The Base Measure Problem and its Solution

Alexey Radul
Google Reseach

Boris Alexeev
Google Research

Abstract

Probabilistic programming systems generally compute with probability density functions, leaving the base measure of each such function implicit. This mostly works, but creates problems when densities with respect to different base measures are accidentally combined or compared. Mistakes also happen when computing volume corrections for continuous changes of variables, which in general depend on the support measure. We motivate and clarify the problem in the context of a composable library of probability distributions and bijective transformations. We solve the problem by standardizing on Hausdorff measure as a base, and deriving formulas for comparing and combining mixed-dimension densities, as well as updating densities with respect to Hausdorff measure under diffeomorphic transformations. We also propose a software architecture that implements these formulas efficiently in the common case. We hope that by adopting our solution, probabilistic programming systems can become more robust and general, and make a broader class of models accessible to practitioners.

1 Introduction

Suppose we are designing a composable library of software to represent probability distributions and transformations thereof which, following The TFP Team (2019), we will call "bijectors". TensorFlow Probability (The TFP Team, 2019) and PyTorch Distributions (Paszke et al., 2019) are such libraries in their own right, targeting the probabilistic machine learning space. General-purpose probabilistic programming languages such as Stan (Carpenter et al., 2017), BLOG (Li and Russell, 2013), Anglican (Tolpin et al., 2016), or Venture (Mansinghka et al., 2014) must of necessity also include such libraries, to implement their primitive distributions and deterministic functions.

Suppose furthermore that we are designing this library to operate on explicitly vector-valued probability distributions.[1] This is the case for TensorFlow Probability and PyTorch Distributions, for instance, to take advantage of vectorized hardware; and studying the general case is instructive even for a scalar design.

In this setting, it's conventional to represent a probability distribution P as a function $p : \mathbb{R}^n \to \mathbb{R}$ computing the probability density of P at points in $\mathbb{R}^n$, together with a sampler $x \sim s()$ drawing random variates $x \in \mathbb{R}^n$ distributed according to P. Given an invertible map $f : \mathbb{R}^n \to \mathbb{R}^n$, the pushforward fP is then sampled as $x' \sim f(s())$, and its density is computed as

$$fp(x') = \frac{1}{|\det(Jf_x)|} p(x), \quad \text{for } x = f^{-1}(x'). \qquad (1)$$

The Jacobian-determinant correction $1/|\det(Jf_x)|$ accounts for the possibility that f changes the volume of an infinitesimal volume element near x. The Jacobian determinant can be computed by forming the Jacobian of f, for example with automatic differentiation; but for many functions f, it's available more efficiently. Thus a conventional choice is to package such f, together with their inverse f^{-1}, in a `Bijector` class with a method for computing said Jacobian determinant.

This conventional architecture admits a serious bug. We name this bug the *Base Measure Problem*, because it consists of neglecting the base measure with respect to which we are computing densities. Our contributions answer these questions:

- What's the problem? We give a clear and intuitive example of the Base Measure Problem in Section 2;

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

[1] No distinction need be made for our purposes between vectors, matrices, tuples, or other structures, as long as joint distributions over non-trivial powers of $\mathbb{R}$ are in scope.

- How common is it? We briefly survey several areas where the Base Measure Problem recurs in different guises in Section 3;

- What's the right answer? We propose a more nuanced standard base measure in Section 4, and derive complete density manipulation formulas for that choice using standard results; and

- How do we fix our software? We detail common special cases of our formulas in Section 5, as well as how to arrange a software library to optimize them. We also report in Section 6 on a preliminary test of integrating these ideas into an existing library of probabilistic software.

We stress that while we do propose to explicitly represent information about measures, all the information we will need will be local to a single point (the point itself and various directional derivatives thereat), and thus require no symbolic algebra to compute with.

2 Motivating Example

Consider the uniform distribution P on the unit circle in $\mathbb{R}^2$. The natural probability density function to write down for this is

$$p(x, y) = \begin{cases} \frac{1}{2\pi} & \text{when } x^2 + y^2 = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Of course, if we were sticklers we would note that the base measure implied by the type of the samples is Lebesgue measure on $\mathbb{R}^2$; and with respect to this base measure the density of P is $+\infty$ on points on the unit circle. But that's clearly less helpful to our users than $1/2\pi$, and $1/2\pi$ is *a* density for P, just with respect to[2] Lebesgue measure along the circle. So let's wing it and go with that.

Now, a user looking at a density computed for a sample has no way to know which base measure we meant, so we have created an instance of the base measure problem. To see how it bites us, let f be a somewhat contrived non-isotropic scaling of $\mathbb{R}^2$, given by $f(x, y) = (2x, 20y)$. What happens when we try to compose our uniform distribution P with our non-isotropic scaling f? The sampler is fine, but the conventional density rule (1) gives

$$\begin{aligned} fp_{\text{bad}}(x', y') &= \frac{1}{40} p(f^{-1}(x', y')) \\ &= \begin{cases} \frac{1}{80\pi} & \text{when } (x'/2)^2 + (y'/20)^2 = 1, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

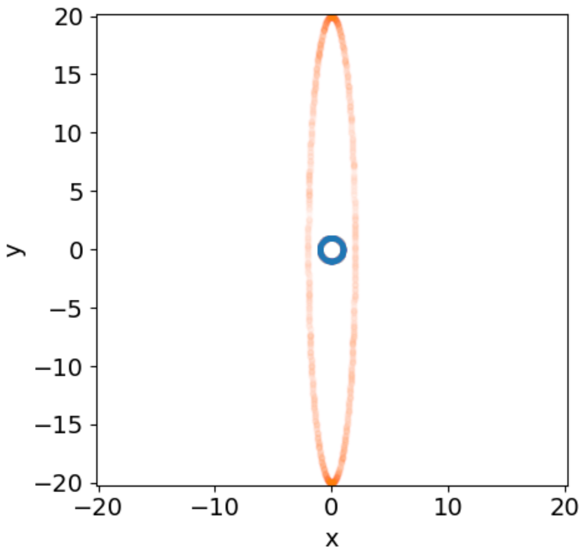

Figure 1: 3000 samples from the uniform distribution on the unit circle (blue), and the same samples scaled non-isotropically by 2x along the x axis and 20x along the y axis (orange). The transformed distribution is clearly not uniform, even though the original distribution is, and the Jacobian of the transformation is constant over $\mathbb{R}^2$.

