Adaptive Estimation for Approximate k-Nearest-Neighbor Computations

Daniel LeJeune
Rice University

Richard G. Baraniuk
Rice University

Reinhard Heckel
Rice University

Abstract

Algorithms often carry out equally many computations for "easy" and "hard" problem instances. In particular, algorithms for finding nearest neighbors typically have the same running time regardless of the particular problem instance. In this paper, we consider the approximate k-nearest-neighbor problem, which is the problem of finding a subset of $O(k)$ points in a given set of points that contains the set of k nearest neighbors of a given query point. We propose an algorithm based on adaptively estimating the distances, and show that it is essentially optimal out of algorithms that are only allowed to adaptively estimate distances. We then demonstrate both theoretically and experimentally that the algorithm can achieve significant speedups relative to the naïve method.

1 INTRODUCTION

A large number of algorithms in machine learning and signal processing are based on distance computations. The algorithms for solving the associated computational problems are typically designed to perform well on a set of problem instances, in a worst-case or average-case sense, but do not necessarily have optimal or close-to-optimal computational complexity on any given problem instance. As a consequence, these algorithms often have running times and guarantees that are the same for "easy" and "hard" problems.

Ideally, we would like an algorithm that adapts to any given problem instance and only carries out the computations necessary for that problem instance. In this paper, we consider an approach for speeding up algorithms and evaluating their complexity that is based on adapting to a given problem instance with random adaptive sampling techniques (Bagaria et al., 2018a,b). This approach is applicable to a variety of algorithms that are based on distance computations. Adding such adaptivity to algorithms can significantly speed up the algorithms' running times, since computationally easy problems have accordingly smaller running times.

For concreteness, we focus on the problem of *approximate k-nearest-neighbor* (k-NN) computation. Specifically, given a query point $\mathbf{x}$ and a set of n points, $\mathcal{X}$, our goal is to find a subset containing the k nearest neighbors of the query point. Our intuition is that an "easy" k-NN problem instance is one where there is a set of at least k points that are close to the query point, and the other points are rather far, such that the evaluation of only a few coordinate-wise distances of the far points is sufficient to know that they are farther away than the close points. Contrarily, a "hard" problem instance is one where the distance from $\mathbf{x}$ to all other points is very similar, and thus it is difficult to find a subset of k nearest neighbors.

We note that other formulations of the approximate k-NN problem are common as well. For example, Andoni and Indyk (2006) consider a formulation where points can be returned whose distance from the query point is within a multiplicative factor to its nearest points.

We propose an algorithm, which we call the *adaptive k-NN algorithm*, that adaptively estimates the distances and exploits the fact that for finding a set containing the k nearest neighbors, it is not necessary to compute all distances exactly. In particular, for easy problem instances, coarse estimates are sufficient to identify a subset containing the k nearest neighbors. Contrary to previous approaches, in particular that of Bagaria et al. (2018b), we focus on identifying a set *containing* the k nearest neighbors, since this is computationally considerably cheaper than identifying the *exact* set of k nearest neighbors.

We prove that the adaptive k-NN algorithm is *near*

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

instance-optimal for a restricted class of problems in the class of randomized algorithms that are based on adaptively estimating the distances. In a nutshell, the proof strategy is as follows: guaranteeing a solution to a given computational problem based on estimated distances (say, k-NNs) requires sufficiently good estimates of the distances. With standard change of measure techniques (Kaufmann et al., 2016) we can derive instance-dependent lower bounds on the sample complexity required to obtain sufficiently good estimates. This sample complexity is also a lower bound on the computational complexity of the respective algorithm, since at the very least the distances have to be computed sufficiently well to be sure a subset of the k nearest neighbors can be identified with high probability. Further, we show that this computational complexity can also be achieved by designing an approximate k-NN algorithm that estimates the distances adaptively in time almost linear in the sample complexity.

2 RELATED WORK ON k-NN

There are many highly efficient algorithms that solve versions of the (approximate) k-NN problem. If the dimension of the data points is low, then the k-d tree algorithm (Bentley, 1975) performs very well. This algorithm builds a balanced k-d tree and traverses the tree to find the nearest neighbors. Contrary to our approach, the algorithm is based on pre-processing the data and thus becomes efficient only when performing many queries, so that the cost of building the tree becomes negligible. In addition, k-d trees become inefficient in high dimensions.

In order to overcome complexity in high dimensions, a number of works have proposed to find solutions that are approximate in that the algorithm is only asked to return points that are close in distance to the true nearest neighbors, for example, by using random projections (Ailon and Chazelle, 2006). Perhaps the most popular class of algorithms for performing approximate nearest-neighbor search is based on locality sensitive hashing (Andoni and Indyk, 2006). This class of algorithms works very well in theory and practice, but again uses a pre-processing step that is not negligible if only one query is executed.

If many k-NN queries are carried out on the same dataset, then the k-d algorithm for small dimensions and locality sensitive hashing algorithms for higher dimensions are significantly more efficient than algorithms based on adaptively estimating scores, such as the algorithm proposed here, since then the amortized pre-processing costs become negligible. In contrast, our approach is efficient in high dimensions and when we only carry out one or very few queries.

A setting particularly relevant to our approach is that in which the dataset is rapidly changing, where the assumption of other k-NN algorithms that preprocessing costs become negligible over time no longer holds. One such example is in the Impicit Maximum Likelihood Estimation procedure by Li and Malik (2018), where at each iteration nearest-neighbor queries must be performed against a set of samples from the new estimate of the distribution.

