8. 乱数とヒストグラム (指定された分布に従う乱数)

2016/11/28

1 準備

講義で説明した、変換法及び棄却法によって指定された分布に従う乱数を生成し、そのヒストグラムを作成し、確かに指定した分布に従っていることを確認しましょう。前回の実験でダウンロードした

`src\samples\Transform.java`
`src\samples\Rejection.java`

を使います。

2 変換法

定義域が $[a, b)$ である確率密度関数 $f(x)$ に対して、確率分布関数 $F(x)$ は

$$F(x) = \int_a^x f(y)\mathrm{d}y \tag{2.1}$$

です。この確率分布関数 $F(x)$ の逆関数 $F^{-1}(x)$ が得られる場合、変換法を用いて確率密度関数 $f(x)$ に従う乱数を生成することができます。

クラス `Transform` は、変換法で指定された分布に従う乱数を生成するクラスです。確率分布関数の逆関数 $F^{-1}(x)$ は、`java.util.function.Function` インターフェイスのインスタンスとして定義します。

`Function<Double, Double> invProDist` は、入力 (最初の `Double`) と戻り値 (二番目の `Double`) を指定した関数 `invProDist` の定義です。コンストラクタでは、$F^{-1}(x)$ を引数 `invProDist` として与えて、インスタンスを初期化します。

メソッド `getNext()` は、`Math.random()` によって一様乱数 $r \in [0, 1)$ を生成し、$F^{-1}(r)$ を戻り値として返します。`Function<Double, Double> invProDist` は、イ

ンターフェイスであり、関数そのものではないことに注意します。実際に使用する具体的な関数は、そのインターフェイスのメソッド `apply()` です。

課題 1　メソッド `getNext()` 内の記述を確認しましょう。

具体的な確率分布関数の逆関数 $F^{-1}(x)$ は、`main()` メソッドで定義されています。配布した例では、定義域 $[0,1)$ の指数確率密度

$$f(x) = Ae^{-x}, \quad A = \frac{e}{e-1} \tag{2.2}$$

に対応した、確率分布関数の逆関数

$$F^{-1}(r) = -\ln(1 - r/A) \tag{2.3}$$

を用いています。なお、java 中の `Math.log(x)` は自然対数 $\ln(x)$ を、`Math.E` は自然対数の底 e を数値的に与えます。

課題 2　式 (2.2) で定義される指数関数分布に従う乱数を生成し、その結果を gnuplot を用いて、期待される確率密度関数とともに図示することで、正しく乱数が生成できることを示しなさい。gnuplot 中で定数 e(自然対数の底) を数値的に与えるためには `exp(1)` とする。

3　棄却法

確率密度関数 $0 \le f(x) < m$ の不定積分である確率分布関数 $F(x)$ が得られ、さらにその逆関数が得られることは非常に稀な場合です。変換法が使えない場合に、指定した分布に従う乱数を生成するために用いられるのが棄却法です。

棄却法では、確率密度関数の定義域 $[a,b)$ に一様乱数 x を生成し、もう一つの一様乱数 $y \in [0,1)$ が $y < f(x)/m$ ならば x を採用し、それ以外の場合には棄却することで、確率分布関数 $f(x)$ に従う乱数を生成する方法です。棄却した場合には、次の一様乱数の組 (x,y) を生成して、乱数を得られるまで上記を繰り返します。

課題 3　`Rejection` は、棄却法で乱数を生成するクラスである。コンストラクタは、乱数の範囲、確率密度関数、及びその最大値を与る。メソッド `getNext()` を完成させなさい。

配布した `Rejection` クラスの `main()` メソッドでは、確率密度関数が

$$f(x) = \frac{2}{\pi}\sin^2(x), \quad x \in [0,\pi) \tag{3.1}$$

である場合に対応した場合が記述されています。Math.sin() は正弦関数を、Math.PI は π を表しています。

課題 4　式 (3.1) で定義される三角関数の二乗で定義される確率密度に従う乱数を生成なさい。その結果を gnuplot を用いて、期待される確率密度関数とともに図示することで、正しく乱数が生成できることを示しなさい。また、gnuplot では pi が π を数値的に与えます。

4　(発展)π の推計

```
int n = 1000;
int m = 0;
for( int i = 0 ; i < n ; i++ ){
    double x = Math.random();
    double y = Math.random();
    if( x * x + y * y < 1. ) m++;
}
double p = 4. * m / n;
```

Program 4.1: π の推計

乱数を使って π の値を推計することを考えましょう。区間 $[0, a)$ の乱数を二つ (x, y) を多数 (n 個) 生成します。このうち $x^2 + y^2 < 1$ を満たすものの数を m とします。m/n は、n が十分に大きければ、一辺の長さが 1 である正方形の面積に対する、半径 1 の扇型の面積の比、つまり $\pi/4$ となるでしょう。Program 4.1 は、そのようなことを行うプログラムの部分です。変数 p には、π の近似値が保存されるでしょう。

課題 5　n を 100 から、51200 まで、2 倍にしながら計算しなさい。p が π に近づく様子を、gnuplot でデータを可視化することで、示しなさい。