Sparse Randomized Partition Trees for Nearest Neighbor Search

Kaushik Sinha
Wichita State University
Wichita, KS, USA
kaushik.sinha@wichita.edu

Omid Keivani
Wichita State University
Wichita, KS, USA
oxkeivani@shockers.wichita.edu

Abstract

Randomized partition trees have recently been shown to be very effective in solving nearest neighbor search problem. In spite of enjoying strong theoretical guarantee, it suffers from high space complexity, since each internal node of the tree needs to store a d dimensional projection direction leading to a $O(nd)$ space complexity for a dataset of size n. Inspired by the fast Johnson-Lindenstrauss transform, in this paper, we propose a sparse version of randomized partition tree where each internal node needs to store only a few non-zero entries, as opposed to all d entries, leading to significant space savings without sacrificing much in terms of nearest neighbor search accuracy. As a by product of this, query time of our proposed method is slightly better than that of its non-sparse counterpart for large dataset size. Our theoretical results indicate that our proposed method enjoys the same theoretical guarantee as that of the original non-sparse RP-tree. Experimental evaluations on four real world dataset strongly suggest that nearest neighbor search performance of our proposed sparse RP-tree is very similar to that of its non-sparse counterpart in terms of accuracy and number of retrieved points.

1 Introduction

Due to its wide variety of applications, nearest neighbor search is an extremely important problem in the field of machine learning in particular, and computer science in general. The basic problem is as follows: given a set of n d-dimensional data points $\mathcal{S} = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ and a query point $q \in \mathbb{R}^d$, one needs to build a data structure using $\mathcal{S}$, so that nearest point (when measured using appropriate distance metric) to q from $\mathcal{S}$ can be found quickly. The naive linear time solution that requires to scan through each data point $x_i \in \mathcal{S}$ often becomes impractical for large n and d. Towards this end, in recent years there has been a conscious effort towards designing sub-linear time algorithms for solving this problem. Most of these efforts can broadly be classified into two groups, namely, (a) tree based methods ([19, 5, 12, 15, 13, 4, 20, 7, 8]) and (b) methods based on hashing ([10, 2, 9, 16, 17]). Basic principle for both these approaches is to quickly retrieve a smaller subset $\mathcal{S}' \subset \mathcal{S}$ and perform linear scan within this retrieved $\mathcal{S}'$. Locality sensitive hashing (LSH) [10, 2, 9] is a representative of these hashing based methods that provides a sub-linear time solution for nearest neighbor search. Unfortunately, the LSH parameters (hash code length and number of hash tables) do not allow the user to have fine-grained control over the accuracy-efficiency tradeoff (for example, specifying a particular hash code length and number of hash tables does not provide any information/control on the number of retrieved points). To address these issues, recently a tree based method, namely randomized partition tree (RP-tree) has been proposed for nearest neighbor search [7, 8]. RP-tree uses a random binary tree data structure that exploits random projection at the non-leaf nodes. Unlike LSH, nearest neighbor search using RP-tree allows the user to have fine-grained control over the accuracy-efficiency tradeoff, i.e., the user just needs to set leaf node size n_0 and the number of trees L and the maximum number of retrieved points is upper bounded by $L * n_0$ (unlike LSH). More importantly, RP-tree enjoys strong theoretical guarantee by bounding the failure probability (that RP-tree fails to find exact nearest neighbor of a query) to arbitrarily small constant when dataset exhibits certain property, such as doubling measure or doubling dimension [8]. Also, on many real world datasets, empirical performance of RP-tree in solving nearest neighbor search has been reported to be superior as compared to that of LSH [18].

In spite of enjoying strong theoretical guarantee and superior empirical performance, main drawback of RP-tree

Appearing in Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54. Copyright 2017 by the authors.

data structure is its high storage overhead. An RP-tree has[1] $O(n)$ non-leaf nodes, each of which needs to store a d-dimensional projection direction, leading to $O(nd)$ storage per tree. Inspired by the recent seminal work on fast Johnson-Lindenstrauss transform [1] and its application to fast LSH [6], in this paper we propose a sparse version of RP-tree where each non-leaf node only needs to store many fewer entries than d without sacrificing much in terms of nearest neighbor search accuracy. As a by product, this results in faster query time as compared to non-sparse RP-tree. The theoretical analysis of RP-tree relies heavily on a data-dependent quantity called "potential function" [7, 8], which is used to bound the failure probability of nearest neighbor search using non-sparse RP-tree. Potential function essentially captures the inherent difficulty of nearest neighbor search and depends on relative position of the query points and the data points. We show that our proposed method yields a failure probability that is same as that of non-sparse RP-tree, except a small additional additive and multiplicative factor. These additive and multiplicative factors depend on user controllable parameters (ϵ and δ), which in turn controls how much sparsity our proposed method can handle. Consequently, we can reuse the proof technique developed in [7, 8] by simply plugging in failure probability estimate of our proposed method and thereby ensuring the same theoretical guarantee of the non-sparse RP-tree. We make the following contributions in this paper:

- We show that our proposed sparse RP-tree has space complexity $\Theta\left(\frac{n\log\left(\frac{nd}{\delta}\right)\log(1/\delta)}{\epsilon^2}\right)$ as opposed to $\Theta(nd)$ for non-sparse RP-tree, where $\epsilon \in (0,1)$ and $\delta \in (0,1)$ are error and confidence parameters defined in section 3. For example, by setting $\epsilon = \Theta\left(\sqrt{\frac{\log\left(\frac{nd}{\delta}\right)\log\left(\frac{1}{\delta}\right)}{d^\rho}}\right)$ for some $\rho, 0 < \rho < 1$, sparse RP-tree achieves $O(nd^\rho)$ space complexity.

- We prove that, for any given query point, at any internal node of our proposed sparse RP-tree, the expected fraction of non-nearest neighbor points that fall between the query point and its nearest neighbor upon projection is very similar to its non-sparse RP-tree counterpart, except an additional small (user controllable) multiplicative and additive term. Analysis of the nearest neighbor search failure probability of original RP-tree relies heavily on the above expected fraction. This indicates that all theoretical guarantees of non-sparse RP-tree for nearest neighbor search essentially hold for our proposed sparse version. More importantly, by setting $\epsilon = \Theta\left(\sqrt{\frac{\log\left(\frac{nd}{\delta}\right)\log\left(\frac{1}{\delta}\right)}{d^\rho}}\right)$ as above, the additional multiplicative term $(1+\epsilon)$ tends to 1 as for large d.

- We present an empirical evaluation of our proposed method on four real world datasets and show that nearest neighbor search performance of our proposed method is very similar to that of the non-sparse version, in terms of accuracy and number of retrieved points.

Rest of the paper is organized as follows. In section 2, we provide an overview of non-sparse RP-tree and introduce our proposed sparse version. In section 3, we provide theoretical analysis of our proposed method. We present our experimental evaluations in section 4 and conclude in section 5.

