ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

KIV/VSP - Průběžná práce

Generování náhodných čísel

s exponenciálním rozdělením

Příklad: 1 / 4

Jiří Kučera (A08N0092P)
Narozen 15. 2. 1985
kalwi@students.zcu.cz

Zadání

Vytvořte generátor rozdělení jako funkci v jazyce Java, C či Pascal/Delphi (parametry rozdělení jsou zároveň parametry funkce generátoru) s využitím vhodné metody (inverzní transformace, kompoziční, vylučovací, atd.), napište hlavní program, který bude možné spustit s následujícím formátem parametrů (argumentů programu):

program (počet generovaných čísel) (parametr rozdělení1) [parametr rozdělení2] [...]

Sledované statistiky jsou: střední hodnota, rozptyl a charakter rozdělení – histogram.

Zadané rozdělení:

Exponenciální rozdělení (má 1 parametr "lambda").

Řešení:

Generování náhodných čísel

Ke generování náhodných čísel je ve většině programovacích jazyků k dispozici generátor náhodných čísel s rovnoměrným rozdělením pravděpodobnosti. Ke generování náhodných čísel s exponenciálním rozdělením jsem použil metodu inverzní transformace, která normální rozdělení transformuje na požadované podle distribuční funkce, resp. podle inverzní funkce k distribuční funkci:

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

Exponenciální rozdělení s parametrem λ má hustotu pravděpodobnosti:

$$f(x) = \begin{cases} 0 & pro\ x \leq 0 \\ \lambda e^{-\lambda x} & pro\ x > 0 \end{cases}$$

Distribuční funce je obecně (x ve funkci hustoty pravděpodobnosti je přeznačeno na t):

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

Pro exponenciální rozdělení tedy platí (pro x < 0 je F(x) = 0):

$$F(x) = \int_{0}^{x} \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x}$$

Inverzní funkce k uvedené distribuční funkci je:

$$F^{-1}(y) = -\frac{1}{\lambda} \ln(1 - y)$$

Budou-li se tedy dosazovat náhodně generovaná čísla do této funkce, rozdělení pravděpodobnosti vypočtených hodnot bude exponenciální.

Výpočet sledovaných hodnot

Teoretická hodnota střední hodnoty se podle její definice odvodí z hustoty rozdělení (nezapomeňme, že na (-∞, 0) je f(x) = 0):

$$E\{x\} = \int_{-\infty}^{\infty} xf(x)dx = \int_{0}^{\infty} \lambda x e^{-\lambda x} dx$$

Integraci provedeme metodou per partes:

$$E\{x\} = \begin{vmatrix} u = x\lambda, & du = \lambda dx \\ dv = e^{-\lambda x}dx, & v = -\frac{1}{\lambda}e^{-\lambda x} \end{vmatrix} = \left[-xe^{-\lambda x}\right]_0^{\infty} + \int_0^{\infty} e^{-\lambda x}dx = 0 - \left[\frac{1}{\lambda}e^{-\lambda x}\right]_0^{\infty} = \frac{1}{\lambda}$$

Obdobně vypočteme rozptyl:

$$D\{x\} = \int_{-\infty}^{\infty} x^2 f(x)dx - E^2(x) = \int_{0}^{\infty} x^2 \lambda e^{-\lambda x} dx - \frac{1}{\lambda^2}$$

Integrujeme opět per partes:

$$D\{x\} = \begin{vmatrix} u = \lambda x^2, & du = 2\lambda x \\ dv = e^{-\lambda x}, & v = -\frac{1}{\lambda}e^{-\lambda x} \end{vmatrix} - \frac{1}{\lambda^2} = \left[-x^2 e^{-\lambda x}\right]_0^{\infty} + \frac{1}{\lambda}\int_0^{\infty} 2\lambda x e^{-\lambda x} dx - \frac{1}{\lambda^2}$$

$$D\{x\} = 0 + \frac{2}{\lambda}\cdot\frac{1}{\lambda} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2}$$

Získali jsme tedy výsledné teoretické vztahy pro výpočet střední hodnoty a rozptylu:

$$E\{x\} = \frac{1}{\lambda}, \qquad D\{x\} = \frac{1}{\lambda^2}$$

Nejlepším odhadem střední hodnoty je aritmetický průměr:

$$E\{x\} \cong \overline{x} = \frac{1}{n}\sum_{1}^{n} x_i$$

Tento vzorec je rovnou možno použít pro průběžný výpočet bez ukládání generovaných čísel. Jelikož dopředu známe počet generovaných čísel, je možno ukládat přímo součty x_i / n. Při neznalosti n před dokončením výpočtu by při ukládání pouze x_i hrozilo přetečení sumační proměnné, a bylo by proto nutno vzorec ještě upravit.

