Adaptive Mode Estimation of Continuous Distribution*

Anirudh Singhal [1], Subham Pirojiwala [2], Guide : Prof. Nikhil Karamchandani

***Abstract*— In this paper, we study the problem of finding the point with the nearest k^{th} neighbour, from a dataset of n points, in a high dimensional space. One of the applications involves finding the mode of the continuous distribution, from which the points in the dataset have been drawn. We present an algorithm that adaptively estimates the k^{th} neighbour distance for each point and uses a concentration bound based on Law of the iterated logarithm. We then demonstrate experimentally that the algorithm performs better than the naive method, and another method that is based on finding the set of k-NN as the first step.**

I. INTRODUCTION

The mode of a dataset is a widely used parameter for various applications in machine learning, signal processing, etc. In this paper we use the results of [1] to form a problem which can be used to adaptively estimate mode of a dataset drawn from continuous underlying distribution in various regimes. First, we formally define our problem and the regimes in which we study our problem. Next, we propose an algorithm (Algorithm 1) which can be used to adaptively estimate the mode of a dataset drawn from a continuous underlying distribution. Finally we provide a comprehensive ablation study to understand various parameters involved in the algorithm.

II. RELATED WORK

The exact problem we propose has not been studied extensively but some related problems have been studied. [2] studies the problem of finding the approximate k-Nearest Neighbour (k-NN) set, which is finding a set of O(k) points that contains the set of k-NN. One specific case of which is finding the set of exact k-NN, [3] deals with the problem of finding the nearest neighbour graph from noisy distance samples and [4] discusses algorithms to find out best-K arms from N stochastic bandit arms with unknown reward distribution.

As a direct application, our algorithm can be used to estimate the mode of a continuous distribution, from which the points in the dataset have been drawn. In this domain, [1] presents practical mode estimators that is based on finding the k-NN, which estimates mode optimally, under general distributions.

[5] studies the problem of estimating the mode of a discrete probability distribution where they query an oracle to get i.i.d. samples generated from that distribution.

However, the problem of adaptively estimating the mode of a dataset whose points are taken from an underlying continuous distribution has not been studied in our knowledge.

III. PROBLEM FORMULATION

Let $\mathcal{X}$ denote a set of n points represented as follows.

$$\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$$

The points in $\mathcal{X}$ are normalized so that $\|\mathbf{x}_i\|_\infty \leq 1/2$ for all the possible values of i. Let $\mathbf{x}_{i_k}$ denote the k^{th} nearest neighbour of the point $\mathbf{x}_i$ in the set $\mathcal{X}$. Then we wish to calculate the point $\mathbf{x}_{m_k}$ which is referred as *Estimated Mode using k^{th} Nearest Neighbour distance*. The value of m_k is given by the following equation.

$$m_k \triangleq \underset{i\in\{1,2,\cdots,n\}}{\operatorname{argmin}}\ d_{i,i_k} \tag{1}$$

Where $d_{i,j}$ is the distance function between the points $\mathbf{x}_i$ and $\mathbf{x}_j$. In this work triangular inequality and symmetry properties for the distance function

* Work done as a part of RnD project in the Spring Semester'20

[1]Anirudh is a Dual Degree student in Electrical Engineering Department, **Roll Number**: 16D070032, **Email**: singhalanirudh18@gmail.com

[2]Subham is a Dual Degree student in Electrical Engineering Department, **Roll Number**: 160040045, **Email**: pirojiwalasubham@gmail.com

are not assumed. The works of [1] proved that the above quantity correctly estimates the mode for unimodal continuous distribution for certain bounds on k.

We study this problem in the following two regimes :

Regime 1 : This is a high-dimensional regime where the value of m is extremely large. It was found out by [2] that in this regime it is unnecessary to compute exact distance to find a set that contains k nearest neighbours of a point. In this regime, the distance between two points is estimated by calculating distance along a randomly sampled dimension. This is done by sampling $T_{i,j}$ indices uniformly at random with replacement from $\{1, 2, \cdots, m\}$. Therefore, the estimate of the distance $d_{i,j}$ at the time instant $T_{i,j}$ is given by the following equation.

$$\widehat{d_{i,j}}\left(T_{i,j}\right) = \frac{1}{T_{i,j}} \sum_{p \in \left\{p_1, \ldots, p_{T_{i,j}}\right\}} \left|[\mathbf{x}_i]_p - [\mathbf{x}_j]_p\right|^2 \quad (2)$$

Where $[\mathbf{x}_i]_p$ is the p^{th} dimension of the point x_i. In this paper we use Python syntax, in which we store the distance estimates $\widehat{d_{i,j}}$ for $j \in [n]\backslash i$ in an array $\widehat{d_i}$ and the number of dimensions sampled $T_{i,j}$ for $j \in [n]\backslash i$ in an array T_i.

Regime 2 : In this regime the true distance between any two points is unknown, but we can query an oracle which returns the noisy estimate, $Q(i, j)$, of the true distance between any two points which is given as follows.

$$Q(i, j) = d_{i,j} + \eta \quad (3)$$

Where η is assumed to be distributed as zero mean sub Gaussian Random Variable whose scale parameter is $\sigma = 1$.

