D-003

点密度を利用した近似近傍検索の改良
An Improvement of Approximate Nearest Neighbor Search with Point Density

平野　裕† 鹿山　俊洋† 大音　真由美†
Yutaka Hirano Toshihiro Kayama Mayumi Oto

1. 研究背景

組込み機器において地図、画像、センサデータなどの大規模な多次元データを扱う場合、データベースを組み込み機器に搭載することにより、データ管理のコストを抑え、開発を容易にすることができる。また、多次元データに対する基本的な操作をデータベースが提供することにより、さらに容易に多次元データを扱えるようになることが期待される。

本論文では、多次元データに対する k 近傍検索問題を議論する。これは、地図上であれば「この地点から近いオブジェクトを k 個」、画像であれば「この画像に似ている画像を k 個」をそれぞれ検索することに相当する。

一般に、対象とする空間の次元が高いと k 近傍検索問題は非常に難しくなり、分布にかなりの偏りがない限り線型探索が最も効率の良い解法となる。そのため、完全に正確ではないもののある程度の近傍を少ないコストで検索する近似近傍探索が提案されている。は、Arya らにより提案された、近似近傍検索手法である。

本研究では、Arya らの近似近傍検索アルゴリズムを改良し、最近傍検索に適した多次元インデクス SR-tree に対して適用する。さらに、これをデータベースエンジンに実装し、ページアクセス数、誤差率を計測して検索性能を確かめた。

以降、2 章で SR-tree について、3 章で近似近傍検索についてそれぞれ紹介する。4 章では SR-tree に対する近傍検索及び近似近傍検索のコストと誤差率を実験により測定し、その結果を評価する。

2. SR-tree

SR-treeは、空間インデクス R-treeの変種のひとつである。R-tree は木の各ノードに対して、配下の部分木の含む点を包囲する矩形を保持して近傍検索の枝刈りに用いるが、SR-tree ではそれに加えて配下の点を包囲する球も保持し、矩形と球の共通部分を包囲領域とするインデクスである。図 1 は SR-tree の包囲矩形、包囲球を図示したものである。

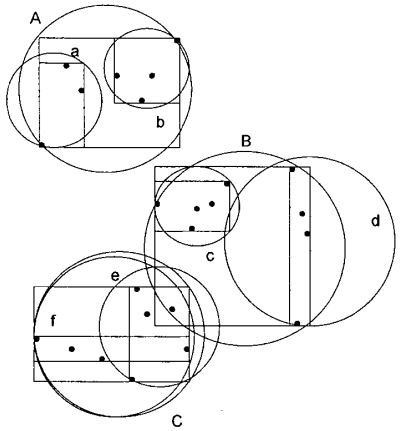

図 1 包囲矩形と包囲球

3. 近似近傍検索

3.1 k-近傍検索アルゴリズム

では、k-近傍を検索するアルゴリズムが提案されている。これは R-tree に対してのアルゴリズムであるが、「包囲矩形までの最小距離」を「包囲矩形と包囲球までの最小距離の最大」におきかえることにより、SR-tree にも適用することができる。

3.2 k-近似近傍検索アルゴリズム

近似係数 ε が与えられたとき、p が q の(1 + ε) – 近似 k 近傍であるということを、p と q の距離が q の真の k 近傍と q の距離の(1 + ε)倍を超えないことと定義する。

