未知スコア関数で評価される多次元データを対象とするLSHを用いたTop-k検索アルゴリズム

池田　達樹† 　常　　穹† 　宮崎　　純†

† 東京工業大学情報理工学院　〒 152-8550 東京都目黒区大岡山
E-mail: †ikeda@lsc.c.titech.ac.jp, ††{q.chang,miyazaki}@c.titech.ac.jp

あらまし　本研究では，スコア関数が未知である場合に高次元のデータに対して適用可能な Top-k 検索アルゴリズムを提案する．Top-k 検索は，検索システムによって与えられるスコアが上位となる k 件のデータを，すべてのデータを見ることなく効率良く取得する検索である．様々な状況における Top-k 検索アルゴリズムが考案されており，そのほとんどが検索対象のデータをスコアリングするスコア関数の性質が明らかであることが条件である．提案手法はスコア関数が未知である場合にも有効である．評価実験では提案手法の実行時間と精度について評価を行う．

キーワード　情報検索, Top-k

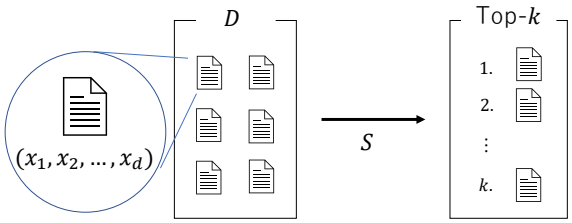


図 1　Top-k 検索の流れ

1　はじめに

インターネットや Web の普及によって，我々の身の回りでは大量の電子化された情報で溢れている．ユーザーはその大量の情報の中から自分に必要となる情報を選ばなければならない．その手段として情報検索と呼ばれる技術が存在する．情報検索はユーザーの要求に適合する情報を情報集合の中から見つけ出すための技術である．

検索速度を向上させる研究の一つに Top-k 検索がある．Top-k 検索とは，検索システムやデータベースシステムにおいて，与えられる評価スコアが上位となる k 件のデータを効率よく取得する検索である．一般に，ユーザーは情報の中から，より自分のニーズに適している上位の結果に関心を持つため，Top-k 検索を高速に処理できるシステムは有用である．

図 1 に Top-k 検索の流れを示す．検索対象のデータ集合を D とし，それぞれのデータは d 個の属性を持つものとする．図 1 の $x_1, x_2, ..., x_d$ はデータの各属性値を表している．データは d 個の属性値をスコア関数 S に適用することでスコアがつけられ，スコアが高いものから順にランキングされるものとする．Top-k 検索はデータアクセスの制約やスコア関数の性質によって様々な研究が進められている [2], [7], [8], [9], [10], [11].

このように様々な状況における Top-k 検索アルゴリズムが考案されているが，検索対象のデータのインデックスを事前に作成できない状況，且つスコア関数の性質や特徴が未知な状況において適用可能な Top-k 検索アルゴリズムは知られていない．既存の多くの Top-k 検索アルゴリズムは事前に検索対象のデータがインデックス付けされていることを仮定している．しかし例えば検索エンジンでは，入力キーワードの組み合わせによって Web ページのスコアが変わり，そのすべての組み合わせに対してインデックスを構築することは不可能である．

佐々木ら [3] は，この問題に対し，スコア関数が値を取りうる空間をメッシュに分割する，メッシュ分割法により対応した．しかし，この手法はスコア関数の性質が既知である必要があった．また，メッシュ分割法はその性質上，メッシュの形成やスコア関数のピークを探索する計算量は高次元になるにつれて指数関数的に増加してしまう．

スコア関数のピークに近い点は Top-k の候補となりうる．この事実に注目し，Top-k の候補となりうるデータを抽出する問題をスコア関数のピークの近傍点探索問題 [12] と捉えた．次元の呪いの影響により高次元における効率的な最近傍点探索アルゴリズムは知られていない．そこで確率的に近傍点を探索する手法の研究が進められている．

本研究では，Datar ら [6] が提案した安定分布による Locality-Sensitive Hashing(LSH) を利用し，スコア関数のピークの周辺のデータを確率的に取得することで，高次元で有用な Top-k 検索アルゴリズムを提案する．提案手法について実行速度とその精度について評価を行う．

2　準　　備

Top-k 検索について議論をする前に，本章では 2.1 節でランキングの評価指標である nDCG を，2.2 節で提案手法で参考にした最近傍探索問題について述べる．

2.1　nDCG

nDCG は検索した情報の順序を考慮した評価尺度である．ク

エリに対する適合度が高い情報が上位に位置するほど評価値が高くなる．k 件の情報を検索したときの nDCG@k は式 (1) に示す DCG@k を用いて式 (2) で定義される．

$$DCG@k = score_1 + \sum_{i=2}^{k} \frac{score_i}{\log_2 i} \tag{1}$$

$$nDCG@k = \frac{DCG@k}{iDCG@k} \tag{2}$$

ここで iDCG は真の正しいランキングを用いて得られる DCG を表す．$score$ は情報の適合度を表す．

2.2 最近傍点探索問題

最近傍点探索問題 [12] とは，多数のデータからなる集合の中から，ある一つのクエリに最も近いデータを探索する問題であり，光学文字認識やレコメンデーションシステムなど様々な分野で使われている．この問題は基本的にはデータ集合のすべてのデータとクエリの距離をすべて計算し最小値を求める全探索を用いることで解決する．データ量を n, 次元数を d とすれば，$O(nd)$ の計算量を要する．大量のデータ，高次元のデータを取り扱う際には効率のよい探索アルゴリズムが求められる．

