What are Monte Carlo Methods?

The subject of Monte Carlo methods can be viewed as a branch of “experimental mathematics” in which one uses random numbers to conduct experiments. Typically the experiments are done on a computer using anywhere from hundreds to billions of random numbers.

Two categories of Monte Carlo experiments

(1) Direct simulation of a random system.

(2) Addition of artificial randomness to a system.
**Direct simulation of a random system** We study a real system that involves some randomness, and we wish to learn what behavior can be expected without actually watching the real system.

We first formulate a mathematical model by identifying the key random (and nonrandom) variables which describe the given system. Then we run a random copy of this model on a computer, typically many times over different values of random variables. Finally we collect data from these runs and analyze the data. Here are some applications:

(a) Operations research: e.g., hospital management.

(b) Reliability theory.

(c) Physical, biological, and social sciences: models with complex nondeterministic time evolution, including particle motion, population growth, epidemics, and social phenomena.
Addition of artificial randomness to a system One represents the underlying problem (which may be completely deterministic) as part of a different random system and then performs a direct simulation of this new system.

Example: estimating $\pi$.

(a) Markov chain Monte Carlo: for problems in statistical physics (e.g., proteins, quantum systems) and in Bayesian statistics (for complicated posterior distributions).

(b) Optimization: e.g. solving the traveling salesman and knapsack problems by simulated annealing or genetic algorithms.
Figure 1: Monte Carlo estimation of $\pi$. 
Basic problems of Monte Carlo

(a) Draw random variables

\[ X \sim \pi(x) \]

Sometimes with unknown normalizing constant

(b) Estimate the integral

\[ I = \int h(x)\pi(x)dx = E_{\pi} h(X) \]
Example: Estimate the integral $I = \int_0^1 h(x)dx$.

- Draw $u_1, \ldots, u_n$ iid from Uniform(0,1).

- Law of large numbers:
  $$\hat{I} = \frac{1}{n}[h(u_1) + \cdots + h(u_n)] \to I, \quad \text{as } n \to \infty$$

- Error Computation:
  - Unbiased $E(\hat{I}) = I$.
  - Variance $\text{Var}(\hat{I}) = \frac{\text{Var}(h)}{n} = \frac{1}{n} \int_0^1 (h(x) - I)^2dx$. 


**Pseudo-random number generator**

- A sequence of pseudo-random number \( (U_i) \) is a deterministic sequence of numbers in \([0,1]\) having the same relevant statistical properties as a sequence of random numbers.

- von Neumann’s “middle square” method: start with 8653, square it and make the middle 4 digits:
  
  \[ 8653, 8744, 4575, 9306, 6016, 1922, \ldots \]

- Congruential generator:
  
  \[ X_i = aX_{i-1} \mod M; \quad \text{and} \quad U_i = X_i/M, \]

  say, \( a = 7^5 = 16807 \), and \( M = 2^{31} - 1 \).

- Other methods and statistical testing (see Ripley 1987; Marsaglia and Zaman, 1993)
0.1 The inversion method

**Theorem 0.1** (Probability Integral Transformation Theorem) If $U \sim \text{Uniform}[0, 1]$ and $F$ is a cdf, then $X = F^{-1}(U) \equiv \inf \{ x : F(x) \geq u \}$ follows $F$.

**Example:** Generation of exponential random variables. $F(x) = 1 - \exp(-\lambda x)$. Set $F(x) = U$, and derive that $x = -\frac{1}{\lambda} \log(1 - U)$.

