An Improved Nearest Neighbor Search Algorithm Based on LAESA

Kazuaki Yamaguchi Yoichi Kondo
Faculty of Engineering
Kobe University
1-1 Rokkodai, Nada, Kobe, 657–8501
JAPAN
ky@eedept.kobe-u.ac.jp http://www.eedept.kobe-u.ac.jp/~ky/

Abstract: Nearest neighbor search is to find the nearest object to the query from a set S. Nearest neighbor search is one of the important parts of pattern recognition and multimedia data management. We introduce an improved algorithm based on the current fastest algorithm, LAESA, to find nearest neighbors in general metric spaces. Our algorithm is faster than mvp-tree, the most well-known method, and requires smaller storage than LAESA.

Key-Words: nearest neighbor, metric space, branch and bound, LAESA, mvp-tree, distance axiom

1 Introduction

NNS (Nearest Neighbor Search) is to find the nearest object to the query q from a set of objects S of n objects. NNS can be applied for pattern recognition and multimedia data management. NNS consists of following two procedures.

1. To construct an appropriate data structure from an object set S and its distance function $d(\cdot,\cdot)$.
2. To search the nearest object to the query q from S as fast as possible.

We refer to the procedure 1 and procedure 2 as precomputing and search, respectively.

We assume that the distance function $d : S^2 \mapsto R$ satisfies the distance axiom, where R is the set of real numbers. Thus, any x, y, z in S satisfy the following.

$$\begin{aligned} d(x,y) = 0 &\Leftrightarrow x = y \\ d(x,y) &= d(y,x) \\ d(x,y) + d(y,z) &\geq d(x,z) \end{aligned}$$

In many cases, objects in S are converted into feature vectors during precomputing process, because vectors are much easier to deal with than objects themselves. NNS algorithms to deal with feature vector are SR-tree[4], Voronoi diagram[7] and so on.

But there are some objects, for example, chemical compounds, which are impossible to be converted into feature vectors without losing structural information. There exist several NNS algorithms, which only use distance function. For example, AESA[9], LAESA[5], vp-tree[10], mvp-tree[1], the algorithm in [8] and so on. AESA is the fastest algorithm among the NNS algorithm only using distance function, although it requires quadratic storage size. So, LAESA is the fastest algorithm among practical algorithm. We propose an efficient algorithm based on LAESA.

In article [5], it is assumed that the computation time of distance function is very expensive, and it is assumed that the search time can be evaluated only with the number of distance computation. This criterion is useful because it is independent from the difference of the experimental environment, and more detailed than the $O(\cdot)$ evaluation. In this paper, we use this evaluating criterion.

2 Preliminaries

Here we briefly introduce the relationship between NNS and branch and bound, and briefly show the idea of LAESA.

2.1 Branch and bound

Branch and bound is a generic algorithm to solve optimization problems. The branch and bound is described in the following.

- Partition an original minimization (maximization) problem into several smaller minimization (maximization) subproblem.
- Calculate the lower (upper) bound of the evaluate function of each subproblem.
- Solve each subproblem in the order of smaller lower bound (bigger upper bound).

NNS can be thought as an optimization problem to find the object s_j, which satisfies $d(q, s_j) = \min_{s \in S} d(q, s)$, from a set $S = \{s_1, s_2, \cdots, s_n\}$. So, we can apply branch and bound for NNS, and actually most NNS algorithms use branch and bound implicitly.

A simple algorithm of calculating lower bound is shown in [2]. The following inequality is derived from the distance axiom.

$$\begin{aligned} d(q, s') \quad \geq \quad & |d(q, s) - d(s, s')| \qquad (1) \\ & \text{for any } s, s' \in S \end{aligned}$$

If we compute $d(s_1, s_2)$, $d(s_1, s_3)$, $\cdots$, $d(s_1, s_n)$ before we receive q, we can immediately obtain the lower bounds $|d(q, s_1) - d(s_1, s_2)|$, $|d(q, s_1) - d(s_1, s_3)|$, $\cdots$,$|d(q, s_1) - d(s_1, s_n)|$ for $d(q, s_2)$,$d(q, s_3)$,$\cdots$, $d(q, s_n)$ respectively, only by calculating $d(q, s_1)$ when q is given.

