Computational Projects

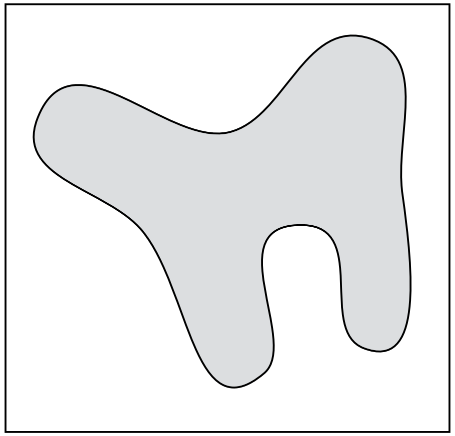
Lecture 7: Monte Carlo integration

Dr Rob Jack, DAMTP

http://www.maths.cam.ac.uk/undergrad/catam/part-ia-lectures

Numerical integration

Aim: Calculate area A of shaded region



The outer square is $\Omega = [0,1]^2$.

We are given some function $f : \Omega \to \mathbb{R}$ and the shape is

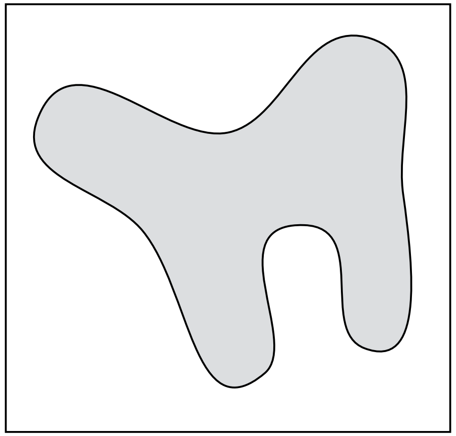
$$S = \{x \in \Omega : f(x) > 0\}$$

Let $\chi_S(x) = 1$ if $x \in S$ and $\chi_S(x) = 0$ otherwise.

We want $A = \int \chi_S \mathrm{d}x$.

Numerical integration

Aim: Calculate area A of shaded region



Simple method:

Partition the square into smaller squares of size h

The total number of these squares is $N_{\mathrm{tot}} = 1/h^2$

Count the number of squares n for which the central point is shaded

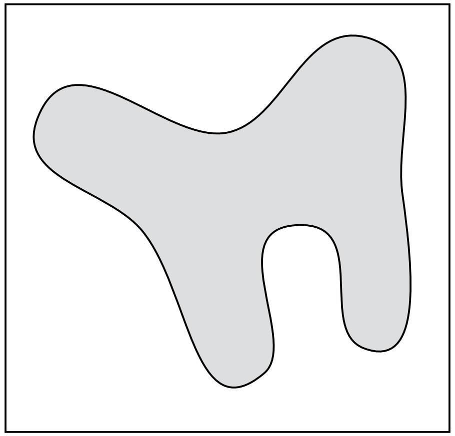
Estimate the area as $(n/N_{\mathrm{tot}}) = h^2 n$

Numerical effort is $O(h^{-2})$

Scaling of errors depends on the shape

Numerical integration

Aim: Calculate area A of shaded region



The *simple method* is ok but it gets very costly if we consider shapes in high dimensional spaces

In dimension d, the numerical effort to make a grid is $O(h^{-d})$

"Monte Carlo" method

Generate N random points in the outer square

If n of them are in the shaded region then estimate the area as

$$\frac{n}{N} A_0$$

where A_0 is the area of outer square

Monte Carlo integration

The name "Monte Carlo" comes from the famous casino, it was chosen because the method uses random numbers

It may seem like a bad idea to use a random process to estimate an integral (the answer obviously depends on the random numbers that we choose)

In fact such methods are used a lot. They have some nice properties:

On average, the method gives the right answer

$$\mathbb{E}(\text{estimated area}) = A$$

The variance of the estimated area goes to zero as $N \to \infty$

$$\mathrm{Var}(\text{estimated area}) = A(1-A)/N$$

(follows from the central limit theorem)

Monte Carlo integration

Let $\chi_i = 1$ if the ith random point is inside the chosen region and zero otherwise

$$\text{estimated area} = \frac{1}{N}\sum_{i=1}^{N} \chi_i$$

Use linearity of the expectation and that $\mathbb{E}(\chi_i) = A$, we get

$$\mathbb{E}(\text{estimated area}) = A$$

Since the random points are independent, we see that

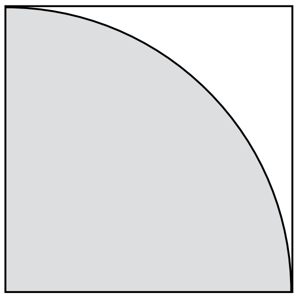
$$\text{Var}(\text{estimated area}) = N \times \text{Var}(\chi_i/N).$$

