# Variance Reduction in Direct Monte Carlo

## Importance Sampling

Sachin Shanbhag

Department of Scientific Computing
Florida State University,
Tallahassee, FL 32306.

## Preliminaries

Consider the expected value of a function $f(x)$

$$E_\pi[f] = \int_a^b f(x)\pi(x)\, dx, \tag{1}$$

where $\pi(x)$ is normalized probability distribution with

$$\int_a^b \pi(x) dx = 1.$$

Setting $g(x) = f(x)\pi(x)$

$$E_\pi[f] = \int_a^b g(x)\, dx = (b-a)\bar{g}. \tag{2}$$

There are two ways to calculate this expected value.

# Calculating Expected Values

## Method 1: Sampling from Uniform distribution

This is our standard way of computing integrals with MC.

- draw $n$ samples $X_1, X_2, ..., X_n$ from $u(x) = U[a, b]$
- estimate $\bar{g}$ by averaging

$$\bar{g} \approx \frac{1}{n} \sum_{j=1}^{n} g(X_j) = \frac{1}{n} \sum_{j=1}^{n} \pi(X_j) f(X_j) \tag{3}$$

- hence, estimate $E_\pi[f] = \bar{g}(b - a)$:

$$E_\pi[f] = \frac{b-a}{n} \sum_{j=1}^{n} \pi(X_j) f(X_j)$$

We sampled from the uniform distribution on the domain.

## Calculating Expected Values

If it is convenient to sample from the distribution $\pi(x)$ directly, we can propose another method.

**Method 2: Sampling from target distribution**

- draw $n$ samples $X_1, X_2, ..., X_n$ from $\pi(x)$
- estimate $E_\pi[f]$ directly by a simple average:

$$E_\pi[f] \approx \frac{1}{n} \sum_{j=1}^{n} f(X_j) \tag{4}$$

Note the absence of $(b-a)$ and $\pi(X_j)$ in the computation relative to method 1.

Let us consider a simple example to demonstrate these two methods.

## Example

Consider,

$$E_\pi[f] = \int_{-10}^{10} f(x)\pi(x)dx = \int_{-10}^{10} g(x)dx \qquad (5)$$

with $f(x) = x^2$, and $\pi(x) = \mathcal{N}(0,1)$, the unit normal distribution,

$$\pi(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}).$$

Note that $\pi(x)$ is almost normalized,

$$\int_{-10}^{10} \pi(x)dx \approx 1,$$

and the true solution of the problem is $E_\pi[f] = 1.0$.

## Python Code

```
f = lambda x: x**2
p = lambda x: 1./np.sqrt(2.*np.pi) * np.exp(-x**2/2.)
g = lambda x: f(x) * p(x)

# numerical integral
scipy.integrate.quad(g, -10, 10)
(1.0, 7.348033275594856e-10)
# Method 1
x     = np.random.uniform(-10, 10, size=10000)
gbar1 = np.mean(g(x))
fbar1 = gbar1 * (10. - (-10.))  # close match
1.00636345198
```

## Python Code

```
# Method 2
x = np.random.normal(0, 1, size=10000)
fbar = np.mean(f(x)) # close match
1.00961719771
```

Unlike the first method, the second method:

(i) does not involve multiplication with $(b - a)$
(ii) works well even when the distribution is not properly normalized.

## Side Note

Now consider our old friend in light of the two methods,

$$I = \int_a^b f(x)\, dx = (b-a)\langle f \rangle.$$

Here,

$$\begin{aligned} \langle f \rangle &= \frac{1}{b-a}\int_a^b f(x)dx \\ &= \int_a^b f(x)u(x)dx \\ &= E_u[f], \end{aligned} \tag{6}$$

since, $u(x) = U[a,b] = 1/(b-a)$.

To compute $\bar{f}$ (estimate of $\langle f \rangle$), we effectively used "method 2".

## Side Note

### Algorithm

- draw samples $X_1, X_2, ..., X_n$ from $\pi(x) = u(x) = U[a, b]$
- estimate $E_u[f]$ by a simple average:

$$E_u[f] \approx \frac{1}{n} \sum_{j=1}^{n} f(X_j)$$

To find $I$ we would have to multiply $E_u[f]$ by the domain size $(b - a)$.