Vp-tree and mvp-tree recursively use branch and bound method. LAESA, which provides more tight lower bound than these algorithms, calculates less number of distances, even though LAESA is non-recursive.

2.2 LAESA

In this section we briefly introduce the method LAESA.

2.2.1 Precomputing

The precomputing process is as follows :

1. Choose a subset B with m elements from S with a heuristic algorithm[5].
2. Calculate $d(b_i, s_j)$ for all $b_i \in B$, $s_j \in S$.

The data structure of LAESA is the 2-dimensional matrix to store $d(b_i, s_j)$.

2.2.2 Search algorithm

The search algorithm is as follows :

1. Calculate $d(q, b_1)$,$d(q, b_2)$,$\cdots$,$d(q, b_m)$.
2. Calculate
$$A(s_j) = \max_{1 \leq i \leq m} |d(q, b_i) - d(b_i, s_j)|$$
for each $s_j \in S - B$.
3. Calculates $d(q, s_j)$ for each $s_j \in S - B$, in increasing order of $A(s_j)$. Stop the calculation if an element whose distance to q is smaller than the smallest lower bound of the rest, is found.

2.2.3 Features

The number of distance computation of LAESA is smaller than other methods. According to the article [5] and other articles, the number of distance computations of LAESA is asymptotically constant, no matter how big n is. We verified the fact with some experiments. The numbers of distance computations asymptotically become 13, 90 and 614 if we use Euclidean metric space with the dimension of 5, 10 and 15 respectively.

But to obtain the result above, the number m must be considerably bigger, whereas LAESA requires to store a 2-dimensional matrix of mn elements. For example, the best number of m is 48 if S is the set of points in 10-dimensional Euclidean metric space [5]. And LAESA also requires to calculates the lower bound for all elements in $S - B$, whose computation requires $O(mn)$ time.

3 Our algorithm

We present an improved algorithm based on LAESA. Our algorithm requires smaller storage for the data structure than LAESA. Our algorithm reduces the computation time for computing the lower bounds.

Our method uses an integral parameter c (≥ 2). Theoretically c can be any integer, but the appropriate number for c is between 2^8 and 2^{10}. Our method also uses a real number ρ, a function $f(\cdot,\cdot)$ and m-dimensional vector $\phi(s_j)$ defined in the following.

$$\begin{aligned}
\rho &= \max_{1\leq i\leq m,\ 1\leq j\leq n} d(b_i,s_j)/c && (2)\\
f(b_i,x) &= \lceil d(b_i,x)/\rho \rceil && (3)\\
\phi(s_j) &= (f(b_1,s_j), f(b_2,s_j), \cdots,\\
&\qquad f(b_m,s_j)) && (4)
\end{aligned}$$

From the equation (2), $0 \leq f(b_i,s_j) < c$ for any i,j is easily obtained. We sometimes refer to $f(b_i,s_j)$ as $f(i,j)$ in the following. Define

$$d_\infty(x,y) = \max_{i=1}^{m} |x_i - y_i| \qquad (5)$$

for m-dimensional vectors $x = (x_1,x_2,\cdots,x_m)$ and $y = (y_1,y_2,\cdots,y_m)$. $d_\infty(x,y)$ is called ℓ_∞ distance. We refer to $d_\infty(\phi(b_i),\phi(s_j))$ and $d_\infty(\phi(q),\phi(s_j))$ as $d_\infty(b_i,s_j)$ and $d_\infty(q,s_j)$ respectively.

Theorem 1

$$\rho(d_\infty(q,s_j)-1) \leq A(s_j) \leq \rho(d_\infty(q,s_j)+1)$$

for any $s_j \in S$.

