情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-CVIM-169 No.13
2009/11/26

最近傍探索の理論とアルゴリズム

和田俊和†

†和歌山大学　システム工学部　情報通信システム学科

最近傍探索の研究においては，「次元の呪縛」と呼ばれる現象が問題を困難にしてきた．この現象は，数学的に厳密に定義されていないが，「蓄積されるデータの分布の次元が一定の値を超えると，いかなるアルゴリズムでも全探索と等価になる」という現象である．最近傍探索研究の歴史は，これを解決するために行われてきたと言っても過言ではない．これまでに数々の研究がなされてきたが，次元の呪縛が解けたという報告はこれまでになく，近年はこの現象を回避するための「近似最近傍」を探索する研究が盛んにおこなわれるようになってきた．本チュートリアルでは，近似を含むものと含まないもの両方について高速な最近傍探索アルゴリズムの解説を行う．その後に，果たして次元の呪縛を解くことが可能であるか否かについて再度検討を行う．

Theory and Algorithms for Nearest Neighbor Search

Toshikazu Wada†

†Dept. of Computer and Communication Sciences, Wakayama University

The phenomenon so called “curse of dimensionality” makes the Nearest Neighbor (NN) search problem difficult and attractive. Without this phenomenon, NN search is just a boring problem. One-NN search problem is to find the closest pattern to a given query, and k-NN search is to find k-closest patterns. For solving these problems, so many accelerated algorithms have been proposed. However, the curse tells us that every accelerated NN search algorithm becomes linear (exhaustive) search when the stored data form a high dimensional distribution. Researchers who tackled this problem are sometimes regarded as daydreamers, because they fought with an unbeatable ghost. Recently, they changed their mind to stop this straight forward fighting. Instead, they are focusing on approximate NN search, which does not face the phenomenon. In this tutorial, some typical exact and approximate NN search algorithms are introduced, and we revisit the phenomenon, “curse of dimensionality” so as to solve this problem.

1. 概要

最近傍探索とは，多数のデータから成る集合があり，その集合に属さないある１つのデータが与えられた際に，そのデータに最も近いデータを集合の中から探す問題である．この探索の際に与えられるデータを「クエリ」（質問）と呼ぶ．最近傍識別問題のように，集合それ自体がある一つのクラス(カテゴリ)を表しているような場合には蓄積されるデータを「プロトタイプ」と呼ぶことはあるが，最近傍探索の問題では単に「データ」と呼ぶことのほうが多い．形式的に説明すると，データ $\boldsymbol{x}, \boldsymbol{y}$ 間に距離 $d(\boldsymbol{x}, \boldsymbol{y})$ が定義されており，データ集合 $S = \{\boldsymbol{x}_i\}$ からクエリ $\boldsymbol{q}$ に対して最も近接するデータ（最近傍解）を探してくる問題が最近傍探索であり，最近傍解 $\boldsymbol{NN}(\boldsymbol{q}, S)$ は次式で定義される．

$$\boldsymbol{NN}(\boldsymbol{q}, S) \equiv \arg\min_{\boldsymbol{x} \in S} d(\boldsymbol{q}, \boldsymbol{x}) \tag{1}$$

この問題の基本的な解き方は，S 中のデータと，クエリ $\boldsymbol{q}$ との距離を全て計算し，最小値を与えるデータを求める全探索・総当たり探索(full search)，(linear search, brute-force search, exhaustive search などとも呼ばれる）である．これは，どのような場合にも正しい結果を与え，データ量を $N = |S|$，データの次元数を D とすれば，$O(ND)$ の計算量を要する．これは，全探索には「データ数」と「次元数」の両方に比例する計算量が必要であることを意味しており，大量・高次元のデータを扱う場合には，効率の良い探索アルゴリズムが必要になる．

完全照合探索（exact-match search）の場合には，二分探索やハッシュ法(hash algorithm)，木探索(tree search)など，線形探索よりも効率の良い方法が存在する．このことを考えれば，最近傍探索の場合にも効率の良い探索法が存在すると考えることは自然である．この高速な探索法を追及することが，これまでの最近傍探索研究の歴史であった．

Bentley による kd-木[1]はその代表例であり，様々なデータに対して極めて高速な計算結果をもたらすという大成功をもたらした．kd-木を用いた最近傍探索は，一種の A*アルゴリズム（あるいは分枝限定法）であり，巧妙かつシンプルな仕組みで最近傍解を求めることができるものであった[1]．

1.1 最近傍探索研究の原動力

しかし，このような成功の一方で深刻な事実が分かってきた．データ数 N に対して kd-木の二分探索が $O(\log_2 N)$ の比較演算で終わることから，当初これが計算量であると思われてきたが，実はそうではないということが明らかになってきた．二分探索の結果得られるのは「暫定解」であり，真の最近傍ではない．そのため，暫定解を利用

1 その後提案された，近似ではない最近傍探索アルゴリズムは全て，A*あるいは分枝限定法（これらは本質的に等価である）のアルゴリズムである．

Vol.2009-CVIM-169 No.13
2009/11/26

した枝刈りを行いながら，可能性のある解を全て調べるという計算が行われる．この計算は，最近傍になる可能性が高いものから順次調べていくため俗に priority search とも呼ばれる．次元数が高い場合には，この priority search の方が計算の大半を占めることになることが判明したのである．ここで言う「次元数」とは，データを表現するための次元数 D ではなくデータ分布を表現する次元数（ここではこれを Z と表す）のことである．Z が大きく，最悪の場合には，priority search での枝刈りは全く機能せず，全探索と同じ回数の距離計算を強いられることがある[3]. 多くの研究者はこの Z によって支配されない計算法を目指して様々なアルゴリズムを開発してきたが，いまだに実現できていない．

この問題に対する明確な結論は得られていないが，以下の事柄が多くの研究者によって信じられるようになってきた．

「分布を表現する次元数 Z が十分大きい場合，いかなる最近傍探索アルゴリズムも線形探索と同じかそれ以上の回数の距離計算を行う．」

注意すべきことは，これは証明された定理ではなく，最近傍探索の研究者が数々の実験を通じて得た仮説だということである．したがって，この仮説が間違っている可能性もあることにも注意して頂きたい．

データを表現する空間の次元と，データの分布の次元の間には，$Z \le D$ という関係がある．例えば，次元 D の空間で発生させた一様分布の次元数 Z は D と一致するが，次元 D の空間でより低い次元の空間にデータが偏在する場合には，$Z \le D$ となる．上の仮説は，この Z がいかなる最近傍探索アルゴリズムをもってしても超えることのできない壁，すなわち計算量に関与していることを意味している2．しかし，これまでのところ，この Z と計算量の関係を示した研究は存在しない．

そもそも，上の仮説がいかなる距離空間についても成立するのであれば，固定次元の空間だけではなく，距離の公理を満足しさえすれば，木構造やグラフ構造を要素とする距離空間でも成立するはずである．それゆえ，データ分布の次元数 Z は，累積寄与率のカーブ等からではなく，データ相互間の距離分布から導出されるべきものである．しかし，データ間の距離分布から，どのように Z を計算すれば良いかすらも分かってはいない．

以上が最近傍探索の研究を通じて浮かび上がってきた「仮説」であり，多くの研究者がこの成立を暗黙のうちに認めていることから，「性質」と呼んでもよい内容である．これは俗に「次元の呪縛」（curse of dimensionality）などとも呼ばれている3．最近傍探索の研究では，次元の呪縛の影響を受けない探索法を一度は目指し，それがどうやら不可能であることに気がつくと，次元の呪縛の影響を受けにくい近似最近傍探索法へと流れが変わってきた．これらのことから，**「次元の呪縛」こそが最近傍探索研究の原動力であり，それによって生じる様々な困難さがこの研究分野をより魅力的なものにしている**とも言える．しかし，これほどポピュラーな内容であるにもかかわらず，これを数学的な命題として表し，その成立を証明することはこれまで行われてこなかった．本稿では，チュートリアルとして，基本的な最近傍探索アルゴリズムを紹介しながら，この次元の呪縛の問題についても，可能な限りアプローチする．

