CS-448 Sublinear Algorithms for Big Data Analysis December 15th, 2021

Lecture 13

Lecturer: Michael Kapralov *Scribes: Michael Kapralov*

We let $\mu^* \in (0,1]$ denote the kernel density of a dataset P in $\mathbb{R}^d$ at point $\mathbf{q} \in \mathbb{R}^d$:

$$\mu^* = K(P,\mathbf{q}) := \frac{1}{|P|}\sum_{\mathbf{p}\in P} K(\mathbf{p},\mathbf{q}).$$

It's good to have the Gaussian kernel $K(\mathbf{p},\mathbf{q}) = e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$ in mind for the rest of the lecture. Although the algorithm that we design works for general kernels. The algorithm is from [2].

1 Kernel Density Estimation Using Andoni-Indyk LSH

In this section, we present an algorithm for estimating KDE, using the Andoni-Indyk LSH framework. In order to state the main result of this section for general kernels, we need to define a few notions first. Thus, we state the main result for Gaussian kernel in the following theorem, and then state the general result, Theorem 8, after presenting the necessary definitions.

Theorem 1 *Given a kernel $K(\mathbf{p},\mathbf{q}) := e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$ for any $a > 0$, $\epsilon = \Omega\left(\frac{1}{\text{polylog}n}\right)$, $\mu^* = n^{-\Theta(1)}$ and a data set of points P, using Algorithm 1 for preprocessing and Algorithm 2 for the query procedure, one can approximate $\mu^* := K(P,\mathbf{q})$ (see Definition 3) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}\right)$, for any query point $\mathbf{q}$. Additionally, the space consumption of the data structure is*

$$\min\left\{\epsilon^{-2}n\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}, \epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}\right\}.$$

Throughout this section, we refer to Andoni-Indyk LSH's main result stated in the following lemma.

Lemma 2 ([1]) *Let $\mathbf{p}$ and $\mathbf{q}$ be any pair of points in $\mathbb{R}^d$. Then, for any fixed $r > 0$, there exists a hash family $\mathcal{H}$ such that, if $p_{\text{near}} := p_1(r) := \Pr_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p}-\mathbf{q}|| \le r]$ and $p_{\text{far}} := p_2(r,c) := \Pr_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p}-\mathbf{q}|| \ge cr]$ for any $c \ge 1$, then*

$$\rho := \frac{\log 1/p_{\text{near}}}{\log 1/p_{\text{far}}} \le \frac{1}{c^2} + O\left(\frac{\log t}{t^{1/2}}\right),$$

for some t, where $p_{\text{near}} \ge e^{-O(\sqrt{t})}$ and each evaluation takes $dt^{O(t)}$ time.

Remark From now on, we use $t = \log^{2/3} n$, which results in $n^{o(1)}$ evaluation time and $\rho = \frac{1}{c^2} + o(1)$. In that case, note that if $c = O\left(\log^{1/7} n\right)$, then

$$\frac{1}{\frac{1}{c^2} + O\left(\frac{\log t}{t^{1/2}}\right)} = c^2(1-o(1)).$$

Definition 3 *For a query $\mathbf{q}$, and dataset $P = \{\mathbf{p}_1,\ldots,\mathbf{p}_n\}$, we define*

$$\mu^* := K(P,\mathbf{q}) := \frac{1}{|P|}\sum_{\mathbf{p}\in P} K(\mathbf{p},\mathbf{q})$$

where for any $\mathbf{p} \in P$, $K(\mathbf{p},\mathbf{q})$ is a monotone decreasing function of $||\mathbf{q}-\mathbf{p}||$. Also, we define

$$w_i := K(\mathbf{p}_i,\mathbf{q}).$$

From now on, we assume that μ is a quantity such that

$$\mu^* \leq \mu \tag{1}$$

We also use variable $J := \left\lceil \log_2 \frac{1}{\mu} \right\rceil$.

