Probabilistic Reasoning, from Graphs to Circuits

Breandan Considine

Abstract

- Classical statistical modeling and inference requires pen-and-paper derivation
- Computers are steadily improving at logical reasoning and symbolic processing
- Can we employ computer-aided reasoning to assist with probabilistic modeling?
- Can we design an inference system to derive algorithms from first principles?

Denotational Semantics

The grammar of probabilistic modeling can be described approximately as follows:

$$\begin{array}{llll}
\mathcal{P} \to \text{Gaussian} & \mathcal{V} \to \mathcal{V}, \mathcal{V} & \mathcal{S} \to \mathcal{V} \sim \mathcal{P} & \mathcal{P} \to \prod_{i=1}^{n} \mathcal{P} \\
\mathcal{P} \to \text{Bernoulli} & \mathcal{P} \to P(\mathcal{V}) & \mathcal{E} \to \mathcal{V} \pm \mathcal{V} & \mathcal{P} \to \sum_{i=1}^{n} \mathcal{P} \\
\mathcal{P} \to \text{Dirichlet} & \mathcal{P} \to P(\mathcal{V} \mid \mathcal{V}) & \mathcal{E} \to \mathcal{V} \times \mathcal{V} & \mathcal{P} \to \int \mathcal{P} d\mathcal{V} \\
\mathcal{V} \to A \mid \ldots \mid Z & \mathcal{P} \to P(\mathcal{E}) & \mathcal{E} \to \mathbb{E}[\mathcal{E}] & \mathcal{P} \to \mathcal{P} \div \mathcal{P}
\end{array}$$

Given a distribution over a set X, we can *sample* it to produce a *random variable*:

$$\frac{\Gamma \vdash P(X) : X \times \Sigma \to \mathbb{R}^+ \quad x \sim P(X)}{\Gamma \vdash x : (X \times \Sigma \to \mathbb{R}^+) \rightsquigarrow X}\,\text{Sample}$$

The joint distribution $P(X, Y)$ is a distribution over the product space $X \times Y$:

$$\frac{\Gamma \vdash P(X) : X \times \Sigma \to \mathbb{R}^+ \quad \Gamma \vdash P(Y) : Y \times \Sigma \to \mathbb{R}^+}{\Gamma \vdash P(X, Y) : X \times Y \times \Sigma \to \mathbb{R}^+}\,\text{Joint}$$

If we have a joint distribution $P(X, Y)$ and see $Y = y$, this observation is called *conditioning* and the resulting distribution over X is called a *conditional distribution*:

$$\frac{\Gamma \vdash P(X, Y) : X \times Y \times \Sigma \to \mathbb{R}^+ \quad \Gamma \vdash y : Y}{\Gamma \vdash P(X \mid Y = y) : X \times \Sigma \to \mathbb{R}^+}\,\text{Cond}$$

We can use Bayes' rule to exchange the order of a conditional distribution as follows:

$$\frac{\overbrace{P(X \mid Y)}^{\text{Liklihood}}\ \overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(Y \mid X)}_{\text{Normalize}} \propto \underbrace{P(X \mid Y)}_{\text{Observe}}\,\underbrace{P(Y)}_{\text{Sample}}}\,\text{Bayes}$$

When a conditional distribution $P(X \mid Y)$ does not depend on its prior Y, or may be factorized into $P(X)P(Y)$, we conclude that X and Y are independent RVs:

$$\frac{P(X \mid Y) = P(X)}{X \perp Y}\,\text{Indep} \qquad \frac{P(X, Y) = P(X)P(Y)}{X \perp Y}\,\text{Fact}$$

If a joint distribution $P(X, Y \mid Z)$ can be factored as the product of conditionals $P(X \mid Z)P(Y \mid Z)$, X and Y are said to be *conditionally independent given* Z:

$$\frac{P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z)}{X \perp Y \mid Z}\,\text{CondIndep}$$