2 Z 以外にもデータ数 N 等も計算量に関与している．

3 識別タスクでの次元の呪縛の問題とは関連はあるが，発生する具体的な問題は異なる．

2. 最近傍探索問題

最近傍探索問題は，数種類の類似問題に分類することができる．前章の式(1)で述べた解を求める問題は，最近傍データを 1 つ求める問題であるが，これを複数個求めるという問題も存在する．これは，k -最近傍探索問題と呼ばれ，1 番目から k 番目までの最近傍を求めるという問題である．このように最近傍を決められた個数だけ求めるのではなく，クエリからある一定距離 d の範囲内に存在するデータ全てを探してくる range search の問題も存在する．

k -最近傍探索は最近傍識別よりも汎化性能に優れた k -最近傍識別等の応用に用いられ，range search は Parzen 窓等の密度推定で用いられる．式(1)で述べた内容は，k -最近傍として見れば 1-最近傍探索であり，range search として見れば，d が可変でたまたま最近傍距離と一致するケースであると見なすこともできる．

3 章で紹介するアルゴリズムは，1-最近傍探索法として見ると，役に立ちそうにないものもあるが，k -最近傍探索，あるいは range search の手法として見れば優れた性能を発揮するものもある．この点に注目して頂きたい．

2.1 最近傍探索で用いる距離尺度と同族汎距離

下記に示す 3 つの公理を満足する尺度 $d(\boldsymbol{x},\boldsymbol{y})$ を一般に距離と呼ぶ．

$$P_{d1}(d) \equiv {}^{\forall \boldsymbol{x},\boldsymbol{y}} d(\boldsymbol{x},\boldsymbol{y}) \ge 0,\ \ d(\boldsymbol{x},\boldsymbol{x}) = 0 \tag{2}$$

$$P_{d2}(d) \equiv {}^{\forall \boldsymbol{x},\boldsymbol{y}} d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x}) \tag{3}$$

$$P_{d3}(d) \equiv {}^{\forall \boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z}) \ge d(\boldsymbol{x},\boldsymbol{z}) \tag{4}$$

例えば，ユークリッド距離

$$d_E(\boldsymbol{x},\boldsymbol{y}) = \| \boldsymbol{x} - \boldsymbol{y} \|_{L2}$$

は，上記の公理を満足するが，それを二乗した

$$d_E^2(\boldsymbol{x},\boldsymbol{y}) = \| \boldsymbol{x} - \boldsymbol{y} \|_{L2}^2$$

は P_{d3} の三角不等式を満足しないため，これを距離と呼ぶことはできない（例えば，$3+3>5$ であるが，$3^2+3^2<5^2$ となる．）．

しかし，以下に示すように，P_{d1} や P_{d3} を満足しない尺度の中に，距離尺度と密接な関係を持つものがある．

Vol.2009-CVIM-169 No.13
2009/11/26

上記 3 公理を満足するある距離 $d(\boldsymbol{x},\boldsymbol{y})$ に対して，P_{d2} および，以下を満足する尺度を $\rho(\boldsymbol{x},\boldsymbol{y})$ で表わす．

$$P_{\rho 1}(\rho;d) \equiv {}^{\forall \boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} d(\boldsymbol{x},\boldsymbol{y}) \le d(\boldsymbol{x},\boldsymbol{z}) \leftrightarrow \rho(\boldsymbol{x},\boldsymbol{y}) \le \rho(\boldsymbol{x},\boldsymbol{z}) \tag{5}$$

このような尺度を用いて全探索を行っても，距離 $d(\boldsymbol{x},\boldsymbol{y})$ のもとで最近傍探索を行ったことになる．事実，多くの高速な最近傍探索アルゴリズムは P_{d1} や P_{d3} に依存した計算は行っていない．

本稿ではこのような $\rho(\boldsymbol{x},\boldsymbol{y})$ を「汎距離」と呼ぶことにする．汎距離の集合のうち，P_{d1} と P_{d3} を満足する部分集合が距離であり，汎距離は距離の概念を一般化したものとなっていることが分かる．

$\rho(\boldsymbol{x},\boldsymbol{y})$ は，距離 $d(\boldsymbol{x},\boldsymbol{y})$ を

$$P_{m1}(m) \equiv {}^{\forall \delta \ge 0} m(x+\delta) \ge m(x) \tag{1}$$

を満足する関数 $m(\cdot)$ で変換することによって得られる．すなわち，

$$\rho_m(\boldsymbol{x},\boldsymbol{y};d) = m(d(\boldsymbol{x},\boldsymbol{y})) \tag{7}$$

である（証明略）．ここでは変換関数と元の距離尺度を明示するために明示的に $\rho_m(\cdot,\cdot;d)$ と表している．

このように距離尺度を変換した尺度が再び距離の公理を満足するケースに関して，以下の定理が成り立つ．

【定理１：凹単調増加関数による距離の変換】

関数 m に対して，性質 P_{m1} に加えて，

$$P_{m2}(m) \equiv m(0) = 0 \tag{8}$$

$$P_{m3}(m) \equiv {}^{\forall x,y,0\le\alpha\le 1} m(\alpha x + (1-\alpha)y) \ge \alpha m(x) + (1-\alpha)m(y) \tag{9}$$

という性質を課せば，m によって距離尺度を変換した結果は，距離の公理を満足する尺度となる．（証明は付録１）

実際に P_{m1}，P_{m2}，P_{m3} を同時に満足する変換関数 $m(x)=\sqrt{x}$ を用いると，$\sqrt{3}+\sqrt{3}>\sqrt{5}$ となり，三角不等式は満足される．

この一方で，以下の定理も成り立つ．

【定理２：凸関数による距離の変換】

$$P_{m4}(m) \equiv {}^{\forall x,y,0\le\alpha\le 1} m(\alpha x + (1-\alpha)y) < \alpha m(x) + (1-\alpha)m(y) \tag{10}$$

を満足する凸関数によって距離を変換した結果は，常に三角不等式 P_{d3} を満足しないわけではない．

【証明】

$m(\cdot)$ が P_{m1} を満足する単調増加関数である場合，逆関数 $m^{-1}(\cdot)$ も単調増加であり，

$$\rho_m(\boldsymbol{x},\boldsymbol{y}) = m(d(\boldsymbol{x},\boldsymbol{y}))$$
$$d(\boldsymbol{x},\boldsymbol{y}) = m^{-1}(\rho_m(\boldsymbol{x},\boldsymbol{y}))$$

という双方向の変換が可能である．ここで，$m(\cdot)$ が P_{m3} の凹性を有する場合，汎距離 $\rho_m(\boldsymbol{x},\boldsymbol{y})$ は距離の公理を満足する．この距離を逆変換した $m^{-1}(\rho_m(\boldsymbol{x},\boldsymbol{y}))$ は $d(\boldsymbol{x},\boldsymbol{y})$ と一致するため，これもまた距離である．これら 2 回の変換のうち 2 番目の変換で用いている凹の単調増加関数の逆関数 $m^{-1}(x)$ は凸単調増加関数であり，これを距離 $\rho_m(\boldsymbol{x},\boldsymbol{y})$ に適用することで距離尺度が得られるケースがあることは，「凸関数によって距離を変換した結果は常に三角不等式 P_{d3} を満足しない」という命題の反例となっているため，この命題は成立しない．

証明終り

以上を整理すると，以下のようになる．

- 最近傍探索には必ずしも距離そのものは必要なく，同族汎距離で良い場合がある．
- 同族の汎距離は，元の距離を単調増加関数で変換したものになっている．
- 凹単調増加関数によって距離を変換したものは距離の公理を満足する
- 凸単調増加関数で距離を変換した汎距離は常に三角不等式を満足しないわけではない．