Definition 4 (Geometric weight levels) *For any $j \in [J]$*

$$L_j := \left\{ \mathbf{p}_i \in P : w_i \in \left(2^{-j}, 2^{-j+1}\right] \right\}.$$

This implies corresponding distance levels (see Figure 1), which we define as follows

$$\forall j \in [J] : \; r_j := \max_{s.t. \;\; f(r) \in (2^{-j}, 2^{-j+1}]} r.$$

where $f(r) := K(\mathbf{p}, \mathbf{p}')$ for $r = ||\mathbf{p} - \mathbf{p}'||$. Also define $L_{J+1} := P \setminus \cup_{j \in [J]} L_j$.[1]

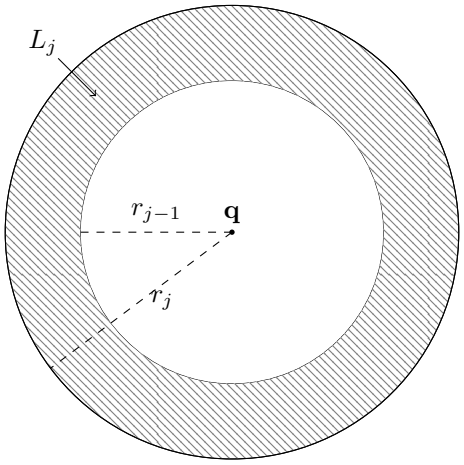

Figure 1: Illustration of definition of r_j's based on L_j's.

We start by stating basic bounds on collision probabilities under the Andoni-Indyk LSH functions in terms of the definition of geometric weight levels L_j (Definition 4):

Claim 5 *Assume that kernel K induces weight level sets, L_j's, and corresponding distance levels, r_j's (as per Definition 4). Also, for any query $\mathbf{q}$, any integers $i \in [J+1], j \in [J]$ such that $i > j$, let $\mathbf{p} \in L_j$ and $\mathbf{p}' \in L_i$. And assume that $\mathcal{H}$ is an Andoni-Indyk LSH family designed for near distance r_j (see Lemma 2). Then, for any integer $k \geq 1$, we have the following conditions:*

1. $\Pr_{h^* \sim \mathcal{H}^k} [h^*(\mathbf{p}) = h^*(\mathbf{q})] \geq p_{\text{near},j}^k$,
2. $\Pr_{h^* \sim \mathcal{H}^k} [h^*(\mathbf{p}') = h^*(\mathbf{q})] \leq p_{\text{near},j}^{kc^2(1-o(1))}$,

where $c := c_{i,j} := \min \left\{ \frac{r_{i-1}}{r_j}, \log^{1/7} n \right\}$ (see Remark 1) and $p_{\text{near},j} := p_1(r_j)$ in Lemma 2.

Proof If $\mathbf{p} \in L_j$ by Definition 4, we have

$$||\mathbf{q} - \mathbf{p}|| \leq r_j.$$

[1] One can see that $L_{J+1} = \{\mathbf{p}_i \in P : w_i \leq 2^{-J}\}$.

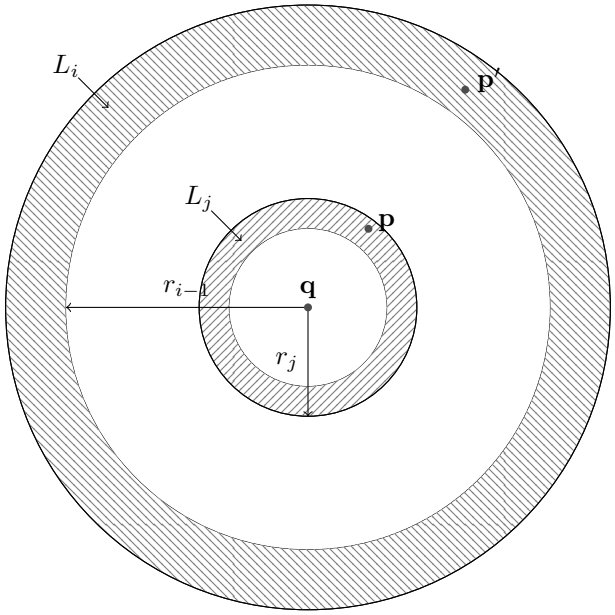

Figure 2: Illustration of r_j and r_{i-1} in terms of L_j and L_i.

