Stats 102C - Lecture 2-3: Importance Sampling

Miles Chen, acknowledgements: Michael Tsiang

Week 2 Friday

Section 1

Importance Sampling

Monte Carlo Integration

We can use Monte Carlo integration to estimate the following expectation:

$$\mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x) f(x) dx \approx \frac{1}{n} \sum_{j=1}^{n} h(x_j) = \bar{h}_n$$

The calculation of $\bar{h}_n$ depends on our ability to produce random samples from the PDF $f(x)$.

What can be done if we are not able to sample from $f(x)$ directly?

Alternatives

Currently, we are generating random values by using R's built-in distribution functions.

We can generate random values from many probability distributions: uniform, normal, beta, gamma, exponential, chi-squared, f, poisson, binomial, etc.

In some cases, we might not have a built-in random distribution function.

What can be done if we are not able to sample from $f(x)$ directly?

Option 1) We can discuss strategies for generating random values from different distributions. (next week)

Option 2) We can perform importance sampling

Importance Sampling

The goal is to estimate:

$$\mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x) f(x) dx \approx \frac{1}{n} \sum_{j=1}^{n} h(x_j) = \bar{h}_n$$

We can't generate values from $f(x)$.

We instead choose to generate values from an entirely different distribution: $g(x)$.

The only requirement is that the support of $f(x)$ has to be within the support of $g(x)$. That is, $g(x) > 0$ for all $x \in \mathcal{X}$.

Importance Sampling

We then re-express the expectation $\mathbb{E}_f[h(X)]$ as an expectation in terms of $g(x)$

$$\begin{aligned}
\mathbb{E}_f[h(X)] &= \int_{\mathcal{X}} h(x)f(x)dx \\
&= \int_{\mathcal{X}} h(x)f(x)\frac{g(x)}{g(x)}dx \\
&= \int_{\mathcal{X}} h(x)\frac{f(x)}{g(x)}g(x)dx \\
(f(x) = 0 \text{ for } x \notin \mathcal{X}) &= \int \left[h(x)\frac{f(x)}{g(x)}\right] g(x)dx \\
&= \mathbb{E}_g\left[h(X)\frac{f(X)}{g(X)}\right]
\end{aligned}$$

Importance Sampling

If we are able to sample from the density function $g(x)$ but not $f(x)$, we can estimate $\mathbb{E}_f[h(X)]$ as follows:

1. Generate $x_1, x_2, \ldots, x_n \overset{\text{iid}}{\sim} g(x)$.
2. Compute $h(x_1)\frac{f(x_1)}{g(x_1)}, h(x_2)\frac{f(x_2)}{g(x_2)}, \ldots, h(x_n)\frac{f(x_n)}{g(x_n)}$.
3. Estimate $\mathbb{E}_f[h(X)]$ by

$$\mathbb{E}_f[h(X)] = \mathbb{E}_g\left[h(X)\frac{f(X)}{g(X)}\right] \approx \frac{1}{n}\sum_{i=j}^{n} h(x_j)\frac{f(x_j)}{g(x_j)}$$

Importance Weight

How does this work?

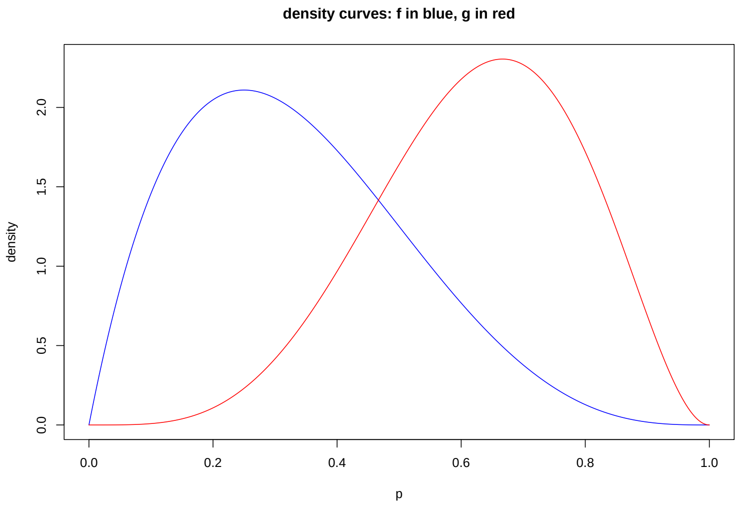
We draw random values from $g(x)$, but we want to estimate the expected value of a function $h(x)$ as if the random values X were being drawn from the distribution $f(x)$.