2 Sparse and non-sparse Randomized Partition Tree

2.1 Overview of non-sparse RP-tree

A randomized partition tree is a space partitioning binary tree data structure, whose root node contains the whole space or a subset of the space of interest (the complete dataset of n objects). The tree is constructed recursively from the root node by splitting each non-leaf node in a randomized fashion to create left and right child nodes, until each leaf node of the final tree contains at most a pre-specified number of points, say n_0. Splitting at any non-leaf node is performed by first projecting all data points belonging to that node onto a random projection direction and then choosing random split point that creates left and right child nodes based on the split point and projected data points. In order to answer a nearest neighbor query using this data structure, the query point is routed to a particular leaf node, following the path from root to leaf using the same splitting rule, and its nearest neighbor within that leaf node is returned. By construction, RP-tree is balanced and require[2] $O\left(d(n_0 + \log(n/n_0))\right)$ time to answer a query which is $O(d\log n)$ for constant n_0.

Due to its random nature, at any internal node, there is a positive probability that query point and its nearest neighbor might end up in different subtrees, leading to failure of nearest neighbor search. It was shown in [7, 8] that such probability can be bounded by a geometric quantity called "potential function", where, at any internal node containing

[1]Note that depending on data-structure construction and type of query processing, randomized partition trees can be of several types, such as random projection trees, spill trees and virtual spill trees [7, 8]. In this paper, we consider random projection trees.

[2]$O(d\log(n/n_0))$ time is required to traverse the tree to reach to a leaf node containing the query point. Once in this leaf node, nearest neighbor of q is obtained by searching from the n_0 data points located at this leaf node.

m points and q, potential function is defined as,

$$\Phi_m(q,S) = \frac{1}{m}\sum_{i=2}^{m}\left(\frac{\|x_{(1)}-q\|_2}{\|x_{(i)}-q\|_2}\right) \qquad (1)$$

In the above definition, $x_{(1)}, x_{(2)}, \ldots$ denotes an ordering of the x_i by increasing distance from q. It is easy to see that $\Phi_m(q,S) \in [0,1]$. The idea presented in [7, 8] was to first estimate the expected fraction of non-nearest neighbor points that fall in between query point and its nearest neighbor upon projection (this quantity is bounded from above by potential function) and then use Markov's inequality to bound the above probability in terms of potential function, specifically by the quantity $\Phi_m(q,S)\log\left(\frac{2}{\Phi_m(q,S)}\right)$. Finally, the failure probability the RP-tree is estimated by taking a union bound of the above probabilities along the path from root node to appropriate leaf node. Note that each internal node of RPT needs to store a pair consisting of a d dimensional projection vector and a split point. Space required to store these projections directions is $\sum_{i=0}^{\ln(n/n_0)} 2^i \cdot d = O(dn)$ for constant n_0. Moreover, if L independent such RPTs are constructed, total memory requirement for storing all the projection directions will be $O(Ldn)$.

2.2 Sparse RP-tree

To reduce the $O(d)$ space complexity at each internal node of an RP-tree, we propose a sparse RP-tree where, at each internal node, the random projection direction $U \in \mathbb{R}^d$ is made sparse by pre-multiplying U with random $d\times d$ diagonal matrix B, whose entries are drawn i.i.d from a Bernoulli distribution with success probability p. It is easy to see that for small p, only few entries of this new projection direction BU is non-zero, and these are the entries that need to be stored at each internal node. However, this poses a potential problem because if the entries of data points x_i and query q that corresponds to the nonzero indices of BU are zero, then $(BU)^\top x_i = (BU)^\top q = 0$. Inspired by [1], we solve this problem by densifying x_i and q with an application of norm preserving random rotation using a Walsh-Hadamard matrix and a random diagonal matrix. In particular, let H be a $d\times d$ Walsh-Hadamard matrix whose entries are given by $H_{ij} = d^{-1/2}(-1)^{\langle i-1, j-1\rangle}$, where $\langle i-1, j-1\rangle$ is the dot product (modulo 2) of the vectors i, j expressed in binary. Also, let D be a $d\times d$ diagonal matrix whose entries are drawn independently from $\{-1,1\}$ with success probability 1/2. It is easy to see that $\|HDx_i\| = \|x_i\|$, $\|HDq\| = \|q\|$ and $\|HD(x-q)\| = \|x-q\|$. This simple modification leads to our sparse RP-tree which is shown in Algorithm 1 and 2.

Note that while answering query q, we first need to apply the same transformation and find nearest neighbors of HDq. As we will show in the next section, for any fixed $\epsilon, \delta \in (0,1)$, setting $p = \min\left\{1, \Theta\left(\frac{(1+\epsilon)\log\left(\frac{nd}{\delta}\right)\log(1/\delta)}{\epsilon^2 d}\right)\right\}$ leads to expected fraction of non-nearest neighbors of q that falls between q and its nearest neighbor to be same as that of non-sparse RP-tree except an additional multiplicative factor $(1+\epsilon)$ and an additive factor $(\delta+\eta(\epsilon))$, where $\eta(\epsilon)$ is an increasing function of ϵ defined in Corollary 2. This reduces the space complexity of sparse RP-tree to $\Theta\left(\frac{n\log\left(\frac{nd}{\delta}\right)\log(1/\delta)}{\epsilon^2}\right)$ as compared to $\Theta(nd)$ in case of non-sparse RP-tree. Note also that by property of Walsh-Hadamard matrix, any matrix vector multiplication involving $d\times d$ Walsh-Hadamard matrix can be computed in $O(d\log d)$ time. As a by product of this, query time of our proposed sparse RP-tree becomes $O\left(d\log d + \frac{\log n \log\left(\frac{nd}{\delta}\right)\log(1/\delta)}{\epsilon^2}\right)$. For large n (compared to d), this query time can be potentially much faster as compared to that of its non-sparse version (see Corollary 3 for details).