これらの事実は，高次元空間における最近傍探索を困難にしている「次元の呪縛」を緩和するためのヒントを与えてくれるが，これに関しては後に詳しく述べる．

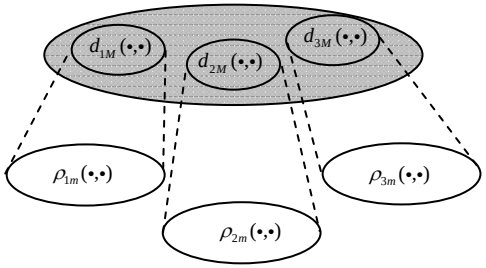
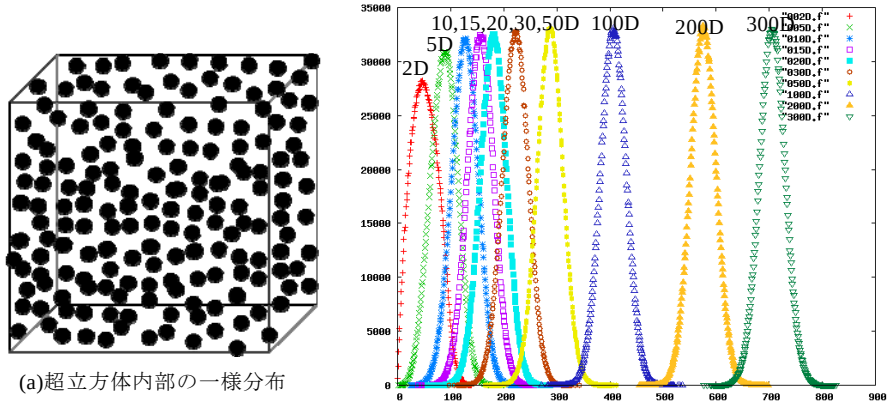


図 1．相互に変換可能な距離と汎距離の関係

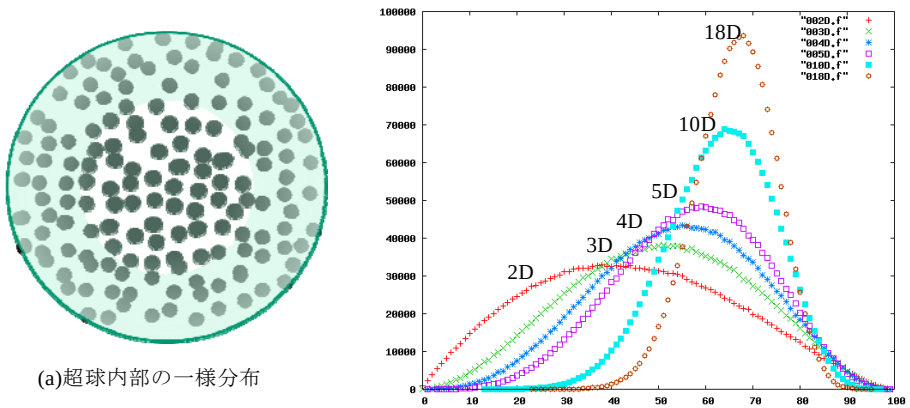
Vol.2009-CVIM-169 No.13
2009/11/26



(a)超立方体内部の一様分布

(b)相互距離の頻度分布

図 2. 超立方体内部に一様分布する点間の相互距離分布と次元に対するその変化



(a)超球内部の一様分布

(b)相互距離の頻度分布

図 3. 超球内部に一様分布する点間の相互距離分布と次元に対するその変化

2.2 高次元空間における距離分布の性質

実際の最近傍探索アルゴリズムを見る前に，高次元空間における最近傍探索が困難であるという事実を理解しておく必要がある．

クエリから各データまでの距離を計算したとき，

- 距離の大小差が大きく，ばらついていれば，効率よくクエリに最も近いデータを絞り込むことができる．
- 距離の大小差が小さく，クエリから遠方にデータがまとまって分布する場合には，クエリに最も近いデータを絞り込む計算の効率は落ちる．

ということが一般的に言える．これを確認するために，超立方体内部で一様乱数によって，ランダムな点を発生させ，これらの点相互のユークリッド距離分布に対する頻度分布を求めた．その結果，次元数の上昇に伴い点間の距離分布は図のように一定幅のまま距離が増加するという現象が起きる. この図では, 立方体の一辺は 100 である．

このような距離の頻度分布を，球の内部の一様乱数について求めると，図に示す形状になる．この球の直径は 100 である．

これらのグラフから，超立方体の場合も超球の場合も共通して次のことが言える．

1)次元が増えると点間の距離が増加する，つまり点と点の間が遠のく．

2)次元が増えても点間の距離のばらつきは広がらず，超球の場合はむしろ狭くなる．

一方，球と立方体で異なるのは次のことである．

3)球の場合は直径の制約があるため最遠点の上限が決まっているのに対して，立方体の場合は最遠点が次元数とともに増加する．

これらから，高次元空間では，クエリから見るとデータは全て遠方にあり，しかもそれらの距離がある一定の幅に収まっていることが分かる．このような状況では，クエリから見て最も近い点を効率よく絞り込むことは極めて困難になる．ただし，このような状況は立方体の方が顕著であり，超球の場合には顕著には現れないと言える．

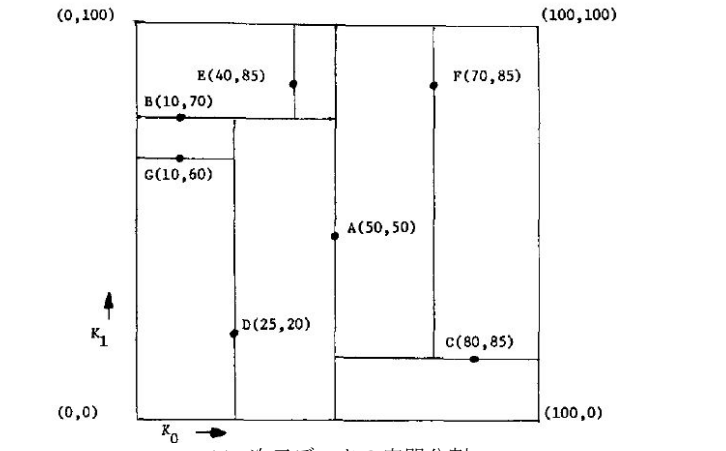
なぜ高次元空間では，こういった距離分布になるのかという問題については，付録 2 を参照のこと．

3. 最近傍探索アルゴリズム

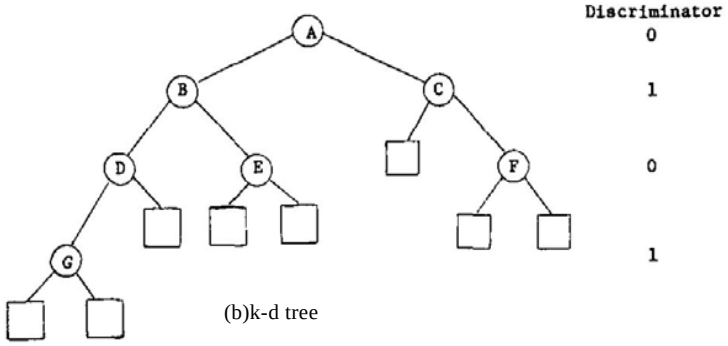
3.1 k-d 木

k-d木とは，1975年にACMが主催したGeorge E. Forsythe Student Paper Competitionで次席となった博士課程の学生John Louis Bentleyによる単著の論文で最初に提案されたデータ構造およびそれを用いたアルゴリズムである．Bentleyはこのデータ構造をassociative retrievalのためのものと説明しているが，ここで言うassociativeというのは，単一の検索キーを用いるのではなく，複数の検索キーを用いるという程度の意味である．実際，論文中では複数キーを用いたExact Match，Partial Match, Intersection Query, Region Query，などの検索法のうちの一つとして, Nearest Neighborの話を取り上げているにすぎず，k-d木を用いた最近傍探索については[2]で詳しく述べられている．