- Generate $U \sim \text{Unif}[0, 1]$.
- Set $x = -\frac{1}{\lambda} \log(U)$.
- $s \sim \exp(\lambda)$. 
Suppose we want to generate the value of a discrete random variable \( X \) having probability mass function
\[
P(X = x_j) = p_j, \quad j = 0, 1, \ldots, \quad \sum_j p_j = 1
\]

- Generate a random number \( U \).
- if \( U < p_0 \), set \( X = x_0 \) and stop.
- if \( U < p_0 + p_1 \), set \( X = x_1 \) and stop.
- if \( U < p_0 + p_1 + p_2 \), set \( X = x_2 \) and stop.
- \( \vdots \)

**Remark:** If the \( x_i, i \geq 0 \), are ordered so that \( x_0 < x_1 < \cdots \) and if we let \( F \) denote the distribution function of \( X \), then
\[
F(x_k) = \sum_{i=0}^k p_i
\]
and so
\[
X \text{ will be equal to } x_j \text{ if } F(x_{j-1}) \leq U < F(x_j)
\]
In other words, after generating a random number $U$ we determine the value of $X$ by finding the interval $[F(x_{j-1}), F(x_j)]$ in which $U$ lies [or, equivalently, by finding the inverse of $F(U)$]. It is for this reason that the above is called the inversion method for generating $X$. 
Example 1 If we want to simulate a random variable $X$ such that

$$p_1 = 0.2, \quad p_2 = 0.15, \quad p_3 = 0.25, \quad p_4 = 0.4,$$

where $p_j = P(X = j)$.

We could generate $u$ and do the following:

- if $U < 0.2$ set $X = 1$ and stop.
- if $U < 0.35$ set $X = 2$ and stop.
- if $U < 0.6$ set $X = 3$ and stop.
- Otherwise set $X = 4$.

However, a more efficient procedure is the following

- if $U < 0.4$ set $X = 4$ and stop.
- if $U < 0.65$ set $X = 3$ and stop.
- if $U < 0.85$ set $X = 1$ and stop.
- Otherwise set $X = 2$.  

Example 2  (Geometric random variables)

\[ P(X = i) = pq^{i-1}, \quad i \geq 1, \quad \text{where } q = 1 - p \]

Since

\[
\sum_{i=1}^{j-1} P(X = i) = 1 - P(X > j - 1)
\]

\[ = 1 - P(\text{The first } j - 1 \text{ trials are all failures}) \]

\[ = 1 - q^{j-1} \]

We can generate the value of \( X \) by generating a random number \( U \) and setting \( X \) equal to that value \( j \) for which

\[ 1 - q^{j-1} \leq U < 1 - q^j \]

or, equivalently, for which

\[ q^j < 1 - U \leq q^{j-1} \]
Thus, we can define $X$ by

$$X = \min\{j : q^j < 1 - U\}$$

$$X = \min\{j : j \log(q) < \log(1 - U)\}$$

$$= \min\{j : j > \frac{\log(1 - U)}{\log(q)}\}$$

Hence, we can express $X$ as

$$X = \text{int}\left(\frac{\log(1 - U)}{\log(q)}\right) + 1$$

where $\text{int}(x)$ denotes the integer part of $x$. We can also write $X$ as

$$X = \text{int}\left(\frac{\log(U)}{\log(q)}\right) + 1,$$

Because $1 - U$ is also uniformly distributed on $(0,1)$. 
Example 3 (Poisson random variables) The random variable $X \sim \text{Poisson}(\lambda)$,

$$p_i = P(X = i) = e^{-\lambda} \frac{\lambda^i}{i!}$$

for $i = 0, 1, \ldots$.

For $p_i$ and $p_{i+1}$, we have the following recursive relationship,

$$p_{i+1} = \frac{\lambda}{i+1} p_i, \quad i \geq 0$$

Algorithm

- Generate a random number $U$.
- $i = 0, p = e^{-\lambda}, F = p$.
- If $U < F$, set $X = i$ and stop.
- $p = \lambda p/(i + 1), F = F + p, i = i + 1$. 
• Goto step 3.

