Supplementary material for improved nearest neighbor search using auxiliary information and priority functions

Omid Keivani [1] **Kaushik Sinha** [1]

1. Introduction

This supplementary material is organized as follows. Algorithms for query processing for defeatist search with auxiliary information as well as guided prioritized search are presented in section 2. In section 3, we provide proofs of theorems presented in the paper. Additional experiments are presented in section 4.

2. Query processing

In this section we presents algorithm for query processing for defeatist search with auxiliary information, guided prioritized search and the combined approach. These algorithms are presented in Algorithm 2, 3 and 4. For easy reference we provide RPT construction algorithm with auxiliary information as well.

3. Proof of theorems

Proofs of all theorems presented in submitted manuscript are provided below.

3.1. Proof of theorem 1

Proof. For any $x_{(i)} \in \mathcal{S}$, using lemma 1 of (Li & Malik, 2016), we get, $\Pr\left(|U^\top(q-x_{(i)})| \le |U^\top(q-x_{(1)})|\right) \le 1 - \frac{2}{\pi}\arccos\left(\frac{\|q-x_{(1)}\|_2}{\|q-x_{(i)}\|_2}\right)$. Noting that for any z, $\arccos(z) = \frac{\pi}{2} - \arcsin(z)$, and the inequality $\theta \ge \sin\theta \ge \frac{2\theta}{\pi}$, for $0 \le \theta \le \frac{\pi}{2}$, we get $\Pr\left(|U^\top(q-x_{(i)})| \le |U^\top(q-x_{(1)})|\right) \le \frac{\|q-x_{(1)}\|_2}{\|q-x_{(i)}\|_2}$. Let Z_i be indicator variable that takes value 1 if $|U^\top(q-x_{(i)})| \le |U^\top(q-x_{(1)})|$, and 0 otherwise. Then $\mathbb{E}(Z_i) \le \frac{\|q-x_{(1)}\|_2}{\|q-x_{(i)}\|_2}$. Let $Z = \sum_{i=1}^{|S|} Z_i$. Then Z indicates the number of points in S whose distance from q upon projection is smaller than $|U^\top(q-x_{(1)})|$. Using Markov's inequality, $\Pr(Z > k) \le \frac{\mathbb{E}(Z)}{k} = \frac{\sum_{i=1}^{|S|}\mathbb{E}(Z_i)}{k} \le \frac{1}{k}\sum_{i=1}^{|S|}\frac{\|q-x_{(1)}\|_2}{\|q-x_{(i)}\|_2}$. □

Algorithm 1 RPT construction with auxiliary information

Input : data $\mathcal{S} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, maximum number of data points in leaf node n_0, auxiliary index size c, m independent random vectors $\{V_1, \ldots, V_m\}$ sampled uniformly from $\mathcal{S}^{d-1}$.
Output : tree data structure
function `MakeTree`(S, n_0)

if $|S| \le n_0$ **then**
  return leaf containing S
else
  Pick U uniformly at random from $\mathcal{S}^{d-1}$
  Let v be the median of projection of S onto U
  Set `ail` be the set of indices of c points in S so that upon projection onto U, they are the c closest points to v from the left.
  Set `air` be the set of indices of c points in S so that upon projection onto U, they are the c closest points to v from the right.
  Construct a $c \times m$ matrix $\mathtt{L_{cnn}}$ whose i^{th} row is the vector $(V_1^\top x_{\mathtt{ail}(i)}, V_2^\top x_{\mathtt{ail}(i)}, \ldots, V_m^\top x_{\mathtt{ail}(i)})$.
  Construct a $c \times m$ matrix $\mathtt{R_{cnn}}$ whose i^{th} row is the vector $(V_1^\top x_{\mathtt{air}(i)}, V_2^\top x_{\mathtt{air}(i)}, \ldots, V_m^\top x_{\mathtt{air}(i)})$.
  `Rule`$(x) = (x^\top U \le v)$
  `LSTree` = `MakeTree`$(\{x \in S : \mathtt{Rule}(x) = \mathtt{true}\}, n_0)$
  `RSTree` = `MakeTree`$(\{x \in S : \mathtt{Rule}(x) = \mathtt{false}\}, n_0)$
  return (`Rule`, `LSTree`, `RSTree`)
end if