Vol.2009-CVIM-169 No.13
2009/11/26



(a)2 次元データの空間分割



(b)k-d tree

図 4. k-d 木とそれが表わす 2 次元平面分割（[1]より引用 ）

k-d 木は，空間分割を表わしており，探索空間は図4に示すようにデータの位置を通過する分割面によって区切られる空の箱に分解されている．これはk-d treeの分枝節にデータが格納されていることに相当する．木を構築する際には，分割次元は節の深さに応じて順に切り替わるようにし，データの中でその次元の中央値が分割点として選択されている．

最近傍探索を行う際には，k-d 木を2分探索木として利用し，各節の分割次元についてクエリと節のデータの大小関係を比較しながら葉まで到達する．葉に到達する直前の分枝節を暫定的な最近傍解とし，クエリと暫定解との距離を計算する．その後，バックトラックを行いながらクエリと各節の距離を計算し，暫定解よりも距離が近い節が見つかれば暫定解を更新する．このバックトラックの際に，クエリを中心とし，暫定解までの距離を半径とする超球が交わりを持つ分割面に関しては，再帰呼び出しで最近傍になるかどうかを確かめる．このように，バックトラックと再帰呼び出しを繰り返しながら根まで戻ってきた時点で探索が終了する．

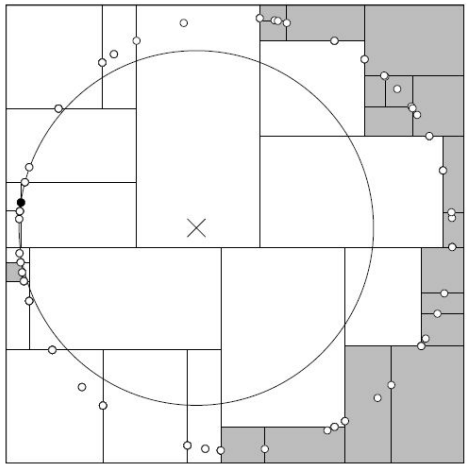


図 5. k-d 木の探索効率が落ちるとき（[7]より引用 ）

最初の2分探索は，データ数を N とすると，$O(\log_2 N)$ の計算量を必要とする．暫定解を求めた後にバックトラックと再帰呼び出しを起こすことから，実際の計算量はこれよりも多い筈であり，[3]では最悪時の探索に要する計算量が $O(k \cdot N^{1-1/k})$ となり，全探索になり得ることが示されている．図5では，円形に配置されたデータに対して，その中心にクエリを与えた場合に距離計算が行われるBoxが白で表わされており，距離計算が回避できるケースはほとんどないことを示している．

このアルゴリズムでは，葉に到達する直前の節に暫定的な最近傍が格納されているため，2分探索の段階で暫定的な最近傍として選ばれるのは，格納データの一部になる．このため，最近傍候補の絞り込み効率が悪いという問題点があった．これを改善するために全てのデータを葉ノードに蓄積するという方法が広く用いられるようになってきた[4]. この場合格納されるデータは，それぞれが 1 つだけ入るBoxに分けられるが，どのようにその箱を作るかというsplitting ruleが問題になる．[4]では，分割次元を最も広がりの広い（データをその次元に射影した時の最大値から最小値を引いた値が最も大きい）ものを選び，分割位置をデータの中央値とするstandard splitが採用されている．

Vol.2009-CVIM-169 No.13
2009/11/26

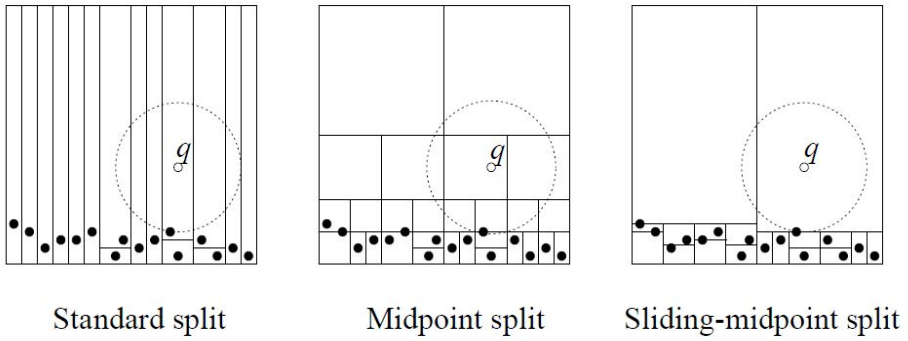


図 6. k-d 木とそれが表わす 2 次元平面分割（[5]より引用 ）

しかし，これでは図6に示すように，縦横比が極端なBoxが現れ，2分探索後にチェックする箱の個数が増加するため，効率のよい計算が行えない．この問題を解決するため，[5]では分割次元の選択は同じで，その軸の中央を分割するMidpoint splitルール，およびMidpoint splitで空のBoxが生成された場合に，必ず 1 つのデータが含まれるように分割面をスライドさせる Sliding-midpoint splitルールが提案された．図6を見て分かるように，Sliding-midpoint splitが最も矩形との交わりが少なくなり，2分探索後のサーチも効率的に行える．これを採用したANN[6]では，後述する近似計算だけでなく，さらに

- 2分探索時に各節でクエリと分割面までの距離を記録しておき，この値が最も小さい順に各Boxのチェックと暫定解の更新を行う．
- Boxのチェックを行う際には，暫定解とクエリとの距離（暫定距離）を参照し，Box内のデータとクエリとの距離が暫定距離を超えた時点で距離計算を打ち切る

といった工夫が凝らされており，より高速な最近傍探索が行えるようになっている．

3.2 AESA,LAESA

k-d 木では，格納されるデータはベクトルで表現されており，各座標値にアクセスできるということが前提条件になる．この条件が成立しない場合，すなわち，各データはベクトルでないため，その構成要素も分からないが，各データ間の距離は与えられているというケースでは，空間の Box 分割は用いることができない．k-d 木の探索アルゴリズムが三角不等式を用いることなく実現されていたのに対して，AESA[8]は，距離の三公理を積極的に用いた最近傍探索アルゴリズムである．

このアルゴリズムでは，全格納データ P 間の距離を事前に計算して表にまとめておき，その中からクエリに対する最近傍とはなり得ないものを除外していくというアルゴリズムである．ここでは，図 7 の記号に合わせ，クエリを x で表わす[4]．P の部分集合である「テスト集合」U を考え，U の各要素と x の間の距離を全て計算したとき，最短距離を与えるデータ n が見つかったとする．もちろん，$U \subset P$ であるので，n は P 中の最近傍とは言えない．ここで，s を U のある要素とすると，P の中で x の最近傍になり得ない集合 E は次式で表わせる．

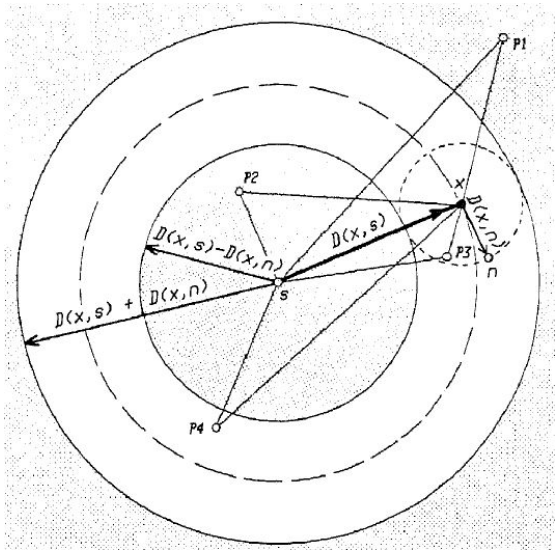


図 7. AESA における候補の削減原理（[8]より引用 ）

$$E = P - \{ p \mid D(x,s) - D(x,n) < D(p,s) < D(x,s) + D(x,n) \} \tag{11}$$

これは，図 6 のドーナッツ状の領域外の格納データを表わしている．

各点 u から半径 $D(x,u)$ の超球を描き，その交わりの最も近くにあるデータを求めれば最近傍を求めることはできるが，球が多数になればこの交わりはクエリ x と一致してしまう．この球の交わりから最も近いデータを求める計算を次式で近似する．

$$s = \arg\min_{q \in P-E} \sum_{u \in U} | D(q,u) - D(u,x) | \tag{12}$$

AESA では，この計算で求められる近似最近傍点を s とし，式(11)の適用によって候補を減らしていく． s の選択は図 8 のように起き，実際のアルゴリズムは図 9 のように