Importance sampling works because whatever value we draw from $g(x)$ gets weighted by $\frac{f(x)}{g(x)}$.

For illustrative purposes, let's pretend:

- f is a Beta(2, 4) distribution
- g is a Beta(5, 3) distribution

We will also pretend that my computer can only draw values from $g(x)$ but not from $f(x)$. (this is silly, but go with it)


density curves: f in blue, g in red
density
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
p

Importance sampling

- f is a Beta(2, 4) distribution. The mean of f is $\frac{2}{2+4} = 0.333$
- g is a Beta(5, 3) distribution. The mean of f is $\frac{5}{5+3} = 0.625$

Let's say I didn't know the mean of f and I wanted to estimate the expected value of f via Monte Carlo. If I could sample from f directly, it's easy. Just generate a bunch of random values x_j from f and take the mean.

But if I can't sample from f, I will have to sample from a different distribution like g.

If I sample a bunch of random values x_j from $g(x)$ and take the mean, my Monte Carlo estimate will produce a value close to 0.625, which is the mean of a Beta(5, 3) distribution.

```r
set.seed(120)
n <- 10^5
x <- rbeta(n, 5, 3)
mean(x)
```

```
## [1] 0.6251737
```

Importance weight

The amount I need to weigh each observation is the importance weight:

$$w(x_j) = \frac{f(x_j)}{g(x_j)}$$

Compared to a sample that would come from f, with my sample from g, I'm getting way too many values that are close to 0.7, and not enough values that are close to 0.2.

My solution is to weight the values appropriately. When I draw a value from g that is close to 0.7, I'm going to weight those values less. Compared to f, I get too many of them, so I reduce their weight. The value of $g(0.7) > f(0.7)$ so $w(0.7) < 1$

When I draw a value from g that is close to 0.2, I'm going to weight those values more heavily. g rarely produces a value near 0.2, so when I get one, I have to make it "count." The value of $g(0.2) < f(0.2)$ so $w(0.2) > 1$

Importance weights

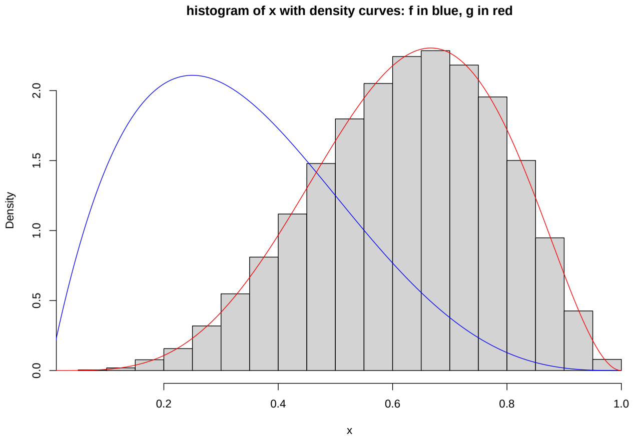
Using the same sample from earlier, I can multiply each value x_j by its weight $w(x_j)$ and estimate the mean of $f(x)$ even though my sample of x_j values came from $g(x)$.

```r
w <- dbeta(x, 2, 4) / dbeta(x, 5, 3) # vector of weights
mean(x * w)
```

```
## [1] 0.3328865
```