IV. Algorithm for Regime 1

We propose AdaptiveModeEstimation (Algorithm 1) which adaptively estimates the quantity x_{m_k} where m_k is given by the eq. 1 with a probability of $1 - \delta$. Our Algorithm is inspired from Upper Confidence Bound(UCB) algorithm [6], in which we continuously sample the point with the minimum Lower Confidence Bound (LCB) and stop when a certain stopping criteria is met.

To calculate the LCB and UCB for the k^{th} Nearest Neighbour of a point $x_i \in \mathcal{X}$ we utilise the subroutine AdaptiveKNN (Algorithm 2) which is a variant of the method proposed by [2]. This subroutine is based on actively estimating the set $\mathcal{X}_{near}$ which contains k nearest points of the point x_i. Once this set is found another subroutine, FindKthBest (Algorithm 3) is used to find k^{th} nearest neighbour of x_i from the set $\mathcal{X}_{near}$ almost surely. The algorithm 2 works by adaptively estimating the distance $d_{i,j} = \frac{1}{m}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2$. The arrays $\widehat{d_i}$ and T_i are provided as an input to the Algorithm 2.

The confidence bounds used are a non-asymptotic version of the law of iterated logarithm [7], which is given by the following equation.

$$\alpha(u) \propto \sqrt{\frac{\log(\log(u)n/\delta)}{u}} \quad (4)$$

Let UCB and LCB denote two arrays of size $n-1$ which contains the upper bounds and lower bounds for the distance estimates $\widehat{d_i}$ with T_i number of samples respectively. Let $(\cdot)$ be a permutation of $[n-1]$ such that $\widehat{d}[(1)] \leq \widehat{d}[(2)] \leq \ldots \widehat{d}[(n-1)]$, then at every round of Algorithm 2 two points q_1 and q_2 are sampled which are given by the following equations.

$$q_1 = \underset{i \in \{(1), \ldots, (k)\}}{\arg\max}\ UCB[i] \quad (5)$$

$$q_2 = \underset{i \in \{(k+1), \ldots, (n-1)\}}{\arg\min}\ LCB[i] \quad (6)$$

The distance $\widehat{d_{i,j}}$ is updated using the following function.

$$ED(\widehat{d_{i,j}}, T) = \begin{cases} \frac{T-1}{T}\widehat{d_{i,j}} + \frac{1}{T}\left|[\mathbf{x}_i]_p - [\mathbf{x}_j^i]_p\right|^2 & T < m \\ \frac{1}{m}\left\|\mathbf{x}_i - \mathbf{x}_j^i\right\|_2^2 & T = m \end{cases} \quad (7)$$

Where ED stands for estimated distance and p is chosen randomly from $\{1, 2, \cdots, m\}$. Once the set of k points closest to x_i are identified, Algorithm 2 invokes Algorithm 3 to find the k^{th} nearest neighbour. Let $(\cdot)$ and $\{\cdot\}$ be two permutations of $[n-1]$ such that $UCB[(1)] \leq UCB[(2)] \leq \ldots UCB[(n-1)]$ and $LCB[\{1\}] \leq LCB[\{2\}] \leq \ldots LCB[\{n-1\}]$, then Algorithm 2 calculates the

Algorithm 1 AdaptiveModeEstimation

```
**Input parameters :** $k$, $n$, $\delta$ and $num\_steps$
for $i$ in $range(n)$ do
    Initialize $\widehat{d}_i$ and $T_i$ as zero vectors
    $\widehat{d}_i, T_i, u, l \leftarrow AdaptiveKNN(\widehat{d}_i, T_i, x_i)$
    $U_k[i] \leftarrow u$, $L_k[i] \leftarrow l$
end for
Calculate $b_1$ and $b_2$ as per eq. 10 and 11 respectively.
while $U_k[b_1] > L_k[b_2]$ do
    $d, t, u, l \leftarrow AdaptiveKNN(\widehat{d}_{b_1}, T_{b_1}, b_1)$
    $\widehat{d}_{b_1} \leftarrow d, T_{b_1} \leftarrow t$
    $U_k[b_1] \leftarrow u$, $L_k[b_1] \leftarrow l$
    Update the values of $b_1$ and $b_2$
end while
$m_k \leftarrow b_1$
return $x_{m_k}$
```

confidence interval of the k^{th} nearest neighbour of x_i as follows.

$$UCB_k = UCB[(k)] \tag{8}$$

$$LCB_k = LCB[\{k\}] \tag{9}$$

Where UCB_k and LCB_k are the Upper Confidence Bound and Lower Confidence Bound of the k^{th} nearest neighbour of x_i respectively.

The AdapativeModeEstimation (Algorithm 1) is inspired from the UCB algorithm, in which we model each node as an arm and the distance of the point from it's k^{th} nearest neighbour is modelled as the reward of the arm. In this setting we wish to find the arm with the least reward. First, we find the UCB_k and LCB_k for all $i \in \{1, \cdots, n\}$ and store them in the arrays U_k and L_k respectively. $U_k[i]$ is the upper bound to the distance of x_i from its k^{th} neighbour. Similarly, $L_k[i]$ is the lower bound. At every round the quantities b_1 and b_2 are calculated as follows.

$$b_1 = \arg\min_{i \in \{1,\ldots,n\}} L_k[i] \tag{10}$$

$$b_2 = \arg\min_{i \in \{1,\ldots,n\} \backslash b_1} L_k[i] \tag{11}$$

The Algorithm 1 terminates when $U_k[b_1] < L_k[b_2]$, and outputs x_{b_1} as the result.

