More support for symbolic disintegration

Praveen Narayanan
Computer Science
Indiana University
Bloomington, Indiana, USA
pravnar@indiana.edu

Chung-chieh Shan
Computer Science
Indiana University
Bloomington, Indiana, USA
ccshan@indiana.edu

1 Introduction

In this work we present an automatic conditioning tool for observing distributions over $\mathbb{R}$ that are *mixtures of discrete and continuous parts*, as well distributions over *disjoint sums* and *dependent products*.

This is useful in several scenarios. The primary example is Russell's GPA problem [Hay et al. 2017], where the nationality of a student – hypothesized using *clamped* normal distributions – is to be determined using their observed GPA. This involves conditioning on mixtures of continuous and discrete distributions over $\mathbb{R}$. A different kind of scenario emerges in the *single-site* proposal distribution for the Metropolis-Hastings MCMC sampler, where for any given transition we perturb at most one dimension of a multi-dimensional state. This involves conditioning on a dependent product of variables arising from mixture distributions. Finally, the *reversible-jump* MCMC sampler presents a third scenario, where the number of state dimensions can differ between transitions. This involves conditioning on a disjoint sum of expressions of varying dimensions. The conditioning tool of this work handles all three kinds of applications.

In this paper we will demonstrate our conditioning tool on mixture distributions. We will start by illustrating a workflow that uses the current state of the art in symbolic conditioning – due to Shan and Ramsey [2017] – before describing how we improve upon their method.

2 The state of the art

Imagine that you are comparing hypotheses that explain a phenomenon. For example, consider a Gaussian mixture model, where you posit that a real-valued variable t (the phenomenon) must arise from one of two normal distributions, $\mathcal{N}(2,3)$ and $\mathcal{N}(5,6)$ (the hypotheses). We can express this model in the core Hakaru probabilistic language:

```
gmm :: M (R × Bool)
gmm = do {x ↖∼ normal 2 3;  ⊕  do {x ↖∼ normal 5 6;
          return (x , true)}        return (x , false)}
```

For an observed value t, we would normally compare these two hypotheses by comparing two numbers: the *densities* of $\mathcal{N}(2,3)$ and $\mathcal{N}(5,6)$ at the point t. These densities, if notated d_1 and d_2 respectively, are functions with familiar analytic forms:

$$d_1 = \lambda t.\ \mathbf{sqrt}\ (2\pi \cdot 3^2)^{-1} \cdot \mathbf{exp}\ (-(t-2)^2 \cdot (2 \cdot 3^2)^{-1}) \quad (1)$$

$$d_2 = \lambda t.\ \mathbf{sqrt}\ (2\pi \cdot 6^2)^{-1} \cdot \mathbf{exp}\ (-(t-5)^2 \cdot (2 \cdot 6^2)^{-1}) \quad (2)$$

Formally, *density* is a binary relation between distributions, and if a distribution m has a density d with respect to a distribution b, we write $m = b \otimes d$. In **gmm**, both hypotheses have densities with respect to the *Lebesgue* distribution:

$$m_1 = \mathcal{N}(2,3) = \mathbf{lebesgue} \otimes d_1 \quad (3)$$

$$m_2 = \mathcal{N}(5,6) = \mathbf{lebesgue} \otimes d_2 \quad (4)$$

This allows us to conveniently compare m_1 and m_2 by comparing d_1 and d_2 instead; in doing so we are implicitly choosing to use the densities with respect to **lebesgue**. Shan and Ramsey [2017] help us automate this step with a conditioning tool called **disintegrate**, which when given a model of type $\mathbb{M}\ (\alpha \times \beta)$ produces a *disintegration*, i.e., a posterior distribution of type $\alpha \to \mathbb{M}\ \beta$. In the case of the mixture model **gmm**, we get the posterior as the following $\mathbb{R} \to \mathbb{M}$ `Bool` function:

```
disintegrate gmm                                              (5)
  ≡ λt. do {factor (d1 t);  ⊕  do {factor (d2 t);
            return true}            return false}
```

From this posterior representation we can directly read off the desired densities d_1 and d_2.

