Lecture 9: Numerical Integration, Differentiation and Random Number Generation

Yuya Takahashi

May 28, 2019

1 Introduction

- Consider the probit model with likelihood

$$L(\theta; X, D) = \prod_{i=1}^{n} \Phi\left(\frac{-X_i\beta}{\sigma}\right)^{\mathbf{1}(D_i=1)} \Phi\left(\frac{X_i\beta}{\sigma}\right)^{\mathbf{1}(D_i=0)}$$

  and loglikelihood

$$\mathcal{L}(\theta; X, D) = \sum_{i:D_i=1} \ln\Phi\left(\frac{-X_i\beta}{\sigma}\right) + \sum_{i:D_i=0} \ln\Phi\left(\frac{X_i\beta}{\sigma}\right)$$

  with a normalization $\sigma = 1$.

- In order for us to be able to evaluate this likelihood, we need to evaluate the normal cdf. This is given by the integral

$$\Phi\left(-X_i\beta\right) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{-X_i\beta} e^{-\frac{1}{2}s^2} ds$$

  which does not have an analytical solution. This illustrates the point that we need to learn how to evaluate integrals numerically.

2 Newton-Cotes Formulas

- Let f be a continuous function whose domain includes the closed interval $[x_1, x_{N+1}]$. We now investigate ways of approximating the definite integral

$$\int_{x_1}^{x_{N+1}} f(x)dx$$

- The most straightforward numerical integration technique would be to approximate the integral by the use of Riemann sums.

- Approximations obtained in this way are also called Newton Cotes formulas. In general we do not use them but they help introduce the idea.

- If we divide the interval into $N+1$ equally spaced points of size Δx, a Riemann approximation to the integral would be

$$L_N = \sum_{j=1}^{N} f\left(x_j\right)\Delta x$$

if we evaluate using the left hand endpoints.

- The right hand endpoints rule

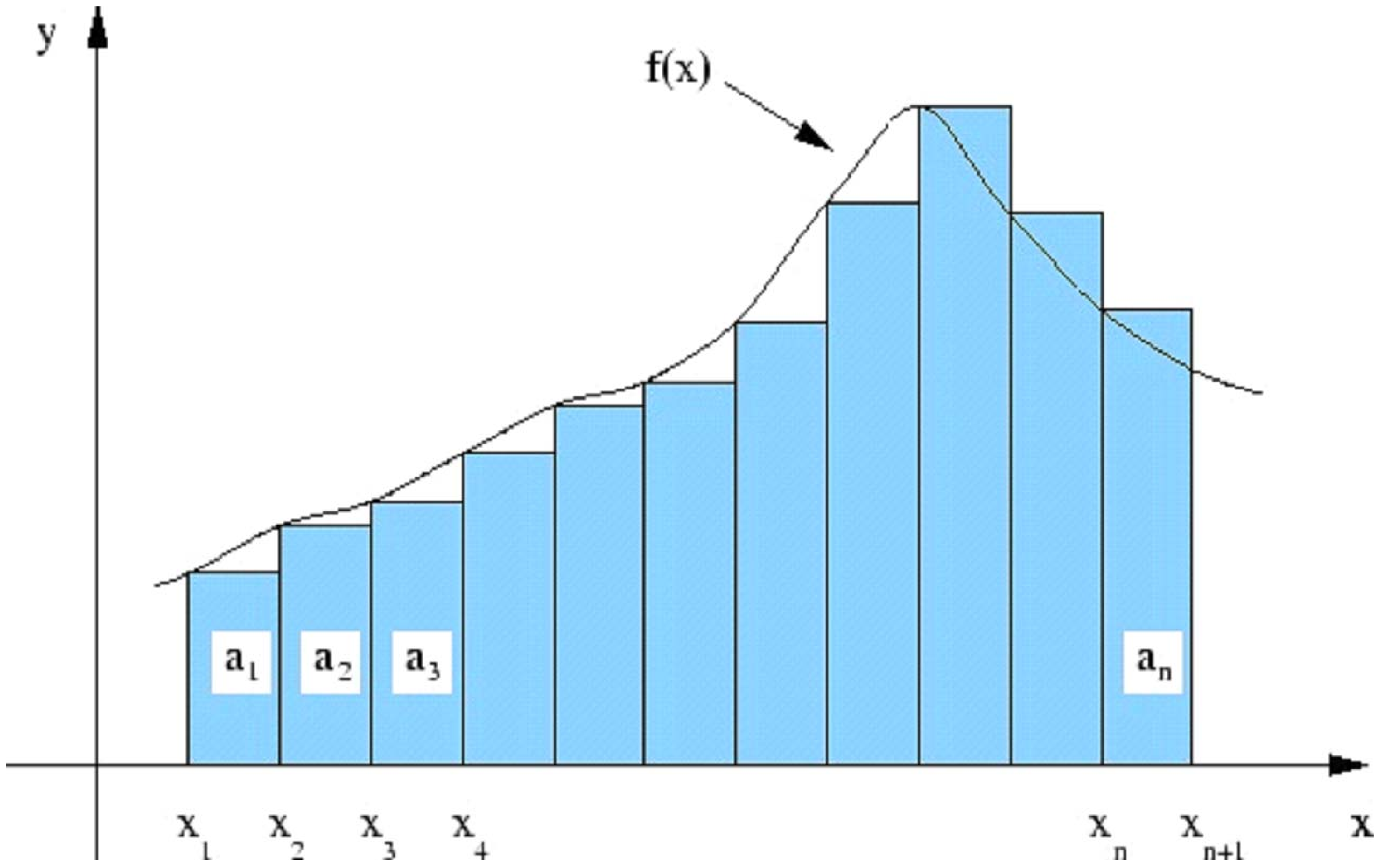
$$R_N = \sum_{j=1}^{N} f\left(x_{j+1}\right) \Delta x$$

