lotus-illustrative

November 13, 2019

0.0.1 An illustrative example of invoking LOTUS as we done in (5) with a simple simulation check

We claim that the following two expressions, E_1 and E_2, are equvialent as the second expression can be obtained via invoking LOTUS to the first expression.

$$E_1 = \int p_x(x) g(f(x)) dx = \mathbb{E}_{x \sim p_x}[g(f(x))] \approx \frac{1}{N} \sum_{i=1}^{N} g(f(x_i)) \text{ where } x_i \sim p_x$$

and

$$E_2 = \int p_y(y) g(y) dy = \mathbb{E}_{y \sim p_y}[g(y)] \approx \frac{1}{N} \sum_{i=1}^{N} g(y_i) \text{ where } y_i \sim p_y,$$

where $y = f(x)$ and $g(y)$ is an arbitary function that ensure the integral is finite.

For simulation purpose, consider a 3-dimensional Gaussian $p_x(x) = \mathcal{N}(x; [0, 2, 1], [1, 4, 1])$ and a projection function f that project x to y as $y = f(x) = [3x_1 + 2x_2, 3x_3]$.

[1]:
```julia
using Distributions

p_x = MvNormal([0.0, 1.0, 1.0], sqrt.([1.0, 4.0, 1.0]))

f(x) = [3x[1] + 2x[2], 3x[3]];
```

As $x_1 \sim \mathcal{N}(0, 1)$, $x_2 \sim \mathcal{N}(1, 4)$ and $x_3 \sim \mathcal{N}(0, 1)$, y_1 is a sum of two scaled Gaussian distributed random variables and y_2 is a scaled Gaussian random variate. It's not hard to see that $y \sim \mathcal{N}([2, 3], [25, 9])$.

[2]:
```julia
p_y = MvNormal([2.0, 3.0], sqrt.([25.0, 9.0]));
```

In our simulation, we choose $g(y) = \left(\frac{\mathcal{N}(y;[1,1],[4,4])}{\mathcal{N}(y;[2,2],[25,25])}\right)^2$, a squared density ratio, to mimic (5).

[3]:
```julia
g(y) = (pdf(MvNormal([1.0, 1.0], sqrt.([4.0, 4.0])), y) / pdf(MvNormal([2.0, 2.
↪0], sqrt.([25.0, 25.0])), y))^2;
```

We then run simulation to check if the MC estimation of $E_1 = \int \mathcal{N}(x; [0, 1], [1, 4]) g(f(x)) dx$ and $E_2 = \int \mathcal{N}(y; 2, 25) g(y) dy$ are consistent via their MC estimates.

[4]:
```
n_mc = 1_000_000

Ê  = mean([g(f(rand(p_x))) for _ in 1:n_mc])
Ê  = mean([g(rand(p_y))    for _ in 1:n_mc])

@info "Equivalence checking" Ê  Ê  abs(Ê  - Ê );
```

```
Info: Equivalence checking
  Ê  = 4.563892207100851
  Ê  = 4.562799846529573
  abs(Ê  - Ê ) = 0.0010923605712784834
@ Main In[4]:6
```

Note $g(y)$ doesn't have to be a squared density ratio. E.g. $g(y) = \tanh(y_1 * y_2 - 0.5)$ is also applicable.

[5]:
```
g(y) = tanh(prod(y) - 0.5)

Ê  = mean([g(f(rand(p_x))) for _ in 1:n_mc])
Ê  = mean([g(rand(p_y))    for _ in 1:n_mc])

@info "Equivalence checking" Ê  Ê  abs(Ê  - Ê );
```

```
Info: Equivalence checking
  Ê  = 0.16186534795276478
  Ê  = 0.160809966192148
  abs(Ê  - Ê ) = 0.001055381760616786
@ Main In[5]:6
```

[ ]:
```

```