サンプリング確率の更新を利用したDiamond Samplingの再検討

仲摩　隼人† 　天方　大地††,††† 　原　隆浩††

† 大阪大学工学部電子情報工学科　〒 565–0871 大阪府吹田市山田丘 1-5
†† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1-5
††† JST さきがけ
E-mail: †{nakama.hayato,amagata.daichi,hara}@ist.osaka-u.ac.jp

あらまし　近年，パターン認識タスクの 1 つである推薦システムの需要が拡大している．推薦システムでは，非負値行列分解（NMF：Non-negative Matrix Factorization）と呼ばれる技術を利用することで評価行列の値を予測し，推薦へと応用している．評価行列の各成分はベクトルの内積と等しいため，内積の値が大きいほどユーザが商品に満足する可能性が高いといえる．本研究では，NMF により与えられた 2 つの行列から，あるクエリのベクトルとの内積を最大化するアイテムを探す，最大内積探索問題（MIPS）と呼ばれる問題に取り組む．与えられたクエリに対して全てのアイテムとの内積を計算する方法は，無駄な内積の計算が多く，非効率的である．内積計算は，機械学習においてニューラルネットワークの出力を求めるなどの用途で使用されるため，計算の高速化が非常に重要となる．この問題を解決するため，サンプリングにより計算時間の短縮を図ったアルゴリズムがこれまで提案されてきた．本研究では，サンプリングベースの代表的なアルゴリズムである Diamond Sampling を改良し，より効率的で精度の高い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示す．

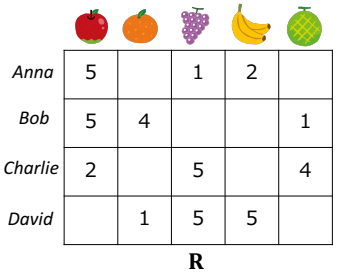
キーワード　MIPS，サンプリング，確率

1. はじめに

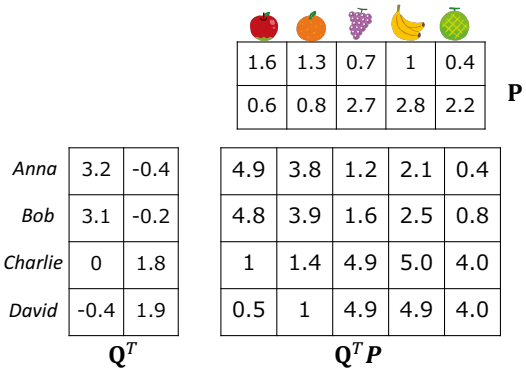
近年，Amazon(注1)，Netflix(注2)，および YouTube(注3) などの数多くのサービスにおいて，推薦システムが利用されている．推薦システムとは，膨大な情報からユーザの嗜好を予測し，商品やサービスを適切に推薦するシステムである [14]．推薦システムにおいては，主に MF（Matrix Factorization）と呼ばれる技術を利用してユーザが個々のアイテムにつけるであろう評価を予測する [13], [21]．

MF は，協調フィルタリング [3] において次元削減を実現する手法であり，推薦システムにおける最も一般的なアプローチの一つである．具体的には，ユーザの評価を有する $m \times n$（ユーザ数を m，アイテム数を n とする）の行列 R を，ユーザの特徴を表す $k \times m$ の行列 Q，アイテムの特徴を表す $k \times n$ の行列 P に分解する．これにより，ユーザの属性およびアイテムの属性がベクトルにより表現されることになる．そして，分解した行列を再度掛け合わせることにより，評価行列の値の予測が可能となり，推薦へと応用される．評価行列の各成分はベクトルの内積と等しいため，内積の値が大きいほど，ユーザが商品に満足する可能性が高いといえる．そこで，最大の内積値を求める問題である，最大内積探索問題（MIPS）と呼ばれる問題を解くことが重要となってくる [11]．

例 1. 図 1 は，4 人の各ユーザが 5 個のアイテムに与えた評価（1–5 の 5 段階，大きいほど高評価）を表す行列 R を示す．ユーザが未評価であるアイテムの評価を予測するために，行列 D を 2 つの行列（ユーザ行列 Q, アイテム行列 P）に分解する．項



(a) 評価行列 R



(b) ユーザおよびアイテムの特徴行列 Q, P および評価の予測を表す行列 $Q^T P$

図 1: 単純な MF モデルの例

(注1)：https://www.amazon.co.jp
(注2)：https://www.netflix.com
(注3)：https://www.youtube.com