Now we have definitely made a mistake. First of all, fp_{bad} doesn't integrate to 1, because the perimeter of the ellipse $(x'/2)^2 + (y'/20)^2 = 1$ is approximately 81.28, which is considerably less than $80\pi \approx 251.32$. Second, as we can see by drawing a few samples and plotting them in Figure 1, the true distribution fp isn't uniform! It's clearly denser near $(0, 20)$ than $(2, 0)$.[3]

2.1 What went wrong?

The problem is that when computing the volume change induced by the change of variables f, we forgot that the base measure for P wasn't Lebesgue on $\mathbb{R}^2$. It's actually Lebesgue along the circle, and f changes arc length differently at different points. Indeed, locally near the point (x, y), the circle is the line in the direction $(-y, x)$. The directional derivative of f is $(-2y, 20x)$, and the change in arclength that f induces is therefore $\sqrt{4y^2 + 400x^2}$. Hence the correct density

[2]Technically, the density function is the Radon–Nikodym derivative of our probability measure with respect to the base measure.

[3]With respect to arc length as the base measure.

is

$$fp_{\text{good}}(x', y') = 1/2\pi\sqrt{y'^2/100 + 100x'^2} \quad \text{when } (x'/2)^2 + (y'/20)^2 = 1, \text{ and } 0 \text{ otherwise.} \tag{2}$$

3 How common is this problem?

While we chose to introduce the base measure problem on a continuous example, it actually occurs most often when transforming discrete distributions embedded in $\mathbb{R}^n$. The density function of, say, the Bernoulli distribution is 1/2 at 0 or 1 and 0 elsewhere, but this is a density with respect to counting measure, not Lebesgue measure on $\mathbb{R}$. Therefore, when transforming this distribution with a bijector, we should not apply any Jacobian correction, because all bijections are counting-volume-preserving.

The same problem also shows up when dealing with distributions on symmetric matrices, such as the Wishart or LKJ distributions. The $k \times k$ symmetric matrices occupy a lower-dimensional sub-manifold of $\mathbb{R}^{k\times k}$, and the density of the Wishart distribution is with respect to Lebesgue measure on that submanifold rather than all of $\mathbb{R}^{k\times k}$. Bijections will in general change the volume under that measure differently than the volume under Lebesgue measure on $\mathbb{R}^{k\times k}$.

Sometimes changes of variables occur in the inference algorithm rather than the model. For example, the Jacobian of the deterministic transformation h in the reversible-jump MCMC framework (Green and Hastie, 2009) is due to interpreting h as a change of variables. One must therefore compute this density correction with respect to the proper base measure whenever jumping between submanifolds of $\mathbb{R}^n$. For example, such jumps would occur in model selection involving a model with a latent von Mises-Fisher-distributed random variable (which is defined on the unit sphere).

The same problem can *also* show up without any changes of variables at all. It suffices to compare the density of different points under the same distribution, or the density of the same point under different distributions. For instance, the Indian GPA problem discussed in e.g. Wu et al. (2018) boils down to treating a density under counting measure as comparable to a density under Lebesgue measure on $\mathbb{R}$, even though the former represents an infinitely larger mass of probability. This incarnation of the base measure problem is more widely studied in the literature, albeit without a name. Our solution smoothly covers this scenario, as we will see in Section 5.3.

4 What's the right answer?

The root of the base measure problem is that we didn't want to compute with measures directly, but lost the base measure when representing probability distributions with densities. It's not actually possible to infer the correct base measure from the data type representing the sample: a point on the unit circle in $\mathbb{R}^2$ is represented with two floating-point numbers, but using Lebesgue measure on $\mathbb{R}^2$ as the base is not helpful.

We propose standardizing on Hausdorff measure as a universal base measure for probability densities. Specifically, for P a distribution on a d-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^n$, let's define the density to be with respect to the d-dimensional Hausdorff measure H^d. We propose d-Hausdorff measure because it provides a coherent definition for "d-dimensional volume" of a surface embedded in $\mathbb{R}^n$, even when the surface is curved.[4] We scale H^d to agree with the standard volume of straight surfaces, and thus with Lebesgue measure when $d = n$. We begin with fixed-dimension distributions for simplicity, and briefly address mixed-dimension distributions in Section 4.2.

4.1 Manifolds of dimension d

What do we need to represent about a sample x besides its density $p(x)$ in order to compute in this scheme? Clearly we need to represent d; but, as the example of the unit circle shows, that's not enough to transform densities correctly. Recall that our scaling transformation had a constant Jacobian when viewed as a function from $\mathbb{R}^2$ to $\mathbb{R}^2$, but we have to apply a position-dependent correction to account for its effect on the unit circle density. The additional thing we need to represent is the tangent space $\mathcal{T}_x$ of $\mathcal{M}$ at the sample point x. Notably, we do not need to computationally represent any non-local information about $\mathcal{M}$ or P—just the tangent space at x, its dimensionality d, and the density $p(x)$.

