Path Finding under Uncertainty through Probabilistic Inference

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

June 8th, 2015

Paper: `http://arxiv.org/abs/1502.07314`
Slides: `http://offtopia.net/ctp-pp-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 $\bot$.
- Upon returning F, $\mathcal{P}$ expects $x \sim F$.
- Upon returning $\bot$, $\mathcal{P}$ terminates.

A program is run by calling $\mathcal{P}$ repeatedly until termination.
The probability of each **trace** is

$$p_{\mathcal{P}}(\boldsymbol{x}) = \propto \prod_{i=1}^{|\boldsymbol{x}|} p_{F_i}(x_i) \prod_{j=1}^{|\boldsymbol{y}|} p_{G_j}(y_j)$$

.

Outline

Inference Objective

- 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).

Inference Objective

- 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).
- Approximately **compute integral** of the form

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

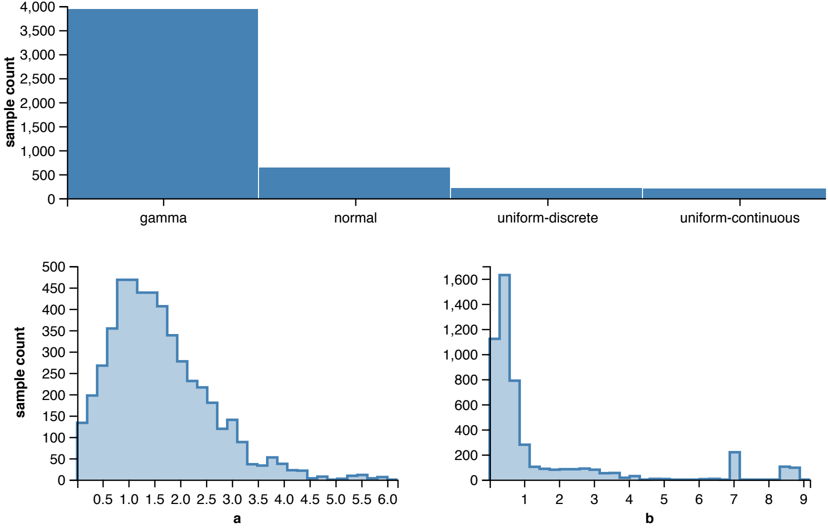
Inference Objective

- 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).
- Approximately **compute integral** of the form

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

- Suggest **most probable explanation** (MPE) - most likely assignment for all non-evidence variables given evidence. ✓

Example: Inference Results



Outline

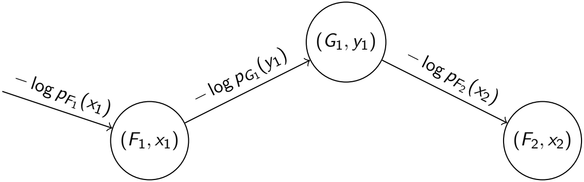
Connection between MAP and Shortest Path

Maximizing the (logarithm of) trace probability

$$\log p_{\mathcal{P}}(\boldsymbol{x}) = \sum_{i=1}^{|\boldsymbol{x}|} \log p_{F_i}(x_i) + \sum_{j=1}^{|\boldsymbol{y}|} \log p_{G_j}(y_j) + C$$

corresponds to finding the shortest path in a graph $G = (V, E)$:

- $V = \{(F_i, x_i)\} \cup \{(G_j, y_j)\}$.
- Edge costs are $-\log p_{F_i}(x_i)$ or $-\log p_{H_j}(y_j)$.



Marginal MAP as Policy Learning

In Marginal MAP, assignment of a *part* of the trace $\boldsymbol{x}^\theta$ is inferred. In a probabilistic program:

- $\boldsymbol{x}^\theta$ becomes the program output $\boldsymbol{z}$.
- $\boldsymbol{z}$ is marginalized over $\boldsymbol{x} \setminus \boldsymbol{x}^\theta$.
- $\boldsymbol{x}^\theta_{MAP} = \arg\max p_{\mathcal{P}}(\boldsymbol{z})$.