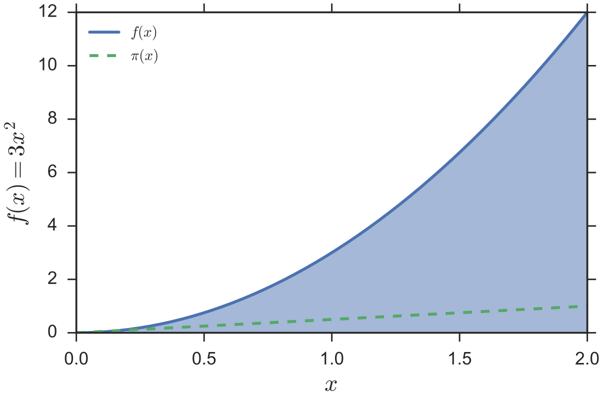
Now suppose, instead of choosing points uniformly, we “focused” on important regions of the domain.

This idea is best illustrated with an example in mind.

## Importance Sampling

Consider the simple integral:

$$I = \int_0^2 3x^2 dx.$$



## Importance Sampling

Integral $=$ area under the curve

Points close to $b = 2$ contribute more to the integral than points close to $a = 0$.

Therefore instead of choosing points uniformly between $a$ and $b$, suppose we draw points from the distribution $\pi(x) = x/2$, which is shown by the dashed line.

Of course, we will need to unbias our estimate.

Informally, instead of taking a simple average of uniformly weighted points, we will take a weighted average.

## Math behind the Idea

Since, $I = E_u[f](b-a)$, and

$$\begin{aligned} E_u[f] &= \int_a^b f(x)u(x)dx \\ &= \int_a^b f(x)u(x)\frac{\pi(x)}{\pi(x)}dx \\ &= \frac{1}{b-a}\int_a^b \frac{f(x)}{\pi(x)}\pi(x)dx \\ I &= \int_a^b \frac{f(x)}{\pi(x)}\pi(x)dx = E_\pi\left[\frac{f}{\pi}\right], \end{aligned}$$

where $\pi(x)$ is normalized over the domain.

We can use method 2 to evaluate this integral.

## IS: Algorithm

This observation yields the algorithm for importance sampled MC integration:

1. Draw $X_1, X_2, ..., X_n$ from $\pi(x)$
2. Estimate $I = E(f/\pi)$ via

$$I \approx \frac{1}{n} \sum_{i=1}^{n} \frac{f(X_i)}{\pi(X_i)}.$$

One can estimate the error in $I$ by keeping track of the standard deviation of $\{f(X_i)/\pi(X_i)\}$.

## Back to Example

Find the integral,

$$I = \int_0^2 3x^2 dx.$$

by (importance) sampling from $\pi(x) = x/2$.

Solution

First, we can use the transformation rule to sample from $\pi(x) = x/2$.

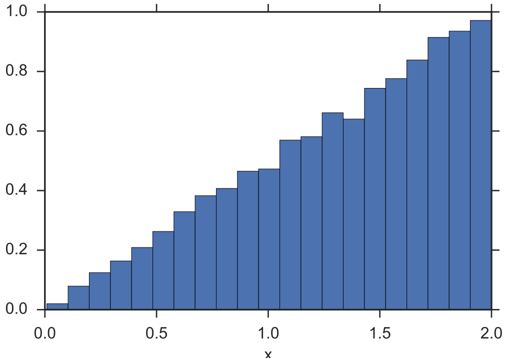
If $u \sim U[0, 1]$, then

$$u = F(x) = x^2/4 \implies x = \sqrt{4u}$$

```
def drawLinearDist(npts):
    u = np.random.rand(npts)
    return np.sqrt(4.0*u)
```

## Back to Example

Let us test by plotting the histogram.