最近傍探索問題は厳密にクエリとの距離が最小値であるデータを取得することを目標とし，様々な手法が考えられてきた．しかし，高次元になるにつれて次元の呪いにより既存手法の検索速度が全探索法とほとんど変わらないことが示されている．そこで近似的に近傍点を探す手法が提案されてきた．

2.2.1 Locality-Sensitive Hashing(LSH)

近似近傍点探索アルゴリズムの一つに Indyk ら [5] によって提案された Locality-Sensitive Hashing(LSH) がある．LSH の基本的な考え方としては，類似しているデータは同じハッシュバケットに属し，類似していないデータは異なるバケットに属する確率が高くなるようなハッシュ関数 h を複数用いてクエリと同じハッシュ値のデータを取り出し近傍点を求める，というものである．クエリを $\boldsymbol{q}$, データ集合の各データを $\boldsymbol{v}$, 2 つのデータ $\boldsymbol{x}, \boldsymbol{y}$ の距離を $D(\boldsymbol{x}, \boldsymbol{y})$ としたとき，ハッシュ関数 h は式 (3) のような条件を満たす場合に Locality Sensitive であるという．

$$\begin{aligned} &D(\boldsymbol{v}, \boldsymbol{q}) \leqq r_1 \Rightarrow Pr[h(\boldsymbol{q} = h(\boldsymbol{v}))] \geqq p_1 \\ &D(\boldsymbol{v}, \boldsymbol{q}) > r_2 \Rightarrow Pr[h(\boldsymbol{q} = h(\boldsymbol{v}))] < p_2 \end{aligned} \tag{3}$$

ただし，$0 \leqq p_2 \leqq p_1 \leqq 1$, $r_2 = cr_1$, $c > 1$ は定数である．

具体的に用いられるハッシュ関数には，Datar らの安定分布を利用したもの [6] がある．このハッシュ関数は，安定分布から生成されたベクトル $\boldsymbol{a}$，閉区間 $[0, W](W > 0)$ から一様ランダムに選ばれた実数 b を用いて，式 (4) で表される．

$$h(\boldsymbol{v}) = \lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{v} + b}{W} \rfloor \tag{4}$$

LSH の前処理では，式 (5) のように，j 個のハッシュ関数 h からなるハッシュテーブル g を L 個用意する．図 2 は LSH のイメージを表したものである．

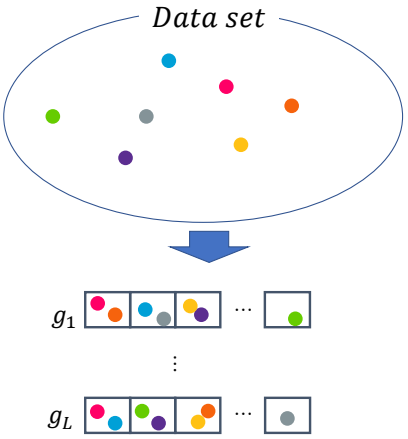


図 2　LSH のイメージ

$$g(\boldsymbol{v}) = \{h_1(\boldsymbol{v}), h_2(\boldsymbol{v}), ..., h_j(\boldsymbol{v})\} \tag{5}$$

データ集合のすべてのデータに対して，ハッシュテーブル g_i ごとにハッシュ値を計算し，バケットに格納していく．その後，クエリ $\boldsymbol{q}$ のハッシュ値を計算し，ハッシュテーブル $g_1, ..., g_L$ を探索して $\boldsymbol{q}$ と同じバケットに格納されたデータを抽出することで一定以上の確率で最近傍点を見つけることができる．

ここで，LSH における j, L, W のパラメータの意味することについて説明する．j は高次元データを低次元のデータに圧縮した先の次元数と捉えることができる．式 (3) より，クエリ $\boldsymbol{q}$ とデータ $\boldsymbol{v}$ が同じバケットに属する確率は，式 (6) で表される．

$$Pr[h(\boldsymbol{q}) = h(\boldsymbol{v})] \geq p_1^j \tag{6}$$

すなわち，j が大きくなるほど，$\boldsymbol{q}$ と $\boldsymbol{v}$ が同じバケットに属する確率は低くなる．そこで，ハッシュテーブル g の個数である L を増やすことによって，同じバケットに属する機会を増やしている．また，図 3 に示すように，W の値は $a \cdot v$ を実数直線上に写像した際に，どれほどの範囲でハッシュ値を同じとするかを定めるものである．

