Fast k-Nearest Neighbour Search via Dynamic Continuous Indexing

Supplementary Material

Ke Li KE.LI@EECS.BERKELEY.EDU
Jitendra Malik MALIK@EECS.BERKELEY.EDU
University of California, Berkeley, CA 94720, United States

Below, we present proofs of the results shown in the paper. We first prove two intermediate results, which are used to derive results in the paper. Throughout our proofs, we use $\{p^{(i)}\}_{i=1}^n$ to denote a re-ordering of the points $\{p^i\}_{i=1}^n$ so that $p^{(i)}$ is the i^{th} closest point to the query q. For any given projection direction u_{jl} associated with a simple index, we also consider a ranking of the points $\{p^i\}_{i=1}^n$ by their distance to q under projection u_{jl} in nondecreasing order. We say points are ranked before others if they appear earlier in this ranking.

Lemma 14. *The probability that for all constituent simple indices of a composite index, fewer than n_0 points exist that are not the true k-nearest neighbours but are ranked before some of them, is at least* $\left[1 - \frac{1}{n_0 - k}\sum_{i=2k+1}^{n} \frac{\left\|p^{(k)} - q\right\|_2}{\left\|p^{(i)} - q\right\|_2}\right]^m$.

Proof. For any given simple index, we will refer to the points that are not the true k-nearest neighbours but are ranked before some of them as *extraneous points*. We furthermore categorize the extraneous points as either *reasonable* or *silly*. An extraneous point is reasonable if it is one of the $2k$-nearest neighbours, and is silly otherwise. Since there can be at most k reasonable extraneous points, there must be at least $n_0 - k$ silly extraneous points. Therefore, the event that n_0 extraneous points exist must be contained in the event that $n_0 - k$ silly extraneous points exist.

We find the probability that such a set of silly extraneous points exists for any given simple index. By Theorem 3, where we take $\{v_{i'}^s\}_{i'=1}^{N'}$ to be $\{p^{(i)} - q\}_{i=1}^k$, $\{v_i^l\}_{i=1}^N$ to be $\{p^{(i)} - q\}_{i=2k+1}^n$ and k' to be $n_0 - k$, the probability that there are at least $n_0 - k$ silly extraneous points is at most $\frac{1}{n_0 - k}\sum_{i=2k+1}^{n}\left(1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\left\|p^{(k)} - q\right\|_2}{\left\|p^{(i)} - q\right\|_2}\right)\right)$. This implies that the probability that at least n_0 extraneous points exist is bounded above by the same quantity, and so the probability that fewer than n_0 extraneous points exist is at least $1 - \frac{1}{n_0 - k}\sum_{i=2k+1}^{n}\left(1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\left\|p^{(k)} - q\right\|_2}{\left\|p^{(i)} - q\right\|_2}\right)\right)$. Hence, the probability that fewer than n_0 extraneous points exist for all constituent simples indices of a composite index is at least $\left[1 - \frac{1}{n_0 - k}\sum_{i=2k+1}^{n}\left(1 - \frac{2}{\pi}\cos^{-1}\left(\frac{\left\|p^{(k)} - q\right\|_2}{\left\|p^{(i)} - q\right\|_2}\right)\right)\right]^m$. Using the fact that $1 - (2/\pi)\cos^{-1}(x) \leq x \quad \forall x \in [0, 1]$, this quantity is at least $\left[1 - \frac{1}{n_0 - k}\sum_{i=2k+1}^{n}\frac{\left\|p^{(k)} - q\right\|_2}{\left\|p^{(i)} - q\right\|_2}\right]^m$. □

Lemma 15. *On a dataset with global relative sparsity (k, γ), the probability that for all constituent simple indices of a composite index, fewer than n_0 points exist that are not the true k-nearest neighbours but are ranked before some of them, is at least* $\left[1 - \frac{1}{n_0 - k}O\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)\right]^m$.

Proof. By definition of global relative sparsity, for all $i \geq 2k + 1$, $\left\|p^{(i)} - q\right\|_2 > \gamma\left\|p^{(k)} - q\right\|_2$. By applying this recursively, we see that for all $i \geq 2^{i'}k + 1$, $\left\|p^{(i)} - q\right\|_2 > \gamma^{i'}\left\|p^{(k)} - q\right\|_2$. It follows that $\sum_{i=2k+1}^{n}\frac{\left\|p^{(k)} - q\right\|_2}{\left\|p^{(i)} - q\right\|_2}$ is less than $\sum_{i'=1}^{\lceil\log_2(n/k)\rceil - 1} 2^{i'}k\gamma^{-i'}$. If $\gamma \geq 2$, this quantity is at most $k\log_2\left(\frac{n}{k}\right)$. If $1 \leq \gamma < 2$, this quantity is:

$$\begin{aligned}
&k\left(\frac{2}{\gamma}\right)\left(\left(\frac{2}{\gamma}\right)^{\lceil\log_2(n/k)\rceil - 1} - 1\right) / \left(\frac{2}{\gamma} - 1\right) \\
&= O\left(k\left(\frac{2}{\gamma}\right)^{\lceil\log_2(n/k)\rceil - 1}\right) \\
&= O\left(k\left(\frac{n}{k}\right)^{1-\log_2\gamma}\right)
\end{aligned}$$

Combining this bound with Lemma 14 yields the desired result.

□

Lemma 6. *For a dataset with global relative sparsity (k, γ), there is some $\tilde{k} \in \Omega(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma}))$ such that the probability that the candidate points retrieved from a given*

composite index do not include some of the true k-nearest neighbours is at most some constant $\alpha < 1$.

Proof. We will refer to points ranked in the top $\tilde{k}$ positions that are the true k-nearest neighbours as *true positives* and those that are not as *false positives*. Additionally, we will refer to points not ranked in the top $\tilde{k}$ positions that are the true k-nearest neighbours as *false negatives*.

When not all the true k-nearest neighbours are in the top $\tilde{k}$ positions, then there must be at least one false negative. Since there are at most $k-1$ true positives, there must be at least $\tilde{k}-(k-1)$ false positives.

Since false positives are not the true k-nearest neighbours but are ranked before the false negative, which is a true k-nearest neighbour, we can apply Lemma 15. By taking n_0 to be $\tilde{k}-(k-1)$, we obtain a lower bound on the probability of the existence of fewer than $\tilde{k}-(k-1)$ false positives for all constituent simple indices of the composite index, which is $\left[1-\frac{1}{\tilde{k}-2k+1}O\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)\right]^m$. If each simple index has fewer than $\tilde{k}-(k-1)$ false positives, then the top $\tilde{k}$ positions must contain all the true k-nearest neighbours. Since this is true for all constituent simple indices, all the true k-nearest neighbours must be among the candidate points after $\tilde{k}$ iterations of the outer loop. The failure probability is therefore at most $1-\left[1-\frac{1}{\tilde{k}-2k+1}O\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)\right]^m$. So, there is some $\tilde{k} \in \Omega(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma}))$ that makes this quantity strictly less than 1.

□

Theorem 7. *For a dataset with global relative sparsity (k,γ), for any $\epsilon > 0$, there is some L and $\tilde{k} \in \Omega(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma}))$ such that the algorithm returns the correct set of k-nearest neighbours with probability of at least $1-\epsilon$.*

Proof. By Lemma 6, the first $\tilde{k}$ points retrieved from a given composite index do not include some of the true k-nearest neighbours with probability of at most α. For the algorithm to fail, this must occur for all composite indices. Since each composite index is constructed independently, the algorithm fails with probability of at most α^L, and so must succeed with probability of at least $1-\alpha^L$. Since $\alpha < 1$, there is some L that makes $1-\alpha^L \geq 1-\epsilon$. □

Theorem 8. *The algorithm takes $O(\max(dk\log(n/k), dk(n/k)^{1-1/d'}))$ time to retrieve the k-nearest neighbours at query time, where d' denotes the intrinsic dimension of the dataset.*

Proof. Computing projections of the query point along all u_{jl}'s takes $O(d)$ time, since m and L are constants. Searching in the binary search trees/skip lists T_{jl}'s takes $O(\log n)$ time. The total number of candidate points retrieved is at most $\Theta(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma}))$. Computing the distance between each candidate point and the query point takes at most $O(\max(dk\log(n/k), dk(n/k)^{1-\log_2\gamma}))$ time. We can find the k closest points to q in the set of candidate points using a selection algorithm like quickselect, which takes $O(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma}))$ time on average. Since the time taken to compute distances to the query point dominates, the entire algorithm takes $O(\max(dk\log(n/k), dk(n/k)^{1-\log_2\gamma}))$ time. Since $d' = 1/\log_2\gamma$, this can be rewritten as $O(\max(dk\log(n/k), dk(n/k)^{1-1/d'}))$. □

Theorem 9. *The algorithm takes $O(dn + n\log n)$ time to preprocess the data points in D at construction time.*

Proof. Computing projections of all n points along all u_{jl}'s takes $O(dn)$ time, since m and L are constants. Inserting all n points into mL self-balancing binary search trees/skip lists takes $O(n\log n)$ time. □