- The midpoint rule

$$M_N = \sum_{j=1}^{N} f\left(\frac{x_{j+1} + x_j}{2}\right) \Delta x$$

- This type of approximation are also called rectangular rules since all we are doing is approximating the integral by summing rectangular areas.

- It is clear that not choosing a sufficiently small step-size Δa can lead to systematic error


y
f(x)
a_1
a_2
a_3
a_n
x_1
x_2
x_3
x_4
x_n
x_{n+1}
x

- The choice of the interpolating function can be refined to obtain more accuracy with fewer evaluations of the function $f(x)$

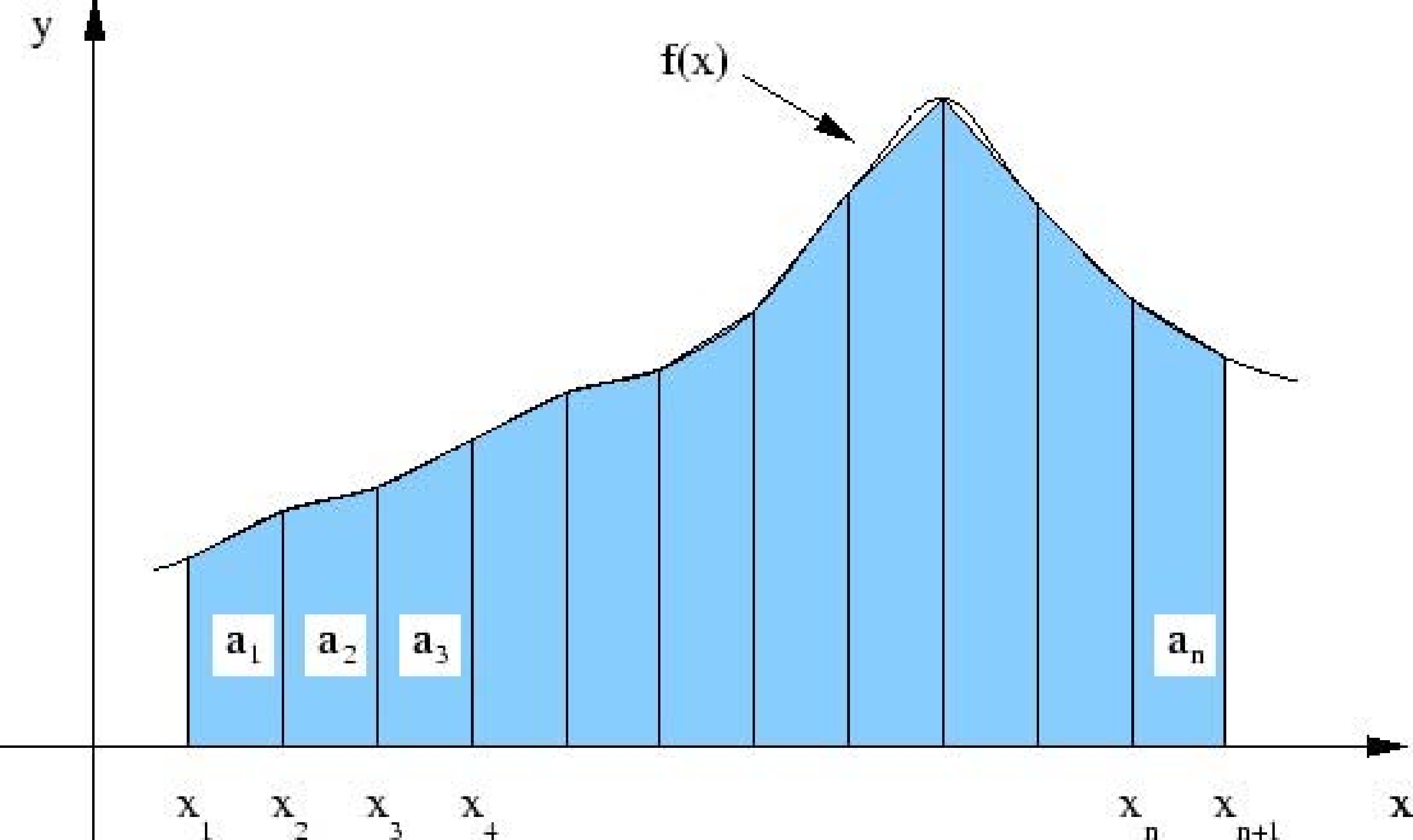
- The trapezoidal rule uses trapezoids (i.e., a rectangle with a triangle on top) instead of rectangles to approximate the area. The trapezoidal rule takes the form

