CSIS0351/CSIS8601: Randomized Algorithms
Lecture 5B: More on Measure Concentration: Black Box Sampling and Pollard's Kangaroo Algorithm
Instructor: Hubert Chan
Date: 6 Oct 2011

These lecture notes are supplementary materials for the lectures. They are by no means substitutes for attending lectures or replacement for your own notes!

1 Limitation of Black Box Sampling

Consider the black box sampling problem that is described as follows. Suppose a black box contains red balls and blue balls. The number of red balls, which can be zero, is unknown. One is allowed to sample (with replacement) uniformly at random from the bag. Call an access to the black box *positive* if one samples a red ball and *negative* otherwise. The goal is to estimate the probability that an access is positive, that is, the fraction of red balls, with multiplicative error at most $\epsilon \in (0, 1)$.

We claim that without the assumption that there is at least one red ball in the black box, no algorithm, whether deterministic or randomized, can give an estimate with multiplicative error at most ϵ with failure probability at most $\delta \leq \frac{1}{4}$. That is, if p is the true fraction of red balls in the black box and $\widehat{p}$ is the estimate given by a particular algorithm, then no algorithm can guarantee that $Pr[|\widehat{p} - p| \geq \epsilon p] \leq \delta$ for $\epsilon \in (0, 1)$ and $\delta \in (0, \frac{1}{4}]$.

We prove the claim by contradiction. Suppose, on the contrary, that there is an algorithm $\mathcal{A}$ that can give an estimate with multiplicative error at most ϵ with failure probability at most δ.

1.1 The High Failure Probability Scenario

We intend to find a case in which the algorithm fails with probability larger than $\frac{1}{4}$, which leads to a contradiction. Observe that the randomness of an answer (estimate) comes from two parts: (1) the algorithm $\mathcal{A}$ itself and (2) the procedure of sampling from the black box. Both parts give a chance for the algorithm to fail.

Consider the case with some $p > 0$. Note that $\widehat{p} = 0$ implies $|\widehat{p} - p| > \epsilon p$. That is, if $\mathcal{A}$ returns 0, then it must fail. Intuitively, it is reasonable that $\mathcal{A}$ returns 0 given all accesses it makes are negative. Also, when p is a very small positive number, it is likely that a large number of initial accesses are all negative.

Let E_1 be the event that $\mathcal{A}$ returns 0 and E_2 the event that the first N accesses are all negative, where N is some positive integer. Note that $\mathcal{A}$ fails if both E_1 and E_2 happens in the scenario with $p > 0$. Also note that $\mathcal{A}$ gives an answer only based on its accesses (positive or negative), independent of p. Clearly, we have

$$
\begin{aligned}
Pr[\mathcal{A} \text{ fails } |\{p>0\}] &\geq Pr[E_1|\{p>0\}] \\
&\geq Pr[E_1 \cap E_2|\{p>0\}] \\
&= Pr[E_1|\{p>0\} \cap E_2] \cdot Pr[E_2|\{p>0\}] \\
&= Pr[E_1|E_2] \cdot Pr[E_2|\{p>0\}] \\
&= P_a P_b.
\end{aligned}
$$

Here we let $P_a := Pr[E_1|E_2]$ and $P_b := Pr[E_2|\{p > 0\}]$. If we can find P_a and P_b such that $P_a P_b > \frac{1}{4}$, then we can conclude that in this case $\mathcal{A}$ fails with probability larger than $\frac{1}{4}$.

1.2 Finding P_a by Considering $\mathcal{A}$'s Performance

Since the mechanism of $\mathcal{A}$ is invisible to us, we can not say for sure that $\mathcal{A}$ returns 0 given the first N accesses are all negative. It may return any feasible answer in range $[0, 1]$. Then how do we bound P_a?

Note that the randomness of the event $E_1|E_2$ only comes from $\mathcal{A}$ itself, independent of the value of p. This broadens our choices of p to investigate $P_a = Pr[E_1|E_2]$.

Consider the case that $p = 0$. Then any access to the black box must be negative. On the other hand, since $|\widehat{p} - 0| \leq \epsilon \cdot 0$ implies $\widehat{p} = 0$, the algorithm $\mathcal{A}$ gives a correct answer if and only if it returns 0. Therefore returns 0 with probability at least $1 - \delta \geq \frac{3}{4}$.