Similarly using the fact that the kernel is decaying, for $\mathbf{p}' \in L_i$ we have

$$||\mathbf{q} - \mathbf{p}'|| \geq r_{i-1} \geq c \cdot r_j.$$

So, by Lemma 2 and Remark 1 the claim holds. Figure 2 shows an instance of this claim. ■

Now, we prove an upper-bound on sizes of the geometric weight levels, i.e., L_j's (see Definition 4).

Lemma 6 (Upper bounds on sizes of geometric weight levels) *For any $j \in [J]$, we have*

$$|L_j| \leq 2^j n\mu^* \leq 2^j n\mu.$$

Proof For any $j \in [J]$ we have

$$\begin{aligned} n\mu \geq n\mu^* &= \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q}) && \text{By Definition 3} \\ &\geq \sum_{i \in [J]} \sum_{\mathbf{p} \in L_i} K(\mathbf{p}, \mathbf{q}) \\ &\geq \sum_{\mathbf{p} \in L_j} K(\mathbf{p}, \mathbf{q}) \\ &\geq |L_j| \cdot 2^{-j} \end{aligned}$$

which proves the claim. ■

Definition 7 (Cost of a kernel) *Suppose that a kernel K induces geometric weight levels, L_j's, and corresponding distance levels, r_j's (see Definition 4). For any $j \in [J]$ we define* cost *of kernel K for weight level L_j as*

$$\text{cost}(K, j) := \exp_2 \left(\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2 (1 - o(1))} \right\rceil \right),$$

where $c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$. *Also, we define the general* cost *of a kernel* K *as*

$$\text{cost}(K) := \max_{j\in[J]} \text{cost}(K, j).$$

Description of algorithm: The algorithm runs in J phases. For any $j \in [J]$, in the j'th phase, we want to estimate the contribution of points in L_j to $K(P, \mathbf{q})$. We show that it suffices to have an estimation of the number of points in L_j. One can see that if we sub-sample the data set with probability $\min\{\frac{1}{2^j n\mu}, 1\}$, then in expectation we get at most $O(1)$ points from L_i for any $i \le j$. Now, assume that a point $\mathbf{p} \in L_j$ gets sampled by sub-sampling, then we want to use Andoni-Indyk LSH to distinguish this point from other sub-sampled points, efficiently. Thus, we want to find the appropriate choice of k for the repetitions of Andoni-Indyk LSH (see Claim 5). Suppose that we call Claim 5 with some k (which we calculate later in (3)). Then we have

$$\Pr_{h^*\sim\mathcal{H}^k}[h^*(\mathbf{p}) = h^*(\mathbf{q})] \ge p_{\text{near},j}^k,$$

which implies that in order to recover point $\mathbf{p}$ with high probability, we need to repeat the procedure $\widetilde{O}\left(p_{\text{near},j}^{-k}\right)$ times. Another factor that affects the run-time of the algorithm is the number of points that we need to check in order to find $\mathbf{p}$. Basically, we need to calculate the number of points that hash to the same bucket as $\mathbf{q}$ under h^*'s. For this purpose, we use the second part of Claim 5, which bounds the collision probability of far points, i.e., points such as $\mathbf{p}' \in L_i$ for any $i > j$. Intuitively, for any point $\mathbf{p}' \in L_i$ for any $i > j$, by Claim 5 we have

$$\Pr_{h^*\sim\mathcal{H}^k}[h^*(\mathbf{p}') = h^*(\mathbf{q})] \le p^{kc^2(1-o(1))}$$

where $c := c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$ and $p := p_{\text{near},j}$[2]. On the other hand, by Lemma 6, for $i = j+1, \ldots, J$ we have

$$|L_i| \le 2^i n\mu^* \le 2^i n\mu.$$

Then, one has the following bound,

$$\begin{aligned}
&\mathbb{E}\left[|\{\mathbf{p}' \in L_i : h^*(\mathbf{p}') = h^*(\mathbf{q})\}|\right] \\
&\le 2^i n\mu \cdot \frac{1}{2^j n\mu} \cdot p^{kc^2(1-o(1))} && \text{Sub-sampling and then applying LSH} \\
&= 2^{i-j} \cdot p^{kc^2(1-o(1))}. && (2)
\end{aligned}$$

