Homework 4: Monte Carlo & Algorithm Analysis

UC Irvine CS177: Applications of Probability in Computer Science

Due on November 7, 2019 at 11:59pm

Question 1: (25 points)

You are designing a new robot to give campus tours. But while testing the robot in Aldrich Park, it malfunctions and begins driving randomly along a path. Let $X = 0$ indicate the robot's position when the malfunction occurs, positive X values indicate movement to the north, and negative X values indicate movement to the south.

a) Suppose that every minute, the robot travels south 5 *meters with probability* 1/2*, or north* 5 *meters with probability* 1/2*. The directions of travel at successive minutes are independent. After one hour, what is the expected position of the robot? What is the standard deviation of the robot's position?*

b) Suppose instead that the motor controllers on the robot are biased, so that every minute it travels south 5 *meters with probability* 3/4*, or north* 5 *meters with probability* 1/4*. The directions of travel at successive minutes are independent. After one hour, what is the expected position of the robot? What is the standard deviation of the robot's position?*

c) You were taking a break when the robot began malfunctioning, and did not observe its movement. Under which motion model, the one from part (a) or part (b), do you think it would be faster to find your lost robot? Justify your answer.

Question 2: (15 points)

Your friend has written a computer program that supposedly produces a sequence of independent random numbers that are uniformly distributed between 0 and 1. To test this program, you ask it to generate $n = 1000$ random samples, and compute their average.

a) Suppose that you find that the average of the $n = 1000$ *samples is* 0.55*. Using the central limit theorem, if the generated numbers truly had a uniform distribution, what is the approximate probability of their average being greater than or equal to 0.55? Do you believe that your friend's random number generator is correct?*

b) Suppose that you find that the average of the $n = 1000$ *samples is* 0.50*, exactly equal to the true mean of the target uniform distribution. Does this give you confidence that your friend's random number generator is correct? Why or why not?*

Question 3: (20 points)

Let X be a non-negative random variable with expected value $E[X] = 4$, and let $a > 0$ be some positive number. We derive and evaluate various *upper bounds* on $P(X \geq a)$.

a) Suppose that $0 < a < E[X]$. What is the best upper bound you can give for $P(X \geq a)$?

b) Use Markov's inequality to give an upper bound on $P(X \geq a)$ that is valid for any $a > 0$.

c) For any non-negative random variable, the proof of Markov's inequality (from lecture and the textbook) can be slightly generalized to show that

$$P(X \geq a) \leq \frac{E[X^2]}{g(a)},$$

where $g(a) > 0$ is some constant that depends on a. Prove this bound and find $g(a)$.

d) Suppose that in addition to knowing $E[X] = 4$, you know that $Var(X) = 10$. Apply the inequality from part (c) to upper bound $P(X \geq a)$. For which values of $a > 0$ is this bound better (tighter) than the Markov's inequality bound from part (b)?

Algorithm 1 Insertion Sort

1: **input**: A list L of n real numbers
2: **for** Each element $x \in L$ from $L(2)$ to $L(n)$ **do**
3: **while** x is less than the element preceding x in L **do**
4: swap that element with x
5: **end while**
6: **end for**
7: **return** The sorted list L

Question 4: (40 points)

Given a list of n distinct real numbers, a sorting algorithm seeks to reorder the list from smallest to largest. As comparisons between elements are relatively slow, sorting algorithms are commonly judged by the number of comparisons they make. Algorithm 1 is called insertion sort because each element in sequentially "inserted" into its proper place in the partially sorted list. We also provide a Python implementation of the insertion sort algorithm.

a) Compute the maximum possible number of comparisons between list elements made by the deterministic insertion sort in Algorithm 1, in terms of n.

b) To avoid the worst case, randomized insertion sort first randomly permutes the list, and then runs insertion sort. Let X be an integer random variable equal to the number of comparisons made by randomized insertion sort. Compute $E[X]$ in terms of n.

c) Monte Carlo sampling can be used to estimate the cumulative distribution function of X. Use Python to execute the provided `randomized_insertion_sort` *1000 times for lists of length $n = 10$, and plot the empirical CDF $F_X(x)$. Compute and report the average number of comparisons across your 1000 Monte Carlo trials.*

d) Repeat part (c) for lists of length $n = 100$, and discuss how the distribution of comparisons changes as the list length grows.