Approximate Nearest Neighbor

Duong Hoang and Trang Tran

1 Problem

Given a set of n points P in $X = \mathbb{R}^d$, the nearest-neighbor (NN) problem is to find a point p in P that is closest to a query point q. This problem is of importance to many applications involving similarity searching such as data compression, databases and data mining, information retrieval, image and video databases, machine learning, pattern recognition, and statistics and data analysis. Suffering from the curse of dimensionality, existing algorithms either have query time linear in n and d, or have query time sub-linear in n and polynomial in d but exponential preprocessing cost n^d, making them infeasible for a large d (e.g., $d \geq 100$) which is common in practice.

It is observed that insisting on the absolute nearest neighbor is often an overkill, due to the heuristical nature of the use of distance metric and the selection of features in practice. Therefore the paper in discussion [2] aims to solve an approximate relaxation to nearest neighbor problem, namely the **ϵ-nearest neighbor** problem (ϵ-NN): to find a point $p \in P$ that is an ϵ-approximate nearest neighbor of the query q, in that $\forall p' \in P, d(p,q) \leq (1+\epsilon)d(p',q)$.

2 Main results

The current paper proposes three algorithms for the ϵ-NN problem, which work with different conditions on ϵ and different norms l_p. They are summarized in the following table.

Conditions	Preprocessing/Space	Query
$\epsilon > 1$, $p \in [1,2]$	$\tilde{O}(n^{1+1/\epsilon} + dn)$	$\tilde{O}(dn^{1/\epsilon})$
$0 < \epsilon < 1$, any p	$\tilde{O}(n) \times \tilde{O}(1/\epsilon)^d$	$\tilde{O}(d)$
$\epsilon > 0$,$p \in [1,2]$	$(nd)^{O(1)}$	$\tilde{O}(d)$

Our summary only covers proposition 1 in detail.

In this report we use P to denote the set of all points that we search in, $n = |P|$, $B(p,r)$ for the ball of radius r centered at point p, $|B(p,r)|$ for the number of points in $B(p,r)$, $d(p,q)$ for the distance between two points p and q, and $\Delta(S)$ for the maximum distance between any two points in some set S.

3 Algorithm outline and proofs

ϵ-NNS is solved by first reducing to approximate point location in equal balls (ϵ-PLEB), using the *ring-cover tree* data structure. ϵ-PLEB is then solved by using either bucketing method or locality-sensitive hashing (LSH). Combined with this reduction, LSH-based solution for ϵ-PLEB in Hamming metric gives rise to proposition 1, while bucketing-based solution and the application of the dimensional reduction technique in addition to bucketing give rise to proposition 2 and 3, respectively.

3.1 Reduction of ϵ-NNS to ϵ-PLEB

This reduction is based on the idea that in any point set P one can find either a *ring-separator* or a *cover*. Either construct allows decomposing P into strictly smaller sets $S_1, \ldots, S_l$ such that $|S_i| \leq c|P|$ for some $c < 1$ and $\sum_i |S_i| \leq b|P|$ for $b < 1 + 1/\log^2 n$. Therefore while searching in P one can restrict the search to one of the S_i.

Ring separator and cluster (***Definition 4,5*** and ***Theorem 1***) Consider two balls centered at a point p: $B(p,r)$ and $B(p, 2(1+1/\epsilon)r)$. They partition space into three non-overlapping regions: $B(p,r)$, the ring between the two balls, and the rest of space $P \setminus B(p, 2(1+1/\epsilon)r)$. If the first and the last regions each has at least $\alpha|P|$ points in P, P is said to contain a ring-separator $R(p, r, 2(1+1/\epsilon)r)$, else it contains a cluster S with $|S| \geq (1 - 2\alpha|P|)$ (S consists of the points in the ring between the two balls). The idea of a ring-separator is that if $q \in B(p, (1+1/\epsilon)r)$ then q cannot be much closer to any other point in $P \setminus B(p, 2(1+1/\epsilon)r)$, hence the search can be restricted to $B(p, 2(1+1/\epsilon)r)$. Otherwise if $q \notin B(p, (1+1/\epsilon)r)$ then p is no worse than any other point in $B(p,r)$ as a candidate for an ϵ-NN of q, so the search can be restricted to $P \setminus B(p,r)$.

