ハッシュを利用した近似最近傍探索における隣接バケット参照の精度とメモリ使用量の理論式の導出

武藤　大志†　　多田　匡志†　　岩村　雅一†　　黄瀬　浩一†

† 大阪府立大学大学院工学研究科　〒 599-8531 堺市中区学園町 1-1
E-mail: †{mutoh,tada}@m.cs.osakafu-u.ac.jp, †{masa,kise}@cs.osakafu-u.ac.jp

あらまし　近似最近傍探索は，クエリと最も距離が近い点を探索する最近傍探索の計算時間，メモリ使用量を大幅に削減する手法である．一般に精度，計算時間，メモリ使用量はトレードオフの関係にあり，その関係を解析することは，様々な場面に近似最近傍探索を適用する上で，重要な課題である．本稿では，ハッシュを利用した近似最近傍探索において，文献 [1]〜[4] で行われている“ 隣接バケットを参照する ”方策のモデル化を行い，精度とメモリ使用量に関して理論式を求める．そして，実験とシミュレーションにより理論式の妥当性を検証する．

キーワード　近似最近傍探索, Locality Sensitive Hashing，隣接バケット，理論式導出

Derivation of Theoretical Formulae of Accuracy and Memory Amount on Accessing Neighboring Buckets in Hash-Based Approximate Nearest Neighbor Search

Tomoyuki MUTO†, Masashi TADA†, Masakazu IWAMURA†, and Koichi KISE†

† Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuencho, Naka, Sakai, 599-8531 Japan
E-mail: †{mutoh,tada}@m.cs.osakafu-u.ac.jp, †{masa,kise}@cs.osakafu-u.ac.jp

Abstract　Approximate nearest neighbor search is a technique which greatly reduces processing time and required amount of memory for nearest neighbor search. Generally, there are the relationships of trede-off among accuracy, processing time and memory amount. Thus, analysis on the relationships is an important task for actual use of approximate nearest neighbor search method. In this paper, we construct a model of approximate nearest neighbor search methods with accessing neighboring buckets [1]〜[4], and derive theoretical formulae in accuracy and memory amount. We compare simulated values with experimented values.

Key words　Approximate Nearest Neighbor Search, Locality Sensitive Hashing, Neighboring Buckets Accessing Hashing, Derive Theoretical Formulae

1. まえがき

近年，大規模なデータベースを用いた様々なアプリケーションが開発されている．そのようなアプリケーションの中には物体認識の様に蓄えられた大量のデータの中から類似のデータを探すことにより情報を処理するものがある．物体認識では，クエリから得られた特徴と最も類似した特徴をデータベース中から求める事が多いため最近傍探索が用いられる．最近傍探索は，データがベクトルで表現されていれば，クエリとの距離が最小となるデータ (最近傍点) を探索する問題である．探索には膨大なデータの中から計算時間を抑えて高速に類似例を検索することが求められる．また，メモリ使用量の削減は実用化に際して必須である．高速な最近傍探索を実現するために，これまでに様々な改良手法が提案されているが，どの手法もデータ数や次元数に対して指数オーダの計算時間あるいはメモリ使用量を必要とする．

その問題を回避するために，最近傍探索と比べて計算時間とメモリ使用量を近似を用いて削減することを目的とした近似最近傍探索という枠組みが近年注目されている．近似最近傍探索は，探索結果の誤りを許容することで，最近傍探索と比べて計算時間とメモリ使用量を大幅に削減することができる．しかし一般に，検索の精度と，計算時間，メモリ使用量の 3 者はト

レードオフの関係にあり，計算時間やメモリ使用量を削減すればするほど精度は低下してしまう．故に，この 3 者間の関係を解析することは重要な研究課題である．本研究では，近似最近傍探索の中でも，特にハッシュを利用した近似最近傍探索に着目する．

ハッシュを利用した近似最近傍探索として Locality Sensitive Hashing(LSH) [5]〜[7] が知られている．LSH は Indyk らによって，探索に要するメモリ使用量と計算時間について解析されており注目を集めている．LSH を用いれば最近傍探索手法よりメモリ使用量や計算時間を削減する事ができるが，それらをより向上させるために隣接バケットを参照し，LSH と精度を同等に保ちつつ計算時間とメモリ使用量を抑える方策が [1]〜[4] で提案されている．しかし，これらの隣接バケットを参照する方策に関する解析はまだ十分にされていない．そこで本研究では解析の前段階として，その隣接バケットを参照する方策をモデル化し，精度とメモリ使用量の理論式を導出する．これ以降，近似最近傍点と最近傍点が一致する確率を (探索) 精度として議論する．理論式の導出に当たり，LSH の解析では近傍点が探索される確率について求められており，近似最近傍点と真の最近傍点が一致する確率を求める事が出来ないので，全く別の解析モデルを構築し導出する．

