CSE 312: Foundations of Computing II Autumn 2022

Problem Set 9

Due: Friday, December 9, by 11:59pm. (No extensions / late days allowed)

Instructions

Solutions format, collaboration policy, and late policy. See PSet 1 for further details. The same requirements and policies still apply. Also follow the typesetting instructions from the prior PSets.

Solutions submission. You must submit your solution via Gradescope. In particular:

- For the solutions to Tasks 1-4, and the written part of Task 5, submit under "PSet 9 [Written]" a *single* PDF file containing the solution to all tasks in the PSet. Each numbered task should be solved on its own page (or pages).
- For the programming part (sub-task **c)** of Task 5), submit your code under "PSet 9 [Coding]" as a file called mcmc_knapsack.py.
- Submit your optional solution to Extra Credit Task 6 under"PSet 9 [Extra]".

Task 1 – It is with a Heavy Tail [16 pts]

All of the continuous distributions we have considered so far are either non-zero only on a finite range or, like the normal or exponential distributions, have densities that go to 0 exponentially quickly in x. It turns out that many quantities (and hence random variables) that we observe in practice display a "heavy tailed" behavior in that the probability density function at x is behaves roughly like $1/x^c$ rather than roughly $1/e^x$ of $1/e^{x^2/2}$. It isn't hard to show that we need $c > 1$ for this to work. Using $\alpha + 1$ instead of c for $\alpha > 0$, we note that $\int_1^\infty x^{-\alpha-1}dx = \frac{-1}{\alpha} \cdot x^{-\alpha}\Big|_1^\infty = 0 - \frac{-1}{\alpha} = \frac{1}{\alpha}$ and hence the following is a probability density function.

$$f(x;\alpha) = \begin{cases} \frac{\alpha}{x^{\alpha+1}} & \text{for } x \geqslant 1 \\ 0 & \text{otherwise} \end{cases}$$

a) If random variable P has pdf $f_P(x) = f(x;\alpha)$ for $\alpha > 0$, compute $\mathbb{E}[P]$. You will need different cases depending on whether α is greater than 1, equal to 1, or less than 1.

b) You are given i.i.d. samples $x_1, x_2, \ldots, x_n$ from the distribution given by f with parameter α. Find the MLE for α.

Task 2 – Uniform Distribution [24 pts]

Let $x_1, x_2, \ldots, x_n$ be independent samples from Unif$(0,\theta)$, the continuous uniform distribution on $[0,\theta]$. Here, θ is the unknown parameter.

a) Write down the likelihood function $\mathcal{L}(x_1, \ldots, x_n\,;\theta)$.

b) Explain why $\hat{\theta} = \max\{x_1, \ldots, x_n\}$ is the MLE estimator for θ.

To this end, explain why this value of $\hat{\theta}$ maximizes $\mathcal{L}(x_1, x_2, \ldots, x_n\,;\theta)$ directly, by describing the behavior of the likelihood function (as a function of θ) and inferring the maximum from this description, rather than trying to use calculus.

c) For the remaining parts of this question we suppose that this uniform distribution is on $[0, \theta]$ for a particular fixed $\theta > 0$, and let

$$\hat{\Theta}_n = \hat{\theta}(X_1, \ldots, X_n) = \max\{X_1, \ldots, X_n\}$$

be a *random variable*, where $X_1, \ldots, X_n$ are independent and follow the uniform distribution on $[0, \theta]$. Compute the CDF $F_{\hat{\Theta}_n}(x)$.

Hint: Focus first on the interval $0 \leqslant x \leqslant \theta$, but when you're done with that, don't forget to also define $F_{\hat{\Theta}_n}(x)$ on the rest of the real numbers.

d) From your answer to **c)** compute the probability density function $f_{\hat{\Theta}_n}(x)$ of $\hat{\Theta}$.

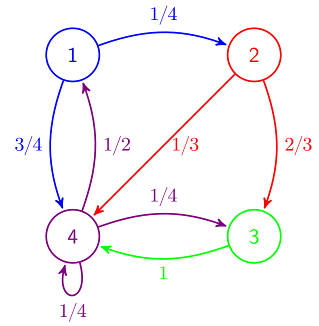
e) From your answer to **d)**, compute $\mathbb{E}\left[\hat{\Theta}_n\right]$. Why is $\hat{\theta}$ not an unbiased estimator of θ?

f) Starting from the value of $\mathbb{E}\left[\hat{\Theta}_n\right]$ you computed in **e)**, show how to make a small modification to the MLE studied so far to produce an unbiased estimator of θ and explain why it is unbiased.

Note: There also are other unbiased estimators of θ, but we want you to produce one that directly uses what you have inferred in **e)**.