Cover (***Definition 6*** and ***Theorem 2***) From a set A_i containing only one point p, we can add to A_i all the points in $B(p,r)$, and keep repeating the process for all the newly added points, while making sure $\frac{1}{b}|\bigcup_{p \in A_i} B(p,r)| \leq |A_i| \leq c|P|$. If we keep growing these "connected components" A_i's for all the points in a cluster S while maintaining $r \geq d\Delta(A)$ for $A = \bigcup_i A_i$, the resulting A_i's constitute a *cover* for S.

The main result that the reduction is based on is ***Corollary*** 1, which states that for any P, $0 < \alpha < 1$, $\beta > 1$, $b > 1$, P contains either an (α, α, β)-ring separator $R(p, r, \beta r)$, or a $(b, \alpha, \frac{1}{(2\beta+1)\log_b n})$-cover. A cover, like a ring separator, helps reducing the search space. The idea is that one can define three values $r_k < r < r_0$ such that $r = \frac{\gamma\Delta(A)}{\log_b n}$, $r_0 = (1 + \frac{1}{\epsilon})\Delta(A)$ and $r_k = \frac{r}{1+\epsilon}$ so that one of the following three cases happen.

Searching in a cover *Case 1*: if $q \notin B(a, r_0) \forall a \in A$ (this can be answered by invoking PLEB), then all the points in A have relatively similar distances to q and we can pick any point in A as an ϵ-NN candidate then restrict the search to $P - A$. *Case 2*: if $\exists a \in A_i$ such that $q \in B(a, r_k)q$ (again, using PLEB), any point outside of $\bigcup_{p \in A_i} B(p,r)$ cannot be much closer to q than a is, so the search can be restricted to $S_i = \bigcup_{p \in A_i} B(p,r)$. *Case 3*: if $\exists a \in A$ such that $q \in B(a, r_0)$ (using PLEB) but $q \notin B(a', r_k) \forall a' \in A$ (using PLEB), then we have to search in both A and $P - A$. The search in $P - A$ is done recursively, while A is searched as follows. Generate a sequence of shrinking radii $r_i = r_0/(1+\epsilon)^i$ and for each r_i generate an instance of PLEB. Use binary search to find the smallest r_i for which $\exists p \in A$ such that $q \in B(p, r_i)$. p is thus an ϵ-NN for q in A.

Ring-cover tree Given any point set P and an $\epsilon > 0$, $\beta = 2(1 + \frac{1}{e})$, $b = \frac{1}{\log^2 n}$ and $\alpha = \frac{1 - 1/\log n}{2}$, we can build a ring-cover tree recursively, where each node is either a ring node or a cover node according to **Corollary** 1. The paper provides gives a proof for the claim that the search procedure discussed above produces an ϵ-NN for q. We have, however, found holes in this proof, which we discuss in Section 4.

Space complexity ***Lemma 4*** shows that a ring-cover tree requires space at most $O(npolylogn)$. This leads to the following claim (***Corollary 2***): given an algorithm for PLEB which uses $f(n)$ space on an instance of size n where $f(n)$ is convex, a ring-cover tree for an n-point set P requires total space $O(f(npoly\log n))$. This is also the link between the ϵ-NN to PLEB reduction and the PLEB subroutine, needed to prove the space and time complexity of the three algorithms that the paper introduces.

3.2 Solving ϵ-PLEB with locality-sensitive hashing

A hash family $\mathcal{H} = \{h : S \to U\}$ is (r_1, r_2, p_1, p_2)-sensitive if for any 2 points $u, v \in S$

- if $d(u, v) \leq r_1$ then $\Pr_{\mathcal{H}}[h(u) = h(v)] \geq p_1$
- if $d(u, v) \leq r_2$ then $\Pr_{\mathcal{H}}[h(u) = h(v)] \leq p_2$

The LSH-based solution to ϵ-PLEB involves 2 parts: during preprocessing, store the points in P into buckets; upon querying with q, answer the query by searching the appropriate buckets.

