CD-002

内積空間におけるカーディナリティ推定

平田皓平 *　天方大地 *　原隆浩 *
Kohei Hirata　Daichi Amagata　Takahiro Hara

抄録

近年，推薦システムでは，MF（Matrix Factorization ）と呼ばれる，ユーザの未評価アイテムに対して，評価の予測値を取得する技術が用いられることが多い．MF は評価行列から各ユーザのベクトルおよび各アイテムのベクトルを得る．ユーザとアイテムとの内積は，ユーザのアイテムに対する評価や興味を表す．ここで，あるアイテムとの内積が大きくなるユーザ数を知ることで，そのアイテムの市場規模を知ることができる．この数を内積空間におけるカーディナリティとする．本研究では，これを高速・高精度に推定するアルゴリズムを提案する．実データを用いた実験により，提案アルゴリズムの有効性を確認する．

1　はじめに

近年，e コマースやソーシャルネットワーキングサービスの普及により，様々なアプリケーションで推薦システムが利用されている [12, 20]．推薦システムでは，協調フィルタリングが用いられることが多い [19]．協調フィルタリング手法の 1 つとして MF（Matrix factorization）と呼ばれる技術がある [3, 9, 25]．MF は，ユーザの未評価アイテムに対して評価の予測値を取得する手法である．MF は，ユーザがアイテムに対して行った評価を記録した評価行列を，ユーザの行列とアイテムの行列を表す 2 つの行列に分解する．そして，分解した 2 つの行列を掛け合わせることで，ユーザの未評価アイテムへの評価の予測値を得る．この評価の予測値を基にユーザにアイテムが推薦される．また，ユーザの行列とアイテムの行列から得られる評価行列の各要素は，ユーザベクトルとアイテムベクトルの内積である．内積が大きいほど，ユーザがアイテムに対して満足する可能性が高く，また，興味を示す可能性が高い．

ここで，あるアイテムに興味を示すユーザ数を基に，そのアイテムの影響度・購買規模を推定できる．あるアイテムに興味を示すユーザ数を知ることは市場規模を推定する上で重要である．内積空間におけるカーディナリティを，あるアイテムベクトルをクエリとした時，クエリとの内積が大きくなるユーザベクトルの数とする．

内積空間におけるカーディナリティを求める単純な手法は，対象となる全てのデータとクエリとの内積を計算し，内積が閾値以上となるかを調べるものである．しかし，多くの e コマース企業では大量のデータを管理し，また，ベクトルの次元数も一般的に大きいため，この手法では膨大な時間がかかってしまう．また，カーディナリティは厳密である必要はなく，近似値で良い．そのため，カーディナリティ推定技術には，効率性と高精度性が求められる．クエリとの距離が閾値以下となるデータ数を推定する問題の研究は行われている [7, 10, 14, 15, 26] が，内積空間におけるカーディナリティを推定する研究は行われていない．本研究では内積空間におけるカーディナリティを高速に推定するための新しいアルゴリズムを提案する．

提案アルゴリズムは，オフラインフェーズとオンラインフェーズに分かれる．オフラインフェーズでは，データをノルム順にソートしておき，ノルムの大きさに応じてグループを作成する．次に，内積空間におけるカーディナリティを角距離に関するカーディナリティに置き換えるためにデータを変換する．そして，角距離に関するカーディナリティを推定する手法 [27] を使用するために，変換したデータを LSH（Locality Sensitive Hashing）[5] でハッシュテーブルに格納しておく．オンラインフェーズでは，コーシーシュワルツの不等式により，クエリとの内積が閾値以上になることがないデータを除外する．そして，除外されなかったデータに対して，[27] を用いることで，内積空間におけるカーディナリティを求める．

本研究の貢献は以下の通りである．

*大阪大学大学院情報科学研究科マルチメディア工学専攻

・内積空間におけるカーディナリティ推定アルゴリズムを提案する．

・実データを用いた実験により，提案アルゴリズムの有効性を確認する．

以降では，2 章で問題定義および予備知識の説明を行い，3 章で関連研究を紹介する．そして，4 章で提案アルゴリズムの詳細を説明し，5 章で実データを用いた実験により提案アルゴリズムの性能評価を行う．最後に 6 章でまとめを行う．

2 予備知識

2.1 定義

定義 1（内積）． ベクトル $x = (x_0, ..., x_{d-1}) \in \mathbb{R}^d$ およびベクトル $y = (y_0, ..., y_{d-1}) \in \mathbb{R}^d$ が与えられた時，x と y の内積を $x \cdot y$ と表記する．また，$x \cdot y = \sum_{i=0}^{d-1} x_i y_i$ である．

