Midterm

Due: 2025-4-10 23:59 | 4 Questions, 100+10 Pts

Name: Jiacong Fang, ID: 2200017849

Problem 1 (10')

For today's dinner, you are making a decision between two restaurants, *Yannan* and *Shaoyuan*. You know that one of them is a strictly better choice under all circumstances, and everyone prefers it to the other. However, you are the only person in the world who has no idea *which* is the better one. You decide to ask some folks on `treehole.pku.edu.cn` to give scores for the two restaurants. According to the following responses, which one will you choose?

- Alice gives a score of 84 for *Yannan*.
- Bob gives a score of 3.92 for *Shaoyuan*.
- Carol gives a score of 99999 for *Yannan*.
- Dave gives a score of 666 for *Shaoyuan*.
- Eve gives a score of $-\pi/6$ for *Yannan*.
- ...

At first glance, it seems impossible that one can do better than random guessing. However, we are going to show that this is not the case.
Formally, consider two score sequences $(x_1, \cdots, x_n), (y_1, \cdots, y_n) \in D^n$, where $D \subseteq \mathbb{R}$, such that either of the two cases is true:

- Case 0: $x_i < y_i,\ \forall i \in [n]$.
- Case 1: $x_i > y_i,\ \forall i \in [n]$.

We need to design a potentially randomized algorithm $\mathcal{A}$, whose output is either 0 or 1, to distinguish the two cases. Notably, we are restricted to "half-blind" algorithms: For each $i \in [n]$, algorithm $\mathcal{A}$ can only query one of the two values of $\{x_i, y_i\}$; once it decided to query one value, the other one in the i'th pair becomes inaccessible.

a. (3') Suppose $n = 1$ and $D = \mathbb{R}$. Design a half-blind algorithm $\mathcal{A}_1$ such that for any real numbers x, y,

$$\Pr[\ \mathcal{A}_1 \text{ is correct} \mid x_1 = x,\ y_1 = y\ ] > \frac{1}{2}.$$

b. (7') Suppose $D = [0, 100]$, $|x_i - y_i| \geq 1$. Design a half-blind algorithm $\mathcal{A}$ for sufficiently large n, and show that it has $1 - 1/2^{p(n)}$ success probability given any feasible input, where $p(n) = \Omega(n)$.

Direction: *For each i, your half-blind algorithms can decide freely which one of $\{x_i, y_i\}$ to observe. However, it can only observe one of the two values. You should give proofs for the error bounds.*

Problem 2 (15') (Knapsack Counting)

Recall the 0-1 Knapsack problem you have encountered before. We are given n objects of weight $0 \le a_1 \le \cdots \le a_n \le b$ where b is the total volume of the knapsack, and we are asked to find a 0-1 assignment $x_1, \cdots, x_n$ such that $\sum_{i=1}^{n} a_i x_i \le b$. In the counting version, we will compute the number of valid assignments, the set of which is denoted as S, instead of finding the optimal solution. The key idea here is similar to #DNF; that is, sampling from a restricted space. Suppose b is much larger than n^2.

We will use the *scaling method*. Define $a'_i := \lfloor n^2 a_i / b \rfloor$, then $0 \le a'_i \le n^2$. Note the floor operator may cause error when trying to recover a_i from a'_i, but the total error is no more than $n(b/n^2) = b/n$. Let $S' := \left\{ \boldsymbol{x} : \sum_{i=1}^{n} a'_i x_i \le n^2 \right\}$ be the set of solutions to the rescaled problem.

a. (2') Prove that $S \subseteq S'$.

Apparently, the converse does not necessarily hold. However, we claim that any $\boldsymbol{x} \in S'$ can be shoehorned into S by setting at most one of the x_i from 1 to 0.

b. (5') Prove the claim above, which implies $|S'| \le (n+1)|S|$.

In other words, with probability at least $1/(n+1)$, a uniformly random element of S' is in S. So the remaining task here is to compute $|S'|$ and sample from S'. Let $C(k, m) := \left| \left\{ \boldsymbol{x} : \sum_{i=1}^{k} a'_i x_i \le m \right\} \right|$.

c. (3') Design an algorithm based on dynamic programming that computes $C(i, j)$ for all $0 \le i \le n$ and $0 \le j \le n^2$. Your algorithm should take $O(n^3)$ time.