And just to show it's not a trick, the histogram of x shows that x came from the "wrong" distribution.

```
hist(x, freq = FALSE, main = "histogram of x with density curves: f in blue, g
lines(p, f, col = "blue"); lines(p, g, col = "red")
```



Importance Sampling

We can revise our algorithm using importance weights:

If we are able to sample from the density function $g(x)$ but not $f(x)$, we can estimate $\mathbb{E}_f[h(X)]$ as follows:

1. Generate $x_1, x_2, \ldots, x_n \overset{\text{iid}}{\sim} g(x)$.
2. Compute weights: $w(x_1) = \frac{f(x_1)}{g(x_1)}, w(x_2) = \frac{f(x_2)}{g(x_2)}, \ldots, w(x_n) = \frac{f(x_n)}{g(x_n)}$.
3. Estimate $\mathbb{E}_f[h(X)]$ by

$$\mathbb{E}_f[h(X)] = \mathbb{E}_g\left[h(X)\frac{f(X)}{g(X)}\right] \approx \frac{1}{n}\sum_{i=j}^{n} h(x_j)w(x_j)$$

Example: Folded Normal Distribution

The folded normal distribution is what youl would get if you took the absolute value of the normal distribution.

To make the total area $= 1$, the PDF of the folded normal is twice as tall as the PDF of the regular normal distribution, but exists only for positive values of x.

$$f(x) = 2 \cdot \frac{1}{\sqrt{2\pi}} e^{-x^2/2} = \sqrt{\frac{2}{\pi}} e^{-x^2/2}, \text{ for } x \geq 0$$

The support of $f(x)$ is $\mathcal{X} = [0, \infty)$

What is the mean of this distribution? That is, what is $\mathbb{E}_f[X]$?

(In this example $h(x) = x$)

We'll use importance sampling in this contrived scenario.

The answer

Technically, we don't have to use importance sampling because we can generate values from the folded normal distribution.

```r
set.seed(321)
n <- 10^6
Z <- rnorm(n)
mean(abs(Z))
```

```
## [1] 0.7973812
```

But we are going to pretend that we can't use generate from the normal distribution.

Example: Folded Normal Distribution

Instead of generating values from the folded normal PDF $f(x)$, we will generate values from an exponential distribution.

The PDF of an exponential distribution with $\lambda = 2$ is:

$$g(x) = \lambda e^{-\lambda x} = 2e^{-2x}, \text{ for } x \geq 0$$

The support of $g(x)$ contains the support of $f(x)$. (In fact, the support is identical for both distributions.)

We can sample from $g(x)$ with `rexp()`

Example: Folded Normal Distribution

1. Generate $x_1, x_2, \ldots, x_n \overset{\text{iid}}{\sim} \text{Exp}(\lambda = 2)$.
2. Compute weights: $w(x_j) = \frac{f(x_j)}{g(x_j)}$

$$\begin{aligned} w(x_j) &= \frac{f(x_j)}{g(x_j)} \\ &= \frac{\sqrt{\frac{2}{\pi}} e^{-x_j^2/2}}{2e^{-2x_j}} \\ &= \frac{2 * \mathtt{dnorm(x_j)}}{\mathtt{dexp(x_j, rate = 2)}} \end{aligned}$$

3. Estimate $\mathbb{E}_f[h(X)]$ ($h(x) = x$):

$$\mathbb{E}_f[h(X)] = \mathbb{E}_g\left[h(X)\frac{f(X)}{g(X)}\right] \approx \frac{1}{n}\sum_{i=j}^{n} h(x_j)w(x_j)$$

```r
# importance sampling estimate
set.seed(321)
n <- 10 ^ 6
X <- rexp(n, rate = 2)
W <- 2 * dnorm(X) / dexp(X, rate = 2)
mean(X * W)
```

