LSHスキームによる最近傍探索アルゴリズムのいくつかの変形と高性能化

小林 卓夫　清水 郁子　東京農工大学
{ tkobaya, ikuko } @cc.tuat.ac.jp

2008.11.6改訂

目的とアプローチ

LSHスキームによる近似最近傍探索(ANNS)アルゴリズムの派生形を作成し，高性能なアルゴリズムを実現する．

性能の良いハッシュ関数の選択．

高次元の球面空間におけるANNS問題を対象とする．

LSHスキームによるANNSアルゴリズムの基本形

$H \in F$: ハッシュ関数の族，sim : 適当な類似度関数に対して，$sim(x,y) = \Pr(H(x) = H(y))$ となるとき，$\max\{sim(q,p_i)\}$ である p_i が q の最近傍である．

探索アルゴリズム：$\sum_{l=1}^{L} \psi(H(q) = H(p_i)) \geq 1$ である $\{p_i\}$ に対して，$\max\{d(q,p_i)\}$ を計算する．

ここで，$\psi(Q) = 1(Q \text{ is true}), 0\ (Q \text{ is false})$ とする．

派生アルゴリズムの作成

アルゴリズムの作成

$P = \{p_i\}$: prototypes，q : query，

$P(q,k)$: a set of k closest point(s) in P from q

$P(q;r)$: qから半径 r 以内の P の点の集合

$A = \{a_h\}$: 適当な点集合

(A1)： $sim(q,p_i) = \Pr(A(q,1) \cap A(p_i,1) \neq \phi)$

⇓

(A2)： $sim(q,p_i) = \Pr(A(q,k_q) \cap A(p_i,k_p) \neq \phi)$

⇓

(B1)： $sim(q,p_i) = \Pr(A(q;r_q) \cap A(p_i;r_p) \neq \phi)$

⇓

(B1')： $sim(q,p_i) = \Pr(p_i \in \bigcup_{a_h \in A(q;r_q)} P(a_h;r_p))$

⇓

(B2)： $sim(q,p_i) = \Pr(p_i \in \bigcup_{a_h \in A(q,k_q)} P(a_h,k_p))$

ハッシュ関数の作成

$x = (x_1, \ldots, x_d)$

Algorithm－(A1,A2)で用いるハッシュ関数：

$A = \{a_h\} = \{-1,+1\}^{d'}$ $(1 \leq d' \leq d)$ のとき，

$A(x,1) = \{ (sign(x_1), \ldots, sign(x_{d'})) \}$，

$H(x) = sign(x_1) \ldots sign(x_{d'})$

Algorithm－(B2)で用いるハッシュ関数：

$|x_{k_1}| > |x_{k_2}| > \ldots > |x_{k_{d'}}| > \ldots > |x_{k_d}|$ とする．

$R_h = \{ x \mid h = sign(x_{k_1})k_1 \ldots sign(x_{k_{d'}})k_{d'} \}$

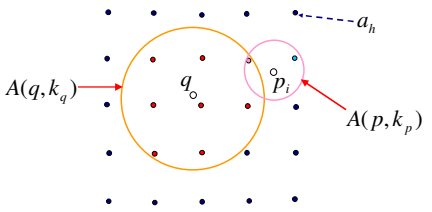
ただし，$^{\forall} v \in R_h$ に対して，

$a_h \ s.t. \min E[d(v,a_h)]_{v \in R_h}$ とするとき

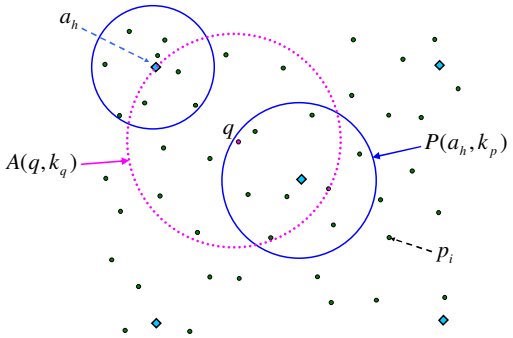
$\Pr(P(v,1) \in P(a_h,k_p)) \geq \alpha$ となるように

d' を決定する．

アルゴリズムーA2 (k_q=8, k_p=2)