There are a few recent success stories where adaptive randomized algorithms significantly speed up computational problems: the Monte-Carlo tree search method for decision processes (Kocsis and Szepesvári, 2006), hyperparameter optimization in machine learning (Li et al., 2018), finding the medoid in a large collection of high-dimensional points (Bagaria et al., 2018a), and solving discrete optimization problems involving distance computations adaptively (Bagaria et al., 2018b). All four works apply standard bandit algorithms in a creative way. Most related to our work is that of Bagaria et al. (2018b), which proposes an efficient algorithms for solving the k-NN problem using an adaptive sampling strategy, similar to the one proposed here. The main difference is that we consider the approximate k-NN problem, which is a more general problem that contains the problem of finding the exact k-NN as a special case. In order to solve the approximate k-NN problem, we have to solve a non-standard approximate bandit problem. In addition, we provide an algorithm that is *near instance-optimal* in the class of algorithms that *estimate* the distances for a restricted class of possible data points.

3 PROBLEM STATEMENT

Suppose we are given a set of high-dimensional points

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

and our goal is to find, for another given point $\mathbf{x} \in \mathbb{R}^m$, a set of size $O(k)$ that includes the k nearest neighbors of $\mathbf{x}$ to $\mathcal{X}$ in ℓ_2-distance (our results and discussion generalize to other distances, such as the ℓ_1-distance). This is a generalization of the exact k-nearest-neighbor problem and has applications in a vast number of classification tasks (Hastie et al., 2009).

For convenience, we assume that all points are normalized such that $\|\mathbf{x}\|_\infty \leq 1/2$, where $\|\mathbf{x}\|_\infty$ denotes the largest absolute value of the vector $\mathbf{x}$.

We can brute-force solve the problem by computing all distances and then sorting, which yields a worst case complexity of $O(mn + n \log n)$. Our intuition is that it is unnecessary to compute the distances exactly, and that by approximating the distances we can save computations.

4 THE ADAPTIVE k-NN ALGORITHM

The idea behind our *adaptive k-NN algorithm* is to adaptively estimate the distances. Then, the problem of finding a superset containing the k nearest neighbors reduces to a *multi-armed bandit* problem with the goal of identifying a set of size $k+h$ containing the k smallest arms. Using that—up to a logarithmic factor—the sample complexity of the corresponding algorithm is equal to the computational complexity, we can provide an upper bound on the computational complexity of the algorithm by proving an upper bound on the sample complexity. Likewise, we can prove a lower bound for all algorithms that use adaptive estimates of the distances.

Our algorithm is inspired by the Hamming-LUCB algorithm in (Heckel et al., 2018) which in turn builds on the Lower-Upper Confidence Bound (LUCB) strategy, a popular algorithm for identifying the top-k items in a bandit problem (Kalyanakrishnan et al., 2012) and for ranking from pairwise comparisons (Heckel et al., 2019). The algorithm is based on actively identifying sets $\widehat{\mathcal{S}}_{\text{close}}$ and $\widehat{\mathcal{S}}_{\text{far}}$ consisting of k and $n-k-h$ points, respectively, such that with high confidence the points in the first set have a smaller distance to $\mathbf{x}$ than the points in the second set. Once we have found such sets, the set

$$\widehat{\mathcal{S}} = \{1, \dots, n\} \setminus \widehat{\mathcal{S}}_{\text{far}}$$

contains the closest k points. Note that the cardinality of $\widehat{\mathcal{S}}$ is $k+h$, so to obtain a set of cardinality $O(k)$ conaining the k nearest neighbors, we choose h on the order of k. We adaptively estimate the normalized squared distances

$$d_i = \frac{1}{m}\|\mathbf{x} - \mathbf{x}_i\|_2^2$$

by sampling indices $j_1, \dots, j_{T_i}$ uniformly at random[1] from all indices $\{1, \dots, m\}$ and then estimating the squared distance as

$$\widehat{d}_i(T_i) = \frac{1}{T_i} \sum_{j \in \{j_1, \dots, j_{T_i}\}} |[\mathbf{x}_i]_j - [\mathbf{x}]_j|^2,$$

where $[\mathbf{x}]_j$ denotes the j-th coordinate of $\mathbf{x}$. The key idea is to estimate the distances only sufficiently well as to be able to obtain the two sets $\widehat{\mathcal{S}}_{\text{close}}$ and $\widehat{\mathcal{S}}_{\text{far}}$ from them.

Let T_i be the counter of the number of samples used for estimating the respective distance. We define a confidence bound based on a non-asymptotic version of the law of the iterated logarithm (Kaufmann et al., 2016; Jamieson et al., 2014); it is of the form[2]

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

where u is an integer corresponding to the number of samples, and δ is a parameter such that the algorithm succeeds with probability at least $1-\delta$. Within each round, we also let $(\cdot)$ denote a permutation of $[n]$ such that $\widehat{d}_{(1)} \le \widehat{d}_{(2)} \le \cdots \le \widehat{d}_{(n)}$. We then define the indices

$$q_1 = \operatorname*{arg\,max}_{i \in \{(1),\dots,(k)\}} \widehat{d}_i + \alpha_i, \tag{1}$$

$$q_2 = \operatorname*{arg\,min}_{i \in \{(k+1+h),\dots,(n)\}} \widehat{d}_i - \alpha_i, \tag{2}$$

where $\alpha_i = \alpha(T_i)$. These indices are the analogues of the standard indices of the Lower-Upper Confidence Bound (LUCB) strategy from the bandit literature (Kalyanakrishnan et al., 2012) for the bottom k and top $n-h-k$ arms. The LUCB strategy for exact bottom-k recovery would update the scores q_1 and q_2 (for $h=0$) at each round. Our strategy will go after what it "thinks" are the bottom k items, $\widehat{\mathcal{S}}_{\text{close}} = \{(1), \dots, (k)\}$, and what it "thinks" are the top $n-k-h$ items, $\widehat{\mathcal{S}}_{\text{far}} = \{(k+1+h), \dots, (n)\}$. Moreover, the algorithm keeps all the other items, $\widehat{\mathcal{S}}_{\text{middle}} = \{(k+1), \dots, (k+h)\}$, in consideration for inclusion in these sets by keeping their confidence intervals below the confidence intervals of the items in $\widehat{\mathcal{S}}_{\text{far}}$ (see (3) in the algorithm below). This is crucial to ensure that the algorithm does not get stuck trying to distinguish the middle items, which in general requires many samples.

