赛题二：离散高斯分布

程俊杰 ✉

2024 年 5 月 6 日

1 第一题

第一题的标准差 $\sigma = 0.75$，中心为 $c = 0$，± 5 的采样概率总共为 $2.38 * 10^{-10}$，因此只在 $[-4, 4]$ 中采样足够应付比赛。概率矩阵的前 8 列为：

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{1}$$

8 比特随机数组成一个无符号数，有如下可能：

1. 0*******：取值范围为 [0, 127)，共 128 种可能，在概率矩阵的第一列采样成功，采样值为 $\{0\}$；
2. 10******：取值范围为 [128, 192)，共 64 种可能，在概率矩阵的第二列采样成功，采样值为 $\{1\}$；
3. 110*****：取值范围为 [192, 224)，共 32 种可能，在概率矩阵的第三列采样成功，采样值为 $\{1\}$；
4. 1110****：取值范围为 [224, 240)，共 16 种可能，在概率矩阵的第五列采样成功，采样值为 $\{0, 1\}$；
5. 11110***：取值范围为 [240, 248)，共 8 种可能，在概率矩阵的第六列采样成功，采样值为 $\{1, 2\}$；

6. 111110**：取值范围为 [248, 252)，共 4 种可能，在概率矩阵的第七列采样成功，采样值为 $\{1, 2\}$；
7. 1111110*：取值范围为 [252, 254)，共 2 种可能，在概率矩阵的第八列采样成功，采样值为 $\{1, 2\}$；
8. 1111111*：取值范围为 [254, 255]，共 2 种可能，无法在概率矩阵的前八列采样成功。

首先看情况 1 至 6，对于一个随机 8 比特无符号数，有：

- $128^{[1]} + 16/2^{[4]} = 136$ 种可能对应的采样值为 0；
- $64^{[2]} + 32^{[3]} + 16/2^{[4]} + 8/2^{[5]} + 4/2^{[6]} = 110$ 种可能对应的采样值为 1；
- $8/2^{[5]} + 4/2^{[6]} = 6$ 种可能对应的采样值为 2；
- 对于非 0 采样值，注意是 $\pm i$ 共同的采样可能。

因此，维护一个长度为 $136 + 110 + 6 = 252$ 的采样表，其中 0 的数量为 136,± 1 的数量分别为 55，± 2 的数量分别为 3，则可以通过 8 比特随机数以 $\frac{252}{256} = \frac{63}{64}$ 的概率直接查表得到采样值，而且是带正负号的。

对于情况 7，若 8 比特无符号数为 252，则认为采样值为 1；若为 253，则认为采样值为 2，这两个数也存在采样表中。由于在第八列才采样成功，用尽了 8 比特无符号数中的所有随机比特，因此还需要一个额外的随机比特确定正负号。

对于情况 8，继续运行 Knuth-Yao 算法直到采样成功。

对于任意一个 8 比特随机数，其落在情况1 至6 的概率为 $\frac{63}{64}$，称为关键路径，关键路径上的操作是影响采样速率最主要的因素。路径上的操作有：

- 获取随机数，随机数生成器存储了 512 字节的随机数，获取随机数实际上是一个查表操作，即访问一次内存；
- 根据随机数查采样表，访问一次内存，返回采样值；
- 由于这些内存需要经常访问，实际上是常驻缓存的。

这意味着关键路径已经被精简成两次缓存访问，一次采样共需要不到 8 个时钟周期（现代 CPU 的缓存命中延迟可以低至 1-2 个时钟周期，这里多出

来的时钟周期应该是随机数填充、if 判断、前 8 比特采样失败等情况所造成的。)，这已经没办法继续优化了。

一些理论上但实际不可行的优化：

- 减少获取随机数的时间：
  - 直接放弃随机数，维护一个非常长的采样表 samples 以及一个计数器 cnt，每次采样返回 samples[cnt++]。可以将采样时间减少到 4 个时钟周期，但是采样多少次就需要事先存好多大的采样表，内存上不可行。
  - 直接通过 RDRAND 指令获取硬件随机数，但是该指令安全性很高，类似于真随机数（是不是真随机数我也不清楚，但是这个指令跟 CPU 中的熵源有关系，Intel 和 AMD 的熵源是什么我也没查），每获取一个随机数需要 463 个时钟周期，得不偿失。
  - 自己实现一个更快的随机数生成器，但是官方给的随机数生成器速率每生成一个随机数大约只需要 4 个时钟周期，实现一个更快的可能性不大。
- 减少访问采样表的时间：
  - 直接通过随机数计算出采样值，但是计算随机数的指令也需要通过访问缓存获取、执行计算也需要时间，不如直接查采样表快。

不严谨地说，任何超过 2 次内存（缓存）访问的算法，采样速度都不可能低至 8 个时钟周期。而获取随机数需要一次内存访问，得到对应的采样值至少需要一次内存访问，因此不存在内存访问次数小于 2 的采样算法。

时间复杂度：8 个时钟周期，主频为 2.8GHz 的 PC 上采样速率为 $3.57 * 10^8$ 样本/秒。

空间复杂度：概率矩阵的行对应采样范围，列对应采样精度，5 行 24 列的矩阵足够满足一亿次采样的精度。因此共需要 $5*(24-8)$(前 8 列无需存储）+ $(24-8)$(列和向量）+ 254(采样表）= 390 字节的存储空间。

A 第一题性能分析图

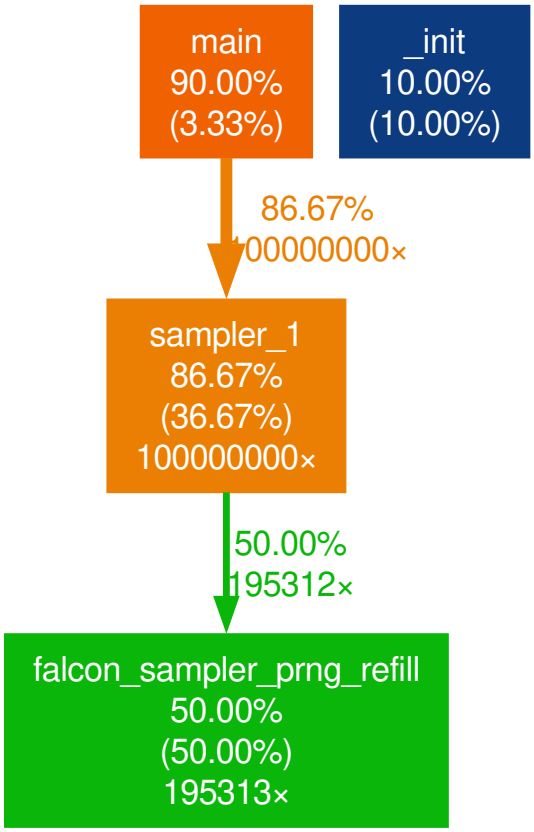

图 1: 各函数运行时间占比