3 Our contribution

The **gmm** model mixed two continuous hypotheses – what if one of them were instead a discrete distribution? Specifically, consider a model where the phenomenon is known to be distributed according to $\mathcal{N}(2,3)$, *clamped at* 0. We can model this as a mixture of a (continuous) normal distribution and a (discrete) point-mass *Dirac* distribution:

```
clamped :: M (R × Bool)
clamped = do {x ↖∼ normal 2 3;  ⊕  do {x ↖∼ normal 2 3;
              observe (0 ≤ x);          observe (x < 0);
              return (x , true)}        return (0 , false)}
```

Given a real-valued observation t, how do we compare a continuous and a discrete hypothesis? In **gmm**, both hypotheses had densities with respect to Lebesgue. This remains

PPS 2018, January 9, 2018, Los Angeles, California, USA

Application	Model	Base
GPA problem	**do** {$x \leftsquigarrow$ **normal** 3 1; $g \leftsquigarrow$ **return** (**min** 4 (**max** 0 x)) **return** (g , **true**)} $\oplus$ **do** {$x \leftsquigarrow$ **normal** 7 2; $g \leftsquigarrow$ **return** (**min** 10 (**max** 0 x)) **return** (g , **false**)}	**lebesgue** $\oplus$ **return** 0 $\oplus$ **return** 4 $\oplus$ **return** 10
Single-site proposal	**do** {$x \leftsquigarrow$ **normal** 0 1; $y \leftsquigarrow$ **normal** 0 1; $p \leftsquigarrow$ (**do** {$x' \leftsquigarrow$ **normal** x 1; **return** (x' , y)}) $\oplus$ (**do** {$y' \leftsquigarrow$ **normal** y 1; **return** (x , y')}); **return** (((x , y) , p) , ())}	**do** {$p \leftsquigarrow$ **do** {$x \leftsquigarrow$ **lebesgue**; $y \leftsquigarrow$ **lebesgue**; **return** (x , y)}; $a \leftsquigarrow$ **lebesgue** $\oplus$ **return** (**fst** p); $b \leftsquigarrow$ **lebesgue** $\oplus$ **return** (**snd** p); **return** (p , (a , b))}
Reversible-jump proposal	**do** {$x \leftsquigarrow$ **normal** 0 1; $y \leftsquigarrow$ **normal** 0 1; $x' \leftsquigarrow$ **normal** x 1; **return** ((**inr** (x , y) , **inl** x') , ())} $\oplus$ **do** {$x' \leftsquigarrow$ **normal** 0 1; $x \leftsquigarrow$ **normal** x' 1; $y \leftsquigarrow$ **normal** x' 1; **return** ((**inl** x' , **inr** (x , y)) , ())}	**do** {$m \leftsquigarrow$ **return** (**do** {$l \leftsquigarrow$ **lebesgue**; **return** (**inl** l)} $\oplus$ **do** {$x \leftsquigarrow$ **lebesgue**; $y \leftsquigarrow$ **lebesgue**; **return** (**inr** (x , y))}); $e \leftsquigarrow m$; $e' \leftsquigarrow m$; **return** (e , e')}

Figure 1. Models and bases over mixtures, sums, and products

true of the first hypothesis in **clamped**:

$$m_1' = \textbf{do } \{x \leftsquigarrow \textbf{normal } 2\ 3; \textbf{ observe } (0 \le x); \textbf{ return } x\} \quad (6)$$

$$d_1' = \lambda t.\ \textbf{if } (0 \le t) \textbf{ then } d_1\ t \textbf{ else } 0 \quad (7)$$

$$m_1' = \textbf{lebesgue} \otimes d_1' \quad (8)$$

The same cannot be said of the second hypothesis. While it does not have a density with respect to Lebesgue, it does have a density with respect to Dirac at 0:

$$m_2' = \textbf{do } \{x \leftsquigarrow \textbf{normal } 2\ 3; \textbf{ observe } (x < 0); \textbf{ return } 0\} \quad (9)$$

$$d_2' = \lambda t.\ \textbf{if } (t = 0) \textbf{ then } \frac{1 - \textbf{erf } (\textbf{sqrt } (2) / 3)}{2} \textbf{ else } 0 \quad (10)$$

$$m_2' = (\textbf{return } 0) \otimes d_2' \quad (11)$$

Although d_1' and d_2' both return non-negative real numbers, it is intuitively a unit mismatch to compare these functions. To compare densities we must derive them with respect to the same *base distribution*. For **clamped**, that base distribution is **lebesgue** $\oplus$ **return** 0.