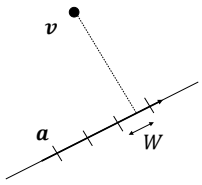


図 3　LSH の W のイメージ

3 関 連 研 究

本章では，既存の Top-k 検索アルゴリズムを紹介する．

3.1 Top-k 検索アルゴリズム

3.1.1 Threshold Algorithm

Threshold Algorithm(TA) は，Fagin らによって提案された [1].TA は，スコア関数が単調であり，それぞれの属性のソー

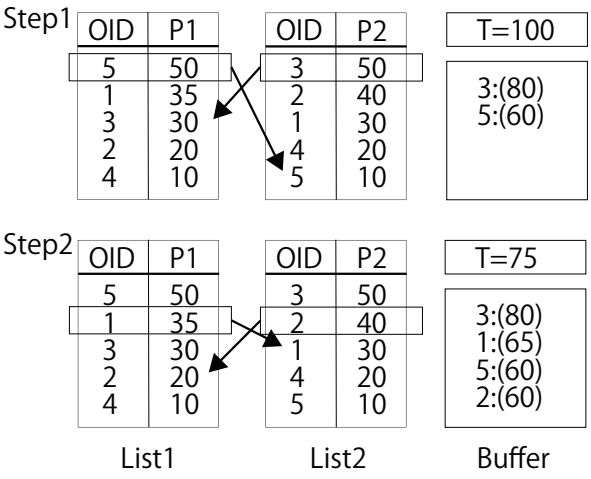


図 4　TA の 例

トされたリストが用意されているときに適用できるアルゴリズムである．TA は，各リストを並行にアクセスし，未アクセスのデータのスコアの上限を推定しながら行う．上限は既にアクセスされたデータのスコアを超えないように推定される．スコアが推定された上限を超えるデータ数が要求されている数以上になったときにアルゴリズムは停止する．

図 4 に TA の例を示す．データ d は (p_1, p_2) で表されるものとし，スコア関数は $S(d) = p_1 + p_2$ を仮定する．データの各属性値が降順にソートされたリストが用意されているものとする．各リストの先頭からデータの属性値を取得し，残りの属性値をもう一つのリストからランダムアクセスによって取得する．得られた 2 つの属性値から先頭のデータのスコアを計算する．計算されたデータはスコアの降順に記録されていく．未取得のデータのスコアのしきい値 T は，両方のリストで最後に取得された属性値を S に適用して計算される．図 4 の step1 ではしきい値は $50 + 50$ を計算して 100 になる．step2 では同様に $T = 75$ となり，75 以上のスコアを持つ ID が 3 であるデータが Top-k の解として保持される．保持される解の数が k 以上となるか，リストを読み終わった時点でアルゴリズムは終了する．

TA の問題点として，TA はスコア関数が非単調であるものに対しては適用することができない．なぜなら，非単調なスコア関数では，未アクセスのデータの上限のスコアを正しく推定することができないからである．また，TA は top-k の対象となるデータが事前に与えられ，各属性値のソートされたリストを必要とする．よってそのようなリストが用意できない場合には適用することができない. 動的にスコアが決定される状況，例えばスコア関数の入力として，ユーザーのプロファイルが与えられたときに検索結果が変化するケースなどが該当する．

3.1.2　メッシュ分割法 (等分割法)

メッシュ分割法は佐々木らによって提案された [3]．メッシュ分割法は，検索対象データに関するインデックスを必要とせず，任意のスコア関数に対して適用可能なアルゴリズムである．事前にスコア関数 S が取りうる空間をメッシュに分割し，各メッシュの取りうる最大値と最小値を記録する．クエリ実行時にはデータ集合 D に含まれるデータをメッシュ分割方法 H に従ってメッシュに配置し，各メッシュが取りうる最大値，最小値を利用して効率的に探索することで Top-k 検索を行う．

佐々木らは，メッシュの分割方法について等分割法を提案した．等分割法とは，データの各属性値の定義域を $2^h (h \geqq 1)$ 個に等分する分割方法である．

このメッシュ分割法は，各メッシュの最大値，最小値を適切に求めるために，スコア関数が既知であることが必要である．したがって，機械学習によるランキング学習などには適用できない．これはスコア関数が複雑であり，各メッシュの最大値，最小値を求めることが困難なためである．また，データ分布を考慮していないため，一部のメッシュにデータが偏るような分布の場合，例えば一つのメッシュにすべてのデータが属するような場合では，全探索を行っているのと同義になる．等分割法ではより探索空間を絞り込むためにメッシュを細かく分割する必要がある．例えば 100 次元の時を考えると，$h = 2$ のときに各属性値の定義域は 4 分割されるため $4^{100} \simeq 10^{60}$ 個のメッシュを，$h = 5$ のときには $32^{100} \simeq 10^{150}$ 個のメッシュを保持しなくてはならず，高次元の実装は困難である．

筆者は過去の研究において，スコア関数のピークを囲むようにメッシュを分割することで等分割法のメッシュへのアクセスを改善した [4]．しかしピークを超直方体で囲むことでメッシュを構築しているために高次元ではメッシュ内のデータのスコアに大きな差が生じてしまい，取り出すデータ数が増加してしまうこととなった．

3.2　課　題　点

以上を踏まえて Top-k アルゴリズムには，スコア関数とデータ分布の観点から表 1 に示すように大きく 4 つに分類される．

表 1　Top-k アルゴリズムの分類

		データ分布	
		既知	未知
スコア関数	既知	TA など	メッシュ分割法
	未知	メッシュ分割法, 提案手法	

また，既存の手法は高次元になるほど次元の呪いによってデータを探索する空間が広がってしまい，効率を落とすことが課題となっている．よって以下をすべて満たすようなアルゴリズムは未だに知られていない．

- スコア関数の性質が未知である
- 検索対象のデータのインデックスを事前に作成できない
- 検索対象のデータが高次元である

ここで，すべてのデータをスコアリングしソートする全探索法 (Naive) について考える．全探索法の計算量は $\mathrm{O}(nf + n \log n)$ と表される．ただし，f は一つのデータをスコアリングする時間とする．実際の検索では，データのスコアリングにかかる時間が全体の殆どの割合を占めていることが知られている．例えばランキング学習といった機械学習を用いたスコア関数は計算コストが高い．すなわちスコアリングする前にすべてのデータ

