Проверка гипотезы о равномерности распределения генератора случайных чисел игры DOOM

Ковалев Д., СКБ171

Проверка гипотезы о равномерности распреде-ления генератора случайных чисел игры DOOM

Ковалев Даниил, СКБ171

Генератор случайных чисел, доступный по ссылке https://github.com/id-Software/DOOM/blob/master/linuxdoom-1.10/m_random.c, не зависит от специфичных для языка Си функций, поэтому перепишем его здесь на языке Python для удобства использования. Случайные числа, генерируемые функцией, должны подчиняться дискретному равномерному распределению на целых числах от 0 до 255.

[1]:
```
rndtable = [
      0,   8, 109, 220, 222, 241, 149, 107,  75, 248, 254, 140,  16,  66,
     74,  21, 211,  47,  80, 242, 154,  27, 205, 128, 161,  89,  77,  36,
     95, 110,  85,  48, 212, 140, 211, 249,  22,  79, 200,  50,  28, 188,
     52, 140, 202, 120,  68, 145,  62,  70, 184, 190,  91, 197, 152, 224,
    149, 104,  25, 178, 252, 182, 202, 182, 141, 197,   4,  81, 181, 242,
    145,  42,  39, 227, 156, 198, 225, 193, 219,  93, 122, 175, 249,   0,
    175, 143,  70, 239,  46, 246, 163,  53, 163, 109, 168, 135,   2, 235,
     25,  92,  20, 145, 138,  77,  69, 166,  78, 176, 173, 212, 166, 113,
     94, 161,  41,  50, 239,  49, 111, 164,  70,  60,   2,  37, 171,  75,
    136, 156,  11,  56,  42, 146, 138, 229,  73, 146,  77,  61,  98, 196,
    135, 106,  63, 197, 195,  86,  96, 203, 113, 101, 170, 247, 181, 113,
     80, 250, 108,   7, 255, 237, 129, 226,  79, 107, 112, 166, 103, 241,
     24, 223, 239, 120, 198,  58,  60,  82, 128,   3, 184,  66, 143, 224,
    145, 224,  81, 206, 163,  45,  63,  90, 168, 114,  59,  33, 159,  95,
     28, 139, 123,  98, 125, 196,  15,  70, 194, 253,  54,  14, 109, 226,
     71,  17, 161,  93, 186,  87, 244, 138,  20,  52, 123, 251,  26,  36,
     17,  46,  52, 231, 232,  76,  31, 221,  84,  37, 216, 165, 212, 106,
    197, 242,  98,  43,  39, 175, 254, 145, 190,  84, 118, 222, 187, 136,
    120, 163, 236, 249
]

rndindex = 0
```

[2]:
```
def doom_random():
    global rndindex
    rndindex = (rndindex + 1) % 256;
    return rndtable[rndindex]
```

[3]:
```
sample = [doom_random() for i in range(100000)]
```

Генератор, по сути, просто идет по порядку по значениям массива rndtable, переходя после последнего к первому. Несмотря на абсолютную детерминированность, отсутствие даже какого-либо seed и период всего 256, в игре генератор должен показывать себя хорошо, т.к. множество компонент игры одновременно использовали один и тот же генератор с одним и тем же счетчиком rndindex.

Распределение дискретное, поэтому для определения принадлежности выборки дискретному равномерному распределению от 0 до 255 будем использовать критерий хи-квадрат с уровнем значимости 0.05. Будем наращивать объем выборки, начиная с 100, и, когда 5 раз подряд отклоним гипотезу о принадлежности указанному распределению, остановимся. Будем считать, что начиная с n, на котором мы остановились, можно отличить случайную величину, выработанную генератором, от истинно случайной последовательности.

[4]:
```
import numpy as np
from scipy import stats
from scipy import optimize
from bisect import bisect_right
```

[5]:
```
def find_quantile_chi2(df):
    return optimize.bisect(lambda x: stats.chi2.cdf(x, df) - 0.95, 0,
↪10**9)
```

[6]:
```
def find_chi2(var_series, cdf, minimum=0, maximum=255):
    n = len(var_series)
    k = maximum + 1
    edges = np.arange(minimum, maximum + 1).astype(np.int32)
    p_arr, n_arr = [], []
    for i in range(k - 1):
        p_arr.append(cdf(edges[i + 1]) - cdf(edges[i]))
        n_arr.append(bisect_right(var_series, edges[i + 1]) -
↪bisect_right(var_series, edges[i]))
    p_arr, n_arr = np.array(p_arr), np.array(n_arr)
    return {'k': k, 'chi2': n * np.sum((p_arr - n_arr / n)**2 /
↪(p_arr))}
```