Since we have $O\left(\log \frac{1}{\mu}\right)$ geometric weight levels, then the expression in (2) for the worst i, bounds the run-time up to $O\left(\log \frac{1}{\mu}\right)$ multiplicative factor. In order to optimize the run-time up to $\widetilde{O}(1)$ multiplicative factors, we need to set k such that the expression in (2) gets upper-bounded by $O(1)$ for all $i > j$. So, in summary, for any fixed $j \in [J]$, we choose k such that any weight level L_i for $i \ge j$ contributes at most $\widetilde{O}(1)$ points in expectation to the hash bucket of the query, i.e., $h^*(\mathbf{q})$. One can see that we can choose k as follows

$$k := k_j := \frac{-1}{\log p} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil. \tag{3}$$

For sampling the points in L_{J+1}, it suffices to sample points in the data set with probability $\frac{1}{n}$ (see line 17 in Algorithm 1), since the size of the sampled data set is small and there is no need to apply LSH. One can basically scan the sub-sampled data set.

[2]The indices are dropped for $c_{i,j}$ and $p_{\text{near},j}$ for ease of notation.

Algorithm 1 Preprocessing

procedure PREPROCESS(P, ϵ)
  ▷ P represents the set of data points
  ▷ ϵ represents the precision of estimation
  $K_1 \leftarrow \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$ ▷ C is a universal constant
  $J \leftarrow \left\lceil \log \frac{1}{\mu} \right\rceil$ ▷ We use geometric weight levels with base 2, see Definition 4
  for $a = 1, 2, \ldots, K_1$ **do** ▷ $O(\log n/\epsilon^2)$ independent repetitions
    for $j = 1, 2, \ldots, J$ **do** ▷ $J = \left\lceil \log \frac{1}{\mu} \right\rceil$ geometric weight levels
      $K_2 \leftarrow 100 \log n \cdot p_{\text{near},j}^{-k_j}$
      ▷ See Claim 5 and (3) for definition of $p_{\text{near},j}$ and k_j
      $p_{\text{sampling}} \leftarrow \min\{\frac{1}{2^j n\mu}, 1\}$
      $\widetilde{P} \leftarrow$ sample each element in P with probability p_{sampling}.
      for $\ell = 1, 2, \ldots, K_2$ **do**
        Draw a hash function from hash family $\mathcal{H}^{k_j}$ as per Claim 5 and call it $H_{a,j,\ell}$
        Run $H_{a,j,\ell}$ on $\widetilde{P}$ and store non-empty buckets
      end for
    end for
    $\widetilde{P}_a \leftarrow$ sample each element in P with probability $\frac{1}{n}$
    Store $\widetilde{P}_a$ ▷ Set $\widetilde{P}_a$ will be used to recover points beyond L_{J+1}
  end for
end procedure

Algorithm 2 Query procedure

procedure QUERY($P, \mathbf{q}, \epsilon, \mu$)
  ▷ P represents the set of data points
  ▷ ϵ represents the precision of estimation
  $K_1 \leftarrow \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$ ▷ C is a universal constant
  $J \leftarrow \left\lceil \log \frac{1}{\mu} \right\rceil$ ▷ We use geometric weight levels with base 2, see Definition 4
  for $a = 1, 2, \ldots, K_1$ **do** ▷ $O(\log n/\epsilon^2)$ independent repetitions
    for $j = 1, 2, \ldots, J$ **do** ▷ $J = \left\lceil \log \frac{1}{\mu} \right\rceil$ geometric weight levels
      $K_2 \leftarrow 100 \log n \cdot p_{\text{near},j}^{-k_j}$ ▷ See Claim 5 and (3) for definition of $p_{\text{near},j}$ and k_j
      for $\ell = 1, 2, \ldots, K_2$ **do**
        Scan $H_{a,j,\ell}(q)$ and recover points in L_j
      end for
    end for
    Recover points from L_{J+1} in the sub-sampled dataset, $\widetilde{P}_a$.
    $S \leftarrow$ set of all recovered points in this iteration
    for $\mathbf{p}_i \in S$ **do**
      $w_i \leftarrow K(\mathbf{p}_i, \mathbf{q})$
      if $\mathbf{p}_i \in L_j$ for some $j \in [J]$ **then**
        $p_i \leftarrow \min\{\frac{1}{2^j n \mu}, 1\}$,
      else if $\mathbf{p}_i \in P \setminus \cup_{j \in [J]} L_j$ **then**
        $p_i \leftarrow \frac{1}{n}$
      end if
    end for
    $Z_a \leftarrow \sum_{\mathbf{p}_i \in S} \frac{w_i}{p_i}$
  end for
end procedure

Now, we present the main result of this section.