4.1 Logarithmic Computational Complexity for Each Iteration

We next describe several implementation details that are critical for ensuring that each iteration of the adaptive k-NN algorithm has computational complexity $O(\log(n))$ and not $O(n\log(n))$, which a naïve implementation of computing the permutations via sorting and max and min computations would have. The key to achieving a logarithmic computational complexity is realizing that since only two distance estimates are updated in each iteration, the orderings of the distance estimates and confidence bounds will not change much between iterations, and at each iteration we only update the distance estimate for some indices that mini-

[1] Alternatively, one could select $j_1, \dots, j_m$ uniformly at random without replacement, in which case $\widehat{d}_i(m) = d_i$ exactly. This could be implemented efficiently using ciphers, such as those described by Black and Rogaway (2002).

[2] The constants involved can explicitly be chosen as $\alpha(u) = \sqrt{\frac{2\beta(u,\delta/n)}{u}}$ with $\beta(u,\delta') = \log(1/\delta') + 3\log\log(1/\delta') + 1.5\log(1+\log(u))$.

Algorithm 1: Adaptive k-NN

Input: Confidence parameter δ, approximation parameter h

Initialization: For every $i \in [n]$, sample an index j uniformly at random from $[m]$ and set $\widehat{d}_i(1) = |[\mathbf{x}_i]_j - [\mathbf{x}]_j|^2$, $T_i = 1$.

Do until termination:

Let $(\cdot)$ denote a permutation of $[n]$ such that $\widehat{d}_{(1)} \leq \widehat{d}_{(2)} \leq \ldots \widehat{d}_{(n)}$.

For q_1 and q_2 defined by equation (2), define the index

$$b_2 = \operatorname*{arg\,max}_{i \in \{q_2, (k+1), \ldots, (k+h)\}} \alpha_i. \tag{3}$$

For $\ell \in \{q_1, b_2\}$, increment $T_\ell \leftarrow T_\ell + 1$, sample an index j uniformly at random from $[m]$, and update $\widehat{d}_\ell \leftarrow \frac{T_\ell - 1}{T_\ell}\widehat{d}_\ell + \frac{1}{T_\ell}|[\mathbf{x}_\ell]_j - [\mathbf{x}]_j|^2$. If $T_\ell = m$, then compute the distance d_ℓ exactly and set $\widehat{d}_\ell = d_\ell$ and $\alpha_\ell = 0$.

End Loop once the termination condition holds:

$$\widehat{d}_{q_1} + \alpha_{q_1} \leq \widehat{d}_{q_2} - \alpha_{q_2}. \tag{4}$$

Return $\widehat{\mathcal{S}} = \{(1), \ldots, (k+h)\}$ as an estimate of the set containing the k nearest neighbors.

mize or maximize some quantities relating to the confidence bound. This makes the algorithm amenable to the use of a *heap data structure* (Cormen et al., 2009) to reduce computational complexity.

Figure 1 illustrates how a total of seven heaps can be used to implement the adaptive k-NN algorithm efficiently. For each of $\widehat{\mathcal{S}}_{\text{close}}$, $\widehat{\mathcal{S}}_{\text{middle}}$, and $\widehat{\mathcal{S}}_{\text{far}}$, a set of two or three coupled heaps is maintained, providing ordering information on both the distance estimates (so that we can maintain our permutation at each iteration) and the confidence bounds (so that we can select which distance estimates to update). For example, to determine q_2, we can simply extract the minimum from the min-heap on $\widehat{\mathcal{S}}_{\text{far}}$ defined on the lower confidence bounds $\widehat{d}_i - \alpha_i$, which has a computational cost of $O(1)$. Later in the iteration, if we update the distance estimate at q_2, we update both the distance estimate min-heap and lower confidence bound min-heap on $\widehat{\mathcal{S}}_{\text{far}}$ accordingly, which has a computational cost of $O(\log(n))$. At the end of the iteration, after making such updates across all three sets of heaps, the distance estimate ordering may not be maintained; e.g., the largest distance estimate in $\widehat{\mathcal{S}}_{\text{middle}}$ may be larger than the smallest distance estimate in $\widehat{\mathcal{S}}_{\text{far}}$. Items from each set must be swapped with items from other sets accordingly to restore the distance estimate ordering.

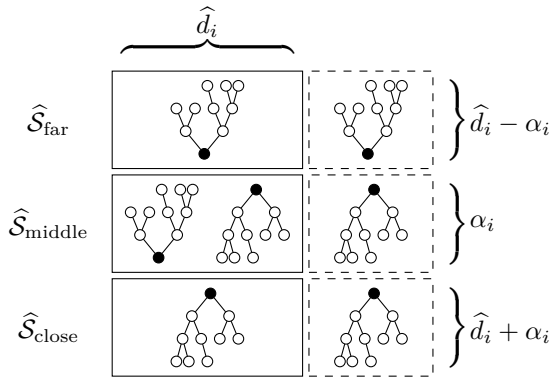

Figure 1: Illustration of how seven heaps can be used in the adaptive k-NN algorithm. Upward-branching trees indicate min-heaps and downward-branching trees indicate max-heaps.

Only two distance estimates are updated at each iteration, so at most a constant number of swaps that does not depend on n are required, and each swap involves updating the appropriate heaps, yielding that the computational cost of restoring the distance estimate ordering is also $O(\log(n))$. Thus, the overall complexity per iteration is $O(\log(n))$. We ask the reader to refer to our implementation[3] for further details.

