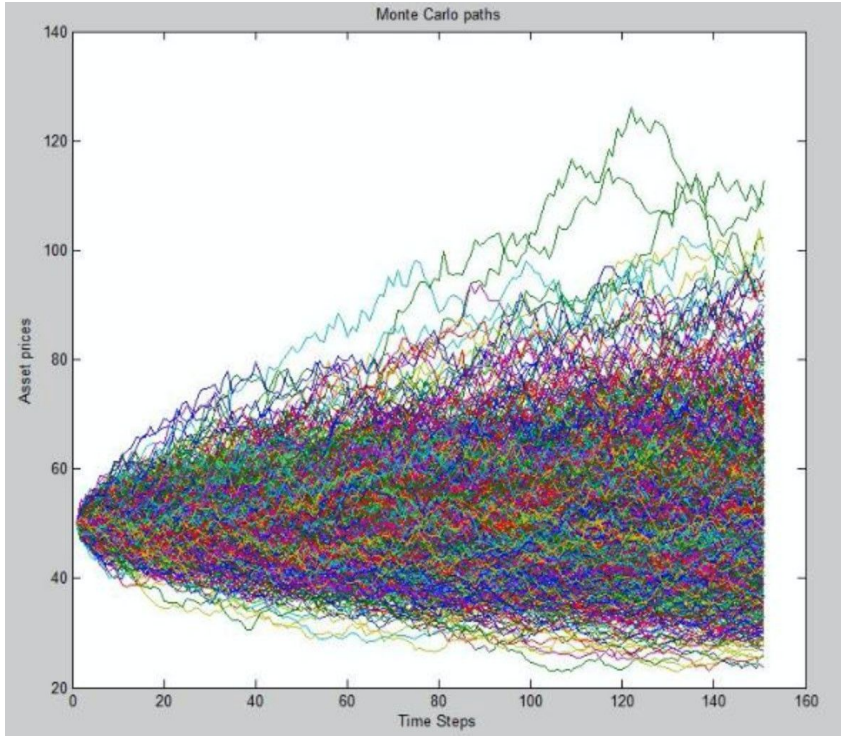
@ техносфера

Генерация псевдослучайных чисел при помощи нейросетей

Руслан Аллаяров
Никита Балабин
Даниил Бойко

Для чего нужны случайные числа?



Как генерируются случайные числа сейчас?

Зависит от применения

Используемые для симуляций

$s_{i+1}=(as_i+b) \bmod n$

Криптостойкие

Завязаны на куче параметров, затрудняя криптографические атаки

Как проверяют случайность чисел?

Birthday spacings

Overlapping permutations

Ranks of matrices

Monkey tests

Count the 1s

Parking lot test

Minimum distance test

Random spheres test

The squeeze test

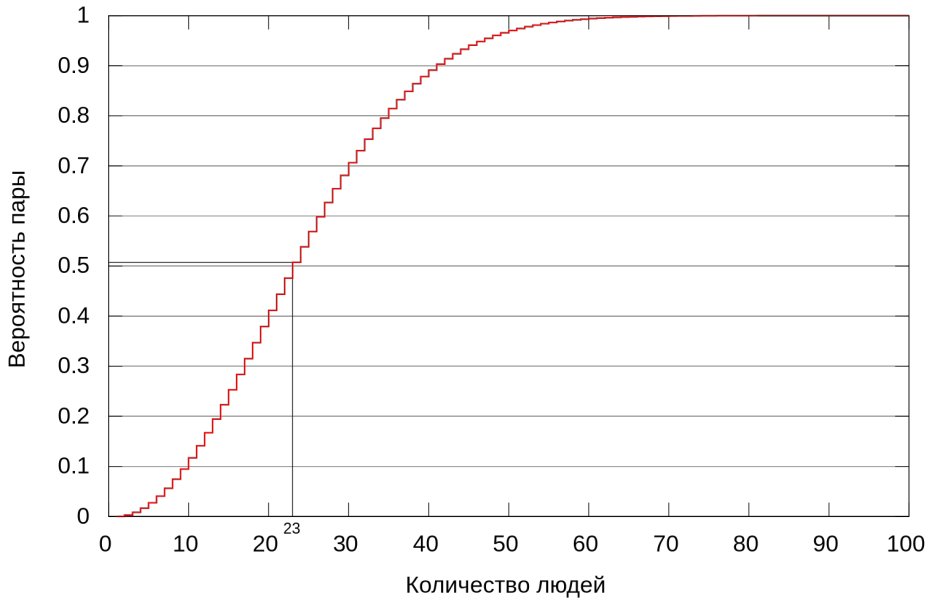
Overlapping sums test

Runs test

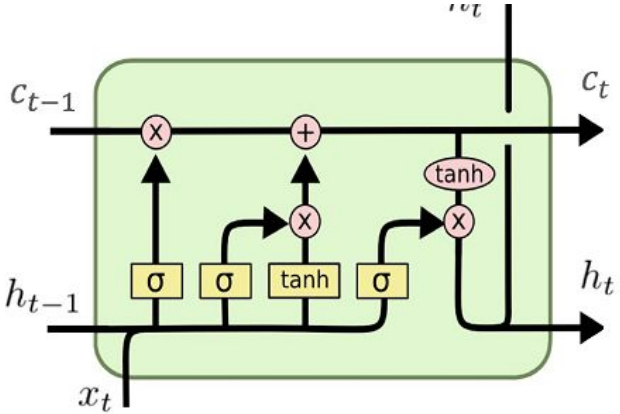
The craps test

Пример теста — Birthday spacings

выбираются случайные точки на большом интервале. Расстояния между точками должны быть асимптотически распределены по Пуассону



Как работает генерация при помощи LSTM-сети?