2. 関 連 手 法

2.1 Locality Sensitive Hashing

Locality Sensitive Hashing(LSH) [5]〜[7] はハッシュを利用した近似最近傍探索手法の 1 つである．ここでは LSH の中でも本研究に関連する，ベクトル空間で用いることが出来る文献 [6] の LSH の概要について述べる．

LSH が近似最近傍点を求めることができるのは局所性に鋭敏な (Locality Sensitive) ハッシュ関数を用いるためである．局所性に鋭敏なハッシュ関数とは，距離が近い点同士は同じハッシュ値を取る確率が高く，距離が遠い点同士は同じハッシュ値を取る確率が小さいハッシュ関数である．

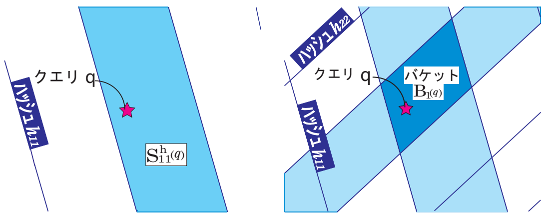
文献 [6] の LSH では次式のハッシュ関数を用いる．

$$h_{ji}(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{a}_{ji} \cdot \boldsymbol{p} + b_{ji}}{w} \right\rfloor \tag{1}$$

ただし，$\boldsymbol{a}_{ji}$ は各次元の要素の値がガウス分布から独立に選ばれた d 次元ベクトル，w はハッシュ幅であり，b_{ji} は区間 $[0, w]$ から一様に選ばれた実数である．

$h_{ji}(q)$ についてクエリ q と同じハッシュ値をとる点 $p*$（すなわち $h_{ji}(q) = h_{ji}(p*)$ を満たす $p*$）が存在し得る空間を $\mathrm{S}^h_{ji}(q)$ とする．図 1(a) の着色部分は $\mathrm{S}^h_{11}(q)$ を図示したものである．LSH は，このように局所性に鋭敏なハッシュ関数を用いることで，高確率でクエリに近い点が存在する空間内の点にのみ距離計算を適用し，計算時間を削減する．

ところが特徴空間が高次元の場合，$\mathrm{S}^h_{ji}(q)$ が大きくなってしまい，明らかに最近傍点になり得ない点も距離計算の対象とする効率の悪い探索になることがある．そこで，LSH はハッシュ関数を複数用いてハッシュ関数群を構成し，この問題を緩和する．



(a) $\mathrm{S}^h_{11}(q)$　(b) $\mathrm{S}^h_{11}(q) \cap \mathrm{S}^h_{12}(q)$

図 1　$g_j(q)$ で距離計算の対象となる点が存在し得る空間

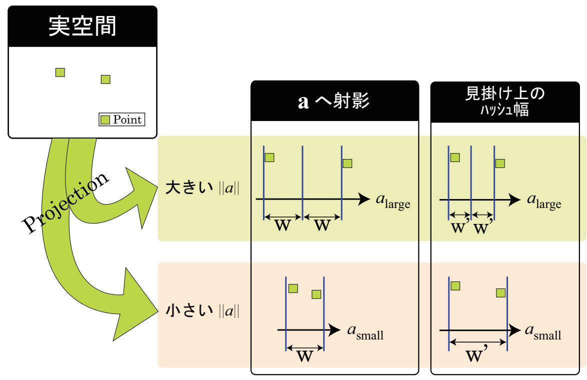


図 2　$\boldsymbol{a}$ の大きさの変化による見掛け上のハッシュ幅 (w') の変化

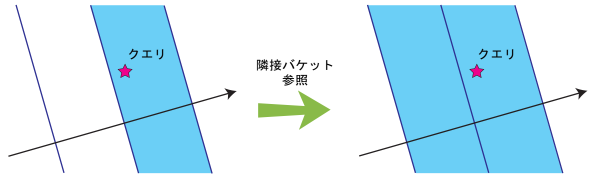
る．図 1(b) はハッシュ関数群 g_1 を $g_1 = \{h_{11}, h_{12}\}$ とした時の例で，g_1 を構成する h_{11}，h_{12} において $\mathrm{S}^h_{11}(q)$ と $\mathrm{S}^h_{12}(q)$ の両方に入った点のみを距離計算の対象にする．これを一般化する．k 個のハッシュ関数 $h_{j1}, h_{j2}, \ldots, h_{jk}$ を組み合わせて，関数群 $g_j = \{h_{j1}, \ldots, h_{jk}\}$ を作る．このとき $g_j(q) = g_j(p*)$，すなわち $^\forall i, h_{ji}(q) = h_{ji}(p*)$ を満たす点 $p*$ のみを距離計算の対象とする．g_j において同じ値を取る空間，つまり $\mathrm{B}^g_j(q) = \bigcap_{j=1}^{k} \mathrm{S}^h_{ji}(q)$ をバケットと呼ぶ．