を如何に効率よくふるいにかけるかが非常に重要となる．データはスコア関数のピークに近づくほどスコアが高くなると推測できる．そこで，Top-k の候補となるデータの抽出をスコア関数のピークの最近傍探索問題として捉えて提案手法を考案した．

4 提案手法

本章では，LSH を用いた Top-k 検索アルゴリズムを提案する．スコア関数のピークを予め調査し，ピークの近傍データを Top-k の候補として抽出することで高次元の場合にもスコアリングするデータ数を減らすことが狙いである．

提案手法はデータが与えられる前のオフラインと与えられた後のオンラインの 2 つの段階に分かれる．

オフラインではスコア関数に対して訓練データを与え，そのピークを探索する．ここで，スコア関数は性質が未知でありデータの分布には偏りがあることも考慮してピークの探索を行う．探索したピークが複数得られた場合に，それぞれのピークのスコアに応じて優先順位をつける．

オンラインでは，まず検索対象のデータ集合のすべてのデータに対して LSH を適用しハッシュバケットに格納する．その後オフライン時に求めたピークと同じバケットに格納されたデータを抽出し，取り出されたデータに対してスコアリングとソートを行い Top-k を返す．

4.1 スコア関数のピークの探索

ランキング学習などの機械学習で得られるような性質が未知であるスコア関数では，傾きを計算してピークを探索することは困難である．また，複数のピークが存在する場合に，実際のデータ分布とはかけ離れているピークの情報は重要ではない．そこで提案手法では訓練データセットを与え，式 (7) のようにある訓練データ $\boldsymbol{q}$ がピークである条件をそのデータのスコアが距離 ϵ 内の他の任意のデータ $\boldsymbol{v}$ のスコアよりも高いこととした．

$$
\begin{aligned}
&d : dimension, f : score function \\
&B(\boldsymbol{q}, \epsilon) = \boldsymbol{v} : |\boldsymbol{q} - \boldsymbol{v}| \leqq \epsilon,\ B \subset R^d \\
&if\ f(\boldsymbol{q}) \geqq f(\boldsymbol{v})\ \forall \boldsymbol{v} \in B(\boldsymbol{q}, \epsilon)\ then\ \boldsymbol{q}\ is\ the\ peak
\end{aligned} \tag{7}
$$

4.2 ピークの優先順位

4.1 節で得られたそれぞれのピークにはスコアの差が生じる場合がある．他のピークに比べて極端にスコアが低いようなピークの周囲のデータを取得する必要はないため，ピークに優先順位をつける必要がある．その優先順位をつける目安として，図 5 で表されるような v-optimal ヒストグラムを用いる．v-optimal ヒストグラムは，各ビンの度数が可能な限り均一になるように設定したヒストグラムである．すべての訓練データのスコアを横軸とし，各ビンに属するデータ数が均等になるようにヒストグラムを形成する．各ピークはどのビンに属するかによって優先順位が定められる．すなわちスコアが高いビンに属するピークの優先順位が高くなるように設定される．ただし，同じビンに属するピークの優先順位は等しいものとする．

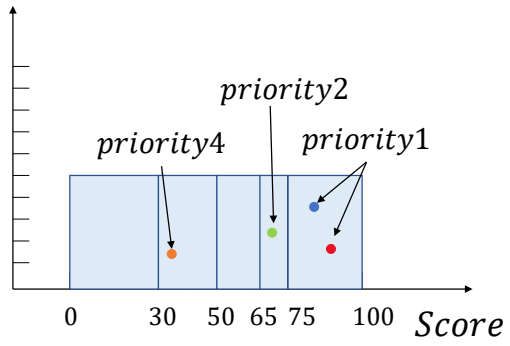


図 5　v-optimal ヒストグラムとピークの優先順位

4.3 データ抽出

オンラインでは，検索対象のデータすべてに LSH を適用し，バケットに格納する．図 6 はオンラインの流れのイメージを表したものである．4.1, 4.2 節で得られたピークを優先順位に沿って LSH に適用し，同じバケットに属するデータを抽出する．LSH に適用するピークの優先順位の割合が，求めたい Top-k の全体のデータ集合に占める割合を超えたタイミングでデータの抽出を終了する．

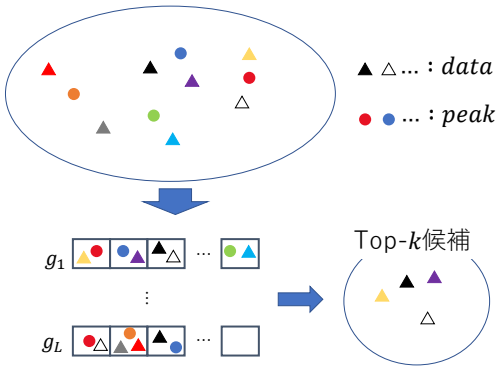


図 6　オンラインのアルゴリズムのイメージ

例 1 (データ抽出)．検索対象のデータ数が 1.0×10^4 のとき，*Top-1000* の結果，すなわち上位 *10%*を求めたいとする．*v-optimal* ヒストグラムのビン数が *20* だと仮定すると，スコアが上位 *10%*に属するビンからピークを取り出す．すなわち，*v-optimal* ヒストグラムの *2* 番目のビンに属するすべてのピークを *LSH* に適用した時点で，データの抽出を終了する．