問題定義. データセット P，クエリ q，および閾値 τ が与えられたとき，$q \cdot p \geq \tau$ を満たすデータ $p \in P$ の数 (カーディナリティ) を推定する．

2.2 角距離に関するカーディナリティ推定の既存手法

文献 [27] では，単位超球面上のベクトル $\bar{p}_0, ..., \bar{p}_{n-1}$ が要素である集合 $\bar{P}$，単位超球面上のクエリベクトル $\bar{q}$，および閾値 a が与えられたとき，$\bar{q}$ と $\bar{p}$ とのなす角度を $\theta_{\bar{q}\bar{p}}$ とし，$\theta_{\bar{q}\bar{p}} \leq a$ となる $\bar{p} \in \bar{P}$ の数を推定するアルゴリズムが提案されている．

定義 2（角距離に関するカーディナリティ推定）． 単位超球面上のベクトル $\bar{p}_0, ..., \bar{p}_{n-1}$ が要素である集合 $\bar{P}$，単位超球面上のクエリベクトル $\bar{q}$，および閾値 a が与えられたとする．また，$\bar{q}$ と $\bar{p}$ とのなす角度を $\theta_{\bar{q}\bar{p}}$ とし，$\theta_{\bar{q}\bar{p}}$ が a 以下となるデータ $\bar{p}$ を要素とする集合を $A_{\bar{q}}$ とする．このとき，$|A_{\bar{q}}|$ を推定する．

この手法では，オフラインフェーズで角距離のための LSH[5] を用いて $\bar{P}$ をハッシュテーブルで管理しておき，オンラインフェーズでは作成したハッシュテーブルを使用し，$\theta_{\bar{q}\bar{p}} \leq a$ を満たすデータを重点的にサンプリングすることでカーディナリティを推定する．オフラインフェーズおよびオンラインフェーズについて詳細を説明する．

2.2.1 オフラインフェーズ

文献 [5] で提案されている LSH は，角度が小さいベクトル同士が高確率で同じハッシュ値を持つハッシュ関数である．このハッシュ関数を式 1 に示す．r は各要素が平均 0，分散 1 の正規分布に従うベクトルである．

$$h_r(\bar{p}) = \begin{cases} 0 & (\text{if} \quad r \cdot \bar{p} \leq 0) \\ 1 & (\text{otherwise}) \end{cases} \tag{1}$$

また，2 つのベクトル $\bar{q}$ および $\bar{x}$ が同じハッシュ値になる確率を式 2 に記す．$\theta_{\bar{q}\bar{x}}$ は $\bar{q}$ と $\bar{x}$ のなす角度であり，$\theta_{\bar{q}\bar{x}} = \cos^{-1}(\bar{q} \cdot \bar{x})$ である．

$$\Pr[h_r(\bar{q}) = h_r(\bar{x})] = 1 - \frac{\theta_{\bar{q}\bar{x}}}{\pi} \tag{2}$$

このハッシュ関数を t 個組み合わせたハッシュベクトル $g(\bar{p}) = (h_0(\bar{p}), ..., h_{t-1}(\bar{p}))$ を $\bar{p}_0, ..., \bar{p}_{n-1}$ に対して求める．そして，求めたハッシュベクトルをキーとするハッシュテーブルで $\bar{p}$ を管理しておく．さらに，これを K 回繰り返して K 個のハッシュテーブルを作成する．また，各バケットに入っているベクトル数を記録しておく．

2.2.2 オンラインフェーズ

各テーブルにおいて，クエリ $\bar{q}$ のハッシュベクトルを求める．次に，求めたハッシュベクトルからハミング距離が μ 以下となるキーを持つバケットを探す．ただし，本研究において，μ は a を用いてチューニングする．LSH の性質から，ハミング距離が μ 以下のバケットに存在するベクトルは，高確率でクエリとの角距離が a 以下となるベクトルである．つまり，これらのバケットからベクトルをランダムにサンプリングすることで，$\theta_{\bar{q}\bar{p}} \leq a$ を満たすベクトルを重点的にサンプリングできる．以降ではサンプリングの対象となり得るベクトルを集めた集合をサンプリングプールという．

次に，サンプル $\bar{x}$ から $|A_{\bar{q}}|$ を推定する手順について説明する．まず，ハッシュベクトルの長さが t であるため，式 2 よりサンプル $\bar{x}$ およびクエリ $\bar{q}$ のそれぞれのハッシュベクトルがハミング距離 i となる確率は式 3 となる．

$$\mathbb{P}(d_{\bar{q}\bar{x}} = i | \theta_{\bar{q}\bar{x}}) = \binom{t}{i} \left(1 - \frac{\theta_{\bar{q}\bar{x}}}{\pi}\right)^{t-i} \left(\frac{\theta_{\bar{q}\bar{x}}}{\pi}\right)^{i} \tag{3}$$