Algorithm 1 Sparse RP-tree

Input : data $\mathcal{S} = \{x_1,\ldots,x_n\} \subset \mathbb{R}^d$, maximum number of data points in leaf node n_0
Preprocessing : Pre-multiply each $x_i \in \mathcal{S}$ and let $S = \{HDx_i : x_i \in \mathcal{S}\}$.
Output : tree data structure
function MakeTree(S, n_0)

```
1: if $|S| \le n_0$ then
2:     return leaf containing $S$
3: else
4:     Rule = ChooseRule$(S)$
5:     LeftTree = MakeTree$(\{x \in S : \text{Rule = true}\}, n_0)$
6:     RightTree = MakeTree$(\{x \in S : \text{Rule = false}\}, n_0)$
7:     return (Rule, LeftTree, RightTree)
8: end if
```

Algorithm 2 Function ChooseRule for sparse RP-tree

Input : data S
Output : rule
function ChooseRule(S)

```
1: Pick $U$ uniformly at random from the unit sphere by choosing each of its coordinate independently at random from a standard Normal distribution
2: Pick a diagonal matrix $B$ whose entries are drawn independently from a Bernoulli distribution with success probability $p$.
3: Pick $\beta$ uniformly at random from $[1/4, 3/4]$
4: Let $v$ be the $\beta$-fractile point on the projection of $S$ onto $BU$
5: Rule$(x) = (x^\top BU \le v)$
6: return (Rule)
```

3 Analysis of sparse RP-tree for nearest neighbor search

In this section we present theoretical analysis of our proposed sparse RP-tree for nearest neighbor search. Since structurally, sparse and non-sparse versions of RP-tree are very similar except the sparse random projection direction, if we can get an estimate of the expected fraction of non-nearest neighbors that fall between query q and its nearest neighbor upon projection at any internal node, we can essentially reuse the proof technique developed in [7, 8] to bound the failure probability of nearest neighbor search, simply by plugging in the corresponding estimate for sparse version of RP-tree. What we will show in this section is that the above estimate for sparse RP-tree is very similar to that of non-sparse RP-tree except an small additional multiplicative as well as small additive term. More importantly, these additional terms are user controllable and can be made as small as one wants at the expense of how much sparsity our proposed method can handle for a fixed d. Before we present the actual proof, we provide a high level proof sketch. Due to space limitation we defer proofs of various auxiliary lemmas to the supplementary material.

3.1 Proof sketch

Crux of the analysis is to solve the following problem: given any $x, y, q \in \mathbb{R}^d$ with $\|q - x\| \leq \|q - y\|$, what is the probability that upon projection onto a random direction U, $U^\top y$ falls strictly between $U^\top q$ and $U^\top x$, which is equivalent to asking what is the probability that $U^\top(y - q)$ falls strictly between 0 and $U^\top(x - q)$. In [7, 8], without loss of generality, this problem is solved by assuming that $x = \|x\|e_1 = (\|x\|, 0, \ldots, 0)$ and simplifying the proof by taking advantage of that assumption. In our proposed method we can not make this assumption since we are densifying the query and data points by applying Walsh-Hadamard transform. Additionally, in our case projection direction is not U but BU, where B is a $d \times d$ diagonal matrix whose entries are drawn independently from a Bernoulli distribution. Letting $x_B = (BU)^\top HDx, y_B = (BU)^\top HDy, q_B = (BU)^\top HDq$ and $X_1 = (BU)^\top HD(x - q), X_2 = (BU)^\top HD(y - q)$, we observe the conditioned on B, $(X_1, X_2)^\top$ follows a bivariate normal distribution with zero mean and covariance matrix $C_B = \begin{pmatrix} \|x_B - q_B\|^2 & (x_B - q_B)^\top(y_B - q_B) \\ (x_B - q_B)^\top(y_B - q_B) & \|y_B - q_B\|^2 \end{pmatrix}$. Using this observation we develop a new proof technique to find the probability that X_2 fall strictly between 0 and X_1 in Lemma 4. Note that Lemma 4 can be applied to non-sparse version of RP-tree and we can recover Lemma 1 of [7]. Note that in non-sparse case (where H, B, D are identity matrix) $(X_1, X_2)^\top$ follows a bivariate normal distribution with zero mean and covariance matrix $\begin{pmatrix} \|x - q\|^2 & (x - q)^\top(y - q) \\ (x - q)^\top(y - q) & \|y - q\|^2 \end{pmatrix}$ and due to assumption on x, y, q, second diagonal $\|y - q\|^2$ is at least as large as the first diagonal $\|x - q\|^2$. This makes the proof simpler. In sparse case however, due to random choice of B, $\|y_B - q_B\|^2$ may be even smaller than $\|x_B - q_B\|^2$ and this makes the proof of Lemma 4 more involved. Next, we observe that taking expectation with respect to B, $\mathbb{E}_B(C_B) = \begin{pmatrix} p\|x - q\|^2 & p(x - q)^\top(y - q) \\ p(x - q)^\top(y - q) & p\|y - q\|^2 \end{pmatrix}$. Moreover, we show in Lemma 6 that over random choice of B, entries of C_B are tightly concentrated[3] near their respective exceptions with high probability. This gives us the desired value of Bernoulli success probability p. Equipped with this, using Lemma 4, 6 and technical Lemma 5 we prove the main theorem (Theorem 1) for sparse RP-tree.

3.2 Proof of main result

Here we present the main theorem of this paper. Statement of all auxiliary lemmas are presented at the end of this section. Due to space limitation, their proofs are deferred to the supplementary material.

Theorem 1. *Let H be $d \times d$ Walsh-Hadamard matrix. Pick any $q, x, y \in \mathbb{R}^d$ with $\|q - x\| \leq \|q - y\|$. Pick any random $U \in \mathbb{R}^d$ whose entries are drawn i.i.d from a standard Normal distribution and a random diagonal matrix $D \in \mathbb{R}^{d \times d}$, whose entries are ± 1 and drawn independently and uniformly. Pick any $\epsilon, \delta \in (0, 1)$ and a random diagonal matrix $B \in \mathbb{R}^{d \times d}$ whose entries are drawn i.i.d from a Bernoulli distribution with success probability $p = \min\left\{1, \Theta\left(\frac{(1+\epsilon)\log\left(\frac{nd}{\delta}\right)\log(1/\delta)}{\epsilon^2 d}\right)\right\}$ and are independent from entries of U and B. Let $\mathcal{B}$ be the event, $\mathcal{B} \equiv \{U^\top BHDy$ falls between $U^\top BHDq$ and $U^\top BHDx\}$. Then the following holds.*

$$\Pr(\mathcal{B}) \leq \begin{cases} \frac{1}{2}(1 + \epsilon)\frac{\|q - x\|}{\|q - y\|} + \delta, & \textit{if } 1_{\mathcal{C}} > 0 \\ 1 & \textit{otherwise.} \end{cases}$$

where, the indicator function $1_{\mathcal{C}}$ is defined as,

$$1_{\mathcal{C}} = \begin{cases} 1, & \textit{if } (x - q)^\top(y - q) \leq (1 - 2\epsilon)\|q - x\|\|q - y\| \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. For ease of readability we use the following notation. Let $x_H = HDx, y_H = HDx, q_H = HDq$ and let $x_B = Bx_H, y_B = By_H, q_B = Bq_H$. Also, let $X_1 = U^\top BHD(x - q) = U^\top B(x_H - q_H) = U^\top(x_B - q_B)$ and $X_2 = U^\top BHD(y - q) = U^\top B(y_H - q_H) =$

[3]The diagonal terms are more tightly concentrated than the off-diagonal terms as norm is much better preserved than inner product in this case.