4.4 計算量について

ここで，全探索法と提案手法の計算量を比較する．本節において用いる文字を表 2 に示した．表 3 は各計算量を比較したものである．なお，データ格納とは，各データを LSH に格納することを指す．

スコア関数によって全探索法よりも提案手法が効率よく探索を行うことができるということを示す．まずは実験結果より全探索法と提案手法の両方においてソートにかかる時間は大きく差がつかないことが過去の研究から分かっている．次に提案手

表 2　本節で用いる文字

文字	意味
f	スコア関数の計算コスト
j, L	LSH のパラメータ
m'_i	あるピーク $peak_i$ から取り出されたデータ数
$\bar{m'}$	m' の平均
m	スコアリングするデータ数
n	検索対象のデータ数
p	LSH に適用するピークの数

表 3　計算量の比較

	データ格納	データ抽出	スコアリング	ソート
全探索法	-	-	$O(nf)$	$O(n \log n)$
提案手法	$O(jLn)$	$O(jLp\bar{m'})$	$O(mf)$	$O(m \log m)$

法におけるデータ格納とデータ抽出についてそれぞれ考察を行う．データをハッシュへ追加する際の計算時間は，データ数 n と LSH のパラメータである j, L に依存する．データに対してハッシュ関数 h を j 回適用して一つのハッシュ値を得る．この計算を L 個のハッシュテーブルに対して行う．これは，データの次元数が一定であればスコア関数の計算時間によらないことを示している．データ抽出には事前に求めたスコア関数のピークの数と，それぞれのピークと同じバケットに属しているデータ数に依存する．LSH に適用するピークの数 p だけハッシュ値を計算し，同じバケットに属するデータを取り出す．あるピーク $peak_i$ から取り出されたデータ数を m'_i とする．m'_i の平均を $\bar{m'}$ とする．m'_i を合計し重複しているデータを取り除いた値が m となる．提案手法のスコアリングの計算時間は抽出されたデータの個数 m によって変化する．以上を踏まえると，提案手法が全探索法よりも効率よく探索できる条件は式 (8) で表される．

$$\frac{O(jL \cdot (n + p \cdot \bar{m'}))}{O(n - m)} < O(f) \tag{8}$$

LSH の各パラメータを適切に設定することで，ピークと同じバケットに属するデータ数や，抽出されるデータ数を抑えることができる．そこで，スコア関数のピークの数 p とピークと同じバケットに属するデータの数の平均 $\bar{m'}$，抽出されるデータ数 m が n に比べて十分に小さいとき，式 (8) は次の式 (9) で近似される．

$$O(jL) < O(f) \tag{9}$$

すなわち，一つのデータのハッシュ値を計算する時間が，一つのデータをスコアリングする時間よりも短いとき，提案手法は効率的に探索を行うことができる．

5　評 価 実 験

本章では，提案手法を全探索法と比較し，実行時間と精度の観点で評価を行う．

5.1　実 験 環 境

実験は，CPU Intel(R) Xeon(R) E3-1220 v6 @ 3.00GHz とメインメモリ 32GB で行った．オペレーティングシステムは CentOS Linux 7.7.1908 である．すべてのアルゴリズムは C++ で実装した．提案手法と全探索法の実行時間と精度の評価を行った．

実験で使うパラメータについて述べる．d はデータの次元数，n は検索対象の全データ数，k は取得すべきスコアが上位のデータの件数を表す．j, L, W はそれぞれ 2. 2. 1 節で紹介したものを表すこととし，グラフ中のパラメータは LSH(j, L, W) の順番で表現するものとする．ピークの優先順位をつける v-optimal ヒストグラムのビン数は 20 とする．データはそれぞれの属性において [0,1] の範囲の値を取るものとし，実験データはデータサイズ n と次元数 d，データ分布 $dist$ をパラメータとして生成した．データ分布 $dist$ は一様分布を表す **unif** と，任意の 2 属性の相関が $\rho = 0.8$ である **corr** を用意した．**corr** は $N((0.5, ..., 0.5)^T, \sum_{corr})$ に従う多変量正規乱数を n 個生成した後，範囲 [0,1] で正規化した分布である．$\sum_{corr}$ は式 (10) で表される．

$$\sum_{corr} = \begin{pmatrix} 1 & \rho & \dots & \rho \\ \rho & 1 & \dots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \dots & 1 \end{pmatrix} \tag{10}$$

検索精度の評価指標は nDCG を用いて計測した．なお，提案手法は LSH の各パラメータの値によっては k 件以上のデータを取得できない場合がある．その場合には不足しているデータ分のスコアを 0 として nDCG を計算する．例えば $k = 10$ の検索に対して 6 件のみデータを取り出した場合，7 位から 10 位までのスコアを 0 として計算する．

スコア関数は Rastrigin 関数を用意した．Rastrigin 関数は式 (11) で表される関数であり，多峰性関数である．非常に多くの局所解を持つことが特徴である．データの属性値の定義域が [0,1]，$\alpha = 1$ であるとき，Rastrigin 関数のピークの解析解は一つだけであることが知られている．今回は多峰性の関数における実行時間と精度についても評価を行うため，$\alpha = 1, 2$ として実験を行う．また，実際の検索では一つのデータのスコアリングに時間がかかることが想定される．事前実験において，ランキング学習のスコアリングにかかる時間が Rastrigin 関数の計算時間の約 200 倍を記録したことから，実験のスコアリングの際には一つのデータのスコアリングに 200 回同じ計算を繰り返すものとする．

$$f(x_1, ..., x_d) = 10d + \sum_{i=1}^{d} ((\alpha x_i)^2 - 10cos(2\pi\alpha x_i)) \tag{11}$$