Definitely, we can fetch the value of $|S'|$ from $C(n, n^2)$. But in fact, we can also use this table to sample uniformly from S'.

d. (5') Based on the values of $C(\cdot, \cdot)$, design an algorithm which outputs a uniform sample of S'. Your algorithm should run in $O(n)$ time (excluding the time cost by the random generator).

Problem 3 (50') (Low Diameter Decomposition)

In this problem, you'll learn Low Diameter Decomposition, and use it to give approximate algorithm for some fundamental graph problem like Tree Embedding, All Pair Shortest Path (APSP).

Definition 1 (Low Diameter Decomposition (LDD))**.** *Given an undirected graph $G = (V, E)$, a **Low Diameter Decomposition (LDD)** scheme with approximation factor β and diameter bound D is a randomized algorithm that partitions V into disjoint clusters $V_1, V_2, \ldots, V_k$ satisfying:*

1. ***Bounded Diameter**: For each V_i, the induced subgraph $G[V_i]$ has diameter $\leq D$.*
2. ***Separation Probability**: For any $x, y \in V$,*

$$\Pr\left[x \text{ and } y \text{ lie in different clusters}\right] \leq \beta \cdot \frac{d_G(x,y)}{D},$$

where $d_G(x, y)$ denotes the shortest-path distance between x and y in G.

Remarks:

- The **diameter** of $G[V_i]$ is $\max_{u,v \in V_i} d_G(u, v)$.
- β balances cluster tightness (D) and separation likelihood. Lower β implies better decomposition quality.

a. (10') Prove that the following algorithm gives a LDD with approximation factor $\beta = O(\log n)$, with probability $\geq 1 - n^{-1}$.

Algorithm 1 Low Diameter Decomposition (LDD)

Require: Undirected graph $G = (V, E)$, target diameter $D > 0$
Ensure: Vertex partition $\{V_1, \ldots, V_k\}$ with $\text{diam}(G[V_i]) \leq D$

```
Initialize marked[v] ← FALSE for all v ∈ V
Sample r ~ Geometric(p) with p = min(1, 4 log n / D)
while ∃v ∈ V with ¬marked[v] do
    Select arbitrary unmarked vertex v_0 ∈ V
    Compute B ← {u ∈ V | d_G(v_0, u) ≤ r}
    Create cluster C ← B
    marked[u] ← TRUE for all u ∈ C
    Add C to output partition
return the computed clustering
```

Remarks: The naive way to run LDD takes time $O(n^3)$, since one need to run Dijsktra for n times. However, there are ways to do LDD in $\tilde{O}(n^2)$ time. You will get 10 Bonus Point if you can find out.

We will use this tool to solve approximate ASPS problem. First, we will introduce Low Stretch Tree.

Definition 2. *A randomized low-stretch tree of stretch α for a graph $G = (V, E)$ is a probability distribution $\mathcal{D}$ over spanning trees of G s.t.*

1. *$d_G(x, y) \leq d_T(x, y)$, for all T in the support $\mathcal{D}$.*
2. *$\mathbb{E}_{T \sim \mathcal{D}}[d_T(x, y)] \leq \alpha \cdot d_G(x, y)$, $\forall x, y \in V$*

Theorem 3. *For any metric space $M = (V, d)$, there exists an efficiently sampleable α_B-stretch spanning tree distribution $\mathcal{D}_B$, where*

$$\alpha_B = O(\log n \log \Delta_M)$$

Δ_M is defined as $\max_{x,y} d(x, y)$, we assume $\forall x \neq y, d(x, y) \geq 1$.

We will prove theorem 3 by the following algorithm.

