Path Finding under Uncertainty through Probabilistic Inference

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

June 8th, 2015

Slides: http://offtopia.net/ctp-pp-slides.pdf
Outline

Probabilistic Programming

Inference

Path Finding and Probabilistic Inference

Stochastic Policy Learning

Case Study: Canadian Traveller Problem

Summary
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
(observer d 1)
(observer d 2)
(observer d 4)
(observer 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

\[
p_{\mathcal{P}}(\mathbf{x}) = \propto \prod_{i=1}^{\mid \mathbf{x} \mid} p_{F_i}(x_i) \prod_{j=1}^{\mid \mathbf{y} \mid} p_{G_j}(y_j)
\]
Outline

Probabilistic Programming

Inference

Path Finding and Probabilistic Inference

Stochastic Policy Learning

Case Study: Canadian Traveller Problem

Summary
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

Probabilistic Programming

Inference

Path Finding and Probabilistic Inference

Stochastic Policy Learning

Case Study: Canadian Traveller Problem

Summary
Connection between MAP and Shortest Path

Maximizing the (logarithm of) trace probability

\[ \log p_P(x) = \sum_{i=1}^{\mid x \mid} \log p_{F_i}(x_i) + \sum_{j=1}^{\mid y \mid} \log p_{G_j}(y_j) + C \]

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 \(- \log p_{G_j}(y_j)\).
Marginal MAP as Policy Learning

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

- $x^\theta$ becomes the program output $z$.
- $z$ is marginalized over $x \setminus x^\theta$.
- $x_{MAP}^\theta = \text{arg max } p_P(z)$.

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

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

Probabilistic Programming

Inference

Path Finding and Probabilistic Inference

Stochastic Policy Learning

Case Study: Canadian Traveller Problem

Summary
Policy Learning through Probabilistic Inference

**Require:** agent, Instances, Policies

1. instance ← Draw(Instances)
2. policy ← Draw(Policies)
3. cost ← Run(agent, instance, policy)
4. Observe(1, Bernoulli(e^{-cost}))
5. Print(policy)

The log probability of the output policy is

\[ \log p_p(policy) = -\text{cost}(policy) + \log p_{Policies}(policy) + C \]

When policies are drawn uniformly

\[ \log p_p(policy) = -\text{cost}(policy) + C' \]
Outline

Probabilistic Programming

Inference

Path Finding and Probabilistic Inference

Stochastic Policy Learning

Case Study: Canadian Traveller Problem

Summary
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 $w : E \rightarrow \mathbb{R}$.
- Traversability probabilities: $p_o : E \rightarrow (0, 1]$.

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 $q$.

```lisp
(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: \(\text{CTP}(G, s, t, w, p)\)

1: \(\text{for } v \in V \text{ do} \)
2: \(\text{policy}(v) \leftarrow \text{DRAW}(\text{Dirichlet}(1^{\text{deg}(v)}))\)
3: \(\text{end for} \)
4: \(\text{repeat} \)
5: \(\text{instance} \leftarrow \text{DRAW}(\text{CTP}(G, w, p))\)
6: \((\text{reached}, \text{distance}) \leftarrow \text{STDFS(instance, policy)}\)
7: \(\text{until reached} \)
8: \(\text{Observe}(1, \text{Bernoulli}(e^{-\text{distance}}))\)
9: \(\text{Print}(\text{policy})\)
Inference Results — CTP Travel Graphs

Learned policies:

- Open fraction 1.0
- Open fraction 0.9
- Open fraction 0.8
- Open fraction 0.7
- Open fraction 0.6

Line widths indicate the frequency of travelling each edge.
Outline

Probabilistic Programming

Inference

Path Finding and Probabilistic Inference

Stochastic Policy Learning

Case Study: Canadian Traveller Problem

Summary
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