Odhad rozptylu se provede jako aritmetický průměr čtverce odchylky od střední hodnoty. Pro průběžný výpočet je ale nutno základní vzorec upravit, aby nebylo nutno pamatovat si vygenerovaná čísla. Podobně jako u střední hodnoty je díky znalosti počtu generovaných čísel možno ukládat rovnou podíl čtverců a n a nehrozí tudíž přetečení:

$$D\{x\} \cong (x_i - \overline{x})^2 = \frac{1}{n}\sum_{1}^{n}(x_i - \overline{x})^2 = \frac{1}{n}\sum_{1}^{n} x_i^2 - \overline{x}$$

Pro úplnost ještě nutno poznamenat, že při ukládání uvedených podílů do sumační proměnné by při velmi vysokém počtu generovaných čísel mohlo dojít ke ztrátě přesnosti podílu v důsledku příliš velkého dělitele. Při použití datového typu `double` v Javě a počtu generovaných čísel v řádech statisíců ale tento problém nehrozí.

Histogram

Zobrazovaný histogram je textový a při jeho realizaci bylo nutno vyřešit problém měřítek jeho dimenzí tak, aby pro libovolně zadané parametry byly z histogramu čitelné výsledky (tzn. aby např. velikosti sloupců nebyly tak malé, že by nebyly vůbec vidět, nebo aby naopak nepřetékaly z obrazovky).

Jelikož při exponenciálním rozdělení pravděpodobnosti se generují čísla v rozsahu 0 – ∞, je nutno histogram oříznout zprava. Ořez jsem stanovil v bodě E{x} * 3, kde E{x} je teoretická střední hodnota. Nad tuto mez je již hustota pravděpodobnosti tak malá, že by hodnoty v histogramu již nebyly vidět. Vygenerovaná čísla nad tuto mez se v histogramu ignorují.

Výška sloupců je normalizovaná tak, aby nejvyšší sloupec měl vždy stejnou předem danou výšku (konkrétně 20 textových hvězdiček) a ostatní sloupce pak poměrnou výšku vzhledem k nejvyššímu sloupci.

Uživatelská příručka

Pro běh aplikace je vyžadována platforma Java.

Aplikace se spouští z příkazové řádky a jako argumet je možno zadat parametry výpočtu:

```
run.bat [<počet_generovaných_čísel> <lambda>]
```

Parametry je možno zadat buď oba dva, nebo žádný. Není-li zadán žádný parametr, spustí se výpočet dvakrát pro 100000 vygenerovaných čísel a hodnotu lambda náhodně zvolenou z intervalu (0, 10).

Po spuštění program vypíše teoretické a vypočítané hodnoty střední hodnoty a rozptylu a zobrazí textově histogram, ve kterém každá řádka představuje sloupec histogramu a počet hvězdiček v řádce pak poměrnou výšku sloupce histogramu. Každá řádka je uvozena číslem, které vymezuje interval sloupce (od předchozí hodnoty).

Ukázka výstupu

E_teorie=0.3333333333333333
D_teorie=0.1111111111111111
E_vypocet=0.33284281334282245
D_vypocet=0.11057718055671079

Histogram (pocet = 1000000, lambda = 3.0):
0,1000: **********
0,2000: **********
0,3000: **********
0,4000: ********
0,5000: *****
0,6000: ****
0,7000: ***
0,8000: **
0,9000: *
1,0000: *

Výsledky

Počet generovaných čísel	lambda	Teoretická stř. hodnota	Vypočítaná stř. hodnota	Teoretický rozptyl	Vypočítaný rozptyl
1000	0.5	2	1.92267826	4	3.40760117
100000	0.5	2	1.99605255	4	3.98972525
1000	1	1	0.97366338	1	0.93527947
100000	1	1	0.99268241	1	0.99155438
1000	5	0.2	0.20216319	0.04	0.03800177
100000	5	0.2	0.20080578	0.04	0.04001811

Závěr

Vypočítané střední hodnoty a rozptyly odpovídají jejich teoretickým hodnotám. Přesnost výpočtu se zvyšuje s počtem generovaných čísel, protože použité sumační vzorce pro odhad veličin konvergují k teoretickým hodnotám.