Operational Semantics

To sample, we can use the Kolmogorov-Smirnov transform on a uniform PRNG:

$$\frac{x \sim P(X) \quad \texttt{CDF} : x \mapsto \int P(X = x)dx}{x = \texttt{INV(CDF(PRNG()))}}\,\text{Draw}$$

There are various ways of combining two probability distributions. For independent RVs, we could combine them using the product or convolution distribution:

$$\frac{P(X) \quad P(Y)}{P(X, Y) = P(X)P(Y)}\,\text{Join} \qquad \frac{P(X) \quad P(Y)}{P(X + Y) = P(X) * P(Y)}\,\text{Conv}$$

In general, to combine two arbitrary RVs, we need to know their dependence relation. If two variables are related by a dyadic function $f(x, y)$, we can use Fubini-Tonelli:

$$\frac{\dfrac{x \sim P(X) \quad y \sim P(Y) \quad f : X \times Y \to Z}{P(Z \mid X = x, Y = y) = \int_{X \times Y} f(x, y) d(x \times y)}}{\int_Y \int_X f(x, y) dx dy = \int_X \int_Y f(x, y) dy dx}\,\text{FuTon}$$

To remove an RV from a joint distribution $P(X, Y)$ we can *marginalize*, or integrate over the conditional. The resulting distribution is called a *marginal distribution*:

$$\frac{\Gamma \vdash P(X, Y) : X \times Y \times \Sigma \to \mathbb{R}^+}{\Gamma \vdash P(X) : X \times \Sigma \to \mathbb{R}^+ \propto \int_Y P(X \mid Y = y) dy}\,\text{Margin}$$

Probabilistic Circuits

A semiring algebra has two operators, $\oplus$ and $\otimes$ which are closed under distributivity:

$$\frac{X \otimes (Y \oplus Z)}{(X \otimes Y) \oplus (X \otimes Z)}\,\text{LDist} \qquad \frac{(Y \oplus Z) \otimes X}{(X \otimes Y) \oplus (X \otimes Z)}\,\text{RDist}$$

The sum-product network (SPN) is a commutative semiring on simple distributions:

$$PC \to v \sim \mathcal{D} \qquad PC \to PC \oplus PC \qquad PC \to PC \otimes PC$$

A Bayesian belief network or Bayes network (BN) is an acyclic DGM of the form:

$$P(x_1, \ldots, x_D) = \prod_{i=1}^{D} P(x_i \mid \texttt{parents}(x_i))$$

Given a BN, we can compile it to an SPN using the procedure from Butz (2019):

procedure BN2SPN(b: BN): SPN
  c ← VariableEliminate(b)
  s ← RedistributeParams(c)
  s ← CompileMarginalized(s)
  return Reduce(s)
end procedure

procedure Reduce(s_0: SPN): SPN
  s_1 ← AddTerminals(s_0)
  s_1 ← MergeProducts(s_1)
  if $s_0 = s_1$ **then return** s_1
  else return Reduce(s_1)
end procedure