さらに LSH は，複数のハッシュ関数群を用いて距離計算の対象を増やし，精度向上を図っている．ハッシュ関数群 g_1 と g_2 があったときバケット $\mathrm{B}^g_1(q)$，またはバケット $\mathrm{B}^g_2(q)$ のいずれかに入った点を距離計算の対象とすると，対象となる点数を増やすことができる．これを一般化すると，LSH は関数群 g_j を L 個用いて，$g_j (1 \leqq j \leqq L)$ において一度でもクエリと同じバケットに入った点に距離計算を適用して，精度を向上させている．

ここでハッシュ幅 w が一定でも，見掛け上のハッシュ幅は変動することに注意する．図 2 が示すように，$\boldsymbol{a}$ の大きさが変動すると，内積計算後の値も変動する為，実空間上に現れるハッシュ幅も変動してしまう．これを見掛け上のハッシュ幅 w' とし，$w' = w / \parallel \boldsymbol{a} \parallel$ で定義する．

2.2 隣接バケットを参照する手法

前述の通り，LSH を近似最近傍探索に適用する際の問題点は，g_j の数が少ない (L の値が小さい) と最近傍点が距離計算の対象となる確率が低く，g_j の数が多い (L の値が大きい) と計算時間やメモリ使用量が大きくなってしまうことである．これは，LSH はクエリ周辺の点を効率的に距離計算の対象にできない為である．この問題を解決するために，高速に特定物体認識をする野口らの手法 [1] と，近似最近傍探索手法である Prin-



(a) クエリがハッシュ値境界付近にある　(b) クエリが中心付近にある

図 3　ビンとクエリの位置

cipal Component Hashing(PCH) [2]，Multi-Probe LSH(MP LSH) [3]，Multi-Valued Hashing(MVH) [4] の 4 つの手法では，“隣接バケットを参照する”方策を用いている．

まず，4 つの手法の共通点について述べる．図 3(a) のように，クエリがハッシュ値境界の付近にある場合，最近傍点がほぼ 1/2 の確率で隣のビンに入ってしまう為，最近傍点が距離計算の対象から漏れてしまう事がある．LSH ではその確率を下げるために複数のハッシュ関数群を用いる．しかし，常に図 3(b) のようなクエリが中心付近に入るようなバケットを構築出来れば，少ないハッシュ関数群で最近傍点を効率良く距離計算の対象にする事が出来るはずである．これが“隣接バケットを参照する”方策に共通する考えである．しかし，どの様な場合に隣接バケットを参照するかがそれぞれの手法で異なるため，その違いについて以下で説明する．

物体認識手法である野口らの手法 [1] は，画像から抽出される特徴ベクトルの近似最近傍探索によって特定物体認識をする．ハッシュを利用した近似最近傍探索を用い，クエリとハッシュ値境界間の距離が設定した閾値よりも小さくなる場合に隣接するバケットを参照する．

PCH [2] は LSH と同様，ハッシュを利用した近似最近傍探索手法である．データの分布が正規分布であると仮定して主成分を求め，その軸上でハッシュ値を区切るため，LSH のようにハッシュ関数群を複数構築することができない．そこで，クエリが入ったビンの δ 個隣のビンまで参照して精度を向上させる．δ は事前に設定した値である．

MP LSH [3] は，LSH の改良を試みた近似最近傍探索手法である．LSH よりもメモリ使用量を削減するために，最近傍点がクエリと同じビンに入る確率が低いハッシュ関数から順に隣接ビンも参照する．

MVH [4] は，ハッシュで距離の概算を算出して距離計算の候補を絞り込む近似最近傍探索手法である．クエリに近いほど得票数を多くするため，クエリが入ったビンの t 個隣りのビンまで参照し，クエリが入ったビン内の点に $t+1$ 票，両隣のビン内の点に t 票，以下 $t-1,\ldots,1$ 票と投票する．この処理を複数のハッシュ関数に対して施すと得票数が多い点程，クエリの近傍点である確率が高くなる．MVH は距離計算無しでも近似最近傍点を探索することができる．

3. 隣接バケット参照モデル

本節では，前節で紹介した隣接バケットを参照する方策をモデル化する．モデル化したものを本研究では Neighboring Buckets Accessing Hashing(NBAH) と呼ぶ．本研究でモデル化する NBAH の概要を挙げておく．ハッシュ関数は式 (1) を用いる．隣接バケットを参照する基準は野口らの手法と同様とする．すなわち，クエリとハッシュ値境界間の距離が閾値よりも小さいか否かで決める．

