Reviewing Expected Value

In this lesson, we are going to do the derivation all over again.

WE'LL COVER THE FOLLOWING

In the previous lesson, we deduced the idea behind the "importance sampling" technique for determining the average value of a function from `double` to `double` — call it `f` — when it is applied to samples from a possibly-non-normalized weighted distribution of doubles — call it `p`.

Revision

In this lesson, we are going to do revise the concepts we learned in the previous lesson. We'll again be using the technique "things equal to the same are equal to each other", but this time we're going to start from the other end. Let's jump right in!

Again, for pedagogical purposes, we are going to consider only distributions with support from 0.0 to 1.0; we'll eliminate that restriction when we can.

We discovered a while back that the expected value of `f` applied to samples of `p` is equal to the quotient of `Area(x => f(x) * p.Weight(x)` divided by `Area(x => p.Weight(x))`. The latter term is the normalization constant for `p`, which we might not know.

Let's take any weighted distribution `q` also with support from 0.0 to 1.0; it also might not be normalized.

We're now going to do some manipulations to our expression that are

identities. We'll start with the fact that

```
Area(x => f(x) * p.Weight(x))
```

obviously must be equal to:

```
Area(x => (f(x) * p.Weight(x) / q.Weight(x)) * q.Weight(x))
```

We've just divided and multiplied by the same quantity, so that is no change. And we've assumed that `q` has the same support as `p`, so the weight we're dividing by is always non-zero.

Similarly,

```
Area(p.Weight)
```

must be equal to

```
(Area(q.Weight) * (Area(p.Weight) / Area(q.Weight)))
```

for the same reason.

So the quotient of these two quantities must also be equal to the expected value of `f` applied to samples from `p`; they're the same quantities! Our original expression

```
Area(x => f(x) * p.Weight(x) /
 Area(x => p.Weight(x))
```

is equal to:

```
Area(x => (f(x) * p.Weight(x) / q.Weight(x)) * q.Weight(x)) /
  (Area(q.Weight) * (Area(p.Weight) / Area(q.Weight)))
```

For any suitable `q`.

Let's call that value `exp_fp` for "expected value of `f` on samples of `p`". We've just written that value in two ways, one very straightforward, and one excessively complicated.

Unsurprisingly, our next question is: what is the expected value of function `g`

```
x => f(x) * p.Weight(x) / q.Weight(x)
```

over samples from `q`?

We know that it is estimated by this quotient of areas:

```
Area(x => g(x) * q.Weight(x)) /
Area(q.Weight)
```

The denominator is the normalization constant of `q` which we might not know.

Call that value `exp_gq`, for "expected value of `g` on samples of `q`".

What is the Relationship Between `exp_gq` and `exp_fp`?

Well, just look at the two expressions carefully; they differ by no more than a constant factor. `exp_fp` is equal to `exp_gq * Area(q.Weight) / Area(p.Weight)`.

And now we are right where we ended last time.

Summing Up

- We have deduced that importance sampling works:
    - **An estimate of the expected value of `g` applied to samples from `q` is proportional to the expected value of `f` applied to samples from p.**
    - The proportionality constant is exactly the quotient of the normalization constants of `q` and `p`.
    - If `q` and `p` are known to be normalized, then that constant is 1.
- We can extend this result to `q` and `p` with any support.
    - All we need for an accurate estimate is that `q` has support in places where `f(x) * p.Weight(x)` has a lot of area.
    - But it is also nice if `q` has a low weight in where that function has a small area.

- We have two serious problems:
    - How do we find a good q?
    - We are required to know the normalization constants of both p and q, either of which might be hard to compute in general.
- Once again, the previous statement contains a subtle error.
    - Can you figure it out?

Show Hint

In the next lesson, we'll implement a version of this algorithm entirely based on sampling; we do not actually need to do the quadrature to compute the normalization factors!