$$T_N = \sum_{j=1}^{N} \frac{f\left(x_{j+1}\right) + f\left(x_j\right)}{2} \Delta x = \frac{L_N + R_N}{2}$$

- If instead of approximating the integral with a straight line between two points, we approximate with a polynomial on three points, we obtain Simpson's $\frac{1}{3}$ rule

$$S_N = \frac{T_N + 2M_N}{3}$$

- That is, the integral between any two of our points $\int_{x_j}^{x_{j+1}} f(x)dx$ would be approximated by dividing the $[x_j, x_{j+1}]$ interval further in half. Call this middle


y
f(x)
a_1
a_2
a_3
a_n
x_1
x_2
x_3
x_4
x_n
x_{n+1}
x

point $x_{j,j+1}$ then

$$\int_{x_j}^{x_{j+1}} f(x)dx \approx \left(x_{j+1} - x_j\right) \frac{f(x_j) + 4f(x_{j,j+1}) + f\left(x_{j+1}\right)}{6}$$

- All of the preceding formulas assume that we know in advance how many points we want to use when evaluating the integral

- A natural refinement would be instead to start increasing the number of points until some convergence criterion (i.e., until the evaluation of a given interval changes by only ε) is reached.

- For example, for the trapezoidal rule the simple integration routine

$$T_N = \sum_{j=1}^{N} \frac{f(x_{j+1}) + f(x_j)}{2} \Delta x$$

where N is known in advance. So we need to find N

- Routine:

    1. Define $T^{old} = 0$
    2. For $j = 1, ..., J_{MAX}$
        (a) Calculate
        $$T^{new} = \sum_{i=1}^{j} \frac{f\left(x_{i+1}\right) + f\left(x_i\right)}{2} \Delta x_j$$
        (b) If $|T^{new} - T^{old}| \leq \varepsilon |T^{old}|$ then exit the loop and finish. Otherwise
        (c) Redefine $T^{old} = T^{new}$ and continue the loop.

- The j subscript on Δx means we adjust it so that the interval of integration is divided into j equally spaced intervals of size Δx_j

- Up to now we have dealt with proper integrals

- To deal with integrals that have infinite limits we would need a routine that does not need evaluation at the endpoints

- The trapezoid rule we have been using until now is of no use to us for this case

- Instead we could use a generalization of the midpoint rule

- The key idea behind evaluating improper integrals is to make a change of variables so as to map an infinite range of integration into a finite one

- A good example that works for functions that decrease toward infinity faster than $\frac{1}{x^2}$ is

$$\int_a^b f(x)\,dx = \int_{\frac{1}{b}}^{\frac{1}{a}} \frac{1}{t^2} f\left(\frac{1}{t}\right) dt$$

  for $ab > 0$.

- This change of variables can be used for integrals between $b = \infty$ and $a > 0$ or $a = -\infty$ and $b < 0$.

- In general you should not use this Newton Cotes formulas for either definite or indefinite integrals. This is a good introduction to approximating integrals but the preferred method is to use Gaussian Quadratures

3 Gaussian Quadrature

- All Newton-Cotes rules are of the form
$$\int_a^b f\,(x)\,dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$
for some nodes $x_i \in [a,b]$ and weights ω_i. Then, we chose the coefficients ω_i so that the local approximation is correct and the nodes x_i are chosen arbitrarily; usually x_i are equidistant.