Let us now formalize what we are actually trying to compute and how we can compute it.

Definition 1. *Given a probability distribution P over any space Ω, and any function $f : \Omega \to \Omega'$, the* push-forward *distribution fP is the measure*

$$fP(S) = P(f^{-1}(S))$$

for all events $S \subset \Omega'$ for which $f^{-1}(S)$ is measurable in Ω.

[4]We propose standardizing at all in order to minimize the amount of information carried by the base measure, so it can be as implicit as possible. The only degree of freedom in Hausdorff measure is the dimension, and that is unavoidable.

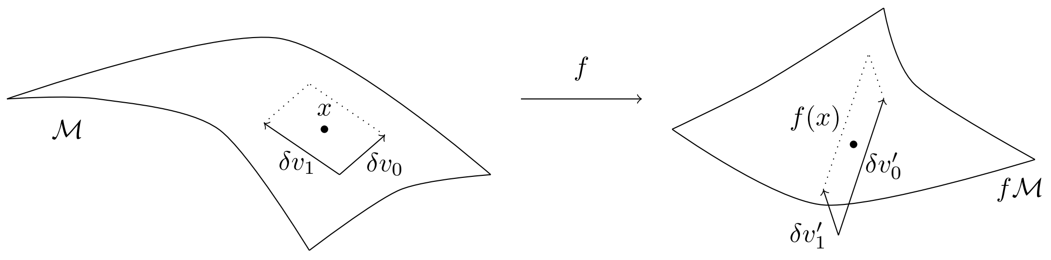

Figure 2: Change of local 2-volume under a transformation from $\mathbb{R}^3$ to $\mathbb{R}^3$. On the left, the small parallelogram with sides δv_0 and δv_1 is tangent to $\mathcal{M}$ at x and forms a volume element. On the right, f takes $\mathcal{M}$ to $f\mathcal{M}$, x to $f(x)$, and the volume element to the small parallelogram tangent to $f\mathcal{M}$ at $f(x)$ with sides $\delta v'_0$ and $\delta v'_1$. When computing the density at $f(x)$ of the pushforward distribution fP supported on $f\mathcal{M}$, we have to correct for the change in volume of this parallelogram, but we have to disregard any stretching or compression f may do in the perpendicular direction. Theorem 1 formalizes this for arbitrary dimensions.

The sampler for fP is just to apply f to a sample from P. We seek a tractable density function for fP, and fortunately the following theorem gives it to us for nice f.

Theorem 1. *Consider*

- *A probability distribution P over a d-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^n$,*
- *with density p with respect to d-dimensional Hausdorff measure H^d;*
- *a diffeomorphism $f : \mathbb{R}^n \to (f\mathbb{R}^n \subseteq \mathbb{R}^m)$; and*
- *a point $x \in \mathcal{M}$.*
- *Let v_i be an arbitrary basis for the tangent space $\mathcal{T}_x$ to $\mathcal{M}$ at x, and pack it into a d-by-n matrix V.*

Then

1. *The pushforward fP is supported on the pushforward manifold $f\mathcal{M}$ in $\mathbb{R}^m$.*
2. *The directional derivatives $v'_i = df(x + \varepsilon v_i)/d\varepsilon$ are a basis for the tangent space $f\mathcal{T}_{f(x)}$ to $f\mathcal{M}$ at $f(x)$. Let us pack them into a d-by-m matrix V'.*
3. *The density fp of fP with respect to H^d at $x' = f(x)$ is*

$$fp(x') = p(x)\frac{\sqrt{\det(VV^T)}}{\sqrt{\det(V'V'^T)}}. \qquad (3)$$

Claims 1 and 2 are standard, and give us the rule for propagating a basis for $\mathcal{T}_x$ to x'. Claim 3 amounts to a restatement in probability terms of standard notions of change of volume. The intuition is that the relevant volume is volume in $\mathcal{T}_x$ (see Figure 2). We choose as a volume element a small parallelepiped: centered at x and with sides δv_i for $\delta \to 0$.[5] The volume of this element is $\delta^d\sqrt{\det(VV^T)}$. The function f transforms this volume element to the parallelepiped with center x' and sides $\delta v'_i$. Therefore, the volume correction we are looking for is $\sqrt{\det(V'V'^T)}/\sqrt{\det(VV^T)}$. We reproduce a formal proof of Claim 3 in Appendix A.

As an aside, Theorem 1 applies without modification to discrete spaces: $d = 0$, 0-Hausdorff measure is, up to scaling, identical with the usual counting measure, and we are free to define all functions of a discrete domain to be vacuously diffeomorphic. The matrices V and V' have zero rows, and we recover the standard result that $fp(x') = p(x)$ for p a probability mass function on a discrete space.

Observe that the inputs required for Theorem 1 are all local: the point x, a basis v_i for $\mathcal{T}_x$, the density $p(x)$ of P at x, and the ability to compute the value and directional derivatives of f at x. We do not need any non-local information about P, $\mathcal{M}$, or f, except that f is invertible. Observe also that the outputs given by Theorem 1 give enough information about fP at $x' = f(x)$ to apply the same theorem again with a new diffeomorphism $g : \mathbb{R}^m \to (g\mathbb{R}^m \subseteq \mathbb{R}^k)$. We invite the interested reader to verify that pushing distributions forward by Theorem 1 commutes with function composition, that is, the double pushforward $g(fP)$ computes the same densities as a single pushforward by the composition $(g \circ f)P$. This equivalence lends itself to modularity in software, permitting, for example, the distribution $g(fP)$ to be represented taking advantage of special structure in f or g, without requiring the user to derive that special structure for $g \circ f$.