Algorithm 2 AdaptiveKNN

```
**Input parameters :** $\widehat{d}_i$, $T_i$ and $x_i$. The parameters $k$, $n$, $\delta$ & $num\_steps$ are inherited from AdaptiveModeEstimation(Algorithm 1). The definition of the function $ED(\cdot,\cdot)$ is as per eq. 7.
$\mathcal{X}_i \leftarrow \mathcal{X} \backslash \mathbf{x}_i$
Let the elements of $\mathcal{X}_i$ be denoted as $\{\mathbf{x}^i_1, \ldots, \mathbf{x}^i_{n-1}\}$
for $j$ in $range(n-1)$ do
    if $T_i[j] = 0$ then
        $T[j] \leftarrow 1$
        $\widehat{d}[j] \leftarrow ED(0, T[i])$
    else
        Initialize the variables as follows
        $T[j] \leftarrow T_i[j]$
        $\widehat{d}[j] \leftarrow \hat{d}_i[j]$
    end if
    $LCB[j] \leftarrow \widehat{d}[j] - \alpha(T[j])$
    $UCB[j] \leftarrow \widehat{d}[j] + \alpha(T[j])$
end for
for $step$ in $range(num\_steps)$ do
    Let $(\cdot)$ denote a permutation of $[n-1]$ such that
    $\widehat{d}[(1)] \leq \widehat{d}[(2)] \leq \ldots \widehat{d}[(n-1)]$
    Calculate $q_1$ and $q_2$ as per equations 5 and 6
    if $UCB[q_1] < LCB[q_2]$ then
        $\widehat{d}, T, LCB, UCB \leftarrow FindKthBest(\widehat{d}, T)$
    else
        for $\ell \in \{q_1, q_2\}$ do
            Increment $T[\ell] \leftarrow T[\ell]+1$ and sample an index p uniformly at random from [m], and update $d[\ell]$ as in 7.

            Update LCB and UCB as follows:-
            $LCB[\ell] \leftarrow \widehat{d}[\ell] - \alpha(\ell)$
            $UCB[\ell] \leftarrow \widehat{d}[\ell] + \alpha(\ell)$
        end for
    end if
end for
Let $(\cdot)$ and $\{\cdot\}$ denote two permutations of $[n-1]$ such that
$UCB[(1)] \leq UCB[(2)] \leq \ldots UCB[(n-1)]$
$LCB[\{1\}] \leq LCB[\{2\}] \leq \ldots LCB[\{n-1\}]$
$UCB_k \leftarrow UCB[(k)]$, $LCB_k \leftarrow LCB[\{k\}]$
return $\widehat{d}$, $T$, $UCB_k$ and $LCB_k$
```

Algorithm 3 FindKthBest

```
**Input parameters** : $\widehat{d}$ and $T$. The parameters $k$, $n$ & $\delta$ are inherited from AdaptiveKNN(Algorithm 2).
**for** $i$ in $range(n-1)$ **do**
    $LCB[i] \leftarrow \widehat{d}[i] - \alpha(T[i])$
    $UCB[i] \leftarrow \widehat{d}[i] + \alpha(T[i])$
**end for**
Let $(\cdot)$ and $\{\cdot\}$ denote two permutations of $[n-1]$ such that
$UCB[(1)] \leq UCB[(2)] \leq \ldots UCB[(n-1)]$
**if** $UCB[(k-1)] \geq LCB[(k)]$ **then**
    **for** $\ell \in \{(k),(k-1)\}$ **do**
        Increment $T[\ell] \leftarrow T[\ell]+1$ and sample an index p uniformly at random from [m], and update $d[\ell]$ as in 7.

        Update LCB and UCB as follows:-
        $LCB[\ell] \leftarrow \widehat{d}[\ell] - \alpha(\ell)$
        $UCB[\ell] \leftarrow \widehat{d}[\ell] + \alpha(\ell)$
    **end for**
**end if**
**return** $\widehat{d}$, $T$, $UCB$ and $LCB$
```

V. Correctness of the Algorithm 1

In this section prove that the point returned by the algorithm 1 is the one given by the eq. 1. First we introduce some notation. For a point $\mathbf{x}_i$, let $\mathcal{X}_i$ denote the set $\mathcal{X}\backslash\mathbf{x}_i$ whose points are represented by $\{\mathbf{x}_1^i, \ldots, \mathbf{x}_{n-1}^i\}$. The distance between between $\mathbf{x}_i$ and $\mathbf{x}_j^i$ is represented by d_j^i and let $\widehat{d^i}_j(T_j^i)$ denote the estimate of d_j^i after T_j^i samples. The following is derived from the non-asymptomatic version of Law of Iterated Logarithms ([7]) derived in Lemma 3 of [2].

Lemma 1: The following event occurs with probability $1-\delta$ *for* $\delta \in (0, 0.05)$ *:*

$$\mathcal{E}_\alpha := \{\left|\widehat{d}_j^i(T_j^i) - d_j^i\right| \leq \alpha_j^i, \quad \forall i \in [n], \forall j \in [n-1], \forall t \geq 1\} \tag{12}$$

Where the value of α_j^i *is given by* $\alpha_j^i = \alpha\left(T_j^i\right) = \sqrt{\frac{2\beta\left(T_j^i,\delta'\right)}{T_j^i}}$, $\delta' = \frac{\delta}{n\times(n-1)}$ *and* $\beta(u,\delta') = \log(1/\delta') + 3\log\log(1/\delta') + 1.5\log(1+\log(u))$.