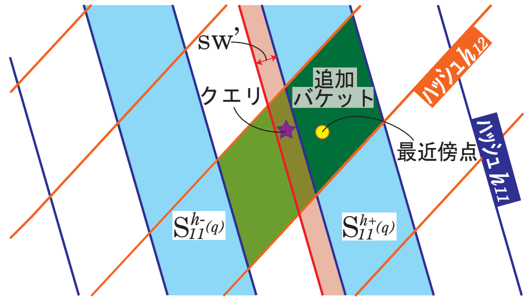


図 4　NBAH の概要

NBAH を定式化する．まず，隣接するビンを参照するか否かを決定する閾値 s を図 4 のように設定し，クエリとハッシュ値境界間の距離が sw' よりも小さい時には，隣接ビンもクエリが入ったビンと同様に扱うとする．図 4 のように，h_{ji} に対して $h_{ji}(q)\pm 1$ となる隣接ビンの空間をそれぞれ $S^{h^+}_{ji}(q)$，$S^{h^-}_{ji}(q)$ と置く．図 4 の場合を例にとると，h_{11} に対して sw' よりも近い所にクエリがあるので，$S^{h^-}_{11}(q)$ 内の点も $S^{h}_{11}(q)$ 内の点と同様に扱うことになる．それによってバケット $S^{h}_{11}(q)\cap S^{h}_{12}(q)$ 内の点だけではなくバケット $S^{h^-}_{11}(q)\cap S^{h}_{12}(q)$ 内の点も距離計算の対象にする．このような処理を h_{ji} 毎に施せば，複数のハッシュ関数においてクエリが境界値に近い値を取っても対応することができる．

ハッシュ関数に式 (1) を用いるため，$\boldsymbol{a}$ はハッシュ幅 w で区切られる．クエリとハッシュ値境界間の距離が sw' より小さいときに隣接バケットも探索するために

$$h^{+}(\boldsymbol{p}^{*}) = \left\lfloor \frac{\boldsymbol{a}\cdot\boldsymbol{p}^{*}+b}{w} + sw' \right\rfloor$$
$$h^{-}(\boldsymbol{p}^{*}) = \left\lfloor \frac{\boldsymbol{a}\cdot\boldsymbol{p}^{*}+b}{w} - sw' \right\rfloor$$

となるような $h^{+}(\cdot)$ と $h^{-}(\cdot)$ を設定して下記の処理を LSH に追加し，NBAH を実現する．

- もし $h(q)-h^{-}(q)\neq 0$ なら $h_{ji}(q)-1$ 内の点も $h(q)$ 内の点と同様に扱う．
- もし $h(q)-h^{+}(q)\neq 0$ なら $h_{ji}(q)+1$ 内の点も $h(q)$ 内の点と同様に扱う．

4. 理論値導出

本節では，前節でモデル化したモデルより理論値を導出する．本稿では精度とメモリ使用量に関してそれぞれ理論値を導出する．データの次元数を d とする．

4.1 精　　度

1. で述べた通り，本研究では最近傍点と近似最近傍点が一致する確率を精度として導出する．一般にデータの分布は未知であるが，一様分布に対する解析は比較的容易で目安として用いる事が出来る為，本稿では一様分布を仮定する．

初めに理論値の導出手順を以下に示す．

(i)．クエリから距離 R の位置に最近傍点があると仮定する．

(ii)．(i) の時にハッシュ関数 h_{ji} において最近傍点がクエリと同じハッシュ値を取る確率 $\mathrm{P}_h(u,s,w',R)$ を求める．

(iii)．クエリとハッシュのビンの位置関係 u を考慮して期待値 $\mathrm{P}_h(s,w',R)$ を求める．

(iv)．ハッシュ幅の変動 w' を考慮して期待値 $\mathrm{P}_h(s,w,R)$ を求める．

(v)．最近傍点までの距離 (R) の確率密度関数を求め，R に対する期待値 $\mathrm{P}_h(s,w)$ を求める．

(vi)．最近傍点がクエリと同じバケットに入る確率 $\mathrm{P}(s,w)$ を求める．

手順 (i)〜(v) までで，ハッシュ関数 h_{ji} において最近傍点がクエリと同じビンに入る確率を求め，手順 (vi) で，ハッシュ関数群 $_gj$ において最近傍点がクエリと同じバケットに入る確率を求める．

上記の手順で述べた通り，クエリから距離 R の位置に最近傍点があるとする．このとき，半径 R の d 次元超球を考えると，最近傍点は必ず超球の表面上に存在する．つまり，超球がクエリの入ったビンの中に全て含まれていれば，クエリと最近傍点は必ず同じビンに入る．超球の表面のうち，クエリが入ったビン空間 $\mathrm{S}^h(q)$ 内にある部分が小さくなればなるほどクエリと最近傍点が同じビンに入る確率は下がる．これをモデル化したものが図 5 である．このモデルにおいて，図中の着色部分の表面積が超球全体の表面積に占める割合を求めることにより，クエリと最近傍点が同じビンに入る確率 P_h を求める．但し，超球は軸に対して垂直な直径に対して対称なので，以下では半超球についてのみ考える．