As in the previous example, we'd like to automatically derive densities in order to compare hypotheses. We cannot use the existing disintegrator to do this, since it implicitly uses **lebesgue** as the base distribution. Thus, we generalize **disintegrate** by letting it take an extra argument for the base distribution. Our extended disintegrator produces the following posterior:

$$\begin{aligned} &\textbf{disintegrate clamped } (\textbf{lebesgue} \oplus \textbf{return } 0) \quad (12)\\ &\equiv \lambda t.\ (\textbf{do } \{\textbf{observe } (0 < t); \textbf{ factor } (d_1\ t); \textbf{ return true}\} \oplus (\textbf{do } \{\textbf{observe } (t = 0); x \leftsquigarrow \textbf{normal } 2\ 3; \textbf{ observe } (x < 0); \textbf{ return false}\}) \end{aligned}$$

We can read off the density for m_1' from the first branch of the output in eq. (12). Note that it is slightly different from d_1': the constraint on t correctly goes from $0 \le t$ (when the base is **lebesgue**) to $0 < t$ (for the new base).

For the density of m_2', the second branch of the output in eq. (12) gives us d_2' *pre-simplification*. We can obtain d_2' with exact inference via computer algebra [Carette and Shan 2016].

4 Applications

We have demonstrated a tool for conditioning mixture distributions over $\mathbb{R}$. Using **disintegrate** we can now correctly condition Russell's GPA problem [Hay et al. 2017]. The model and base distribution for this problem are shown in fig. 1.

Another source of applications can be found in MCMC samplers [Andrieu et al. 2003]. Here the disintegrator proves useful for comparing densities in the acceptance ratio. Our generalized **disintegrate** conditions distributions over *dependent product spaces* to handle the *single-site* proposal of a Metropolis-Hastings sampler. It can also condition distributions over *disjoint sums* to handle a *reversible-jump* proposal. The models and bases for these applications are also shown in fig. 1.

5 Ongoing work: inferring bases

So far we have generalized **disintegrate** to accept a base distribution as argument, and we have seen a variety of bases across applications. We are currently extending the disintegrator to automatically *infer* a base distribution that permits disintegration for a given model. We envision that, given the model alone, the disintegrator will produce a set of *density constraints* on the base distribution. For example, the call **disintegrate clamped** b currently produces:

$$\{\textbf{lebesgue} <: b\ ,\ \textbf{return } 0 <: b\} \quad (13)$$

Here the relation $m <: b$ states that m has a density with respect to b. This relation is a preorder, and we expect to infer a *principal base measure* whenever the set of constraints can be satisfied.

To satisfy the set of constraints in eq. (13) we have to construct the mixture base distribution used in eq. (12). We aim

for the disintegrator to give a helpful error result when the constraints are unsatisfiable.

We have illustrated symbolic conditioning over a variety of base distributions, and this can be a powerful tool for Bayesian inference. By automatically inferring base distributions we can further increase productivity and correctness in probabilistic programming.

References

Christophe Andrieu, Nando De Freitas, and et al. 2003. An Introduction to MCMC for Machine Learning. (2003).

Jacques Carette and Chung-chieh Shan. 2016. *Simplifying Probabilistic Programs Using Computer Algebra.* Springer International Publishing, Cham, 135–152. https://doi.org/10.1007/978-3-319-28228-2_9

Nicholas Hay, Siddharth Srivastava, Yi Wu, and Stuart Russell. 2017. The Extended Semantics For Probabilistic Programming Languages. In *Workshop on Probabilistic Programming Semantics at POPL 17.*

Chung-chieh Shan and Norman Ramsey. 2017. Exact Bayesian Inference by Symbolic Disintegration. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017).* ACM, New York, NY, USA, 130–144. https://doi.org/10.1145/3009837.3009852