3.2. Proof of theorem 2

Proof. Let $c = kn_0$. Since we are using median split, it is easy to see that exactly $\lceil\frac{k}{2}\rceil$ levels from the leaf node level (and excluding leaf node level) will have less than c points on each side of the median. Note that at the leaf node level, we have at most $\lceil\frac{n}{n_0}\rceil$ nodes. Now consider the level just above the leaf node level. Total number of nodes at this level is at most $\frac{1}{2}\cdot\lceil\frac{n}{n_0}\rceil$ and on each side of the median we have at most n_0 points. Since $n_0 < c$, on each side of the median, for each node at this level, will store a matrix of size $n_0 \times (m+1)$ matrix ($n_0 \times m$ matrix for m dimensional representation of n_0 points and additional $n_0 \times 1$ space for storing index of these n_0 points). If we further go one level up, maximum number of nodes at this

[1]Department of Electrical Engineering & Computer Science, Wichita State University, KS, USA. Correspondence to: Kaushik Sinha <kaushik.sinha@wichita.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Algorithm 2 Query processing using defeatist search with auxiliary information

Input : RP tree constructed using Algorithm 1 from main paper, m independent random vectors $\{V_1, \ldots, V_m\}$ sampled uniformly from $\mathcal{S}^{d-1}$, query q, number of candidate neighbors at each node c'.
Output : Candidate nearest neighbors

Set $\mathcal{C}_q = \emptyset$.
Set $\tilde{q} = (V_1^\top q, V_2^\top q, \ldots, V_m^\top q)$
Set `current_node` to be root node of the input tree.
while `current_node` $\neq$ `leaf_node` **do**
  if $U^\top q < v$ **then**
    $\mathtt{A} = \mathtt{R_{cnn}}$
    $\mathtt{ai} = \mathtt{air}$
    `current_node = current_node.left`
  else
    $\mathtt{A} = \mathtt{L_{cnn}}$
    $\mathtt{ai} = \mathtt{ail}$
    `current_node = current_node.right`
  end if
  Sort the rows of `A` in increasing order according to their distance from $\tilde{q}$ and let array `a` contains these sorted indices.
  $\mathcal{C}_q = \mathcal{C}_q \cup \{\mathtt{ai}(\mathtt{a}(1)), \mathtt{ai}(\mathtt{a}(2)), \ldots, \mathtt{ai}(\mathtt{a}(c'))\}$
end while
Set $\mathtt{leaf}_q$ to be the indices of points in S that lie in `leaf_node`.
$\mathcal{C}_q = \mathcal{C}_q \cup \mathtt{leaf_q}$
return $\mathcal{C}_q$

level is $\frac{1}{4} \cdot \lceil \frac{n}{n_0} \rceil$ and on each side of the median we have at most $2n_0$ points and so on. Therefore total additional space complexity for storing auxiliary information at $\lceil \frac{k}{2} \rceil$ levels from the leaf node level is,

$$
\begin{aligned}
2(m+1)\sum_{i=1}^{\lceil \frac{k}{2} \rceil} \frac{1}{2^i} \cdot \lceil \frac{n}{n_0} \rceil \cdot (2^{i-1} n_0) &\leq 2(m+1)n \sum_{i=1}^{\lceil \frac{k}{2} \rceil} \frac{1}{2} \\
&= (m+1)n \lceil \frac{k}{2} \rceil
\end{aligned}
$$

For the remaining levels, we store $c \times (m+1)$ matrix on each side of the median at each internal node. Total additional space required to store this auxiliary information is,

$$
\begin{aligned}
& 2(m+1)c \left(\sum_{i=0}^{\log \lceil \frac{n}{n_0} \rceil - \left(\lceil \frac{k}{2} \rceil + 1 \right)} 2^i \right) \\
= \; & 2(m+1)c \left(\frac{2^{\log \lceil \frac{n}{n_0} \rceil - \lceil \frac{k}{2} \rceil} - 1}{2-1} \right) \\
\leq \; & 2(m+1)c \frac{1}{2^{\lceil \frac{k}{2} \rceil}} \lceil \frac{n}{n_0} \rceil \\
\leq \; & 2(m+1)n \frac{k}{2^{\lceil \frac{k}{2} \rceil}}
\end{aligned}
$$