**Remark:** Based on the property of Poisson distribution, if $\lambda$ is very large, one efficient generation algorithm is as follows: Let $I = \text{int}(\lambda)$, first generate a random number to determine $X$ is larger or smaller than $I$, then searches downward starting from $X = I$ in the case where $X \leq I$ and upwards starting from $X = I + 1$ otherwise.

Average number of searches

$$= 1 + E[|X - \lambda|]$$

$$= 1 + \sqrt{\lambda}E\left[\frac{|X - \lambda|}{\sqrt{\lambda}}\right]$$

$$= 1 + \sqrt{\lambda}E[|Z|], \quad \text{where Z} \sim N(0, 1)$$

$$= 1 + 0.798\sqrt{\lambda}$$
Example 4 (Binomial Random Variables)

$X \sim \text{Binomial}(n, p)$, and

$$P(X = i) = \frac{n!}{i!(n-1)!} p^i (1-p)^{n-i}$$

for $i = 0, 1, \ldots, n$.

The recursive identity,

$$P(X = i + 1) = \frac{n - i}{i + 1} \frac{p}{1 - p} P(X = i)$$

Algorithm:

- Generate a random number $U$.
- $c = p/(1-p), i = 0, pr = (1 - p)^n, F = pr$.
- If $U < F$, set $X = i$ and stop.
- $pr = \frac{c(n-i)}{i+1} pr, F = F + pr, i = i + 1$. 

• goto step 3.

**Remark:** As in Poisson case, when the mean $np$ is large it is better to first determine if the generated value is less than or equal to $I = \text{int}(np)$ or whether it is larger than $I$. Then decide to search downward or upward.
0.2 Continuous Cases

Example 1 Suppose we want to generate $X$ from the distribution

$$F(x) = x^n, \quad 0 < x < 1$$

Solution: Let $x = F^{-1}(u)$, then

$$u = F(x) = x^n \quad \text{or, equivalently,} \quad x = u^{1/n}$$

Hence we have the following algorithm for generating a random variable from $F(x)$.

- Generate a random number $U \sim U(0, 1)$.
- Set $X = U^{1/n}$. 
Example 2  (Exponential distribution) If $X$ is an exponential random variable with rate 1, then its distribution function is given by

$$F(x) = 1 - e^{-x}$$

Let $x = F^{-1}(u)$, we have

$$M_u = F(x) = 1 - e^{-x}$$

or, taking logarithms,

$$x = -\log(1 - u)$$

Hence we can generate an exponential with parameter 1 by generating a random number $U$ and then setting

$$X = F^{-1}(U) = -\log(1 - U) = -\log(U)$$

In general, an exponential random variable with rate $\lambda$ can be generated by generating a random number $U$, and setting

$$X = -\frac{1}{\lambda} \log(U).$$
Example 3 (Gamma distribution) Suppose we want to generate the value of a gamma \((n, \lambda)\) random variable.

\[
F(x) = \int_0^x \frac{\lambda e^{-\lambda y} (\lambda y)^{n-1}}{(n-1)!} dy
\]

Since it is not possible to give a closed form expression for the inverse \(F^{-1}(x)\), we cannot use the inverse transform method here. However, by using the result that a gamma \((n, \lambda)\) random variable \(X\) can be regarded as being the sum of \(n\) independent exponentials, each with rate \(\lambda\), we can make use of example 2 to generate \(X\). Particularly, we can generate a gamma \((n, \lambda)\) random variable by generating \(n\) random numbers \(U_1, \ldots, U_n\) and then setting

\[
X = -\frac{1}{\lambda} \log U_1 - \cdots - \frac{1}{\lambda} \log U_n
\]

\[
= -\frac{1}{\lambda} \log(U_1 \cdots U_n)
\]
Where the use of the identity \( \sum_{i=1}^{n} \log(x_i) = \log(x_1 \cdots x_n) \) is computationally time saving in that it requires only one rather than \( n \) logarithm computations.
0.3 The rejection method

- Generate $x$ from $g(x)$, where $g(x)$ is called the envelope distribution.
- Draw $u$ from $\text{unif}[0, cg(x)]$.
- Accept $x$ if $u < \pi(x)$.
- The accepted $x$ follows $\pi(x)$.