It is important to note the **Lemma 1** holds when the value of $\widehat{d^i}_j(T_j^i)$ is sum of T_j^i independent random variables whose values are bounded between 0 and 1, and that expected value of $\widehat{d^i}_j(T_j^i)$ is equal to d_j^i. Both of these conditions hold in our setting by definition.

Let, $LCB_j^i = \widehat{d^i}_j(T_j^i) - \alpha_j^i$ and $UCB_j^i = \widehat{d^i}_j(T_j^i) + \alpha_j^i$. Then by **Lemma 1**, if the event $\mathcal{E}_\alpha$ occurs the following is true.

$$LCB_j^i \leq d_j^i \leq UCB_j^i, \forall i \in [n] \ \&\ \forall j \in [n-1] \tag{13}$$

Without loss of generality we can assume that $d_1^i \leq d_2^i \leq \ldots \leq d_{n-1}^i$. Therefore, we can say that under this assumption $d_k^i = d_{i,i_k}$, which is the distance between $\mathbf{x}_i$ and its k^{th} nearest neighbour.

Let $(\cdot)$ and $\{\cdot\}$ be two permutations of $[n-1]$ such that :

$$UCB_{(1)}^i \leq UCB_{(2)}^i \leq \ldots UCB_{(n-1)}^i \tag{14}$$

$$LCB_{\{1\}}^i \leq LCB_{\{2\}}^i \leq \ldots LCB_{\{n-1\}}^i \tag{15}$$

Theorem 1: If the event $\mathcal{E}_\alpha$ *occurs, then the distance of* k^{th} *nearest neighbour of* $\mathbf{x}_i$ *denoted by* d_k^i *(or* d_{i,i_k}*) is bounded as follows.*

$$LCB_{\{k\}}^i \leq d_k^i \leq UCB_{(k)}^i, \forall i \in [n], \forall k \in [n-1] \tag{16}$$

Where the definition of $(\cdot)$ *and* $\{\cdot\}$ *is as per eqs. 14 and 15 respectively.*

The proof of **Theorem 1** is provided in the Appendix A.

The values of b_1 and b_2 is given as follows in the new notation :

$$b_1 = \underset{i\in\{1,\ldots,n\}}{\arg\min}\ LCB_{\{k\}}^i$$

$$b_2 = \underset{i\in\{1,\ldots,n\}\backslash b_1}{\arg\min}\ LCB_{\{k\}}^i$$

We show that when the event $\mathcal{E}_\alpha$ occurs, the value returned by the algorithm (b_1) is same as m_k as defined in eq. 1. If the event $\mathcal{E}_\alpha$ occurs, when the algorithm terminates we have the following.

$$UCB_{(k)}^{b_1} < LCB_{\{k\}}^{b_2} \tag{17}$$

From **Theorem 1** we have the following two equations :

$$LCB_{\{k\}}^{b_2} \leq d_k^i, \forall i \in [n]\backslash b_1 \tag{18}$$

$$d_k^{b_1} \leq UCB_{(k)}^{b_1} \tag{19}$$

Plugging in the eqs. 18 and 19 in the eq. 17 we get the following result.

$$d_k^{b_1} \leq UCB_{(k)}^{b_1} < LCB_{\{k\}}^{i} \leq d_k^i, \forall i \in [n-1] \backslash b_1$$

$$\implies d_k^{b_1} < d_k^i, \forall i \in [n] \backslash b_1$$

Therefore, when the event $\mathcal{E}_\alpha$ occurs, we have $b_1 = m_k$. As a result, the algorithm 1 provides the corrects result with a probability $1-\delta$ (as the event $\mathcal{E}_\alpha$ occurs with a probability $1-\delta$ according to **Lemma 1**).

VI. Query Complexity

Theorem 2: For a dataset $\mathcal{X}$, our algorithm ran with parameter δ yields the estimated mode m_k has query complexity at most $N(\mathcal{X})$ with probability at least 1 - δ. The logarithmic terms in n and the double logarithmic terms are absorbed in $\widetilde{O}$

$$\mathcal{N}(\mathbf{x}_i, \mathcal{X}_i) = \widetilde{O}\left(\sum_{j=1}^{n-1} \min\left(\Delta_{i,j}^{-2}, m\right) + \sum_{j=1}^{k-1} \min\left(\Delta_{i,j}^{*-2}, m\right)\right) \tag{20}$$

$$\Delta_{i,j} = \begin{cases} d_{k+1}^i - d_j^i & j < k \\ 0 & j = k \\ d_j^i - d_k^i & j > k \end{cases} \tag{21}$$

$$\Delta_{i,j}^* = d_j^i - d_k^i \tag{22}$$

$$N(\mathcal{X}) = \sum_{i=1}^{n} \mathcal{N}(\mathbf{x}_i, \mathcal{X} \backslash \mathbf{x}_i) \tag{23}$$

The notation $\widetilde{O}$ absorbs factors logarithmic in n and doubly logarithmic in the gaps.

Proof :
To prove **Theorem 2** we make use of following results which have been proved by [2] and [8] respectively.