Algorithm:

```
proc preprocessing(P):
  for each p ∈ P:
    for j = 1 to ℓ
      draw k random hash function h ∈ H
      g_j(p) = (h_1(p), h_2(p), ..., h_k(p))
      put p into bucket g_j(p)

proc isNearNeighbor(q,P,r):
  r_2 ← (1 + ε)r
  for i=1...ℓ:
    bucket_to_search ← g_i(q)
    for p_j ∈ the bucket above
      if p_j ∈ B(q, r_2) return p_j
  return null
```

The following claims are critical in analyzing this algorithm.

Theorem 1. *If there exists a hash family $\mathcal{H}$ that is (r_1, r_2, p_1, p_2)-sensitive, then there exists an algorithm for ϵ-PLEB ($r_1 = r, r_2 = (1 + \epsilon)r$) which uses $O(dn + n^{1+\rho})$ space and $O(n^{\rho})$ evaluations of hash function for each query.*

Proof sketch: The algorithm above would give correct answer with high probability, if the two events below happen with high probability

(1) Collision of hash on q and its near neighbors: $E_1 \equiv g_j(p^*) = g_j(q)$ if $p^* \in B(q, r_1)$

(2) No-collision of hash on q and any of the point farther than r_2 from q: $E_2 \equiv g_j(p') \neq g_j(q)$ for all $p' \in P - B_{r_2}(q)$, $\forall j = 1 \dots \ell$

By locality-sensitivity of $\mathcal{H}$, one can choose k, ℓ to ensure the above events happen with high probability. In particular, the authors chose $k = \log_{\frac{p1}{p2}} 2n$, $\ell = n^{\rho}$, in which $\rho = -\frac{\ln p_1}{\ln p_1/p_2}$.

Proposition 1. *Let $D(p, q)$ be the Hamming metric for $p, q \in \Sigma^d$, where Σ is a finite alphabet. Then for any $r, \epsilon > 0$, the family $\mathcal{H}$ of d hash functions, each mapping the point $(p_1, p_2, \dots, p_d)$ in d-dimension to the coordinate in the corresponding dimension, i.e.*

$$\mathcal{H} = \{h_i : h_i((p_1, p_2 \dots p_d)) = p_i, i = 1, \dots d\}$$

is $(r, (1 + \epsilon)r, 1 - \frac{r}{d}, 1 - \frac{r(1+\epsilon)}{d})$-sensitive.

Using the hash family proposed by Proposition 1 and applying Theorem 1, we have an algorithm for solving ϵ-PLEB using $O(dn + n^{1/\epsilon})$ space and $O(n^{1/\epsilon})$, as claimed and proved in ***Corollary 4***.

3.3 Solution to ϵ-NNS

Given the cost of reduction stated by ***Corollary 2***, and the cost of solving the reduced problem ϵ-PLEB as stated above, we obtain an algorithm for ϵ-NNS that uses $\tilde{O}(dn + n^{1/\epsilon})$ space and $\tilde{O}(n^{1/\epsilon})$ query time, which is the main result stated in ***Proposition 1***.

4 Discussion

We found that a more accurate analysis of the algorithm is available in a later version of this paper [1]. The cost of LSH-based solution to ϵ-PLEB also differ slightly depending on arbitrary choices in the algorithm, such as the number of points to explore before declaring no neighbor is found. To account for different choices of k and ℓ in the LSH-based ϵ-NNS, the costs of LSH-based solution to ϵ-PLEB can be re-written in terms of k and ℓ as following

- Preprocessing: $O(n\ell k)$ hash function evaluations
- Space: at most $O(\ell n + dn)$ of storage is required
- Query: at most $O(\ell k + \ell t D)$, with t being the average number of points per bucket, and D is the time for each distance calculation.

We have found the reduction arguments presented in this paper to have problems. In particular, in the ***Constructing Ring-Cover Trees*** section the paper assumes $r = \frac{\gamma\Delta(A)}{\log_b n}$, while r was shown to be $r = \frac{\gamma\Delta(S)}{\log_b n}$ in previous sections. This is not merely a change of notation because letting A play the role of S leads to other irreconcilable problems. Moreover, part (2c) of the proof of ***Lemma 1*** is unclear: we have been able to prove the same result (i.e., for any $p \in P - S_i$, $d(q, a) \leq \frac{d(p,q)}{1+\epsilon}$), but only for $\epsilon \geq 1$.

References

[1] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of computing*, 8(1):321–350, 2012.

[2] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.