Note that given a positive integer N_0, *the first N_0 accesses* are not equivalent to *all the accesses that $\mathcal{A}$ makes*. Observe that the number of accesses that $\mathcal{A}$ makes is unknown and may even be a variable. However, whenever $\mathcal{A}$ returns an answer, it can only have made a finite number of samples. Let q_k be the probability that $\mathcal{A}$ returns 0, given that it makes k accesses and all the accesses are negative. Then we have

$$
\sum_{k=0}^{\infty} q_k = Pr[\mathcal{A} \text{ returns a correct answer if } p = 0] \geq 1 - \delta \geq 0.75.
$$

Hence, there exists a positive integer N such that $\sum_{k=0}^{N} q_k \geq 0.74$. This means that $\mathcal{A}$ returns a correct answer with probability at least 0.74 if it makes no more than (not necessarily exactly) N accesses. It follows that if the first N accesses are negative, then $\mathcal{A}$ will return 0 with probability at least 0.74, i.e., $P_a \geq 0.74$.

1.3 Finding P_b by Considering sufficiently small p

Now we return to the case that $p > 0$. We want to find an appropriate p such that $P_b = Pr[E_2|\{p > 0\}]$ is large enough. Intuitively, if p is very small, say $p \ll \frac{1}{N}$, then the expected number of positive accesses in the first N samples is $Np \ll 1$. Then by Chernoff Bound the event E_2 given $p > 0$ happens with high probability.

Consider the first N accesses. Let $X_i := 1$ if the i-th access is positive and $X_i := 0$ if the i-th access is negative. Define $Y := \sum_{i=1}^{N} X_i$. Then given p, we have $P_b = 1 - Pr[Y \geq 1]$.

We give a bound for $Pr[Y \geq 1]$. Note that X_i's are independent. To apply Chernoff Bound, we rewrite the probability as $Pr[Y \geq (1+\beta)E[Y]]$, where $(1+\beta)E[Y] = 1$ and β is a constant to be determined later. Since $E[Y] = Np$ is expected to be small, we choose the Chernoff Bound with large β, say $\beta \geq 2$. Observing that $E[Y] = \frac{1}{1+\beta}$, we have

$$Pr[Y \geq (1+\beta)E[Y]] \leq \exp\left(-\frac{\beta^2 E[Y]}{2+\beta}\right) \leq \exp\left(-\frac{1}{2}\beta E[Y]\right) = \exp\left(-\frac{\beta}{2(1+\beta)}\right) \leq 0.65.$$

The last inequality holds if we set $\beta = 10$. Thereafter, it suffices to set $p = \frac{1}{11N}$.

Hence, we have $P_b = 1 - Pr[Y \geq 1] \geq 0.35$. Recalling $P_a \geq 0.74$, we conclude that in the case $p = \frac{1}{11N}$ the failure probability of $\mathcal{A}$ is at least $P_a P_b \geq 0.74 \times 0.35 > \frac{1}{4}$.

2 Discrete Logarithm Problem

Consider a group $G = \{1, \ldots, p-1\}$, in which the operation $\cdot$ is the product of two elements modulo p; that is, for $a, b \in G$, the result of the operation is $a \cdot b = ab \mod p$. We let $ab = a \cdot b$ and $a^n = a \cdot a^{n-1}$ for all positive integer n throughout this section. Within the group G, given two elements g and y, can one find the logarithm of y to the base g, i.e., an element x such that $g^x = y$?

Suppose that, in addition, one knows that x is an element in some range $[a, b]$. A trivial algorithm is to try all elements in $[a, b]$, which takes $O(|b-a|)$ time, assuming that it takes $O(1)$ time to compute g^c for each $c \in [a, b]$. It guarantees to find x but takes too much time. We want to find a faster algorithm.

Formally, given $g, y, a, b \in G$ and that there exists $x \in [a, b]$ such that $g^x = y$, our goal is to find x. We want a randomized algorithm satisfying the following conditions:

(C1) With constant probability, it can find x;

(C2) With constant probability, it takes $O(\sqrt{|b-a|})$ time;

(C3) It requires $O(1)$ memory.

We first give another form of Chernoff Bound, then introduce a randomized algorithm and finally show that it satisfies these conditions.

2.1 Chernoff Bound with Large Range

Recall that the basic Chernoff Bound only applies to $\{0, 1\}$-random variables. We introduce another form of Chernoff Bound that allows the random variables to take real values in a general interval.