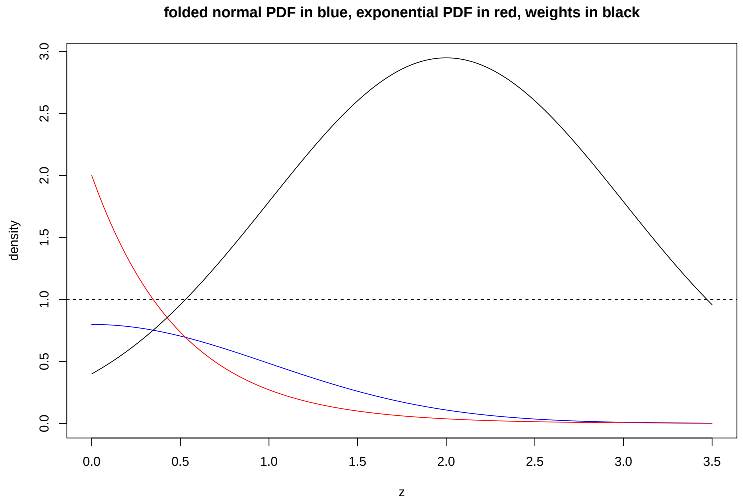
```
## [1] 0.7976598
```

```r
# monte carlo estimate using abs(Z)
mean(abs(Z))
```

```
## [1] 0.7973812
```

```r
# theoretic value
sqrt(2 / pi)
```

```
## [1] 0.7978846
```



We can see where the $g > f$ (red line is above the blue line), the weights are less than 1 and where the $g < f$, the weights are greater than 1.

Another Example - Extreme tails

What if we want to estimate the probability that Z from a standard normal distribution is over 4.5? The true answer is:

```r
1 - pnorm(4.5)
```

```
## [1] 3.397673e-06
```

If we try to approximate this using regular Monte Carlo integration, we would sample many values from the normal distribution and see how many are larger than 4.5.

```r
mean(Z > 4.5)
```

```
## [1] 2e-06
```

Because $\Pr(Z > 4.5)$ is so rare, it happened only 2 times in a million. This is not a very good estimate of the true probability of around 3.4 times in a million.

Extreme tails

We can improve our accuracy by just sampling many more values, or we can improve our performance using importance sampling.

We can consider generating values from $g(x)$, an exponential distribution that begins at 4.5. $\mathcal{X} = [4.5, \infty)$

We know the PDF must integrate to 1:

$$\int_{4.5}^{\infty} g(x)dx = 1$$

$$g(x) = \frac{e^{-x}}{constant} = \frac{e^{-x}}{\int_{4.5}^{\infty} e^{-x}dx} = \frac{e^{-x}}{[-e^{-x}]_{4.5}^{\infty}} = \frac{e^{-x}}{0 + e^{-4.5}} = e^{-(x-4.5)}$$

Extreme tails

The weights are then:

$$w(x_i) = \frac{f(x_i)}{g(x_i)} = \frac{\frac{1}{\sqrt{2\pi}} e^{-x_i^2/2}}{e^{-(x_i - 4.5)}}$$

We simply want the probability of being in the tail, so

$$h(x) = I(x > 4.5) = \begin{cases} 1 & \text{for all } x \geq 4.5 \\ 0 & \text{for all } x < 4.5 \end{cases}$$

I will simply generate random values from the exponential distribution and then add 4.5. The resulting distribution is exactly equal to an exponential distribution that begins at 4.5. This works because of the memory-less property of the exponential distribution.

If all generated values are greater than 4.5, h(x) $=$ 1 for all generated values.

$$\mathbb{E}_f[h(X)] = \mathbb{E}_g\left[h(X)\frac{f(X)}{g(X)}\right] \approx \frac{1}{n}\sum_{i=j}^{n} h(x_j)w(x_j)$$

```r
set.seed(5)
N <- 10^6
X <- rexp(N) + 4.5  # X is generated by g(), an exponential distribution
# weights are found by taking normal density and dividing by exp density
W <- dnorm(X)/ dexp(X - 4.5)
h = 1
mean(h * W) # compare to actual value of 3.397673e-06
```

```
## [1] 3.39082e-06
```