$U^\top(y_B - q_B)$. Event $\mathcal{B}$ can now be written as :

$$\begin{aligned}\mathcal{B} &\equiv \{U^\top BHDy \text{ falls between } U^\top BHDq \text{ and } U^\top BHDx\}\\ &\equiv \{U^\top BHD(y-q) \text{ falls between } 0 \text{ and } U^\top BHD(x-q)\}\\ &\equiv \{U^\top(y_B - q_B) \text{ falls between } 0 \text{ and } U^\top(x_B - q_B)\}\\ &\equiv \{X_2 \text{ falls between } 0 \text{ and } X_1\}\\ &\equiv \{0 < X_2 < X_1\} \cup \{X_1 < X_2 < 0\}\end{aligned}$$

Using Lemma 5, with probability at least $1-\frac{\delta}{2}$, we have $\|x_H - q_H\|_\infty = \|HD(x-q)\|_\infty \le \|x - q\|\sqrt{\frac{2\log(4nd/\delta)}{d}}$ and $\|y_H - q_H\|_\infty = \|HD(y-q)\|_\infty \le \|y-q\|\sqrt{\frac{2\log(4nd/\delta)}{d}}$. Also note that, $\sum_{i=1}^d B_{ii}^2((x_H)_i - (q_H)_i)^2 = (B(x_H - q_H))^\top B(x_H - q_H) = \|x_B - q_B\|^2$, and similarly, $\sum_{i=1}^d B_{ii}^2((x_H)_i - (q_H)_i)^2 = \|y_H - q_H\|^2$ and $\sum_{i=1}^d B_{ii}^2((x_H)_i - (q_H)_i)\cdot((y_H)_i - (q_H)_i) = (x_B - q_B)^\top(y_B - q_B)$. Using this observation and Lemma 6, it follows that $(X_1, X_2)^\top$ follows a bivariate normal distribution with zero mean and covariance matrix given by $C_B = \begin{pmatrix}\|x_B - q_B\|^2 & (x_B - q_B)^\top(y_B - q_B)\\ (x_B - q_B)^\top(y_B - q_B) & \|y_B - q_B\|^2\end{pmatrix}$, where with probability at least $1-\frac{\delta}{2}$, the following holds:

$$(1-\epsilon)p\|q-x\|^2 \le \|x_B - q_B\|^2 \le (1+\epsilon)p\|q-x\|^2 \quad (2)$$

$$(1-\epsilon)p\|q-y\|^2 \le \|y_B - q_B\|^2 \le (1+\epsilon)p\|q-y\|^2 \quad (3)$$

$$|(x_B - q_B)^\top(y_B - q_B) - p(q-x)^\top(q-y)| \le p\frac{\epsilon}{2}\Big(\|q-x\|^2 + \|q-y\|^2\Big) \quad (4)$$

Next we use this information and Lemma 4 to estimate $\Pr(\mathcal{B})$. We consider the following two cases for this purpose.

Case 1: $(q-x)^\top(q-y) \le (1-2\epsilon)\|q-x\|\|q-y\|$

We will show that if $(q-x)^\top(q-y) \le (1-2\epsilon)\|q-x\|\|q-y\|$ then $\|y_B - q_B\|^2 \ge (x_B - q_B)^\top(y_B - q_B)$ and we can use the first case of Lemma 4. To see this suppose the condition $(q-x)^\top(q-y) \le t\|q-x\|\|q-y\|$ holds for some positive t. Then using equation 4 we can write,

$$\begin{aligned}&(x_B - q_B)^\top(y_B - q_B)\\ \le\ & p\left((q-x)^\top(q-y) + \frac{\epsilon}{2}(\|q-x\|^2 + \|q-y\|^2)\right)\\ \le\ & p\left(t\|q-x\|\|q-y\| + \frac{\epsilon}{2}(\|q-x\|^2 + \|q-y\|^2)\right)\\ \le\ & p\left(t\|q-y\|^2 + \frac{\epsilon}{2}(\|q-y\|^2 + \|q-y\|^2)\right)\\ =\ & p\|q-y\|^2(t+\epsilon)\end{aligned}$$

Therefore, the maximum value of $(x_B - q_B)^\top(y_B - q_B)$ can be at most $p\|q-y\|^2(t+\epsilon)$. Now using equation 3 it is easy to see that $\|y_B - q_B\|^2$ can be at least $p\|q-y\|^2(1-\epsilon)$. Therefore, $\|y_B - q_B\|^2 \ge (x_B - q_B)^\top(y_B - q_B)$ if, $p\|q-y\|^2(1-\epsilon) \ge p\|q-y\|^2(t+\epsilon) \Rightarrow t \le (1-2\epsilon)$. Using Lemma 4, we see

$$\begin{aligned}\Pr(\mathcal{B}) &= \frac{1}{\pi}\arcsin\left(\frac{\|x_B - q_B\|}{\|y_B - q_B\|}\right.\\ &\qquad \left.\times\sqrt{1-\left(\frac{(x_B - q_B)^\top(x_B - y_B)}{\|x_B - q_B\|\|x_B - y_B\|}\right)^2}\right)\\ &\le \frac{1}{\pi}\arcsin\left(\frac{\|x_B - q_B\|}{\|y_B - q_B\|}\right)\\ &\le \frac{1}{2}\left(\frac{\|x_B - q_B\|}{\|y_B - q_B\|}\right) \le \frac{1}{2}\left(\frac{\|x-q\|\sqrt{(1+\epsilon)}}{\|y-q\|\sqrt{(1-\epsilon)}}\right)\\ &\le \frac{1}{2}(1+2\epsilon)\left(\frac{\|x-q\|}{\|y-q\|}\right)\end{aligned}$$

Case 2: $(q-x)^\top(q-y) > (1-2\epsilon)\|q-x\|\|q-y\|$

Note that in this case we can have $\|y_B - q_B\|^2 < (x_B - q_B)^\top(y_B - q_B)$. Since C_B is positive definite, its determinant is non-negative, i.e, $\|x_B - y_B\|^2\|y_B - q_B\|^2 \ge ((x_B - q_B)^\top(y_B - q_B))^2$. Combining these two facts it is easy to see that $\|x_B - q_B\|^2 > \|y_B - q_B\|^2$, or in other words, $\frac{\|x_B - q_B\|^2}{\|y_B - q_B\|^2} > 1$. Now using equation 2 and 3, we see that $\frac{\|x_B - q_B\|^2}{\|y_B - q_B\|^2} \le \frac{\|x-q\|^2(1+\epsilon)}{\|y-q\|^2(1-\epsilon)}$. Combining these two facts we see that $\left(1 \le \frac{\|x-q\|^2(1+\epsilon)}{\|y-q\|^2(1-\epsilon)}\right) \Rightarrow \left(\frac{\|y-q\|^2}{\|x-q\|^2} \le \frac{1+\epsilon}{1-\epsilon}\right)$. However, it is assumed that $\|x-q\| \le \|y-q\|$. Therefore, over random choice of B, $\|y_B - q_B\|^2 < (x_B - q_B)^\top(y_B - q_B)$, when the following events holds

$$1 \le \frac{\|y-q\|^2}{\|x-q\|^2} \le \frac{1+\epsilon}{1-\epsilon}$$

and

$$(q-x)^\top(q-y) > (1-2\epsilon)\|q-x\|\|q-y\|$$

This corresponds to the shaded region in Figure 1. Note that for fix q and x, whenever y falls in this shaded region $\Pr(\mathcal{B})$ can be close to 1 in the worst case. However, by choosing small ϵ we can control the volume of this shaded region and make it as small as we want. □

The following corollary follows immediately from Theorem 1.