4 ベクトルではないため，太字も使わない．

Vol.2009-CVIM-169 No.13
2009/11/26

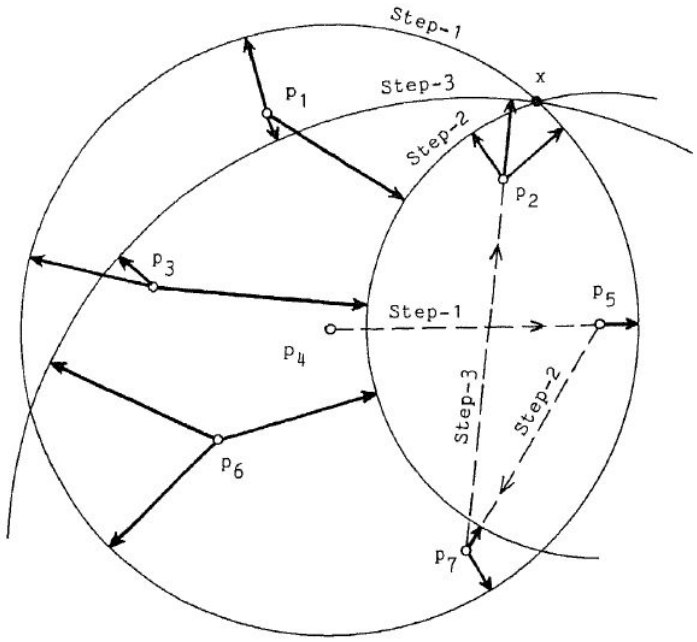


図 8. AESA における s の選択（[8]より引用 ）: 太い矢印は，各ステップで式(12)の右辺最小化する差ベクトル． s は P_4, P_5, P_7 の順に選択される

なる． 最初，円の中心 s は適当に選ぶ（これを c とする）．これを用いて式(12)を計算し，次の円の中心を求めるという計算を反復するのである．

AESA の問題点は，蓄積するデータ間の距離をすべて記憶しておかなければならないため，前処理に時間がかかり，探索時にメモリを大量に消費することである．これらはデータ量の 2 乗に比例して増加する. これを減らし, 蓄積データの一部である Base Prototype についてのみ相互距離計算をしておくだけで最近傍探索が行えるのが LAESA[9]である．

これらのアルゴリズムは，ともに「距離計算回数を削減する」ことを目的とする分枝限定法（A*アルゴリズム）である．しかし，距離計算回数を減らす手掛かりを，クエリと幾つかのデータとの距離計算結果から得ているため，探索時に行う距離計算の回数は必然的に多くなる. また, 距離計算の打ち切り等のテクニックは用いていない. 巨大な表のスキャンを行わなければならない．以上の理由で，実際には k-d 木ほどの速度向上は得られない．

Initialization:
$s := c;$
$U := \{c\};$
$n := c;$
$E := \{c\} \cup \{q \mid q \in P - \{c\} and (D(q,c) \geq 2D(x,c))\}$
Mainloop:
while$(E \neq P)$**do**
$s := \arg\min_{q \in P-E} \sum_{u \in U} |D(q,u) - D(x,u)|;$
$U := U \cup \{s\};$
$E := E \cup \{s\};$
$n := \arg\min_{q \in \{n,s\}} (D(x,q));$
if $n = s$ **then** $Q := U$ **else** $Q := \{s\};$
for each $q \in Q$ **do**
$E := E \cup \{p \mid p \in P - E$ **and**
$(D(p,q) \geq D(x,q) + D(x,n)$ **or** $D(p,q) \leq D(x,q) - D(x,n))\}$
endfor
endwhile

図 9. AESA の基本アルゴリズム（[8]より引用・抜粋 ） 赤枠の部分のみで距離計算を行っている．

3.3 Voronoi 分割，k-means クラスタリングの利用

Geometric Near-neighbor Access Tree (GNAT)[11]は蓄積データの幾つかをピックアップし，これらを母点とする Voronoi 領域（Dirichle 領域）に全データを分割し，各領域内のデータをさらに同じ方法で分割していくというデータ構造とそれを用いた最近傍探索アルゴリズムである（図 10 左）．但し，高次元空間では母点の個数を絞り込まなければ，Voronoi 分割を行うこと自体困難になる．これに対して，近年，類似画像検索で David Nister らが用いている Vocabulary tree[12]は，k-means クラスタリングを反復適用して得られる木構造である（図 10 右）. もともと, この木構造は Fukunaga と Narendra によって最近傍探索を行うためのデータ構造として提案されたものであり[13]，分枝限定法と組み合わせた探索アルゴリズムで利用された．後者のアルゴリズムでは，クラスタリングを行うため，データ間の加算が定義されていない距離空間内のデータには適用できないという問題点がある．

Vol.2009-CVIM-169 No.13
2009/11/26

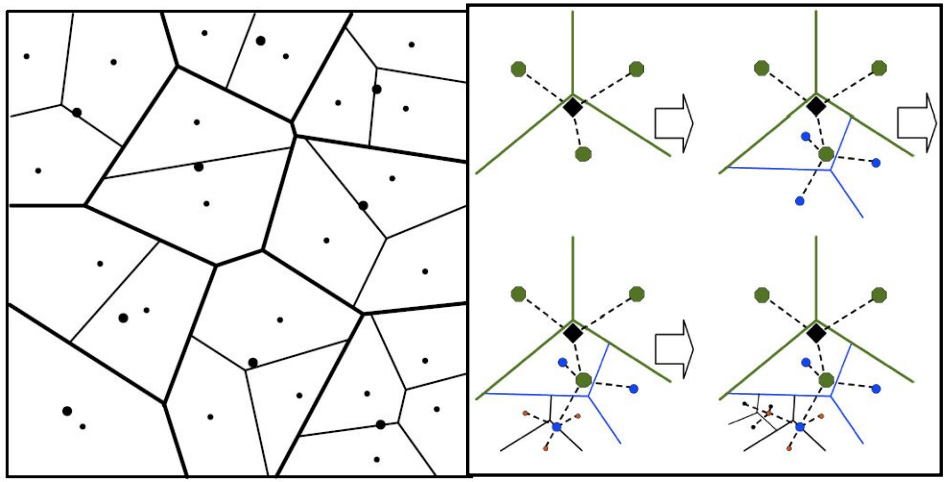


図 10. Simple Gnat(左：[11]より引用)と Vocabulary tree（右：[12]より引用）

これらのアルゴリズムは，AESA や LAESA と同じく，距離計算結果から距離計算対象を絞り込むというものである．したがって，正確な最近傍探索を行うための計算量は多い．但し，バックトラックを起こさない近似最近傍探索での利用価値はまだあるかもしれない．これらのアルゴリズムの分割面は単一の距離計算では得られないため，どの Voronoi 領域に属すかを決定する計算は，母点の数が増え，高次元になるほど，増加していくという問題がある．