Theorem 10. *The algorithm requires $O(d+\log n)$ time to insert a new data point and $O(\log n)$ time to delete a data point.*

Proof. In order to insert a data point, we need to compute its projection along all u_{jl}'s and insert it into each binary search tree or skip list. Computing the projection takes $O(d)$ time and inserting an entry into a self-balancing binary search tree or skip list takes $O(\log n)$ time. In order to delete a data point, we simply remove it from each of the binary search trees or skip lists, which takes $O(\log n)$ time. □

Theorem 11. *The algorithm requires $O(n)$ space in addition to the space used to store the data.*

Proof. The only additional information that needs to be stored are the mL binary search trees or skip lists. Since n entries are stored in each binary search tree/skip list, the additional space required is $O(n)$. □

Theorem 12. *For any $\epsilon > 0$, m and L, the data-dependent algorithm returns the correct set of k-nearest neighbours of the query q with probability of at least $1-\epsilon$.*

Proof. We analyze the probability that the algorithm fails to return the correct set of k-nearest neighbours. Let p^* denote a true k-nearest neighbour that was missed. If the algorithm fails, then for any given composite index, p^* is

not among the candidate points retrieved from the said index. In other words, the composite index must have returned all these points before p^*, implying that at least one constituent simple index returns all these points before p^*. This means that all these points must appear closer to q than p^* under the projection associated with the simple index. By Lemma 2, if we take $\{v_i^l\}_{i=1}^N$ to be displacement vectors from q to the candidate points that are farther from q than p^* and v^s to be the displacement vector from q to p^*, the probability of this occurring for a given constituent simple index of the l^{th} composite index is at most $1 - \frac{2}{\pi}\cos^{-1}\left(\|p^* - q\|_2 / \|\tilde{p}_l^{\max} - q\|_2\right)$. The probability that this occurs for *some* constituent simple index is at most $1 - \left(\frac{2}{\pi}\cos^{-1}\left(\|p^* - q\|_2 / \|\tilde{p}_l^{\max} - q\|_2\right)\right)^m$. For the algorithm to fail, this must occur for all composite indices; so the failure probability is at most $\prod_{l=1}^L \left(1 - \left(\frac{2}{\pi}\cos^{-1}\left(\|p^* - q\|_2 / \|\tilde{p}_l^{\max} - q\|_2\right)\right)^m\right)$.

We observe that $\left\|p^* - q\right\|_2 \leq \left\|p^{(k)} - q\right\|_2 \leq \left\|\tilde{p}^{(k)} - q\right\|_2$ since there can be at most $k-1$ points in the dataset that are closer to q than p^*. So, the failure probability can be bounded above by $\prod_{l=1}^L \left(1 - \left(\frac{2}{\pi}\cos^{-1}\left(\left\|\tilde{p}^{(k)} - q\right\|_2 / \left\|\tilde{p}_l^{\max} - q\right\|_2\right)\right)^m\right)$. When the algorithm terminates, we know this quantity is at most ϵ. Therefore, the algorithm returns the correct set of k-nearest neighbours with probability of at least $1-\epsilon$. □

Theorem 13. *On a dataset with global relative sparsity (k, γ), given fixed parameters m and L, the data-dependent algorithm takes $O\left(\max\left(dk\log\left(\frac{n}{k}\right), dk\left(\frac{n}{k}\right)^{1-\log_2\gamma}, \frac{2d}{\left(1-\sqrt[m]{1-\sqrt[L]{\epsilon}}\right)^{d'}}\right)\right)$ time with high probability to retrieve the k-nearest neighbours at query time, where d' denotes the intrinsic dimension of the dataset.*

Proof. In order to bound the running time, we bound the total number of candidate points retrieved until the stopping condition is satisfied. We divide the execution of the algorithm into two stages and analyze the algorithm's behaviour before and after it finishes retrieving all the true k-nearest neighbours. We first bound the number of candidate points the algorithm retrieves before finding the complete set of k-nearest neighbours. By Lemma 15, the probability that there exist fewer than n_0 points that are not the k-nearest neighbours but are ranked before some of them in all constituent simple indices of any given composite index is at least $\left[1 - \frac{1}{n_0 - k}O\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)\right]^m$. We can choose some $n_0 \in \Theta\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)$ that makes this probability arbitrarily close to 1. So, there are $\Theta\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)$ such points in each constituent simple index with high probability, implying that the algorithm retrieves at most $\Theta\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)$ extraneous points from any given composite index before finishing fetching all the true k-nearest neighbours. Since the number of composite indices is constant, the total number of candidate points retrieved from all composite indices during this stage is $k + \Theta\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right) = \Theta\left(\max(k\log(n/k), k(n/k)^{1-\log_2\gamma})\right)$ with high probability.