**Efficiency:** The acceptance rate $\int \pi(x) dx/[c \int g(x) dx] = 1/c$.

**Example:** simulate from truncated Gaussian

$$\pi(x) \propto \phi(x) I_{x>k}$$

Set $g(x) = \phi(x)$. The efficiency is $1 - \Phi(k)$. 
Figure 2: Illustration of the rejection method.
Example 1  Simulate a distribution with probability \( P = \{0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10\} \). whereas one possibility is to use the inverse transform algorithm, a better approach is to use the rejection method with \( q \) being the discrete uniform density on \( 1, 2, \cdots, 10 \). That is, \( q_j = 1/10, j = 1, 2, \cdots, 10 \). Choose \( c = 1.2 \) by \( c = \max\{p_j/q_j\} = 1.2 \), so the algorithm is as follows:

- Generate a random number \( U_1 \) and set \( Y = \text{int}(10U_1) + 1 \).
- Generate a second random number \( U_2 \).
- If \( U_2 \leq p_Y/1.2 \), set \( X = Y \) and stop. Otherwise return to step 1.
Example 2 Using the rejection method to generate a random variable having density function

\[ f(x) = 20x(1 - x)^3, \quad 0 < x < 1 \]

Since this random variable \((\text{Beta}(2,4))\) is concentrated in the interval \((0,1)\), let us consider the rejection method with

\[ g(x) = 1, \quad 0 < x < 1. \]

To determine the constant \(C\), we maximize the following function

\[ \frac{f(x)}{g(x)} = 20x(1 - x)^3 \]

\[ \frac{d}{dx}(20x(1 - x)^3) = 20[(1 - x)^3 - 3x(1 - x)^2] \]

Setting this equal to 0 shows that the maximal value is attained when \(x = 1/4\), and thus

\[ f(x)/g(x) \leq 20(0.25)(1 - 0.25)^3 = \frac{135}{64} = c \]
Hence,

\[ \frac{f(x)}{cg(x)} = \frac{256}{27} x (1 - x)^3 \]

and thus the rejection procedure is as follows,

- Generate random number \( U_1 \) and \( U_2 \).
- If \( U_2 \leq \frac{256}{27} U_1 (1 - U_1)^3 \), stop and set \( X = U_1 \). Otherwise, return to step 1.

The average number of times that step 1 will be performed is \( c = \frac{135}{64} \approx 2.11 \).
Example 3  Suppose we want to generate a random variable having the Gamma\(1.5,1\) density

\[ f(x) = K x^{1/2} e^{-x}, \quad x > 0 \]

where \( K = 1/\Gamma(1.5) = 2/\sqrt{\pi} \). Because such a random variable is concentrated on the positive axis and has mean 1.5, it is natural to try the rejection technique with an exponential random variable with the same mean. Hence, let

\[ g(x) = \frac{2}{3} e^{-2x/3}, \quad x > 0 \]

We have

\[ \frac{f(x)}{g(x)} = \frac{3}{2} K x^{1/2} e^{-x/3} \]

maximize the ratio, we get 
\( c = \frac{3^{3/2}}{(2\pi)^{1/2}}, \) and

\[ \frac{f(x)}{cg(x)} = (2e/3)^{1/2} x^{1/2} e^{-x/3} \]
So the algorithm is as follows,

- Generate a random number $U_1$, and set $Y = -\frac{3}{2} \log U_1$.
- Generate a random number $U_2$.
- If $U_2 < (2eY/3)^{1/2}e^{-Y/3}$, set $X = Y$, otherwise, return to step 1.
0.4 The polar method for generating normal random variables