Proof

The two inequalities can be proved almost the same way. So we show only the proof of $\rho(d_\infty(q,s_j)-1) \leq A(s_j)$. From the equation (3),

$$\rho f(b_i,x) \leq d(b_i,x) < \rho(f(b_i,x)+1)$$

for any x. Thus,

$$\begin{aligned}
&|d(q,b_i) - d(b_i,s_j)|\\
&= \max\{d(q,b_i)-d(b_i,s_j),\ d(b_i,s_j)-d(q,b_i)\}\\
&\geq \max\{\rho f(q,b_i) - \rho(f(b_i,s_j)+1),\\
&\qquad \rho f(b_i,s_j) - \rho(f(q,b_i)+1)\}\\
&= \rho\max\{f(q,b_i)-(f(b_i,s_j)+1),\\
&\qquad f(b_i,s_j)-(f(q,b_i)+1)\}\\
&= \rho(|f(q,b_i)-f(b_i,s_j)|-1)
\end{aligned}$$

$$\begin{aligned}
A(s_j) &= \max_{i=1}^{m} |d(q,b_i)-d(b_i,s_j)|\\
&\geq \max_{i=1}^{m} \rho(|f(q,b_i)-f(b_i,s_j)|-1)\\
&= \rho((\max_{i=1}^{m} |f(q,b_i)-f(b_i,s_j)|)-1)\\
&= \rho(d_\infty(\phi(q),\phi(s_j))-1)
\end{aligned}$$

□

3.1 Data structure

In most practical cases, the distance is real number, whose size is usually 4 or 8 bytes. So, LAESA requires storage 4 or 8 bytes per object. But our algorithm stores $d(b_i,s_j)$, whose size is $\lceil \log_2 c \rceil$. Our algorithm with $c = 2^8$ requires only 1 byte per object.

Data structures for high dimensional vectors, k-d tree or MD-tree [6], can deal with $\phi(s_i)$. We propose to store $\phi(s_j)$ in such a data structure, whereas LAESA stores $d(b_i,s_j)$ just in a 2-dimensional array. The efficiency of the difference can be seen later.

3.2 Search algorithm

The search algorithm of our method is similar to LAESA.

1. Calculate $d(q,b_1)$,$d(q,b_2)$,$\cdots$,$d(q,b_m)$. Then we immediately obtain $\phi(q)$.

2. Calculate $d_\infty(q,s_j)$ for each element $s_j \in S - B$.

3. Calculates $d(q,s_j)$ for each $s_j \in S - B$, in increasing order of $A(s_j)$. Stop the calculation if an element whose distance to q is smaller than the smallest lower bound of the rest, is found.

If the distance between the query q and s_j is smaller than r in ℓ_∞ metric space, s_j is in the hyper-cube of size $(2\rho)^m$. So, to find s_j in increasing order of $d_\infty(q, s_j)$, which is equivalent to NNS in ℓ_∞ metric space, is efficiently calculated with k-d tree or MD tree.

The only demerit of our algorithm could be the looseness of the lower bound. Theorem 1 shows that the lower bound of our method is always looser than the lower bound of LAESA. But it also shows that the difference of the lower bounds of our method and LAESA is smaller than 2ρ. From that fact we can obtain

$$\rho(d_\infty(q, s_j) - 1)/A(s_j) > 0.99 \qquad (6)$$

if $c = 2^8$, for example. We show the experimental result to make sure the difference of the performances is very slight.

3.3 Experimental Results

In the experiments, we used the points in multi-dimensional Euclidean space.

Figure 1 shows the number of distance computation with $n = 10000$. The constant numbers, 13, 90 and 614 in Figure 1 are the numbers of distance computation by LAESA of dimension 5, 10 and 15 respectively. It shows that the numbers of distance computation of our algorithm converge to the numbers of LAESA if $c \to \infty$, and that the performance of our algorithm is almost equivalent to LAESA when $c \geq 2^8$. We had another experiments with different n. The result was almost same even though n was changed.

The results show that the number of distance computation of our method is almost same as the number of distance computation of LAESA if $c \geq 2^8$.

4 Conclusion

Our method, based on LAESA, can reduce the size of data structure, and requires almost same number of distance computation as LAESA. Our method allows to use multidimensional data structure, so that we can reduce the number of computation of the lower bounds. In the future we would like to make sure if our method is really efficient in practical cases.

