Data Structures for Density Estimation

Anders Aamand [*1] **Alexandr Andoni** [2] **Justin Y. Chen** [1] **Piotr Indyk** [1] **Shyam Narayanan** [1] **Sandeep Silwal** [1]

Abstract

We study statistical/computational tradeoffs for the following density estimation problem: given k distributions $v_1, \ldots, v_k$ over a discrete domain of size n, and sampling access to a distribution p, identify v_i that is "close" to p. Our main result is the first data structure that, given a sublinear (in n) number of samples from p, identifies v_i in time sublinear in k. We also give an improved version of the algorithm of (Acharya et al., 2018) that reports v_i in time linear in k. The experimental evaluation of the latter algorithm shows that it achieves a significant reduction in the number of operations needed to achieve a given accuracy compared to prior work.

1. Introduction

Finding the hypothesis that best matches a given set of samples, i.e., the *density estimation problem*, is a fundamental task in statistics and machine learning. In the finite setting, this task can be formulated as follows: given k distributions $v_1, \ldots, v_k$ over some domain U and access to samples from a distribution p over U, output v_i that is "close" to p. In the "proper" case, we assume that $p = v_j$ for some $1 \leq j \leq k$, and the goal is to output v_i such that $\|p - v_i\|_1 \leq \varepsilon$ for some error parameter $\varepsilon > 0$. In the more general "improper" case, p is arbitrary and the goal is to report v_i such that

$$\|p - v_i\|_1 \leq C \cdot \min_j \|p - v_j\|_1 + \varepsilon$$

for some constant $C > 1$ and error parameter $\varepsilon > 0$.

The density estimation problem has been studied extensively. The work of Scheffe (Scheffe, 1947), Yatracos (Yatracos, 1985), Devroye-Lugosi (Devroye & Lugosi, 2001) showed that $O(\log(k)/\varepsilon^2)$ samples are sufficient to solve the improper version of the problem for some constant $C > 1$ with a constant probability in $O(k^2 \log(k)/\varepsilon^2)$ time. Recently Acharya et al (Acharya et al., 2018) improved the time to $O(k \log(k)/\varepsilon^2)$ time.

Despite the generality of the formulation, this approach yields almost optimal sampling bounds for many natural classes of distributions such as mixtures of Gaussians (Daskalakis & Kamath, 2014; Suresh et al., 2014; Diakonikolas et al., 2019), see also the survey (Diakonikolas, 2016). Furthermore, the method has been extended to enable various forms of privacy (Canonne et al., 2019; Bun et al., 2019; Gopi et al., 2020; Kamath et al., 2020).

When the domain U is a discrete set of size n, the problem can be viewed as a variant of approximate nearest neighbor search over n-dimensional vectors under the L_1 norm, a problem that has been studied extensively (Andoni et al., 2018). Specifically, given a set of k distributions $v_1, \ldots, v_k$, the goal of C-approximate nearest neighbor search is to build a data structure that, given any query distribution p, returns v_i such that $\|p - v_i\|_1 \leq C \cdot \min_j \|p - v_j\|_1$.

However, in the density estimation problem, the query distribution p is not specified fully, but instead the algorithm is given only samples from p. This makes the problem substantially richer and more complex, as in addition to the usual *computational* resources (data structure space, query time), one also needs to consider the number of samples taken from p, a *statistical* resource. Thus, designing data structures for this problem involves making tradeoffs between the computational and statistical resources, with the known algorithms forming the endpoints of the tradeoff curve. In particular:

- If the goal is to optimize the *computational* efficiency of the data structure, one can learn the query distribution p up to an additive error of ε using $O(n/\varepsilon^2)$ samples (Kamath et al., 2015), and use any c-approximate nearest neighbor algorithm for the L_1 norm. In particular, the algorithm of (Andoni & Razenshteyn, 2015) yields (roughly) $O(nk + k^{1+1/(2c-1)})$ space and $O(nk^{1/(2c-1)})$ query time. This leads to an algorithm with polynomial space and sublinear (in k) query

[*] Authors listed alphabetically. [1] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. [2] Data Science Institute, Columbia University, New York, NY 10027, USA. Correspondence to: Anders Aamand <aamand@mit.edu>, Alexandr Andoni <andoni@mit.edu>, Justin Y. Chen <justc@mit.edu>, Piotr Indyk <indyk@mit.edu>, Shyam Narayanan <shyamsn@mit.edu>, Sandeep Silwal <silwal@mit.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

time, but at the price of using a linear (in n) number of samples.

- If the goal is to optimize the *statistical* efficiency of the data structure, then the aforementioned result of (Acharya et al., 2018) yields a data structure that uses only a logarithmic number of samples, but at the price of near linear (in k) query time.

These two point-wise results raise the question of whether "best of both worlds" data structures exists, i.e., whether there are data structures which are efficient in both statistical and computational terms. This is the focus of the paper.

Our results In this paper we initiate the study of computational/statistical tradeoffs of the (discrete) density estimation problem. Our main theoretical contributions are as follows:

- We present the first data structure that solves the *proper* version of the problem with polynomial space, sublinear query time *and* sublinear sampling complexity. This demonstrates that one can achieve non-trivial complexity bounds on all three parameters (space, query time, sampling complexity) *simultaneously*.

- We also present an improved version of the data structure from (Acharya et al., 2018) for the *improper* case, reducing its query time from $O(k\log(k)/\varepsilon^2)$ to $O(k/\varepsilon^2)$, i.e., linear in k, while retaining its optimal sampling complexity bound of $O(\log(k)/\varepsilon^2)$.

On the empirical side, we experimentally evaluate the linear-time algorithm and compare its performance to the (Acharya et al., 2018) algorithm on synthetic and on real networking data. These experiments display the practical benefits of the faster algorithm. For example, for synthetic data, we demonstrate that our faster algorithm achieves over **2x** reduction in the number of comparisons needed to achieve the same level of accuracy. Similarly, our experiments on network data show up to **5x** reduction in the number of comparisons.

Open questions We view a major part of the contribution of this paper as initiating the study of statistical/computational tradeoffs in data structures for density estimation. We design a data structure with polynomial space and sublinear query and sampling complexity but only in the proper case and only achieving query and sampling bounds that are slightly sublinear in the input parameters. *Extending these results (1) to the improper case, (2) with stronger sublinear bounds, or (3) showing lower bounds are all exciting open problems.*

1.1. Preliminaries

We assume the discrete distributions $v_1, \ldots, v_k$ over the domain $[n]$ are fully specified. We assume k, the number of distributions, is much larger than n but still polynomially related to n. For instance, $n^{1.01} \le k \le n^C$ for $C = 100$ suffices. This mimics the classic nearest neighbor search (NNS) literature setting, where the aim is to get algorithms sublinear in the dataset size, which exactly corresponds to k in our setting.

Let p be the unknown distribution we sample from. We actually attain results stronger than the truly proper case: we relax the equality condition and assume that there exists v^* among the v_i's such that $\|p - v^*\|_2 \le \frac{\varepsilon}{2\sqrt{n}}$.

We use the standard Poissonization trick and take $s' = \text{Pois}(s)$ samples [1]. This doesn't affect the sample complexity asymptotically as $s' = \Theta(s)$ with high probability. This is a standard trick used in distribution testing (Canonne, 2020; Szpankowski, 2011; Valiant & Valiant, 2011) and ensures that for all $i \in [n]$, the number of samples observed that are equal to i when sampling from p is distributed as $\text{Pois}(s \cdot p(i))$ and the counts are independent. We say $b \lesssim a$ for $a, b > 0$ if $b \le C \cdot a$ for some absolute positive constant C which does not depend on a or b.

We recall the guarantees of ℓ_∞ and ℓ_2 NNS data structures.

Theorem 1.1. *(Indyk, 2001) There exists some absolute constant C such that given a dataset $X \subset \mathbb{R}^n$ with $|X| = k$, for every $\delta \in (0,1)$, we can solve the $\frac{C}{\delta}\log\log n$-approximate nearest neighbor problem on X in ℓ_∞ using $\tilde{O}(nk^{1+\delta})$ space and $\tilde{O}(n)$ query time.*

The following theorem is obtained by using a standard combination of randomized dimensionality reduction theorem of Johnson-Lindenstrauss and locality-sensitive hashing (Har-Peled et al., 2012).

Theorem 1.2. *(Har-Peled et al., 2012) Given a dataset $X \subset \mathbb{R}^n$ with $|X| = k$, for every $c > 1$, we can solve the c-approximate nearest neighbor problem on X in ℓ_2 using $(k^{1+\rho} + kn)\log^{O(1)} k$ space and $(k^\rho + n)\log^{O(1)} k$ query time where $\rho \le 1/c$.*

Lastly, we define some convenient notation used shortly.

Definition 1.3. For a vector $x \in \mathbb{R}^n$ and a set $A \subseteq [n]$, let $x_A \in \mathbb{R}^m$ denote the vector which is equal to x in coordinates of set A and zero otherwise. $x(i)$ refers to the ith coordinate of the vector x.

2. Motivating our algorithm

As we discuss in the next section, our algorithm works by considering the empirical sample distribution $\hat{p}$ (where $\hat{p}_i = X(i)/s$ if i is sampled $X(i)$ times) and delicately combines nearest neighbor search data structures both for ℓ_2 and ℓ_∞ distance. Before describing our algorithm, we first

[1] $\text{Pois}(s)$ denotes a draw from a Poisson distribution with parameter s.

explain why simpler approaches do not work. First, since we are interested in finding a close distribution in ℓ_1 distance, it is a natural idea to simply construct an approximate ℓ_1 nearest neighbor data structure on the known distributions and directly querying it for $\hat{p}$. As the following lemma illustrates, this approach fails to return a close distribution even with access to an *exact* nearest neighbor data structure which returns a true nearest neighbour to the empirical distribution. Surprisingly, it can even be *much worse* than simply returning a uniformly random known distribution which would succeed with probability $1/k$. All proofs in this section are deferred to Appendix A.

Lemma 2.1. *For any $k \leq \binom{n}{n/2}$, there exist distinct distributions $p, q_1, \ldots, q_k$ over $[n]$ such that $\|p - q_i\|_1 = 1$ for each $i \in [k]$ and if $\hat{p}$ is the empirical distribution obtained from sampling $s = n/2$ elements from p, then the probability that $\|p - \hat{p}\|_1 \leq \min_{i \in [k]}(\|q_i - \hat{p}\|_1)$ is $\exp(-\Omega(k))$. In fact, if $k \geq Cn \log n$ for a sufficiently large constant C, the probability is 0.*

Another idea is to instead use an ℓ_2 nearest neighbor data structure on the empirical distribution. Perhaps less surprisingly, this approach fails too, but whereas for ℓ_1 the issue was the *light elements*, for ℓ_2 it is the *heavy elements*. For simplicity, we consider the case of two distributions.

Lemma 2.2. *There exists distributions p, q over $[n]$ with $\|p - q\|_1 = 1$ such that for any sample size $s = o(n)$, with probability $\Omega(1)$, $\|p - \hat{p}\|_2 > \|q - \hat{p}\|_2$.*

Finally, it should be clear that ℓ_∞ nearest neighbor search on the empirical distribution $\hat{p}$ does not work with $s = o(n)$ samples. Indeed, consider distributions where all probabilities are either $\frac{2}{n}$ or 0 and the most sampled coordinate i. Even with an exact nearest neighbor data structure, we could return an arbitrary distribution q such that $q(i) = 2/n$.