ここから手順 (ii) について述べる．図 5 のように u をクエリからベクトル $\boldsymbol{a}$ 方向のハッシュ値境界までの距離とする．この時，u はハッシュのビンの中でのクエリの位置を表す．図 5 のモデルは R と w' と s と u の関係で場合分けした結果考えられる全ての状態を表す．まず u と sw' の関係に着目すると，

(I)．隣接ビンを参照しない $(u > sw')$

(II)．隣接ビンを参照する $(u \leqq sw')$

の 2 つに分けることができる．さらに，(I)，(II) それぞれに対して

(A)．半超球全体がクエリと同じビンに入る

(B)．半超球全体がクエリと同じビンに入りきらない

の 2 つに分けることができる．ここで，(A) と (B) のどちらに分けられるかは，u と R と w' の関係によって決まることを述べておく．この結果，図 5 に示すように (IA)(IIA)(IB)(IIB) の 4 つの状態に場合分けされる．

図 5 の着色部分の面積を求めるために，図 5 の様に

$$\theta(u) = \begin{cases} 0, & \text{for (IA) and (IIA) (図. 5(a))} \\ \cos^{-1}\frac{u}{R}, & \text{for (IB) (図. 5(c))} \\ \cos^{-1}\frac{u+w'}{R}, & \text{for (IIB) (図. 5(d)).} \end{cases} \tag{2}$$

表 1　積分範囲と $\theta(u)$.

		α	β	$\theta(u)$
(i)	(IIA)	0	sw'	0
	(IB)	sw'	R	$\cos^{-1}\frac{u}{R}$
	(IA)	R	w'	0
(ii)	(IIB)	0	$R-w'$	$\cos^{-1}\frac{u+w'}{R}$
	(IIA)	$R-w'$	sw'	0
	(IB)	sw'	w'	$\cos^{-1}\frac{u}{R}$
(iii)	(IIB)	0	sw'	$\cos^{-1}\frac{u+w'}{R}$
	(IB)	sw'	w'	$\cos^{-1}\frac{u}{R}$

を満たす $\theta(u)$ をおく．また，半径 R の d 次元超球の表面積は $\mathrm{Sa}^d(R) = \frac{2\pi^{d/2}}{\Gamma(d/2)}R^{d-1}$ とおけるので，図 5 のそれぞれの着色部分の表面積 $\mathrm{Sa}_0^d(R)$ は

$$\mathrm{Sa}_0^d(R) = \int_0^{R\cos\theta(u)} \mathrm{Sa}^{d-1}(x)\ dx \tag{3}$$

$$= \int_{\theta(u)}^{\frac{\pi}{2}} \mathrm{Sa}^{d-1}(R\sin\phi)R\ d\phi \tag{4}$$

で求められる．但し，式 (4) は，式 (3) の積分変数 x を $x = R\sin(\phi)$ を用いて変換して導いた．故に，図 5 の着色部分の表面積が半超球全体の表面積に占める割合 $\mathrm{P}_h(u,s,w',R)$ は

$$\mathrm{P}_h(u,s,w',R) = \frac{\mathrm{Sa}_0^d(R)}{\mathrm{Sa}^d(R)/2} \tag{5}$$

である．

次に手順 (iii) について述べる．$\mathrm{P}_h(u,s,w',R)$ は u の関数である．u はハッシュのビンに対するクエリの位置を表しており，u の値は一定ではないので u に対する期待値を求める．式 (1) の b_{ji} は一様分布 $\mathrm{U}(0,w')$ に従う乱数であり，u も同じ分布に従うため期待値 $\mathrm{P}_h(s,w',R)$ は

$$\mathrm{P}_h(s,w',R) = \frac{1}{w'}\int_0^{w'} \mathrm{P}_h(u)\ du \tag{6}$$

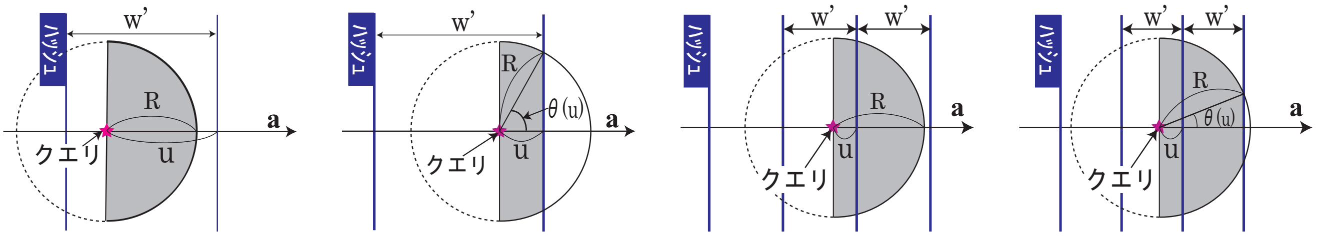
となる．但し，被積分関数内に u により場合分けされる部分が存在する為，$\mathrm{P}_h^{\alpha,\beta}(s,w',R) = \frac{1}{w'}\int_\alpha^\beta \mathrm{P}_h(u)\ du$ となるような $\mathrm{P}_h^{\alpha,\beta}(s,w',R)$ を設定して，積分区間を分けながら計算する必要がある．図 5 に示す状態と α，β と $\theta(u)$ の関係に関しては表 1 を参照されたい．