3.4 VP 木，MVP 木

距離空間において空間分割木を構築する際に，データ群の中で見晴らしの良い点（Vantage point）を見つけ，その Vantage Point 空の距離が閾値以上か以下かでデータを 2 分するデータ構造を，VP 木[14]と呼ぶ．この方法では，空間を分割する境界は球面となり，図 11 に示すように k-d 木とは全く異なる分割結果が得られる．この手法は，k-d 木がベクトル空間用のデータ構造であったのに対して，VP 木は距離空間用の木構造であるとも言える．探索時の分岐操作に必要な距離計算回数は GNAT や k-d 木を用いる場合よりも少ないが，木の深さは深くなる．しかし，原理的に考えて，Voronoi 分割や k-means クラスタリングを行う場合よりも計算量的には少なくなるはずである．さらに，VP 木を改良し，木を辿るために用いた距離計算結果を利用して，木を辿った後の絞り込み計算を効率化する MVP 木[15]などの変種も存在する．

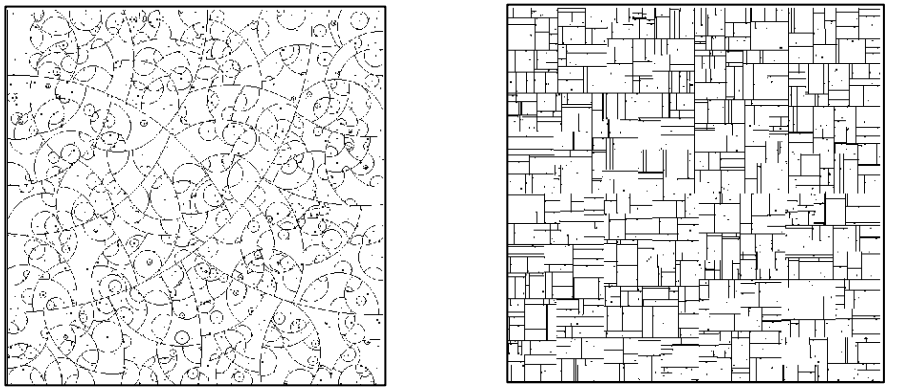


図 11. VP 木(左：より引用)と k-d 木（右：[14]より引用）

4. 近似最近傍探索

以上に述べたアルゴリズムは，すべて厳密な意味での最近傍を探索するものであり，例外なく「A*アルゴリズム」あるいは「分枝限定法」と呼ばれる組み合わせ最適化アルゴリズムの一種であった．この種のアルゴリズムは，緩和解を求めるにあたって，目的関数の下界の見積もりをどの程度正確に行うかによって性能は大きく変化する．これまでの研究は，結局こういった性能の微調整だったと見なすこともできよう．

これらのアルゴリズムは 1)initial guess と 2)verification の 2 段階から成っていると見なすこともできる．このように考えると，実用的には 2)の処理をいい加減に行うことで計算量を減らしつつ，大幅な精度低下を引き起こさない「近似最近傍探索」が有用である応用が多く，近年よく研究されるようになっている．以下では，この研究の概要について述べる．基本的には，最近傍探索アルゴリズムが「A*アルゴリズム」あるいは「分枝限定法」である以上，全ての正確な最近傍探索アルゴリズムには，それを簡略化した近似最近傍探索が存在する．ここでは，代表的な研究例，もしくは正確なアルゴリズムがもともと存在しなかったケースについてのみ説明する．

4.1 ANN

これは，3.1 で述べた k-d 木を基本的なデータ構造とする近似最近傍探索の方法である．3.1 で述べたように，k-d 木は一旦葉ノードに到達すると，暫定解とクエリとの距離を半径とする超球をクエリを中心にして描き，この球と交わりを持つ Box 内のデータに関してクエリとの距離計算を行う．図 5 は，この超球の半径が大きくなりすぎて，多

Vol.2009-CVIM-169 No.13
2009/11/26

くのデータとの距離計算をしなければならなくなった状況を表わしている．この例のように，超球の半径を小さくすれば，超球と重なる Box の個数が減るので，後段の verification が高速化する．ANN では，正確な探索に必要な半径が r であったとき，それを 1 より大きな定数 ε を用いて，$r'=r/\varepsilon$ を新たな半径として verification が行われる．この方法のメリットは，得られる近似最近傍の許容誤差（最近傍と近似最近傍の間の誤差）がコントロールできることである．具体的には許容誤差の大きさは，$r(\varepsilon-1)/\varepsilon$ と表わすことができる．しかし，このような誤差の許容量を導入したことでどれだけ計算量が減るのかは，球と Box の配置で変化するため，明確に議論をすることは困難である．これは，精度と速度の関係を理論的に明らかにできないということを意味している．

4.2 LSH

これに対し，計算量と精度の関係を議論することができる理論的枠組が Indyk らによって示された[18]．これは，複数のハッシュ関数を用いて，クエリから複数のハッシュ値を求め，ヒットしたハッシュ・ビンに格納されていたデータ集合についてクエリとの距離計算を行い，近似最近傍を求める手法である．このときに用いるハッシュ関数 $h(q)$ が，次のような条件を満足するとき Locality Sensitive であると言う．

$$\begin{aligned} D(v,q) \le r_1 &\Rightarrow \Pr[h(q)=h(v)] \ge p_1 \\ D(v,q) > r_2 &\Rightarrow \Pr[h(q)=h(v)] < p_2 \end{aligned} \tag{13}$$

ただし，$p_2 \le p_1$，$r_2 = cr_1$，c は定数，である．N を格納するデータ数としたとき，近似最近傍探索は下記の回数のバケット内の探査で終了することが証明されている．

$$L = N^{\rho(c)} = N^{p_1/p_2} \tag{14}$$

これだけでは，具体性が無く，実際のハッシュ関数も示されていないので，具体的な性能については議論できない．

実用的に用いられる LSH は p-stable LSH[19]と呼ばれ，$O(\rho(c))=1/c$ という性能指標を持つ．これは，c の値が 1 よりかなり大きくなれば $r_2 > r_1$，$p_1 > p_2$ の傾向は強まり，最近傍になる可能性が高いものを少数だけ調べようとする．この結果，速度は向上するが精度は低下する．逆に c を減らせば，多くの候補を生成しておき，それを全て吟味するように動作するため，速度は遅いが精度は向上する．すなわち，横軸を時間，縦軸を誤差として描いた速度対精度の曲線と $1/c$ が本質的に等価になるのである．P-stable LSH は，等方性の Gauss 関数からランダムにサンプリングした複数本の軸に入力ベクトルを射影し，量子化するというハッシュ関数を用いている．この Gauss 関数が p-stable であるため，$O(\rho(c))=1/c$ という性質が導出できるのである．

我々は，p-stable LSH の射影軸を格納するデータの PCA で求め，軸上の周辺分布の累積確率から，ビンに格納されるデータの個数の期待値を一定にした

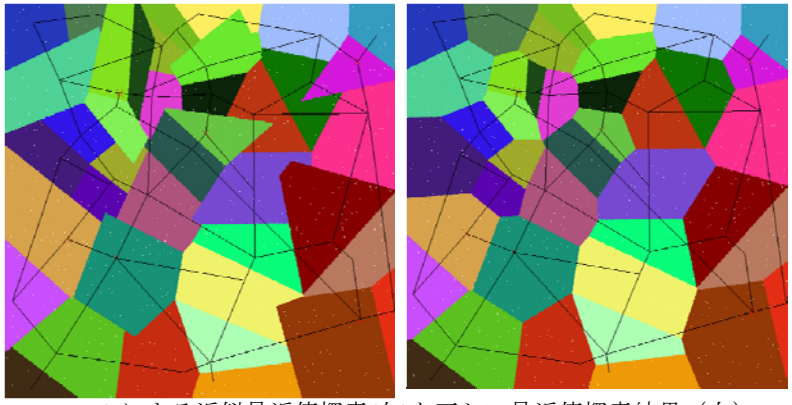


図 12．NFTG による近似最近傍探索(左)と正しい最近傍探索結果（右）

Principal Component Hashing (PCH)を提案している[20]．PCH では，格納したデータを各主軸上に射影したデータをあらかじめ持っているため，距離計算の打ち切りが行いやすい．しかも，LSH のようにクエリによって計算時間がかかりすぎたり，逆にヒットしたビンにデータがなく，全く候補が生成できないまま探索が失敗することが無く，基本的にはクエリの種類に依存せず，一定速度で近似最近傍候補が見つかる．PCH はデータおよびその主軸への射影の分布が Gaussian にしたがうことを仮定している．この仮定を緩め，周辺分布のヒストグラムと累積ヒストグラムからハッシュビンへの分解を行う A-PCH も提案されてはいるが[20]，射影軸の方向は依然として PCA で計算されているのが実情である．