## Error in Importance Sampling: Insight

For uniform sampling (Method 1), we saw earlier that

$$\sigma_I^2 = \frac{V^2}{n}\sigma_f^2.$$

That is, the error $\sigma_I \approx 0$, as $f(x) \approx$ constant.

For importance sampling:

$$\sigma_I^2 = \frac{\sigma_{f/\pi}^2}{n},$$

where

$$\sigma_{f/\pi}^2 = \left\langle \left(\frac{f}{\pi}\right)^2 \right\rangle - \left\langle \left(\frac{f}{\pi}\right) \right\rangle^2$$

That is, the error $\sigma_I \approx 0$, as $\pi(x) \approx f(x)$. Pick a function $\pi(x)$ which resembles $f(x)$, but is easier to sample from.

## Uniform Sampling

Now let's compare uniform and importance sampling for this problem.

We begin by writing python code for each:

```
def uniformSampling(npts):
    """draw points uniformly distributed"""
    b  = 2.
    a  = 0.
    x  = np.random.uniform(a, b, npts)
    f  = 3.*x**2

    intg = (b-a) * np.mean(f)
    stdI = (b-a) * np.std(f)/np.sqrt(npts)

    return intg, stdI
```

## Importance Sampling

```python
def importanceSampling(npts):
    """draw points from linear distribution"""
    x  = drawLinearDist(npts)
    f  = 3.*x**2
    p  = x/2.0   #pi(x)

    intg = np.mean(f/p)
    stdI = np.std((f/p))/np.sqrt(npts) # error

    return intg, stdI
```

## Comparison

Use $n = 50$ darts. The true solution is 8.

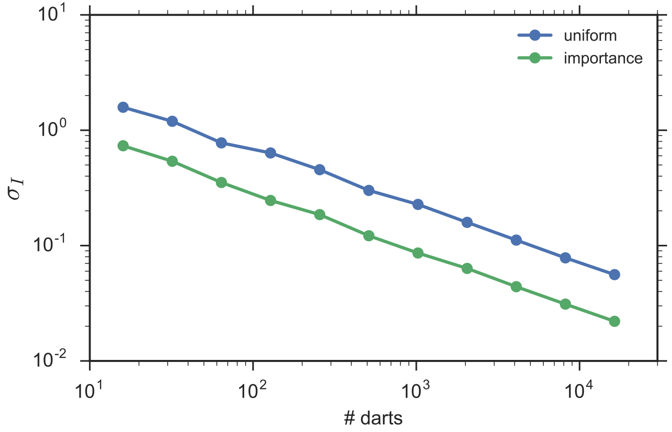
```
print(uniformSampling(50))
(8.8636221283119916, 1.0940151684845272)

print(importanceSampling(50))
(7.817575629020471, 0.40950491361094382)
```

Notice that the error is smaller for importance sampling.

This difference is more pronounced for integrals of functions that are sharply peaked.

## Comparison of Error

For this particular example, $\sigma_I \sim 1/\sqrt{n}$.



Importance sampling tries to reduce the error by modifying the numerator of $\sigma_I$.

## Intuition

Practically, for “flat” integrands (the area of a circle example), the principal goal is to cover all parts of the domain; uniform sampling works perfectly fine.

If the integrand has only a few sharp peaks, which contribute most of the weight to the integral, then importance sampling might be helpful.

Analogy 1: If you want to find the number of holes on a golf course, you don’t want to sprinkle darts all over the place.

Analogy 2: If you want to find the average height of a palm tree between the US and Japan, you don’t want to randomly throw darts, since most of them will land in the Pacific Ocean. You want to focus on islands and other land masses.

## Summary

- Two methods to solve the integral,

$$E_\pi[f] = \int_a^b f(x)\pi(x)\, dx$$

  involving sampling $X_i \sim U[a,b]$, and $X_i \sim \pi(x)$
- Importance Sampling: Can write the integral,

$$I = \int_a^b f(x)\, dx = \int_a^b \frac{f(x)}{\pi(x)} \pi(x)\, dx$$

  and use method 2.
- The error with importance sampling is:

$$\sigma_I^2 = \frac{\sigma_{f/\pi}^2}{n}.$$