3. Main algorithm and proof intuition

As motivated by the previous section, our algorithm deals with 'heavy' and 'light' elements separately. The first challenge is to formally define the notions of heavy and light elements. The most natural choice is to declare a domain element heavy for a distribution v if the probability mass v places on the element is larger than some threshold γ. An issue arises when we want to employ this definition for p, the distribution we are sampling from. Since we only get sample access to p, we will have tiny, but non-negligible, error on estimates of the probabilities of p, which introduces some ambiguity on the exactly partition of the domain into heavy and light elements. Thus, we are motivated seek an unequivocal definition of heavy and light.

Towards this goal, we can define a heavy and light partitioning of $[n]$ according to one of the fixed distributions v_i, which we know fully, via Algorithm 1. It partitions the domain $[n]$ with respect to a fixed threshold γ based on the probability masses of v_i. When we sample from p, we wish to use one of these partitions of the domains (obtained from inputting the v_i's into Algorithm 1) as the definition of heavy and light elements for p. However, we still wish to ensure that any element that we define as heavy for p does not possess probability mass significantly greater than γ. Lemma B.3 proves that $\hat{p}$ and p are $\tilde{O}(1/\sqrt{n})$ close in ℓ_∞, where $\hat{p}$ is the empirical distribution after sampling from p. Let v^∞ denote the closest distribution to $\hat{p}$ among the $\{v_i\}_{i=1}^k$ in ℓ_∞ distance. As there exists a choice of v^∞ (namely v^*) which is $O(1/\sqrt{n})$ close to p (and $\hat{p}$) in ℓ_∞, $\|v^\infty - p\|_\infty \leq O(1/\sqrt{n})$. Hence, if we use the heavy/light partitioning of v^∞, then every domain element classified as heavy for p has probability mass at least $\gamma - O(1/\sqrt{n})$ and no light element has mass more than $\gamma + O(1/\sqrt{n})$.

The above discussion naturally leads us to our preprocessing algorithm, Algorithm 2. In Algorithm 2, we first group all the known distributions $\{v_i\}_{i=1}^k$ into k groups such that the ith group consists of all distributions which are close to v_i in ℓ_∞ distance. Let v^∞ denote the closest distribution to $\hat{p}$ in ℓ_∞ and consider v^∞'s group, denoted as S_∞. We essentially use the heavy / light partitioning of the domain induced by v^∞ as the definition of heavy or light for p. All distributions in v^∞'s group will also share the same heavy / light partitioning as v^∞. By adjusting the log factors, we can ensure that v^* is also included in v^∞'s group. Hence, we only need to consider the distributions in S_∞.

The key observation is that we can now *completely* disregard the heavy elements. This is because p will be sufficiently close to *all* distributions in S_∞ in ℓ_1 distance, restricted to the heavy elements. This is formalized in Lemma 3.4.

Thus, it suffices to consider the distributions in S_∞, restricted to the light domain elements. It turns out that ℓ_2 is a suitable metric to use on the light elements. Therefore, our main algorithm simply performs an ℓ_2 nearest neighbor using query $\hat{p}$ and searches for the closest distribution in S_∞ to $\hat{p}$, restricted to the light domain elements. Since we know v^* is in S_∞, we can guarantee that we will find a distribution which is 'close' in ℓ_1 distance on the light elements. We conclude by combining the fact that all distributions in S_∞ are close to p on the heavy elements already, and thus the total ℓ_1 error must be small.

To summarize, the overall algorithm outline is the following. In the preprocessing step, we create an ℓ_∞ data structure on all the known distributions $\{v_i\}_{i=1}^k$. Then we group all of the known distributions into k groups, one for each v_i, with the property that all distributions in a fixed group are $\tilde{O}(1/\sqrt{n})$ close in ℓ_∞ distance. We additionally instantiate an ℓ_2 nearest neighbor data structure within each group, but only on the light domain elements, which is defined consistently within each group (using v_i for v_i's group). See

Algorithm 2 for details.

For the final algorithm, presented formally in Algorithm 3, we take s samples from the unknown distribution p and let $\hat{p}$ denote the empirical distribution. We query the ℓ_∞ nearest neighbor search data structure on $\hat{p}$ which helps us narrow down to a fixed group of distributions close on the heavy elements, as defined in the preprocessing stage. We then query the ℓ_2 NNS data structure for this group on $\hat{p}$, but restricted to the light domain elements in the group[2]. The returned distribution is our output for the closest distribution to p. The main guarantee of our algorithms is given below.

Theorem 3.1. *Set $s = \Theta\left(\frac{n}{\varepsilon^2(\log k)^{1/4}}\right) = o(n)$ in Algorithm 3 and $\gamma = 1/n^{5/12}$ in Algorithm 2. Let $\tilde{v}$ denote the output of Algorithm 3. Then the preprocessing algorithm, Algorithm 2, runs in time polynomial in k, requires $\tilde{O}(nk^2)$ space, and the query time of Algorithm 3 is $\tilde{O}(n) + k^{1-1/(\log k)^{1/4}} = o(k)$. Furthermore, with probability $1 - o(1)$, Algorithm 3 returns distribution $\tilde{v}$ satisfying $\|p - \tilde{v}\|_1 \le \varepsilon$.*

In summary, we use polynomial preprocessing time, $s = o(n)$ samples, and our query time is $o(k)$, with the latter two quantities being sublinear in the domain size and the number of distributions, respectively. We present auxiliary lemmas in Sections 3.1 and 3.2 (proofs in Appendices B and C, respectively) and prove Theorem 3.1 in Appendix D.

Algorithm 1 Heavy light decomposition

```
1: Input: An ordered list of distributions $q_1, \dots, q_t$
2: Output: Vectors $(q_i)_H, (q_i)_L$ for every $i \in [t]$ (recall Definition 1.3)
3: procedure HEAVY-LIGHT($\{q_i\}_{i=1}^t$)
4:     $H, L \leftarrow \emptyset$
5:     for $j = 1$ to $n$ do
6:         if $q_1(j) \ge \gamma$ then    ▷ We use $q_1$ to define the heavy and light elements for all $q_1, \dots, q_t$.
7:             $H \leftarrow H \cup \{j\}$
8:         else
9:             $L \leftarrow L \cup \{j\}$
10:        end if    ▷ $H \cup L$ is a disjoint partition of $[n]$.
11:    end for
12:    Return: vectors $\{(q_i)_H, (q_i)_L\}_{i=1}^t$    ▷ We use the distribution $q_1$ to set the heavy and light domain elements for all other distributions.
13: end procedure
```

3.1. Auxiliary lemma for heavy elements

Lemma 3.2 shows that any pair of distributions in a fixed group, as defined in the preprocessing step, are close in ℓ_1 distance, when the ℓ_1 distance is restricted to the heavy domain elements in the group.

[2]To ensure independence, we actually query using an empirical distribution using fresh samples.

Algorithm 2 Preprocessing

```
1: Input: Distributions $\{v_i\}_{i=1}^k$ according to Section 1.1
2: Output: An $\ell_\infty$ data structure, a number of $\ell_2$ NNS data structures such that each $v_i$ is associated with a unique $\ell_2$ data structure
3: procedure PREPROCESSING($\{v_i\}_{i=1}^k$)
4:     $D^\infty \leftarrow \ell_\infty$ data structure on $\{v_i\}_{i=1}^k$
5:     for all $i \in [k]$ do
6:         $S_i \leftarrow \{v' \in \{v_i\}_{i=1}^k \mid \|v_i - v'\|_\infty \le O((\log n)^2 \cdot (\log\log n)/\sqrt{n})\}$
7:     end for
8:     for $j = 1$ to $k$ do
9:         Write $S_j = \{w_1, \dots, w_t\}$ where $w_1 = v_j$
10:        $\{(w_i)_H, (w_i)_L\}_{i=1}^t \leftarrow$ Heavy-Light($S_j$)
11:        $D_j \leftarrow \ell_2$ NNS data structure on $\{(w_i)_L\}_{i=1}^t$ ▷ $S_j$ corresponds to a well defined partition of $[n]$ into heavy and light elements based on $v_j$ via Algorithm 1
12:    end for
13:    Return: $D^\infty$, data structures $\{D_j\}_{j=1}^k$
14: end procedure
```

Algorithm 3 Sublinear Time Hypothesis Selection

```
1: Input: Distributions $\{v_i\}_{i=1}^k$; preprocessed data structures $D^\infty, \{D_j\} \leftarrow$ Preprocessing($\{v_i\}_{i=1}^k$); Poi($s$) samples from query distribution $p$
2: Output: A distribution $v_j$
3: procedure SUBLINEAR-HYPOTHESIS-
4: SELECTION($\{v_i\}_{i=1}^k$)
5:     $\hat{p} \leftarrow$ empirical distribution of the first half of the samples from $p$
6:     $v^\infty \leftarrow$ output of $D^\infty$ on query $\hat{p}$ with approximation $c = O(\log(n) \cdot \log\log n)$
7:     $D' \leftarrow \ell_2$ NNS data structure corresponding to $v^\infty$
8:     $L \leftarrow$ the light domain elements for $D'$
9:     $\hat{p}' \leftarrow$ empirical distribution of the second half of the samples from $p$
10:    $\tilde{v} \leftarrow$ output of $D'$ on $\hat{p}'_L$ with approximation $c = 1 + \frac{s\varepsilon^2}{32n}$
11:    Return: $\tilde{v}$
12: end procedure
```

Lemma 3.2. *Suppose $\gamma = 1/n^C$ in Algorithm 1. Consider the sets $S_j = \{w_1, \dots, w_t\}$ defined in line 9 of Algorithm 2. Consider the vectors $(w_i)_H$, which are the heavy subsets of the distributions in S_j, as defined in line 10 of Algorithm 2. Then for all j and for all $w, w' \in S_j$, we have*

$$\|w_H - w'_H\|_1 \le O\left(\frac{(\log n)\log\log n}{n^{1/2-C}}\right).$$

Lemma 3.3 shows that v^*, the distribution close to p as defined in Section 1.1, must belong to the same group as v^∞, the distribution returned after querying $\hat{p}$ in the ℓ_∞ data structure in Algorithm 3.

Lemma 3.3. *Suppose $s \ge \Omega(n/(\log n)^{1/2})$. Consider the output v^∞ on $\hat{p}$ when inputted into the ℓ_∞ data structure D^∞, as done in line 6 of Algorithm 3. Let S_∞ denote the group of v^∞ from Algorithm 2. Assuming the event in Lemma B.3 holds, it must be that $v^* \in S_\infty$.*

The final 3.4 below argues that S_∞, the group of v^∞ in the preprocessing algorithm, has the property that p must be close in ℓ_∞ distance to every member of S_∞. Consequently, p must be close to every distribution in S_∞ in ℓ_1 distance, restricted to the heavy elements of the group.