4.2 Guarantees and Optimality of the Adaptive k-NN Algorithm

We next establish guarantees on the adaptive k-NN algorithm's success as well as on its computational complexity. The computational complexity depends on the gaps between the distances, defined as $\Delta_{i,j} = d_i - d_j$ through the function

$$N(\mathbf{x}, \mathcal{X}, h) = \widetilde{O}\Bigg(\sum_{i=1}^{k} \min(\Delta_{i,k+1+h}^{-2}, m) + \sum_{i=k+1+h}^{n} \min(\Delta_{k,i}^{-2}, m) + h \min(\Delta_{k,k+1+h}^{-2}, m) \Bigg). \tag{5}$$

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

Theorem 1. *For any points $\mathbf{x}$, $\mathcal{X}$, the adaptive k-NN algorithm run with parameters δ and h yields a set of size $k + h$ that contains the k nearest neighbors and has computational complexity at most*

$$N(\mathbf{x}, \mathcal{X}, h)$$

with probability at least $1 - \delta$.

[3] See `https://github.com/dlej/adaptive-knn`.

Note that the computational complexity of the adaptive k-NN algorithm is upper-bounded by the complexity of the naïve brute force method, $O(mn + n \log n)$, but is potentially significantly smaller. In particular, the computational complexity is small if there is a large gap between the k-th closest point and the $(k+h)$-th closest point. Taking $h = k$, for example, the algorithms returns a set of size $O(k)$ containing the k nearest neighbors. We next present two examples, one where the sample complexity of the adaptive k-NN algorithm is small, and one where it does not realize savings over the brute force method.

Both of these examples assume the data lies in a low-dimensional linear subspace, a regime where one might consider using projection-based k-NN methods. However, such methods require knowledge of the dimension of the subspace and often have projection cost at least $O(mn)$. Even if the dimension is known and the projection is as efficient as possible (such as subsampling), our method will still have the advantage in that it adapts to the distances.

First, consider a p-dimensional subspace spanned by a matrix $\mathbf{U} \in \mathbb{R}^{m \times p}$ that has orthogonal columns. In addition, suppose that the columns of $\mathbf{U}$ are incoherent with respect to the standard basis vectors $\mathbf{e}_i$. Specifically, suppose that the maximum inner product between a column of $\mathbf{U}$ and a standard basis vector obeys

$$\max_{i,j} \left| \left\langle \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|_2}, \mathbf{e}_j \right\rangle \right| \leq \frac{B}{\sqrt{m}}$$

for some constant $B \geq 1$. We say $\mathbf{U}$ is *incoherent* if this bound holds for small values of B. Suppose then that the columns of $\mathbf{U}$ are normalized to ℓ_2-norm $\sqrt{m/p}/B$ and that the dataset $\mathcal{X}$ and query point $\mathbf{x}$ lie in that subspace, i.e.,

$$\mathbf{x} = \mathbf{U}\mathbf{y}, \quad \mathcal{X} = \{\mathbf{U}\mathbf{y}_i : \mathbf{y}_i \in \mathcal{Y}\},$$

where $\mathbf{y} \in \mathbb{R}^p$ and $\mathcal{Y} \subset \mathbb{R}^p$ are the associated coefficient vectors. Assume that the associated coefficient vectors are normalized to have ℓ_2-norm equal to $1/2$. Denote the gaps in the coefficient space by $\Delta^{\mathbf{y}}_{i,j}$. From these assumptions, we are guaranteed that $\mathbf{x}$ and the points in $\mathcal{X}$ are bounded in ℓ_∞ norm by $1/2$. In addition, we have that

$$\frac{1}{m}\|\mathbf{x} - \mathbf{x}_i\|_2^2 = \frac{1}{pB^2}\|\mathbf{y} - \mathbf{y}_i\|_2^2.$$

Then $(\Delta_{i,j})^{-2} = (\Delta^{\mathbf{y}}_{i,j})^{-2} B^4$, so the computational complexity of the adaptive k-NN algorithm behaves like $\widetilde{O}\left(n(\Delta^{\mathbf{y}}_{k,k+1+h})^{-2}B^4\right)$ for large m; i.e., the computational complexity does not scale linearly with m. Hence, we can expect the adaptive k-NN algorithm to achieve significant computational savings when the data lies in a low-dimensional subspace of $\mathbb{R}^m$. Furthermore, the algorithm is able to realize these savings *without having this subspace or its dimension specified*. We illustrate this ability in our experiments below.

Next, suppose that the subspace is coherent with respect to the identitiy matrix. For example, consider the extreme case where all points lie in the one-dimensional subspace spanned by a single standard basis vector $\mathbf{e}_i$. Then, estimation of the distances to an accuracy of $O(1)$ requires at least $O(m)$ samples, and thus the adaptive k-NN algorithm will always have the same sample complexity as the naïve brute force algorithm.

We next show that the algorithm is optimal among active algorithms that estimate the distances by sampling indices when the data points satisfy $[\mathbf{x}]_j \in \{-1/2, 1/2\}$. We note that it is only the coordinates of the data that are so constrained; the normalized distances themselves can be essentially any values between 0 and 1 for large enough m.

Theorem 2. *For any $\delta \in (0, 0.14]$, let $\mathcal{A}$ denote an algorithm that can only interact with the data by sampling coordinates uniformly at random and yields, for any $\mathbf{x}$ and $\mathcal{X}$, the k nearest neighbors with probability at least $1-\delta$. Then, when $\mathcal{A}$ is run on any set of data points $\mathbf{x}$, $\mathcal{X}$ such that each coordinate of each point is either $-\frac{1}{2}$ or $\frac{1}{2}$, it has expected sample complexity at least*

$$N_{low}(\mathbf{x}, \mathcal{X}, h) = c' \left(\sum_{i=1}^{k-h} \Delta^{-2}_{i,k+1+h} + \sum_{i=k+1+h}^{n} \Delta^{-2}_{k-h,i} \right),$$

where $c' = \log\left(\frac{1}{2\delta}\right) \min_{\ell \in \{k-h, k+1+h\}} \{d_\ell(1-d_\ell)\}$.