Corollary 2. *Let H be $d\times d$ Walsh-Hadamard matrix. Pick any random $U \in \mathbb{R}^d$ whose entries are drawn i.i.d from a standard Normal distribution and a random diagonal matrix $D \in \mathbb{R}^{d\times d}$, whose entries are ± 1 and drawn independently and uniformly. Pick any $\epsilon, \delta \in (0,1)$ and a random diagonal matrix $B \in \mathbb{R}^{d\times d}$ whose entries are drawn i.i.d from a Bernoulli distribution with success probability $p = \min\left\{1, \Theta\left(\frac{(1+\epsilon)\log\left(\frac{nd}{\delta}\right)\log(1/\delta)}{\epsilon^2 d}\right)\right\}$ and are independent from entries of U and B. Pick any $q, x_1, \ldots, x_n \in \mathbb{R}^d$. If these points are projected to BU then the expected fraction of the projected x_i that fall between q and $x_{(1)}$ is at*

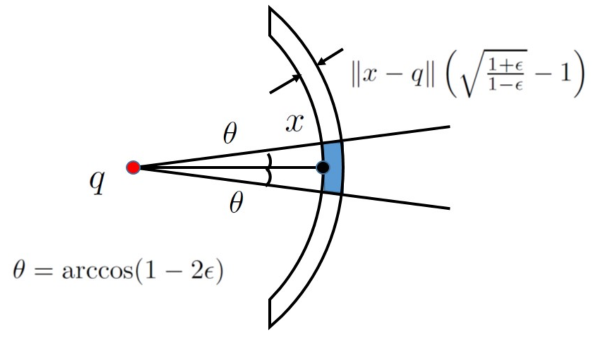
$\|x-q\|\left(\sqrt{\frac{1+\epsilon}{1-\epsilon}}-1\right)$
x
θ
q
θ
$\theta = \arccos(1-2\epsilon)$

Figure 1: Any point $y \in \mathbb{R}^d$ that satisfy $\|x-q\| \le \|y-q\| \le \|x-q\|\left(\sqrt{\frac{1+\epsilon}{1-\epsilon}}-1\right)$ and $(q-x)^\top(q-y) > (1-2\epsilon)\|q-x\|\|q-y\|$ lies in the shaded region.

most $\frac{1}{2}(1+\epsilon)\Phi_n(q,\{x_1,\dots,x_n\})+\delta+\eta(\epsilon)$*, where* $\eta(\epsilon)$ *is the fraction of the points* x_i *that satisfy* $\|x_{(1)}-q\| \le \|x_i-q\| \le \|x_{(1)}-q\|\left(\sqrt{\frac{1+\epsilon}{1-\epsilon}}-1\right)$ *and* $(q-x_{(1)})^\top(q-x_i) > (1-2\epsilon)\|q-x_{(1)}\|\|q-x_i\|$.

Proof. Let $A_\epsilon = \left\{x_i : \|x_{(1)}-q\| \le \|x_i-q\| \le \|x_{(1)}-q\|\left(\sqrt{\frac{1+\epsilon}{1-\epsilon}}-1\right) \text{ and } (q-x_{(1)})^\top(q-x_i) > (1-2\epsilon)\|q-x_{(1)}\|\|q-x_i\|\right\}$. Note that $|A_\epsilon| = n\eta(\epsilon)$. Let Z_i be the indicator variable that that takes value 1 if $x_{(i)}$ falls between $x_{(1)}$ and q in projection and zero otherwise. Let $Z=\sum_{i=2}^n$. Using Theorem 1, it is easy to see that

$$\begin{aligned}
\mathbb{E}(Z) &= \sum_{i=2}^n \mathbb{E}(Z_i) = \sum_{i=2}^n \Pr(Z_i=1)\\
&= \sum_{x_{(i)}\in A_\epsilon} \Pr(Z_{(i)}=1) + \sum_{x_{(i)}\notin A_\epsilon} \Pr(Z_{(i)}=1)\\
&\le \sum_{x_{(i)}\in A_\epsilon} 1 + \sum_{x_{(i)}\notin A_\epsilon}\left(\frac{1}{2}(1+\epsilon)\frac{\|q-x_{(1)}\|}{\|q-x_{(i)}\|}+\delta\right)\\
&\le n\eta(\epsilon) + \sum_{i=2}^n\left(\frac{1}{2}(1+\epsilon)\frac{\|q-x_{(1)}\|}{\|q-x_{(i)}\|}+\delta\right)\\
&= n\eta(\epsilon) + \frac{1}{2}(1+\epsilon)\sum_{i=2}^n \frac{\|q-x_{(1)}\|}{\|q-x_{(i)}\|} + (n-1)\delta\\
&\le (1+\epsilon)n\Phi_n(q,\{x_1,\dots,x_n\}) + n\left(\eta(\epsilon)+\delta\right)
\end{aligned}$$

Therefore, the expected fraction of the points that fall between $x_{(1)}$ and q is at most $(1+\epsilon)\Phi_n(q,\{x_1,\dots,x_n\}) + (\epsilon(\eta)+\delta)$. ☐

Note that in case of original RP-tree, the expected fraction of non-neighbor points that fall between q and its nearest neighbor $x_{(1)}$ upon projection is $\frac{1}{2}\Phi_n(q,\{x_1,\dots,x_n\})$. It is easy to see that for our proposed sparse version, the additional multiplicative term $(1+\epsilon)$ and the additive term $(\eta(\epsilon)+\delta)$ can be made small. To see this, fix $\rho, 0<\rho<1$, and set $\epsilon = \Theta\left(\sqrt{\frac{\log\left(\frac{nd}{\delta}\right)\log\left(\frac{1}{\delta}\right)}{d^\rho}}\right)$. While on one hand this ensures the space complexity of sparse RP-tree to be $O(nd^\rho)$ as opposed to $O(nd)$ in case of original RP-tree, on the other hand, for fixed ρ, as d increases, ϵ decreases (in fact $\epsilon \to 0$ as $d \to \infty$), and consequently the additional multiplicative term $(1+\epsilon)$ approaches 1. In addition, as can be seen from Figure 1, volume of the shaded region tends to zero as $\epsilon \to 0$, therefore, the fraction of the data points that fall within this shaded region, $\eta(\epsilon)$, tends to zero as well. Finally, the confidence parameter δ can be chosen arbitrarily small as its effect is reflected in terms of $\log(1/\delta)$ in p.