Lemma 3.4. *Consider the same setting as in Lemma 3.3 and suppose $\gamma = 1/n^C$. Let H denote the heavy domain elements of group S_∞. Then for all $w \in S_\infty$, we have $\|p - w\|_\infty \le O\left(\frac{(\log n)^2 \log\log n}{\sqrt{n}}\right)$. Consequently, we also have $\|p_H - w_H\|_1 \le O\left(\frac{(\log n)^2 \log\log n}{n^{1/2-C}}\right)$.*

Essentially, the lemmas above ensure that v^* will be in the group S_∞ which in turn has the property that for all the distributions are close in ℓ_1 on the heavy elements. In order to actually find the closest distribution in ℓ_1, we have to switch our attention to the light elements within the group and for this we need the lemmas of the next section.

3.2. Auxiliary lemmas for light elements

We now state auxiliary lemmas concerning light elements. Recall the definition of L from line 8 in Algorithm 3. We first show the expected value of $\|s \cdot \hat{p}'_L - s \cdot v_L\|_2$ captures the ℓ_2 distance between p and v, restricted to the light elements. Note that we are using the empirical distribution $\hat{p}'$ which does not share any samples with $\hat{p}$, insuring the independence of L and $\hat{p}'$. We also define T in the lemma below as $T = \sum_{i \in L} p(i)$.

Lemma 3.5. $\mathbb{E}[\|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2] = s \cdot T + s^2 \|p_L - v_L\|_2^2$.

We now derive concentration of the estimator around its expected value. First we consider the case where v is sufficiently close to p.

Lemma 3.6. *Suppose p satisfies $\|p_L - v_L\|_2 \le \frac{\varepsilon}{2\sqrt{n}}$. Set $t = \frac{s^2\|p_L - v_L\|_2^2}{4} + \frac{s^2\varepsilon^2}{4n}$ and $Z_1 = \|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2$. If $s = \Omega\left(\max\left(\frac{n\|p_L\|_2}{\varepsilon^2}, \frac{n^{2/3}}{\varepsilon^{4/3}}\right)\right)$ then $\mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \ge t] \le 0.01$.*

Next we consider the case where $\varepsilon/\sqrt{n} \lesssim \|p_L - v_L\|_2$. In this case, we need to obtain a stronger concentration result since later we union bound over possibly $\Omega(k)$ different distributions which are in group S_∞.

Lemma 3.7. *Suppose p satisfies $\|p_L - v_L\|_2 \ge 0.75 \cdot \frac{\varepsilon}{\sqrt{n}}$. Set $t = \frac{s^2\|p_L - v_L\|_2^2}{4} + \frac{s^2\varepsilon^2}{4n}$, suppose $\gamma = 1/n^{5/12}$, and let $Z_1 = \|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2$ denote our estimator. If $s = \Omega\left(\frac{n^2\gamma^3 \log(k)^2}{\varepsilon^4}\right)$ then $\mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \ge t] \le 1/\text{poly}(k)$.*

Putting it all together: Proof of Theorem 3.1 We give a high level outline of the proof by showing how to combine the prior results to prove the main theorem and defer the full proof to Appendix D. See Section 3 for additional inituition.

Lemma 3.4 states that we can essentially ignore the heavy elements in the group S_∞ and furthermore, Lemma 3.3 proves that $v^* \in S_\infty$. Thus we can restrict our attention to the light elements of the group belonging to v^∞. Lemma 3.5 states that the expected value of the (scaled) ℓ_2 distance squared from $\hat{p}'_L$, the empirical distribution, to v^*_L is at most $s + s^2\|p_L - v^*_L\|_2^2 \le s + s^2\varepsilon^2/(4n)$ (assuming T is a constant for the sake of simplicity of the discussion). On the other hand, the expected value of the same statistic for any distribution v in S^∞ which is 0.99ε far from p in ℓ_1 has to be at least $s + s^2\|p_L - v_L\|_2^2 \ge s + (0.99)^2 s^2\varepsilon^2/n$. This is bigger than the corresponding value for v^* by a factor of roughly $1 + O(\varepsilon^2 s^2/n)$. These calculations are only true in expected value so we use Lemmas 3.6 and 3.7 to prove that the quantities are close to their expected value with high probability. Since the values of the ℓ_2 statistic are sufficiently far apart in these two cases, the ℓ_2 NNS data structure outputs a close distribution $\tilde{v}$ satisfying the theorem guarantees.

4. Improving the runtime for classic hypothesis testing

We now switch our focus to algorithms for the classic hypothesis selection problem in the *improper* setting. Recall from Section 1.1 that we are given a set of k distributions $\mathcal{V} = \{v_1, \ldots, v_k\}$ over $[n]$ and a set of samples $\mathcal{S}$ from some distribution p also over $[n]$. Given the samples, our goal is to output $\hat{v} \in \mathcal{V}$ such that $\|p - \hat{v}\|_1 \le C \cdot \min_j \|p - v_j\|_1 + \varepsilon$ with probability at least $1 - \delta$ for some constant C and some small parameters ε and δ. In the case $k = 2$, this problem can be solved using the so called Scheffe test (Scheffe, 1947) which we will discuss shortly. This test uses $O(\frac{\log 1/\delta}{\varepsilon^2})$ samples and returns a $\hat{v} \in \mathcal{V}$ satisfying the above bound with $C = 3$. In (Devroye & Lugosi, 2001) this was extended to the case $k > 2$. They showed that with $s = O((\log k + \log \frac{1}{\delta})/\varepsilon^2)$ samples, running the Scheffe test for each pair of distributions and outputting the one with the most wins, yields an estimate that satisfies the bound above with $C = 9$. This is referred to as the Scheffe tournament. The running time for this tournament, clearly scales with $O(k^2)$. However, using a knockout tournament style algorithm, (Suresh et al., 2014) showed that the running time can be reduced to $O\left(\frac{k}{\varepsilon^2}(\log k + \log \frac{1}{\delta})\right)$ for a fairly large constant C even when we are only given sample access to the distributions in $\mathcal{V}$. In (Acharya et al., 2018) the authors showed that a simple algorithm `Quick-Select` obtains $C = 9$ with the same running time and sample size. Here we show how to reduce this runtime to $O\left(\frac{k}{\varepsilon^2}\log\frac{1}{\delta}\right)$ with a slight blow-up in the value of C. For instance, when $\delta = \Omega(1)$ this shaves off a factor of $\log k$ of the running time. To obtain this bound, we will allow our algorithm a simple preprocessing step, namely for each pair of distributions (v_i, v_j), we compute and store the total variation distance between them. This can be done in total time $O(k^2 n)$.

To achieve this improvement, we recall the classic Scheffe test (Scheffe, 1947). This test can be seen as an algorithm for the hypothesis selction problem for the case $k = 2$. The algorithm takes as input two distributions v_1 and v_2 over $[n]$ and samples from an unknown distribution p over $[n]$. Let's define $S = \{i \in [n] : v_1(i) > v_2(i)\}$. Upon receiving the samples, the algorithm computes μ_S the frequency of samples in S. It then outputs v_1 if $|v_1(S) - \mu_S| \le |v_2(S) - \mu_S|$. Otherwise, it outputs v_2. Note that the values $v_1(S)$ and $v_2(S)$ can be computed directly from the total variation distance between v_1 and v_2. It was shown in (Devroye & Lugosi, 2001) that for $k = 2$ and with s samples, with probability $1 - \delta$ the Scheffe test outputs a distribution v satisfying that

$$\|p - v\|_1 \le 3 \cdot \min_{j\in\{1,2\}} \|p - v_j\|_1 + \sqrt{\frac{10\log\frac{1}{\delta}}{s}}. \quad (1)$$

Except for a simple modification, our algorithm is similar to the algorithm proposed in (Suresh et al., 2014). Assume for simplicity that k is a power of two — this assumption can easily be dispensed with. Define $\delta_i = \delta/4^i$ and $s_i = \frac{10\log(1/\delta_i)}{\varepsilon^2}$. Finally, define $\mathcal{V}_1 = \mathcal{V}$. Our algorithm first initializes a set $\mathcal{C} \leftarrow \emptyset$. Then for $i = 1, \dots, \lg k$, it performs the following steps:

1. Randomly select a subset of $\min\{k^{1/3}, |\mathcal{V}_i|\}$ elements from $|\mathcal{V}_i|$ and move them to $\mathcal{C}$.
2. Randomly form $|\mathcal{V}_i|/2$ pairs of distributions in $\mathcal{V}_i$ and run the Scheffe test on each pair using the set $\mathcal{S}_i$ consisting of the first s_i elements of the sample.
3. Define $\mathcal{V}_{i+1}$ to be the set of $|\mathcal{V}_i|/2$ winners.

Finally, our algorithm runs the Scheffe test on all pairs of distributions in $\mathcal{C}$ using a single set of $\frac{10\log\left(\binom{|\mathcal{C}|}{2}/\delta\right)}{\varepsilon^2}$ new samples from p (independent of the samples used in step 2. above). It outputs the distribution $\hat{v}$ in $\mathcal{C}$ with the most wins among these comparisons (breaking ties arbitrarily).

The difference between this algorithm and the algorithm in (Suresh et al., 2014) is that we do not use the entire sample in the lower levels of the tournament tree. Intuitively, for a subtree of the tournament tree containing 2^i distributions, we only need to consider a sample of size s_i in order to union bound over the bad events that the distribution in $\mathcal{V}$ closest to p loses to either of the 'far' distributions in this subtree. As running a single Scheffe test with a sample of size s takes $O(s)$ time, the running time used for the knockout tournament is therefore $O\left(\sum_{i=1}^{\lg k} s_i k/2^i\right) = O\left(\frac{k}{\varepsilon^2}\log\frac{1}{\delta}\right)$. We defer the full proof to Appendix E.

Theorem 4.1. *Assume that $\delta \ge k^{-1/4}$. With probability $1 - O(\delta)$, the algorithm of Section 4 outputs a distribution $\hat{v}$ with $\|p - \hat{v}\|_1 \le 27 \cdot \min_j \|p - v_j\|_1 + O(\varepsilon)$. The algorithm uses $s = O((\log k + \log\frac{1}{\delta})/\varepsilon^2)$ samples from p and has running time $O\left(\frac{k}{\varepsilon^2}\log\frac{1}{\delta}\right)$.*

5. Experiments

We experimentally evaluate the faster tournament algorithm given in Section 4 and compare its performance to the base knockout tournament (Suresh et al., 2014) on synthetic and on real networking data.[3] These experiments display the practical benefits of the faster algorithm.

The algorithms have several key parameters which we vary throughout the experiments. *nAllPairs* is the number of distributions in each level of the tournament which are randomly sampled to compete in an all-pairs tournament at the end of the algorithm. This parameter is used in both the base tournament and our fast tournament. *fastConst* controls the number of samples used in each level of the fast tournament. In particular, at the ith level, the fast tournament uses *fastConst* $\cdot\, i$ samples for each Scheffe test. While adjusting these parameters slightly does not change the constant factors in the analysis in Section 4, we find they make a large impact empirically and thus test the algorithms under various parameter settings.

We measure computation cost by the number of Scheffe operations performed by each tournament. One Scheffe operation is one comparison of a pair of distributions at a given sampled element (the basic computation performed during a Scheffe test). Below, we describe the experimental setup for the datasets.

5.1. Synthetic Experiments

Setup We compare two synthetic datasets corresponding to *half-uniform* and *Zipfian* distributions. The half-uniform dataset consists of $k = 8192$ distributions over a domain of size $n = 500$. Each distribution is uniform over a random $n/2$ sized subset of the domain. We consider a number of samples $s \in \{20, 30, 40, 50, 60\}$.