ここで，$d_{\bar{q}\bar{x}}$ は，サンプル $\bar{x}$ のハッシュベクトルとクエリ $\bar{q}$ のハッシュベクトルとのハミング距離を表す．また，

$\bar{q}$ のハッシュベクトルからハミング距離が μ 以下となるバケットに，$\bar{x}$ が入る確率 $p(\bar{x})$ は式 4 から求まる．

$$p(\bar{x}) = \sum_{i=0}^{\mu} \mathbb{P}(d_{\bar{q}\bar{x}} = i | \theta_{\bar{q}\bar{x}}) \tag{4}$$

次に，確率変数 Z を式 5 で定義する．

$$Z = \begin{cases} \frac{\sum_{k=1}^{K} C_{\bar{q}}^{k}(\mu)}{K \cdot p(\bar{x})} & (\text{if} \quad \theta_{\bar{q}\bar{x}} \leq a) \\ 0 & (\text{otherwise}) \end{cases} \tag{5}$$

ここで $C_{\bar{q}}^{k}(\mu)$ は，テーブル k においてクエリ $\bar{q}$ のハッシュベクトルからハミング距離が μ 以下となるバケットに入っているベクトル数の合計値である．s 回サンプリングを行い，式 5 から $Z_1, Z_2, ..., Z_s$ を求める．これらの平均値 $\frac{\sum_{i=1}^{s} Z_i}{s}$ をカーディナリティ $|A_{\bar{q}}|$ の推定値とする．また，文献 [27] で定理 1 が成り立つことが示されている．

定理 1. $\mathbb{E}[\frac{\sum_{i=1}^{s} Z_i}{s}] = |A_{\bar{q}}|$

3 関連研究

様々な距離空間においてカーディナリティを推定するアルゴリズムが提案されている [7, 10, 14, 15, 26]．文献 [14] では，メトリック空間におけるカーディナリティ推定アルゴリズムが提案されている．このアルゴリズムは，カーネル密度推定を利用したものである．クエリとサンプル点の距離にチューニングされたカーネル関数を適応させることで，クエリとの距離が閾値以下になる確率を推定し，その確率からカーディナリティを推定する．文献 [26] では，深層学習を利用し，特徴抽出モデルと回帰モデルで構成されたアルゴリズムを提案している．これらの手法は距離空間におけるカーディナリティを推定するものであり，内積には対応していない．

また，クエリベクトル q および P が与えられたとき，q との内積が上位 k 個となる p を解として出力する k-MIPS と呼ばれる問題を解くためのアルゴリズムが提案されている [1, 2, 6, 11, 16, 17, 21–24]．さらに，k-MIPS をユークリッド空間における k 最近傍検索に置き換えることで解を求めるアルゴリズムも提案されている [3, 4, 8, 13, 18]．文献 [11] では，FEXIPRO というアルゴリズムが提案されている．このアルゴリズムは，特異値分解，整数近似，および正数変換の 3 つの要素からなる枝刈りを行い，残ったデータに対してシーケンシャルスキャンをすることで高速に k-MIPS を解く．k-MIPS を解くための手法において，閾値を τ にすることにより本問題を解くことはできる．しかし，これらの手法は推定ではなく検索を行うため，ベクトルのアクセス回数が多く，カーディナリティ推定に対して非効率的である．

4 提案アルゴリズム

内積空間は，角距離空間に変換することが可能である．そのため，この変換後，2.2 節で紹介したアルゴリズムを用いてカーディナリティ推定が可能となる．しかし，この方法（ベースライン）は角度の性質のみ考慮されるため，サンプリングプール内には条件を満たさないデータも含まれている．そこで，コーシーシュワルツの不等式を用いて，ノルムが小さく，クエリとの内積が閾値を満たさないデータを除外することにより，不要なデータをサンプルする確率を下げることができる．これにより，ベースラインよりも必要なデータを重点的にサンプリングすることができる．つまり，同じサンプリング回数でもベースラインより高精度にカーディナリティを推定できる．

まず，内積空間におけるカーディナリティを角距離に関するカーディナリティに置き換える手法について説明する．次に，提案アルゴリズムのオフラインフェーズおよびオンラインフェーズについて説明する．