Note that the above lower bound does not depend on the gaps involving the items $k-h+1, \ldots, k+h$. However, in the case when $d_{k-h} = d_k$, we can relate the lower bound and the upper bound by

$$N(\mathbf{x}, \mathcal{X}, 2h) \leq \widetilde{O}(N_{\text{low}}(\mathbf{x}, \mathcal{X}, h)),$$

so that we see that, up to rescaling of the approximation parameter h and logarithmic factors, the upper and lower bounds match. We emphasize that the lower bound only applies to algorithms that interact with the data by *uniformly sampling* the distances and only when we constrain the data points. Thus, Theorem 2 only tells us that the adaptive k-NN algorithm is optimal among algorithms based on adaptively estimating the distances, but it does not state that algorithms based on other strategies cannot perform better.

5 EXPERIMENTS

We run the adaptive k-NN algorithm both on artificial data restricted to low-dimensional subspaces and

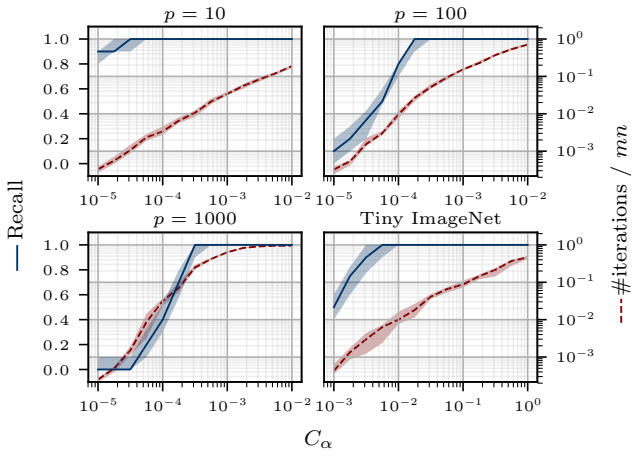

Figure 2: Effect of varying C_α for the adaptive k-NN algorithm on the random data lying in 10-dimensional (upper left), 100-dimensional (upper right), and 1000-dimensional (lower left) subspaces and on images from Tiny ImageNet (lower right). In each plot both the recall (blue, solid) and fraction of iterations used (red, dashed) are shown.

on real data to demonstrate the effectiveness in reducing computation over the naïve algorithm. The artificial data is generated via $\mathbf{x} = c\mathbf{Q}\mathbf{y}$, where $\mathbf{Q} \in \mathbb{R}^{m\times p}$ is an i.i.d. Gaussian matrix, normalized to have unit-norm columns, $\mathbf{y}$ are drawn uniformly at random from the unit sphere, and c is the largest scalar such that $\|\mathbf{x}\|_\infty \leq 1/2$ for all $\mathbf{x}$ in the generated dataset. The real data comes from the Tiny ImageNet dataset (2015), taking pixel values in $[0, 1]$. For each trial, we select $\mathcal{X}$ by sampling n points at random (without replacement for Tiny ImageNet) and then select the query point $\mathbf{x}$ by drawing another sample. For these experiments we used $n = 1000$ and $m = 12288$, where the choice for m comes from the dimensionality of the Tiny ImageNet images, which are $64 \times 64 \times 3$. Futher, we use $\alpha(u) = \sqrt{\frac{C_\alpha \log(1+(1+\log(u))n/\delta)}{u}}$ and vary C_α.

Figure 2 depicts the fraction of $\mathcal{S}_{\text{close}}$ contained in $\widehat{\mathcal{S}}$ (recall) along with ratio of the number of iterations taken by the adaptive k-NN algorithm versus the naïve algorithm as we vary C_α. Here $k = 10$, $h = 10$, and $\delta = 0.001$. We perform 20 random trials at each C_α and plot the median value (lines) and interquartile range (shaded area) over the trials. As we expect from our previous discussion, for low-dimensional subspaces (e.g., $p = 10$) we see significant computational savings (multiple orders of magnitude) by using the adaptive k-NN algorithm over the naïve method while still being able to return a set containing all of the k nearest neighbors. For larger p, such as $p = 1000$, this advantage is nearly non-existent. On Tiny ImageNet, we see similar performance gains to the small p setting, which can be explained by the fact that, like most real datasets, the data can be well-approximated as lying in a low-dimensional subspace.

6 PROOF OF THEOREM 1

The proof of Theorem 1 relies on relating the sample complexity to the computational complexity. We use that the computational complexity of the adaptive k-NN algorithm is no more than $\log(n)$ times the sample complexity of the adaptive k-NN algorithm. To see this, note that, as discussed in Section 4.1, each iteration has computational cost at most $O(\log(n))$. The initialization of the heaps at the start of the algorithm can be done in $O(n)$ computations using Floyd's algorithm, which is smaller or equal to the sample complexity. As a consequence, the computational complexity of the adaptive k-NN algorithm is no larger than $O(\log(n))$ times the sample complexity.

For notational convenience, we assume throughout that the distances are ordered so that

$$d_1 \leq d_2 \leq \ldots \leq d_n.$$

We begin by showing that the estimate $\widehat{d}_i(T_i)$ is guaranteed to be α_i-close to d_i, for all i, with high probability.