The Zipfian dataset consists of $k = 4096$ distributions over

[3]Code available at https://github.com/justc2/datastructdensityest

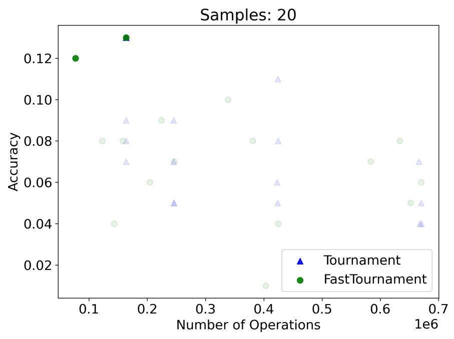

Figure 1. Grid search on half-uniform data with 20 samples.

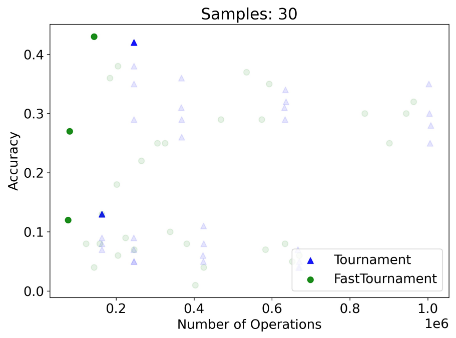

Figure 2. Grid search on half-uniform data with up to 30 samples.

a domain of size $n = 250$. Each distribution is a random permutation of the standard Zipfian distribution where the ith element has probability proportional to $1/i$. We consider a number of samples $s \in \{20, 30, 40\}$.

In both cases, queries are formed by taking samples from one of the distributions in the dataset and performance is measured by the accuracy of the algorithms (i.e., the fraction of queried points for which the algorithm returned the true distribution). As runtime can always be decreased by not using all of the sampled elements, for results using s samples, we also report all results using fewer than s samples. For these experiments, we grid search over *fastConst* $\in \{5, 10, 15, 20\}$ and *nAllPairs* $\in \{0, 10, 20, 30\}$. The reported results are averaged over 5 random sets of 20 queries each.

Results We will focus on the results for the half-uniform dataset as the conclusions for the Zipfian dataset are similar (see specific comments at the end of this subsection). In Figures 1-5, we compare the accuracy and computational cost of the base tournament and our fast tournament under the various parameters setting described above. For both algorithms, the upper envelope of points (i.e., the discrete approximation of the Pareto curve for the accuracy/time tradeoff) are highlighted. As the number of samples increases, the accuracy both algorithms dramatically increases

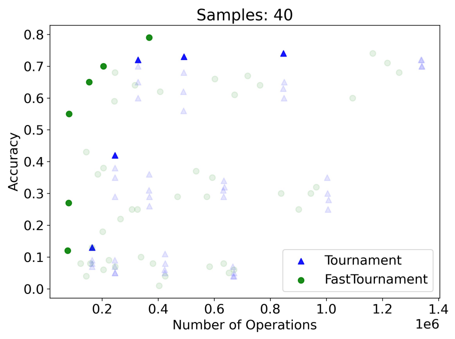

Figure 3. Grid search on half-uniform data with up to 40 samples.

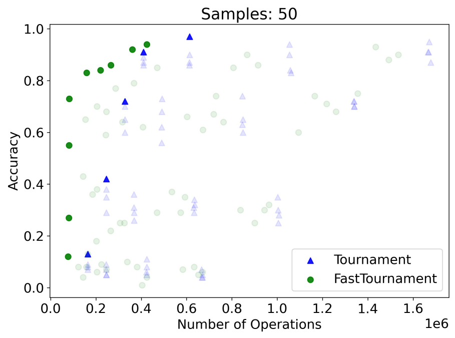

Figure 4. Grid search on half-uniform data with up to 50 samples.

from 13% at 20 samples up to 98% at 60 samples.

Across different levels of sampling, the fast tournament is able to attain similar accuracy to the base tournament while using significantly fewer samples. For example, at 60 samples, the fast tournament is able to achieve accuracy of 88% using fewer than 165,000 operations while the slow tournament requires more than 400,000 operations to achieve accuracy greater than 80% (an improvement of more than $2.4\times$). In general, we find that the most important factor influencing accuracy is the number of total samples. On the other hand, using fewer samples in earlier rounds of the tournament via small *fastConst* has a moderate to negligible impact on accuracy. Finally, the points of the right side of the plots with many operations correspond to larger values of *nAllPairs*. Increasing *nAllPairs* (at least beyond a certain point) has small impact on accuracy while significantly increasing computation due to the quadratic dependence on the all-pairs comparison at the end of the tournament.

The overall results for the Zipfian datset in Figures 6-8 are very similar though the Zipfian distributions are qualitatively different from the half-uniform distributions as they contain several elements with quite large probabilities.

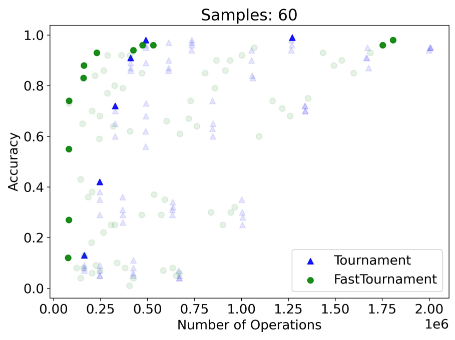

Figure 5. Grid search on half-uniform data with up to 60 samples.

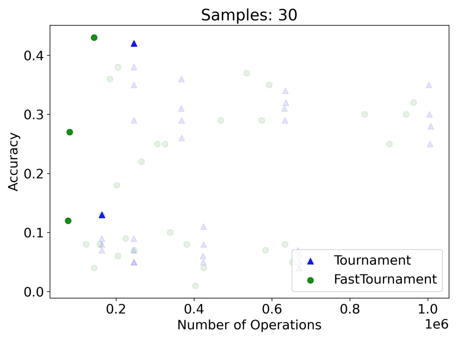

Figure 7. Grid search on Zipfian data with up to 30 samples.

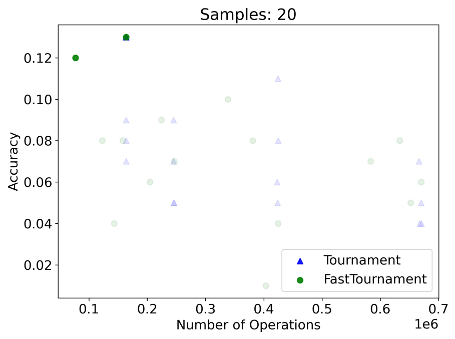

Figure 6. Grid search on Zipfian data with 20 samples.

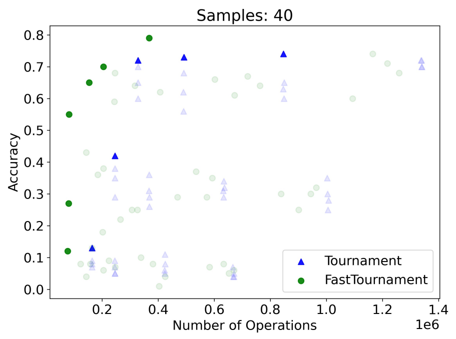

Figure 8. Grid search on Zipfian data with up to 40 samples.

5.2. Networking Experiments

Setup The underlying network data we use comes from the CAIDA Anonymized Internet Trace internet traffic dataset[4], which is IP traffic data collected at a backbone link of a Tier1 ISP in a data center in NYC. Within each minute, there are approximately $\approx 3.5 \cdot 10^7$ packets recorded.

We split 7 minutes of the IP data into 2,148 chunks, each representing ≈ 170ms and approximately 10^5 packets. For each chunk, we construct a distribution corresponding to the empirical distribution over source IP addresses in that chunk. The support sizes of the chunks (and therefore of the distributions) are approximately 45,000. The goal of approximate nearest neighbor search within this context is to identify similar traffic patterns to the current chunk of data within past data. Further, algorithms for density estimation allow this computation to be done on subsampled traffic data, which is a common practice in networking to make data acquisition feasible (cis).

For a series of 100 query chunks, we use the prior 2,048 chunks as the set of distributions to search over. We test the algorithms in two parameter regimes: 100 samples and $\mathit{fastConst} = 10$ as well as 250 samples and $\mathit{fastConst} = 25$. In both regimes, we test with $\mathit{nAllPairs} = 0$ and $\mathit{nAllPairs} = 5$. To measure performance, we report the total variation distance of the distribution returned by the tournament algorithms as well as the true nearest neighbor distance and average distance across all 2,048 distributions. Results are averaged over 10 trials for each of the 100 queries and one standard deviation is shaded.

Results In Figures 9 and 10, we plot the performance of the base and fast tournaments on the networking data across samples 100 and 250 and for *nAllPairs* set to 0 and 5, respectively. In all parameter settings, across the 100 queries, the fast tournament returns a distribution within roughly the same distance as the base tournament. The fast tournament uses $5\times$ (when $\mathit{nAllPairs} = 0$) or $2.4\times$ (when $\mathit{nAllPairs} = 5$) less number of operations. Interestingly, neither algorithm performs better with the inclusion of an all-pairs comparison at the end of the tournament. Even with the relatively small sampling rate compared to a domain of all possible IPs and support size of 45k, the tournament algorithms are able to recover distributions close to the nearest neighbor distance. Comparing the results with 100 and 250 samples, the distance to the distribution returned

[4]From CAIDA internet traces 2019, https://www.caida.org/catalog/datasets/monitors/passive-equinix-nyc/

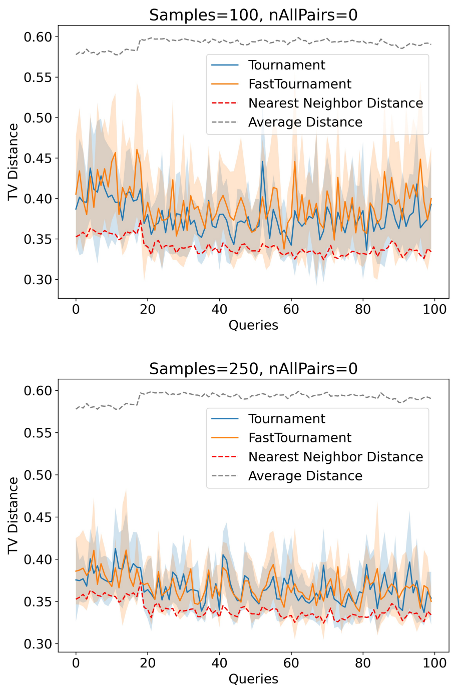

Figure 9. Networking experiments with *nAllPairs* $= 0$. The base tournament uses $5\times$ more operations than the fast tournament.

