Denotational account of approximate Bayesian inference

Adam Ścibior

Cambridge-Tübingen PhD programme

PROBPROG 2018

Formal syntax

t, s, r	$::=$		terms
		x	variable
	$\mid$	$\lambda x.t$	function abstraction
	$\mid$	$t\ s$	function application
	$\mid$	$()$	unit
	$\mid$	(t, s)	tuple creation
	$\mid$	$\textbf{match}\ t$ $\textbf{with}\ (y, z) \to s$	tuple inspection
	$\mid$	n	natural numbers
	$\mid$	$t + s$	addition

Type system

$$
\begin{array}{llll}
\alpha, \beta, \gamma & ::= & & \text{types} \\
& & \mathbb{N} & \text{natural numbers} \\
& \mid & \alpha \to \beta & \text{function} \\
& \mid & \mathbf{1} & \text{unit} \\
& \mid & \alpha * \beta & \text{finite product}
\end{array}
$$

$$
\frac{}{\Gamma \vdash () : \mathbf{1}} \quad \frac{}{\Gamma \vdash n : \mathbb{N}} \quad \frac{\Gamma \vdash t : \mathbb{N} \qquad \Gamma \vdash s : \mathbb{N}}{\Gamma \vdash t + s : \mathbb{N}}
$$

$$
\frac{}{\Gamma \vdash \alpha : \alpha}((\alpha : \alpha) \in \Gamma) \quad \frac{\Gamma, \alpha : \alpha \vdash t : \beta}{\Gamma \vdash \lambda\alpha : \alpha.t : \alpha \to \beta} \quad \frac{\Gamma \vdash t : \beta \to \alpha \qquad \Gamma \vdash s : \beta}{\Gamma \vdash t\, s : \alpha}
$$

$$
\frac{\Gamma \vdash t : \alpha \qquad \Gamma \vdash s : \beta}{\Gamma \vdash (t, s) : \alpha * \beta} \quad \frac{\Gamma \vdash t : \beta * \gamma \qquad \Gamma, \alpha : \beta, \beta : \gamma \vdash s : \alpha}{\Gamma \vdash \textbf{match}\ t\ \textbf{with}\ (\alpha, \beta) \to s : \alpha}
$$

What are semantics good for?

- reasoning about programs
- proving correctness
- implementing compilers
- designing languages

Popular approaches to formalizing semantics

- operational
- denotational
- axiomatic

Types denote spaces

$$[\![\mathbb{N}]\!] ::= \mathbb{N}$$

$$[\![\mathbf{1}]\!] ::= \mathbf{1}$$

$$[\![\alpha * \beta]\!] ::= [\![\alpha]\!] \times [\![\beta]\!]$$

$$[\![\alpha \to \beta]\!] ::= [\![\beta]\!]^{[\![\alpha]\!]}$$

Terms denote elements

$$[\![n]\!](\rho) ::= n$$

$$[\![t + s]\!](\rho) ::= [\![t]\!](\rho) + [\![t]\!](\rho)$$

$$[\![(t, s)]\!](\rho) ::= ([\![t]\!](\rho), [\![t]\!](\rho))$$

$$[\![x]\!](\rho) ::= \rho(x)$$

$$[\![\lambda x.t]\!](\rho) ::= \lambda y.\, [\![t]\!](\rho[x \to y])$$

Probabilistic programs denote measures

$$[\![\alpha]\!] = \mathrm{M}\,\alpha$$

$$[\![\textbf{bern } p]\!]\,(S) = p \cdot 1_S(\mathit{true}) + (1 - p) \cdot 1_S(\mathit{false})$$

$$[\![\textbf{score } r]\!]\,(S) = r \cdot 1_S(\mathit{unit})$$

Quasi-Borel spaces

A convenient category for higher-order probability theory
Chris Heunen, Ohad Kammar, Sam Staton, Hongseok Yang
in LiCS 2017

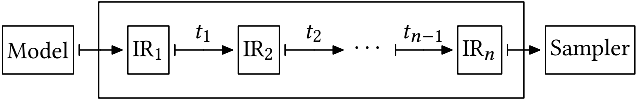
Probabilistic programs are executed as samplers

$$[\![\alpha]\!] = \mathsf{List}\,\mathbb{I} \to \mathbb{R}_+ \times \alpha \times \mathsf{List}\,\mathbb{I}$$