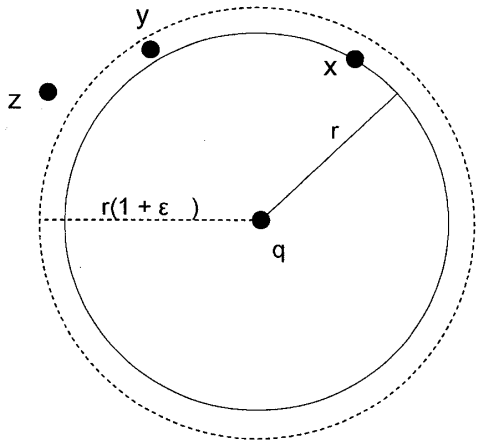

図 2 x と y は q の(1 + ε) – 近似 1 近傍

k-近似近傍検索とは、k と ε が与えられたとき、k 個の点 {(1+ε) - 近似 1 近傍 , … , (1+ε)-近似 k 近傍}を求めることである。Arya らはで、BBD-tree に対し近似近傍検索を適用し、次元 d, 近似係数 ε に対して O(d(1 + 6d / ε)^d log(n)) で計算量が抑えられることを証明した。ここでの近似近傍アルゴリズムとは、クエリ点から包囲領域までの距離を(1+ε)倍して考え、枝刈りの範囲を広げるアルゴリズムである。以下にアルゴリズムを示す。ただし、MINDIST は「包囲矩形と包囲球までの最小距離のうち、大きな方」を求めるための関数である。

```
def kNN(node, query, k, ε ):
 if nodeがリーフノード
   for point in node:
     近傍集合を更新
 else:
   children <- node の子ノード集合
   children を query との MINDIST でソート
   for child in children:
     p <- 近傍集合のうち、queryと最も遠い点
     if MINDIST(child, query) * (1 + ε ) ≧ p とquery の距離:
       break
     kNN(child, query, k)
```

図 3 近似近傍検索のアルゴリズム

3.3 k-近似近傍検索アルゴリズムの改良

上の近似近傍アルゴリズムは、どのノードについても枝刈り範囲を無差別に広げていた。しかし、大きな部分木(配下点数の多いノード)と小さな部分木(配下点数の少ない部分木)はノードの重要性、検索コストとも異なる。ここで、配下点とは、あるノードに(再帰的に)含まれる点データのことである。今、クエリ点までの距離(MINDIST)が同じ二つのノード X, Y を考える。X には配下点が多く、Y には配下点が少ないものとする。X は、配下点数が多いため、配下点がクエリ点 q の近傍に含まれる可能性が高い。また、配下点が多いため検索にコストがかかる。一方、配下点が多いと包囲領域が大きくなる傾向があるため、逆に q の近傍に含まれる可能性が小さくなる傾向もあることが分かる。Y については、逆のことが言える。

ここで、配下点が多いことによる点密度の増加及び包囲領域が大きいことによる点密度の低下の影響は、どちらが大きいのであろうか。もし配下点が多くなっても包囲領域が大きくならないような良いクラスタリングが行われているのであれば、そもそも近似近傍検索の必要性が薄い。そのような良いクラスタリングが行われていないから近似近傍検索を行うことを考えれば、一般的には、包囲領域が大きいことによる点密度の低下の影響が勝ると考えられる。

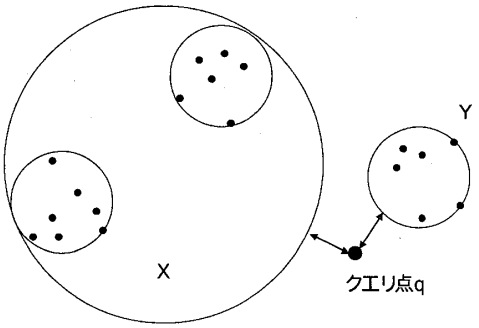

図 4 クエリ点からの最小距離が等しいノード

以上から、大きな部分木は大胆に枝刈りし、小さな部分木は慎重に扱うというアプローチが考えられる。以下は、改良された近似近傍検索のアルゴリズムである。

```
def kNN(node, query, k, ε ):
 if nodeがリーフノード
   for point in node:
     近傍集合を更新
 else:
   children <- node の子ノード集合
   children を query との MINDIST でソート
   for child in children:
     p <- 近傍集合のうち、queryと最も遠い点
     δ  <- min(ε , max(0, f(ε )))
     if MINDIST(child, query) * (1 + δ ) ≧ pとquery の距離:
       break
     kNN(child, query, k)
```