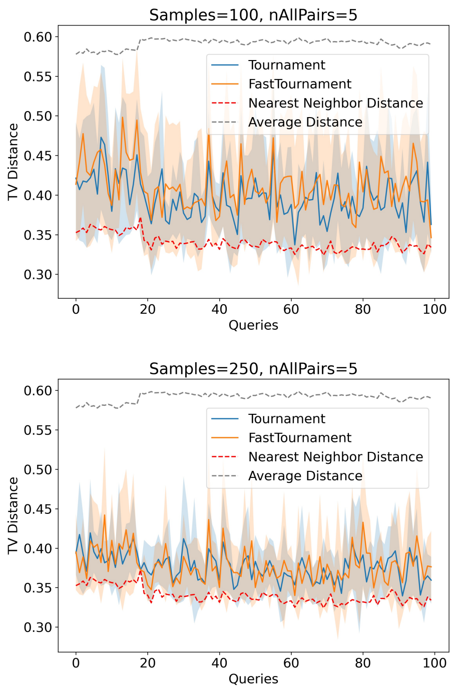

Figure 10. Networking experiments with *nAllPairs* $= 5$. The base tournament uses $2.4\times$ more operations than the fast tournament.

by the algorithms moderately improves and the variance reduces considerably with more samples.

Experiment summary In both the synthetic and real-world experiments, the tournament algorithms perform well in recovering close distributions using few samples. The most important factor in the algorithms' performance is the number of total samples. On the other hand, the fast tournament is able to use very limited sample sizes for earlier rounds of the tournament in order to save up to **5x** on computational cost while retaining essentially the same performance to the base tournament which uses all samples at all steps. According to the theoretical results, this computational gap will only increase for larger k.

6. Conclusion

We introduce the question of sublinear time density estimation, a natural generalization of nearest neighbor search to discrete distributions. We obtain the first algorithm with sublinear sample complexity and query time and with polynomial preprocessing, in the proper case. In the improper case, we improve upon prior results of (Acharya et al., 2018) to obtain a linear time algorithm with optimal sample complexity. Our work raises a number of interesting follow up questions: To what extent can our upper bounds of query and sample complexity be improved? What are the computational-statistical tradeoffs between the sample complexity and query time? While we studied discrete distributions under total variation distance, it is also interesting to ask if we can obtain efficient retrieval algorithms for other distances for distributions, discrete or continuous.

Acknowledgements

Anders Aamand is supported by DFF-International Postdoc Grant 0164-00022B from the Independent Research Fund Denmark. Research supported in part by NSF (CCF-2008733), and ONR (N00014-22-1-2713) grants. Justin Y. Chen is supported by MathWorks Engineering Fellowship, GIST-MIT Research Collaboration grant, NSF award CCF-2006798, and Simons Investigator Award. Justin Y. Chen, Shyam Narayanan, and Sandeep Silwal are supported by an NSF Graduate Research Fellowship under Grant No. 1745302. Piotr Indyk was supported by the NSF TRIPODS program (award DMS-2022448), Simons Investigator Award and MIT-IBM Watson AI Lab. Shyam Narayanan is supported by a Google PhD fellowship.

References

Using netflow sampling to select the network traffic to track. URL `https://www.cisco.com/c/en/us/td/docs/`

ios-xml/ios/netflow/configuration/xe-16/test/nf-xe-16-book/nflow-filt-samp-traff-xe.html.

Acharya, J., Falahatgar, M., Jafarpour, A., Orlitsky, A., and Suresh, A. T. Maximum selection and sorting with adversarial comparators. *The Journal of Machine Learning Research*, 19(1):2427–2457, 2018.

Andoni, A. and Razenshteyn, I. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 793–801, 2015.

Andoni, A., Indyk, P., and Razenshteyn, I. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pp. 3287–3318. World Scientific, 2018.

Bun, M., Kamath, G., Steinke, T., and Wu, S. Z. Private hypothesis selection. *Advances in Neural Information Processing Systems*, 32, 2019.

Canonne, C. L. A survey on distribution testing: Your data is big. but is it blue? *Theory of Computing*, pp. 1–100, 2020.

Canonne, C. L., Kamath, G., McMillan, A., Smith, A., and Ullman, J. The structure of optimal private tests for simple hypotheses. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 310–321, 2019.

Daskalakis, C. and Kamath, G. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Conference on Learning Theory*, pp. 1183–1213. PMLR, 2014.

Devroye, L. and Lugosi, G. *Combinatorial methods in density estimation*. Springer series in statistics. Springer, 2001. ISBN 978-0-387-95117-1.

Diakonikolas, I. Learning structured distributions. *Handbook of Big Data*, 267:10–1201, 2016.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Moitra, A., and Stewart, A. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019.

Gopi, S., Kamath, G., Kulkarni, J., Nikolov, A., Wu, Z. S., and Zhang, H. Locally private hypothesis selection. In *Conference on Learning Theory*, pp. 1785–1816. PMLR, 2020.

Har-Peled, S., Indyk, P., and Motwani, R. Approximate nearest neighbor: Towards removing the curse of dimensionality. 2012.

Indyk, P. On approximate nearest neighbors under $\ell\infty$ norm. *Journal of Computer and System Sciences*, 63(4): 627–638, 2001.

Kamath, G., Singhal, V., and Ullman, J. Private mean estimation of heavy-tailed distributions. In *Conference on Learning Theory*, pp. 2204–2235. PMLR, 2020.

Kamath, S., Orlitsky, A., Pichapati, D., and Suresh, A. T. On learning distributions from their samples. In *Conference on Learning Theory*, pp. 1066–1100. PMLR, 2015.

Scheffe, H. A Useful Convergence Theorem for Probability Distributions. *The Annals of Mathematical Statistics*, 18(3):434–438, 1947. doi: 10.1214/aoms/1177730390. URL https://doi.org/10.1214/aoms/1177730390.

Suresh, A. T., Orlitsky, A., Acharya, J., and Jafarpour, A. Near-optimal-sample estimators for spherical gaussian mixtures. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Szpankowski, W. *Average case analysis of algorithms on sequences*. John Wiley & Sons, 2011.

Valiant, G. and Valiant, P. The power of linear estimators. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 403–412. IEEE, 2011.

Yatracos, Y. G. Rates of convergence of minimum distance estimators and kolmogorov's entropy. *The Annals of Statistics*, 13(2):768–774, 1985.

A. Omitted Proofs of Section 2

Lemma 2.1. *For any $k \le \binom{n}{n/2}$, there exist distinct distributions $p, q_1, \dots, q_k$ over $[n]$ such that $\|p - q_i\|_1 = 1$ for each $i \in [k]$ and if $\hat{p}$ is the empirical distribution obtained from sampling $s = n/2$ elements from p, then the probability that $\|p - \hat{p}\|_1 \le \min_{i\in[k]}(\|q_i - \hat{p}\|_1)$ is $\exp(-\Omega(k))$. In fact, if $k \ge Cn \log n$ for a sufficiently large constant C, the probability is 0.*

Proof. Let $p = (p(1), \dots, p(n))$ where each $p(i) = 1/n$. Let $x(i) \sim \mathrm{Bin}(s, 1/n)$ denote the number of times item i is sampled, so that $\hat{p}(i) = x(i)/s$. Fix a subset $A \subseteq [n]$ with $|A| \le n/2$ and define $q_A = (q_A(1), \dots, q_A(n))$ by $q_A(i) = 2/n$ if $i \in A$ and $q_A(i) = 0$ otherwise. Note that $\|p - q_A\|_1 = 1$. Let $\ell_1 = |\{i \in A \mid x(i) \ge 1\}|$ and $\ell_2 = |\{i \in [n] \setminus A \mid x(i) \ge 1\}|$. If $s \le n/2$, then for all i such that $x(i) \ge 1$ it holds that

$$|p(i) - \hat{p}(i)| = \begin{cases} x(i)/s - p(i), & x(i) \ge 1 \\ p(i), & x(i) = 0, \end{cases}$$

and a similar equation holds for $|q_A(i) - \hat{p}(i)|$. Using this, simple calculations show that $\|\hat{p} - p\|_1 = 2 - \frac{\ell_1+\ell_2}{n}$ and $\|\hat{p} - q_A\|_1 = 2 - \frac{4\ell_1}{n}$. In particular, $\|\hat{p} - q_A\|_1 - \|\hat{p} - p\|_1 = \frac{2}{n}(\ell_1 - \ell_2)$. By symmetry, if A is sampled at random (and in particular A and A^c is sampled with the same probability), the events $\|\hat{p} - q_A\|_1 > \|\hat{p} - p\|_1$ and $\|\hat{p} - q_A\|_1 < \|\hat{p} - p\|_1$ occur with exactly the same probability. Moreover, conditioning on the $x(i)$'s, it is simple to check that regardless of their values, $\ell_1 \ne \ell_2$ with probability $\Omega(1)$. It follows that the probability that $\|\hat{p} - p\|_1 \le \|\hat{p} - q_A\|_1$ is at most $1 - c$ for some constant $c > 0$. Now pick the distributions $q_1, \dots, q_{2k}$ independently and uniformly at random by picking random $A_1, \dots, A_{2k}$ with $|A_i| = n/2$ and defining $q_i = q_{A_i}$ (note that we may have repetitions). By independence, the probability that $\|\hat{p} - p\|_1 \le \min \|\hat{p} - q_i\|_1$ is $\exp(-\Omega(k))$. If $k = O(n \log n)$, it moreover holds with the same high probability that $\{q_1, \dots, q_{2k}\}$ contains at least $2k$ different distributions and so the result follows. If on the other hand $k \ge Cn \log n$ for C sufficiently large, then we can pick $k' = Cn \log n$ distributions $q_1, \dots, q_{k'}$ satisfying the statement of the theorem. The error probability is $P_0 = \exp(-\Omega(k'))$ but there are at most $n^{n/2}$ ways to do the sampling from p. Therefore, if for at least one way of sampling from p it was the case that $\|\hat{p} - p\|_1 \le \min_{i\in[k']} \|\hat{p} - q_i\|_1$, then this event would in fact occur with probability at least $n^{-n/2}$. This is a contradiction since $P_0 = \exp(-\Omega(k')) < n^{-n/2}$ when C is sufficiently large. ☐

Lemma 2.2. *There exists distributions p, q over $[n]$ with $\|p - q\|_1 = 1$ such that for any sample size $s = o(n)$, with probability $\Omega(1)$, $\|p - \hat{p}\|_2 > \|q - \hat{p}\|_2$.*

Proof (sketch). Suppose for simplicity that $n = 2n_0 + 1$ is odd. Let $p = (p(1), \dots, p(n))$ be given by

$$p(i) = \begin{cases} \frac{1}{2}, & i = 1, \\ \frac{1}{2n_0}, & i = 2, \dots, n_0 + 1, \\ 0, & i = n_0 + 2, \dots, 2n_0 + 1, \end{cases}$$

Simple concentration bounds show that with high probability, $\sum_{i=2}^{n}(X(i) - sp(i))^2 \le (1 + o(1))s/2$. Define next $q = (q(1), \dots, q(n))$ by

$$q(i) = \begin{cases} \frac{1}{2} + \frac{1}{\sqrt{s}}, & i = 1, \\ 0, & i = 2, \dots, n_0 + 1, \\ \frac{\frac{1}{2} - \frac{1}{\sqrt{s}}}{n_0}, & i = n_0 + 2, \dots, 2n_0 + 1, \end{cases}$$

Again with the assumption that $s = o(n)$, standard concentration bounds show that $\sum_{i=2}^{n}(X(i) - sq(i))^2 \le (1 + o(1))s/2$ with high probability. It follows from these observations that if $X(1) \ge s/2 + \sqrt{s}$, then $\|p - \hat{p}\|_2 > \|q - \hat{p}\|_2$. By standard properties of the binomial distribution $B(s, 1/2)$, this happens with probability $\Omega(1)$. ☐

B. Omitted Proofs of Section 3.1

Remark B.1. We remark that in the proofs of this and all subsequent sections, the notation $1/\mathrm{poly}(n)$ or $1/\mathrm{poly}(k)$ refers to quantities of the form $1/n^C$ or $1/k^C$ where we can choose C to be an arbitrarily large constant.