Lemma 2.1 (Chernoff Bound with General Range) *Suppose $X_1, X_2, \ldots, X_n$ are independent real-valued random variables, such that for each i, X_i takes values from the interval $[0, R]$, where $R > 0$. Let $Y := \sum_{i=1}^{n} X_i$. Then for any $\epsilon \in (0, 1)$,*

$$Pr[|Y - E[Y]| \geq \epsilon E[Y]] \leq 2\exp(-\frac{\epsilon^2 E[Y]}{3R}).$$

Proof: The proof is similar to that of basic Chernoff Bound. As an example we prove that $Pr[Y \geq (1+\epsilon)\mu] \leq \exp(-\frac{\epsilon^2\mu}{3R})$, where $\mu := E[Y]$.

Suppose $t > 0$. Using Markov's Inequality, the independence of X_i's and moment generating function, we have

$$\begin{aligned} Pr[Y \geq (1+\epsilon)\mu] &= Pr[\exp(t\sum_{i=1}^{n} X_i) \geq \exp(t(1+\epsilon)\mu)] \\ &\leq \exp(-t(1+\epsilon)\mu)E[\exp(t\sum_{i=1}^{n} X_i)] \\ &= \exp(-t(1+\epsilon)\mu)\prod_{i=1}^{n} E[\exp(tX_i)]. \end{aligned}$$

Consider $E[\exp(tX_i)]$. Let $\lambda := 1 - \frac{X_i}{R}$. By the convexity of the function $x \mapsto e^x$, we have

$$\exp(tX_i) = \exp(\lambda \cdot 0 + (1-\lambda)\cdot tR) \leq \lambda e^0 + (1-\lambda)e^{tR} = 1 + \frac{(e^{tR}-1)X_i}{R}.$$

Then $E[\exp(tX_i)] \leq 1 + \frac{(e^{tR}-1)E[X_i]}{R} = 1 + \frac{(e^{tR}-1)\mu}{Rn} \leq \exp\left(\frac{(e^{tR}-1)\mu}{Rn}\right)$. Thereafter we have

$$Pr[Y \geq (1+\epsilon)\mu] \leq \exp(-t(1+\epsilon)\mu)\cdot\exp\left(\frac{(e^{tR}-1)\mu}{R}\right) = \exp\left(\mu\left(\frac{1}{R}e^{tR} - (1+\epsilon)t - \frac{1}{R}\right)\right).$$

Let $g(t) := \frac{1}{R}e^{tR} - (1+\epsilon)t - \frac{1}{R}$. Then $g'(t) = e^{tR} - (1+\epsilon)$ and $g''(t) = Re^{tR} > 0$. It follows that g attains its minimum when $g'(t) = 0$, i.e., when t equals to $t_0 := \frac{1}{R}\ln(1+\epsilon) > 0$. Using the inequality $(1+\epsilon)\ln(1+\epsilon) \geq \epsilon + \frac{\epsilon^2}{3}$ for $0 < \epsilon < 1$, we have $g(t_0) = \frac{1}{R}(\epsilon - (1+\epsilon)\ln(1+\epsilon)) \leq -\frac{\epsilon^2}{3R}$. Therefore, we have

$$Pr[Y \geq (1+\epsilon)\mu] \leq \exp(\mu g(t_0)) = \exp(-\frac{\epsilon^2\mu}{3R}).$$

■

2.2 Pollard's Kangaroo Algorithm

Define $R := |b-a|$ and $K := \sqrt{R}$. Suppose $f : G \mapsto \{1, 2, \ldots, K\}$ is some randomized function such that f is uniform and $f(u)$'s are independent for all $u \in G$. The Pollard's Kangaroo Algorithm is described as follows.

Consider that a kangaroo's behaviour (a sequence of jumps) is measured by elements in G. Specifically, a kangaroo initially stands in a position $d_0 \in G$. Then it jumps to a new position d_1 such that $g^{d_1} = g^{f(g^{d_0})}g^{d_0}$. Note that $d_1 = f(g^{d_0}) + d_0$. We call $f(g^{d_0})$ the jumping *distance*.

Generally, the kangaroo's j-th jump is from d_{j-1} with distance $f(g^{d_{j-1}})$, and the new position is $d_j = f(g^{d_{j-1}}) + d_{j-1}$. Note that the distance of each jump, taking values from $\{1, 2, \ldots, K\}$, only depends on the kangaroo's current position and is known once the jump is finished. Also note that the distances of jumps in a sequence are independent.