図 5 改良された近似近傍検索アルゴリズム

アルゴリズム中の関数 f は、近似係数を再計算する関数である。f を、大きな部分木には大きな近似係数を、小さな部分木には小さな近似係数を返すような関数に設定することにより、すでに述べたような方針で近似係数 δ が計算される。ここで、δ は $0 \leqq \delta \leqq \varepsilon$ をみたすため、与えられた近似係数の最悪保障を逸脱することは無い。

今回は、近似係数計算関数 f を

$$f(\varepsilon) = \gamma\varepsilon \frac{\log(\text{ノードの配下点数})}{\log(\text{木全体の配下点数})}$$

図 6

と定めた。ただし、γ は定数である。

4. 実験

4.1 実験環境

実験は以下のような設定で行った。なお、実験で測定するものがページアクセス数であるため、実験に用いた機器の性能は実験結果に影響しない。

パラメータ	値	説明
ページサイズ	7000[byte]	木のノードを格納するページの大きさ
次元	4, 8, 16, 32, 64, 128	点の次元
点数	40000 ~ 200000	
検索件数	2 ~20	
近似係数	0 ~10	

実験で使用したデータは、80 Milion Tiny Images[1] の画像データを主成分分析で各次元に次元縮約したものを用いた。

```
select id from images where
        knn(p, (select p from images where id = query), k,
eps);
```

実験においては上の SQL 文の query, k, eps のそれぞれに対して値をセットし、検索を行った後ページアクセス数を計測した。クエリは 40000 個の中からあらかじめランダムに 10 個を選び、全ての計測で同じクエリを使用した。

[1] http://people.csail.mit.edu/torralba/tinyimages/

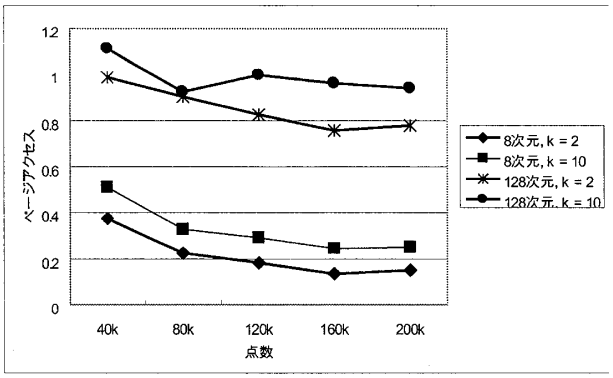

図 7

まず、近似を用いない場合での検索性能を線型探索の理論 I/O アクセス量と比較した。図 7 は、次元と k の設定を変えたときの線型探索と比べてのページアクセス数の比である。線型探索のページアクセス数は、ページサイズを合わせて管理情報を考えずに計算した。

図 7 からわかるように、次元や k が増えると、SR-tree の検索性能は劣化し、128 次元, k = 10 では線型探索と同程度となってしまう。

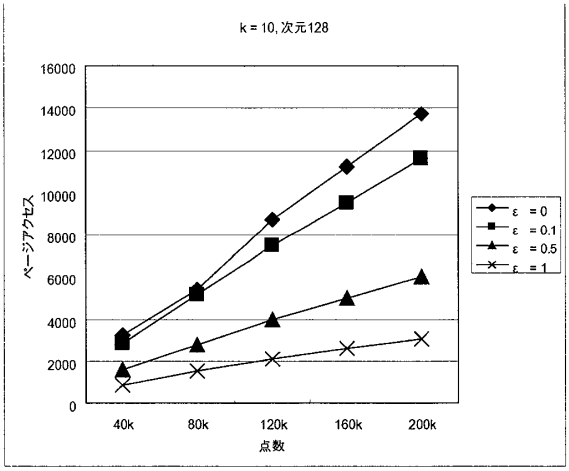

図 8

次に、改良の行われていない近似近傍検索の性能測定を行った。次元と k を固定して、近似係数を変えた場合にページアクセス数が　どのように変化するかを実験した。図 8 は、次元を 128, k を 10 として近似係数を変えた時のページアクセス数のグラフである。近似係数を増やすことにより、ページアクセス数が減少することがわかる。