- However, if one has the freedom to choose not only the weights but also the points at which to evaluate $f(x)$, a careful choice can, in principle, lead to much higher accuracy in evaluating the integral in question

- That is, in theory, since we now have twice as many degrees of freedom, we can achieve formulas with almost twice the order as the Newton Cotes formulas for the same number of function evaluations

- An important thing to understand is that higher order does not automatically translate into higher accuracy

- Gaussian quadrature also approximates the integral by using a weighted sum

$$\int_a^b w(x) f(x) dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

  where the function $w(x)$ can be chosen so as to help us make the problem simpler (like eliminating singularity for example)

Result Suppose $\{\varphi_i(x)\}_{i=0}^{\infty}$ is an orthonormal family of polynomials with respect to $w(x)$ on $[a, b]$. Furthermore define q_i so that $\varphi_i(x) = q_0 + q_1 x + \cdots + q_i x^i$. Let $\bar{x}_i$, $i = 1, ..., n$ be the n zeros of $\varphi_n(x)$. Then $a < \bar{x}_1 < \bar{x}_2 < \cdots < \bar{x}_n < b$, and if $f \in C^{(2n)}[a, b]$, then

$$\int_a^b w(x) f(x) dx = \sum_{i=1}^{n} \omega_i f(x_i) + \frac{f^{(2n)}(\xi)}{q_n^2 (2n)!} \tag{1}$$

for some $\xi \in [a, b]$, where

$$\omega_i = -\frac{q_{n+1}/q_n}{\varphi_n'(\bar{x}_i)\,\varphi_{n+1}(\bar{x}_i)} > 0$$

- If f is a degree $2n-1$ polynomial, the last term in (1) drops out

- This tells us the necessary nodes and weights

- For several common $w(x)$ the quadrature rules (i.e., the polynomials we need to solve) are well known. The most common rules are:

  1. Gauss Legendre: $w(x) = 1$ on the interval $[-1, 1]$

  2. Gauss Chebyshev: $w(x) = (1-x^2)^{-\frac{1}{2}}$ on the interval $[-1, 1]$

  3. Gauss Hermite: $w(x) = e^{-x^2}$ on the interval $(-\infty, \infty)$

4. Gauss Laguerre: $w(x) = x^{\alpha}e^{-x}$ on the interval $(0, \infty)$ for a given α

5. Gauss-Jacobi: $w(x) = (1-x)^{\alpha}(1+x)^{\beta}$ on the interval $[-1, 1]$ for a given α and a given β

- For each of these quadratures, we have to find evaluation points $\{x_i\}_{i=1}^{n}$ and weights $\{\omega_i\}_{i=1}^{n}$. (See an example below)

3.1 Gauss Chebyshev

- The formula

$$\int_{-1}^{1} f(x)\left(1-x^2\right)^{-\frac{1}{2}} dx = \frac{\pi}{n}\sum_{i=1}^{n} f(x_i) + \frac{\pi}{2^{2n-1}}\frac{f^{(2n)}(\xi)}{(2n)!}$$

for some $\xi \in [-1, 1]$, where the quadrature nodes are

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right),\ i = 1, ..., n.$$

- Note that the weight is constant for each node ($\frac{\pi}{n}$).

- Usually, this formula is not directly applicable because instead of evaluating $\int_{-1}^{1} f(x)\left(1-x^2\right)^{-\frac{1}{2}} dx$, we want to compute $\int_a^b f(x)dx$ where

  1. The range of integration is $[a, b]$ instead of $[-1, 1]$
  2. $\left(1-x^2\right)^{-1/2}$ is missing

- To take care of (1), here is a review of how to do change in variables using the Jacobian of the transformation method

- Suppose you want to do an indefinite integral using a rule that gives you points between -1 and 1:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-1}^{1} f\left(\rho(y)\right)\frac{d\rho}{dy}dy$$

- Let

$$y = \rho^{-1}(x) = \frac{2}{1+e^{-x}} - 1$$

  so that as $x \to \infty$ we have $y \to 1$ and as $x \to -\infty$ we have $y \to -1$

- In other words, our transformation is

$$x = \rho(y) = -\ln\left(\frac{1-y}{1+y}\right) = -\ln(1-y) + \ln(1+y)$$

- The derivative (Jacobian)

$$\frac{d\rho}{dy} = \frac{1}{1-y} + \frac{1}{1+y} = \frac{1-y+1+y}{(1-y)\,(1+y)} = \frac{2}{(1-y^2)}$$

