Output-Sensitive Adaptive Metropolis-Hastings for Probabilistic Programs

David Tolpin, Jan Willem van de Meent,
Brooks Paige, Frank Wood
University of Oxford

May 11th, 2015

Paper: `http://arxiv.org/abs/1501.05677`
Slides: `http://offtopia.net/almh-slides.pdf`

Outline

Intuition

Probabilistic program:

- A program with random computations.
- Distributions are conditioned by 'observations'.
- Values of certain expressions are 'predicted' — **the output**.

Can be written in any language (extended by `sample` and `observe`).

Example: Model Selection

```
(let [;; Model
      dist (sample (categorical [[normal 1/4] [gamma 1/4]
                                 [uniform-discrete 1/4]
                                 [uniform-continuous 1/4]]))
      a (sample (gamma 1 1))
      b (sample (gamma 1 1))
      d (dist a b)]

  ;; Observations
  (observe d 1)
  (observe d 2)
  (observe d 4)
  (observe d 7)

  ;; Explanation
  (predict :d (type d))
  (predict :a a)
  (predict :b b)))
```

Definition

A **probabilistic program** is a stateful deterministic computation $\mathcal{P}$:

- Initially, $\mathcal{P}$ expects no arguments.
- On every call, $\mathcal{P}$ returns
  - a distribution F,
  - a distribution and a value (G, y),
  - a value z,
  - or $\perp$.
- Upon returning F, $\mathcal{P}$ expects $x \sim F$.
- Upon returning $\perp$, $\mathcal{P}$ terminates.

A program is run by calling $\mathcal{P}$ repeatedly until termination.
The probability of each **trace** is $\propto \prod_{i=1}^{|\boldsymbol{x}|} p_{F_i}(x_i) \prod_{j=1}^{|\boldsymbol{y}|} p_{G_j}(y_j)$.

Outline

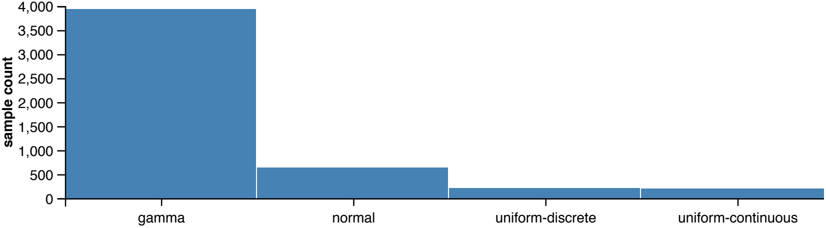
Inference Objective

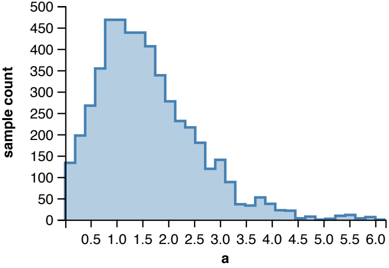
- Suggest **most probable explanation** (MPE) - most likely assignment for all non-evidence variable given evidence.
- Approximately **compute integral** of the form

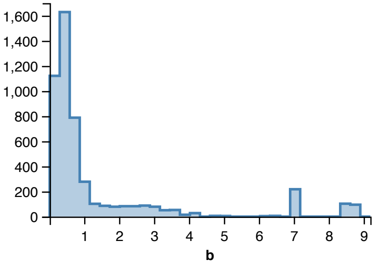
$$\Phi = \int_{-\infty}^{\infty} \varphi(x) p(x) dx$$

- Continuously and **infinitely generate a sequence of samples** drawn from the distribution of the output expression — so that someone else puts it in good use (vague but common). ✓

Example: Inference Results







Importance Sampling

loop

Run $\mathcal{P}$, computing weight $w = \prod_{j=1}^{|\boldsymbol{y}|} p_{G_j}(y_j)$.

output $\boldsymbol{z}$, w.

end loop

- Simple — good.
- Slow convergence (unless one knows the answer) — bad.

Can we do better?

Lightweight Metropolis-Hastings (LMH)

Run $\mathcal{P}$ once, remember $\boldsymbol{x}, \boldsymbol{z}$.
loop
    Uniformly select x_i.
    Propose a value for x_i.
    Run $\mathcal{P}$, remember $\boldsymbol{x'}, \boldsymbol{z'}$.
    Accept $(\boldsymbol{x}, \boldsymbol{z} = \boldsymbol{x'}, \boldsymbol{z'})$ or reject with MH probability.
    Output $\boldsymbol{z}$.
end loop

Can we do better?