Next, we show that for large d, query time of our proposed sparse RP-tree is smaller than that of the original RP-tree, for large dataset size.

Corollary 3. *Fix any* $\rho \in (0,1/2)$ *and* $\epsilon \in (0,1)$*. Choose* d *large enough so that space complexity of proposed sparse RP-tree is* $O(nd^\rho)$*. Then query time of our proposed sparse RP-tree is smaller than that of original RP-tree if* $n \ge d^2$.

Proof. Note that query time of RP-tree (or sparse RP-tree) data structure is the sum of, (a) time required to reach to a leaf node (from root node) and (b) time required to process the data points lying in that leaf node. By construction, maximum number of data points at leaf node of an RP-tree (or Sparse RP-tree) is at most n_0, and consequently second part of query time is $n_0 d$ in both cases. Now, in case of RP-tree, time required to reach a leaf node is $O(d\log n)$ as the depth of the tree is at most $O(\log n)$ and an inner product needs to be computed along the path from root node to a leaf node. Now in case of sparse RP-tree, Walsh Hadamard transform of a d-dimensional query point can be computed in $O(d\log d)$ time. Choose d large enough so that $\epsilon = \Theta\left(\sqrt{\frac{\log\left(\frac{nd}{\delta}\right)\log\left(\frac{1}{\delta}\right)}{d^\rho}}\right)$. This ensures that average number of non-zero coordinates of the projection direction stored at each internal node of sparse RP-tree is $pd = d^\rho$. Therefore, time required to reach to a leaf node (from root node), in case of sparse RP-tree is $O(d\log d + d^\rho \log n)$. Without considering the constants in asymptotic notation we would like to show that $d\log n \ge d\log d + d^\rho \log n$ when $n \ge d^2$. To achieve this, first we claim that $\frac{d}{d-d^\rho} \le 2$. To see this, note that, $\rho < 1/2 \Rightarrow d^\rho < \sqrt{d} \Rightarrow (d-\sqrt{d}) < (d-d^\rho) \Rightarrow \frac{d}{d-d^\rho} < \frac{d}{d-\sqrt{d}} \Rightarrow \frac{d}{d-d^\rho} < 2$, where the last implication follows from that fact that $\frac{d}{d-\sqrt{d}} \le 2$ for any $d \ge 4$. Now if $n \ge d^2$, that would imply $n \ge d^2 \ge d^{\frac{d}{d-d^\rho}}$. taking logarithm on both sides yield, $\log n \ge \frac{d}{d-d^\rho}\log d$. After cross multiplication and rearranging the terms it is easy to see that $d\log n \ge d\log d + d^\rho \log n$. ☐

The following lemmas are required to prove Theorem 1.

Due to space limitation, we defer their proofs to the supplementary material.

Lemma 4. *Pick any $q, x, y \in \mathbb{R}^d$. Pick any random $U = (U_1, \ldots, U_d)^\top \in \mathbb{R}^d$ whose elements are drawn i.i.d from a standard Normal distribution. Let $\mathcal{A}$ be the event $\mathcal{A} \equiv \{U^\top y$ falls (strictly) between $U^\top q$ and $U^\top x\}$. Then the following holds.*

(a) If $\|y-q\|^2 \geq (x-q)^\top(y-q)$, then,

$$\Pr(\mathcal{A}) = \frac{1}{\pi}\arcsin\left(\frac{\|x-q\|}{\|y-q\|}\sqrt{1-\left(\frac{(x-q)^\top(x-y)}{\|x-q\|\|x-y\|}\right)^2}\right)$$

(b) If $\|y-q\|^2 < (x-q)^\top(y-q)$, then,

$$\Pr(\mathcal{A}) = 1-\frac{1}{\pi}\arcsin\left(\frac{\|x-q\|}{\|y-q\|}\sqrt{1-\left(\frac{(x-q)^\top(x-y)}{\|x-q\|\|x-y\|}\right)^2}\right)$$

Lemma 5. *Let $\mathcal{S} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ be a set of n vectors in $\mathbb{R}^d$, H be a $d \times d$ deterministic Walsh-Hadamard matrix and D be a $d \times d$ diagonal matrix, where each D_{ii} is drawn independently from $\{-1, +1\}$ with probability 1/2. Then for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $x_i \in \mathcal{S}$: $\|HDx_i\|_\infty \leq \|x_i\|\sqrt{\frac{2\log(2nd/\delta)}{d}}$.*

Lemma 6. *Let $v_1, v_2 \in \mathbb{R}^d$ be any two vectors such that $\|v_1\|_\infty \leq \|v_1\|\sqrt{2\log\left(\frac{2nd}{\delta}\right)/d}$ and $\|v_2\|_\infty \leq \|v_2\|\sqrt{2\log\left(\frac{2nd}{\delta}\right)/d}$ and let $U = (U_1, \ldots, U_d)^\top \in \mathbb{R}^d$ be a random vector whose entries are drawn i.i.d from a standard Normal distribution. Also, for any $\epsilon, \delta \in (0,1)$, let B be a $d \times d$ diagonal matrix whose diagonal entries B_{ii}'s are drawn i.i.d from a Bernoulli distribution with success probability $p = \min\left(1, \frac{4(1+\epsilon/3)\log\left(\frac{2nd}{\delta}\right)\log(8/\delta)}{\epsilon^2 d}\right)$, and where diagonal entries of B are independent from entries of U. Let $Y_1 = U^\top Bv_1$ and $Y_2 = U^\top Bv_2$. Then the following holds:*

1. *$(Y_1, Y_2)^\top$ follows a bivariate normal distribution with zero mean and covariance matrix $C_B = \begin{pmatrix}\sum_{i=1}^d B_{ii}^2 v_{1i}^2 & \sum_{i=1}^d B_{ii}^2 v_{1i}v_{2i} \\ \sum_{i=1}^d B_{ii}^2 v_{1i}v_{2i} & \sum_{i=1}^d B_{ii}^2 v_{2i}^2\end{pmatrix}$, where C_B is random quantity.*
2. *$\mathbb{E}_B(C_B) = \begin{pmatrix} p\|v_1\|^2 & p(v_1^\top v_2) \\ p(v_1^\top v_2) & p\|v_2\|^2\end{pmatrix}$.*
3. *With probability at least $1-\frac{\delta}{2}$, $(1-\epsilon)p\|v_1\|^2 \leq \sum_{i=1}^d B_{ii}^2 v_{1i}^2 \leq (1+\epsilon)p\|v_1\|^2$ and $(1-\epsilon)p\|v_2\|^2 \leq \sum_{i=1}^d B_{ii}^2 v_{2i}^2 \leq (1+\epsilon)p\|v_2\|^2$.*
4. *With probability at least $1-\frac{\delta}{2}$, $p\left(v_1^\top v_2 - \frac{\epsilon}{2}(\|v_1\|^2+\|v_2\|^2)\right) \leq \sum_{i=1}^d B_{ii}^2 v_{1i}v_{2i} \leq p\left(v_1^\top v_2 + \frac{\epsilon}{2}(\|v_1\|^2+\|v_2\|^2)\right)$.*

