利用座標變換法

數學理論

在上一節曾經提到，當一個隨機變數的累積分布函數的反函數非常難求，甚至求不出來時，我們無法使用逆變換法來產生此分布的隨機變數。而標準常態隨機變數的累積分布函數的反函數就很難求，它的累積分布函數如下：

$$F(a)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{a} e^{\frac{-x^2}{2}}dx$$

現在介紹座標變換法。假定 X、Y 爲兩獨立標準常態隨機變數，它們的聯合機率分佈函數爲：

$$f_{X,Y}(x,y)=\frac{1}{2\pi}e^{\frac{-\left(x^2+y^2\right)}{2}}$$

現在考慮隨機變數 $D=X^2+Y^2$ 和 $\Theta=\tan^{-1}(Y/X)$ 的聯合機率分佈函數：

$$f_{D,\Theta}(d,\theta)=\frac{1}{2\pi}e^{\frac{-d}{2}}\frac{\partial(x,y)}{\partial(d,\theta)}$$

其中 $\frac{\partial(x,y)}{\partial(d,\theta)}$ 爲 Jacobian 矩陣的行列式值

$$\frac{\partial(d,\theta)}{\partial(x,y)}=\begin{vmatrix}\frac{\partial d}{\partial x} & \frac{\partial d}{\partial y}\\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y}\end{vmatrix}=\begin{vmatrix}2x & 2y\\ -y & x\\ \hline x^2+y^2 & x^2+y^2\end{vmatrix}=2$$

$$\therefore\ \frac{\partial(x,y)}{\partial(d,\theta)}=1\Big/\frac{\partial(d,\theta)}{\partial(x,y)}=1/2$$

$$\therefore\ f_{D,\Theta}(d,\theta)=\frac{1}{4\pi}e^{\frac{-d}{2}}\qquad 0<d<\infty,\ 0<\theta<2\pi$$

∴ D 和Θ爲獨立隨機變數，它們的密度函數爲：

$$f_D(d)=\frac{1}{2}e^{\frac{-d}{2}} \quad ; \quad f_\Theta(\theta)=\frac{1}{2\pi}$$

若 U_1 是一個介於 0 到 1 之間的均勻隨機變數，那麼 $-2\log U_1$ 的分布與 D 相同：

$$\begin{aligned} P\{-2\log U_1 < X\} &= P\{\log U_1 < -X/2\} \\ &= P\{U_1 > e^{-x/2}\} \\ &= 1-e^{-x/2} \end{aligned}$$

類似的，若 U_2 是一個介於 0 到 1 之間的均勻隨機變數，那麼 $2\pi U_2$ 的分布與Θ相同。因此，

$$X=\sqrt{D}\cos\Theta=\sqrt{-2\log U_1}\cos(2\pi U_2)$$

$$Y=\sqrt{D}\sin\Theta=\sqrt{-2\log U_1}\sin(2\pi U_2)$$

爲兩獨立常態隨機變數。

函式撰寫

用這種方法來寫標準常態隨機變數產生函式的好處之一是非常的簡單好寫。只需要用 rand() 產生兩個 0 到 1 之間的均勻隨機變數U_1、U_2，再計算 $X=\sqrt{-2\log U_1}\cos(2\pi U_2)$ 即可（即範例程式的第 26 行）。

使用這個方法另外還有一個好處：當蒙地卡羅法所需的執行次數很多時，使用電腦取亂數會發生重複抽取的問題，簡而言之，電腦所抽取的亂數列是個循環數列，每隔數十萬或百萬次就會循環一次，這樣會導致蒙地卡羅法在模擬次數太多時失靈。比起上節利用中央極限定理所撰寫的標準常態隨機變數產生器，此方法有效的提高亂數數列的循環週期，並且提高取樣的標準常態隨機變數的品質，請見 Normal1 project。讀者在產生常態隨機變數之前需執行程式碼如下

```
m_PI=3.14159265359;
m_RandomSeed=(unsigned)time(NULL);
```