Theorem 8 (Query time) *For any kernel K, the expected query-time of the algorithm is equal to $\widetilde{O}\left(\epsilon^{-2} n^{o(1)} \cdot \text{cost}(K)\right)$.*

Assuming Theorem 8, we prove Theorem 1.

Proof of Theorem 1: We first start by proving the query time bound and then we prove the space consumption of the data structure, and the guarantee over the precision of the estimator is given in Claim 11.

Proof of the query time bound: We calculate the cost of Gaussian kernel $e^{-a||\mathbf{x}-\mathbf{y}||_2^2}$. First, we present the weight levels and distance levels induced by this kernel. As per Definition 3, let

$$\mu^* := K(P, \mathbf{q}) = \sum_{\mathbf{p} \in P} e^{-a||\mathbf{p}-\mathbf{q}||_2^2}.$$

By Definition 4, one has

$$\begin{aligned} L_j &:= \left\{ \mathbf{p}_i \in P : w_i \in \left(2^{-j}, 2^{-j+1}\right] \right\} \\ &= \left\{ \mathbf{p}_i \in P : ||\mathbf{p}_i - \mathbf{q}||_2 \in \left[\sqrt{\frac{(j-1)\ln 2}{a}}, \sqrt{\frac{j \ln 2}{a}} \right) \right\}, \end{aligned}$$

which immediately translates to $r_j := \sqrt{\frac{j \ln 2}{a}}$ for all $j \in [J]$. Also, we for all $i \in [J+1], j \in [J]$ such that $i > j$, we have

$$\begin{aligned} c_{i,j} &:= \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\} \\ &= \min\left\{\sqrt{\frac{i-1}{j}}, \log^{1/7} n\right\} \end{aligned}$$

At this point, one can check that

$$\max_{j\in[J]} \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil = (1+o(1))\frac{1}{4}\log\frac{1}{\mu},$$

Therefore, the cost of Gaussian kernel is

$$\mathrm{cost}(K) = \left(\frac{1}{\mu}\right)^{(1+o(1))\frac{1}{4}}.$$

Now, invoking Theorem 8, the statement of the claim about the query time holds.

Proof of the space bound: First, since the query time is bounded by $\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, then the number of hash functions used is also bounded by the same quantity. This implies that the expected size of the space needed to store the data structure prepared by the preprocessing algorithm is $\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, since for each hash function we are hashing at most n points (number of points in the dataset).

For the other bound, we need to consider the effect of sub-sampling the data set. Fix $j \in [J]$. In the phase when we are preparing the data structure to recover points from L_j, we sub-sample the data set with probability $\min\{\frac{1}{2^j n\mu}, 1\}$, and then we apply $\widetilde{O}\left(p_{\mathrm{near},j}^{-k_j}\right)$ hash functions to this sub-sampled data set. Since

$$k_j = \frac{-1}{\log p} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil,$$

by (3), where $p = p_{\mathrm{near},j}$, we have

$$p_{\mathrm{near},j}^{-k_j} = \exp_2\left(\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil - j\right). \tag{4}$$

At the same time, the expected size of the sampled dataset is bounded by $n \cdot \min\{\frac{1}{2^j n\mu}, 1\} \le \frac{1}{\mu}\cdot 2^{-j}$. Putting this together with the equation above, we get that the expected size of the dataset constructed for level L_j is upper bounded by

$$\frac{1}{\mu}\exp_2\left(\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil - j\right). \tag{5}$$

Now for every $i = j+1, \ldots, J$ such that $c_{i,j} = \sqrt{\frac{i-1}{j}}$ one has

$$\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil - j = \max_{i=j+1,\ldots,J+1} \left\lceil j \cdot \frac{i-j}{(i-1)(1-o(1))} \right\rceil - j \le o(J),$$

and for the other values of i we have $\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{\log^{1/7} n(1-o(1))} \right\rceil - j \leq o(J)$ as well. Putting this together with (5) and multiplying by $J = O(\log(1/\mu)) = \mu^{-o(1)}$ to account for the number of choices $j \in [J]$, we get the second bound for the expected size of the data structure $\epsilon^{-2} \left(\frac{1}{\mu^*}\right)^{1+o(1)}$.