目 j に対するユーザ i の評価予測は，行列積 Q^TP の (i,j) 要素，すなわち Q^T の i 番目の行の内積 $q^Tp = \sum_{i=1}^{r} q_i p_i$ によって与えられる．推薦システムの目的は，より高い予測値を持つアイテムを推薦することである．したがって，どのエントリが大きいかを判断する必要があるため，MIPS を解くことが重要となる．

実際に MF で行列分解を行うと，図 1 のように負の値が現れることがある．実世界で利用される値（画素，頻度，および評価値）の多くは非負であり，分解した結果に解釈を与えるなど，非負の性質を保ちたい場合もある．そこで，各行列の要素を全て非負の値にする行列分解の方法として，非負値行列因子分解（NMF : Non-negative Matrix Factorization）と呼ばれる方法を使う場合が多い [7].

実際の評価行列は膨大な大きさをもつため，全ての内積値を正確に計算するには多大な時間がかかる．一方でアプリケーションが求めるベクトルのペアは内積の値が大きくなる上位 k 個のペアのみであるため，それらを効率的に求める方法が必要となる．

MIPS を解決するために，局所性鋭敏ハッシュ（LSH）を利用したアルゴリズム [1], [4], [16], [20] や，木構造を利用したアルゴリズム [15], [17], [22]，類似度グラフを用いたアルゴリズム [18] など，様々なアルゴリズムが提案されてきた．その中でも有効と考えられているものは，内積値を全て計算する代わりに，確率を利用してベクトルのペアのサンプリングを行うサンプリングベースのアルゴリズムである．多くのパターン認識を用いるアプリケーションでは，正確な解は必須ではなく，代わりにサンプリングベースのアルゴリズムで得られる近似解を用いることができる [8]．これにより，値の低い成分を不必要に計算することを避けつつ，高い値を持つ成分のみを識別することが可能となる．

これまでに，文献 [8], [10] などにおいて，サンプリングベースの MIPS のアルゴリズムに関する研究が行われている．しかし，これらの研究において，精度を高く保つためには多くのサンプル数が必要となる．本稿では，より効率的に高い値を持つ成分をサンプリングすることができるアルゴリズムを設計し，必要なサンプル回数を減少させる．

課題. 行列積の top-k エントリを効率的にサンプリングするための課題は，top-k のペアのサンプル確率を高くすることである．文献 [10] の実験においては，サンプル回数が 10^4 以下の場合，精度が低くなっている．データセットのサイズが大きくなるほど，各ベクトルのペアをサンプルする確率に差がつきにくくなり，値の低いエントリをサンプリングしてしまう．そのため効率的にサンプリングを行うには，top-k の内積値をもつペアを選ぶ確率を上げる技術が求められる．

提案アルゴリズムの概要. 大きな内積値をもつペアが選ばれる確率を大きくするため，前処理として行うサンプリングの際に重みを更新していくという方法を提案する．オフライン時に Diamond Sampling を行い，ペアを選択する過程で各辺およびノードが選ばれた回数を記録しておき，一定回数サンプリングを行うごとに，選ばれた回数に基づいて重みを増加させる．更新された重みに基づき確率を再設定し，グラフを更新するという処理を繰り返すことにより，確率の差が拡大していき，精度の改善につながることが期待できる．

貢献と構成. 以下に，本研究の貢献を示す．(i) 効率的に精度の高いサンプリングが可能なアルゴリズムを提案する（4 章）．(ii) 実データを用いた実験により，提案アルゴリズムの性能を評価する（5 章）．また，2 章において既存のアルゴリズムについて説明，3 章において関連研究について議論し，6 章で本稿をまとめる．

2. 予 備 知 識

本章では，文章中の表記についての補足および問題定義を行い，その後，提案アルゴリズムで利用する既存研究のアルゴリズムを紹介する．

2.1 表記と問題定義

$[n] = 1, ..., n$ と表記する．また，$A \in \mathbb{R}^{m \times d}$ および $B \in \mathbb{R}^{n \times d}$ とする．A と B の k 番目の行はそれぞれ a_{k*} と b_{k*} で，A の i 番目の列は a_{*i}，B の j 番目の列は b_{*j} で表される．$C = A^TB$ とすると，

$$c_{ij} = \vec{a_i} \cdot \vec{b_j} = \sum_k a_{ki} b_{kj} \, \forall i \in [m], \, j \in [n].$$