After retrieving all the k-nearest neighbours, if the stopping condition has not yet been satisfied, the algorithm would continue retrieving points. We analyze the number of additional points the algorithm retrieves before it terminates. To this end, we bound the ratio $\left\|\tilde{p}^{(k)} - q\right\|_2 / \left\|\tilde{p}_l^{\max} - q\right\|_2$ in terms of the number of candidate points retrieved so far. Since all the true k-nearest neighbours have been retrieved, $\left\|\tilde{p}^{(k)} - q\right\|_2 = \left\|p^{(k)} - q\right\|_2$. Suppose the algorithm has already retrieved $n' - 1$ candidate points and is about to retrieve a new candidate point. Since this new candidate point must be different from any of the existing candidate points, $\left\|\tilde{p}_l^{\max} - q\right\|_2 \geq \left\|p^{(n')} - q\right\|_2$. Hence, $\left\|\tilde{p}^{(k)} - q\right\|_2 / \left\|\tilde{p}_l^{\max} - q\right\|_2 \leq \left\|p^{(k)} - q\right\|_2 / \left\|p^{(n')} - q\right\|_2$.

By definition of global relative sparsity, for all $n' \geq 2^{i'}k + 1$, $\left\|p^{(n')} - q\right\|_2 > \gamma^{i'}\left\|p^{(k)} - q\right\|_2$. It follows that $\left\|p^{(k)} - q\right\|_2 / \left\|p^{(n')} - q\right\|_2 < \gamma^{-\lfloor\log_2((n'-1)/k)\rfloor}$ for all n'. By combining the above inequalities, we find an upper bound on the test statistic:

$$\begin{aligned}
&\prod_{l=1}^L \left(1 - \left(\frac{2}{\pi}\cos^{-1}\left(\frac{\left\|\tilde{p}^{(k)} - q\right\|_2}{\left\|\tilde{p}_l^{\max} - q\right\|_2}\right)\right)^m\right) \\
&\leq \prod_{l=1}^L \left(1 - \left(1 - \frac{\left\|\tilde{p}^{(k)} - q\right\|_2}{\left\|\tilde{p}_l^{\max} - q\right\|_2}\right)^m\right) \\
&< \left[1 - \left(1 - \gamma^{-\lfloor\log_2((n'-1)/k)\rfloor}\right)^m\right]^L \\
&< \left[1 - \left(1 - \gamma^{-\log_2((n'-1)/k)+1}\right)^m\right]^L
\end{aligned}$$

Hence, if $\left[1 - \left(1 - \gamma^{-\log_2((n'-1)/k)+1}\right)^m\right]^L \leq \epsilon$, then $\prod_{l=1}^L \left(1 - \left(\frac{2}{\pi}\cos^{-1}\left(\left\|\tilde{p}^{(k)} - q\right\|_2 / \left\|\tilde{p}_l^{\max} - q\right\|_2\right)\right)^m\right) < \epsilon$. So, for some n' that makes the former inequality true, the stopping condition would be satisfied and so the algorithm must have terminated by this point, if not earlier. By rearranging the former inequality, we find that in order for it to hold, n' must be at least $2/\left(1 - \sqrt[m]{1 - \sqrt[L]{\epsilon}}\right)^{1/\log_2\gamma}$. Therefore, the number of

points the algorithm retrieves before terminating cannot exceed $2/\left(1-\sqrt[m]{1-\sqrt[L]{\epsilon}}\right)^{1/\log_2\gamma}$.

Combining the analysis for both stages, the number of points retrieved is at most

$$O\left(\max\left(k\log\left(\frac{n}{k}\right), k\left(\frac{n}{k}\right)^{1-\log_2\gamma}, \frac{2}{\left(1-\sqrt[m]{1-\sqrt[L]{\epsilon}}\right)^{\frac{1}{\log_2\gamma}}}\right)\right)$$

with high probability.

Since the time taken to compute distances between the query point and candidate points dominates, the running time is

$$O\left(\max\left(dk\log\left(\frac{n}{k}\right), dk\left(\frac{n}{k}\right)^{1-\log_2\gamma}, \frac{2d}{\left(1-\sqrt[m]{1-\sqrt[L]{\epsilon}}\right)^{\frac{1}{\log_2\gamma}}}\right)\right)$$

with high probability.

Applying the definition of intrinsic dimension yields the desired result.

□