最近，k-d 木を構築する際の分割軸候補を常に複数本持っておくようにし，その中から一つを乱数で選択して，変動を含む k-d 木を複数作成する方法がよく研究されている．この手法で生成される複数の k-d 木を Random forest と呼ぶ[21]．Random forest は，それらの根集合を起点とする複数の 2 分探索に用いられ，葉に到達した際に得られる最近傍候補を対象として距離計算を行い，近似最近傍を求める手法である．この手法は，一つの k-d 木を用いた 2 分探索を 1 つのハッシュ関数と見なせば，完全に LSH の枠組みにはまっている．

4.3 NFTG

ここまでは，表や木構造といったインデックス構造を利用した最近傍探索について述べてきたが，直前の最近傍探索結果が次の探索の初期値として用いることができる場合などには，グラフ構造を用いた方が効果的な場合もある．我々は，「ある頂点から出発し，それに隣接する頂点とクエリ間の距離を比較して，よりクエリとの距離が短い

頂点に異動して，同じ計算を繰り返す」Nearest First Traversing（NFT）によって最近傍探索を行う方法を提案している[22]．これはグラフ構造をインデックス構造として画像上の対象追跡に最近傍探索を利用した際に用いた最近傍探索法である．この際に用いるグラフ構造は，Delaunay グラフが理論的には最も正しく，上述の NFT アルゴリズムで正確な最近傍探索が行える．しかし，Delaunay グラフは 5 次元程度の比較的低い次元の空間でも作成するのが困難であること，高次元では一つの頂点と他の頂点を繋ぐ辺の本数が多くなるため，距離計算回数が増加するなどの問題がある．このため，クエリがグラフ上のいずれかの頂点と一致している場合には必ず正しい探索が行えることのみを保証した NFTG が提案された．図 11 は，2 次元の最近傍探索結果であるが，見つけられた頂点 ID に固有の色でクエリを色付けしている．このため，正しい最近傍探索の場合には Voronoi 分割と一致し，近似最近傍探索では多少の違いが生じ得る．ただし，各頂点のすぐ近辺に与えられた頂点はほぼ正しい結果と一致していることが確認できる．

5. 次元の呪縛を解くために

次元の呪縛の問題に再び議論を戻そう．2 章，3 章の議論で，AESA などのアルゴリズムでは三角不等式の制約が使われるが，k-d 木や GNAT，k-means tree, VP tree などでは三角不等式は用いられない．したがって，定理 1 で述べた凹関数（上に凸な関数）による距離変換については，これら三角不等式を用いないアルゴリズムを利用する際には考える必要はなく，同族汎距離で最近傍探索が行える．したがって，凸関数で距離を変換した場合，図 2 や図 3 に示した距離の特異的な分布が緩和され，分布を原点近くに戻すことができると考えられる（本稿初出）．効果の大小は結果を待たなければならないが，次元の呪縛をある程度はいなすことはできそうである．

6. まとめ

以上，最近傍探索について，基本原理と，若干の考察，次元の呪縛，各アルゴリズム，そして次元の呪縛を解くための方針について述べた．

冒頭にも述べたが，次元の呪縛は，最近傍探索を研究する多くの人が一度は経験する困った現象である．特に最近では，SIFT や SURF などの高次元局所特徴の多用で，この現象に出くわす機会が増えているはずである．「こんなものが無ければ楽なのに」と考えれば，その通りだが，そうであれば最近傍探索はただの解きやすい問題ということになってしまう．困難さや敵があるから刺激的なストーリーができるのである．これがなければ，本当につまらない研究分野になってしまうだろう．

チュートリアルでありながら，こういったことについて考察し，何らかの方針が打ち出せたことは幸いである．というのも，長い研究の歴史の中でだれも呪縛を解くことができず，近似最近傍探索という「楽な」方向に傾きつつある現在の研究に何らかの影響を与えられるのではないかと思うからである．

最も近いものを探す最近傍探索の研究は，実はその問題の単純さからは想像ができないほど奥深く，荘厳な自然の摂理が支配する神聖な研究分野である．本稿を通じてこの分野に興味を抱いていただき，この分野の新たな研究に取り組んでいただける方が一人でも増えれば幸いである．

参考文献

k-d tree

[1] Bentley, J.L., “Multidimensional binary search trees used for associative searching,” Communications of the ACM, Vol.18, No.9, pp.509-517, 1975

[2] Friedman, J.H., Bentley, J.L., and Finkel, R.A. An algorithm for finding best matches in logarithmic time. Stanford CS Rep.75-482.

[3] Lee, D. T.; Wong, C. K. (1977), "Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees", *Acta Informatica* **9** (1): 23–29, doi:10.1007/BF00263763

[4] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for _nding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3(3):209{226, 1977.

[5] S. Maneewongvatana and D. M. Mount. It's okay to be skinny, if your friends are fat. 4th Annual CGC Workshop on Comptutational Geometry, 1999.

[6] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. ``An optimal algorithm for approximate nearest neighbor searching," In Proc. 5th ACM-SIAM Sympos. Discrete Algorithms, pp. 573-582, 1994.

[7] Andrew Moore, PhD Thesis “Efficient Memory_based Learning for Robot Control”, PhD Thesis; Technical Report No209; Computer Laboratory, University of Cambridge, 1991

AESA

[8] Vidal, R., “An algorithm for finding nearest neighbor in (approximately) constant average time,” Pattern Recognition Letters, No. 4, pp. 145-158, 1986

LAESA

[9] Mico, L., Oncina, J., and Vidal, E., “A new version of the nearest-neighbor approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements,” Pattern Recognition Letters, No. 15, pp. 9-17, 1994

k-means tree
[10] Fukunaga, K. and Narendra, P. M. “A branch and bound algorithm for computing k-nearest neighbors,” *IEEE Trans. Comput.*, 24, pp. 750–753. 1975
GNAT
[11] Brin, S., “Near neighbor search in large metric spaces,” in Proc. of 21st Conf. on very large database , Zurich, Switzerland, pp. 574-584, 1995
Vocabulary tree
[12] Nister, D. and Stewenius, H. "Scalable Recognition with a Vocabulary Tree," Proc. of cvpr, vol. 2, pp.2161-2168, 2006
K-means tree
[13] Fukunaga, K. and Narendra, P. M. “A branch and bound algorithm for computing k-nearest neighbors,” *IEEE Trans. Comput.*, 24:750–753, 1975
VP-tree
[14] Yianilos, P. Y., “Data structures and algorithms for nearest neighbor search in general metric spaces,” in Proc. of the Fourth Annual ACM-SIAM Symp. on Discrete Algorithms, Austin, TX, pp. 311-321, 1993
MVP-tree
[15] Bozkaya, T. and Ozsoyoglu, M.. Distance-based indexing for high dimensional metric spaces. In Proceedings of the 1997 ACM SIGMOD, pages 357--368. ACM, 1997
ANN
[16] Arya, S., Mount, D.M., Netanyahu, N. S., Silverman, R., and Wu, A. Y., “An optimal algorithm for approximate nearest neighbor searching,” Journal of the ACM, Vol.45, pp. 891-923, 1998
[17] ANN: Library for Approximate Nearest Neighbor Searching (http://www.cs.umd.edu/~mount/ANN/)
LSH
[18] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multidimensional spaces. In Proceedings of the 29th ACM Symposium on Theory of Computing, pages 618--625, 1997.
[19] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In Proc. of the 20th ACM Symposium on Computational Geometry, pp. 253-262, 2004.
PCH
[20] Y. Matsushita and T. Wada, “Principal Component Hashing: An Accelerated Approximate Nearest Neighbor Search,” Proc. of PSIVT 2009, pp. 374-385, 2009
Random Forest
[21] Silpa-Anan, C. and Hartley, R. “Optimised KD-trees for fast image descriptor matching”, Proc. of CVPR, Digital Object Identifier: 10.1109/CVPR.2008.4587638, 2008
NFTG
[22] Sakagaito, J. and Wada, T. "Nearest First Traversing Graph for Simultaneous Object Tracking and Recognition," Proc. of cvpr, pp.1-7, 2007