4.1 角距離への変換

データセット P 内のデータ p およびクエリ q を式 6 および式 7 で変換する．ただし，M_0 は P 内のデータの最大のノルムとする．

$$\bar{p} = \frac{1}{M_0} \left(p_0, p_1, ..., p_{d-1}, \sqrt{M_0^2 - ||p||^2} \right) \tag{6}$$

$$\bar{q} = \frac{1}{||q||} (q_0, q_1, ..., q_{d-1}, 0) \tag{7}$$

ここで，$||\bar{p}|| = 1$ および $||\bar{q}|| = 1$ であるため，$q \cdot p \geq \tau$ のとき，

$$\bar{q} \cdot \bar{p} = \frac{q \cdot p}{||q|| M_0} \geq \frac{\tau}{||q|| M_0} \iff \cos\theta_{\bar{q}\bar{p}} \geq \frac{\tau}{||q|| M_0} \tag{8}$$

となる．また $a = \cos^{-1} \frac{\tau}{||q|| M_0}$ とすると，式 8 から

$$\theta_{\bar{q}\bar{p}} \leq a \tag{9}$$

となる．

つまり，本問題の $q \cdot p \geq \tau$ を満たす p の数は，$\theta_{\bar{q}\bar{p}} \leq a$ を満たす $\bar{p}$ の数 $|A_{\bar{q}}|$ となる．

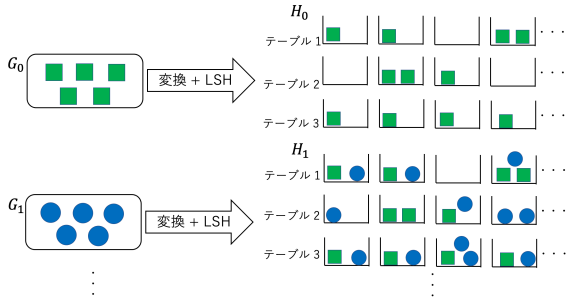

図 1: ハッシュテーブルに格納する例

4.2 オフラインフェーズ

オンラインフェーズでクエリとの内積が閾値を満たさないデータを効率的に除外するために，データをノルムの降順にソートし，グループに分割する．次に，内積空間におけるカーディナリティを角距離におけるカーディナリティに置き換えるためにデータ p を変換する．そして，ハッシュテーブルでデータを管理する．オフラインフェーズのアルゴリズムの詳細をアルゴリズム 1 に示す．

Algorithm 1: Offline phase

Input: Set P of n data vectors, number of elements of group n', length of hash vector t, and number of hash tables K

Output: number of groups b, maximum norms $\{M_0, M_1, ..., M_{b-1}\}$, and sets $\{H_0, H_1, ..., H_{b-1}\}$

1 Compute $||p||$ for all $p \in P$
2 Sort P in descending order of $||p||$
3 $b = \lceil \frac{n}{n'} \rceil$
4 $j = 0$
5 **for** $i = 0, n', 2n', ..., (b-1)n'$ **do**
6 $M_j = ||p_i||$
7 $G_j = \emptyset$
8 **for** $m = i, i+1, ..., i+(n'-1)$ and $m \leq n$ **do**
9 $G_j = G_j \cup \{p_m\}$
10 $j = j + 1$
11 **for** $j = 0, 1, ..., b-1$ **do**
12 **for** $\forall p \in G_j$ **do**
13 Transform p into $\bar{p}$ by equation 6
14 Make H_j by equation 1

まず，データをノルムの降順にソートする（1–2 行）．1 つのグループに入るデータ数を n' とし，n' からグループの数 b を求める（3 行）．その後，データをグループに分割する（4–10 行）．ソートされたデータ上位 n' 個をグループ G_0 のデータ，次の上位 n' 個をグループ G_1 のデータとしてデータをグループに分割する．ここでグループ G_j に属しているデータの最大ノルムを M_j とする．

次に，各グループ G_j 内のデータについて，式 6 を用いて p を $\bar{p}$ に変換する．その後，2.2.1 項で述べた手法と同様に式 1 のハッシュ関数を使用し，長さ t のハッシュベクトルを求める．そして，求めたハッシュベクトルをキーとするハッシュテーブルで p を管理する．これを K 回繰り返し，K 個のハッシュテーブルでデータを管理する．さらに，ハッシュテーブル内の各バケットに入っているデータ数を記録する（11–14 行）．ただし，グループ G_j における K 個のハッシュテーブルで構成される集合を H_j とし，グループ G_j 内のデータのみでなく，グループ G_0 から G_j までのデータをまとめて H_j 内の K 個のハッシュテーブルが管理するように作成する．