4 Experimental Results

In this section we report empirical performance of our proposed sparse RP-tree data structure in solving nearest neighbor search problem. We used four real world datasets of varied dimensionality. Number of data points to construct our proposed data structure and number of queries are listed in Table 2. In our experiments, we randomly split each dataset D into two disjoint subsets S and Q, such that, $D = S \cup Q$, where all data points in S were used to construct the RP-trees (or their sparse versions) and these tree data structure was then used to find 10 nearest neighbors for each query q from Q. We evaluated our proposed method using number of trees (L) chosen from the set $\{8, 16, 32, 64, 128\}$. For each choice of L, we created L independent sparse RP-trees. Now given a query point q, we retrieved union of all the points corresponding to L leaf nodes of these trees (call this set R) and find the 10 nearest neighbors from R. We say that our method accurately finds 10 nearest neighbors only if our answer is same as the true 10 nearest neighbors obtained by performing a linear scan over the entire dataset. Averaging over all queries in Q, we report nearest neighbor accuracy and standard deviation of our proposed method. We also report average number of retrieved points (size of R). In all our experiments Q contains 5000 query points, except for USPS dataset for which Q contain 2298 data points. We set n_0 to be 100 for all our experiments.

4.1 Datasets

Details of the datasets used for our experimental evaluations are listed in table.2. The USPS dataset contains hand written digits. The AERIAL dataset contains texture information of large areal photographs [14]. The COREL dataset is available at the UCI repository [3]. After removing the missing data we keep only 50,000 instances. This SIFT dataset contains SIFT image descriptors, introduced in [11]. The original dataset contains 1 million image descriptors. We used 50,000 image descriptors from this dataset for our experiments. USPS had only 9298 data points. Therefore, we used 7000 for building the data structure and the remaining as query points.

4.2 Comparison of sparse and non-sparse RP-tree

To empirically demonstrate the effectiveness of our proposed method we used multiple p values, where p is Bernoulli success probability of choosing a non-zero coordinates of a projection direction at each internal node of an RP-tree. In particular, we used p values from the set $\{0.1, 0.3, 0.5, 0.7, 1.0\}$. Note that, we only need to store the non-zero coordinates of projection direction at internal node, since other coordinates do not contribute in computing an inner product (1-D projection). Thus, if we have m number of internal nodes in a tree, the non-sparse version

Aerial	$p = 1$		$p = 0.7$		$p = 0.5$		$p = 0.3$		$p = 0.1$	
$L = 8$	**503**	$\mathbf{0.716 \pm 0.21}$	505	0.709 ± 0.22	**503**	0.707 ± 0.22	508	0.711 ± 0.22	508	0.701 ± 0.22
$L = 16$	923	0.885 ± 0.14	925	0.884 ± 0.14	**921**	0.885 ± 0.14	925	$\mathbf{0.888 \pm 0.14}$	930	0.878 ± 0.15
$L = 32$	1617	$\mathbf{0.975 \pm 0.06}$	1620	0.974 ± 0.06	**1615**	0.974 ± 0.06	1619	$\mathbf{0.975 \pm 0.06}$	1626	0.974 ± 0.06
$L = 64$	2691	$\mathbf{0.998 \pm 0.01}$	2702	$\mathbf{0.998 \pm 0.01}$	**2686**	$\mathbf{0.998 \pm 0.01}$	2692	$\mathbf{0.998 \pm 0.02}$	2703	$\mathbf{0.998 \pm 0.01}$
$L = 128$	4243	$\mathbf{1 \pm 0.00}$	4246	$\mathbf{1 \pm 0.00}$	**4220**	$\mathbf{1 \pm 0.00}$	4239	0.999 ± 0.00	4261	0.999 ± 0.00
Corel	$1 - p = 0$		$1 - p = 0.3$		$1 - p = 0.5$		$1 - p = 0.7$		$1 - p = 0.9$	
$L = 8$	**510**	$\mathbf{0.757 \pm 0.18}$	**510**	0.750 ± 0.18	**510**	0.748 ± 0.18	513	0.752 ± 0.18	514	0.754 ± 0.18
$L = 16$	**936**	$\mathbf{0.925 \pm 0.10}$	**936**	0.923 ± 0.11	939	0.920 ± 0.11	939	0.921 ± 0.10	940	0.924 ± 0.10
$L = 32$	**1644**	$\mathbf{0.990 \pm 0.03}$	1648	$\mathbf{0.990 \pm 0.04}$	1652	0.989 ± 0.04	1651	0.988 ± 0.04	1647	0.989 ± 0.04
$L = 64$	**2742**	$\mathbf{1 \pm 0.01}$	2749	$\mathbf{1 \pm 0.01}$	2752	0.999 ± 0.01	2753	0.999 ± 0.01	**2742**	$\mathbf{1 \pm 0.01}$
$L = 128$	4324	$\mathbf{1 \pm 0.00}$	4347	$\mathbf{1 \pm 0.00}$	4342	$\mathbf{1} \pm 0.00$	4338	$\mathbf{1 \pm 0.00}$	**4318**	$\mathbf{1 \pm 0.00}$
SIFT	$1 - p = 0$		$1 - p = 0.3$		$1 - p = 0.5$		$1 - p = 0.7$		$1 - p = 0.9$	
$L = 8$	**546**	$\mathbf{0.444 \pm 0.24}$	**546**	0.441 ± 0.24	547	0.437 ± 0.24	547	0.434 ± 0.24	**546**	0.433 ± 0.24
$L = 16$	1062	$\mathbf{0.637 \pm 0.23}$	1058	0.633 ± 0.23	1057	0.635 ± 0.23	1061	0.631 ± 0.23	**1056**	0.625 ± 0.23
$L = 32$	2007	$\mathbf{0.824 \pm 0.17}$	**2003**	0.822 ± 0.17	**2003**	0.822 ± 0.17	2009	0.819 ± 0.17	2008	0.818 ± 0.17
$L = 64$	**3669**	$\mathbf{0.948} \pm 0.09$	3678	0.946 ± 0.09	3676	0.947 ± 0.09	3678	0.945 ± 0.09	3683	0.946 ± 0.09
$L = 128$	**6387**	$\mathbf{0.993 \pm 0.03}$	6400	$\mathbf{0.993 \pm 0.03}$	6400	$\mathbf{0.993 \pm 0.03}$	6399	$0.992 \pm \mathbf{0.03}$	6405	$\mathbf{0.993 \pm 0.03}$
USPS	$1 - p = 0$		$1 - p = 0.3$		$1 - p = 0.5$		$1 - p = 0.7$		$1 - p = 0.9$	
$L = 8$	**496**	0.740 ± 0.22	513	0.730 ± 0.22	497	0.733 ± 0.22	505	0.743 ± 0.22	507	$\mathbf{0.749 \pm 0.21}$
$L = 16$	907	0.907 ± 0.13	930	0.906 ± 0.13	**901**	0.904 ± 0.14	916	$\mathbf{0.909 \pm 0.13}$	923	0.906 ± 0.13
$L = 32$	**1567**	$\mathbf{0.981 \pm 0.05}$	1595	$\mathbf{0.981 \pm 0.06}$	1573	0.980 ± 0.06	1578	$\mathbf{0.981 \pm 0.05}$	1602	$\mathbf{0.981 \pm 0.05}$
$L = 64$	2541	$\mathbf{0.998 \pm 0.02}$	2587	$\mathbf{0.998 \pm 0.02}$	2553	$\mathbf{0.998 \pm 0.01}$	**2537**	$\mathbf{0.998 \pm 0.02}$	2575	$\mathbf{0.998 \pm 0.01}$
$L = 128$	3781	$\mathbf{1 \pm 0.00}$	3806	$\mathbf{1 \pm 0.00}$	3782	$\mathbf{1 \pm 0.00}$	**3768**	$\mathbf{1 \pm 0.00}$	3795	$\mathbf{1 \pm 0.00}$