Summing these two terms, additional space requirement is,

$$
(m+1)n \left(\lceil \frac{k}{2} \rceil + \frac{2k}{2^{\lceil \frac{k}{2} \rceil}} \right) \leq 6(m+1)n
$$

where the last inequality follows from the fact that $\lceil \frac{k}{2} \rceil + \frac{2k}{2^{\lceil \frac{k}{2} \rceil}} \leq 6$ for all $k \leq 10$. In addition, we also need to store m random projection directions for the entire tree requiring extra md space. Therefore, total additional space requirement for storing auxiliary information is at most $6(m+1)n + md \leq (m+1)(6n+d)$.

Now, note that we want to choose number of projection directions m in such a way that for all auxiliary data points

Algorithm 3 Query processing using prioritized guided search

Input : RP tree constructed using Algorithm 1 from main paper, query q, number of iterations t
Output : Candidate nearest neighbors

Set $\mathcal{C}_q = \emptyset$.
Set $\mathcal{P}$ to be an empty priority queue.
Set `current_node` to be root node of the input tree.
if $t > 0$ **then**
  while `current_node` $\neq$ `leaf_node` **do**
    if $U^\top q < v$ **then**
      `current_node = current_node.left`
    else
      `current_node = current_node.right`
    end if
    Compute `priority_value`
    $\mathcal{P}$.`insert(current_node, priority value)`.
  end while
  Set $\mathtt{leaf}_q$ to be the indices of points in S that lie in `leaf_node`.
  $\mathcal{C}_q = \mathcal{C}_q \cup \mathtt{leaf_q}$
  Set $t = t - 1$
  Set `current_node` $= \mathcal{P}$.`extract_max`
end if
return $\mathcal{C}_q$

Algorithm 4 Query processing using combined approach with auxiliary information

Input : RP tree constructed using Algorithm 1, m independent random vectors $\{V_1, \ldots, V_m\}$ sampled uniformly from $\mathcal{S}^{d-1}$, query q, number of iterations t, number of candidate neighbors at each node c'.
Output : Candidate nearest neighbors

Set $\mathcal{C}_q = \emptyset$, set $\mathcal{B}$ to be an empty binary search tree
Set $\mathcal{P}$ to be an empty priority queue, set $\texttt{count} = 0$
Set $\tilde{q} = (V_1^\top q, V_2^\top q, \ldots, V_m^\top q)$
Set $\texttt{current\_node}$ to be root node of the input tree.
if $t > 0$ **then**
  while $\texttt{current\_node} \neq \texttt{leaf\_node}$ **do**
    if $U^\top q < v$ **then**
      $\texttt{A} = \texttt{R}_{\texttt{cnn}}, \texttt{ai} = \texttt{air}$
      $\texttt{current\_node} = \texttt{current\_node.left}$
    else
      $\texttt{A} = \texttt{L}_{\texttt{cnn}}, \texttt{ai} = \texttt{ail}$
      $\texttt{current\_node} = \texttt{current\_node.right}$
    end if
    Compute $\texttt{priority\_value}$, set $\texttt{count} = \texttt{count} + 1$
    Sort the rows of $\texttt{A}$ in increasing order according to their distance from $\tilde{q}$ and let array $\texttt{a}$ contains these sorted indices.
    Insert $\{\texttt{ai}(\texttt{a}(1)), \ldots, \texttt{ai}(\texttt{a}(c'))\}$ into $\mathcal{B}$ with value $\texttt{count}$
    Set $\texttt{struct} = (\texttt{count}, \texttt{current\_node})$
    $\mathcal{P}.\texttt{insert}(\texttt{struct}, \texttt{priority value})$.
  end while
  Set $\texttt{leaf}_q$ to be the indices of points in S that lie in $\texttt{leaf\_node}$.
  Set $\mathcal{C}_q = \mathcal{C}_q \cup \texttt{leaf}_\texttt{q}$, $t = t - 1$
  Set $\texttt{current\_node} = (\mathcal{P}.\texttt{extract\_max}).\texttt{current\_node}$
  Delete from $\mathcal{B}$ candidate set with value $(\mathcal{P}.\texttt{extract\_max}).\texttt{count}$
end if
return $\mathcal{C}_q \cup \{$all candidate set from $\mathcal{B}\}$