Using also that $\chi_i^2 = \chi_i$ we get

$$\mathrm{Var}(\text{estimated area}) = \frac{1}{N}(A - A^2) = \frac{A(1-A)}{N}$$

Spheres and hyperspheres

We can take $f(x) = 1 - \|x\|^2$ so that the shape is a quarter of a circle



Integral $A = \pi/4$

Area of unit circle $4A = \pi$

Now consider d spatial dimensions

Volume V_d of a d-dimensional hypersphere with unit radius can be obtained as $2^d A_d$

. . . where A_d is the analogous integral in d dimension

For even d then $V_d = \frac{\pi^{d/2}}{(d/2)!}$; for odd d then $V_d = \frac{\pi^{(d-1)/2}}{(1/2)\cdot(3/2)\cdots(d/2)}$

Example program

```
function [ estV,estFrac ] = sphereVolMC( dim,nn )
%sphereVolMC : MC for hypersphere volume
%   Monte Carlo estimate of volume of hypersphere in d dimensions
%   outputs the volume and the fraction of the corresponding hypercube
estV = 0.0;
for i=1:nn
    x = rand(dim,1);  % column vector with 'dim' random numbers
    xNorm2 = x' * x;  % squared norm of vector
                      % (remember x' is transpose of x)
    if ( xNorm2  < 1 )
       estV = estV + 1;
    end
end
estV = estV/nn;
% this is the integral (volume of one "quadrant")
estFrac = estV;
% mult by 2^dim to get the vol of the full hypersphere
estV = estV * 2^dim;
end
```

sphereVolMC.m , mcTest.m

Error analysis (random case)

In methods that use random numbers, there are two types of error

Random errors (or "statistical uncertainties") come from the specific choice of the random numbers. The standard deviation (square root of variance) is useful for quantifying them:

$$\text{Random error} = \sqrt{A(1-A)/N} = O(N^{-1/2})$$

Systematic errors (or "bias") can sometimes mean that the method does **not** give the right answer on average

$$\mathbb{E}(\text{estimated area}) = A + a$$

where a is the systematic error. In this case $a = 0$, the method is unbiased.

Evaluation of random methods requires analysis of **both** types of error

"Random" numbers

How does the computer generate random numbers?

> *"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"*
>
> von Neumann, 1951

In computing, so-called random numbers are usually *pseudo-random*

The computer generates these numbers by a deterministic rule, which is designed to mimic a sequence of i.i.d. random numbers

[Some computers can also use physical (hardware) devices to generate "true" random numbers]

Random "seed"

The computer generates sequences of (pseudo)-random numbers by a deterministic rule...

There are a finite number of possible sequences, we have to choose which sequence to use.

This is done by "seeding" the random number, which means choosing an integer (the "seed")

If we run the same program with the same seed, we get deterministic (reproducible) output. (Useful for debugging)

If we use different seeds, we get different outputs. A common trick is to use a seed related to the current date and time

`mcTest.m`

Other integrals

So far we estimated $\int_{[0,1]^2} \chi_S \mathrm{d}x$

It can be shown that if we have some $f : [0,1]^d \to \mathbb{R}$ then we can estimate $I = \int_{[0,1]^d} f \mathrm{d}x$ as

$$\text{Estimate of } I = \frac{1}{N}\sum_i f(x_i)$$

where the x_i are chosen independently at random from $[0,1]^d$.

The variance of our estimate is

$$\frac{1}{N}\int_{[0,1]^d} (f-I)^2 \mathrm{d}x$$

... if this last integral is finite then the method is reasonable...

Hyperspheres in high dimension

What happens for integrals in high dimension d (eg $d = 30$)?

For d even, we have $V_d = \frac{\pi^{d/2}}{(d/2)!} \approx \frac{1}{\sqrt{\pi d}} \exp\left[-\frac{d}{2}(\log d - C)\right]$ with $C = 1 + \log(2\pi)$.

... these are very small numbers when d is large

(Stirling's approximation)

For our method, the fraction of points inside the sphere is $A_d = 2^{-d} V_d$ which is even smaller...

If there are very few random points inside the sphere then the method will be inaccurate

Random error $\sqrt{\frac{1-A_d}{NA_d}}$ tends to be very large if A_d is small

(Note: methods based on "grids" are also impractical in high dimension)

Importance sampling

"Most" of the high-dimensional hypersphere is close to the origin. We can improve our method by putting more of our random points near the origin

Define some function (density) $\rho : [0,1]^d \to (0,\infty)$ with $\int_{[0,1]^d} \rho \, \mathrm{d}x = 1$. Then write

$$A = \int_{[0,1]^d} \chi_S \, \mathrm{d}x = \int_{[0,1]^d} \rho \cdot \frac{\chi_S}{\rho} \mathrm{d}x$$