次に手順 (iv) について述べる．ここでは見かけ上のハッシュ幅が変化することを考慮する．前述の通り，$\|\boldsymbol{a}\|$ の値が変化する為，見掛け上のハッシュ幅 w' は w を一定にしていても変動する．それ故，これまでに導出したすべての式の w' を w/A で置き換えて A に関する期待値を求める．但し，$\|\boldsymbol{a}\| = A$ とする．$\boldsymbol{a}$ の各次元の要素はガウス分布に従う為，A の値は自由度 $d^{1/2}$ の χ^2 分布に従う．従って，$\mathrm{P_h}(s,w,R)$ は

$$\mathrm{P}_h(s,w,R) = \int_0^\infty \frac{A^{(d/4)-1/2}e^{-A/2}}{2^{d/4}\Gamma(d^{1/2}/2)}\mathrm{P}_h(s,\frac{w}{A},R)\ dA \tag{7}$$

となる．

次に手順 (v) について述べる．ここまで，クエリと最近傍点間の距離 R が既知という仮定で議論してきたが，実際は既知で



(a) 半超球が $\mathrm{S}^h(q)$ に全て含まれる (IA)

(b) 半超球が $\mathrm{S}^h(q)$ に含まれない部分がある (IB)

(c) 半超球が $\mathrm{S}^h(q) \cup \mathrm{S}^{h^\pm}(q)$ に全て含まれる (IIA)

(d) 半超球が $\mathrm{S}^h(q) \cup \mathrm{S}^{h^\pm}(q)$ に含まれない部分がある (IIB)

図 5　解析モデル

はない．そこで，R の確率分布を求める必要がある．R の確率分布は順序統計量 [8] を用いて求めることができる．クエリと最近傍点間の距離 R が従う確率密度関数を $p(R)$ とし，クエリから各データ点までの距離の確率密度関数を $f(R)$，$f(R)$ の累積分布関数を $F(R)$ とする．また，データ数を n とすると，

$$p(R) = n[1 - F(R)]^{n-1} f(R) \tag{8}$$

が成り立つ．式 (8) は，データの分布が一様でありデータがクエリから半径 $R_{\max}$ の d 次元超球空間に分布しているとすると，

$$p(R) = n\left[1 - \frac{V^d(R)}{V^d(R_{\max})}\right]^{n-1} \frac{\mathrm{Sa}^d(R)}{V^d(R_{\max})} \tag{9}$$

となる．但し，$V^d(r)$ は半径 r の d 次元超球の体積で $V^d(r) = \frac{\pi^{d/2} r^d}{\Gamma((d/2)+1)}$ を表す．以上より，$\mathrm{P}_h(s,w,R)$ の R に対する期待値 $\mathrm{P}_h(s,w)$ は式 (8) と式 (9) より

$$\mathrm{P}_h(s,w) = \int_0^{R_{\max}} p(R)\mathrm{P}_h(s,w,R)\, dR \tag{10}$$

となる．以上で，ハッシュ関数 h_{ji} において最近傍点がクエリと同じハッシュ値を取る確率が求められた．

最後に手順 (vi) について述べる．ここでは，最近傍点が少なくとも 1 回クエリと同じバケットに入る確率を導出する．その確率を $\mathrm{P}(s,w)$ とおくと，

$$\mathrm{P}(s,w) = 1 - [1 - \{\mathrm{P}_h(s,w)\}^k]^L \tag{11}$$

と求めることができる [5]．これを以下で簡単に説明する．ハッシュ関数群 g_j においてある点 p がクエリと同じバケットに入る（つまり $g_j(q) = g_j(p)$ を満たす）確率は，各 h_{ji} で $h_{ji}(q) = h_{ji}(p)$ となる事象が独立で起こるため，$\{\mathrm{P}_h(s,w)\}^k$ である．g_j は互いに独立なので $^\forall j,\ g_j(q) \neq g_j(p)$ となる事象は確率 $[1 - \{\mathrm{P}_h(s,w)\}^k]^L$ で起こる．$^\exists j, g_j(q) = g_j(p)$ を満たす点 p は距離を計算する対象になるので，p が距離計算の対象になる確率は $^\forall j, g_j(q) \neq g_j(p)$ の余事象の確率と等しい．