Algorithm 2 Low Stretch Tree Construction, $\mathrm{LST}(M, \delta)$

Require: Metric space $M = (V, d)$, target diameter $D = 2^\delta$
Ensure: Spanning tree T with low stretch. **Invariant**: $\mathrm{diameter}(M) \leq 2^\delta$
1: **if** $|V| = 1$ **then**
2: **return** trivial tree containing the single point
3: Partition V into clusters $C_1, \ldots, C_t \leftarrow \mathrm{LDD}(M, D/2)$
4: **for** $j = 1$ **to** t **do**
5: Let M_j be M restricted to C_j
6: Recursively build $T_j \leftarrow \mathrm{LST}(M_j, \delta - 1)$
7: Connect roots $r_2, \ldots, r_t$ to r_1 with edges of length 2^δ
8: **return** final tree T rooted at r_1

Lemma 4. *If the random tree T returned by some call LST(M, δ) has root r, then*

1. *every vertex x in T has distance $d(x, r) \leq 2^{\delta+1}$*
2. *the expected distance between any $x, y \in T$ has $\mathbb{E}[d_T(x, y)] \leq 8\delta\beta d(x, y)$. (Recall that β is the approximate factor in LDD)*

If we can prove Lemma 4, then one can see from Problem a that if $\beta = O(\log n)$, then with high probability, $d_T(x, y) \geq d(x, y)$ for any x, y, thus theorem 3 can be proved.

b. (20') Prove the Lemma 4.

c. (10') Prove the following theorem.

Theorem 5. *There's an algorithm that output $O(\log n \log \Delta)$ approximation of ASAP on an undirected graph in $\tilde{O}(n^2)$ time, and success with probability $\geq 1 - \frac{1}{poly(n)}$.*

(This means, the algorithm output $d'(x, y)$ for any pair x, y, and satisfies $d(x, y) \leq d'(x, y) \leq \log n \log \Delta \cdot d(x, y)$, where $d(x, y)$ is the length of shortest path between x, y.)

d. (10') Prove the following theorem.

Theorem 6. *There's an algorithm that output $O(\log n)$ approximation of ASAP on an undirected graph in $\tilde{O}(n^2)$ time, and success with probability $\geq 1 - \frac{1}{poly(n)}$.*

Problem 4 (25') (Sub-Network Approximation for Two-Layer Neural Networks)

Definition 7 (Two-Layer Neural Network)**.** *Consider a wide neural network $f_M(x;\theta) : \mathbb{R}^d \to \mathbb{R}$ of the form*

$$f_M(x;\theta) := \sum_{j=1}^{M} a_j \sigma(w_j^\top x),$$

where $\theta = \{a_1, \ldots, a_M, w_1, \ldots, w_M\}$ denotes the parameters of the network, $a_j \in \mathbb{R}$ are the coefficients, and $w_j \in \mathbb{R}^d$ are the weights associated with the j-th neuron. Here, $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function, such as ReLU or sigmoid.

Definition 8 (Parameter Norm)**.** *The norm of the parameters θ is defined as*

$$\|\theta\|_{\mathcal{P}} := \sum_{j=1}^{M} |a_j| \|w_j\|_2,$$

where $\|w_j\|_2$ denotes the Euclidean norm of the weight vector w_j, and $\|\theta\|_{\mathcal{P}}$ is assumed to be finite.

Let $\pi = \mathrm{Unif}(\mathbb{S}^{d-1})$ denote the uniform distribution over the unit sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$. The goal is to approximate the network $f_M(x;\theta)$ with a sparse subset of the neurons, while controlling the error in expectation. Indeed, please prove the following result:

Theorem 9. *For any integer $m \in \mathbb{N}$, there exists a sparse vector $\tilde{a} = (\tilde{a}_1, \ldots, \tilde{a}_M)$ such that $\|\tilde{a}\|_0 = m$, i.e., the vector $\tilde{a}$ contains exactly m non-zero entries, which satisfies the following approximation bound:*

$$\mathbb{E}_{x\sim\pi} \left(\sum_{j=1}^{M} \tilde{a}_j \sigma(w_j^\top x) - \sum_{j=1}^{M} a_j \sigma(w_j^\top x) \right)^2 \lesssim \frac{\|\theta\|_{\mathcal{P}}^2}{dm}.$$

Remark 10. *In essence, this result demonstrates that it is possible to approximate the network with a sparse representation without incurring significant loss in performance, provided the number of selected neurons m is sufficiently large relative to the network's parameter norm $\|\theta\|_{\mathcal{P}}$, and the size of the input space d.*