今回の実験で用いるパラメータを表 4 に示す．LSH の各パラメータは実験ごとに変更する．

5.2　ピークの探索

本実験では，4. 1 節にあったように訓練データ同士の距離を比較し，ピークを求めることとした．なお，訓練データ数を増

表 4　実験で用いるパラメータ

パラメータ	値
データ数　n	1.0×10^6
次元数　d	500
データ分布　$dist$	**unif**, **corr**
Rastrigin 関数　α	$1, 2$
求める上位件数　k	100
LSH(j, L, W)	実験ごとに設定

加させるほどより正確にピークを見つけることが可能である．今回は訓練データ数を 5.0×10^5 に固定し実験を行った．訓練データは計測するデータとは別に生成した．ϵ は式 (12) で求めた値を設定した．これは仮にすべてのデータが空間に格子状に散らばっていると仮定したときの近傍格子の距離である．

$$\epsilon = \frac{1}{\sqrt[d]{n} - 1} \tag{12}$$

表 5 は Rastrigin 関数の $\alpha = 1, 2, dist =$ **unif** のもとで，訓練データセットから得られた優先順位ごとのピークの個数を表したものである．優先順位は 4. 2 節で述べたものを指し，値が小さいものほど先に LSH に適用されるものとする．

表 5　探索により得られたピークの数

		(α, d, $\frac{n}{1.0\times10^5}$)			
		(1,10,5.0)	(2,10,5.0)	(1,500,5.0)	(2,500,5.0)
優先順位	1	22	2202	1	1
	2	7	206	0	0
	3	6	94	0	0
	4	3	54	0	0
	5	6	46	0	0
	6 以降	621	535	0	0

$\alpha = 1$ のとき，各属性値が [0,1] の範囲において Rastrigin 関数のピークの解析解は一つだけであることが知られている．そのため $\alpha = 2$ のときよりも探索で得られたピークの個数は少ない．訓練データ同士の比較によってピークを求めているため，解析解のピークの数とは異なる．各属性値の定義域が [0,1]，$\alpha = 2$ のときの Rastrigin 関数は多峰性の関数であるが，$d = 500$ において探索で得られたピークの数が解析解のピークの数よりも遥かに少ないことについて考察する．これは，データ数を固定し次元数を大きくした場合，次元の呪いによって点密度が疎になることが原因である．高次元ではピークであるかどうか判断するための比較対象のデータとのユークリッド距離が大きく，Rastrigin 関数の細かい起伏を無視してしまうために得られるピークの数が少なくなってしまう．なお予備実験により，ピークの優先順位をつける v-optimal ヒストグラムのビン数を増加させると LSH に適用するピークの数は減少する傾向にあることがわかっている．

5. 3　実 験 結 果

それぞれの実験は 5 回繰り返し，実行時間と nDCG の平均値と標準誤差をグラフ化した．

5. 3. 1　$d = 500, dist =$ **unif** の実験結果

表 6 に本節で用いた LSH のパラメータを示す．

表 6　LSH のパラメータ ($d = 500$)

パラメータ	値
j(LSH)	$5, 7, 10$
L(LSH)	$3, 5, 7$
W(LSH)	$3, 5, 7, 10, 15$

図 7 は $d = 500, dist =$ **unif**, $\alpha = 1$ とした場合の全探索法と提案手法の実験結果を示したものである．図 7(b) は，LSH のパラメータ $L = 5, W = 7$ で固定したときの j の変化による各手法の実行時間と nDCG を示したものである．j が増加するほどそれぞれのデータは分散してバケットに属するようになり，最終的に取り出すデータ量を抑えることができるため，スコアリングにかかる時間が減少することが分かる．また j の値に応じて，ハッシュ値を求める際に式 (4) の計算回数が定まるため，j の増加に応じてデータのハッシュへの追加にかかる時間も増加する．nDCG は j の増加に伴い，各データがピークと同じバケットに属する条件が厳しくなるため，減少している．

図 7(c) は LSH のパラメータ $j = 7, W = 7$ で固定したときの L の変化による各手法の実行時間と nDCG を示したものである．L の値が小さい場合にはハッシュテーブルの数が少ないため，データのハッシュへの追加にかかる計算時間を抑えることができる．L の値が大きい場合にはデータのハッシュへの追加にかかる計算時間は増加するものの，データがピークと同じバケットに属する機会が増えることで nDCG の増加が見込める．

図 7(d) は LSH のパラメータ $j = 7, L = 5$ で固定したときの W の変化による各手法の実行時間と nDCG を示したものである．W は $\boldsymbol{a} \cdot \boldsymbol{v}$ の写像先においてどれほどの幅でハッシュ値の近似を行うかに関わるパラメータである．そのため，W の増加に応じて各データが同じバケットに入る確率が高くなり，抽出するデータが増加する．それに伴いスコアリングにかかる時間も増加している．