Lemma 2: For any point $\mathbf{x}_i$ in dataset $\mathcal{X}$, adaptive k-NN algorithm specified in [2] ran with parameter δ yields the set of k nearest neighbours and has query complexity at most $N_1(\mathbf{x}_i, \mathcal{X}_i)$ with probability at least 1 - δ as shown by [2].

$$N_1(\mathbf{x}_i, \mathcal{X}_i) = \widetilde{O}\left(\sum_{j=1}^{n-1} \min\left(\Delta_{i,j}^{-2}, m\right)\right) \tag{24}$$

The value of $\Delta_{i,j}$ is as per eq. 21

Lemma 3: For a point $\mathbf{x}_i$ and a set $\mathcal{X}_i^k$ of size k consisting k Nearest Neighbours of $\mathbf{x}_i$, UCB algorithm specified in [8] ran with parameter δ yields the k^{th} nearest neighbour of $\mathbf{x}_i$ and has query complexity at most $N_2(\mathbf{x}_i, \mathcal{X}_i^k)$ with probability at least 1 - δ.

$$N_2(\mathbf{x}_i, \mathcal{X}_i) = \widetilde{O}\left(\sum_{j=1}^{k-1} \min\left(\Delta_{i,j}^{*-2}, m\right)\right) \tag{25}$$

The value of $\Delta_{i,j}^*$ is as per eq. 22.

For a given point $\mathbf{x}_i$, finding the set of k nearest neighbours using adaptive k-NN algorithm proposed in [2] and then using UCB algorithm proposed in [8] yields the k^{th} nearest neighbour of $\mathbf{x}_i$ with probability at least 1 - δ. Repeating this for every $\mathbf{x}_i$ in $\mathbf{X}$ yields k^{th} nearest neighbour for every point, that can be used to estimate the mode.

In our algorithm, for each point $\mathbf{x}_i$, the sampling strategy followed in subroutine 2 is same as the strategy followed in adaptive k-NN algorithm proposed in [2] until the set of k nearest neighbours is found, followed by a sampling strategy proposed by the UCB algorithm in [8] until the k^{th} nearest neighbour is found.

Hence from **Lemma 2** and **Lemma 3**, for a point $\mathbf{x}_i$, number of queries is at most $N_1(\mathbf{x}_i, \mathcal{X})$ + $N_2(\mathbf{x}_i, \mathcal{X})$. Summing this over all points in $\mathcal{X}$ gives us the upper bound specified in **Theorem 2**.

VII. Experiments: Regime 1

A. Datasets

Two kinds of datasets were used, namely, Artificial dataset and Tiny ImageNet dataset.

- **Artificial dataset** : The artificial data is generated via $\mathbf{x} = \mathbf{cQy}$, where $\mathbf{Q} \in \mathbb{R}^{m \times p}$ is an i.i.d. Gaussian matrix, normalised to have unit-norm columns, y are drawn uniformly at random from the unit sphere, and c is the largest scalar such that $\|\mathbf{x}\|_\infty \leq \frac{1}{2}$ for all $\mathbf{x}$ in the generated dataset.
- **Tiny ImageNet dataset** : The data data comes from the Tiny ImageNet dataset (2015), taking pixel values in [0,1]. For each trial, we select the dataset by sampling n points at random (without replacement for Tiny ImageNet). For these experiments we

Algorithm 4 Naive+

```
**Input parameters :** $k$, $n$ and $\delta$.
**Default parameter :** $num\_steps = m \times n$
$m_k \leftarrow 1$
$d_{mode} \leftarrow \infty$
**for** $i$ in $range(n)$ **do**
    Initialize $\widehat{d_i}$ and $T_i$ as zero vectors
    $\widehat{d_i}, T_i, u, l \leftarrow AdaptiveKNN(\widehat{d_i}, T_i)$
    $d_{i,i_k} \leftarrow$ Exact $k^{th}$ neighbour distance
    **if** $d_{i,i_k} < d_{mode}$ **then**
        $d_{mode} \leftarrow d_{i,i_k}$
        $m_k \leftarrow i$
    **end if**
**end for**
**return** $x_{m_k}$
```

used m = 12288, where the choice for m comes from the dimensions of the Tiny ImageNet images, which is $64 \times 64 \times 3$.

B. Baselines

To gauge the performance of our algorithm, we compared it against the performance of two algorithms, which we call Naive and Naive+.

- **Naive** : In this algorithm, distance between each pair is computed exactly, and then the mode is selected. Thus, this algorithm makes mn^2 queries.
- **Naive+** : In this algorithm, for each point in the dataset, set of k-NN is found as specified in [2], and then the k^{th} neighbour is found with a probability of 1 - δ. Then the point with the nearest k^{th} neighbor is picked as the mode. The detailed algorithm is presented in Algorithm 4

C. Experiment 1: Variation of parameters

The concentration bound used in the experiments is

$$\alpha(u) = \sqrt{\frac{C_\alpha \log(1 + (1 + \log(u))n/\delta)}{u}}$$

where C_α is varied, and the accuracy and the the number of queries is plotted against C_α. This concentration bound is same as the one used in [2] during experiments and is used to estimate the sub-Gaussian parameter σ .

When the concentration bound

$$\alpha(u) = \sqrt{\frac{2\beta(u, \delta/n)}{u}}$$

with $\beta(u, \delta') = \log(1/\delta') + 3 \log\log(1/\delta') + 1.5 \log(1 + \log(u))$ was used, the accuracy was 1 with number of queries almost equal to that of Naive algorithm implying that is bound is extremely loose. As a result empirically determining σ by varying C_α is required.