[5]Since the v_i are vectors in $\mathcal{T}_x$, it would be more natural to let x be the vertex of the parallelepiped. However, we want to end up with a neighborhood of x, so we shift the parallelepiped to have x in the center.

4.2 Mixed-dimension manifolds

Theorem 1 has given us a formula for transforming densities of probability distributions on d-dimensional manifolds embedded in $\mathbb{R}^n$, and justifies standardizing on d-dimensional Hausdorff measure as a universal base measure for densities of such distributions.

The results generalize directly to finite unions of manifolds, of potentially different dimensions. Namely, let P be a sum of measures P_j on manifolds $\mathcal{M}_j$ of dimensions d_j, where without loss of generality we require that the intersection of any $\mathcal{M}_j$ and $\mathcal{M}_k$ have strictly lower dimension than at least one of d_j and d_k. In this situation, we use the sum of the corresponding Hausdorff measures H^{d_j} on those manifolds as the base measure. Then the density $p(x)$ representing P is given by the density $p_j(x)$ of the P_j corresponding to the lowest-dimensional manifold $\mathcal{M}_j$ that x is on. If in addition to $\mathcal{M}_j$, x is on manifolds of higher dimension, any densities $p_{j'}$ corresponding to measures on those higher-dimensional manifolds become zero with respect to H^{d_j}.

If x occurs on multiple support manifolds of minimal dimension, then by assumption x is part of a lower-dimensional intersection, whose total mass under P is necessarily zero. We are therefore formally free to assign whatever probability density we wish to x. We conjecture that reporting the sum $\sum_{\{j|d_j \text{minimal}\}} P_j(x)$ will follow the principle of least surprise, but are prepared to be corrected by future experience in probabilistic programming.

4.3 Scalar special case

It is worth explicitly drawing the link to the case of scalar-valued probability distributions. Here we meet the fortuitous accident that $\mathbb{R}^1$ has only two subspaces. The tangent space of any manifold embedded in $\mathbb{R}^1$ is therefore either all of $\mathbb{R}^1$, or the zero vector space $\mathbb{R}^0$. The reader of course recognizes the familiar distinction between continuous and discrete probability distributions.

In the special case of scalar distributions, $\mathcal{T}_x$ carries no information besides its dimension (which is 0 or 1). The base measure problem therefore reduces to keeping track of said dimension. Jacobs (2021) provides an elegant way to do this with arithmetic on formal infinitesimals.

If we restrict our modeling language to sampling and observing only scalar valued distributions, and to transforming them with only unary scalar functions, we are effectively forcing the support manifold $\mathcal{M}$ to be axis-aligned. In this case, infinitesimal arithmetic on the probabilities carries all the information we need, and therefore suffices to compute the correct answers. The actual basis of $\mathcal{T}_x$ is only needed in the presence of changes of variables that act jointly on multiple coordinates of $\mathbb{R}^n$.

Said tangent space basis is also only needed for computing probability densities correctly. Once the densities have been computed, the only information we still need in order to combine and compare them is indeed the dimension of $\mathcal{T}_x$, which we can profitably package together with the density as a formal infinitesimal. We work out the consequences of this approach on SMC and MCMC in Section 5.3.

5 How do we fix the software?

We discuss how to adjust current probabilistic programming software to fix the base measure problem. The largest change is needed to enable correct changes of variables (Section 5.1); given that, updating common inference algorithms is a local adjustment, which we explicate for completeness in Section 5.3.

5.1 Correct changes of variables

Correctly updating densities under changes of variables requires two new architectural features of probabilistic programming software: First, querying a probability distribution P at a point x must produce a local measure, which includes the density $p(x)$, the local dimension d_x, and a representation of the tangent space $\mathcal{T}_x$ at x to the support manifold $\mathcal{M}$. Second, changes of variable induced by deterministic transformations must take the tangent space into account. The current standard architecture can be seen as the specialization of the above to the case when the dimension d is fixed, and the support manifold $\mathcal{M}$ is the entire host space $\mathbb{R}^n$.

To elaborate, we know from Section 4.1 that to compute the density of a pushforward fP at a point x' with respect to d-Hausdorff measure it suffices to compute

- The preimage point $x = f^{-1}(x')$;
- The density $p(x)$ of P at x;
- A basis v_i for the tangent space to the support $\mathcal{M}$ of P at x;
- The directional derivatives v'_i of f at x in directions v_i, forming a basis for the tangent space at x' to the support of fP; and
- The correction term $\sqrt{\det(VV^T)}/\sqrt{\det(V'V'^T)}$ from (3).
- For composition, we also need to return the pushforward tangent basis v'_i.

At first glance this may seem prohibitively expensive: even if we have an automatic differentiation system ready to hand that lets us compute the needful directional derivatives of f, do we really have to do that, and then perform two full matrix multiplies and determinants, for every probability density evaluation?