Table 1: For each value of p, the left column is size of R and the right one is accuracy $\pm$ standard deviation. Bold values are the best values among that row.

Dataset	# points in S	# of queries	# dimension
AERIAL	45000	5000	60
COREL	45000	5000	89
SIFT	45000	5000	128
USPS	7000	2298	256

Table 2: Dataset description

needs to store $(m \cdot d)$ coordinates (where d is the data dimensionality), whereas sparse version will store $(m \cdot d \cdot p)$ coordinates on average. Thus percentage of space savings is $(m \cdot d - m \cdot d \cdot p)/(m \cdot d) = (1 - p)$ percentage. Thus $p = 1.0$ corresponds to original non-sparse RP-tree (no space savings), whereas, $p = 0.1$, we have 90% space savings.

Goal of this experiment was to evaluate[4] how sparse RP-tree answers nearest neighbor search query compared to its non-sparse counterpart for various values of p. The results for four datasets are reported in Table 1. As can seen from the table, in most cases non-sparse RP-tree has the best performance in terms of higher accuracy and lower number of retrieved points (R). More importantly, both accuracy and number of retrieved points of sparse RP-trees are very close to that of non-sparse RP-trees, even when p equals to 0.1.

[4]Note that in this paper we do not compare our results with LSH. Such a comparison can be found in [18] which demonstrate that original non-sparse RP-tree performs better than LSH in terms of nearest neighbor search accuracy and number of retrieved points.

5 Conclusion

In this paper we have presented a sparse version of randomized partition tree for performing nearest neighbor search. We have shown that at each internal node of our proposed sparse RP-tree, we only need to store a few non-zero entries, as opposed to all d entries, leading to huge significant space savings without sacrificing much in terms of nearest neighbor search accuracy. We have also shown that an additional advantage of our approach is slightly improved query time compared to non-sparse RP-tree for large dataset size. We have theoretically shown that, at any internal node, the expected fraction of non-nearest neighbor points that fall between the query point and its nearest neighbor upon projection is very similar to its non-sparse RP-tree counterpart, except an additional small (user controllable) multiplicative and additive term. This indicates that all theoretical guarantees of non-sparse RP-tree for nearest neighbor search essentially hold for our proposed sparse version as well. We have demonstrated that our experimental evaluations on four real world datasets strongly agree with our theoretical results and nearest neighbor search accuracies of our proposed sparse method is very similar to that of its non-sparse counterpart in terms of accuracy and number of retrieved points.

References

[1] N. Ailon and B. Chazelle. The fast Jonson-Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39:302–322, 2009.

[2] A. Andoni and P. Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[3] K. Bache and M. Lichman. UCI machine learning repository, 2013. Available at : `http://archive.ics.uci.edu/ml`.

[4] A. Beygelzimer, S. Kakade, and J. Langford. Cover Trees for Nearest Neighbor. In *23rd International Conference on Machine Learning*, 2006.

[5] P. Ciaccia, M. Patella, and P. Zezula. M-tree : An Efficient Access Method for Similarity Search in Metric Spaces. In *23rd VLDB International Conference*, 1997.

[6] A. Dasgupta, R. Kumar, and T. Sarlos. Fast locality sensitive hashing. In *17th ACM Conference on Knowledge Discovery and Data Mining*, 2011.

[7] S. Dasgupta and K. Sinha. Randomized partition trees for exact nearest-neighbor search. In *26th Annual Conference on Learning Theory*, 2013.

[8] S. Dasgupta and K. Sinha. Randomized Partition Trees for Nearest Neighbor Search. *Algorithmica*, 72(1):237 – 263, 2015.

[9] M. Datar, N. Immorlica, P. Indyk, and C. S. Mirrokni. Locality-Sensitive Hashing Based on p-Stable Dis. In *The 20th ACM Symposium on Computational Geometry*, 2004.

[10] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *25th International Conference on Very Large Databases*, 1999.

[11] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[12] N. Katayama and S. Satoh. The ST-tree : An Index Structure for High-dimensional Nearest Neighbor Queries. In *Annual SIGMOD Conference*, 1997.

[13] T. Liu, A. W. Moore, A. Gray, and K. Yang. An Investigation of Practical Approximate Nearest Neighbor Algorithms. In *18th Annual Conference on Neural Information Processing Systems*, 2004.

[14] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieving of large image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[15] J. McNames. A Fast Nearest Neighbor Algorithm Based on a Principal Axis Search Tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):964–976, 2001.

[16] A. Shrivastava and P. Li. Fast Near Neighbor Search in High-Dimensional Binary Data. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2012.

[17] A. Shrivastava and P. Li. Densifying One Permutation Hashing via Rotation for Fast Nearest Neighbor Search. In *31st International Conference on Machine Learning*, 2014.

[18] K. Sinha. LSH vs Randomized Partition Trees : Which One to Use for Nearest Neighbor Search? In *13th International Conference on Machine Learning and Applications*, 2014.

[19] J. K. Uhlmann. Satisfying General Proximity/Similarity Queries with Mteric Trees. *Information Processing Letters*, 40:175–179, 1991.

[20] N. Verma, S. Kpotufe, and S. Dasgupta. Which Spatial Partition Trees are Adaptive to Intrinsic Dimension? In *25th International Conference on Uncertainty in Artificial Intelligence*, 2009.