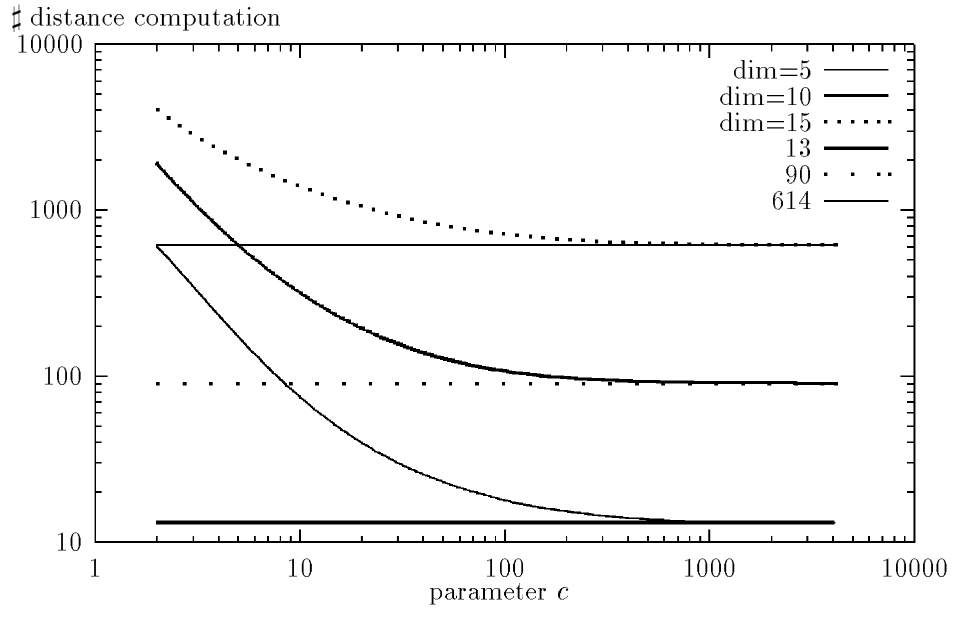

Figure 1: Comparison with LAESA

References

[1] T.Bozkaya, M.Ozsoyoglu, "Distance-based indexing for high-dimensional metric spaces," *Proceedings of the 1997 ACM SIGMOD*, 1997, pp.357–368.

[2] C.D.Feustel, L.G.Shapiro, "The nearest neighbor problem in an abstract metric space," *Pattern Recognition Letters*, vol.1, no.2, 1982, pp.125–128.

[3] A.K.Jain, R.Dubes, *Algorithms for clustering data* Prentice-Hall, 1988.

[4] Norio Katayama, Shin'ichi Satoh, "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," *Proceedings of the 1997 ACM SIGMOD*, 1997, pp.369–380.

[5] M.L.Mico, J.Oncina, E.Vidal, "A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements," *Pattern Recognition Letters*, vol.15, no.1, 1994, pp.9–17.

[6] Y.Nakamura, S.Abe, Y.Ohsawa, M.Sakauchi, "Md-Tree: a Balanced Hierarchical Data Structure for Multi-Dimensional Data with Highly Efficient Dynamic Characteristics," *Ninth International Conference on Pattern Recognition*, 1988, pp.375–378.

[7] V.Ramasubramanian, K.K.Paliwal, "Voronoi projection-based fast nearest-neighbor search algorithms: box-search and mapping table-based search techniques," *Digital Signal Processing*, vol.7, no.4, 1997, pp.260–277.

[8] D.Shasha, T.Wang, "New techniques for best-match retrieval," *ACM Trans. Inf. Systems,* vol.8, no.2, 1990, pp.140–158.

[9] E.Vidal, "An algorithm for finding nearest neighbours in (approximately) constant average time," *Pattern Recognition Letters*, vol.4, no.3, 1986, pp.145–157.

[10] P.N.Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," *Proceedings of the 1993 ACM-SIAM Symposium on Discrete Algorithms*, 1993, pp.311–321.