Proof of the precision of the estimator: First, we prove the following claim, which guarantees high success probability for recovery procedure.

Claim 9 (Lower bound on probability of recovering a sampled point) *Suppose that we invoke Algorithm 1 with (P, ϵ). Suppose that in line 11 of Algorithm 1, when $k = k^*$ and $j = j^*$, we sample some point $\mathbf{p} \in L_{j^*}$. We claim that with probability at least $1 - \frac{1}{n^{10}}$, there exists $\ell^* \in [K_2]$ such that $H_{k^*,j^*,\ell^*}(\mathbf{p}) = H_{k^*,j^*,\ell^*}(\mathbf{q})$.*

Proof By Claim 5 we have

$$\Pr_{h^* \sim \mathcal{H}^k} [h^*(\mathbf{p}) = h^*(\mathbf{q})] \geq p_{\text{near},j}^{k_j}.$$

Now note that we repeat this process for $K_2 = 100 \log n \cdot p_{\text{near},j}^{-k_j}$ times. So any point $\mathbf{p}$ which is sampled from band L_{j^*} is recovered in at least one of the repetitions of phase $j = j^*$, with high probability. ■

Now, we argue that the estimators are unbiased (up to small inverse polynomial factors)

Claim 10 (Unbiasedness of the estimator) *For every $\mu^* \in (0, 1)$, every $\mu \geq \mu^*$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, estimator Z_a for any $a \in [K_1]$ constructed in* QUERY$(P, \mathbf{q}, \epsilon, \mu)$ *(Algorithm 2) satisfies the following:*

$$(1 - n^{-9}) n\mu^* \leq \mathbb{E}[Z_a] \leq n\mu^*$$

Proof Let $\mathcal{E}$ be the event that every sampled point is recovered and let $Z := Z_a$ (see line 23 in Algorithm 2). By Claim 9 and union bound, we have

$$\Pr[\mathcal{E}] \geq 1 - n^{-9}$$

We have that $\mathbb{E}[Z] = \sum_{i=1}^n \frac{\mathbb{E}[\chi_i]}{p_i} w_i$ with $(1 - n^{-9}) p_i \leq \mathbb{E}[\chi_i] \leq p_i$, where we now define $\chi_i = 1$ if point $\mathbf{p}_i$ is sampled and recovered in the phase corresponding to its weight level, and $\chi_i = 0$ otherwise. Thus

$$(1 - n^{-9}) n\mu^* \leq \mathbb{E}[Z] \leq n\mu^*. \tag{6}$$

■

Remark We proved that our estimator is unbiased[3] for **any choice** of $\mu \geq \mu^*$. Therefore if $\mu \geq 4\mu^*$, by Markov's inequality the estimator outputs a value larger than μ at most with probability $1/4$. We perform $O(\log n)$ independent estimates, and conclude that μ is higher than μ^* if the median of the estimated values is below μ. This estimate is correct with high probability, which suffices to ensure that we find a value of μ that satisfies $\mu/4 < \mu^* \leq \mu$ with high probability by starting with some $\mu = n^{-\Theta(1)}$ (since our analysis assumes $\mu^* = n^{-\Theta(1)}$) and repeatedly halving our estimate (the number of times that we need to halve the estimate is $O(\log n)$ assuming that μ is lower bounded by a polynomial in n, an assumption that we make).

Claim 11 (Variance bounds) *For every $\mu^* \in (0, 1)$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, using estimators Z_a, for $a \in [K_1]$ constructed in* QUERY$(P, \mathbf{q}, \epsilon, \mu)$ *(Algorithm 2), where $\mu/4 \leq \mu^* \leq \mu$, one can output a $(1 \pm \epsilon)$-factor approximation to μ^*.*

[3]Up to some small inverse polynomial error.