Let $X$ and $Y$ be independent unit normal random variables and let $R$ and $\theta$ denote the polar coordinates of the vector $(X, Y)$. That is

$$R^2 = X^2 + Y^2$$
$$\tan \theta = \frac{Y}{X}$$

Since $X$ and $Y$ are independent, we have the joint density

$$f(x, y) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} = \frac{1}{2\pi} e^{-(x^2+y^2)/2}$$

To determine the joint density of $R^2$ and $\theta$, we make the change of variables

$$d = x^2 + y^2 \quad \theta = \tan^{-1} \left( \frac{y}{x} \right)$$

We have (with $|J| = 2$),

$$f(d, \theta) = \frac{1}{2} \frac{1}{2\pi} e^{-d/2}, \quad 0 < d < \infty, \quad 0 < \theta < 2\pi \quad (1)$$
However, as this is equal to the product of an exponential density having mean 2 (namely, \( \frac{1}{2} e^{-d/2} \)) and the uniform density on \((0, 2\pi)\), it follows that \( R^2 \) and \( \theta \) are independent, with \( R^2 \) being exponential with mean 2 and \( \theta \) being uniformly distributed over \((0, 2\pi)\).

We can now generate a pair of independent standard normal random variables \( X \) and \( Y \) by using (1) to first generate their polar coordinates and then transforming back to rectangular coordinates. The algorithm is as follows:

**Algorithm 1**

- generate random number \( U_1 \) and \( U_2 \).
- \( R^2 = -2 \log(U_1) \), set \( \theta = 2\pi U_2 \).
- let
  
  \[
  X = R \cos \theta = \sqrt{-2 \log U_1} \cos(2\pi U_2) \\
  Y = R \sin \theta = \sqrt{-2 \log U_1} \sin(2\pi U_2)
  \]  

(2)
The above transformation is known as Box-Muller transformation. However, the above algorithm is not very efficient: the reason for this is the need to compute the sine and cosine trigonometric functions. There is a way to get around this time-consuming difficulty by an indirect computation of the sine and cosine of a random angle. The algorithm is as follows.

Generate $U_1$ and $U_2$ from $U(0,1)$, and set $V_1 = 2U_1 - 1$, $V_2 = 2U_2 - 1$. Then $(V_1, V_2)$ is uniformly distributed in the square of area 4 centered at $(0, 0)$. Suppose now that we continually generate such pairs $(V_1, V_2)$ until we obtain one that is contained in the circle of radius 1 centered at $(0, 0)$—that is, until $(V_1, V_2)$ such that $V_1^2 + V_2^2 \leq 1$. It now follows that such a pair $(V_1, V_2)$ is uniformly distributed in the circle. If we let $R$ and $\theta$ denote the polar coordinates of this pair, then it is not difficult to verify that $R$ and $\theta$ are independent, with $R^2$ being uniformly distributed on $(0, 1)$ and with $\theta$ being uniformly distributed over $(0, 2\pi)$. Since $\theta$ is thus a random angle, it follows that we can generate the sine and cosine of a random angle by generating a random point $(V_1, V_2)$ in
the circle and setting

\[
\begin{align*}
\sin \theta &= \frac{V_2}{R} = \frac{V_2}{(V_1^2 + V_2^2)^{1/2}} \\
\cos \theta &= \frac{V_1}{R} = \frac{V_1}{(V_1^2 + V_2^2)^{1/2}}
\end{align*}
\]

Following the Box-Muller transformation, we can generate independent unit normals as follows,

\[
\begin{align*}
X &= (-2 \log(U))^{1/2} \frac{V_1}{(V_1^2 + V_2^2)^{1/2}} \\
Y &= (-2 \log(U))^{1/2} \frac{V_2}{(V_1^2 + V_2^2)^{1/2}}
\end{align*}
\tag{3}
\]

Since \(R^2 = X^2 + Y^2\) is itself uniformly distributed over (0,1) and is independent of the random angle \(\theta\), we can use it as the random number \(U\) needed in equation (3). Therefore, letting \(S = R^2\), we obtain that

\[
\begin{align*}
X &= (-2 \log(S)/S)^{1/2} V_1 \\
Y &= (-2 \log(S)/S)^{1/2} V_2
\end{align*}
\tag{4}
\]
are independent unit normals when \((V_1, V_2)\) is a randomly chosen point in the circle of radius 1 centered at the origin, and \(S = V_1^2 + V_2^2\).