The following is a standard tail bound for Poisson distributions[5].

Lemma B.2. *If $Y \sim Poi(\lambda)$, then for any $t > 0$, $\mathbb{P}(|Y - \lambda| \geq t) \leq 2\exp\left(-\frac{t^2}{2(\lambda+t)}\right)$.*

Lemma B.3 below bounds the ℓ_∞ distance between $\hat{p}$, the empirical distribution, and p, the unknown distribution.

Lemma B.3. *Let $\hat{p}$ denote the empirical distribution of Algorithm 3. With probability at least $1 - 1/poly(n)$, we have that for all $i \in [n]$,*

$$|\hat{p}(i) - p(i)| \leq O\left(\max\left(\sqrt{\frac{p(i)\log n}{s}}, \frac{\log n}{s}\right)\right).$$

Proof. Note that $s \cdot \hat{p}(i)$ is distributed as $\text{Poi}(s \cdot p(i))$. By setting $t = C' \cdot \max\left(\sqrt{s \cdot p(i)\log n}, \log n\right)$ in Lemma B.2 for a large enough constant C', we see that the probability $s \cdot \hat{p}(i)$ deviates by t is at most $1/\text{poly}(n)$, where we can make the degree of the polynomial arbitrarily large by increasing C'. The proof is completed by dividing by s. □

Lemma 3.2. *Suppose $\gamma = 1/n^C$ in Algorithm 1. Consider the sets $S_j = \{w_1, \dots, w_t\}$ defined in line 9 of Algorithm 2. Consider the vectors $(w_i)_H$, which are the heavy subsets of the distributions in S_j, as defined in line 10 of Algorithm 2. Then for all j and for all $w, w' \in S_j$, we have*
$\|w_H - w'_H\|_1 \leq O\left(\frac{(\log n)\log\log n}{n^{1/2-C}}\right)$.

Proof. There can be at most $1/\gamma = n^C$ heavy elements since each heavy element has probability mass at least γ, by Algorithm 1. Let H be the set of heavy elements. We know that any two distributions w and $w' \in S_j$ are at most $O((\log n) \cdot (\log\log n))/\sqrt{n}$ apart in ℓ_∞. Thus, the ℓ_1 difference restricted to the heavy elements is at most $n^C \cdot O\left(\frac{(\log n)\cdot(\log\log n)}{\sqrt{n}}\right) = O\left(\frac{(\log n)\log\log n}{n^{1/2-C}}\right)$. □

Lemma 3.3. *Suppose $s \geq \Omega(n/(\log n)^{1/2})$. Consider the output v^∞ on $\hat{p}$ when inputted into the ℓ_∞ data structure D^∞, as done in line 6 of Algorithm 3. Let S_∞ denote the group of v^∞ from Algorithm 2. Assuming the event in Lemma B.3 holds, it must be that $v^* \in S_\infty$.*

Proof. We know $\|p - v^*\|_\infty \leq \|p - v^*\|_2 \leq \frac{1}{2\sqrt{n}}$ by our assumption on v^* given in Section 1.1. Furthermore, Lemma B.3 implies that $\|p - \hat{p}\|_\infty \leq O\left(\frac{(\log n)^{3/4}}{\sqrt{n}}\right)$ so by adjusting constants, it follows from the triangle inequality that $\|\hat{p} - v^*\|_\infty \leq O\left(\frac{(\log n)^{3/4}}{\sqrt{n}}\right)$. By definition of v^∞, the distribution returned by the ℓ_∞ data structure in line 6 of Algorithm 3, we know that $\|\hat{p} - v^\infty\|_\infty \leq c\|\hat{p} - v^*\|_\infty$, where $c = O(\log(n) \cdot \log\log n)$ set in line 6 of Algorithm 3. Hence,

$$\begin{aligned}\|v^\infty - v^*\|_\infty &\leq \|\hat{p} - v^\infty\|_\infty + \|\hat{p} - v^*\|_\infty \\ &\leq (c+1)\|\hat{p} - v^*\|_\infty \\ &\leq O\left(\frac{(\log n)^{1.75} \cdot (\log\log n)}{\sqrt{n}}\right),\end{aligned}$$

so v^* must be in S_∞ from line 6 of Algorithm 2. □

Lemma 3.4. *Consider the same setting as in Lemma 3.3 and suppose $\gamma = 1/n^C$. Let H denote the heavy domain elements of group S_∞. Then for all $w \in S_\infty$, we have $\|p - w\|_\infty \leq O\left(\frac{(\log n)^2\log\log n}{\sqrt{n}}\right)$. Consequently, we also have $\|p_H - w_H\|_1 \leq O\left(\frac{(\log n)^2\log\log n}{n^{1/2-C}}\right)$.*

Proof. Take any $w \in S_\infty$. We know that

$$\begin{aligned}\|\hat{p} - w\|_\infty &\leq \|\hat{p} - v^\infty\|_\infty + \|v^\infty - w\|_\infty \\ &\leq \|\hat{p} - v^\infty\|_\infty + O\left(\frac{(\log n)^2 \cdot (\log\log n)}{\sqrt{n}}\right),\end{aligned}$$

[5] see e.g. https://math.stackexchange.com/questions/2434883/chernoff-style-bounds-for-poisson-distribution 2434922

where the last inequality holds from the construction of S_∞ in Algorithm 2. Since $\|\hat{p}-v^\infty\|_\infty \le c\|\hat{p}-v^*\|_\infty$, for c defined in line 6 of Algorithm 3, by definition,

$$\begin{aligned}
&\|p-w\|_\infty \le \|\hat{p}-w\|_\infty + \|p-\hat{p}\|_\infty \\
&\le c\|\hat{p}-v^*\|_\infty + \|p-\hat{p}\|_\infty + O\left(\frac{(\log n)\cdot(\log\log n)}{\sqrt{n}}\right).
\end{aligned}$$

The proof of Lemma 3.3 implies that $\|p-\hat{p}\|_\infty, \|\hat{p}-v^*\|_\infty \le O((\log n)^{3/4}/\sqrt{n})$, so by adjusting constant factors, the first inequality in the lemma statement holds, as $O\left((\log n)^2 \log\log n/\sqrt{n}\right)$ is the dominant term.

The second inequality in the lemma statement holds by an identical reasoning as used in the proof of Lemma 3.2, as there are at most n^C heavy coordinates. □

C. Omitted Proofs of Section 3.2

Recall that $T = \sum_{i\in L} p(i)$.

Lemma 3.5. $\mathbb{E}[\|s\cdot\hat{p}'_L - s\cdot v_L\|_2^2] = s\cdot T + s^2\|p_L - v_L\|_2^2$.

Proof. Note that the light elements refer to elements in L defined in Algorithm 3. Set $X(i) := s\cdot\hat{p}'_L(i)$ and recall that $X(i) \sim \text{Pois}(s\cdot p_L(i))$. We have

$$\begin{aligned}
&\mathbb{E}[\|s\cdot\hat{p}'_L - s\cdot v_L\|_2^2] \\
&= \sum_{i=1}^n \mathbb{E}[X(i)^2 + s^2 v_L(i)^2 - 2sX(i)v_L(i)] \\
&= \sum_{i=1}^n sp_L(i) + \sum_{i=1}^n \left(s^2 p_L(i)^2 - 2s^2 p_L(i)v_L(i) + s^2 v_L(i)^2\right) \\
&= sT + s^2\|p_L - v_L\|_2^2.
\end{aligned}$$ □

We now bound the variance. The following lemma bounds the variance one term at a time.

Lemma C.1. *Let* $X(i) = s\cdot\hat{p}'_L(i)$. *We have* $\text{Var}[(X(i)-sv_L(i))^2] \le 4sp_L(i)\cdot(sp_L(i)-sv_L(i))^2 + 6(sp_L(i))^2 + sp_L(i)$.

Proof. Define the auxiliary variables $y := sp_L(i)$ and $z := sv_L(i)$. We have

$$\begin{aligned}
\text{Var}[(X(i)-sv_L(i))^2] &= \mathbb{E}[(X(i)-sv_L(i))^4] - \mathbb{E}[(X(i)-sv_L(i))^2]^2 \\
&= \mathbb{E}[(X(i)-sv_L(i))^4] - (s^2p_L(i)^2 - 2s^2p_L(i)v_L(i) + s^2v_L(i)^2 + sp_L(i))^2 \\
&= \mathbb{E}[(X(i)-sv_L(i))^4] - (y + y^2 - 2yz + z^2)^2.
\end{aligned}$$

Since $X(i)$ is a Poisson random variable with parameter y, we know that

$$\begin{aligned}
\mathbb{E}[X(i)^3] &= y + 3y^2 + y^3, \\
\mathbb{E}[X(i)^4] &= y + 7y^2 + 6y^3 + y^4.
\end{aligned}$$

By expanding, we have

$$\begin{aligned}
\mathbb{E}[(X(i)-sv(i))^4] &= \mathbb{E}[X(i)^4] - 4\mathbb{E}[X(i)^3]z + 6\mathbb{E}[X(i)^2]z^2 - 4\mathbb{E}[X(i)]z^3 + z^4 \\
&= (y + 7y^2 + 6y^3 + y^4) - 4(y + 3y^2 + y^3)z + 6(y + y^2)z^2 - 4yz^3 + z^4.
\end{aligned}$$

Putting everything together gives us

$$\begin{aligned}
&\mathrm{Var}[(X(i) - sv_L(i))^2] = \mathbb{E}[(X(i) - sv_L(i))^4] - (y + y^2 - 2yz + z^2)^2 \\
&= \big((y + 7y^2 + 6y^3 + y^4) - 4(y + 3y^2 + y^3)z + 6(y + y^2)z^2 - 4yz^3 + z^4\big) - (y + y^2 - 2yz + z^2)^2 \\
&= 4y^3 - 8y^2 z + 4yz^2 + 6y^2 + y - 4yz \\
&= 4y(y - z)^2 + 6y^2 + y - 4yz \\
&\leq 4y(y - z)^2 + 6y^2 + y \\
&= 4sp_L(i) \cdot (sp_L(i) - sv_L(i))^2 + 6(sp_L(i))^2 + sp_L(i).
\end{aligned}$$
□

The following lemma bounds the total variance.

Lemma C.2. $\mathrm{Var}[\|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2] \leq 4s^3\|p_L\|_2\|p_L - v_L\|_2^2 + 6s^2\|p_L\|_2^2 + sT$.