Lemma 3 (Kaufmann et al., 2016, Lemma 7)**.** *For any $\delta \in (0, 0.05)$, with probability at least $1 - \delta$, the event*

$$\mathcal{E}_\alpha := \left\{ \left| \widehat{d}_i(t) - d_i \right| \leq \alpha_i,\ \forall i \in [n],\ \forall t \geq 1 \right\} \tag{6}$$

occurs. The statement continues to hold for any $\delta \in (0, 1)$ with $\alpha_i = \alpha(T_i) = \sqrt{\frac{2\beta(T_i,\delta')}{T_i}}$, $\beta(t, \delta') = 2\log(125\log(1.12t)/\delta')$.

Lemma 3 is a non-asymptotic version of the law of the iterated logarithm from (Kaufmann et al., 2016) and (Jamieson et al., 2014). Note that the lemma uses that $\widehat{d}_i(t)$ is a sum of t independent random variables, each bounded between 0 and 1, and $\mathbb{E}\left[\widehat{d}_i(t)\right] = d_i$.

We first show that, on the event $\mathcal{E}_\alpha$ defined in equation (6), the set $\widehat{\mathcal{S}}$ contains the k nearest neighbors. On the event $\mathcal{E}_\alpha$, we have by the termination condition (4) (which is satisfied since the algorithm has terminated) that the items in the set $\widehat{\mathcal{S}}_{\text{close}} = \{(1), \ldots (k)\}$ have smaller distances than the items in the set $\widehat{\mathcal{S}}_{\text{far}} = \{(k+1+h), \ldots, (n)\}$. Because there are k distances that are smaller than the distances in $\widehat{\mathcal{S}}_{\text{far}}$, the set $\widehat{\mathcal{S}}_{\text{far}}$ cannot contain any of the k-nearest neighbors, i.e., $\widehat{\mathcal{S}}_{\text{far}} \subset \{1, \ldots, n\} \setminus \mathcal{S}_{\text{close}}$. Thus $\mathcal{S}_{\text{close}} \subset \widehat{\mathcal{S}}$.

We next show that on the event $\mathcal{E}_\alpha$, the adaptive k-NN algorithm terminates after the desired number of samples. Let $\gamma \coloneqq \frac{d_k + d_{k+1+h}}{2}$, and define the event that item i is bad as

$$\mathcal{E}_{\text{bad}}(i) = \begin{cases} \widehat{d}_i > \gamma - 3\alpha_i, & i \in \{1, \dots, k\} \\ \widehat{d}_i < \gamma + 3\alpha_i, & i \in \{k+1+h, \dots, n\} \\ \alpha_i > \frac{d_{k+1+h} - d_k}{4}, & \text{otherwise.} \end{cases}$$

Lemma 4. *If $\mathcal{E}_\alpha$ occurs and the termination condition (4) is false, then either $\mathcal{E}_{\text{bad}}(q_1)$ or $\mathcal{E}_{\text{bad}}(b_2)$ occurs.*

Lemma 4 is proved in Section A.1 in the supplementary material. Given Lemma 4, we can complete the proof in the following way. For an item i, define

$$\Delta_i = \begin{cases} d_{k+1+h} - d_i, & i \in \{1, \dots, k\} \\ d_i - d_k, & i \in \{k+1+h, \dots, n\} \\ d_{k+1+h} - d_k, & \text{otherwise,} \end{cases}$$

and let $\widetilde{T}_i$ be the smallest integer u satisfying the bound $\alpha(u) \leq \Delta_i / 8$. A simple calculation (see Section A.2 in the supplementary material) yields the following fact.

Fact 5. *On the event $\mathcal{E}_\alpha$, if $T_i \geq \widetilde{T}_i$ holds, then $\mathcal{E}_{\text{bad}}(i)$ does not occur.*

Let $t \geq 1$ be the t-th iteration of the steps in the algorithm, and let q_1 and b_2 be the two items selected in the algorithm. Note that in each iteration only those distances are estimated. By Lemma 4, we can therefore bound the total number distance estimate updates by

$$\begin{aligned} &2 \sum_{t=1}^{\infty} \mathbf{1}(\mathcal{E}_{\text{bad}}(q_1) \cup \mathcal{E}_{\text{bad}}(b_2)) \\ &\leq 2 \sum_{t=1}^{\infty} \sum_{i=1}^{n} \mathbf{1}((i = q_1 \cup i = b_2) \cap \mathcal{E}_{\text{bad}}(i)) \\ &\overset{\text{(i)}}{\leq} 2 \sum_{t=1}^{\infty} \sum_{i=1}^{n} \mathbf{1}((i = q_1 \cup i = b_2) \cap T_i(t) \leq \widetilde{T}_i) \\ &\overset{\text{(ii)}}{\leq} 2 \sum_{i=1}^{n} \widetilde{T}_i. \end{aligned} \tag{7}$$

For inequality (i), we used Fact 5, and inequality (ii) follows because $T_i(t) \leq \widetilde{T}_i$ can only be true for $\widetilde{T}_i$ iterations t.

We conclude the proof by noting that the definition of $\alpha(\cdot)$ yields the following upper bound (see Section A.3 in the supplementary material).

Fact 6. *For c sufficiently large,*

$$\widetilde{T}_i \leq c \log\left(\frac{n}{\delta}\right) \frac{\log(2\log(2/\Delta_i))}{\Delta_i^2}. \tag{8}$$

Applying this inequality to the right-hand side of equation (7) above concludes the proof.

7 PROOF OF THEOREM 2

Consider an algorithm $\mathcal{A}$ that can only interact with the data by sampling a index j uniformly at random and then obtaining $|[\mathbf{x}_i]_j - [\mathbf{x}]_j|$ in response. Because the points satisfy $[\mathbf{x}]_j \in \{-1/2, 1/2\}$, this is equivalent to drawing from a distribution ν_i corresponding to a binary $\{0,1\}$ random variable that has mean d_i. The problem of finding a subset containing the k nearest neighbors then corresponds to the problem of identifying a subset of all n distribution consisting of $k+h$ distributions containing the k smallest means. Here we will consider only the case where $h < k$. This is an *approximate* version of the bottom-k identification problem in the bandit literature (Kalyanakrishnan et al., 2012). Thus, to prove Theorem 2, we provide a lower bound on the sample complexity required to find a subset of $k+h$ distributions containing the k smallest means, and this lower bound is also a lower bound on the computational complexity.