Suppose we have two kangaroos. The first one is tamed, starting from a known position b, while the second one is wild, starting from an unknown position x, the value of which we want to find. Note that the starting position of the wild kangaroo is always behind that of the tamed one, with a distance at most R. The tamed kangaroo first jumps for T steps, for some T to be determined later; whenever the tamed kangaroo lands on a position, it lays down a "trap" to potentially catch the wild kangaroo and stands in the position after the last jump to wait for the wild kangaroo. Then, we let the wild kangaroo jump. Observe that the jump distance of the tamed or the wild kangaroo only depends on its current position. Hence, if the wild kangaroo lands on one of the traps set by the tamed kangaroo, it will follow all the steps of the tamed kangaroo afterwards and therefore will eventually land on the final position of the tamed kangaroo.

Let $\{s_0, s_1, \ldots\}$ and $\{t_0, t_1, \ldots\}$ be the sequences of positions of the tamed and wild kangaroos, respectively. Observe that $s_0 = b$, $s_{i+1} = f(g^{s_i}) + s_i$ for $i \in \{0, 1, \ldots, T-1\}$ and $t_0 = x$, $t_{j+1} = f(g^{t_j}) + t_j$ for $j \geq 0$. Based on the situation when the algorithm stops, it either finds x or fails.

(1) The wild kangaroo lands on the final position of the tamed kangaroo, i.e., $g^{s_T} = g^{t_j}$ for some $j \geq 0$. Then, $b + \sum_{k=0}^{T-1} f(g^{s_k}) = x + \sum_{l=0}^{j-1} f(g^{t_l})$, which implies $x = b + \sum_{k=0}^{T-1} f(g^{s_k}) - \sum_{l=0}^{j-1} f(g^{t_l})$. Since b and all the $f(\cdot)$ values are known, the algorithm finds x.

(2) The wild kangaroo jumps past the final position of the tamed kangaroo without stepping on any of the traps, then the algorithm fails to find x. Observe this happens when $\sum_{l=0}^{j-1} f(g^{t_l}) > b + \sum_{k=0}^{T-1} f(g^{s_k})$, which can be used a stopping condition. Since the jumping distance is finite and at least 1, the algorithm will terminate in finite time.

2.3 Checking the Three Conditions

We show that by choosing appropriate T, the Pollard's Kangaroo Algorithm satisfies conditions (C1), (C2) and (C3).

Checking Condition (C3). Firstly we consider the memory requirement. Observe that we just need to maintain the sum of the jump sizes for each kangaroo. Hence, for the tamed kangaroo, one counter is used to keep track of $b + \sum_{k=0}^{i-1} f(g^{s_k})$ until i reaches T and another variable for storing the the final position $\alpha := g^{s_T}$. For the wild kangaroo, another counter is used to keep track of the sum $\sum_{l=0}^{j-1} f(g^{t_l})$ and the current position $\beta = g^{t_j}$. Therefore, the algorithm uses only $O(1)$ memory.

To check conditions (C1) and (C2), we observe that the tamed kangaroo jumps for exactly T times. After the tamed kangaroo finishes, the wild kangaroo starts jumping. The process stops if (1) the wild kangaroo jumps to one of the tamed kangaroo's positions or (2) the wild kangaroo passes over all the T positions without stepping on any of them. If the first condition happens, we say the wild kangaroo is *trapped*, and there exist $i \in \{0, 1, \ldots, T\}$, $j \geq 0$ such that $s_i = t_j$; otherwise, we say the wild kangaroo *escapes*, and it jumps exactly m times, where m is the smallest positive integer such that $s_T < t_m$.

Checking Condition (C1). We say a jump is *dangerous* if it is possible that the wild kangaroo gets trapped after that jump. That is, a jump is dangerous if it starts in a position d satisfying

that there is at least one $i \in \{0, 1, \ldots, T\}$ such that $s_i - d \leq K$. Note that once the wild kangaroo commits a dangerous jump, all subsequent jumps are dangerous since $s_{i+1} - s_i \leq K$ for all i.

If the wild kangaroo starts with a dangerous jump, then each jump falls in a trap with probability at least $\frac{1}{K}$. Then the probability that it is not trapped after cK jumps is at most

$$(1 - \frac{1}{K})^{cK} \leq \exp\left(-\frac{1}{K}cK\right) = e^{-c} \leq \frac{1}{4},$$

where c is a constant and the last inequality holds if $c \geq \ln 4$. Therefore, with constant probability the wild kangaroo will be trapped within cK dangerous jumps.