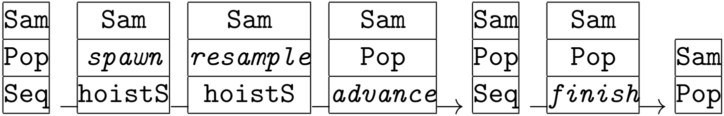
$$[\![\textbf{bern}\ p]\!] = \lambda(u :: u_s).\ (1, u < p, u_s)$$

$$[\![\textbf{score}\ r]\!] = \lambda u_s.\ (r, unit, u_s)$$

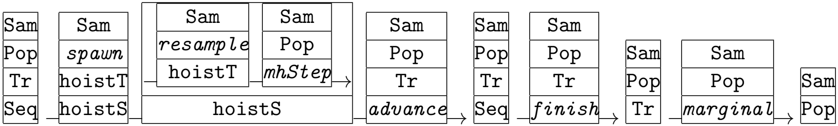
Inference as program transformation



Sequential Monte Carlo



Resample-Move Sequential Monte Carlo



Proving correctness

Denotational validation of higher-order Bayesian inference
Adam Ścibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K. Moss, Chris Heunen, Zoubin Ghahramani
in POPL 2018

Practical advantages

Functional programming for modular Bayesian inference
Adam Ścibior, Ohad Kammar, Zoubin Ghahramani
in ICFP 2018

`https://github.com/adscib/monad-bayes`

Sequential Monte Carlo

for $i = 1 : N$ **do**
  $W_i = \frac{1}{N}$
end for
} Spawn particles

for $t = 1 : T$ **do**
  $W \leftarrow \frac{1}{N} \sum_i W_i$
  for $i = 1 : N$ **do**
    $A_i \sim Categorical(\{W_j\}_{j=1}^{N})$
    $\tilde{X}_i \leftarrow X_{A_i}$
    $W_i \leftarrow W$
} Resample
  end for
  for $i = 1 : N$ **do**
    $X_i^t \sim p(x^t | \tilde{X}_i^{t-1})$
    $W_i^t = W_i^{t-1} \frac{p(dx^t, y^t | \tilde{X}_i^{t-1})}{p(dx^t | \tilde{X}_i^{t-1})}(X_i^t)$
  end for
} Advance
end for

Sequential Monte Carlo

```
smc :: MonadInfer m ⇒ Int → Int → Seq (Pop m) a → Pop m a
smc k n =
  finish .
  compose k (advance . hoistS resample) .
  hoistS (spawn n >>)
```

Resample-Move Sequential Monte Carlo

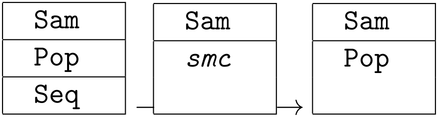
for $i = 1 : N$ **do**
    $W_i = \frac{1}{N}$
end for
} Spawn particles

for $t = 1 : T$ **do**
    $W \leftarrow \frac{1}{N} \sum_i W_i$
    for $i = 1 : N$ **do**
        $A_i \sim Categorical(\{W_j\}_{j=1}^N)$
        $\tilde{X}_i \leftarrow X_{A_i}$
        $W_i \leftarrow W$
} Resample
    end for
    for $i = 1 : N$ **do**
        for $j = 1 : K$ **do**
            $\tilde{X}_i^t \sim K(\tilde{X}_i^t, \cdot)$
        end for
} MH steps
    end for
    for $i = 1 : N$ **do**
        $X_i^t \sim p(x^t | \tilde{X}_i^{t-1})$
        $W_i^t = W_i^{t-1} \frac{p(dx^t, y^t | \tilde{X}_i^{t-1})}{p(dx^t | \tilde{X}_i^{t-1})}(X_i^t)$
    end for
} Advance
end for

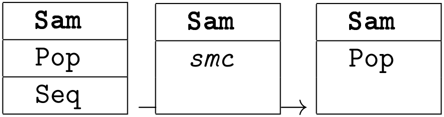
Resample-Move Sequential Monte Carlo

```
rmsmc :: MonadInfer m ⇒ Int → Int → Int →
           Seq (Tr (Pop m)) a → Pop m a
rmsmc k n t = marginal . finish .
    compose k (advance . hoistS (
        compose t mhStep . hoistT resample)) .
    (hoistS . hoistT) (spawn n >>)
```

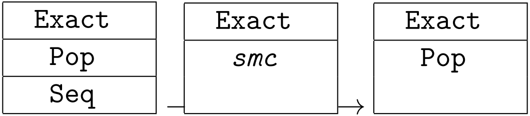
Testing



Testing



Testing



Future directions

- gradient-based inference
- provably correct implementations
- novel compositions