Proof. Summing the result of Lemma C.1 for all coordinates i gives us

$$\begin{aligned}
&\mathrm{Var}[\|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2] \\
&\leq \sum_{i=1}^{n} 4sp_L(i) \cdot (sp_L(i) - sv_L(i))^2 + 6(sp_L(i))^2 + sp_L(i) \\
&= \sum_{i=1}^{n} 4s^3 p_L(i) \cdot (p_L(i) - v_L(i))^2 + 6s^2\|p_L\|_2^2 + sT \\
&\leq 4s^3\|p_L\|_2\|p_L - v_L\|_2^2 + 6s^2\|p_L\|_2^2 + sT.
\end{aligned}$$
□

Lemma 3.6. *Suppose p satisfies $\|p_L - v_L\|_2 \leq \frac{\varepsilon}{2\sqrt{n}}$. Set $t = \frac{s^2\|p_L - v_L\|_2^2}{4} + \frac{s^2\varepsilon^2}{4n}$ and $Z_1 = \|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2$. If $s = \Omega\left(\max\left(\frac{n\|p_L\|_2}{\varepsilon^2}, \frac{n^{2/3}}{\varepsilon^{4/3}}\right)\right)$ then $\mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \geq t] \leq 0.01$.*

Proof. By Chebyshev's inequality and Lemma C.2,

$$\begin{aligned}
\mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \geq t] &\leq \frac{\mathrm{Var}[Z_1]}{t^2} \\
&\lesssim \frac{s^3\|p_L\|_2\|p_L - v_L\|_2^2}{s^4\varepsilon^4/n^2} + \frac{s^2\|p_L\|_2^2}{s^4\varepsilon^4/n^2} + \frac{s}{s^4\varepsilon^4/n^2} \\
&\leq \frac{n\|p_L\|_2}{s\varepsilon^2} + \left(\frac{n\|p_L\|_2}{s\varepsilon^2}\right)^2 + \frac{n^2}{s^3\varepsilon^4} \leq 0.01
\end{aligned}$$

where the last inequality holds if $s \geq C\max\left(\frac{n\|p_L\|_2}{\varepsilon^2}, \frac{n^{2/3}}{\varepsilon^{4/3}}\right)$ for a sufficiently large constant C. □

We then consider the case where $\varepsilon/\sqrt{n} \lesssim \|p_L - v_L\|_2$. As stated in the main body, we need to obtain a stronger concentration result since later we union bound over possibly $\Omega(k)$ different distributions which are in group S_∞.

To obtain a stronger concentration result, we need the coordinates of both $\hat{p}_L$ and v_L to be bounded. For each v_i, we know that $\|(v_i)_L\|_\infty \leq 2\gamma$ by construction if we set $\gamma \geq \tilde{\Omega}(1/\sqrt{n})$. Lemma B.3 readily implies a similar statement for $\hat{p}_L$ (with high probability).

Remark C.3. Therefore, in the following concentration bound, we utilize the fact that $\max_i\{p_L(i), v_L(i)\} \leq O(\gamma)$, which holds with high probability.

The main tool we use is Bernstein's concentration inequality on $Z_1 = \|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2$. Towards this, we first prove a bound on a variance like quantity.

Lemma C.4. *With probability $1 - 1/\mathrm{poly}(k)$ (over the randomness in $\hat{p}'$), we have*

$$\max_{i \in [n]} |(s \cdot \hat{p}'_L(i) - s \cdot v_L(i))^2 - \mathbb{E}[(s \cdot \hat{p}'_L(i) - s \cdot v_L(i))^2]| \leq O\left(\log k \cdot (s\gamma)^{1.5}\right).$$

Proof. Define $X(i) = s \cdot \hat{p}'_L(i)$. Recall each $X(i)$ is distributed as $\text{Pois}(s \cdot p_L(i))$. From Lemma B.2, we have that with probability $1 - 1/\text{poly}(k)$, $|X(i) - sp_L(i)| \le O\left(\max\left(\sqrt{sp_L(i)\log k}, \log k\right)\right)$ for *all* i. Call this event $\mathcal{E}$. Now

$$\begin{aligned}
&(X(i) - s \cdot v_L(i))^2 - \mathbb{E}[(X(i) - s \cdot v_L(i))^2] \\
&= X(i)^2 - 2sv_L(i)X(i) + s^2 v_L(i)^2 - (s^2 p_L(i)^2 - 2s^2 p_L(i)v_L(i) + s^2 v_L(i)^2 + sp_L(i)) \\
&= X(i)^2 - s^2 p_L(i)^2 - sp_L(i) + 2sv_L(i)(sp_L(i) - X(i)).
\end{aligned}$$

Thus conditioning on $\mathcal{E}$ and recalling Remark C.3, we have

$$|X(i)^2 - s^2 p_L(i)^2 - sp_L(i)| \le O((sp_L(i))^{1.5}\log k) = O((s\gamma)^{1.5}\log k)$$

and

$$|2sv_L(i)(sp_L(i) - X_L(i))| \le O(s\gamma \cdot (\sqrt{s\gamma} + 1)\log k)$$

for all i. Altogether, we have that under $\mathcal{E}$,

$$\begin{aligned}
&\max_i |(X(i) - s \cdot v_L(i))^2 - \mathbb{E}[(X(i) - s \cdot v_L(i))^2]| \\
&\le O\left(\log k \cdot (s^{1.5}\gamma^{1.5})\right).
\end{aligned}$$

□

Lemma 3.7. *Suppose p satisfies $\|p_L - v_L\|_2 \ge 0.75 \cdot \frac{\varepsilon}{\sqrt{n}}$. Set $t = \frac{s^2\|p_L - v_L\|_2^2}{4} + \frac{s^2\varepsilon^2}{4n}$, suppose $\gamma = 1/n^{5/12}$, and let $Z_1 = \|s \cdot \hat{p}'_L - s \cdot v_L\|_2^2$ denote our estimator. If $s = \Omega\left(\frac{n^2\gamma^3\log(k)^2}{\varepsilon^4}\right)$ then $\mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \ge t] \le 1/\text{poly}(k)$.*

Proof. Define $X(i) = s \cdot \hat{p}'_L(i)$. Recall Bernstein's inequality: for a random variable $R = \sum_i R_i$ where R_i are independent, it states

$$\mathbb{P}(|R - \mathbb{E}[R]| \ge t) \le 2\exp\left(-\frac{t^2/2}{\text{Var}(R) + tM/3}\right) \tag{2}$$

where M is such that $|R_i - \mathbb{E}[R_i]| \le M$ with probability 1 for every i. In our case, $Z_1 = \sum_i (X(i) - s \cdot v_L(i))^2$, so we must first bound the maximum deviation of

$$|(X(i) - s \cdot v_L(i))^2 - \mathbb{E}[(X(i) - s \cdot v_L(i))^2]|$$

for all i. To do so, we condition on the event $\mathcal{E}$ of Lemma C.4. This gives

$$\mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \ge t] = \mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \ge t \mid \mathcal{E}] \cdot \mathbb{P}(\mathcal{E}) + \mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \ge t \mid \mathcal{E}^c] \cdot \mathbb{P}(\mathcal{E}^c).$$

We have $\mathbb{P}[|Z_1 - \mathbb{E}[Z]| \ge t \mid \mathcal{E}^c] \cdot \mathbb{P}(\mathcal{E}^c) \le \mathbb{P}(\mathcal{E}^c) \le \frac{1}{\text{poly}(k)}$ so it suffices to bound $\mathbb{P}[|Z_1 - \mathbb{E}[Z_1]| \ge t \mid \mathcal{E}]$. We drop the conditioning $\mathcal{E}$ for simplicity. Note that a similar analysis as above also shows that $|\mathbb{E}[Z_1] - \mathbb{E}[Z_1 \mid \mathcal{E}]| \le \frac{1}{\text{poly}(k)}$ which is a much smaller lower order term compared to t so we ignore it for simplicity. A similar statement holds for the variance of Z_1.

Now Bernstein's inequality gives us

$$\mathbb{P}(|Z_1 - \mathbb{E}[Z_1]| \ge t) \lesssim \exp\left(-\frac{t^2/2}{4s^3\|p_L\|_2\|p_L - v_L\|_2^2 + 6s^2\|p_L\|_2^2 + s + tM/3}\right)$$

where we have used Lemma C.2 to substitute in the variance. We can furthermore set $M = O\left(\log k \cdot (s\gamma)^{1.5}\right))$ from Lemma C.4. If we show that

$$\min\left(\frac{t^2}{s^3\|p_L\|_2\|p_L - v_L\|_2^2}, \frac{t^2}{s^2\|p_L\|_2^2}, \frac{t^2}{s}, \frac{t}{M}\right) \ge \Omega(\log k),$$

where the constants in the $\Omega(\log k)$ are sufficiently large, then it follows that

$$\frac{t^2/2}{4s^3\|p\|_2\|p_L - v_L\|_2^2 + 6s^2\|p_L\|_2^2 + s + tM/3} \ge \Omega(\log k)$$

and we get the desired concentration bound. Thus, we proceed to analyze each of the four fractions individually. Using $t = \frac{s^2\|p_L - v_L\|_2^2}{4} + \frac{s^2\varepsilon^2}{4n}$, we see that

$$\frac{t^2}{s^3\|p_L\|_2\|p_L - v_L\|_2^2} \ge \frac{s\|p_L - v_L\|_2^2}{\|p_L\|_2} \ge \frac{s\varepsilon^2}{n\|p_L\|_2},$$
$$\frac{t^2}{s^2\|p_L\|_2^2} \ge \frac{s^2\varepsilon^4}{n^2\|p_L\|_2^2},$$
$$\frac{t^2}{s} \ge \frac{s^3\varepsilon^4}{16n^2}.$$

Thus, $s \ge \Omega(\max(n^{2/3}\log(k)^{2/3}/\varepsilon^{2/3}, n\|p_L\|_2\log k/\varepsilon^2))$ is required. We now consider t/M Recalling their values, we observe that

$$\frac{t}{s^{1.5}\gamma^{1.5}} \ge \frac{s^{0.5}\varepsilon^2}{n\gamma^{1.5}}.$$

(Note that $s\gamma \gg 1$ in our setting). If $s \ge \Omega(n\|p\|_2\log k/\varepsilon^2)$, then $s\varepsilon^2/(n\gamma) = \Omega(\log k)$. To ensure $\frac{s^{0.5}\varepsilon^2}{n\gamma^{1.5}} \ge \Omega(\log k)$, we need to satisfy $s = \Omega\left(\frac{n^2\gamma^3\log(k)^2}{\varepsilon^4}\right)$. Recalling the value of γ, we can easily check that $s \ge \Omega\left(\frac{n^2\gamma^3\log(k)^2}{\varepsilon^4}\right)$ implies all other lower bounds we have imposed on s so far, proving the lemma. □

D. Proof of Theorem 3.1

Theorem 3.1. *Set $s = \Theta\left(\frac{n}{\varepsilon^2(\log k)^{1/4}}\right) = o(n)$ in Algorithm 3 and $\gamma = 1/n^{5/12}$ in Algorithm 2. Let $\tilde{v}$ denote the output of Algorithm 3. Then the preprocessing algorithm, Algorithm 2, runs in time polynomial in k, requires $\tilde{O}(nk^2)$ space, and the query time of Algorithm 3 is $\tilde{O}(n) + k^{1-1/(\log k)^{1/4}} = o(k)$. Furthermore, with probability $1 - o(1)$, Algorithm 3 returns distribution $\tilde{v}$ satisfying $\|p - \tilde{v}\|_1 \le \varepsilon$.*

Proof. Let v^∞ be the output of line 6 of Algorithm 3 and S_∞ be the group of v^∞ from Algorithm 2, the preprocessing step. Let H and L be the partition of $[n]$ into heavy and light elements induced by v^∞. From Lemma 3.4, we know that p is close to $\tilde{v}$ on H as $\tilde{v} \in S_\infty$: $\|p - \tilde{v}\|_1 \le o(1)$. Thus it suffices to bound $\|p_L - \tilde{v}_L\|_1$.