- Coming back to the Gauss-Chebyshev quadrature:

- Use the linear change of variables $x = -1 + 2(y-a)/(b-a)$ and multiply the integrand by $(1-x^2)^{-1/2}/(1-x^2)^{-1/2}$

- Then we have

$$\int_a^b f(y)dy = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{(x+1)(b-a)}{2} + a\right)\frac{\left(1-x^2\right)^{\frac{1}{2}}}{\left(1-x^2\right)^{\frac{1}{2}}}dx$$

- Use Gauss-Chebyshev quadrature to evaluate the RHS of this equation to get

$$\int_a^b f(y)dy \approx \frac{\pi(b-a)}{2n}\sum_{i=1}^{n} f\left(\frac{(x_i+1)(b-a)}{2} + a\right)\left(1-x_i^2\right)^{\frac{1}{2}}$$

where x_i are the Gauss-Chebyshev quadrature nodes over $[-1, 1]$.

3.2 Other Quadrature

- Gauss Legendre quadrature formula arises if we integrate f on the interval $[-1, 1]$ with the weighting function $w(x) = 1$

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} \omega_i f(x_i) + \frac{2^{2n+1}(n!)^4}{(2n+1)!\,(2n)!}\frac{f^{(2n)}(\xi)}{(2n)!}$$

for some $-1 \leq \xi \leq 1$.

- Gauss Hermite: $w(x) = e^{-x^2}$ on the interval $(-\infty, \infty)$ with

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}dx = \sum_{i=1}^{n} \omega_i f(x_i) + \frac{n!\sqrt{\pi}}{2^n}\frac{f^{(2n)}(\xi)}{(2n)!}$$

This formula arises naturally because Normal random variables are used often in economic problems

- Gauss Laguerre: $w(x) = x^{\alpha} e^{-x}$ on the interval $(0, \infty)$ for a given α with

$$\int_0^{\infty} f(x) e^{-x} dx = \sum_{i=1}^{n} \omega_i f(x_i) + (n!)^2 \frac{f^{(2n)}(\xi)}{(2n)!}$$

- Exponentially discounted sums are often used in economics

- Gauss-Laguerre quadrature can be used to compute the present value of infinitely long streams of utility or profits

4 Multidimensional Integrals

- Multidimensional integrals are, simply put, a very hard task using the methods we have shown so far for at least two reasons

- First, the number of function evaluations required to evaluate an integral in n dimensions grows to the n^{th} power of the number needed for a one dimensional integral

- That is, if we require 20 points to evaluate an integral in one dimension, we will require $20^3 = 8,000$ evaluations for an integral in 3 dimensions.

- Second, whereas the limits of integration for a one-dimensional integral are simply a pair of numbers, they can be very complicated functions when moving to a multidimensional case

- Consider the following two-dimensional integral of the form

$$\int_a^b \int_{c(x_2)}^{d(x_2)} f(x_2, x_1) dx_1 dx_2$$

- To evaluate this, define

$$F\left(x_2\right) = \int_{c(x_2)}^{d(x_2)} f(x_2, x_1) dx_1$$

  and

$$H = \int_a^b F\left(x_2\right) dx_2$$

- Then, it is simply a matter of applying any of our routines recursively to perform the multidimensional integration

4.1 Monte Carlo Integration

- Monte Carlo integration can sometimes be the only feasible answer to a hard multidimensional integration problem

- The idea behind Monte Carlo is simple: find a region with simple boundaries that includes the region of integration

- Then, find a simple method to sample random numbers inside the simple regions and then evaluate whether the resulting point is inside or outside the region of integration

- Suppose you want to evaluate the integral

$$\int_a^b f\left(x\right) dx$$

we know from before that this can be approximated by

$$\frac{(b-a)}{N}\sum_{i=1}^{N} f\left(x_i\right)$$

- The difference is that, instead of picking numbers at regular intervals or as the result of solving some polynomial, we pick them randomly in a region that contains our region of integration (where a point is a vector in the case of multidimensional integration)

- A simple example is the estimation of π by a two-dimensional Monte Carlo integration method

$$\begin{aligned}\pi &= \int_{-1}^{1}\int_{-1}^{1} p(x,y)dxdy \\ &\approx \frac{4}{N}\sum_{i=1}^{N} p(x_i, y_i)\end{aligned}$$

where

$$p(x, y) = \begin{cases} 1 & \text{if } x^2 + y^2 \le 1 \\ 0 & \text{otherwise} \end{cases}$$