図 1 に $K = 3$ としてハッシュテーブルでデータを管理する例を示す．オンラインフェーズでは，連続したグループを枝刈りすることでそのグループ内のデータをまとめて枝刈りする．そして，枝刈りされなかったデータを管理しているハッシュテーブルを対象とし，そのハッシュテーブル内のある条件を満たすバケットを探索する手順がある．ここで，枝刈りされなかったデータがグループ G_0 から G_α に属している場合を考える．グループ G_j 内のデータのみを H_j 内の K 個のハッシュテーブルで管理すると，対象のハッシュテーブルは H_0 から H_α 内のものとなり，ハッシュテーブル数は $K(\alpha+1)$ 個となる．一方，グループ G_0 から G_j までのデータをまとめて H_j 内の K 個のハッシュテーブルで管理しておくことで，対象のハッシュテーブルは H_α 内のもののみであり，ハッシュテーブル数は K 個となり，対象のハッシュテーブル数を削減できる．対象のハッシュテーブル数が減少することで，探索するバケット数も減少し，バケットの探索時間を削減できる．

4.2.1 オンラインフェーズ

グループ G_j に属しているデータ p は，$M_{j+1} \leq ||p|| \leq M_j$ であり，コーシーシュワルツの不等式から

$$q \cdot p = ||q||||p|| \cos\theta_{qp} \leq ||q||||p|| \leq ||q|| \cdot M_j \qquad (10)$$

となる．よって，p と q の内積の上限を ub_j とすると，$ub_j = ||q|| \cdot M_j$ となる．ここで，$ub_j < \tau$ となるとき，$M_j \geq M_{j+1} \geq \cdots \geq M_{b-1}$ であるため，以降のグループ

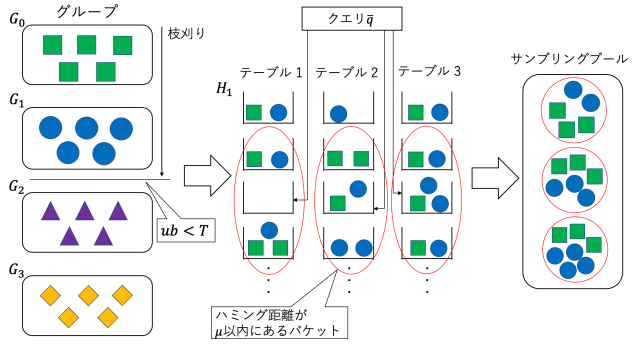

図 2: サンプリングプール作成の例

$G_j, G_{j+1}, ..., G_{b-1}$ における ub_j は τ 未満である．従って，$ub_j < \tau$ となるとき，$G_j, G_{j+1}, ..., G_{b-1}$ に属するデータ p と q との内積は閾値 τ 以上にはならないため，これらのデータを除外できる．

オンラインフェーズのアルゴリズムの詳細をアルゴリズム 2 に示す．まず，ub_j を用いてノルムが小さくクエリ q との内積が閾値 τ 以上とならないグループを除外する（1–5 行）．残ったグループの数を f とする（6 行）．

次に，式 7 より q を $\bar{q}$ に変換する（7 行）．除外されなかったグループ $G_0, G_1, ..., G_{f-1}$ に属するデータを管理しているハッシュテーブル集合 H_{f-1} について，2.2.2 項と同様にサンプリングプール SP を作成する（8–12 行）．また，ハッシュテーブル k において $\bar{q}$ のハッシュベクトルからハミング距離 μ 以下となるキーを持つバケット内のデータ数 $C^k_{\bar{q}}(\mu)$ を求める（13 行）．図 2 にサンプリングプールを作成する例を示す．

最後に，$|A_{\bar{q}}|$ を推定する（14–21 行）．サンプリングプール SP からランダムにサンプル $\bar{x}$ を選び，式 5 を適用することで Z を計算する．このサンプリングを s 回行い，$Z_1, ..., Z_s$ から平均値を求め，平均値 $\frac{\sum_{i=1}^{s} Z_i}{s}$ が $|A_{\bar{q}}|$ の推定値となる．これを解として出力する．

Algorithm 2: Online phase

Input: query q, threshold τ, sampling number s, length of hash vector t, number of hash tables K, number of groups b, sets $\{H_0, H_1, ..., H_{b-1}\}$, and maximum norms $\{M_0, M_1, ..., M_{b-1}\}$