Summing up, the algorithm is as follows,

**Algorithm 2**

- Generate random numbers, \(U_1\) and \(U_2\).
- Set \(V_1 = 2U_1 - 1\), \(V_2 = 2U_2 - 1\), \(S = V_1^2 + V_2^2\)
- If \(S > 1\) return to step 1.
- return the independent unit normals
  \[
  X = (-2 \log(S)/S)^{1/2} V_1 \\
  Y = (-2 \log(S)/S)^{1/2} V_2
  \]
0.5 Multivariate normal distribution

$\mathbf{X} \sim N_d(\mu, \Sigma)$ has the following density function

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left\{ -\frac{(\mathbf{x} - \mu)'\Sigma^{-1}(\mathbf{x} - \mu)}{2} \right\}$$

A direct way of generating random vectors from the distribution is to generate a $d$-vector of i.i.d standard normal deviates $\mathbf{z} = (z_1, z_2, \cdots, z_d)$ and then to form the vector

$$\mathbf{x} = T'\mathbf{z} + \mu^M$$

Where $T$ is a $d \times d$ matrix such that $T'T = \Sigma$. ($T$ could be a Cholesky factor of $\Sigma$, for example.) Then $\mathbf{x}$ has a $N_d(\mu, \Sigma)$ distribution.

Another approach for generating the $d$-vector $\mathbf{x}$ from $N_d(\mu, \Sigma)$ is to generate $x_1$ from $N_1(\mu_1, \sigma_{11})$, generate $x_2$ conditionally on $x_1$, generate $x_3$ conditionally on $x_1$ and $x_2$, and so on.
0.6 Multinomial distribution

The probability function for the $d$-variate multinomial distribution is

$$p(x) = \frac{n!}{\prod x_j!} \prod \pi_j^{x_j}$$

for $\sum \pi_j = 1$, $x_j \geq 0$, and $\sum x_j = n$.

To generate a multinomial, a simple way is to work with the marginals, they are binomials. The generation is done sequentially. Each succeeding conditional marginal is binomial. For efficiency, the first marginal considered would be the one with the largest probability.
Variance Reduction Methods

- **Control variables**—Suppose of interest is $\mu = E(X)$.
  
  - Let $C$ be an r.v. with $E(C) = \mu_c$ and be correlated with $X$.
  
  - Let $X(b) = X + b(C - \mu_C)$.

  $$\text{Var}(X(b)) = \text{Var}(X) + b^2\text{Var}(C) + 2b\text{Cov}(X, C).$$

  Choose $b$ so as to have a reduced variance $\text{Var}(X(b)) = (1 - \rho_{XC}^2)\text{Var}(X)$.

  - Another scenario: $E(C) = \mu$. Then we can form $X(b) = bX + (1 - b)C$.  

• **Antithetic Variates**—a way to produce negatively correlated Monte Carlo samples.

\[ X = F^{-1}(U); \quad X' = F^{-1}(1 - U) \]

Then Cov\((X, X')\) \(\leq 0\).

**Exercise:** prove that \(X\) and \(X'\) are negatively correlated. (J.S. Liu, 2001)
• **Rao-Blackwellization** This method reflects a basic principle in Monte Carlo computation: One should carry out analytical computation as much as possible.