- To estimate $\int_0^1 f(x)dx$, the crude Monte Carlo samples from $[0, 1]$ in a manner unrelated to the integrand $f(x)$; e.g. uniform distribution

- The value of f in some parts of $[0, 1]$ is much more important than in other parts, so the crude Monte Carlo can be very inefficient

- Sampling from a distribution other than the uniform distribution is sometimes called importance sampling

- Importance sampling tries to sample $f(x)$ where its value is important in determining $\int_0^1 f(x)dx$

- Consider $g(x)$ such that $\int_0^1 g(x)dx = 1$ and $g(x) > 0$

- Formal definition of Mote Carlo: the Monte Carlo Integration method consists of an independent sample of values $x_1, x_2, ..., x_N$ from a distribution $G(x)$ and an approximation to the integral

$$
\begin{aligned}
M &= \int_a^b f(x)dx \\
&= \int_a^b \frac{f(x)}{g(x)} g(x)dx
\end{aligned}
$$

  given by

$$
\widehat{M} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)}
$$

  The density $g(x)$ is called the importance function

- Notice that since M can be interpreted as the expected value of $\frac{f(x)}{g(x)}$, it follows from the law of large numbers that $\widehat{M}$ is a consistent estimator for it (so it converges at $\sqrt{N}$)

- Furthermore, it is straightforward to prove that it is also unbiased and that its variance is given by

$$var(\widehat{M}) = \frac{1}{N}\left(\int \frac{f^2(x)}{g(x)}dx - M^2\right)$$

- Notice that the variance of the estimator is an inverse function of the importance function $g(x)$ so an appropriate choice of g should lead to the smallest possible variance

- Simple example:

  • Suppose we want to simulate the mean of a standard normal distribution, truncated to $[0, 1]$. That is, we want to integrate

$$f(x) = \frac{x\phi(x)}{\int_0^1 \phi(z)\,dz}$$

  over $[0, 1]$

- A naive method: take draws x^s from $N(0,1)$ and only keep draws in $[0,1]$

- The simulated mean
$$\frac{\sum_{s=1}^{S} x^s \mathbf{1}(x^s \in [0,1])}{\sum_{s=1}^{S} \mathbf{1}(x^s \in [0,1])}$$
which could be very inefficient if $\sum_{s=1}^{S} \mathbf{1}(x^s \in [0,1])$ is much smaller than S

- Importance sampling: use draws from $U[0,1]$ so $g(x)=1$ for$x \in [0.1]$

- For each draw, the importance weight ω^s is
$$\omega^s = \frac{\phi\left(x^s\right)}{\int_0^1 \phi\left(z\right) dz}$$
and the simulated mean is $\frac{1}{S}\sum_{s=1}^{S} x^s \omega^s$

5 Numerical Differentiation

- The importance of computing numerical derivatives is clear

- The basis of numerical derivatives formulas is Taylor's Theorem

- For $n = 1$, it is true

$$f\left(\bar{x}+h\right) = f(\bar{x}) + f'\left(\bar{x}\right)h + f''\left(\xi\right)\frac{h^2}{2} \tag{2}$$

- The approximation of the first derivative is called the forward difference formula:

$$D_{FD}f\left(\bar{x},h\right) \approx \frac{f\left(\bar{x}+h\right) - f\left(\bar{x}\right)}{h}$$

- (2) implies the approximation error is proportional to h:

$$\left| D_{FD} f\left(\bar{x}, h\right) - f'\left(\bar{x}\right) \right| = \left| \frac{f''\left(\xi\right)}{2} \right| h \tag{3}$$

- Now apply the Taylor's Theorem for $n = 2$

$$\begin{aligned} f\left(\bar{x}+h\right) &= f(\bar{x}) + f'\left(\bar{x}\right) h + f''\left(\bar{x}\right) \frac{h^2}{2} + f^{(3)}\left(\xi_1\right) \frac{h^3}{6} \\ f\left(\bar{x}-h\right) &= f(\bar{x}) - f'\left(\bar{x}\right) h + f''\left(\bar{x}\right) \frac{h^2}{2} - f^{(3)}\left(\xi_2\right) \frac{h^3}{6} \end{aligned}$$

- Subtracting the second line from the first gives

$$f\left(\bar{x}+h\right) - f\left(\bar{x}-h\right) = 2f'\left(\bar{x}\right) h + \left(f^{(3)}\left(\xi_1\right) + f^{(3)}\left(\xi_2\right) \right) \frac{h^3}{6}$$