となる．

問題定義. NMF により二つの行列が与えられたとき，内積の最大値を持つ top-k ペアを出力する．

2.2 既存アルゴリズム

Wedge Sampling. 文献 [8] において，Wedge Sampling と呼ばれるサンプリングアルゴリズムが提案されている．この研究では，内積に比例する確率でサンプリングを行う．まず，2 つの行列 $A(d \times m), B(d \times n)$ を 3 部グラフとして表現する．図 2 は，以下の 2 つの 3×3 行列 A および B を 3 部グラフで表現したものである．

$$A = \begin{pmatrix} 4 & 2 & 0 \\ 1 & 2 & 5 \\ 0 & 3 & 6 \end{pmatrix}$$

$$B = \begin{pmatrix} 2 & 3 & 0 \\ 5 & 3 & 2 \\ 0 & 1 & 4 \end{pmatrix}$$

A の m 列，B の n 列，および A と B の共通の d 行がそれぞれ左，右，および中央のノードに対応し，行列成分を辺の重み w とする．また，各ノードにも次のように重み W_j^i を設定する．

右ノード：$W_j^3 = 1$
中央ノード：$W_j^2 = \sum_{h=1}^{n_3} [B]_{jh}$
左ノード：$W_j^1 = \sum w(v_j^1, u) W(u)$
$u : (v_j^1, u)$ が枝をもつようなすべての中央ノード

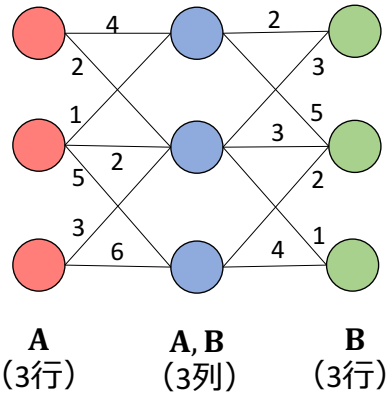


図 2: Wedge Sampling のための 3 部グラフの例

Algorithm 1: Diamond Sampling

Input: $A \in \mathbb{R}^{m \times d}$, $B \in \mathbb{R}^{n \times d}$, and s

for $\forall a_{ki} \neq 0$ **do**
  $w_{ki} \leftarrow |a_{ki}| \|a_{*i}\| \|b_{k*}\|$
$X \leftarrow$ all-zero matrix of size $m \times n$
for $l = 1, ..., s$ **do**
  Sample (k, i) with probality $w_{ki} / \|W\|$
  Sample j with probality $|b_{kj}| / \|b_{k*}\|$
  Sample k' with probality $|a_{k'i}| / \|a_{*i}\|$
  $x_{ij} \leftarrow x_{ij} + \mathrm{sgn}(a_{ki} b_{kj} a_{k'i}) b_{k'j}$
Postprocessing (see Algorithm 2)

Algorithm 2: Postprocessing

Input: $\Omega_s = (i, j) | x_{ij} > 0, t, t' \geqq t$

Extract top-t' entries of X, i.e., $|\Omega'_t| \leqq t'$ and
  $\Omega'_t \leftarrow \{(i, j) \in \Omega_s | x_{ij} \geqq x_{i'j'} \forall (i', j') \in \Omega_s \setminus \Omega'_t\}$
$C \leftarrow$all-zero matrix of size $m \times n$
for $(i, j) \in \Omega'_t$ **do**
  $c_i j \leftarrow {a_i}^T b_j$
Extract top-t entries of C, i.e., $|\Omega_t| \leqq t$ and
  $\Omega_t \leftarrow \{(i, j) \in \Omega'_t | c_{ij} \geqq c_{i'j'} \forall (i', j') \in \Omega'_t \setminus \Omega_t\}$

すなわち右ノードは 1，中央ノードは行列の各行の要素の合計，左ノードは辺とノードの重みの積の合計により重み付けされる．各辺とノードの重みによって生じる確率分布により，グラフから「wedge」と呼ばれる 2 つの経路（左ノードから中央ノードを通り，右ノードへと向かう）をサンプリングすることにより，(i, j) のペアを選ぶ．各ペアはスコアを持っており，選ばれる毎にスコアが加算されていく．s 回サンプリングを行った後，大きなスコアをもつペアを top-k として出力する．

Diamond Sampling. 文献 [10] では，Wedge Sampling を改良した Diamond Sampling と呼ばれるサンプリングアルゴリズムが提案されている．詳細はアルゴリズム 1 および 2 に示す．Wedge Sampling と同様，2 つの行列を 3 部グラフとして表現する．Diamond Sampling の目標は，グラフから繋がった 3 経路をサンプリングすることである．アルゴリズムの概要を図 3 に示す．具体的には次の 4 ステップにより行われる．