stored at the internal nodes along a query routing path (from root node to leaf node) and for the query, pairwise distances are preserved up to a multiplicative error $(1 \pm \epsilon)$ compared to the respective original distances. Total number of such points is $n' = 2c\left(\log\lceil \frac{n}{n_0}\rceil - 1\right) + 1 \leq 2c\log\lceil\frac{n}{n_0}\rceil$, considering all levels except leaf node level. JL lemma tells us that $m = O\left(\frac{\log n'}{\epsilon^2}\right) = O\left(\frac{\log c + \log\log(n/n_0)}{\epsilon^2}\right) = O\left(\log\log(n/n_0)\right)$ would suffice, where the last inequality follows if we fix ϵ and since c is a fixed quantity. therefore total additional space complexity for storing auxiliary information is $O\left((n+d)\log\log(n/n_0)\right)$, which we write as $\tilde{O}(n+d)$ hiding the $\log\log(n/n_0)$ factor. $\square$

3.3. Proof of theorem 3

Proof. Due to overlap, at each successive level size of the internal node reduces by factor of $(\frac{1}{2} + \alpha)$. Simple calculation shows that depth of the tree is at most $k = \log_{\frac{2}{1+2\alpha}}(n/n_0)$. Total number of nodes is,

$$
\begin{aligned}
2^{k-1} &= \frac{1}{2} \cdot 2^{\log_{\frac{2}{1+2\alpha}}(n/n_0)} \\
&= \frac{1}{2} \cdot 2^{\log_2(n/n_0)\cdot \log_{\frac{2}{1+2\alpha}} 2} \\
&= \frac{1}{2} \cdot \left(\frac{n}{n_0}\right)^{\log_{\frac{2}{1+2\alpha}} 2} \\
&= \frac{1}{2} \cdot \left(\frac{n}{n_0}\right)^{\frac{1}{\log_2 \frac{2}{1+2\alpha}}} \\
&= \frac{1}{2} \cdot \left(\frac{n}{n_0}\right)^{\frac{1}{1-\log_2(1+2\alpha)}}
\end{aligned}
$$

Due to median split, total number of internal nodes will be exactly one less than the number of leaf nodes, moreover, we need to store a random projection direction at each of these internal nodes. Therefore, for fixed n_0, space complexity of spill tree is $O\left(dn^{\left(\frac{1}{1-\log_2(1+2\alpha)}\right)}\right)$. $\square$

