

Statistical programming with categorical measure theory & LazyPPL

Swaraj Dash, Younesse Kaddar, Hugo Paquet, Sam Staton



Probabilistic Programming

Bayes' law

$$\underbrace{p(x | d)}_{\text{posterior}} \propto \underbrace{p(d | x)}_{\text{likelihood}} \times \underbrace{p(x)}_{\text{prior}}$$

- `sample` : prior
- `observe` : likelihood, conditioning on the observed datapoints d
- `infer` / `normalize` : posterior, inference from effects (observations d) to causes (parameters x)

Demo

```
linear :: Prob (Double -> Double)
```

```
linear = do
```

```
  a <- normal 0 3
```

```
  b <- normal 0 3
```

```
  let f = \x -> a * x + b
```

```
  return f
```

```
regress :: Double -> Prob (a -> Double) -> [(a, Double)] -> Meas (a -> Double)
```

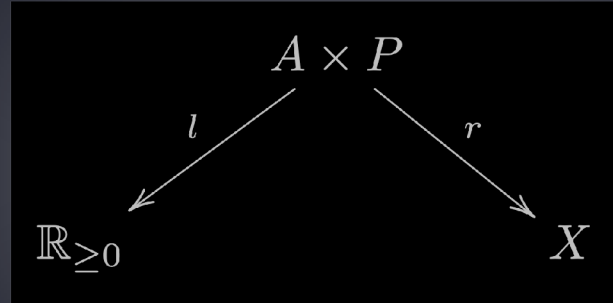
```
regress σ prior dataset = do
```

```
  f <- sample prior
```

```
  mapM_ (\(x, y) -> score $ normalPdf (f x) σ y) dataset
```

```
  return f
```

Semantically



Inference:

- pick a parameter p by sampling a point from the area under the curve of the weight function l
- use this parameter p to determine a result $r(p)$.

Problems

- *Meas* not cartesian closed:
 - no function space like `Double -> Double`
- Not known if there is a strong monad of measures
 - the category of s-finite kernels is distributive symmetric monoidal
 - but is it a Kleisli category?

Inverse Transform Sampling

“ **Noise outsourcing lemma:** The law of every Borel-valued random variable can be obtained as a pushforward of the uniform measure $[0, 1]$. ”

Given a parameterized probability distribution $k: A \rightarrow \mathcal{P}(B)$, there is a function $f: A \times [0, 1] \rightarrow B$ such that for all a ,

$$k(a) \stackrel{\mathcal{D}}{=} f(a, u) \quad \text{where } u \sim \mathcal{U}([0, 1])$$

```
k(a) = do { u ← uniform 0 1; return f (a, p)
```

Quasi-Borel Spaces

- Ω : fixed uncountable standard Borel space
 - LazyPPL: infinite rose trees, with splitting $\gamma: \Omega \cong \Omega \times \Omega$
- **quasi-Borel space** X : set $M_X \subseteq X^\Omega$ of *random elements*
- *Qbs*: cartesian closed, commutative monad of measures
 - LazyPPL: `Prob` vs `Meas` \rightarrow two Kleisli categories
 - Laziness in program evaluation: monoidal category $\mathcal{Kl}(\mathbf{Prob})$ has a terminal unit

Proba and Measure Kernels

- **Prob**: Probability kernels $f: X \times \Omega \rightarrow Y$ modulo equivalence
 - *Composition*: $X \times \Omega \xrightarrow{X \times \gamma} X \times \Omega \times \Omega \xrightarrow{f \times \Omega} Y \times \Omega \xrightarrow{g} Z$
 - Monad on *Qbs*: $X \mapsto X^\Omega$
- **Meas**: Measure kernels $[0, \infty] \xleftarrow{l} X \times \Omega \xrightarrow{f} Y$ mod equiv
 - *Composition*: compose the proba kernels, multiply the weights
 - Monad on *Qbs*: $X \mapsto (X \times \mathbb{R})^\Omega$

Correspondence

Proba	Categorical Proba	Prob Prog
Spaces	Objects	Types
Proba kernels	Morphisms	Programs
Fubini's theorem	Interchange law / Commutativity	Reordering lines
Marginalisation	Semi-cartesianness / Affineness	Discarding / Laziness

Demo

- Poisson point process
- Piecewise regression
- Program induction
- Wiener process
- More on [https://lazyppl.bitbucket.io/!](https://lazyppl.bitbucket.io/)