Output: $\frac{\sum_{i=1}^{s} Z_i}{s}$

1 Compute $||q||$
2 **for** $j = 0, 1, ..., b-1$ **do**
3 　$ub_j = ||q|| \cdot M_j$
4 　**if** $ub_j < \tau$ **then**
5 　　**break**
6 $f = j$
7 $\bar{q} \leftarrow q$ transformed by equation 7
8 $SP = \emptyset$
9 **for** $k = 1, 2, ...K$ **do**
10 　Compute hash vector for $\bar{q}$
11 　$B \leftarrow$ buckets whose keys have less distance than μ from $\bar{q}$
12 　Add $\bar{p} \in B$ to SP
13 　Compute $C^k_{\bar{q}}(\mu)$
14 **for** $i = 1, 2, ..., s$ **do**
15 　Sample $\bar{x}$ from SP
16 　$\theta_{\bar{q}\bar{x}} = \cos^{-1}(\bar{q} \cdot \bar{x})$
17 　**if** $\theta_{\bar{q}\bar{x}} \leq \cos^{-1} \frac{\tau}{||q||M_0}$ **then**
18 　　Compute $p(\bar{x})$ by equation 4
19 　　$Z_i = \frac{\sum_{k=1}^{K} C^k_{\bar{q}}(\mu)}{K \cdot p(\bar{x})}$
20 　**else**
21 　　$Z_i = 0$

5 評価実験

本章では，提案アルゴリズムの性能評価のために行った実験の結果を示す．本実験は，3.2GHz Intel Core i7, および 64GB RAM で構成される PC 上で行い，全てのアルゴリズムは C++で実装した．

5.1 設定

本実験では，提案アルゴリズムおよびベースラインの評価を行った．また，コーシーシュワルツの不等式を用いた厳密解アルゴリズムとの時間比較を行った．このアルゴリズムは，データをノルムの降順にソートしておき，与えられたクエリに対して線形探索を行い，コーシーシュワルツの不等式で得られる内積の上限が閾値未満になる時に探索を終了するアルゴリズムである．

5.1.1 評価指標

本実験では，複数のクエリを発行した時の時間 [msec]（クエリ処理時間の平均）および誤差（各クエリにおける相対誤差の平均）を評価した．ただし，発行したクエリの内，実際のカーディナリティが 0 となるものに対しては，相対誤差を求めることができないため，そのようなクエリは除いて，評価指標を求めた．

5.1.2 データセット

本実験では，3 つの実データセット Netflix[1]，Yahoo! song[2]，および Amazon[3]を用いた．これらは MF によって

[1]http://archive.ics.uci.edu/ml/datasets/Netflix+Prize
[2]https://webscope.sandbox.yahoo.com/index.php
[3]https://jmcauley.ucsd.edu/data/amazon/

表 1: データセットの統計

データセット	Netflix	Yahoo! song	Amazon
ユーザ数	480,189	1,948,882	2,088,620
アイテム数	17,770	98,211	200,941
クエリ数	17,770	9,822	20,095

表 2: パラメータの設定

データセット	パラメータ	値
Netflix	τ	**4**, 4.1, 4.2, 4.3, 4.4, 4.5
	s	**1000**, 5000, 10000
Yahoo! song	τ	60, 70, 80, **90**
	s	**1000**, 5000, 10000
Amazon	τ	**4**, 4.2, 4.4, 4.6
	s	**1000**, 5000, 10000

得られた 50 次元のユーザベクトルおよびアイテムベクトルのデータセットである．Yahoo! song および Amazon では，アイテムベクトルの中からランダムにサンプリングしたものをクエリとして使用した．表 1 にデータセットの統計を示す．

5.1.3 パラメータ

本実験で用いたパラメータの設定を表 2 に示す．太文字で示している値がデフォルトのパラメータである．ハッシュテーブル数 K は 10，ハッシュベクトルの長さ t は 15，およびグループ内のデータ数 n' は 50,000 とした．また，提案アルゴリズムおよびベースラインで共通するハミング距離 μ は，ランダムにサンプリングした複数のクエリについてベースラインを使用し，a を変化させることで適切な値を調べて設定した．

5.2 実験結果

5.2.1 閾値 τ の影響

τ を変化させた時の実験結果を表 3，表 4，および表 5 に示す．τ の増加に伴い，クエリとの内積が τ 以上となる条件を満たすデータ数が減少する．これにより，サンプリングプール内の条件を満たすデータ数も減少し，条件を満たすデータをサンプリングする確率が下がり，相対誤差が大きくなる．一方，τ の増加に伴い，ハミング距離の閾値 μ は小さくなる．これにより，探索するバケットの数が減少し，サンプリングプール内のデータが減少することで．条件を満たすデータをサンプリングする確率が上がり，相対誤差が小さくなる．以上より，τ の増加に伴い，相対誤差が大きくなる影響と相対誤差が小さくなる影響がある．Netflix，Amazon，および Yahoo! song（τ が 80 から 90 の間）では，τ の増加に伴い，相対誤差が増加しているため，相対誤差が大きくなる影響の方が強いと考えられる．また，Yahoo! song（τ が 60 から 80 の間）では，τ の増加に伴い，相対誤差が減少しているため，相対誤差が小さくなる影響の方が強いと考えられる．