4.2　メモリ使用量

メモリ使用量として考えられるのは，ハッシュテーブルの容量とデータの容量である．まずハッシュテーブルの容量について述べる．LSH と NBAH では，h_{ji}1 つ当たり 1 つのハッシュテーブルを必要とする．配列の型により決まる定数を c_1，ポインタの容量を c_2 とする．ハッシュテーブルを配列で構築すれば，$c_1 n$ バイト必要である．さらに，データ数分だけポインタを必要とすると，$c_2 n$ バイト加えなければならない．以上より，ハッシュテーブルの数が kL の時，ハッシュテーブルのメモリ使用量は $nKL(c_1 + c_2)$ バイトである．次にデータの容量について述べる．データ要素の型により決まる定数を c_3，データの ID の付加に必要な容量を c_4 とする．d 次元ベクトルの場合を考えると，1 つのデータの保存に必要な容量は $c_3 d + c_4$ バイトである．つまりデータ数が n の時，$(c_3 d + c_4)n$ バイトの容量が必要である．従って，メモリ使用量は $n\{(c_1 + c_2)kL + (c_3 d + c_4)\}$ と考えられる．

5.　実験とシュミレーション

本節では，導出した理論式の妥当性を調べるために実験とシミュレーションの結果を比較する．実験結果は，実際に NBAH を実装して人工データに対して近似最近傍探索をした結果である．シミュレーション結果は，前節で導出した数式から数値計算により導いた値である．

5.1　実　　験

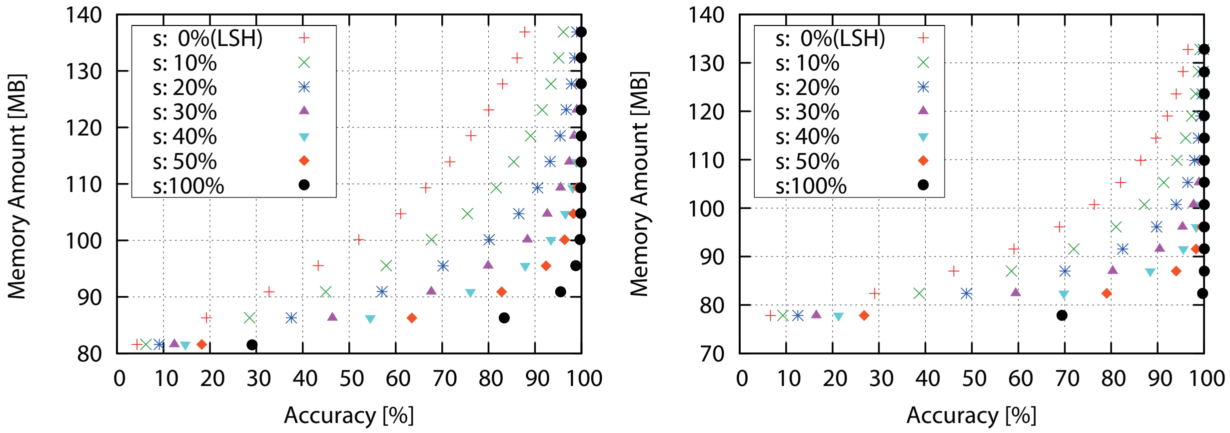
100 次元空間に一様に分布する人工データを作成して実験を行う．各次元の要素の値が $[0, 1300]$ であるような 100 次元のデータを 100000 点生成し，クエリを 1000 点生成した．精度は，近似最近傍点と最近傍点が一致した割合で算出した．またメモリ使用量は，各クエリにおける実測値をパラメータ毎に平均を取った．パラメータは $w = 5000$，$k = 3$ とし，L の値を変動させた．$s = 0$ の時は，隣接するビンを参照する事がないので LSH を表している．$s = 0.5$ の時は，クエリからの距離が近い方の隣接ビンを必ず参照し，$s = 1.0$ の時は，クエリが入った両隣のビンを必ず参照する．

5.2　シミュレーション

4 節で導出した式に基づき計算し，妥当性を検証する．精度は式 (11)，メモリ使用量は $n\{(c_1 + c_2)kL + (c_3 d + c_4)\}$ の値とする．式 (11) には，解析的に解くことが出来ない積分が含まれている為，Monte Carlo 法を用いた数値積分により値を求めた．メモリ使用量は，$c_1 = 4, c_2 = 0, c_3 = 8, c_4 = 4$ として求めた．これは，実装に合わせた値である．c_1, c_4 は int 型なので 4byte，c_3 は double 型なので 8byte，c_2 は，ポインタを用いていないので 0 である．尚，パラメータの値は実験と同じものを用いた．