Determining $\boldsymbol{x}^\theta_{MAP}$ corresponds to learning a policy $\boldsymbol{x}^\theta$ which minimizes the *expected* path length

$$\mathbb{E}_{\boldsymbol{x}\setminus\boldsymbol{x}^\theta}\left[-\sum_{i=1}^{\left|\boldsymbol{x}^\theta\right|}\log p_{F_i^\theta}(x_i^\theta) - \sum_{j=1}^{|\boldsymbol{y}|}\log p_{G_j}(y_j)\right]$$

Outline

Policy Learning through Probabilistic Inference

Require: *agent*, *Instances*, *Policies*

1: $\mathit{instance} \leftarrow \textsc{Draw}(\mathit{Instances})$
2: $\mathit{policy} \leftarrow \textsc{Draw}(\mathit{Policies})$
3: $\mathit{cost} \leftarrow \textsc{Run}(\mathit{agent}, \mathit{instance}, \mathit{policy})$
4: $\textsc{Observe}(1, \mathrm{Bernoulli}(e^{-\mathit{cost}}))$
5: $\textsc{Print}(\mathit{policy})$

The log probability of the output policy is

$$\log p_{\mathcal{P}}(\mathit{policy}) = -\mathit{cost}(\mathit{policy}) + \log p_{\mathit{Policies}}(\mathit{policy}) + C$$

When policies are drawn uniformly

$$\log p_{\mathcal{P}}(\mathit{policy}) = -\mathit{cost}(\mathit{policy}) + C'$$

Outline

Canadian Traveller Problem

CTP is a problem finding the shortest travel distance in a graph where some edges may be blocked.

Given

- Undirected weighted graph $G = (V, E)$.
- The initial and the final location nodes s and t.
- Edge weights $w : E \to \mathcal{R}$.
- Traversability probabilities: $p_o : E \to (0, 1]$.

find the shortest travel distance from s to t — the sum of weights of all traversed edges.

The Simplest CTP Instance — Two Roads

Given

- two roads with probability being open p_1 and p_2,
- costs of each road c_1 and c_2,
- cost of bumping into a blocked road c_b,

learn the optimum policy q.

```
(defquery tworoads
  (loop []
      (let [o1 (sample (flip p1))
            o2 (sample (flip p2))]
        (if (not (or o1 o2)) (recur)
          (let [q (sample (uniform-continuous 0. 1.))
                s (sample (flip (- 1 q)))]
            (let [distance (if s (if o1 c1 (+ c2 cb))
                                 (if o2 c2 (+ c1 cb)))]
              (observe +factor+ (- distance))
              (predict :q q)))))))
```

Learning Stochastic Policy for CTP

Depth-first search based policy:

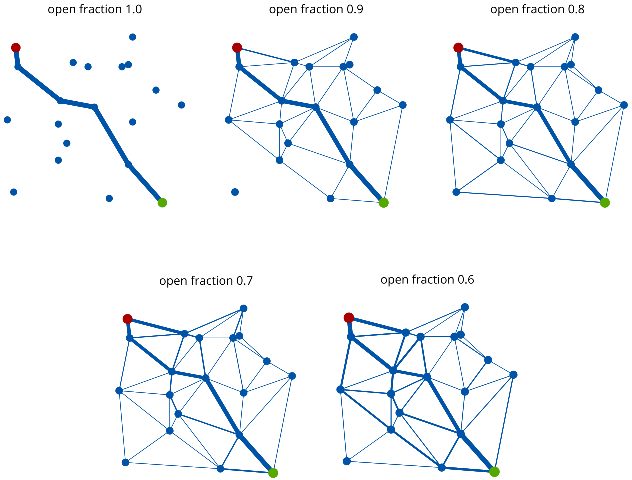
- the agent traverses G in depth-first order.
- the policy specifies the probabilities of selecting each adjacent edge in every node.

Require: $\mathrm{CTP}(G, s, t, w, p)$

for $v \in V$ **do**
    $policy(v) \leftarrow \textsc{Draw}(\mathrm{Dirichlet}(\mathbf{1}^{\deg(v)}))$
end for
repeat
    $instance \leftarrow \textsc{Draw}(\mathrm{CTP}(G, w, p))$
    $(reached, distance) \leftarrow \textsc{StDFS}(instance, policy)$
until $reached$
$\textsc{Observe}(1, \mathrm{Bernoulli}(e^{-distance}))$
$\textsc{Print}(policy)$

Inference Results — CTP Travel Graphs

Learned policies:



Line widths indicate the frequency of travelling each edge.

Outline

Summary

- Discovery of bilateral correspondence between probabilistic inference and policy learning for path finding.
- A new approach to policy learning based on the established correspondence.
- A realization of the approach for the Canadian traveller problem, where improved policies were consistently learned by probabilistic program inference.

Thank You