Now we want to find T such that the track of the tamed kangaroo can support cK dangerous jumps. Let X_i be the distance of the i-th jump of the tamed kangaroo and $\widehat{X}_i$ the distance of the i-th jump of the wild kangaroo. Observe that $X_i = f(g^{s_{i-1}})$ and $\widehat{X}_i = f(g^{t_{i-1}})$. Note that all X_i's are independent and so are all $\widehat{X}_i$'s. Both X_i's and $\widehat{X}_i$'s take values from $\{1, 2, \ldots, K\}$ uniformly at random. Let $Z := \sum_{i=1}^{T} X_i$ and $Y := \sum_{i=1}^{cK} \widehat{X}_i$. We want $Z \geq Y$ with high probability.

Note that $E[Z] = \frac{1}{2}T(K+1)$ and $E[Y] = \frac{1}{2}cK(K+1)$. Then $E[Z] = 2E[Y]$ if we set $T = 2cK$. By Chernoff Bound with general range, we have

$$Pr[Y \geq \frac{3}{2}E[Y]] \leq \exp\left(-\frac{E[Y]}{12K}\right) = \exp\left(-\frac{c(K+1)}{24}\right)$$

and

$$Pr[Z \leq \frac{3}{2}E[Y]] = Pr[Z \leq \frac{3}{4}E[Z]] \leq \exp\left(-\frac{E[Z]}{48K}\right) = \exp\left(-\frac{c(K+1)}{48}\right).$$

Therefore, we have

$$\begin{aligned} Pr[Z \geq Y] &\geq [Pr[Y < \frac{3}{2}E[Y]] \cdot [Pr[Z > \frac{3}{2}E[Y]] \\ &\geq \left(1 - \exp\left(-\frac{c(K+1)}{24}\right)\right) \cdot \left(1 - \exp\left(-\frac{c(K+1)}{48}\right)\right) \\ &= 1 - o(1). \end{aligned}$$

In summary, if $T = 2cK$, where $c \geq \ln 4$, then the algorithm can find x with high probability.

Checking Condition (C2). Note that we have given a number of jumps of the tamed kangaroo such that the track is long enough to trap the wild kangaroo. The length of the track can be as large as $TK = 2cK^2 = 2cR$. Moreover, the starting position of the wild kangaroo can be behind that of the tamed kangaroo with a distance as large as $|b - a| = R$. Therefore, it takes $O(R)$ time for the wild kangaroo to get trapped or escape in the worst case, where it jumps unit distance each time.

We show that with high probability the wild kangaroo gets trapped or escapes after $4cK$ jumps. Recall that $E[Z] = cK(K+1)$. Let $\epsilon_1 = \frac{K-2}{2(K+1)} \geq \frac{1}{8}$, assuming $K \geq 3$, then by Chernoff Bound with general range we have

$$Pr[Z \geq \frac{3}{2}cK^2] = Pr[Z \geq (1+\epsilon_1)E[Z]] \leq \exp\left(-\frac{\epsilon_1^2 E[Z]}{3K}\right) \leq \exp\left(-\frac{c(K+1)}{192}\right).$$

Define $\widehat{Y} := \sum_{i=1}^{4cK} \widehat{X}_i$, where $\widehat{X}_i$ is the distance of the i-th jump of the wild kangaroo. Then $E[\widehat{Y}] = 2cK(K+1)$. Let $\epsilon_2 = 1 - \frac{(3c+2)K}{4c(K+1)} \geq \frac{1}{8}$, assuming $c \geq 4$. Then by Chernoff Bound with general range we have

$$Pr[\widehat{Y} \leq (\frac{3}{2}c+1)K^2] = Pr[\widehat{Y} \leq (1-\epsilon_2)E[\widehat{Y}]] \leq \exp\left(-\frac{\epsilon_2^2 E[\widehat{Y}]}{3K}\right) \leq \exp\left(-\frac{c(K+1)}{96}\right).$$

Therefore, the probability that the wild kangaroo gets trapped or escapes within $4cK$ jumps is

$$\begin{aligned} Pr[\widehat{Y} \geq Z + R] &\geq Pr[Z < \frac{3}{2}cK^2] \cdot Pr[\widehat{Y} > (\frac{3}{2}c+1)K^2] \\ &\geq \left(1 - \exp\left(-\frac{c(K+1)}{192}\right)\right) \cdot \left(1 - \exp\left(-\frac{c(K+1)}{96}\right)\right) \\ &= 1 - o(1). \end{aligned}$$

Hence, the algorithm can terminate in time $O(K) = O(\sqrt{|b-a|})$ with high probability.

To sum up, we have shown that the Pollard's Kangaroo Algorithm can solve the discrete logarithm problem if we set $T = 2cK$, where $c \geq 4$ is a constant.