Apart from C_α, other parameters varied are k, n and num_steps. In all the plots, the number of queries is normalised by dividing with mn^2, which is the number of queries in ”Naive” algorithm.

The default value of parameters are as follows :

- C_α = 0.001 (Artificial) and C_α = 0.02 (TinyImageNet)
- k = 10
- n = 100
- m = 12288
- $num\_steps = \frac{m \times n}{100}$

For each set of parameters, 40-100 random trials are done, then the average accuracy and average number of queries are plotted. For each trial, accuracy = 1, if the estimated mode = actual mode, 0 otherwise.

1) Varying C_α : The results are plotted in figures 1 and 2. As C_α is increased, the concentration bound increases, leading to a better accuracy but higher normalised number of queries to terminate. This plot help us choose the value of C_α, for a given requirement on accuracy.

For further experiments, C_α = 0.001 was chosen for Artificial dataset, while C_α = 0.02 was chosen for Tiny ImageNet dataset. This was chosen by choosing the C_α with minimum number of queries with accuracy = 1.

2) Varying k : The results are plotted in 3 and 4. As k is increased, the accuracy decreases until it is close to n. The normalised number of queries increases initially as k is increased, reaching a maximum at around $\frac{n}{2}$, then decreases.

Given a requirement on accuracy, this plot helps us find the value of k for which the current value of C_α works. Also, if the current C_α works for some $k = k_0$, it should be safe to assume that the same C_α should work for $k < k_0$, as the accuracy decreases monotonically as k increases.

3) Varying n : For the Tiny Imagenet dataset, the accuracy is 1 for all values of n. The normalised number of queries along with time taken is plotted in figure 5. As n increases, normalised number of queries decreases. We can conclude that the rate of increase in the unnormalised number of queries is less than that of mn^2.

For the Artificial dataset, C_α = 0.0001 was chosen for this experiment, so as to observe a change in accuracy. As shown in figure 6, accuracy increases monotonically as n is increased.

4) Varying num_steps : As far as minimizing the number of queries is concerned, num_steps = 1 is the ideal choice. However, to speed up the simulations, higher value of num_steps must be chosen. As shown in figure 7 and 8, normalised number of queries decreases as num_steps is decreased, but is fairly constant after a point. To strike a balance, $num\_steps = \frac{m \times n}{100}$ is chosen for other experiments.

D. Experiment 2: Accuracy vs Number of queries

It might be of useful to know how accurately an algorithm estimates the mode, given the number of queries it can use. To analyse this, accuracy vs normalised number of queries is plotted in figure 9, for our algorithm and Naive+.

Our algorithm has better accuracy for any given number of queries, and the number of queries required to reach accuracy 1 is lower too.

As for each trial, accuracy is set to 0 if the estimated mode (say x_i) is not equal to the actual mode (say x_j), this measurement does not acknowledge if d_{i,i_k} is close to d_{j,j_k} or far. Hence, another parameter, namely, ”relative distance error” was plotted for both the algorithms. Relative distance error is set to 0 if the mode is estimated correctly, else it is set to:

$$\text{Relative Distance Error} = \frac{d_{i,i_k} - d_{j,j_k}}{d_{j,j_k}}$$

Relative distance error vs normalised number of queries is plotted in figure 10. As expected, the error decreases as the number of queries increase, and our algorithm performs better than Naive+.

E. Experiment 3 : Different Elimination strategies

As detailed in Algorithm 1, the point with minimum LCB on the distance to k^{th} neighbour (b_1) is picked for sampling. Another variant is when all the points that are still in contention are picked in a round robin fashion similar to the Action Elimination Strategy [6]. This scheme was used for the experiments mentioned in this section. A comparison was done between the two variants and normalized number of queries was plotted against number of points (n). This is shown in figure 11, which shows that the first variant performs better than the second.

VIII. Future Work

In future work we plan to provide theoretical results for the correctness of algorithm 1 and also find a theoretical bound on its query complexity. Apart from this we also plan to extend our algorithm for regime 2 and show its correctness using various theoretical and experimental results.

References

[1] S. Dasgupta and S. Kpotufe, “Optimal rates for k-nn density and mode estimation,” in *NIPS*, 2014.
[2] D. LeJeune, R. G. Baraniuk, and R. Heckel, “Adaptive estimation for approximate k-nearest-neighbor computations,” *CoRR*, vol. abs/1902.09465, 2019.
[3] B. Mason, A. Tripathy, and R. Nowak, “Learning nearest neighbor graphs from noisy distance samples,” 2019.
[4] H. Jiang, J. Li, and M. Qiao, “Practical algorithms for best-k identification in multi-armed bandits,” *CoRR*, vol. abs/1705.06894, 2017.
[5] D. Shah, T. Choudhury, N. Karamchandani, and A. Gopalan, “Sequential mode estimation with oracle queries,” 2019.
[6] K. Jamieson and R. Nowak, “Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting,” in *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, 2014.
[7] E. Kaufmann, O. Cappé, and A. Garivier, “On the complexity of best arm identification in multi-armed bandit models,” 2014.
[8] K. Jamieson and R. Nowak, “Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting,” in *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, 2014.