5.3　結　　果

妥当性を検証する為に実験値と理論値の相異度を確かめる



(a) 実験結果　　(b) シミュレーション結果

図 6　精度とメモリ使用量の関係

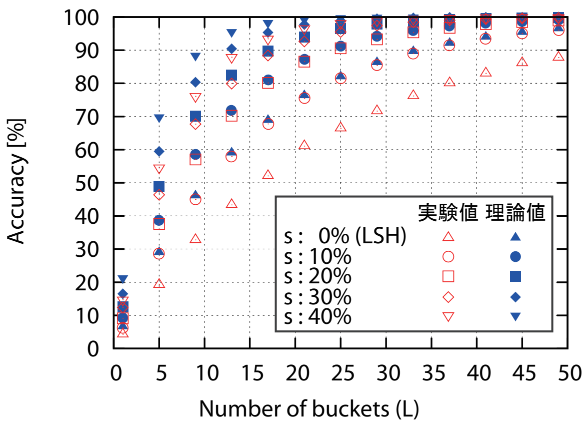


図 7　精度の実験値と理論値の比較

必要がある．図 6(a) と図 6(b) はそれぞれ実験とシミュレーションにより得られた精度とメモリ使用量の関係を表している．図 6(a) と図 6(b) は，パラメータを揃えた時の実験値と理論値なので，この二つの図が完全に一致すれば，そのシミュレーションは妥当であると言える．

メモリ使用量に関しては実験値と理論値がほぼ一致している事が分かる(注1)．精度に関しては，図 6 から分かるように，精度に関しては理論値の方が実験値よりも全体的に上回っている．図 7 は，精度に関して実験値と理論値を比較した図である．図中の塗りつぶされている点が理論値である．また形が同じ点は，s の値が等しい事を示す．図 7 には，s が 0%〜40% までの点をプロットした．図 7 を見ても，理論値の方が全体的に実験値を上回っている．これは，式 (11) には k 乗と L 乗の計算がある為であると考えられる．$0.6^2 > 0.5*0.7 > 0.4*0.8$ の様にある数の累乗の方がその数よりも大きい値と小さい値を掛けたものより大きくなる様に，実験の精度は変動が大きいのに対し理論値は常に期待値を用いて計算している為理論値が実験値を上回ったと考えられる．これを解決する為に，理論値も結果の変動を考慮した計算をし直す必要がある．

6. む す び

本稿では，ハッシュを利用した近似最近傍探索において，隣接バケットを参照する方策のモデル化し，そのモデルから理論値を導出した．実装して実験した結果とシミュレーションの結果を比較した．メモリ使用量に関しては妥当な値を導くことができた．精度に関しては，変動を考慮して実験値により近づけてくことが今後の課題である．他の課題として，計算時間に関する理論値を導出してトレードオフの関係を解析することが挙げられる．

謝辞　本研究の一部は，平成 20 年度 SCAT 研究費助成ならびに科研費補助金基盤研究 (B)(19300062) の補助による．

文　　献

[1] 野口和人，黄瀬浩一，岩村雅一，“近似最近傍探索の多段階化による物体の高速認識，” 画像の認識・理解シンポジウム (MIRU2007) 論文集，pp.111–118，July 2007.

[2] 松下裕輔，和田俊和，“Principal Component Hashing -等確率バケット分割による近似最近傍探索法-，” 画像の認識・理解シンポジウム (MIRU2007) 論文集，pp.127–134，July 2007.

[3] Q. Lv, W. Josephson, Z. Wang, and M. Charikar, “Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search,” Proc. 33th Intl. Conf. on Vaey Large Data Bases (VLDB), pp.950–961, Sept. 2007.

[4] 多田匡志，武藤大志，岩村雅一，黄瀬浩一，“近さの多段階表現に基づく近似最近傍探索，” 信学技報, PRMU2009-110，pp.121–126，Nov. 2009.

[5] P. Indyk and R. Motowani, “Approximate nearest neighbors : Towards removing the curse of dimensionality,” Proc. 30th ACM Symposium on Theory of Computing(STOC’98), pp.604–613, May 1999.

[6] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, “Locality-Sensitive Hashing Scheme Based on p-Stable,” Proc. 20th Annual Symposium on Computational Geometry(SCG2004), pp.253–262, June 2004.

[7] A. Andoni and P. Indyk, “Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions,” Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06), pp.459–468, Oct. 2006.

[8] H.A. David and H.N. Nagaraja, Order Statistics, WILEY-IMTERSCIENCE, Aug. 2003.

(注1)：メモリ使用量の実験値は実装により左右される為注意が必要である．例えば c++で実装する際，malloc や realloc を用いれば問題はないが，vector 等を用いた場合，実際に使用するメモリよりも多くのメモリを確保する事がある為，実験値と理論値が一致しない事がある．