gradients in Higher-Order Probabilistic Programs

[TIANLIN SHI, ALEXEY RADUL AND VIKASH MANSINGHKA] MIT

Multivariate Gaussian

Bayesian Spike-and-Slab

Figure 3: Visual comparison of skewed 2D Gaussian by different methods.

Figure 4: Autocorrelation versus dimension plot for different methods.

Figure 5: Autocorrelation versus dimension plot for different inference methods for spike-and-slab regression.

DP Mixture of Gaussian

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
<tr>
<td>( \text{argmax} )</td>
<td>( \text{argmax} )</td>
</tr>
</tbody>
</table>

Contrition

1. Extend Hamiltonian Monte Carlo (MHC) methods to probabilistic programming by mixing over random regeneration, which in the limiting case recovers HMC.
2. Propose gradient-based inference for high-order probabilistic programs, and allow mixture of inference schemes.
3. Incorporate the automatic differentiation framework to compute gradients for local transitions.

References