LSTM
(Long-Short Term Memory)

1 шаг - обучение модели

Есть число pi, используем его десятичное разложение.

По 100 цифрам пытаемся предсказать 101- ую цифру, используя LSTM.

Параметры: (10, 50, 10) - размерность входного слоя, скрытого, и выходного.

На все выходы навешиваем линейный слой, который уже предсказывает цифру

Итерационный генератор

Используются три последовательности : seed, buffer и input, которые итеративно меняются.
prns - это наша искомая последовательность псевдорандомных чисел.
Вначале inp = buffer = seed.
i-ая итерация:

1) seed shift :

k = LSTM(input);
sh = seed[k];
seed = seed[sh:s] + seed[0:sh], где s - длина шага
r_i = seed[0];

2) buffer sequence update:

buf.append(r_i);
buf = buf[r_i:b] + buf[0:r_i];

3) input sequence generation:

inp = buf[-s:0]
prns.append(r_i)

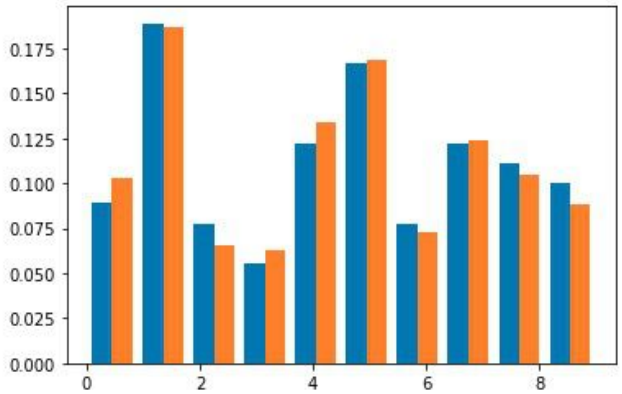
Количество итераций дает нам размер выходной последовательности сгенерированных чисел

Наши результаты

```
aniilboiko$ dieharder -g 202 -f 2.6_million.txt -a
#=============================================#
 Copyright 2003 Robert G. Brown                #
#=============================================#
            |rands/second|
million.txt|  8.86e+06   |
#=============================================#
psamples|  p-value |Assessment
#=============================================#
imes
     100|0.00000000|  FAILED
times
     100|0.00000000|  FAILED
times
     100|0.00000000|  FAILED
 times
     100|0.00000000|  FAILED
 times
     100|0.00000000|  FAILED
```

Прохождение тестов Dieharder — ни один тест не был пройден

Скорее всего это связано с распределением цифр: оно не-равномерное даже в самом seed’e:



Результаты авторов

Они тестировали качество при помощи модуля NIST PRNG, сравнивая с другими алгоритмами получения псевдорандомных чисел. Их модель прошла все тесты, и они приходят к выводу что она применима на практике.

	Frequency	BlockFrequence	CumulativeSumsForward	CumulativeSumsReverse	Runs	LongestRun	Rank	NonOverlappingTemplate	OverlappingTemplate	Universal	ApproximateEntropy	RandomExcursions	RandomExcursionsVariant	Serial(m=5)	LinearComplexity
Proposed System	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O

В сравнение — PRNG из NumPy тесты так или иначе частично проходит:

```
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
     file_input|2.6_millon_numpy_uniform_uint32.txt|  4.90e+06  |
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
# The file file_input was rewound 5 times
   diehard_birthdays|   0|       100|     100|0.53230795|  PASSED
# The file file_input was rewound 43 times
      diehard_operm5|   0|   1000000|     100|0.00000000|  FAILED
# The file file_input was rewound 93 times
  diehard_rank_32x32|   0|     40000|     100|0.00000150|   WEAK
# The file file_input was rewound 116 times
    diehard_rank_6x8|   0|    100000|     100|0.00380984|   WEAK
# The file file_input was rewound 126 times
   diehard_bitstream|   0|   2097152|     100|0.10523603|  PASSED
# The file file_input was rewound 206 times
        diehard_opso|   0|   2097152|     100|0.11884260|  PASSED
# The file file_input was rewound 260 times
        diehard_oqso|   0|   2097152|     100|0.17578994|  PASSED
# The file file_input was rewound 285 times
         diehard_dna|   0|   2097152|     100|0.51210612|  PASSED
# The file file_input was rewound 288 times
diehard_count_1s_str|   0|    256000|     100|0.46407457|  PASSED
# The file file_input was rewound 337 times
diehard_count_1s_byt|   0|    256000|     100|0.00833024|  PASSED
# The file file_input was rewound 338 times
 diehard_parking_lot|   0|     12000|     100|0.19184518|  PASSED
# The file file_input was rewound 339 times
    diehard_2dsphere|   2|      8000|     100|0.99762054|   WEAK
# The file file_input was rewound 339 times
    diehard_3dsphere|   3|      4000|     100|0.52214317|  PASSED
# The file file_input was rewound 428 times
     diehard_squeeze|   0|    100000|     100|0.00000000|  FAILED
# The file file_input was rewound 428 times
        diehard_sums|   0|       100|     100|0.33079044|  PASSED
# The file file_input was rewound 432 times
        diehard_runs|   0|    100000|     100|0.00000036|  FAILED
        diehard_runs|   0|    100000|     100|0.23713528|  PASSED
# The file file_input was rewound 483 times
      diehard_craps|   0|    200000|     100|0.00002889|   WEAK
      diehard_craps|   0|    200000|     100|0.06960765|  PASSED
```

Спасибо за внимание ^^