Fortunately, there are several special cases we can take advantage of to save work, and to recover the performance of the conventional architecture when it gives the correct answer:

- If our distribution P is actually discrete, the tangent space is 0-dimensional, VV^T is the 0x0 matrix, and its determinant is vacuously 1.
- If P is supported on all of $\mathbb{R}^n$, the tangent space is also $\mathbb{R}^n$, and we may take v_i to be the standard basis, so the $\sqrt{\det(VV^T)}$ term is unity and we need not explicitly compute it.
- If in addition f maps $\mathbb{R}^n$ to $\mathbb{R}^n$ and not to $\mathbb{R}^m$ for $m > n$, then the matrix V' is the Jacobian of f at x and itself has a determinant. We can save a matrix multiply and a square root because in this case $\sqrt{\det(V'V'^T)} = |\det V'|$. This is where we recover the conventional Jacobian-determinant correction to probability densities.
- Distributions on structured-sparse matrices, such as lower-triangular matrices, are supported on manifolds with axis-aligned tangent spaces. In such a case, we need only keep track of a mask representing the present dimensions, taking the basis v_i to be a subset of the standard basis. The $\sqrt{\det(VV^T)}$ term again disappears, though if f does not preserve the sparsity, the $\sqrt{\det(V'V'^T)}$ term may need to be computed.
- If in addition f is a univariate transformation applied coordinate-wise, the sparse structure will be preserved, and the correction will just be the product of partial derivatives of f, $\sqrt{\det(V'V'^T)} = \prod_i |\frac{\partial}{\partial x_i} f(x)|$, with i here ranging over the dimensions present in the manifold but not the others.
- Finally, simplices and symmetric matrices may also merit special treatment, because the basis v_i can again be implicit.

This list of special cases suggests ad-hoc polymorphism as a representation strategy. Specifically, a typical probabilistic programming system probably already has a `Bijector` class (hierarchy) for representing transformations f. To implement this strategy for volume corrections, we can add a `TangentSpace` class hierarchy for the tangent space to $\mathcal{M}$ at x. This second hierarchy can have dedicated classes for efficient cases, such as the zero space tangent to a discrete support, the full space tangent to an n-dimensional support in $\mathbb{R}^n$, and so on. To take advantage of special cases based both on the tangent space and the bijector, we can define the density correction function with two-argument dispatch. In languages like Python where only single-argument dispatch is available, we can emulate two-argument dispatch with the Visitor pattern (Gamma et al., 1995). We report on the code complexity of a minimal test of this approach in Section 6.

When working with distributions on high-dimensional spaces, or with large batches of distributions, we expect the cost of the dispatch to be small compared with the cost of the linear algebra the dispatch avoids. Further, on a tracing platform like JAX (Bradbury et al., 2019), TensorFlow (Abadi et al., 2015), or TorchScript (PyTorch Contributors, 2018), the dispatch only occurs once during tracing, avoiding repeated linear algebra during execution.

5.2 Formal infinitesimals

Given a probability distribution P and point x, if we are going to use the density $p(x)$ for any purpose other than computing a pushforward density fp at a point $f(x)$, we no longer need the tangent space $\mathcal{T}_x$—only its dimension d. The semantics of this pair of density and dimension are exactly those of a formal infinitesimal. We recapitulate the notation and basic properties of formal infinitesimals in this section, following the treatment in Jacobs (2021). This serves as preparation for fixing the base measure problem for SMC and MCMC in Section 5.3.

Definition 2 (Jacobs (2021))**.** *A formal infinitesimal is a pair $z = (r, n)$ of a real number r and an integer n, which we write as $r\varepsilon^n$.*

The infinitesimal $z = r\varepsilon^n$ represents the statement

$$\lim_{\varepsilon\to 0} \frac{z}{\varepsilon^n} = r.$$

Arithemtic on formal infinitesimals is defined as

$$r\varepsilon^n \pm s\varepsilon^m = \begin{cases} (r \pm s)\varepsilon^n & \text{when } n = m \\ r\varepsilon^n & \text{when } n < m \\ \pm s\varepsilon^m & \text{when } n > m. \end{cases}$$

$$(r\varepsilon^n)\cdot(s\varepsilon^m) = (r\cdot s)\varepsilon^{n+m}$$

$$(r\varepsilon^n)/(s\varepsilon^m) = \begin{cases} (r/s)\varepsilon^{n-m} & \text{when } s \neq 0 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We will use the theorem

Theorem 2 (Jacobs (2021))**.** *If $f : \mathbb{R}^n \to \mathbb{R}$, and f evaluates successfully on infinitesimal arguments using the above rules as*

$$f(r_1\varepsilon^{k_1}, \dots r_n\varepsilon^{k_n}) = r\varepsilon^k,$$

then (in ordinary arithmetic)

$$\lim_{x\to 0} \frac{f(r_1 x^{k_1}, \dots r_n x^{k_n})}{x^k} = r.$$

As a reminder, the results produced by this theorem are not always the tightest possible. For instance, for g written as $g(x) = (x + x^2) - x$, the above rules of evaluation will give $\lim_{x\to 0} g(x)/x = 0$. This statement is true, but if we had rewritten g as $g(x) = x^2$, we could have obtained the stronger statement $\lim_{x\to 0} g(x)/x^2 = 1$.

Remark: When r is finite and non-zero, the infinitesimal $z = r\varepsilon^n$ is complete, in the sense that it determines all one-term infinitesimal descriptions of z for all degrees n. If, however, we should encounter an infinitesimal $r\varepsilon^n$ with $r = 0$ or $r = \pm\infty$, the same quantity z may admit a better infinitesimal description, with $n' > n$ or $n' < n$, respectively.

5.3 Measure-aware inference algorithms

Once a probabilistic programming system correctly propagates local measures per Section 5.1, defending standard inference algorithms against the base measure problem is just a matter of representing probabilities as formal infinitesimals, per Section 5.2. We note again that just the density and local dimension suffice to form these infinitesimals; the tangent basis is only used for changing variables.