Task 3 – Markov Chains [18 pts]

In this problem we will explore the Markov chain shown in the following figure:



a) Give the transition probability matrix (TPM) $\mathbf{M}$ of the Markov chain in the above figure.

b) Recall that $X^{(t)}$ denotes the state of the Markov chain in the t-th step. Use the law of total probability to compute $\mathbb{P}\left(X^{(2)} = 4 \mid X^{(0)} = 1\right)$. Show your work.

c) Suppose that the initial state is uniformly distributed, i.e., we represent its distribution through the vector

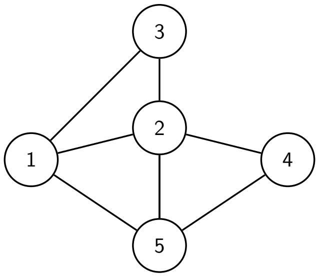
$$\mathbf{q}^{(0)} = \left[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right].$$

Compute $\mathbf{q}^{(0)}\mathbf{M}$, the vector-matrix-product of $\mathbf{q}^{(0)}$ and the TPM $\mathbf{M}$ as **4 simplified fractions**. Describe what each component of $\mathbf{q}^{(0)}\mathbf{M}$ represents.

d) Find the stationary distribution of the Markov chain, i.e., find a probability distribution π such that $\pi\mathbf{M} = \pi$.

Task 4 – Random Walk Analysis [14 pts]

The following figure depicts undirected graph with 5 nodes.



A *random walk* on this graph is a Markov chain whose states are the nodes of this graph. If we are at a particular node i in the graph at some time step, then in the following time step, we will transition to each of its neighbors with equal probability. So if it has one neighbor, we transition there with probability 1. If it has 3 neighbors, we transition to each of those with probability 1/3, and so on.

a) Consider now a random walk on the above graph. Give the transition probability matrix of the corresponding Markov chain.

b) Consider a random walk on an undirected graph with n vertices and m edges, where the degree of vertex i is d_i. (The degree of a vertex is the number of neighbors it has in the graph. So in the graph above, for example, $d_1 = 3$ and $d_4 = 2$.) Prove that a stationary distribution $\pi = [\pi_1, \ldots, \pi_n]$ of the random walk is given by

$$\pi_i = \frac{d_i}{2m} \text{ for all } i \in [n].$$

Note that this is equivalent to showing that if $\pi_i = \frac{d_i}{2m}$ for all i then

- $\pi = \pi \mathbf{M}$ and
- $\sum_{i=1}^{n} \pi_i = 1$

where $\mathbf{M}$ is the transition probability matrix of the Markov chain.

This shows, for example, that for the random walk on the graph shown above

$$\pi = \left[\frac{3}{14}, \frac{4}{14}, \frac{2}{14}, \frac{2}{14}, \frac{3}{14}\right].$$

However, make sure your proof that $\pi_i = \frac{d_i}{2m}$ works for an arbitrary undirected graph, and not just for our specific Markov Chain.

Hint: You can use the fact that $\sum_{i=1}^{n} d_i = 2m$ since the sum counts each edge twice, once from each end.

Task 5 – Knapsacks (Coding) [28 pts]

Markov Chain Monte Carlo (MCMC) is a technique that can be used to heuristically and approximately solve otherwise hard optimization problems (among other things). The general strategy is as follows:

1. Define a Markov Chain with states being possible solutions, and (implicitly defined) transition probabilities that result in the stationary distribution π having higher probabilities on "good" solutions to our problem. We don't actually compute π, but we just want to define the Markov Chain such that the stationary distribution would have higher probabilities on more desirable solutions.
2. Run MCMC, i.e., simulate the Markov Chain for many iterations until we reach a "good" state/solution.

In this question, there is a collection of n items, numbered 0 to $n-1$, available to us, and each has some *value* and some *weight*, both of which are positive real values. We want to find the optimal subset of items that maximizes the total value (the sum of the values of the items we take), subject to the total weight (the sum of the weights of the items we take) being less than some $W > 0$. (This is known as the **knapsack problem**). In items.txt, you'll find a list of potential items with each row containing the name of the item (string), and its value and weight (positive floats).

You will implement an MCMC algorithm which also depends on a parameter T that is not part of the problem definition. Pseudocode is provided below, and a detailed explanation is provided immediately after.