[7]:
```
def uniform_discrete_cdf(x):
    return int(x + 1) / 256
```

[8]:
```
false_cnt = 0
for n in range(100, len(sample), 25):
    print("Выборка объема {:5}: ".format(n), end="")
    ret = find_chi2(sorted(sample[:n]), uniform_discrete_cdf)
    k, chi2 = ret['k'], ret['chi2']
    print(" гипотеза ", end="")
    quantile = find_quantile_chi2(k - 1)
    if quantile > chi2:
        print("   принимается ({:6.3f} > {:6.3f})".format(quantile,
↪chi2))
        false_cnt = 0
    else:
```

```
        print("не принимается ({:6.3f} ≤ {:6.3f})".format(quantile,
↪chi2))
        false_cnt += 1
    if false_cnt >= 5:
        break
```

```
Выборка объема   100:  гипотеза    принимается (293.248 > 236.969)
Выборка объема   125:  гипотеза    принимается (293.248 > 245.152)
Выборка объема   150:  гипотеза    принимается (293.248 > 235.414)
Выборка объема   175:  гипотеза    принимается (293.248 > 232.991)
Выборка объема   200:  гипотеза    принимается (293.248 > 235.139)
Выборка объема   225:  гипотеза    принимается (293.248 > 231.232)
Выборка объема   250:  гипотеза    принимается (293.248 > 235.375)
Выборка объема   275:  гипотеза    принимается (293.248 > 268.784)
Выборка объема   300:  гипотеза не принимается (293.248 ≤ 306.988)
Выборка объема   325:  гипотеза не принимается (293.248 ≤ 346.481)
Выборка объема   350:  гипотеза не принимается (293.248 ≤ 382.404)
Выборка объема   375:  гипотеза не принимается (293.248 ≤ 406.410)
Выборка объема   400:  гипотеза не принимается (293.248 ≤ 429.397)
```

Искомое n больше 275, но не превосходит 300. Ищем более точно:

[9]:
```
false_cnt = 0
for n in range(276, len(sample)):
    print("Выборка объема {:5}: ".format(n), end="")
    ret = find_chi2(sorted(sample[:n]), uniform_discrete_cdf)
    k, chi2 = ret['k'], ret['chi2']
    print(" гипотеза ", end="")
    quantile = find_quantile_chi2(k - 1)
    if quantile > chi2:
        print("   принимается ({:6.3f} > {:6.3f})".format(quantile,
↪chi2))
        false_cnt = 0
    else:
        print("не принимается ({:6.3f} ≤ {:6.3f})".format(quantile,
↪chi2))
        false_cnt += 1
    if false_cnt >= 5:
        break
```

```
Выборка объема   276:  гипотеза    принимается (293.248 > 268.603)
Выборка объема   277:  гипотеза    принимается (293.248 > 268.416)
Выборка объема   278:  гипотеза    принимается (293.248 > 268.223)
Выборка объема   279:  гипотеза    принимается (293.248 > 269.860)
Выборка объема   280:  гипотеза    принимается (293.248 > 273.306)
Выборка объема   281:  гипотеза    принимается (293.248 > 273.077)
Выборка объема   282:  гипотеза    принимается (293.248 > 276.473)
Выборка объема   283:  гипотеза    принимается (293.248 > 278.029)
```

```
Выборка объема   284:  гипотеза    принимается (293.248 > 279.567)
Выборка объема   285:  гипотеза    принимается (293.248 > 279.290)
Выборка объема   286:  гипотеза    принимается (293.248 > 279.009)
Выборка объема   287:  гипотеза    принимается (293.248 > 278.722)
Выборка объема   288:  гипотеза    принимается (293.248 > 281.986)
Выборка объема   289:  гипотеза    принимается (293.248 > 286.992)
Выборка объема   290:  гипотеза    принимается (293.248 > 290.191)
Выборка объема   291:  гипотеза не принимается (293.248 ≤ 293.362)
Выборка объема   292:  гипотеза    принимается (293.248 > 292.996)
Выборка объема   293:  гипотеза не принимается (293.248 ≤ 294.374)
Выборка объема   294:  гипотеза не принимается (293.248 ≤ 293.994)
Выборка объема   295:  гипотеза не принимается (293.248 ≤ 295.346)
Выборка объема   296:  гипотеза не принимается (293.248 ≤ 296.682)
Выборка объема   297:  гипотеза не принимается (293.248 ≤ 296.278)
```

Итак, начиная с $n = 293$ можно отличить случайную величину, выдаваемую генератором Doom, от истинно случайной величины, распределенной дискретно равномерно на $[0, 255]$.