5.2.2 厳密解アルゴリズムとの比較

厳密解アルゴリズムとの比較の実験結果を表 3，表 4，および表 5 に示す．提案手法では，厳密解アルゴリズムよりも高速にカーディナリティを推定できることが分かる．特に，表 5 の $\tau = 4$ の時，提案手法は，誤差が小さく，厳密解アルゴリズムよりも約 280 倍高速である．

5.2.3 サンプリング回数 s の影響

s を変化させた時の実験結果を表 6，表 7，および表 8 に示す．いずれのデータセットの場合も，同じサンプリング回数において，提案手法の方は同程度の時間で小さい相対誤差を達成している．これは，提案手法でノルムの小さいデータを除外したサンプリングプールの作成により，無駄なサンプリングが無くなったことが理由である．また，サンプリング回数の増加に伴い，相対誤差は下がっていることが確認できる．さらに，Amazon の相対誤差が Netflix および Yahoo! song のものと比べ，小さくなっていることが分かる．これは，サンプリングプール内に占める，閾値を満たすデータの割合が Amazon において高く，解となるデータを重点的にサンプリングできるためであると考えられる．

6 結論

本研究では，内積空間におけるカーディナリティ推定問題に取り組んだ．コーシーシュワルツの不等式により，ノルムの小さいデータを除外するとともに，内積空間におけるカーディナリティを角距離に関するカーディナリティに置き換え，サンプリングを用いた手法を利用するアルゴリズムを提案した．実データを用いた実験の結果から，提案手法の有効性を示した．

表 3: τ の影響（Netflix）

τ	4		4.1		4.2	
評価指標	相対誤差	時間	相対誤差	時間	相対誤差	時間
ベースライン	0.293	0.5	0.349	0.4	0.390	0.4
提案手法	0.170	0.5	0.193	0.5	0.227	0.4
厳密解アルゴリズム	0	23.6	0	19.9	0	15.4
τ	4.3		4.4		4.5	
評価指標	相対誤差	時間	相対誤差	時間	相対誤差	時間
ベースライン	0.478	0.4	0.549	0.4	0.637	0.4
提案手法	0.281	0.4	0.321	0.4	0.379	0.4
厳密解アルゴリズム	0	12.1	0	9.3	0	7.4

表 4: τ の影響（Yahoo! song）

τ	60		70		80		90	
評価指標	相対誤差	時間	相対誤差	時間	相対誤差	時間	相対誤差	時間
ベースライン	0.803	0.6	0.470	0.6	0.261	0.6	0.344	0.6
提案手法	0.454	0.6	0.320	0.6	0.205	0.7	0.238	0.6
厳密解アルゴリズム	0	182.3	0	177.2	0	164.3	0	139.5

表 5: τ の影響（Amazon）

τ	4		4.2		4.4		4.6	
評価指標	相対誤差	時間	相対誤差	時間	相対誤差	時間	相対誤差	時間
ベースライン	0.149	0.7	0.190	0.7	0.238	0.6	0.290	0.6
提案手法	0.084	0.8	0.103	0.7	0.130	0.7	0.152	0.7
厳密解アルゴリズム	0	229.2	0	198.7	0	165.3	0	131.2

謝辞

本研究の一部は，文部科学省科学研究費補助金・基盤研究 (A)(18H04095)，JST さきがけ（JPMJPR1931），および JSTCREST（J181401085）の支援を受けたものである．

参考文献

[1] F. Abuzaid, G. Sethi, P. Bailis, and M. Zaharia, “Simdex: Exploiting model similarity in exact matrix factorization recommendations,” CoRR, 2017.

[2] F. Abuzaid, G. Sethi, P. Bailis, and M. Zaharia, “To index or not to index: Optimizing exact maximum inner product search,” Proceedings of the IEEE International Conference on Data Engineering, pp.1250–1261, 2019.

[3] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet, “Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces,” Proceedings of the ACM Conference on Recommender Systems, pp.257–264, 2014.

[4] J.L. Bentley, “Multidimensional binary search trees used for associative searching,” Communications of the ACM, vol.18, no.9, pp.509–517, 1975.

[5] M.S. Charikar, “Similarity estimation techniques from rounding algorithms,” Proceedings of the ACM Symposium on Theory of Computing, pp.380–388, 2002.

[6] Q. Huang, G. Ma, J. Feng, Q. Fang, and A.K. Tung, “Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search,” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp.1561–1570, 2018.