然後就可直接呼叫 NormalGen()函式。在程式中 RandGen()可用來提高亂數序列的循環週期及隨機變數的取樣品質，NormalGen()則利用產生好的亂數建立常態隨機變數。其程式碼列之如下：

```
1.  double m_Max_Rnd, m_PI;
2.  long m_RandomSeed;
3.  #define IA 16807
4.  #define IM 2147483647
5.  #define AM (1.0/IM)
6.  #define IQ 127773
7.  #define IR 2836
8.  #define MASK 123459876
9.  ////
10. double RandGen(long *idum)
11. {
12.       long k;
13.       double ans;
14.       *idum^=MASK;
15.       k = (*idum)/IQ;
16.       *idum = IA*(*idum-k*IQ) - IR*k;
17.       if(*idum<0) *idum += IM;
18.       ans = AM*(*idum);
19.       *idum^=MASK;
20.       return  ans;
21. }
22. inline double NormalGen()
23. {
24.       double R1 = RandGen(&m_RandomSeed);
25.       double R2 = RandGen(&m_RandomSeed);
26.       double Normal = sqrt(-2*log(R1))*cos(2*m_PI*R2);
27.       return Normal;
28. }
```

另一種增進效率的方法

數學理論

上節所提到的方法一般稱爲 Box-Muller 方法。其之所以可行，完全由於一件事實：若 U_1、U_2 是兩個獨立、介於 0 到 1 之間的均勻隨機變數，那麼

$$X = \sqrt{-2\log U_1}\cos(2\pi U_2)$$

$$Y = \sqrt{-2\log U_1}\sin(2\pi U_2)$$

爲兩獨立常態隨機變數。

但是計算正弦和餘弦的值相當費時，降低此法效率。是有方法可以避免這些損失。如果我們找均勻分布在單位圓內的隨機點 (V_1,V_2)，該點和 X 軸正向所形成角度就是在（0，2π）之間均勻分布的隨機變數（和 $2\pi U_2$ 相同），而這個角度的正弦值不需要呼叫 cos() 函數，只要計算 $\frac{V_1}{\sqrt{V_1^2+V_2^2}}$ 即可。因此我們有：

若 U 是介於 0 到 1 之間的均勻隨機變數，且 (V_1,V_2) 爲單位圓內的均勻分布，則

$$X = \sqrt{-2\log U}\frac{V_1}{\sqrt{V_1^2+V_2^2}}$$

$$Y = \sqrt{-2\log U}\frac{V_2}{\sqrt{V_1^2+V_2^2}}$$

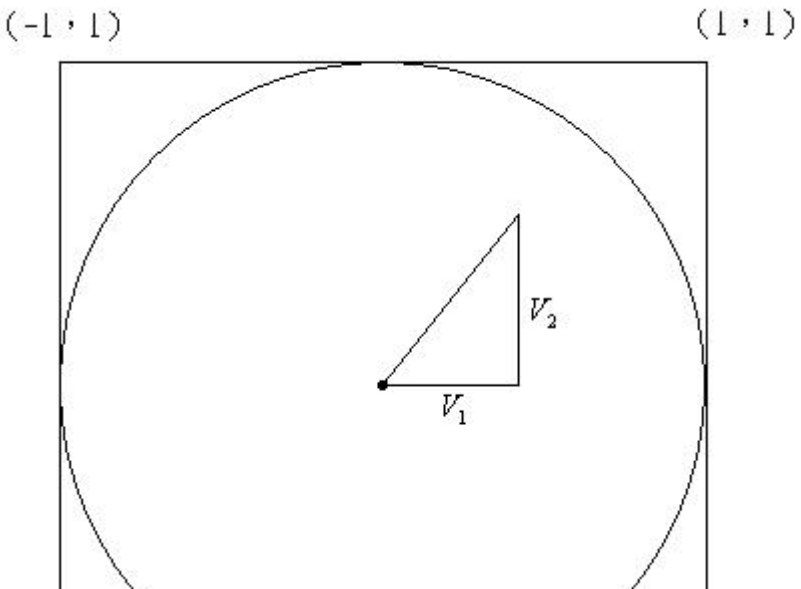
爲兩獨立常態隨機變數。

如何衍生單位圓內的隨機點？首先，若 U 是一個在（0，1）間均勻分布的隨機變數，那麼 2U-1 是在(-1，1)間的均勻分布。我們先衍生隨機變數 U_1、U_2，令

$V_1 = 2U_1 - 1$

$V_2 = 2U_2 - 1$

則 V_1、V_2 在以（0，0）爲中心，邊長爲 2 的正方形上均勻分布。我們不斷的衍生這種數對，但只有當 (V_1,V_2) 落在單位圓內才接受（即當 (V_1,V_2) 滿足 $V_1^2+V_2^2 \le 1$ 時）。如此則 (V_1,V_2) 爲均勻分布在單位圓內的點。