Algorithm 1 Measure-aware resampling for SMC

```
Input: Particles $p_j$, infinitesimal weights $w_j\varepsilon^{d_j}$
Input: Desired number of new particles $N$.
Compute $d = \min d_j$.
if All $w_j = 0$ where $d_j = d$ then
    Error: The resampling is undefined.
end if
if Any $w_j = \pm\infty$ where $d_j > d$ then
    Error: The resampling is undefined.
end if
for $i = 1$ to $N$ do
    Sample index $k_i \in \{j | d_j = d\}$ proportional to $w_j$.
    Output $p_{k_i}$.
end for
Return Infinitesimal self-normalization estimate $(\sum_{j|d_j=d} w_j)\varepsilon^d$.
```

We spell out two standard inference algorithms explicitly. To resample in SMC, just throw out particles of non-minimally-dimensioned weights, as in Algorithm 1. In the one-stage case this reduces to the dimensionality-aware likelihood weighting algorithm of Wu et al. (2018).

To compute the Metropolis-Hastings acceptance or rejection for MCMC, just compare the target density dimension-major, as in Algorithm 2. Note that the proposed state y must live on a same-dimension support manifold of q as the current state x, otherwise the proposal q is not reversible. This can be arranged with the usual reversible-jump MCMC technique from Green and Hastie (2009). However, for general probabilistic programs π, x and y may yield differently-dimensioned target probabilities, and M-H must account for this.

Algorithm 2 Measure-aware Metropolis-Hastings

```
Input: Target $\pi(\cdot)$, proposal $q(\cdot|\cdot)$, current state $x$
Sample proposed state $y \sim q(\cdot|x)$.
Evaluate infinitesimal probabilities
$p(x)\varepsilon^{d_x} = \pi(x)$
$p(y)\varepsilon^{d_y} = \pi(y)$
if $d_y > d_x$ then
    if $p(x) \neq 0$ and $p(y) < +\infty$ then
        Reject: $p(x)\varepsilon^{d_x}$ dominates $p(y)\varepsilon^{d_y}$.
    else
        Error: The comparison is undefined.
    end if
end if
if $d_y < d_x$ then
    if $p(y) \neq 0$ and $p(x) < +\infty$ then
        Accept: $p(y)\varepsilon^{d_y}$ dominates $p(x)\varepsilon^{d_x}$.
    else
        Error: The comparison is undefined.
    end if
end if
Sample $u$ uniformly between 0 and 1.
Accept if $u < \frac{p(y)q(x|y)}{p(x)q(y|x)}$, else Reject.
```

For example, in the much-discussed Indian GPA problem (see Wu et al. (2018) for one restatement), we might do MCMC on a boolean latent variable modeling whether a student is American or Indian. The infinitesimal likelihood of the observed GPA of 4.0 for an American student would be something like $0.1\varepsilon^0$, indicating a point-mass of size 0.1 at the maximum GPA that American high schools report. Similarly, the infinitesimal likelihood of a 4.0 GPA for an Indian student would be $0.1\varepsilon^1$, being the density of a scalar continuous uniform distribution of GPAs between 0 and 10. Algorithm 2 would then correctly accept every proposal to change the latent from “Indian” to “American”, and reject every proposal to change back.

Entity changed	LoC
New support classes (reusable)	44
`Scale` bijector	6
`SphericalUniform` distribution	8
`TransformedDistribution` class	15
Total	73

Table 1: Size of code changes to TensorFlow Probability to fix the base measure problem on the ellipse example. The described patch touches only the parts of TFP that need to change to support specifically the ellipse, but we tried to do it in a code style that would be representative of the effort needed to adopt the fix across all TFP Distributions and Bijectors.

6 Validation

To validate our hypothesis that our proposed changes do not entail unmanageable code complexity, we patched the TensorFlow Probability (TFP) library (The TFP Team, 2019) to correctly compute our example ellipse density from Section 2. The distribution on the ellipse can be represented in TFP as

```
ellipse = TransformedDistribution(
  bijector=Scale([2., 20.]),
  distribution=SphericalUniform(dimension=2))
```

With modest-size (see Table 1) adjustments to these pieces, we were able to make them adhere to our protocol from Section 5.1. These changes cause TFP to compute the correct density on this example, as shown in Figure 3. We emphasize that we are only patching the primitive distributions and bijectors available in TFP, without changing the library's architecture, or the patterns by which its parts may be composed. In particular, all pre-existing programs that use TFP without extending it will continue to work.

We also emphasize that we followed the Visitor pattern (Gamma et al., 1995) in this patch, and it allowed us to take advantage of the available special structure, without hardcoding it in any place it does not belong. Namely, the `Scale` bijector acts independently on each coordinate, and the diagonal of its Jacobian is just its `scale` parameter. Thus our patched TFP is maximally efficient on this example: it does not need to invoke any automatic differentiation system or do any symbolic algebra, but just directly computes the formula (2). Nevertheless, `TransformedDistribution` just invokes (overridable) methods on `Bijector` and `TangentSpace` instances, and needs no custom logic for coordinatewise bijectors.

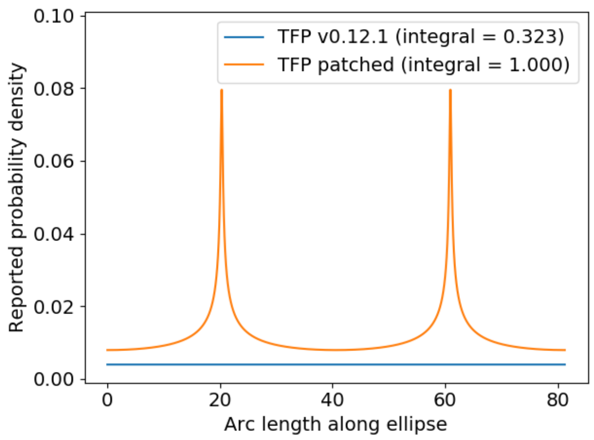

Figure 3: The ellipse density from Section 2 in practice. The flat blue line is the erroneous result computed by TensorFlow Probability (version 0.12.1), a widely used library for probabilistic computing. The orange line showing two maxima is the correct result, computed by TensorFlow Probability after we patched it as described in the main text.