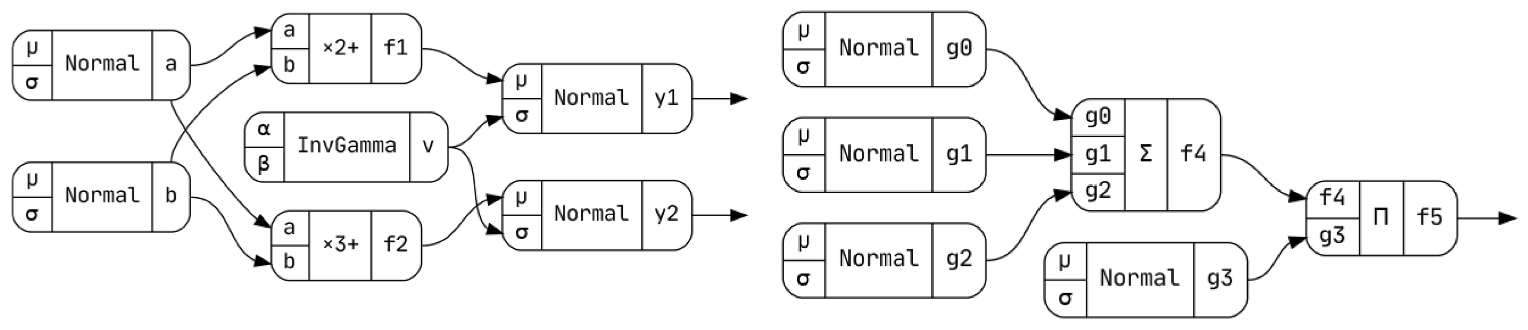
Generative Modeling

Our DSL, Markovian, can construct models in a probabilistic context free grammar. We validate it by implementing some generative models for Bayesian regression:

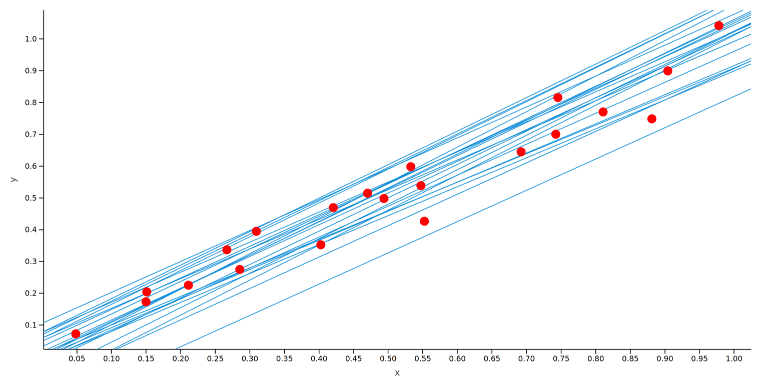
```
val a  by Gaussian(0, 9.0)
val b  by Gaussian(0, 9.0)
val v  by InvGamma(.5, .5)
val f1 by a * 2 + b
val f2 by a * 3 + b
val y1 by Gaussian(f1, v)
val y2 by Gaussian(f2, v)
sample(f1, f2, y3, y4).show()
```

```
val g0 by Gaussian(0.1,  1.0)
val g1 by Gaussian(5.0,  1.0)
val g2 by Gaussian(10.0, 1.0)
val g3 by Gaussian(5.0,  2.0)
val f4 by g0 + g1 + g2
val f5 by g3 * f4
compare(g0, g1, g2, g3
        g4, g5).show()
```

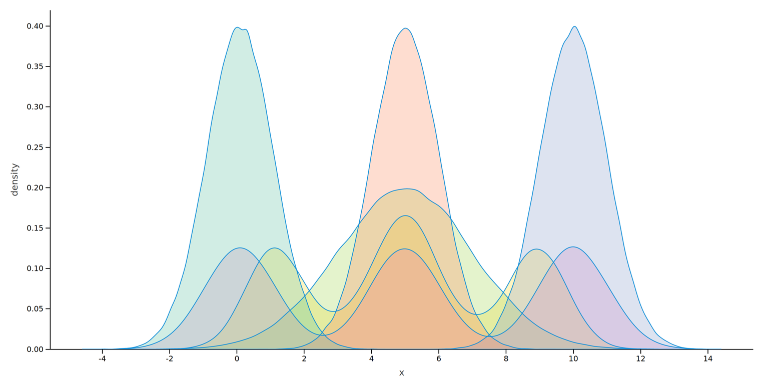
We can depict the sequential programs written above as a computational graphs. The topology of this data structure is permutation invariant to instruction reording.



We can also represent these functions by plotting their probability distributions:



Bayesian linear regression



Sum-product network

We plan extend this DSL to support discriminative models and statistical inference.

Contributions

- Lift numerical computation graph into the domain of probability kernels
- Implements general-purpose combinators for various probability distributions
- Implements domain-specific estimators for the algebra of random variables
- Allows flexible algebraic rewriting to optimize for e.g. latency or numerical stability
- Lowering to numerical values when necessary to perform e.g. sampling/inference

Code available at: https://github.com/breandan/markovian

Paper available at: https://brea.ndan.co/public/probcirc.pdf