Towards this goal, we first introduce some notation required to state a useful lemma (Kaufmann et al., 2016, Lemma 1) from the bandit literature. Let $\nu = \{\nu_i\}_{i=1}^n$ be a collection of n probability distributions, each supported on $\{0,1\}$. Consider an algorithm $\mathcal{A}$, that, at times $t = 1, 2, \dots$, selects the index $i_t \in [n]$ and receives an independent draw X_t from the distribution ν_{i_t} in response. Algorithm $\mathcal{A}$ may select i_t only based on past observations; that is, i_t is $\mathcal{F}_{t-1}$-measurable, where $\mathcal{F}_t$ is the σ-algebra generated by $i_1, X_1, \dots, i_t, X_t$. Algorithm $\mathcal{A}$ has a stopping rule ξ that determines the termination of $\mathcal{A}$. We assume that ξ is a stopping time measurable with respect to $\mathcal{F}_t$ and obeying $\mathrm{P}[\xi < \infty] = 1$.

Let $N_i(\xi)$ denote the total number of times index i has been selected by the algorithm $\mathcal{A}$ (until termination). For any pair of distributions ν and ν', we let $\mathrm{KL}(\nu, \nu')$ denote their Kullback-Leibler divergence, and for any $p, q \in [0,1]$, let $\mathrm{kl}(p,q) \coloneqq p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$ denote the Kullback-Leiber divergence between two binary random variables with success probabilities p, q.

With this notation, the following lemma relates the cumulative number of comparisons to the uncertainty between the actual distribution ν and an alternative distribution ν'.

Lemma 7 (Kaufmann et al., 2016, Lemma 1)**.** *Let ν, ν' be two collections of n probability distributions on $\mathbb{R}$. Then for any event $\mathcal{E} \in \mathcal{F}_\xi$ with $\mathrm{P}_\nu[\mathcal{E}] \in (0,1)$, we*

have

$$\sum_{i=1}^{n} \mathbb{E}_\nu\left[N_i(\xi)\right] \mathrm{KL}(\nu_i, \nu_i') \geq \mathrm{kl}(\mathrm{P}_\nu\left[\mathcal{E}\right], \mathrm{P}_{\nu'}\left[\mathcal{E}\right]). \quad (9)$$

In our setting, since ν_i and ν_i' are binary distributions, $\mathrm{KL}(\nu_i, \nu_i') = \mathrm{kl}(d_i, d_i')$. Let us now use Lemma 7 to prove Theorem 2.

Let $\mathcal{S}_1(\nu)$ be the set of k distributions out of the n distributions $\nu = (\nu_1, \ldots, \nu_n)$ with the smallest means. Define the event

$$\mathcal{E} \coloneqq \left\{\mathcal{S}_1(\nu) \subset \widehat{\mathcal{S}}\right\},$$

corresponding to success of the algorithm $\mathcal{A}$. Recalling that ξ is the stopping rule of algorithm $\mathcal{A}$, we are guaranteed that $\mathcal{E} \in \mathcal{F}_\xi$. Let $\mathcal{M} \coloneqq \{\ell_1, \ldots, \ell_{h+1}\}$ be a set of distinct items from the set of $n-k$ distributions with the largest means denoted by $\mathcal{S}_2(\nu)$. We next construct an alternative distribution ν' such that under that distribution, $\ell_1, \ldots, \ell_{h+1} \notin \mathcal{S}_2(\nu')$, or equivalently, that $\mathcal{M} \subseteq \mathcal{S}_1(\nu')$. Since we assume algorithm $\mathcal{A}$ succeeds for any distribution with probability at least $1-\delta$, we have both $\mathrm{P}_\nu\left[\mathcal{E}\right] \geq 1-\delta$ and $\mathrm{P}_{\nu'}\left[\mathcal{E}\right] \leq \delta$. To see this, note that if $\mathcal{A}$ succeeds under ν', then $\mathcal{M} \subset \widehat{\mathcal{S}}$. As such, there can be at most $k-1$ elements of $\mathcal{S}_1(\nu)$ in $\widehat{\mathcal{S}}$, which means that $\mathcal{E}$ does not occur.

It follows that

$$\begin{aligned}\mathrm{kl}(\mathrm{P}_\nu\left[\mathcal{E}\right], \mathrm{P}_{\nu'}\left[\mathcal{E}\right]) &\geq \mathrm{kl}(\delta, 1-\delta) \\ &= (1-2\delta)\log\frac{1-\delta}{\delta} \geq \log\frac{1}{2\delta}, \end{aligned} \quad (10)$$

where the last inequality holds for $\delta \leq 0.15$. It remains to specify the alternative distribution ν'. The alternative distribution is defined as

$$\nu_i' = \begin{cases} \nu_{k-h}, & i \in \mathcal{M} \\ \nu_i, & \text{otherwise.} \end{cases}$$

To be most precise on avoiding ties, for $\ell \in \mathcal{M}$, one should take $\nu_\ell' = \nu_{k-h} - \varepsilon$ for some $\varepsilon > 0$ and let $\varepsilon \to 0$, but we omit carrying out the associated technical details in this proof. It follows that, under the distribution ν', the items in the set $\mathcal{M}$ are not among the items with the $n-k$ largest means which ensures that $\mathcal{M} \cap \mathcal{S}_2(\nu') = \emptyset$.