If we choose our random points x_i to be distributed with density ρ then

$$\text{Estimate of } A = \frac{1}{N} \sum_{i=1}^{N} \frac{\chi_S(x_i)}{\rho(x_i)}$$

Importance sampling

We choose our random points x_i to be distributed with density ρ, and

$$\text{estimated area} = \frac{1}{N} \sum_{i=1}^{N} \frac{\chi_S(x_i)}{\rho(x_i)}$$

Use linearity of the expectation and that $\mathbb{E}_\rho(\frac{\chi_S(x)}{\rho(x)}) = A$, we get

$$\mathbb{E}(\text{estimated area}) = A$$

Since the random points are independent, we see that

$$\text{Var(estimated area)} = N \times \text{Var}_\rho\left(\frac{\chi_S(x)}{N\rho(x)}\right).$$

Using also that $\chi_S^2 = \chi_S$ and $\int \chi_S \mathrm{d}x = A$ we get

$$\text{Var(estimated area)} = \frac{1}{N} \int \chi_S \left(\frac{1}{\rho} - A\right) \mathrm{d}x = C/N, \quad C \geq 0$$

Random points

How to generate random points with a large density near the origin?

One possibility: fix $\lambda > 0$ and take v uniformly from $[\mathrm{e}^{-\lambda}, 1]$. Let

$$u = \frac{1}{\lambda} \log(1/v)$$

Cumulative distribution function for u

$$\text{Prob}(u < a) = \text{Prob}(v > \mathrm{e}^{-\lambda a}) = \frac{1 - \mathrm{e}^{-\lambda a}}{1 - \mathrm{e}^{-\lambda}}$$

(for $0 < a < 1$)

Probability density function for u

$$\rho(a) = \frac{\mathrm{d}}{\mathrm{d}a} \text{Prob}(u < a) = C\mathrm{e}^{-\lambda a}$$

with normalisation $C = \lambda/(1 - \mathrm{e}^{-\lambda})$.

Let every random point x_i have d random Cartesian components, each generated as an independent sample of u

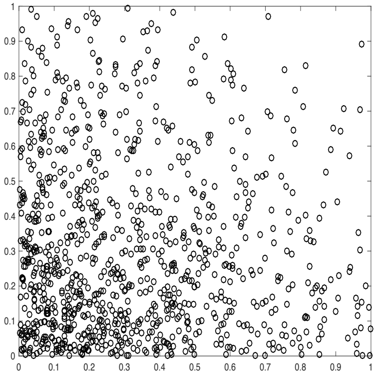
Example function

```
function [ x ] = ranExp( dim,lambda )
%ranExp: random vector in hypercube with exponential density

    % column vector with 'dim' random numbers (uniform from [0,1])
    y = rand(dim,1);
    % transform (elementwise) to random numbers in [exp(-lambda),1]
    y =  exp(-lambda) + (1-exp(-lambda))*y;
    % this gives us random numbers with exponential density
    x = -log(y)/lambda;   % note this is elementwise again

end
```

ranExp.m , mcImp.m



Importance sampling function

```
function [ estV,estFrac ] = sphereVolMCImpExp( dim,nn,lambda )
%sphereVolMCImpExp : MC for hypersphere volume with exp importance sampling
% ...

estV = 0.0;
for i=1:nn
    x = ranExp(dim,lambda);  % random point with specified density
    xNorm2 = x' * x;         % squared norm of vector

   if ( xNorm2  < 1 )
      % compute also the density at the relevant point
      % this is the product of the densities of the independent components
      rho = 1;
      for i=1:dim
         rho = rho * exp(-lambda*x(i)) * lambda / ( 1-exp(-lambda) );
      end

      estV = estV + 1/rho;
    end
end

estV = estV/nn;
% this is the integral (volume of one "quadrant")
estFrac = estV;
% mult by 2^dim to get the vol of the full hypersphere
estV = estV * 2^dim;
end
```

sphereVolMCImpExp.m

Does it work?

From mcImp.m we can see that this method works ok in 20 and 30 dimensions, where our original MC method fails

The relative error of the simple method scales as $\sqrt{\frac{1-A_d}{NA_d}}$

To obtain a relative error of ζ, we need

$$N = \zeta^{-2}\frac{1-A_d}{A_d} = O(\zeta^{-2})$$

For large d, remember that A_d is going to zero faster than exponentially

The importance sampling method still has $N = O(\zeta^{-2})$ but the constant in front is much smaller.

(The limits of small ζ and large d do not commute here...)

Conclusions (general)

We can build effective computational algorithms using random numbers (with a bit of probability theory)

For these algorithms we must evaluate both systematic error (bias) and random error (statistical uncertainty)

Computers usually work with pseudo-random numbers, you need to think about the "seed"

... next (final) lecture

general programming advice
... and some info on part IB computational projects