IX. Figures

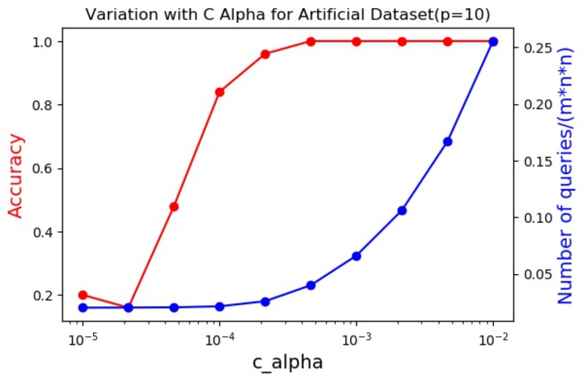

Fig. 1: Varying C_α : Artificial dataset

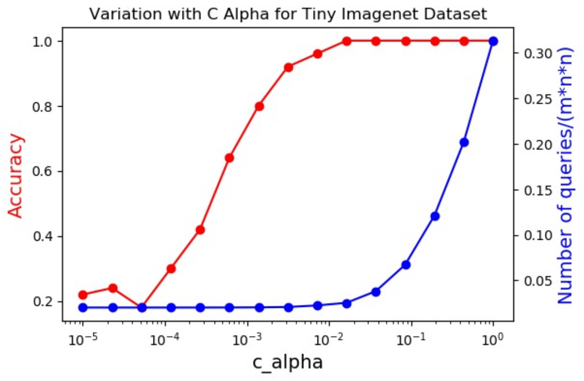

Fig. 2: Varying C_α : Tiny ImageNet dataset

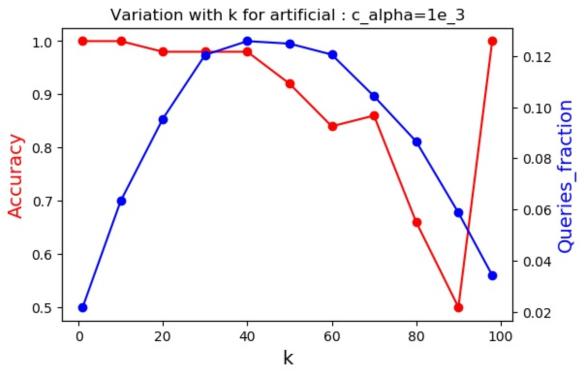

Fig. 3: Varying k : Artificial dataset

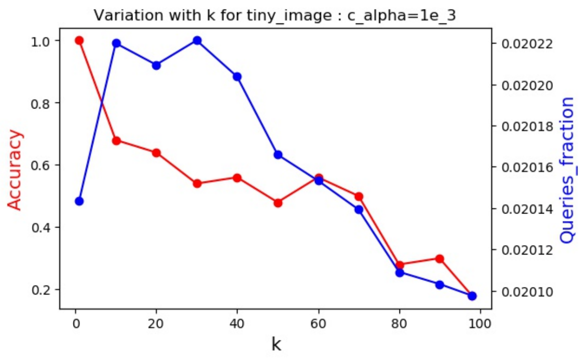

Fig. 4: Varying k : Tiny ImageNet dataset

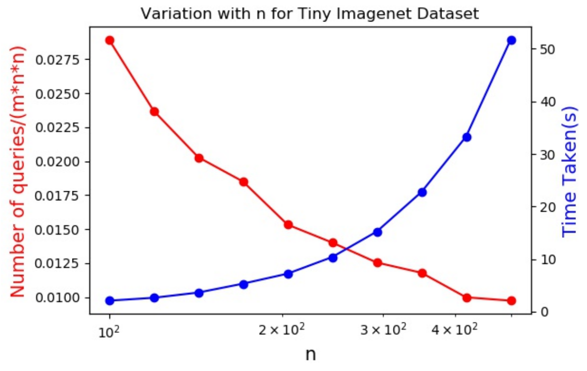

Fig. 5: Varying n : Tiny ImageNet dataset

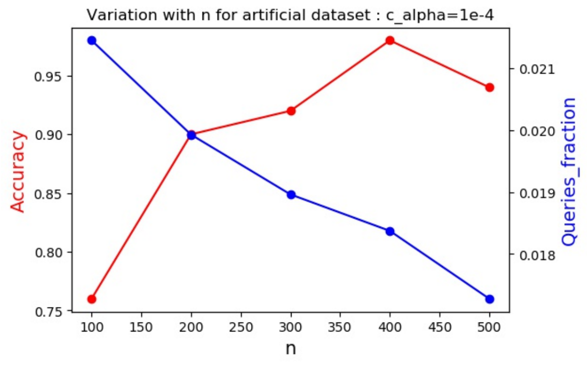

Fig. 6: Varying n : Artificial dataset

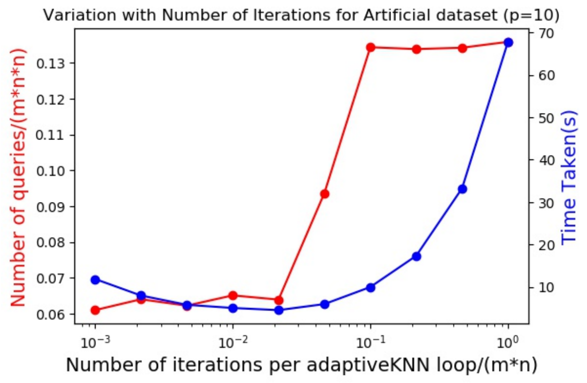

Fig. 7: Varying num_steps : Artificial dataset

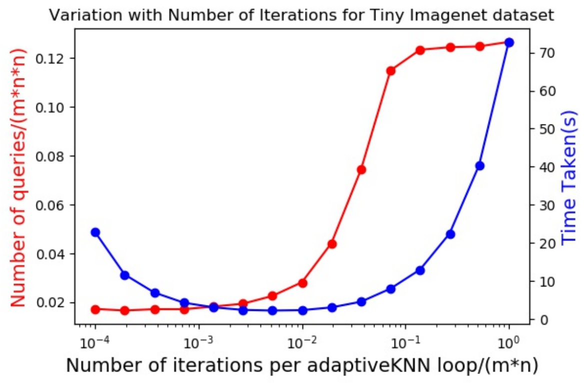

Fig. 8: Varying num_steps : Tiny ImageNet dataset

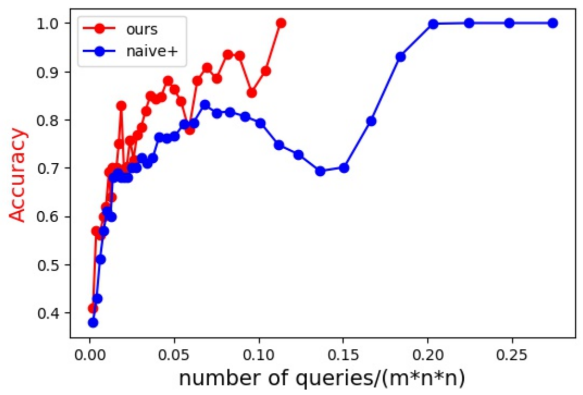

Fig. 9: Accuracy vs Number of queries

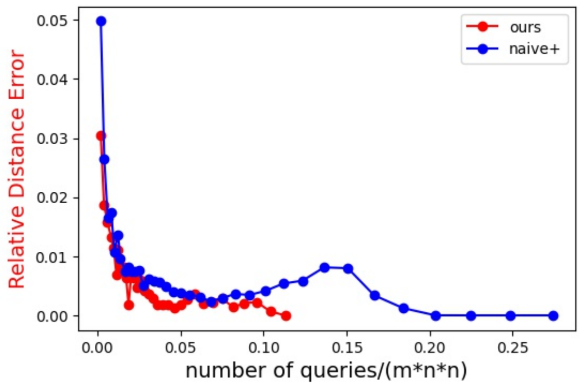

Fig. 10: Relative distance error vs Number of queries

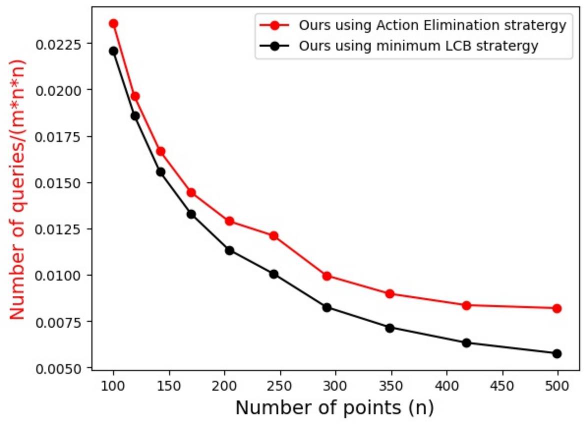

Fig. 11: Comparison of elimination strategy

Appendix A
Proof of Theorem 1

We will prove this theorem by first proving $LCB^i_{\{k\}} \leq d^i_k$ by the help of following result.

Lemma 4: At most $k-1$ values of j exist for $j \in [n]$ such that $d^i_j < LCB^i_{\{k\}}$.