最後に、近似検索方法を改良した時の性能を実験した。

図 9 は、近似無しの問い合わせを基準としたページアクセス比と正解率とのグラフである。次元が 128, k が 20 の条件において、既存手法と提案手法のそれぞれのページアクセス比をプロットした。プロット点は、右上から $\varepsilon = 0, 0.1, 0.5, 1, 2, 3, 5, 10$ の各測定データである。正解率は、k-近似近傍と k-近傍の共通部分の数を k で割ったものとした。

図 6 の関数 f 中の定数 γ は 1 とした。グラフ中では点が右下(正解率が高く、ページアクセス率が低い)にプロットされているほど良いため、既存手法より同じ正解率でのページアクセス比が下がっていることが確認できる。

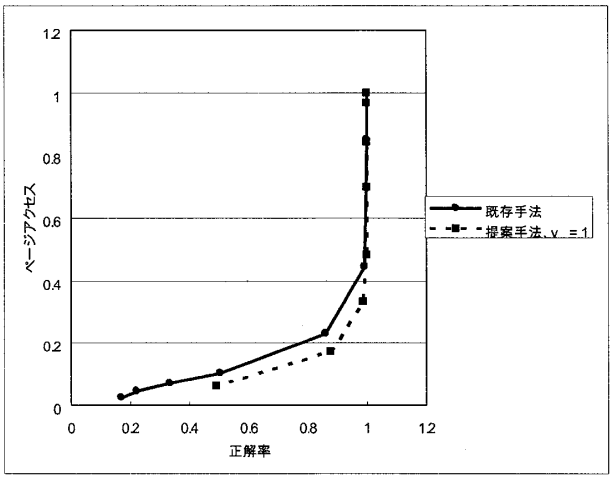

図 9 既存手法との比較

5. 結論

本研究では、多次元インデクス SR-tree に近似近傍検索アルゴリズムを実装し、性能を測定して有効性を確認した。また、近似近傍検索機能をデータベースエンジンに組み込み、近似係数をノードによって再計算することで近似近傍検索アルゴリズムを改良した。

次元が高い場合は SR-tree のみでは線型探索からの大幅な性能向上は難しい。SR-tree に対し近似近傍検索を適用すれば大きな性能向上が得られることを確認した。また、近似係数をノードによって変化させることにより、検索結果の正確さと検索コストのより良いトレードオフを実現できることを示した。

ノードの特性によって枝刈り基準を変えるという一般的な手法の中でも、今回はノードの配下点の数に注目し、特定の関数を使って係数を変化させて実験した。今後、さらに良い枝刈り基準を見つけることによって、より良いトレードオフを実現できると考えている。

参考文献

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A.Y. Wu, ``An optimal algorithm for approximate nearest neighbor searching,'' Journal of the ACM, Vol.45, pp.891-923, 1998891-923, 1998

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B.Seeger, "The R*-tree: an E_cient and Robust AccessMethod for Points and Rectangles," Proc. ACM SIGMOD, Atlantic City, USA, pp. 322--331, May 1990.

[3] Kevin S. Beyer , Jonathan Goldstein , Raghu Ramakrishnan , Uri Shaft, When Is "Nearest Neighbor" Meaningful?, Proceeding of the 7th International Conference on Database Theory, p.217-235, January 10-12, 1999

[4] Cheung, K., Fu, A. Enhanced Nearest Neighbour Search on the R-tree. *SIGMOD Record* 27(3): 16-21, 1998.

[5] A. Guttman, "R-trees: a Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD,Boston, USA, pp.47-57, Jun. 1984.

[6] Norio Katayama and Shin'ichi Satoh, ``The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (May 1997) pp. 369-380.

†(株) 東芝 ソフトウエア技術センター
Toshiba Corporation Corporate Software Engineering Center