A straightforward estimator of $I = E_{\pi} h(x)$ is

$$\hat{I} = \frac{1}{m} \{ h(x^{(1)}) + \cdots + h(x^{(m)}) \}.$$ 

Suppose that $x$ can be decomposed into two parts $(x_1, x_2)$ and that the conditional expectation $E[h(x)|x_2]$ can be carried out analytically (integrating out $x_1$). An alternative estimator of $I$ is

$$\bar{I} = \frac{1}{m} \{ E[h(x)|x_2^{(1)}] + \cdots + E[h(x)|x_2^{(m)}] \}.$$ 

Both $\hat{I}$ and $\bar{I}$ are unbiased because of the simple fact that

$$E_{\pi} h(x) = E_{\pi} [ E\{ h(x)|x_2 \} ].$$
Following from the identity

\[ \text{Var}\{ h(x) \} = \text{Var}\{ E[h(x)|x_2] \} + E\{ \text{Var}[h(x)|x_2] \}, \]

we have

\[ \text{Var}(\bar{I}) = \frac{1}{m} \text{Var}\{ h(x) \} \geq \frac{1}{m} \text{Var}\{ E[h(x)|x_2] \} = \text{Var}(\bar{I}). \]
Importance Sampling and Weighted sample

- Of interest $\mu = \int h(x)\pi(x)dx$.

- We can (and sometimes prefer to) draw samples from $g(x)$, which concentrates on “region of importance” and has a larger support set than that of $\pi(x)$.

- Let $X_1, \ldots, X_N$ be samples from $g(x)$, then
  \[
  \hat{\mu} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i h(X_i),
  \]
  where $w_i = \pi(X_i)/g(X_i)$.

- An alternative estimator is
  \[
  \tilde{\mu} = \frac{1}{N} \sum_{i=1}^{n} w_i h(X_i),
  \]
provided that the normalizing constants of $g$ and $\pi$ are known.
**Importance Sampling**

- For the simpler estimator: $E(\tilde{\mu}) = \mu$ and

$$\text{Var}(\tilde{\mu}) = \frac{1}{N} \text{Var}_g \left\{ \frac{h(x)\pi(x)}{g(x)} \right\} = \frac{1}{N} \text{Var}_g \{ Z \},$$

where $Z = h(x)w(x)$.

- For the more useful one: The normalizing constants of $g$ and $\pi$ are not required, and $E_g w(x) = 1$.

$$E(\hat{\mu}) = E \left\{ \frac{Z}{W} \right\} \approx \mu - \text{Cov}(\tilde{Z}, \tilde{W}) + \mu \text{Var}(\tilde{W}),$$

and

$$\text{Var}(\hat{\mu}) \approx \frac{1}{N} \left[ \mu^2 \text{Var}_g(W) + \text{Var}_g(Z) - 2\mu \text{Cov}_g(W, Z) \right].$$
Hence, the mean squared error (MSE) of $\bar{\mu}$ is

$$SE(\bar{\mu}) = \frac{1}{N}\text{Var}_g\{Z\},$$

and that for $\hat{\mu}$ is

$$MSE(\hat{\mu}) = [E_g(\hat{\mu}) - \mu]^2 + \text{Var}_g(\hat{\mu})$$

$$= \frac{1}{N}MSE(\bar{\mu}) + \frac{1}{N}[\mu^2\text{Var}_g(W) - 2\mu\text{Cov}_g(W, Z)] + O(N^{-2})$$

Then MSE($\hat{\mu}$) is smaller in comparison with MSE($\bar{\mu}$) when $\mu^2\text{Var}_g(W) - 2\mu\text{Cov}_g(W, Z) < 0$. 
**Properly weighted samples**

The set \( \{(X_i, W_i)\}_{i=1}^N \) is said "properly weighted with respect to \( \pi \) if for any \( h(x) \), \( \mu = E_\pi h(x) \) can be approximated as

\[
\hat{\mu} = \frac{1}{\sum_{i=1}^N w_i} \sum_{i=1}^N w_i h(X_i).
\]

Mathematically, this says that

\[
\frac{E[W h(X)]}{E(W)} = E_\pi h(x).
\]

**Rule of thumb in importance sampling:**

\[
n_{\text{eff}} \approx \frac{n}{1 + \text{Var}_\pi(W)}.
\]
Summary of Importance Sampling

- Of interest $\mu = \int h(x)\pi(x)dx$.