付録１．定理１の証明

m が P_{m1}，P_{m2} を満足することから，距離 $d(\boldsymbol{x},\boldsymbol{y})$ を変換した $\rho_m(\boldsymbol{x},\boldsymbol{y};d)$ は，

$$\rho_m(\boldsymbol{x},\boldsymbol{y};d)\geq 0, \rho_m(\boldsymbol{x},\boldsymbol{x};d)=0$$
$$\rho_m(\boldsymbol{x},\boldsymbol{y};d)=\rho_m(\boldsymbol{y},\boldsymbol{x};d)$$

を満足し，性質 P_{d1}，P_{d2} は満たされる．

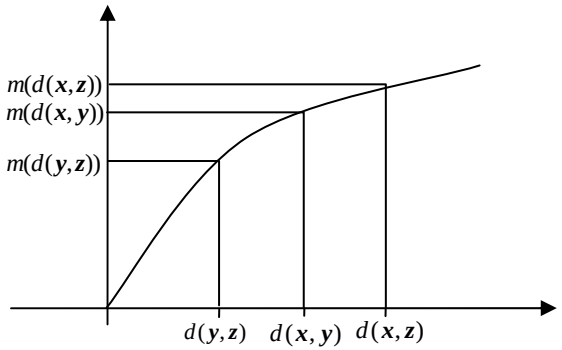
次に，$\rho_m(\boldsymbol{x},\boldsymbol{y};d)$ が三角不等式 P_{d3} を満足することを示す．

1) $d(\boldsymbol{x},\boldsymbol{y})\geq d(\boldsymbol{x},\boldsymbol{z})$ または，$d(\boldsymbol{y},\boldsymbol{z})\geq d(\boldsymbol{x},\boldsymbol{z})$ が成立する場合
m の単調性から
$\rho_m(\boldsymbol{x},\boldsymbol{y};d)\geq\rho_m(\boldsymbol{x},\boldsymbol{z};d)$ または $\rho_m(\boldsymbol{y},\boldsymbol{z};d)\geq\rho_m(\boldsymbol{x},\boldsymbol{z};d)$ が成り立つ．
また，P_{m1}，P_{m2} から $^{\forall x}m(x)\geq 0$ が成り立つので，$\rho_m(\boldsymbol{x},\boldsymbol{y};d)\geq 0$ かつ $\rho_m(\boldsymbol{y},\boldsymbol{z};d)\geq 0$ である．
したがって，三角不等式
$\rho_m(\boldsymbol{x},\boldsymbol{y};d)+\rho_m(\boldsymbol{y},\boldsymbol{z};d)\geq\rho_m(\boldsymbol{x},\boldsymbol{z};d)$
が成り立つ．

2) $d(\boldsymbol{x},\boldsymbol{y})=0$ または $d(\boldsymbol{y},\boldsymbol{z})=0$ の場合
この場合，d に関して三角不等式が成立することから，
$d(\boldsymbol{y},\boldsymbol{z})\geq d(\boldsymbol{x},\boldsymbol{z})$ または $d(\boldsymbol{x},\boldsymbol{y})\geq d(\boldsymbol{x},\boldsymbol{z})$ が成立する．これは 1)の条件と同じであるので，ρ_m に関しても三角不等式は成立する．

3) $d(\boldsymbol{x},\boldsymbol{z})>d(\boldsymbol{x},\boldsymbol{y})>0$ かつ $d(\boldsymbol{x},\boldsymbol{z})>d(\boldsymbol{y},\boldsymbol{z})>0$ の場合
m が P_{m1}，P_{m2}，P_{m3} を満足する，すなわち原点を通過する単調増加の凹関数であるので，明らかに

Vol.2009-CVIM-169 No.13
2009/11/26



$$\forall x \geq y > 0 \quad \frac{m(y)}{y} \geq \frac{m(x)}{x} \tag{15}$$

が成り立つ（上図参照）.
$0 < d(\boldsymbol{x},\boldsymbol{y}) < d(\boldsymbol{x},\boldsymbol{z})$ かつ $0 < d(\boldsymbol{y},\boldsymbol{z}) < d(\boldsymbol{x},\boldsymbol{z})$ であるので，(15)から次の 2 式が成り立つ.

$$\frac{m(d(\boldsymbol{x},\boldsymbol{y}))}{d(\boldsymbol{x},\boldsymbol{y})} \geq \frac{m(d(\boldsymbol{x},\boldsymbol{z}))}{d(\boldsymbol{x},\boldsymbol{z})} \tag{16}$$

$$\frac{m(d(\boldsymbol{y},\boldsymbol{z}))}{d(\boldsymbol{y},\boldsymbol{z})} \geq \frac{m(d(\boldsymbol{x},\boldsymbol{z}))}{d(\boldsymbol{x},\boldsymbol{z})} \tag{17}$$

また，

$$d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z}) \geq d(\boldsymbol{x},\boldsymbol{z})$$

が成立し，$m(d(\boldsymbol{x},\boldsymbol{z}))/d(\boldsymbol{x},\boldsymbol{z}) > 0$ なので，

$$\frac{m(d(\boldsymbol{x},\boldsymbol{z}))}{d(\boldsymbol{x},\boldsymbol{z})}\big(d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z})\big) \geq \frac{m(d(\boldsymbol{x},\boldsymbol{z}))}{d(\boldsymbol{x},\boldsymbol{z})} d(\boldsymbol{x},\boldsymbol{z})$$

も成り立ち，したがって，

$$\frac{m(d(\boldsymbol{x},\boldsymbol{z}))}{d(\boldsymbol{x},\boldsymbol{z})}\big(d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z})\big) \geq m(d(\boldsymbol{x},\boldsymbol{z})) \tag{18}$$

が得られる．不等式(18)および，(16)(17)から

$$\frac{m(d(\boldsymbol{x},\boldsymbol{y}))}{d(\boldsymbol{x},\boldsymbol{y})} d(\boldsymbol{x},\boldsymbol{y}) + \frac{m(d(\boldsymbol{y},\boldsymbol{z}))}{d(\boldsymbol{y},\boldsymbol{z})} d(\boldsymbol{y},\boldsymbol{z}) \geq m(d(\boldsymbol{x},\boldsymbol{z}))$$

が成り立ち，これを整理すると，

$$m(d(\boldsymbol{x},\boldsymbol{y})) + m(d(\boldsymbol{y},\boldsymbol{z})) \geq m(d(\boldsymbol{x},\boldsymbol{z}))$$

すなわち，

$$\rho_m(\boldsymbol{x},\boldsymbol{y};d) + \rho_m(\boldsymbol{y},\boldsymbol{z};d) \geq \rho_m(\boldsymbol{x},\boldsymbol{z};d)$$

が得られ，三角不等式 P_{d3} が成立する.

証明終り