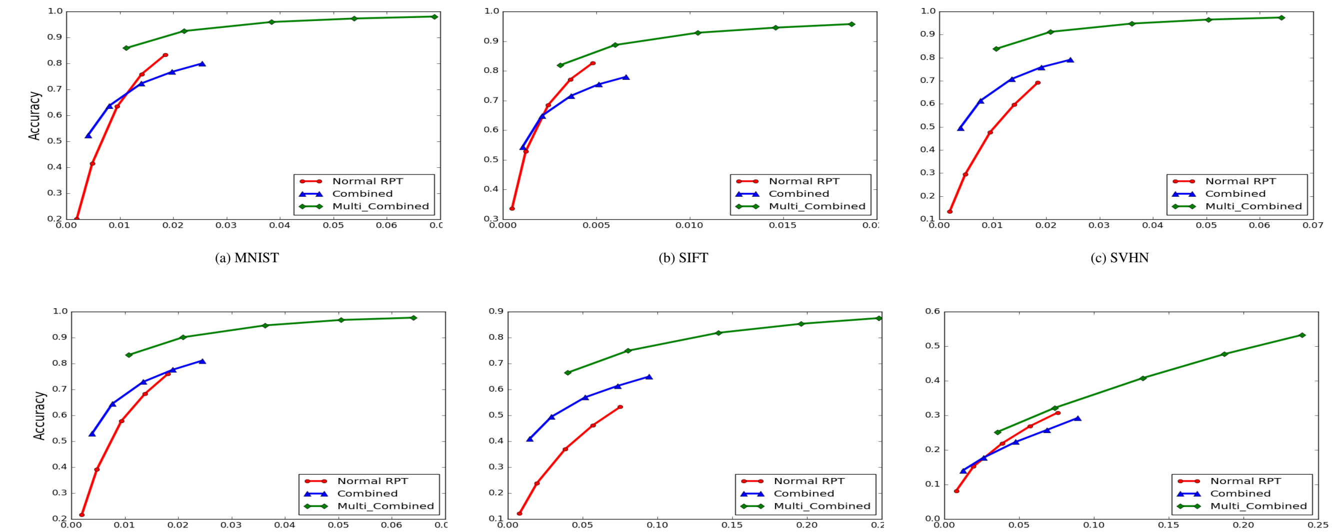
4. Additional experiments

4.1. 1-NN search experiments

In the following we present experimental evaluations for Experiment 3 of the submitted manuscript. Here, we presents for 1-NN search results in Figure 1 and 2. We note that we observe similar trend that we observed for 10-NN search in the submitted manuscript.

References

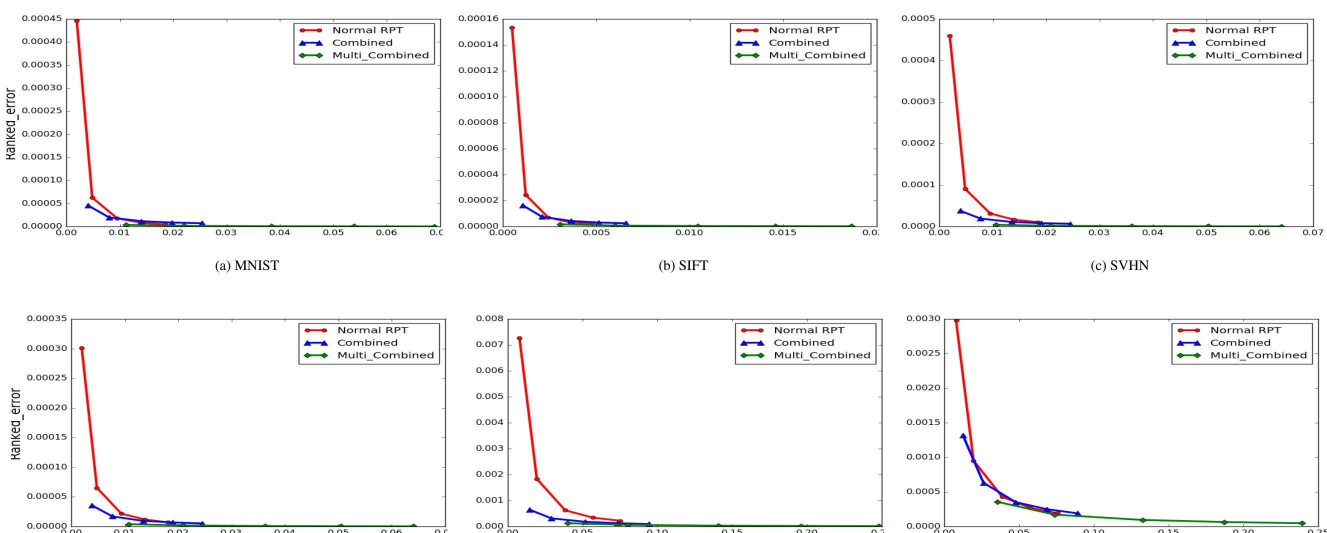
Li, K. and Malik, J. Fast k-nearest neighbor search via dynamic continuous indexing. In *33rd International Conference on Machine Learning (ICML)*, 2016.



(a) MNIST (b) SIFT (c) SVHN

(d) JESTER (e) 20NEWSGROUP (f) SIAM07

Figure 1. 1-NN accuracy for all six datasets. The x-axis represents the ratio of # of retrieved points to the total number of instances. The markers from left to right corresponding to 2, 5, 10, 15 and 20 iterations (for combined and Multi-Combined method) / trees (for normal RPT).



(a) MNIST (b) SIFT (c) SVHN

(d) JESTER (e) 20NEWSGROUP (f) SIAM07

Figure 2. Rank error (1-NN) for all six datasets. The x-axis represents the ratio of # of retrieved points to the total number of instances. The markers from left to right corresponding to 2, 5, 10, 15 and 20 iterations (for combined and Multi-Combined method) / trees (for normal RPT).