- The approximation of the first derivative is called the central difference formula:

$$D_{CD}f\left(\bar{x},h\right) \approx \frac{f\left(\bar{x}+h\right)-f\left(\bar{x}-h\right)}{2h}$$

- How do we choose h? (3) seems to imply that h should be as small as possible

- Remember that computation is never exact. Computing $f(x)$ involves some error

- Let $\bar{e}$ and e_h be the error in computing $f\left(\bar{x}\right)$ and $f\left(\bar{x}+h\right)$, respectively

- What is the total error when we use $\widetilde{f}\left(\bar{x}\right) \equiv f\left(\bar{x}\right)+\bar{e}$ and $\widetilde{f}\left(\bar{x},h\right) \equiv f\left(\bar{x}+h\right)+\bar{e}_h$ where $\bar{e}, e_h < \delta$?

- Let $E(g,h)$ denote the total error and calculate its upper bound as follows:

$$\begin{aligned} E(g,h) &= \left| f'(\bar{x}) - \frac{\widetilde{f}(\bar{x}) - \widetilde{f}(\bar{x}+h)}{h} \right| \\ &\leq \left| f'(\bar{x}) - D_{FD} f(\bar{x},h) \right| + \frac{|\bar{e} - e_h|}{h} \\ &\leq C_{FD} h + \frac{2\delta}{h} \end{aligned}$$

  where $C_{FD} = \max_{\xi \in [\bar{x}, \bar{x}+h]} \frac{f''(\xi)}{2}$.

- By taking the derivative of this upper bound w.r.t. h gives the step size that provides the smallest upper bound:

$$h^* = \sqrt{\frac{2\delta}{C_{FD}}}$$

- A similar result holds for the central difference formulas

$$h^{**} = \sqrt[3]{\frac{2\delta}{C_{CD}}}$$

where $C_{CD} = \max_{\xi_1,\xi_2 \in [\bar{x}, \bar{x}+h]} \frac{f^{(3)}(\xi_1)+f^{(3)}(\xi_2)}{6}$

- Using these results, we consider computing the Jacobian

- Let $f : R^n \rightarrow R^m$ and let $f^i(x)$, $x = [x_1, ..., x_n]$ denote the i-th component function of f

- The central difference is defined as

$$f^i_j = \frac{\partial f^i(\bar{x})}{\partial x_j} \approx \frac{f\left(\bar{x} + \mathbf{e}_j h\right) - f\left(\bar{x} - \mathbf{e}_j h\right)}{2h}$$

where $\mathbf{e}_j$ is the unit vector with one in the j-th position and zeros elsewhere

- In case x_i differs considerably in size, we can use h_j instead of common h

- For example, set

$$h_j = h^{**} \max\left\{|x_j|, 1\right\}$$

- These numerical derivative techniques are used to compute standard errors in non-linear models

- In the context of MLE, one needs to compute the score to estimate the variance-covariance matrix of the parameters (one of homework exercises)

6 Pseudo Random Number Generations

- The building block for any work on stochastic modeling or Monte Carlo technique is the pseudo uniform random number generator

- A uniform random deviate is just a random number that lies within a specified range (typically (0,1)) where any number is equally likely as any other

- There is a logical impossibility in generating random numbers from a computer that can only produce output from a program and is thus entirely predictable (computer scientists will debate this though)

- This is where the word “pseudo” comes to play

- From now on I will refer to pseudo random numbers simply as random numbers or random deviates, you however have been warned as to the “real” nature of these numbers

- Uniform random generators typically create their deviates by solving some deterministic difference equation $u_i = D(u_{i-1})$ given some initial value u_0 (the *seed* of the generator)

- Good generators produce sequences that are virtually indistinguishable from true i.i.d. sequences

- Since any program that you use for numerical computing will contain a uniform random number generator, there is no reason to write your own

- In theory, this is all you need since random numbers from any other distribution can be generated from the uniform distribution by means of the *inverse cdf method*.

Lemma If $u \sim U(0,1)$, then $F^{-}(u) \sim F$ where F^{-} is a generalized inverse of F. That is $F^{-}(u) = \inf\{x : F(x) \geq u\}$.