We prove the **Lemma 4** by contradiction. Let us assume there exists a set of $k-1+n$ points represented by $\{p_1, p_2, \ldots, p_{k-1+n}\}$, where $n \geq 1$ and the following condition holds.

$$d^i_{p_j} < LCB^i_{\{k\}}, \forall j \in [k-1+n]$$

We know that $LCB^i_j \leq d^i_j$ by definition, therefore following is true.

$$LCB^i_{p_j} \leq d^i_{p_j} < LCB^i_{\{k\}}, \forall j \in [k-1+n]$$

This implies atleast $k-1+n$ values of j exist so that $LCB^i_j < LCB^i_{\{k\}}$. However, from eq. 15 we know that exactly $k-1$ such values of j exist. These two statements are only possible simultaneously if $n=0$ however this is a contradiction of our original assumption. Therefore, **Lemma 4** is true.

Next, it's easy to show that $LCB^i_{\{k\}} \leq d^i_k$ because if this is not true and $d^i_k < LCB^i_{\{k\}}$ there will be atleast k values such that $d^i_j < LCB^i_{\{k\}}$, which will be a contradiction of **Lemma 4**. Therefore, we have the following result.

$$LCB^i_{\{k\}} \leq d^i_k \tag{26}$$

And by similar arguments we can also show the following.

$$d^i_k \leq UCB^i_{(k)} \tag{27}$$

By putting together eqs. 26 and 27 proves that **Theorem 1** is true.