Probability Density

Dan Piponi

June 2024

1 Introduction

It is common to represent a probability distribution with a type or class.

```
{-# LANGUAGE MultiParamTypeClasses #-}

class Distribution x p where
  prob :: p → x → Float

data Uniform = U {lo :: Float, hi :: Float}

instance Distribution Float Uniform where
  prob (U a b) x | x < a      = 0
                 | x ⩾ b      = 0
                 | otherwise = 1 / (b - a)
```

Let's specialise to the unit interval:

```
u = U 0 1
```

So what is the meaning of the return value from `prob`? If we were working with discrete probability distributions then we could simply say that it returns a probability. But even though I chose the name `prob` to be fairly conventional (cf. TensorFlow Probability) it is not a probability in the non-discrete case.

```
x = 1 / 3 :: Float
```

We can get a probability straightforwardly if we talk about infinitesimals:

$$P(x \in [x, x+\epsilon)) = 0 + (\texttt{prob u } x)\epsilon$$

where ϵ is an infinitesimal. In other words the probability density function gives the probability that a sample lies in an infinitesimal interval.

Suppose we wish to change coordinates on our unit interval, eg. working with $y = x^2$ instead of x.

$$y = x * x :: \mathit{Float}$$

Let $y = \frac{1}{9}$ represents the same point as $x = \frac{1}{3}$. We find that `prob u (sqrt y)` does not give the probability of y lying in the interval $[\frac{1}{9}, \frac{1}{9} + \epsilon)$. In our definition of `u` was an implicit choice of coordinates that isn't stated explicitly. The traditional description of this failure is that we failed to take into account the Jacobian of the coordinate change.

Let's take that infinitesimal interval idea seriosuly. Define infinitesimals (as in automatic differentiation) in the usual way:

```
data D a = D {real :: a, infinitesimal :: a} deriving Show
lift x = D x 0

instance Num a => Num (D a) where
  D a a' + D b b' = D (a + b) (a' + b')
  D a a' * D b b' = D (a * b) (a * b' + a' * b)

instance Fractional a => Fractional (D a) where
  recip (D a a') = let r = 1 / a in D r (-r * r)

instance (Fractional a, Floating a) => Floating (D a) where
  sqrt (D a a') = let s = sqrt a in D s (1 / (2 * s))
```

Now we use a new definition of `prob` where we explicitly pass in an infinitesimal interval $[x, x + a\epsilon)$ as `D x a`.

```
class Distribution x p where
  prob :: p -> D x -> D Float
```

```
instance Distribution Float Uniform where
  prob (U a b) (D x dx) | x < a      = 0
                        | x >= b     = 0
                        | otherwise = D 0 (dx / (b - a))

x' = D x 1 :: D Float
```

Now, for example, `prob u x'` has the interpretation that the probability of x lying in $[x, x+\epsilon)$ is ϵ.

Now define

```
y' = D (x * x) 1
```

With the previous definitions `prob u (sqrt y)` didn't compute what we intended. Now we get `prob u (sqrt y')` $= \frac{3}{2}\epsilon$. It is correctly telling us that $P(y \in [\frac{1}{9}, \frac{1}{9}+\epsilon))$ is $\frac{3}{2}\epsilon$ (which you can compute yourself using the standard formula in probability theory for changing coordinates).

You can achieve the same effect using bijectors in TensorFlow Probability, and it'll perform very similar steps when it uses AD to compute the Jacobian. But curiously it just sort of came "for free" in this code as a result of us being honest about what `prob` computes: the infinitesimal probability of a quantity lying in an infinitesimal interval.

```
main = do
  print (prob u (sqrt y'))
```