図 8 は $d = 500, dist =$ **unif**, $\alpha = 2$ とした場合の全探索法と提案手法の実験結果を示したものである．100 次元に対して与えられるデータ数が少なく Rastrigin 関数の細かな起伏が無視されているために，データ分布が一様である場合には $\alpha = 1$ のときと同様な結果が得られる．

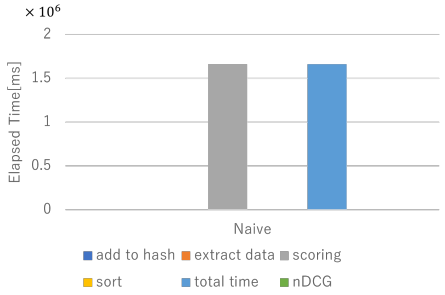
5. 3. 2　$d = 500, dist =$ **corr** の実験結果

表 7 に本節で用いた LSH のパラメータを示す．データ分布が偏っているため，よりピークに近い範囲にデータが多数存在する．

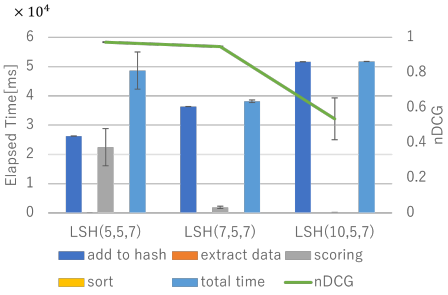
表 7　LSH のパラメータ ($d = 500$)

パラメータ	値
j(LSH)	$10, 20$
L(LSH)	5
W(LSH)	$5, 7, 10$

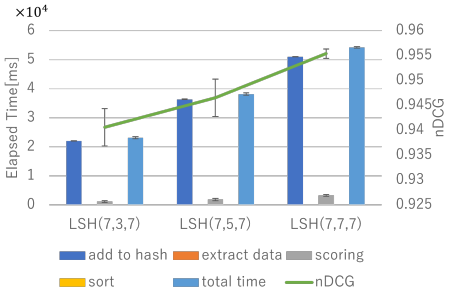
図 9 は $d = 500, dist =$ **corr** とした場合の全探索法と提案手



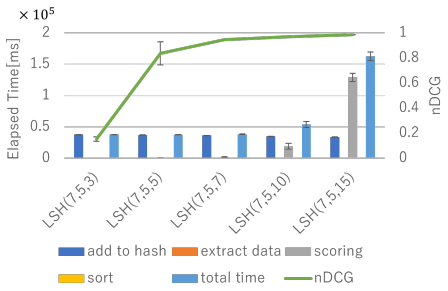
(a) 全探索法の実行時間



(b) LSH の j の変化による実行時間と精度の変化

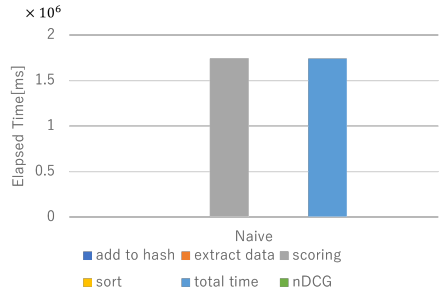


(c) LSH の L の変化による実行時間と精度の変化

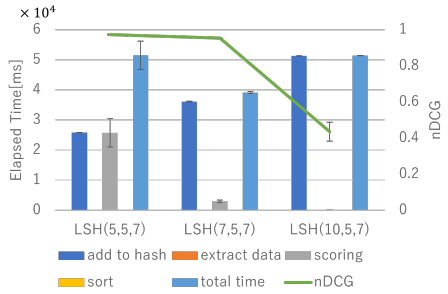


(d) LSH の W の変化による実行時間と精度の変化

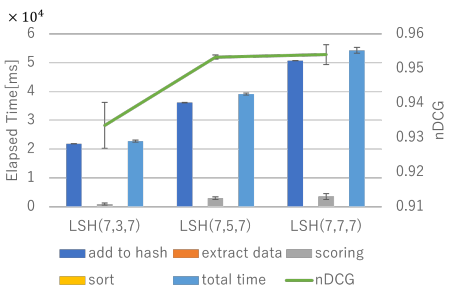
図 7　$d = 500, dist = \mathbf{unif}, \alpha = 1$ の場合の実行時間と精度の変化