Proof By Claim 10 and noting that $Z \leq n^2\mu^*$, where the worst case (equality) happens when all the points are sampled and all of them are recovered in the phase of their weight levels. Therefore,

$$\mathbb{E}\left[Z|\mathcal{E}\right] \cdot \Pr[\mathcal{E}] + n^2\mu^*(1 - \Pr[\mathcal{E}]) \geq \mathbb{E}[Z].$$

Also, since Z is a non-negative random variable, we have

$$\mathbb{E}\left[Z|\mathcal{E}\right] \leq \frac{\mathbb{E}\left[Z\right]}{\Pr[\mathcal{E}]} \leq \frac{n\mu^*}{\Pr[\mathcal{E}]} = n\mu^*(1 + o(1/n^9))$$

Then, we have

$$\begin{aligned}
\mathbb{E}[Z^2] &= \mathbb{E}\left[\left(\sum_{\mathbf{p}_i \in P} \chi_i \frac{w_i}{p_i}\right)^2\right] \\
&= \sum_{i \neq j} \mathbb{E}\left[\chi_i\chi_j \frac{w_i w_j}{p_i p_j}\right] + \sum_{i \in [n]} \mathbb{E}\left[\chi_i \frac{w_i^2}{p_i^2}\right] \\
&\leq \sum_{i \neq j} w_i w_j + \sum_{i \in [n]} \frac{w_i^2}{p_i}\mathbb{I}[p_i = 1] + \sum_{i \in [n]} \frac{w_i^2}{p_i}\mathbb{I}[p_i \neq 1] \\
&\leq \left(\sum_i w_i\right)^2 + \sum_{i \in [n]} w_i^2 + \max_i \left\{\frac{w_i}{p_i}\mathbb{I}[p_i \neq 1]\right\} \sum_{i \in [n]} w_i \\
&\leq 2n^2(\mu^*)^2 + \max_{j \in [J], \mathbf{p}_i \in L_j} \{w_i 2^{j+1}\} n\mu \cdot n\mu^* \\
&\leq 4n^2\mu^2 \qquad \text{Since } \mu^* \leq \mu
\end{aligned}$$

and

$$\mathbb{E}[Z^2|\mathcal{E}] \leq \frac{\mathbb{E}[Z^2]}{\Pr[\mathcal{E}]} \leq n^2\mu^{2-o(1)}(1 + o(1/n^9))$$

Now, since $\mu \leq 4\mu^*$, in order to get a $(1 \pm \epsilon)$-factor approximation to μ^*, with high probability, it suffices to repeat the whole process $K_1 = \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$ times, where C is a universal constant.

Suppose we repeat this process m times and $\bar{Z}$ be the empirical mean, then:

$$\begin{aligned}
\Pr[|\bar{Z} - \mu^*| \geq \epsilon n\mu^*] &\leq \Pr[|\bar{Z} - \mathbb{E}\left[Z\right]| \geq \epsilon\mu^* - |\mathbb{E}\left[Z\right] - n\mu^*|] \\
&\leq \Pr[|\bar{Z} - \mathbb{E}\left[Z\right]| \geq (\epsilon - n^{-9})n\mu^*] \\
&\leq \frac{\mathbb{E}\left[[]\,\bar{Z}^2\right]}{(\epsilon - n^{-9})^2(n^2\mu^*)^2} \\
&\leq \frac{1}{m}\frac{16n^2(\mu^*)^2}{(\epsilon - n^{-9})^2(n^2\mu^*)^2}
\end{aligned}$$

Thus by picking $m = O(\frac{1}{\epsilon^2})$ and taking the median of $O(\log(1/\delta))$ such means we get a $(1 \pm \epsilon)$-approximation with probability at least $1 - \delta$ per query.■

All in all, we proved the expected query time bound, the expected space consumption and the precision guarantee in the statement of the theorem. ■

References

[1] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 459–468. IEEE Computer Society, 2006.

[2] Moses Charikar, Michael Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 172–183. IEEE, 2020.