[7] L. Jin, C. Li, and R. Vernica, “SEPIA: estimating selectivities of approximate string predicates in large databases,” Proceedings of the Very Large Data Bases, vol.17, no.5, pp.1213–1229, 2008.

[8] N. Koenigstein, P. Ram, and Y. Shavitt, “Efficient retrieval of recommendations in a matrix factorization framework,” Proceedings of the ACM International Conference on Information and Knowledge Management, pp.535–544, 2012.

[9] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” IEEE Computer, vol.42, no.8, pp.30–37, 2009.

表 6: s の影響（Netflix）

s	1,000		5,000		10,000	
評価指標	相対誤差	時間	相対誤差	時間	相対誤差	時間
ベースライン	0.293	0.5	0.174	2.3	0.132	4.0
提案手法	0.170	0.5	0.100	2.0	0.085	3.9

表 7: s の影響（Yahoo! song）

s	1,000		5,000		10,000	
評価指標	相対誤差	時間	相対誤差	時間	相対誤差	時間
ベースライン	0.344	0.6	0.172	2.3	0.134	4.6
提案手法	0.238	0.6	0.116	2.5	0.089	4.7

表 8: s の影響（Amazon）

s	1,000		5,000		10,000	
評価指標	相対誤差	時間	相対誤差	時間	相対誤差	時間
ベースライン	0.149	0.7	0.093	2.9	0.081	5.6
提案手法	0.084	0.8	0.059	3.2	0.055	6.0

[10] H. Lee, R.T. Ng, and K. Shim, “Extending q-grams to estimate selectivity of string matching with low edit distance,” Proceedings of the International Conference on Very Large Data Bases, pp.195–206, 2007.

[11] H. Li, T.N. Chan, M.L. Yiu, and N. Mamoulis, “Fexipro: fast and exact inner product retrieval in recommender systems,” Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.835–850, 2017.

[12] G. Linden, B. Smith, and J. York, “Amazon. com recommendations: Item-to-item collaborative filtering,” IEEE Internet Computing, vol.7, no.1, pp.76–80, 2003.

[13] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probe lsh: efficient indexing for high-dimensional similarity search,” Proceedings of the International Conference on Very Large Data Bases, pp.950–961, 2007.

[14] M. Mattig, T. Fober, C. Beilschmidt, and B. Seeger, “Kernel-based cardinality estimation on metric data,” Proceedings of International Conference on Extending Database Technology, pp.349–360, 2018.

[15] A. Mazeika, M.H. Böhlen, N. Koudas, and D. Srivastava, “Estimating the selectivity of approximate string queries,” ACM Transactions on Database Systems, vol.32, no.2, p.12, 2007.

[16] S. Morozov and A. Babenko, “Non-metric similarity graphs for maximum inner product search,” Advances in Neural Information Processing Systems, vol.31, pp.4721–4730, 2018.

[17] B. Neyshabur and N. Srebro, “On symmetric and asymmetric lshs for inner product search,” Proceedings of the International Conference on Machine Learning, pp.1926–1934, 2015.

[18] P. Ram and A.G. Gray, “Maximum inner-product search using cone trees,” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.931–939, 2012.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” Proceedings of the International Conference on World Wide Web, pp.285–295, 2001.

[20] J.B. Schafer, J. Konstan, and J. Riedl, “Recommender systems in e-commerce,” Proceedings of the ACM Conference on Electronic Commerce, pp.158–166, 1999.

[21] A. Shrivastava and P. Li, “Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips),” Advances in Neural Information Processing Systems, vol.27, pp.2321–2329, 2014.

[22] A. Shrivastava and P. Li, “Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips),” arXiv preprint arXiv:1410.5410, 2014.

[23] C. Teflioudi and R. Gemulla, “Exact and approximate maximum inner product search with lemp,” ACM Transactions on Database Systems, vol.42, no.1, pp.1–49, 2016.

[24] C. Teflioudi, R. Gemulla, and O. Mykytiuk, “Lemp: Fast retrieval of large entries in a matrix product,” Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.107–122, 2015.

[25] A. Van den Oord, S. Dieleman, and B. Schrauwen, “Deep content-based music recommendation,” Advances in Neural Information Processing Systems, vol.26, pp.2643–2651, 2013.

[26] Y. Wang, C. Xiao, J. Qin, X. Cao, Y. Sun, W. Wang, and M. Onizuka, “Monotonic cardinality estimation of similarity selection: A deep learning approach,” Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.1197–1212, 2020.

[27] X. Wu, M. Charikar, and V. Natchu, “Local density estimation in high dimensions,” Proceedings of the International Conference on Machine Learning, pp.5296–5305, 2018.