Let N_ℓ be the total number of draws from the distribution ν_ℓ. We have that

$$\begin{aligned}\sum_{\ell \in \mathcal{M}} \mathrm{kl}(\nu_\ell, \nu_\ell')\mathbb{E}_\nu\left[N_\ell\right] &\overset{\text{(i)}}{=} \sum_{i=1}^{n} \mathbb{E}_\nu\left[N_i\right] \mathrm{kl}(\nu_i, \nu_i') \\ &\overset{\text{(ii)}}{\geq} \mathrm{kl}(\mathrm{P}_\nu\left[\mathcal{E}\right], \mathrm{P}_{\nu'}\left[\mathcal{E}\right]) \\ &\geq \log\frac{1}{2\delta}. \end{aligned} \quad (11)$$

Here step (i) follows from the fact that $\mathrm{kl}(\nu_i, \nu_i') = 0$ for all $i \notin [n] \setminus \mathcal{M}$ by definition of the ν_i', and step (ii) follows from Lemma 7. Finally, inequality (11) follows from inequality (10).

We next upper bound the KL divergence on the left hand side of inequality (11). Using the inequality $\log x \leq x - 1$, valid for $x > 0$, we have that for $\ell \in \mathcal{M}$,

$$\mathrm{kl}(\nu_\ell, \nu_\ell') \leq \mathrm{kl}_\ell, \quad \mathrm{kl}_\ell \coloneqq \frac{(d_\ell - d_{k-h})^2}{d_{k-h}(1-d_{k-h})}. \quad (12)$$

Applying inequality (12) to the left hand side of inequality (11) yields

$$\sum_{\ell \in \mathcal{M}} \mathrm{kl}_\ell \mathbb{E}_\nu\left[N_\ell\right] \geq \log\frac{1}{2\delta}, \quad (13)$$

which is valid for each subset $\mathcal{M} \subseteq \mathcal{S}_2(\nu)$ of cardinality $h+1$.

We can therefore obtain a lower bound on $\sum_{i \in \mathcal{S}_2(\nu)} \mathbb{E}_\nu\left[N_i\right]$ by solving the minimization problem:

$$\begin{aligned}&\underset{e_\ell \geq 0}{\text{minimize}} \sum_{\ell \in \mathcal{S}_2(\nu)} e_\ell \quad \text{subject to} \sum_{\ell \in \mathcal{M}} \mathrm{kl}_\ell e_\ell \geq \log\frac{1}{2\delta} \\ &\text{for all } \mathcal{M} \subseteq \mathcal{S}_2(\nu) \text{ of cardinality } h+1. \end{aligned} \quad (14)$$

Since the kl_ℓ are increasing in ℓ (recall that we assume the distances to be ordered such that $d_1 \leq d_2 \leq \ldots \leq d_n$) the solution to this optimization problem is $e_{k+1}, \ldots, e_{k+h} = 0$ and $e_\ell = \log(1/2\delta)/\mathrm{kl}_\ell$ for $\ell \geq k+1+h$.

Using an analogous line of arguments for items in the set $\mathcal{S}_1(\nu)$ (see Section B in the supplemental material), we arrive at the lower bound

$$\begin{aligned}&\log\frac{1}{2\delta} \sum_{i=1}^{k-h} \frac{d_{k+1+h}(1-d_{k+1+h})}{(d_i - d_{k+1+h})^2} \\ &+ \log\frac{1}{2\delta} \sum_{i=k+1+h}^{n} \frac{d_{k-h}(1-d_{k-h})}{(d_{k-h} - d_i)^2}\end{aligned}$$

on the number of comparisons. This concludes the proof.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. DL and RB are partially supported by NSF grants IIS-17-30574 and IIS-18-38177, AFOSR grant FA9550-18-1-0478, ARO grant W911NF-15-1-0316, ONR grants N00014-17-1-2551 and N00014-18-12571, DARPA grant G001534-7500, and DOD Vannevar Bush Faculty Fellowship (NSSEFF) grant N00014-18-1-2047. RH is partially supported by NSF award IIS-1816986.

References

Tiny imagenet visual recognition challenge. `https://tiny-imagenet.herokuapp.com/`, 2015. Accessed: 2019-02-21.

N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 557–563, 2006.

A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468. IEEE, 2006.

V. Bagaria, G. Kamath, V. Ntranos, M. Zhang, and D. Tse. Medoids in almost-linear time via multi-armed bandits. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 500–509, 2018a.

V. Bagaria, G. M. Kamath, and D. N. Tse. Adaptive monte-carlo optimization. *arXiv:1805.08321*, 2018b.

J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, Sept. 1975.

J. Black and P. Rogaway. Ciphers with arbitrary finite domains. In *Proceedings of the Cryptographers' Track at the RSA Conference on Topics in Cryptology*, pages 114–130. Springer, 2002.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3 edition, 2009.

T. J. Hastie, R. J. Tibshirani, and J. J. H. Friedman. *The elements of statistical learning*. Springer, 2 edition, 2009.

R. Heckel, M. Simchowitz, K. Ramchandran, and M. Wainwright. Approximate ranking from pairwise comparisons. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 1057–1066, 2018.

R. Heckel, N. B. Shah, K. Ramchandran, and M. J. Wainwright. Active ranking from pairwise comparisons and when parametric assumptions don't help. *Annals of Statistics*, 2019. arXiv:1606.08842.

K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. lil' UCB : An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, pages 423–439, 2014.

S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. PAC subset selection in stochastic multi-armed bandits. In *Proceedings of the 29th International Conference on Machine Learning*, pages 655–662, 2012.

E. Kaufmann, O. Cappé, and A. Garivier. On the complexity of best-arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17(1):1–42, 2016.

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, pages 282–293. Springer-Verlag, 2006.

K. Li and J. Malik. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.