アルゴリズムーB2 (k_q=2, k_p=8)



評価

乱数により人工的にテストデータを生成し，実験を行う．

探索精度の定義：**探索プログラムが返す点集合を*CAND*とするとき，**

探索精度 $\alpha := E[|P(q,k) \cap CAND| / |P(q,k)|]$

評価項目：**探索精度，探索時間，メモリサイズ**

ファクター：**次元数，プロトタイプ数，分布型，パラメータ (*kq*, *kp*, *A*)**

Kd-treeによるプログラム ANN Library と性能を比較．

実験環境：**Dell VOSTRO-200, Intel Pentium Dual CPU E2160 1.80GHz, 3.24GB RAM, MS Windows XP Professional Ver.2002 SP2, プログラム言語**：C++(MS-C/C++ v.12)

データの生成方法：x_i:正規分布乱数，σ_i:標準偏差，T:ランダムな直交行列

$v = T \times (\sigma_i x_i)^t$，　$u = v/|v|$　　σ_i=1 (uniform)，$\sigma_i = i$ (skewed)

(A2)は*kq*を増加させることにより探索性能が向上する場合がある．また，必要メモリサイズが大幅に減少できる．(A1),(A2)は，既存手法と比べて，大きい次元($d \geqq 20$)・プロトタイプ数で特に効果が高い．

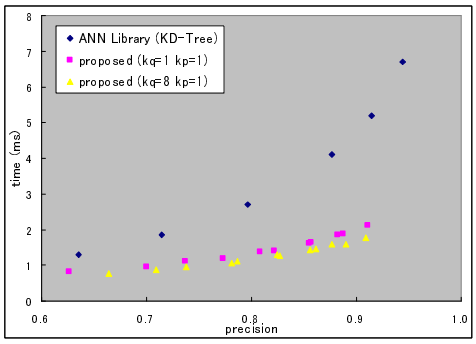
(B2)はANNと比べて著しい性能向上効果が確認できた．しかし非常に大きいメモリサイズを必要とする欠点がある．比較的小さい次元数($d \leqq 15$)に適用できると思われる．

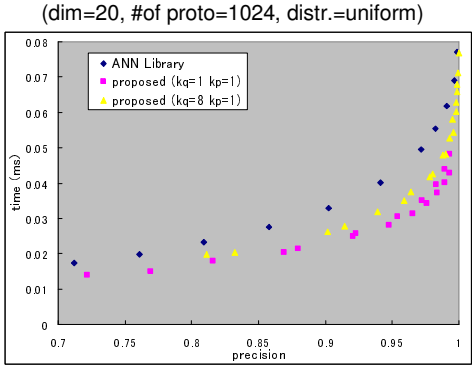
Algorithm-A1,A2
(dim=20, #of proto=1048576, distr.=skewed)

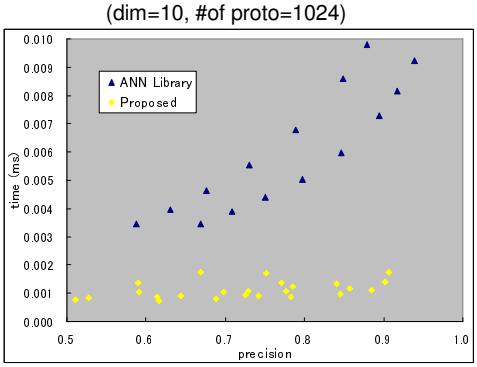
Algorithm-A1,A2
(dim=20, #of proto=1024, distr.=uniform)

Algorithm-B2
(dim=10, #of proto=1024)

探索精度と探索時間の関係







探索精度とメモリサイズの関係