更進一步的，由於 $S = V_1^2 + V_2^2$ 是在（0，1）間均勻分布的隨機變數且獨立於 (V_1, V_2) 與 X 軸形成的角度，（當然也獨立於其餘弦值，即 $\frac{V_1}{\sqrt{V_1^2 + V_2^2}}$ ）

因此可取代 $X = \sqrt{-2\log U}\,\frac{V_1}{\sqrt{V_1^2 + V_2^2}}$ 式中的均勻隨機變數 U，而不需要另外再衍生新的均勻隨機變數。最後，原式變成：

$$X = \sqrt{-2\log S}\,\frac{V_1}{\sqrt{V_1^2 + V_2^2}} = \sqrt{\frac{-2\log S}{S}}V_1$$

$$Y = \sqrt{-2\log S}\,\frac{V_2}{\sqrt{V_1^2 + V_2^2}} = \sqrt{\frac{-2\log S}{S}}V_2$$

函式撰寫

簡單的說，可以用下列演算法來衍生標準常態隨機變數：

1. 衍生兩個（0，1）之間的均勻分布隨機變數 U_1、U_2，
2. 計算 $V_1 = 2U_1 - 1$、$V_2 = 2U_2 - 1$ 和 $S = V_1^2 + V_2^2$。（第 36～38 行）
3. 若 S>1，回到步驟 1，（第 39～40 行）
4. 計算 $X = \sqrt{\frac{-2\log S}{S}}V_1$ 和 $Y = \sqrt{\frac{-2\log S}{S}}V_2$（第 42～43 行）

X、Y 即爲兩獨立標準常態隨機變數。

以下程式是比較 Box-Muller 方法和這節所介紹的方法，程式先要求使用者輸入迴圈執行次數，然後用 Box-Muller 方法產生指定次數個標準常態隨機變數（第 19～29 行）然後再用本節介紹的方法產生同樣多個標準常態隨機變數（第 31～48 行）分別計算執行時間，然後顯示在螢幕上。實驗證實，本節所介紹的方法比起需要呼叫 cos() 函數的 Box-Muller 方法可以快上數倍，隨著蒙地卡羅法所需

的執行次數增加，節省下來的時間將會非常可觀。（例如，在 AMD Athlom64 2.01GHz，1.5GB RAM 的環境下，執行此程式的 19～29 行產生 20 萬個標準常態隨機變數費約時 0.06 秒，而 31～48 同樣產生 20 萬個標準常態隨機變數只花了 0.015 秒）

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <math.h>
4.  #include <time.h>

5.  int main()
6.  {
7.      double pi = 3.14159265358979323846;
8.      double u1, u2, norm1, norm2;
9.      double v1, v2, s;
10.     double begin, end;
11.     long int i;
12.     long int times;
13.
14.     printf("input times: ");
15.     scanf("%ld", &times );
16.
17.     // using cos & sin
18.
19.     begin = clock();
20.     srand((unsigned)time(NULL));
21.     for( i=0 ; i<times ; i++){
22.         u1 = (-1)*log( double(rand())/RAND_MAX );
23.         u2 = (-1)*log( double(rand())/RAND_MAX );
24.
25.         norm1 = sqrt(-2*log(u1))*cos(2*pi*u2);
26.         norm2 = sqrt(-2*log(u1))*sin(2*pi*u2);
27.     }
28.     end = clock();
29.     printf("%lf\n", (end-begin)/CLOCKS_PER_SEC );
30.
31.     // avoid cos & sin
32.
```

```
33.         begin = clock();
34.         srand((unsigned)time(NULL));
35.         for( i=0 ; i<times ; i++){
36.               v1 = 2*(double(rand())/RAND_MAX) - 1;       // 2*u1 - 1
37.               v2 = 2*(double(rand())/RAND_MAX) - 1;
38.               s = v1*v1 + v2*v2;
39.               if( s>1 ){
40.                     i--;
41.               }else{
42.                     norm1 = sqrt(-2*log(s)/s)*v1;
43.                     norm2 = norm1*v2/v1;
44.                     //norm2 = sqrt(-2*log(s)/s)*v2;
45.               }
46.         }
47.         end = clock();
48.         printf("%lf", (end-begin)/CLOCKS_PER_SEC );
49.
50.         return 0;
51.   }
```