- We can (and sometimes prefer to) draw samples from $g(x)$, which concentrates on “region of importance” and has a larger support set than that of $\pi(x)$.

- Let $X_1, \ldots, X_N$ be samples from $g(x)$, then

  $$\hat{\mu} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i h(X_i),$$

  where $w_i = \pi(X_i)/g(X_i)$. 

Sequential Importance Sampling

- Suppose $\mathbf{x} = (x_1, \ldots, x_k)$; of interest is
  \[ \pi (\mathbf{x}) = \pi (x_1) \pi (x_2 | x_1) \cdots \pi (x_k | x_1, \ldots, x_{k-1}) \]

- The trial density can also be decomposed similarly
  \[ g (\mathbf{x}) = g_1 (x_1) g_2 (x_2 | x_1) \cdots g_k (x_k | x_1, \ldots, x_{k-1}) \]

  \[ u (\mathbf{x}) = \frac{\pi (\mathbf{x})}{g (\mathbf{x})} = \frac{\pi (x_1) \pi (x_2 | x_1) \cdots \pi (x_k | x_1, \ldots, x_{k-1})}{g_1 (x_1) g_2 (x_2 | x_1) \cdots g_k (x_k | x_1, \ldots, x_{k-1})} \]

  - The weight can be computed sequentially
  - In practice, we create a sequence of approximations, $\pi (x_1)$, $\pi (x_1, x_2)$, $\cdots$, to guide the IS at each stage.
Sequential Importance Sampling: weight calculation

- Generate $x_1 \sim g_1(x_1)$, set the weight

$$w_1(x_1) = \frac{\pi_1(x_1)}{g_1(x_1)}.$$ 

- Generate $x_2 \sim g_2(x_2|x_1)$, set the weight

$$w_2(x_1, x_2) = w_1(x_1) \frac{\pi_2(x_2|x_1)}{g_2(x_2|x_1)}.$$ 

- Generate $x_k \sim g_k(x_k|x_1, \ldots, x_{k-1})$, set the weight

$$w(x_1, \ldots, x_k) = w_{k-1}(x_1, \ldots, x_{k-1}) \frac{\pi(x_k|x_1, \ldots, x_{k-1})}{g_k(x_k|x_1, \ldots, x_{k-1})}.$$
Variations of Sequential importance sampling (Grassberger, 1997)
Set upper cutoff values $C_t$ and lower cutoff values $c_t$ for $t = 1, \ldots k$.

- **Enrichment**: If $w_t > C_t$, the chain $x_t = (x_1, \ldots, x_t)$ is split into $r$ copies, each with weight $w_t/r$.

- **Pruning**: If $w_t < c_t$, one flips a fair coin to decide whether to keep it. If it is kept, its weight $w_t$ is doubled to $2w_t$.

A smart way to enrich the good partial conformations, and prune the bad ones.

**Similar idea**: resampling.
Related works

- Sequential importance sampling with partial rejection control (Liu, Chen and Wong, 1998)
- Dynamically weighted importance sampling (Liang, 2002)
Example: Polymer Simulation
HP model: HHHHHHPPPPHHPPHHHHHHH
H: hydrophobic amino acid (nonpolar);
P: hydrophilic amino acid (polar).
Self-avoid random walk: fold a protein on a 2D square lattice.

- Add one new position a time to one of the \( k \) \((k \leq 3)\) available positions.

- Question: sampling distribution of the chain?

\[
g(\text{a straight chain}) = \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3}
\]
Self-avoid random walk
85-Mer sequence