7 Related Work

Broadly, all general-purpose probabilistic programming systems are related to the present work, in that they must either produce incorrect results when the base measure problem arises (Warfield, 2020; The Venture Team, 2017), avoid the problem entirely, or somehow solve it. Three specific systems are worth mentioning:

Stan (Carpenter et al., 2017) is an example of a system that avoids the base measure problem. There are two places in a Stan model where the base measure problem might occur: in the automatic unconstraining transformations and in applying densities to transformed parameters. The former Stan can get right because both the base measure and unconstraining transformation are implied by the type of the parameter being unconstrained. The base measure problem occurs, in a sense, but the remedy is both local and internal to the implementation of Stan. The latter Stan pushes on the user: if a user writes a Stan model with a parameter x and a transformed parameter $f(x)$, and wishes to code a density on $f(x)$ that corresponds to the pushforward of some known density $p(x)$, then that user must think about base measures and partial derivatives themselves. Stan does not help; but then again, it also does not claim it would help, so Stan doesn't compute anything visibly "incorrect".

BLOG discusses addressing the base measure problem (not by that name) in Wu et al. (2018). Relative to the present work, that treatment is simpler in two ways: they only discuss scalar random variables, where every tangent space is either $\mathbb{R}$ or the zero vector space; and they do not explicitly discuss transformations of random variables. See Section 4.3 for a more elaborate statement of the relationship of scalar-oriented strategies with the present work.

Jacobs (2021) clarifies the scalar-wise treatment by using formal infinitesimals, which notation we adopt in Section 5.2, and also handles conditioning on (unary) transformations of scalar random variables. Our contributions can be viewed as a concurrently-developed extension of the system of Jacobs (2021) to vector-valued distributions.

Some cases of the base measure problem are addressed by Hakaru's disintegration transform (Narayanan and Shan, 2020), which generalizes density and also needs correct base measures. Hakaru defines a restricted language of base measures with respect to which it can symbolically compute (unnormalized) disintegrations of s-finite measures. As this restricted set includes discrete-continuous mixtures, it is sufficient to correctly handle the Indian GPA problem; but for the present example of the unit circle, it would need a richer notion of constraints and their tangent or normal spaces (Narayanan, 2020).

For example, the following model of the ellipse example from Section 2 is expressible in the system from (Narayanan and Shan, 2020) but gives an incorrect disintegration:

```
x ~ lebesgue
y ~ lebesgue
observe x**2 + y**2 = 1
return (2*x, 20*y)
```

The issue is that the constraint restricts the distribution to a lower-dimensional manifold $\mathcal{M}$, and the tangent space to that manifold is not taken into account when applying the Jacobian correction due to the transformation $f(x, y) = (2x, 20y)$. This tangent space could be recovered, since it is perpendicular to the gradient of the constraint, so we hope that this manifestation of the base measure problem will be relatively easy to fix in Hakaru.

There is also a line of work on formal semantics of probabilistic programming languages, for example Borgström et al. (2013) and Staton et al. (2016). That work taken on its own terms is generally not vulnerable to the base measure problem at all, because the semantics are invariably given in terms of measures, not densities. The present work can be viewed as a bridge to implementation, providing a complete and efficient local representation of a measure.

8 Future Work

A common thread in the above related work is to address base measures dimension-wise. In other words, reduce to the scalar-oriented setting and track one "discrete or $\mathbb{R}$" bit per scalar random variable, as discussed in Section 4.3. This is appealing because no explicit tangent spaces are needed, since they are all either trivial or $\mathbb{R}$. A vector-valued distribution P on a d-dimensional manifold embedded in $\mathbb{R}^n$ can then be modeled as n deterministic scalar functions f_i from d scalar random variables. Computing the density of P involves constraining the outputs of the f_i one by one, with a density correction given by the gradient of f_i as appropriate. After d irredundant constraints, the remaining distribution is discrete, and no further derivatives are needed.

It would be interesting to work out whether the preceding strategy actually works. We predict two difficulties. First, the f_i are in general going to be multiary functions of their scalar inputs. Does a coherent scheme exist for choosing which scalar input(s) to constrain when constraining the output of each f_i? Do the answers depend on this choice? Once some inputs have already been constrained in order to satisfy an output constraint on f_i, will that interfere with constraining the output of a different f_j that consumes some of the same inputs?

Second, even if a joint function f is invertible as a vector-valued function, its coordinates f_i may not be invertible individually. In this case, treating them separately seems to require explicitly manipulating their preimages, which may be non-trivial discrete sets. For example, if we represent a distribution on the unit circle as a distribution on $[0, 2\pi)$ followed by the functions $(\sin\cdot, \cos\cdot)$ then the preimage of sin does not actually give us a unique point at which to evaluate its derivative; whereas the preimage of cos, while no longer contributing a derivative per se, is still needed to identify the correct branch of sin.

9 Conclusion

We have named the *Base Measure Problem* and provided a solution to it. Implementing the solution in probabilistic programming systems should cause negligible loss of performance for cases that were already correctly handled, and expand the set of models in which the system can compute correct probability densities. Implementation does carry a code complexity cost, but that cost is minimized by using two-argument dispatch, or emulating it with a Visitor pattern. Despite correctly accounting for measures, no non-local information is required.