Adaptive MCMC

Adaptation opportunities:

1. Random walk (instead of proposing from priors).
2. Adapting random walk parameters.
3. Selecting each x_i with different probability. ✓

Outline

Lightweight MH with Adaptive Scheduling

Maintains vector of weights $\boldsymbol{W}$ of random choices:

Initialize $\boldsymbol{W}^0$ to a constant.
Run $\mathcal{P}$ once.
for $t = 1 \ldots \infty$ **do**
    Select x_i^t with probability $\alpha_i^t = W_i^t / \sum_{i=1}^{|\boldsymbol{x}^t|} W_i^t$.
    Propose a value for x_i^t.
    Run $\mathcal{P}$, accept or reject with MH probability.
    if accepted **then**
        Compute $\boldsymbol{W}^{t+1}$ based on the *program output.*
    else
        $\boldsymbol{W}^{t+1} \leftarrow \boldsymbol{W}^t$
    end if
end for

Quantifying Influence of Program Output

- Objective: faster convergence of *program output* $\boldsymbol{z}$.
- Adaptation parameter: probabilities of selecting random choices for modification.
- Optimization target: maximize the **change** in the program output:

$$R^t = \frac{1}{|\boldsymbol{z}^t|} \sum_{k=1}^{|\boldsymbol{z}^t|} \mathbf{1}(z_k^t \neq z_k^{t-1}).$$

W_i reflects the anticipated change in $\boldsymbol{z}$ from modifying x_i.

Delayed Changes

Modifying x2 affects the output ...

```
(let [x1 (sample (normal 1 10))
      x2 (sample (normal x1 1))]
  (observe (normal x2 1) 2)
  (predict x1))
```

... but only when x1 is also modified.

Backpropagating rewards

- For each x_i, reward r_i and count c_i are kept.
- A *history* of modified random choices is attached to every z_j.

When modification of x_k accepted:

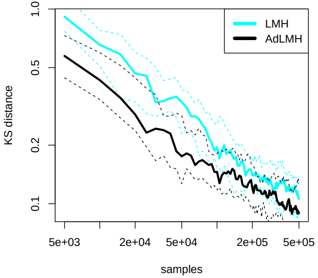
Append x_k to the history.
if $\boldsymbol{z}^{t+1} \neq \boldsymbol{z}^{t}$ **then**
    $w \leftarrow \frac{1}{|history|}$
    for x_m **in** history **do**
        $\overline{r}_m \leftarrow r_m + w, \; c_m \leftarrow c_m + w$
    end for
    Flush the history.
else
    $c_k \leftarrow c_k + 1$
end if

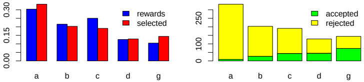
Outline

Convergence — GP hyperparameter estimation

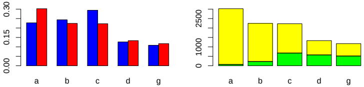
$$f \sim \mathcal{GP}(m, k),$$

$$\text{where } m(x) = ax^2 + bx + c, \quad k(x, x) = de^{\frac{(x-x')^2}{2g}}.$$

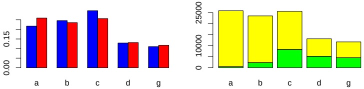




1000 samples



10000 samples

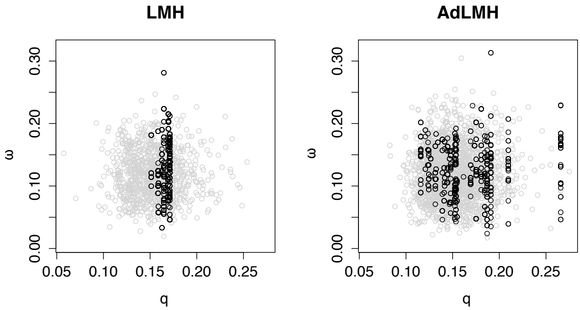


100000 samples

Sample Size — Kalman Smoother

$$\boldsymbol{x}_t \sim \mathrm{Norm}(\boldsymbol{A} \cdot \boldsymbol{x}_{t-1}, \boldsymbol{Q}), \qquad \boldsymbol{y}_t \sim \mathrm{Norm}(\boldsymbol{C} \cdot \boldsymbol{x}_t, \boldsymbol{R}).$$

$$\boldsymbol{A} = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix}, \qquad \boldsymbol{Q} = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix}.$$



100 16-dimensional observations,
500 samples after 10 000 samples of burn-in.

Outline

Summary

- A scheme of rewarding random choices based on program output.
- An approach to propagation of choice rewards to MH proposal scheduling parameters.
- An application of this approach to LMH, where the probabilities of selecting each variable for modification are adjusted.

Thank You