Algorithm 1 MCMC for 0-1 Knapsack Problem

```
subset ← vector of $n$ zeros (indexed by 0 to $n-1$), where subset is always a binary vector in $\{0,1\}^n$ that
  represents whether or not we have each item. (This means that we initially start with an empty knapsack).
best_subset ← subset
for $t = 1, \ldots,$ NUM_ITER do
    $k$ ← a random uniform integer in $\{0, 1, \ldots, n-1\}$.
    new_subset ← subset but with subset[$k$] flipped ($0 \to 1$ or $1 \to 0$).
    $\Delta$ ← value(new_subset) − value(subset)
    if new_subset satisfies weight constraint (total weight $\leqslant W$) then
        if $\Delta > 0$ OR ($T > 0$ AND Unif$(0,1) < e^{\Delta/T}$) then
            subset ← new_subset
    if value(subset) > value(best_subset) then
        best_subset ← subset
```

The extra parameter T in the MCMC algorithm represents a "temperature" that controls the trade-off between exploration and exploitation. The state space $\mathcal{S}$ is the set of all subsets of n items. The algorithm starts with a state (current subset) corresponding to an empty knapsack. At each iteration, the algorithm proposes a new state (proposed subset) as follows: choose a random index k from $\{0, 1, \ldots, n-1\}$. If the item k is not already in the knapsack (current subset), then this proposed subset will just add item k, but if item k is already in the knapsack (current subset), the proposed subset will just remove item k from the knapsack (current subset).

- If the proposed subset is infeasible (doesn't fit in our knapsack because of the weight constraint), we return to the start of the loop and abandon the newly proposed subset.
- Suppose that the proposed subset is feasible. If the proposed subset has higher total value (is better) than the current subset, we will always transition to it (exploitation). Otherwise, if it is worse and $T > 0$, with probability $e^{\Delta/T}$, we update the current subset to the proposed subset, where $\Delta < 0$ is the decrease in total value. This allows us to transition to a "worse" subset occasionally (exploration), and get out of local optima! Repeat this for NUM_ITER transitions from the initial state (subset), and output the highest value subset found during the entire process (which may not be the final subset).

a) What is the size of the Markov Chain's state space $\mathcal{S}$ (the number of possible subsets)?

b) Let's try to figure out what the temperature parameter T does.

A. Suppose that $T = 0$. Will we ever get to a worse subset than before as we transition?

B. Suppose that $T > 0$.

i. For a fixed T, does the probability of transitioning to a worse subset increase or decrease with larger absolute values of Δ (larger absolute values means "more negative" values, since $\Delta < 0$)?

ii. For a fixed Δ, does the probability of transitioning to a worse subset increase or decrease with larger values of T?

iii. Explain briefly how the temperature parameter T controls the degree of exploration we do.

c) Implement the functions value, weight, and mcmc in mcmc_knapsack.py. To this end, you will use the edstem lesson. Remember that only code submitted via Gradescope will be graded.

Hints: To get full score, you **must** use np.random.rand() to generate an uniform value in $[0, 1]$, and np.random.randint(low (inclusive), high (exclusive)) to generate your random index(es). Make sure to read the documentation and hints provided!

One item that is not required or graded that we've included in the lesson for your interest is that we have called the make_plot function to make a plot where the x-axis is the iteration number, and the y-axis is the current knapsack value (not necessarily the current best), for ntrials=10 different runs of MCMC. The plots yield interesting phenomena - for example one can imagine tuning things to choose T that will most reliably produce high knapsack values.

Task 6 – Extra Credit: Let's Make a Hash of it

Suppose that you are planning to store a set of k-bit integers using a hash table of size n. Let $N = 2^k$. (There is nothing special about this choice.)

Hash Function: Fix a prime number p that is at least N. (From number theory, we know that there is at least one prime number between N and $2N$.) Choose your hash function at random using the following procedure:

- Choose an integer a such that $1 \leqslant a \leqslant p-1$ uniformly at random. Then, independent from the first choice, choose an integer b such that $0 \leqslant b \leqslant p-1$ uniformly at random.

- The hash function is then

$$h(x) = ((a \cdot x + b) \bmod p) \bmod n.$$

Show that for any two distinct k bit integers $x \neq y$

$$\mathbb{P}(h(x) = h(y)) \leqslant \frac{2}{n}.$$

Note that the probability here is over the random choice of a and b. You may use without proof the facts:

1. From CSE 311: If $gcd(z, p) = 1$, for each c there is a unique solution a modulo p of $a \cdot z \equiv c \pmod p$

2. For any two numbers $s, t \in \{0, 1, 2, \ldots, p-1\}$, and any $x \neq y \in \{0, 1, 2, \ldots, p-1\}$, there is a unique pair $0 \leqslant a \leqslant p-1$ and $0 \leqslant b \leqslant p-1$ such that

$$s = (a \cdot x + b) \bmod p \quad \text{and} \quad t = (a \cdot y + b) \bmod p,$$

which follows by applying the first fact to $z = x - y$.