10 Acknowledgments

The authors thank Srinivas Vasudevan, Wynn Vonnegut, Praveen Narayanan, and Dave Moore for enlightening discussion. The authors also thank the anonymous reviewers for valuable constructive criticism.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Borgström, J., Gordon, A., Greenberg, M., Margetson, J., and Van Gael, J. (2013). Measure transformer semantics for bayesian machine learning. *Logical Methods in Computer Science*, 9(3).

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M., Leary, C., Maclaurin, D., and Wanderman-Milne, S. (2017–2019). JAX. Specifically the vmap functionality.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1).

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley.

Green, P. J. and Hastie, D. I. (2009). Reversible jump mcmc. *Genetics*, 155(3):1391–1403.

Jacobs, J. (2021). Paradoxes of probabilistic programming: and how to condition on events of measure zero with infinitesimal probabilities. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–26.

Li, L. and Russell, S. J. (2013). The blog language reference. *tech. rep., Technical Report UCB/EECS-2013-51*.

Mansinghka, V., Selsam, D., and Perov, Y. (2014). Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*.

Narayanan, P. (2020). Personal communication.

Narayanan, P. and Shan, C.-c. (2020). Symbolic disintegration with a variety of base measures. *ACM Trans. Program. Lang. Syst.*, 42(2).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037.

PyTorch Contributors (2018). Torch script.

Staton, S., Yang, H., Wood, F., Heunen, C., and Kammar, O. (2016). Semantics for probabilistic programming. *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science.*

The TFP Team (2018–2019). TensorFlow Probability.

The Venture Team (2017). Venture suffers from the base measure problem. Personal communication.

Tolpin, D., van de Meent, J.-W., Yang, H., and Wood, F. (2016). Design and implementation of probabilistic programming language anglican. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages*, IFL 2016, New York, NY, USA. Association for Computing Machinery.

Warfield, R. (2020). Tensorflow probability issue #761: Incorrect probabilities after transforming a quantizeddistribution.

Wu, Y., Srivastava, S., Hay, N., Du, S. S., and Russell, S. (2018). Discrete-continuous mixtures in probabilistic programming: Generalized semantics and inference algorithms. In *International Conference on Machine Learning.*

A Proof of Theorem 1, Claim 3

To prove Theorem 1, Claim 3 formally, we go through the measures P and fP, starting with the following lemma. The only technical trick is to enlarge our parallelepipeds to open sets in $\mathbb{R}^n$ and $\mathbb{R}^m$, so that measuring them with P and fP captures all the mass near x and x', respectively, despite any curvature of $\mathcal{M}$ or $f\mathcal{M}$. Then we will note that in the limit the only thing we care about is the projections of those open sets back to the respective tangent spaces, which are the parallelepipeds we started with.

Lemma 1. *Let P, $\mathcal{M}$, p, x, v_i, and V be as in the statement of Theorem 1. Let G_V be the d-parallelepiped with center x and sides v_i, and let G_{V+} be any n-parallelepiped centered at x with section G_V. Let G_V^δ and G_{V+}^δ be G_V and G_{V+}, respectively, scaled* *by δ about x. Then the density p of P with respect to d-dimensional Hausdorff measure is given by*

$$p(x) = \frac{1}{\sqrt{\det(VV^T)}} \lim_{\delta \to 0} P(G^\delta_{V+})/\delta^d.$$

Proof of lemma. The definition of p being a density for P with respect to H^d is that for all measurable subsets $B \subset \mathbb{R}^n$,

$$P(B) = \int_B p \, dH^d.$$

P is only supported on $\mathcal{M}$, so we may take the integral on the right to be over the intersection $B \cap \mathcal{M}$. We choose for B the neighborhood G^δ_{V+}.

As G^δ_{V+} becomes sufficiently small, we may assume p is constant on $G^\delta_{V+} \cap \mathcal{M}$, so

$$P(G^\delta_{V+}) \approx p(x) H^d(G^\delta_{V+} \cap \mathcal{M}).$$

As $\delta \to 0$, $\mathcal{M}$ near x approaches its tangent space, so $G^\delta_{V+} \cap \mathcal{M}$ becomes G^δ_V. The right-hand term becomes the d-dimensional Hausdorff measure of G^δ_V, which given our choice of scaling for the Hausdorff measure is just the volume thereof, namely $\delta^d \sqrt{\det(VV^T)}$. □

Proof of Claim 3. Let $G_{V'}$ be the d-parallelepiped in $\mathbb{R}^m$ with center x' and sides v'_i. Let $G_{V'+}$ be any m-parallelepiped in $\mathbb{R}^m$ with section $G_{V'}$ centered at x'. Let $G^\delta_{V'+}$ be $G_{V'+}$ scaled by δ about x'. By the lemma, we have

$$fp(x') = \frac{1}{\sqrt{\det(V'V'^T)}} \lim_{\delta \to 0} fP(G^\delta_{V'+})/\delta^d.$$

The preimage $f^{-1}(G^\delta_{V'+})$ approaches an n-parallelepiped centered at x with section G^δ_V. This is because the basis vectors v'_i forming V' are the directional derivatives of f in the directions given by the basis vectors v_i forming V. In directions normal to $\mathcal{M}$, we just need the derivatives of f to be finite. We may thus apply the lemma again to write

$$\begin{aligned}\lim_{\delta \to 0} fP(G^\delta_{V'+})/\delta^d &= \lim_{\delta \to 0} P(f^{-1}(G^\delta_{V'+}))/\delta^d \\ &= \sqrt{\det(VV^T)} p(x),\end{aligned}$$

giving the desired result. □