Lemma 3.3 states that $v^* \in S_\infty$ with probability $1 - 1/\text{poly}(n)$. Since we assumed that $\|p - v^*\|_2 \le \varepsilon/(2\sqrt{n})$, Lemma 3.6 implies that

$$\|s \cdot \hat{p}'_L - s \cdot v^*_L\|_2^2 \le sT + \frac{5s^2\varepsilon^2}{16n} \tag{3}$$

with probability $1 - o(1)$ (note Lemma 3.6 is stated as holding with probability 99% but we are picking a much larger value of s than required by the lemma. Plugging into the Chebyshev inequality bound readily gives us that the failure probability is $o(1)$).

Now we claim that $\tilde{v}$ *cannot* satisfy $\|\tilde{v}_L - p_L\|_2 \ge 0.99 \cdot \varepsilon/\sqrt{n}$. Let us assume for the sake of contradiction that $\|\tilde{v}_L - p_L\|_2 \ge 0.99 \cdot \varepsilon/\sqrt{n}$. Then from Lemma 3.5,

$$\mathbb{E}[\|s \cdot \hat{p}'_L - s \cdot \tilde{v}_L\|_2^2] = sT + s^2\|p_L - \tilde{v}_L\|_2^2,$$

and Lemma 3.7 implies that with probability at least $1 - 1/\text{poly}(k)$, we have

$$\|s \cdot \hat{p}'_L - s \cdot \tilde{v}_L\|_2^2 \ge sT + .75s^2\|p_L - \tilde{v}_L\|_2^2 - 0.25 \cdot \frac{s^2\varepsilon^2}{n} \ge sT + 0.48 \cdot \frac{s^2\varepsilon^2}{n}, \tag{4}$$

where the last inequality follows from the assumption that $\|p_L - \tilde{v}\|_2^2 \ge (.99)^2\varepsilon^2/n$. However, the ratio of the quantities on the right hand side of (4) and (3) is at least

$$\frac{sT + 0.48 \cdot \frac{s^2\varepsilon^2}{n}}{sT + \frac{5s^2\varepsilon^2}{16n}} \ge 1 + \frac{z/2}{T + z}$$

for $z = 5s\varepsilon^2/(16n)$. Now if $T \le z$ then $z/2/(T+z) \ge 1/4$ so the ratio is at least 1.25. Otherwise, if $T > z$, we always know that $T \le 1$ so $T + z \le 2$ and hence the ratio is at least $1 + \frac{z}{4} \ge 1 + \frac{5s\varepsilon^2}{64n}$. Since $(1+x)^{1/2} \ge 1 + x/2.5$ for any $x \in [0,1]$, we have that (in our regime of s) the *square root* of the ratio is strictly larger than $1 + s\varepsilon^2/(32n)$, by setting s appropriately. However, this contradicts the ℓ_2 nearest neighbor search guarantees set in line 10 of Algorithm 3, since it means we return a $\tilde{v}$ which has the property that $\|\hat{p}'_L - \tilde{v}_L\|_2$ is larger than $\|\hat{p}'_L - v^*_L\|_2$ by a factor larger than c set in line 10 (and v^* was also considered by the ℓ_2 data structure).

Therefore, we must have $\|\tilde{v}_L - p_L\|_2 \le 0.99 \cdot \varepsilon/\sqrt{n}$ which implies that $\|\tilde{v}_L - p_L\|_1 \le 0.99 \cdot \varepsilon$. Since we already know that $\|\tilde{v}_H - p_H\|_1 \le o(1)$, we have that $\|\tilde{v} - p\|_1 \le (1 + o(1))0.99 \cdot \varepsilon < \varepsilon$. This proves the approximation.

For the sample complexity, note that if we set $s = \Theta(n/(\log k)^{1/4})$ then s is larger than the values required in Lemmas 3.6 and 3.7. The prepossessing time follows from Theorem 1.1. Finally, the query time follows from 1.2 by plugging in the value of c from line 10 of Algorithm 3 (the ℓ_2 NNS approximation parameter) into statement of Theorem 1.2 and noting that all $\log(k)$ factors can be absorbed into the exponent as $k^{\Theta(1/(\log k)^{1/4})} \gg \text{poly}(\log k)$. Note that we get an extra $\tilde{O}(n)$ runtime from also querying an ℓ_∞ NNS datastructure as well in step 6 of Algorithm 3. Finally, we remark that an alternative expression for the approximation guarantee is $\|p - \tilde{v}\|_1 \le \|p - v^*\|_1 + \varepsilon$. This completes the proof. ☐

E. Omitted Proofs of Section 4

Theorem 4.1. *Assume that $\delta \ge k^{-1/4}$. With probability $1 - O(\delta)$, the algorithm of Section 4 outputs a distribution $\hat{v}$ with $\|p - \hat{v}\|_1 \le 27 \cdot \min_j \|p - v_j\|_1 + O(\varepsilon)$. The algorithm uses $s = O((\log k + \log \frac{1}{\delta})/\varepsilon^2)$ samples from p and has running time $O\left(\frac{k}{\varepsilon^2} \log \frac{1}{\delta}\right)$.*

Proof. The statement about the number of samples is clear. Indeed, for the iterative part of the algorithm, we use a total of $s_{\lg k} = \frac{10 \log(k^2/\delta)}{\varepsilon^2}$ samples and for the final part, we use a further $\frac{10 \log(|\mathcal{C}|^2/\delta)}{\varepsilon^2} \le \frac{10 \log(k^2/\delta)}{\varepsilon^2}$ samples. Regarding the running time, we already argued above that the time used for the knockout tournament is $O\left(\frac{k}{\varepsilon^2} \log \frac{1}{\delta}\right)$. Further, since $|\mathcal{C}| \le k^{1/3} \log k$, running the Scheffe test on all pairs of distributions in $\mathcal{C}$ takes time $O\left(sk^{2/3} \log^2 k\right) = O\left(\frac{k}{\varepsilon^2} \log \frac{1}{\delta}\right)$.

For the bound on the quality of the estimate $\hat{v}$, we define $v^* = \arg\min_{v \in \mathcal{V}} \|p - v\|_1$ and let $\mathcal{W}_0 = \{v \in \mathcal{V} : \|p - v\|_1 \le 3\|p - v^*\|_1 + \varepsilon\}$ and $\mathcal{W}_1 = \mathcal{V} \setminus \mathcal{W}_0$. We first want to argue that with probability $1 - O(\delta)$, some element of $\mathcal{W}_0$ gets added to $\mathcal{C}$. For $i = 1, \dots, \lg k$, we let T_i be the set of $2^i - 1$ distributions that v^* could possibly be paired with in step 2. of the algorithm during the first i rounds of the tournament (this is the set of $2^i - 1$ distributions in the subtree of the tournament tree rooted i steps above v^*). Let A_i denote the event that there exists a distribution v' in $T_i \cap \mathcal{W}_1$ such that v^* loses the Scheffe test to v' using the sample $\mathcal{S}_i$. By the choice of s_i and the bound in (1), we have that

$$\Pr[A_i] \le |T_i \cap \mathcal{W}_1| \delta_i \le \frac{\delta}{2^i},$$

so $\Pr[\bigcup_i A_i] \le \delta$. We now consider two cases: Suppose first that at some stage i, $\frac{|\mathcal{V}_i \cap \mathcal{W}_0|}{|\mathcal{V}_i|} \ge \frac{\log 1/\delta}{k^{1/3}}$. In this case, the probability that no element of $\mathcal{V}_i$ is added to $\mathcal{C}$ during step 1. of iteration i of the algorithm is at most $(1 - \frac{\log 1/\delta}{k^{1/3}})^{k^{1/3}} \le \delta$. If on the other hand, at each step i, $\frac{|\mathcal{V}_i \cap \mathcal{W}_0|}{|\mathcal{V}_i|} < \frac{\log 1/\delta}{k^{1/3}}$, then the probability that v^* gets paired with an element of $\mathcal{W}_0$ in step i (conditioned on it surviving the first $i-1$ steps) is at most $\frac{\log 1/\delta}{k^{1/3}}$. Note that if none of the events A_i occurs and v^* never gets paired with a distribution in $\mathcal{W}_0$, then v^* will win the tournament, and thus get added to $\mathcal{C}$ in some step. Thus, by a union bound, the probability that no element of $\mathcal{W}_0$ gets added to $\mathcal{C}$ is at most

$$\Pr\left[\bigcup_i A_i\right] + \delta + \frac{\log k \log 1/\delta}{k^{1/3}} \le 2\delta + \frac{(\log k)^2}{4k^{1/3}} = O(\delta),$$

where we used $\delta \ge k^{-1/4}$ in the final step.

It was shown in (Devroye & Lugosi, 2001) that running the Scheffe test on all pairs of distributions in a set $\mathcal{C}$ using a sample of size $\frac{10 \log\left(\binom{|\mathcal{C}|}{2}/\delta\right)}{\varepsilon^2}$ and outputting the one $\hat{v}$ with the most wins, we have with probability at least $1 - \delta$ that

$$\|p - \hat{v}\|_1 \le 9 \cdot \min_{v \in \mathcal{C}} \|p - v\|_1 + 4\varepsilon.$$

Since $\mathcal{C}$ contains a distribution of $\mathcal{W}_0$ with probability $1 - O(\delta)$ we obtain that with probability $1 - O(\delta)$,

$$\begin{aligned}\|p - \hat{v}\|_1 &\leq 9 \cdot \min_{v \in \mathcal{C}} \|p - v\|_1 + 4\varepsilon \\ &\leq 9 \cdot (3\|p - v^*\|_1 + \varepsilon) + 4\varepsilon = 27\|p - v^*\|_1 + 13\varepsilon,\end{aligned}$$

as desired. □

Remark E.1. We can also handle error probabilities $\delta < k^{-1/4}$. In this case, we set $\delta_0 = k^{-1/4}$ and run the iterative part of the algorithm above $\ell = \frac{\log 1/\delta}{\log 1/\delta_0}$ times to obtain a candidate set of distributions $\mathcal{C} \subseteq \mathcal{V}$ (ℓ times larger than before) which contains an element of $\mathcal{W}_0$ with probability $1 - O(\delta)$. We then again run the complete tournament on $\mathcal{C}$ with a sample of size $\frac{10 \log\left(\binom{|\mathcal{C}|}{2}/\delta\right)}{\varepsilon^2}$. The bound on the quality of the estimate, follows as above. Moreover, easy calculations show that the number of samples used is still $O(\frac{\log 1/\delta}{\varepsilon^2})$ and that the total running time is

$$O\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta} \left(k + \ell^2 k^{2/3} \log^2 k\right)\right),$$

which is $O\left(\frac{k}{\varepsilon^2} \log \frac{1}{\delta}\right)$ except for extremely small δ.

Remark E.2. In a slightly different model, we are not given the distributions of $\mathcal{V}$ explicitly but can only access them through sampling. In this model, we can skip the preprocessing step and instead estimate the probabilities $v_i(S)$ and $v_j(S)$ through sampling whenever we run the Scheffe test for distributions v_i and v_j. This comes at the cost of a larger value of the constant C. When running the algorithm above in this model, we obtain the same reduction of the running time and with no preprocessing.