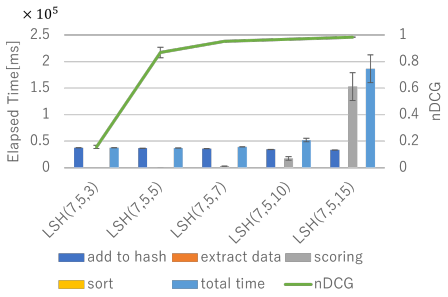
(a) 全探索法の実行時間



(b) LSH の j の変化による実行時間と精度の変化



(c) LSH の L の変化による実行時間と精度の変化



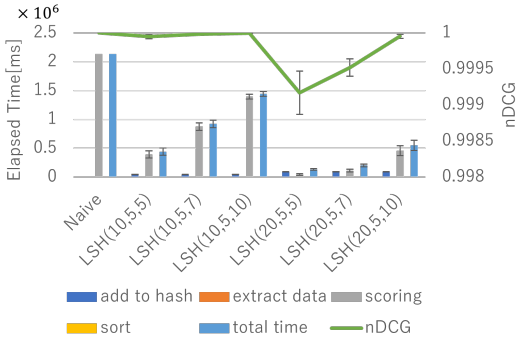
(d) LSH の W の変化による実行時間と精度の変化

図 8　$d = 500, dist = \mathbf{unif}, \alpha = 2$ の場合の実行時間と精度の変化

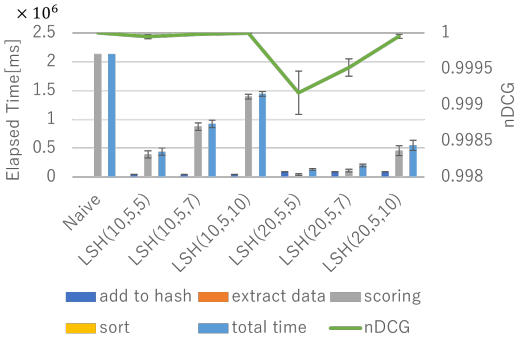
法の実験結果を示したものである．図 9(a) からわかるように元のスコア関数のピークが一つの場合は，データの偏りがある場合にも LSH のパラメータを調整することで nDCG が 0.99 以上を維持しつつ最大で約 15 倍の計算速度を記録することができた．しかし図 9(b) からわかるように，多峰性の関数においてデータの偏りがある場合には，提案手法は有効ではない．これは，データに偏りがある場合にも式 (12) で表される ϵ を用いてピークを探索しており，本来取得すべきピークの数よりも少なくピークを探索していることが原因である．ピークが一つの場合には提案手法が有効であるため，ϵ の値をデータ分布に応じて適切に設定することができれば，より正確にピークを探索することができ効率よく探索を行うことができると考えられる．

6　おわりに

本稿では，スコア関数が未知であり，高次元でのデータ集合に対して適用可能な Top-k 検索アルゴリズムとして，LSH を



(a) $\alpha = 1$ のときの実行時間と精度の変化



(b) $\alpha = 2$ のときの実行時間と精度の変化

図 9　$d = 500, dist =$ **corr** の場合の実行時間と精度の変化

用いた手法を提案した．LSH のパラメータである j, L, W をスコア関数に応じて適切に設定することで，nDCG が 0.99 以上を維持しながら全探索法と比較して最大で約 15 倍の計算速度を記録することを示した．

提案手法では，実際のデータ分布には偏りがあることを考慮し，一定範囲内の訓練データ同士を比較することでスコア関数の擬似的なピークを探索した．この手法は新しいデータが追加されるたびに再度ピークを更新することができる．しかし，ピークを判断するための比較対象のデータとの距離 ϵ の定め方については議論の余地がある．提案手法では，検索対象データが空間に格子状に散らばっていると仮定した格子の隣接距離を ϵ としたが，高次元においてはスコア関数の起伏を無視し多峰性の関数の性質を利用することができない場合がある．また，データの分布に偏りがある際に想定するピークの数よりも少なくピークを探索してしまう．したがって，提案手法の課題としてデータ分布を考慮した正確なピークの探索が挙げられる．

謝　　辞

本研究の一部は，JSPS 科研費 (18H03242，18H03342，19H01138) の助成を受けたものである．

文　　献

[1] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In Journal of Computer and System Sciences, Vol. 66, pp. 614–656, 2003.

[2] Ihab Ilyas, George Beskales, and Mohamed Soliman. A survey of top-k query processing techniques in relational database systems. ACM Computing Surveys, Vol. 40, No. 4, pp.1–58, 2008.

[3] 佐々木 夢, 欅　惇志, 宮崎純, 検索対象データの事前インデックスを必要としない Top-k 検索アルゴリズムの提案と評価, DEIM Forum, C6-2, 2017.

[4] 池田 達樹, 宮崎 純, 未知スコア関数に対する Top-k 検索アルゴリズムの提案, IEICE-DE2020-3, 2020-06-20.

[5] Indyk, Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. STOC'98, section 4.2.

[6] M. Datar, P. Indyk, N. Immorlica and V. Mirrokni, Locality-Sensitive Hashing Scheme Based on p-Stable Distributions, In Proceedings of the Symposium on Computational Geometry, 2004.

[7] Dong Xin, Jiawei Han, Kevin Chen-Chuan Chang. Progressive and selective merge: computing top-k with ad-hoc ranking functions. In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. pp.103–114, 2007.

[8] Zhen Zhang, Seung-won Hwang, Kevin Chang, Min Wang, Christian Lang, and Yuan-Chi Chang. Boolean + ranking: querying a database by k-constrained optimization. In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. pp. 359–370, 2006.

[9] Sayan Ranu, Ambuj K. Singh. Answering Topk Queries Over a Mixture of Attractive and Repulsive Dimensions. arXiv:1111.7165v1 pp.169-180, 2011.

[10] Prasad M Deshpande, Deepak P,Krishna Kummamuru, Efficient Online Top-k Retrieval with Arbitrary Similarity Measures, in Proceedings of the 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, pp.356-367, 2008.

[11] Yufei Tao, Vagelis Hristidis, Dimitris Papadias, Yannis Papakonstantinou. Branch-and-bound processing of ranked queries. ScienceDirect, Information Systems, Volume 32, pp.424-445, 2007.

[12] 和田 俊和, 最近傍探索の理論とアルゴリズム, 2009 情報処理学会 CVIM 研究会, vol.2009-CVIM-169, no.12, pp.1-12, Nov.2009.