Proof We must show that

$$\mathsf{Pr}\left(F^{-}(u) \leq x\right) = F(x).$$

Suppose first that F is continuous and strictly increasing. Then

$$F^{-}(u) \leq x \;\Leftrightarrow\; u \leq F(x)$$

If F is either non decreasing or has discontinuities, then

$$\{x : F(x) \geq u\} = [F^{-}(u), \infty)$$

so

$$F^{-}(u) \leq x \;\Leftrightarrow\; u \leq F(x)$$

In any case it follows that

$$\mathsf{Pr}(F^{-}(u) \leq x) = \mathsf{Pr}(u \leq F(x)) = F(x).$$

- We can generate random numbers from any distribution by generating uniform random numbers first and then evaluating the inverse cdf at those values

- This is not always either feasible or the best way of generating random numbers

- Here is an example in which it works pretty well

Example Suppose we wish to sample from an exponential distribution with parameter λ

$$x \sim EXP(\lambda),\ \lambda > 0.$$

The cdf is

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

so

$$u = F(x) \Longrightarrow x = F^{-}(u) = -\frac{1}{\lambda}\ln(1-u)$$

- The algorithm is then:

1. Sample u from a uniform distribution on (0,1)

2. Define $x = -\frac{1}{\lambda} \ln (1 - u)$.

- An example of a distribution for which this method is not particularly recommended is the standard normal.

- In particular, we require to estimate the inverse cdf of a normal distribution to generate numbers of the form

$$x = \Phi^{-} (u)$$

- Not only can this be costly in terms of computation (although not that much) but it is rather inaccurate for numbers close to 0 or 1

- One way to generate random samples from the standard normal distribution is as follows:

1. Draw u_1, u_2 from $U[0, 1]$ independently

2. Use the transformation

$$x_1 = \cos(2\pi u_1)\sqrt{-2\ln u_2}$$
$$x_2 = \sin(2\pi u_1)\sqrt{-2\ln u_2}$$

Then, x_1 and x_2 are two independent draws from $N(0,1)$

- The final algorithm to sample random numbers from a given distribution that we discuss here is called the accept-reject algorithm.

- This method is based on sampling from a proposal distribution from which it is easy to sample and then accept or reject the proposed number based on some probability so that the accepted numbers have the correct distribution

- One advantage of this method is that you do not need to know the normalizing constant of the distribution you are trying to sample from.

- Let $f(x)$ be the pdf of the distribution we are trying to sample from and let $g(x)$ be the pdf of my proposal distribution

- Then, if we impose the following requirement

Requirement For the accept-reject algorithm to work a constant M such that

$$f(x) \leq Mg(x)$$

for all x in the support of f must exist.

- Algorithm:

  1. Generate $s \sim g$ and $u \sim U(0,1)$
  2. Accept $x = s$ if $u \leq \frac{f(x)}{Mg(x)}$
  3. Otherwise return to 1

- The main problem of accept-reject is finding a good proposal distribution so that we accept the proposed number as often as possible

- Notice that the probability of accepting a draw is

$$\begin{aligned}\mathsf{Pr}\left(u \leq \frac{f\left(x\right)}{Mg\left(x\right)}\right) &= \int_{-\infty}^{\infty}\int_{0}^{\frac{f(x)}{Mg(x)}} du g(x) dx \\ & \int_{-\infty}^{\infty}\frac{f(x)}{Mg(x)}g(x)dx \\ &= \frac{1}{M}\end{aligned}$$

so we want M to be as close to 1 as possible (i.e., we want the proposal and target distributions to be as similar as possible).

- Suppose we want to sample standard normal random numbers using accept-reject and we use a double exponential distribution

$$g(x) = \frac{\alpha}{2}e^{-\alpha|x|},\ \alpha > 0$$

We need to derive the most efficient bound M:

$$\begin{aligned}\frac{f(x)}{g(x)} &= \frac{\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}}{\frac{\alpha}{2}e^{-\alpha|x|}} \\ &= \sqrt{\frac{2}{\pi}}\alpha^{-1}e^{-\frac{1}{2}\alpha^2} \\ &\equiv M(\alpha)\end{aligned}$$

The α that minimizes this ratio is $\alpha = 1$.

- Finally, let us discuss how to draw a sample from a general normal distribution

- Single-dimensional variable from $N(\mu, \sigma^2)$

  1. Generate $u \sim N(0, 1)$

  2. Compute $x = \mu + \sigma u$

- K-dimensional variable from $N(\mu, \mathbf{\Sigma})$

  1. Compute the Choleski decomposition of $\mathbf{\Sigma}$, $\mathbf{\Sigma} = CC'$, where C is a lower triangular matrix

  2. Generate $u_j \sim N(0, 1)$, $j = 1, ..., K$ and let $u = (u_1, ..., u_K)'$

  3. Compute $x = \mu + Cu$