Příloha – výpis programu

Spouštěcí třída Main.java

```
package vsp.uloha14;

import java.util.Random;

/**
 * 1. uloha VSP
 * 4. priklad: Generovani nahodnych cisel - exponencialni rozdeleni
 *
 * @author Jiri Kucera (kalwi@students.zcu.cz)
 */
public class Main {

    public static void main(String[] args) {

        try {
            if (args.length == 0) {
                Random rand = new Random();
                double lambda;

                lambda = rand.nextDouble() * 10;
                compute(100000, lambda);

                lambda = rand.nextDouble() * 10;
                compute(100000, lambda);

            } else if (args.length == 2) {
                compute(Integer.valueOf(args[0]), Double.valueOf(args[1]));
            } else {
                throw new Exception();
            }

        } catch (Exception e) {
            System.out.println("Pouziti:\njava -jar vsp.jar <pocet> <lambda>");
        }

    }

    /**
     * Spusti vypocet s uvedenymi parametry
     * @param count pocet generovanych cisel
     * @param lambda parametr lambda exponencialniho rozdeleni
     */
    private static void compute(int count, double lambda) {
        Statistics s = new Statistics(count, lambda);

        System.out.println("E_teorie=" + s.getTheoreticalExpectedValue());
        System.out.println("D_teorie=" + s.getTheoreticalVariance());
        System.out.println("E_vypocet=" + s.getEvaluatedExpectedValue());
        System.out.println("D_vypocet=" + s.getEvaluatedVariance());

        System.out.println("");

        System.out.println("Histogram (pocet = " + count + ", lambda = " + lambda + "):");

        int[] histogram = s.getHistogram();

        for (int i = 0; i < histogram.length; i++) {
            System.out.print(String.format("%8.4f", s.getHistogramLegend()[i]) + ": " );

            for (int j = 0; j < histogram[i]; j++) {
                System.out.print("*");
            }

            System.out.println("");
        }
        System.out.println("");
    }

}
```

Třída pro výpočet – Statistics.java

```
package vsp.uloha14;

import java.util.Random;

/**
 * Trida pro generovani nahodnych cisel a vypoctu statistik
 * @author Jiri Kucera (kalwi@students.zcu.cz)
 */
public class Statistics {
    private static Random random;

    private static final int H_SIZE = 10;       // pocet sloupcu histogramu
    private static final int H_COL_SIZE = 20;   // maximalni vyska sloupce histogramu

    private double expectedValueE = 0;
    private double varianceE = 0;

    private double expectedValueT = 0;
    private double varianceT = 0;

    private int[] histogram;
    private double[] histogramLegend;

    private double hMax;                // hranice, nad niz budou vygenerovana cisla ignorovana

    /**
     * Konstruktor vytvori objekt pro generovani cisel a vypocet statistik.
     * @param count pocet generovanych cisel
     * @param lambda parametr lambda exponencialniho rozdeleni
     */
    public Statistics(int count, double lambda) {

        random = new Random();

        expectedValueT = 1. / lambda;
        varianceT = 1. / (lambda * lambda);

        hMax = expectedValueT * 3;

        histogram = new int[H_SIZE];
        histogramLegend = new double[H_SIZE];

        for (int i = 0; i < H_SIZE; i++) {
            histogramLegend[i] = (i + 1) * hMax / (double) H_SIZE;
        }

        int highest = 1;
        double sum = 0;
        double sum2 = 0;

        // generovani cisel
        for (int i = 0; i < count; i++) {

            double number = generateRandom(lambda);

            sum += number / count;
            sum2 += number * number / count;

            // pridani do histogramu
            int hIndex = (int) (number * H_SIZE / hMax);

            // cisla mimo rozsah histogramu se ignoruji
            if (hIndex < H_SIZE) {
                histogram[hIndex]++;
                if (histogram[hIndex] > highest) {
                    highest = histogram[hIndex];
                }
            }
        }

        // stredni hodnota
```

```
            expectedValueE = sum;

            // rozptyl
            varianceE = sum2 - expectedValueE * expectedValueE;

            // normalizace histogramu
            for (int i = 0; i < H_SIZE; i++) {
                histogram[i] = histogram[i] * H_COL_SIZE / highest;
            }

        }

        /**
         * @return Vypocitana stredni hodnota
         */
        public double getEvaluatedExpectedValue() {
            return expectedValueE;
        }

        /**
         * @return Teoreticka stredni hodnota
         */
        public double getTheoreticalExpectedValue() {
            return expectedValueT;
        }

        /**
         * @return Vypocitany rozptyl
         */
        public double getEvaluatedVariance() {
            return varianceE;
        }

        /**
         * @return Teoreticky rozptyl
         */
        public double getTheoreticalVariance() {
            return varianceT;
        }

        /**
         * @return Pole s histogramem
         */
        public int[] getHistogram() {
            return histogram;
        }

        /**
         * @return Pole s popisky sloupcu histogramu
         */
        public double[] getHistogramLegend() {
            return histogramLegend;
        }

        /**
         * Vygeneruje nahodne cislo s exponencialnim rozdelenim pravdepodobnosti.
         * @param lambda parametr lambda exponencialniho rozdeleni
         * @return vygenerovane nahodne cislo
         */
        public static double generateRandom(double lambda) {
            return - Math.log(1 - random.nextDouble()) / lambda;
        }

    }
```