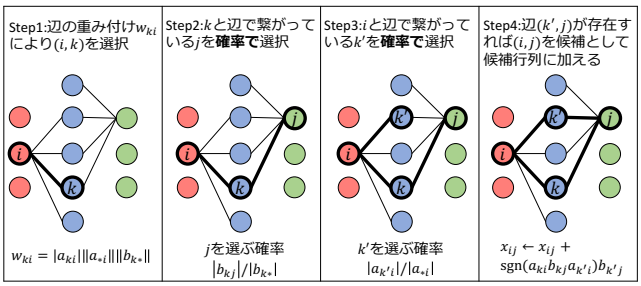


図 3: Diamond Sampling の遷移図

（Step1）辺の重み付け $w_{ki} = |a_{ki}| \|a_{*i}\| \|b_{k*}\|$ により (k, i) を選択する．

（Step2）k との辺が存在するノード j を確率 $|b_{kj}| / \|b_{k*}\|$ により選択する．

（Step3）i との辺が存在するノード k' を確率 $|a_{k'i}| / \|a_{*i}\|$ により選択する．

（Step4）以上のサンプリングの終了後，辺 (k', j) が存在するならば (i, j) を diamond として候補行列 X の要素 x_{ij} にスコアを加算する．辺 (k', j) が存在しなければ候補には加えず，次のサンプリングに移る．

このようなサンプリングを予め決められた回数だけ行い，Postprocessing（後処理）に移る．パラメータとして budget（予算）k' を設定し，候補行列の成分の top-k' を求める．そして求まった k' 個のペアの正確な内積の top-k を解として出力する．このサンプリングの結果として，各ペアを選ぶ確率は内積の 2 乗に比例することが示されている [10].

3. 関 連 研 究

本章では，サンプリングベースとは異なる MIPS に対するアルゴリズムについて説明する．

3.1 厳密解ベースのアルゴリズム

LEMP. 文献 [5], [6] では，分割統治アプローチを利用した MIPS の解決法である LEMP と呼ばれるアルゴリズムが提案されている．最初に入力ベクトルを長さでソートし，各バケットにほぼ等しい大きさのベクトルが含まれるようにバケットに分割する．その後，与えられたクエリに対して局所閾値を計算することで結果に寄与できないグループを枝刈りし，残りの各グループごとに適した検索アルゴリズムを動的に選択していく．

FEXIPRO. 文献 [12] では，3 つの枝刈り技術を使用した MIPS のためのアルゴリズム FEXIPRO が提案されている．FEXIPRO においては，特異値分解（SVD），整数近似，正数への変換という 3 つの手法を組み合わせて使用することにより高速化している．

SVD では，最初の次元が後続の次元と比較してより高い絶対値をもつように，ベクトル内の値の分布を変更する．整数近似では，2 つのベクトルのスカラーの整数部分だけを使うこと

により低速な浮動小数点計算を避ける．正数への変換では，行列内のすべてのベクトルを正の値のみを持つベクトルに変換することで，より厳密に剪定する範囲を決定することができる．

MAXIMUS．文献 [9] では，ハードウェアの効率性と検索空間の削減を活用した方法であり，LEMP および FEXIPRO と比較して高速に計算できる MAXIMUS が提案されている．MAXIMUS は以下の 3 段階のフェーズに分かれている．(1) k-means を使用してユーザのクラスタリングを行う．(2) クラスタの重心より各クラスタのアイテムのソート済みリストを作成する．(3) 各ユーザに対し対応するクラスタのリストを調べ，上位のアイテムが存在しなくなった段階で終了する．また，特定の入力に対して最適な方法をオンラインで選択することのできるオプティマイザ OPTIMUS も提案されている．OPTIMUS は MAXIMUS などのインデックスを用いる方法とインデックスを用いない BMM（ブロック行列乗算）とを比較し，より高速な方法を予測して選択する．

Cone Tree．文献 [15] では，分枝限定法によるアルゴリズム Dual-Tree および新しい木構造ベースのデータ構造 Cone Tree を提案している．Dual-Tree は複数のクエリに対して対応できるアルゴリズムであり，参照セットおよびクエリに対しそれぞれ木を作成する．Cone Tree は既存の Ball Tree [19] に似た構造であるが，分割の際に距離でなくコサイン類似度を使用し，クエリベクトルの方向に基づいてクエリのインデックスを作成する．

類似度グラフ．文献 [18] では，類似度グラフを用いた MIPS の解法を提案している．この論文では，類似度グラフ探索による最近傍探索（Greedy walk）を非メトリック類似検索問題にも拡張することによる，グラフベースの MIPS の解法が示されている．

3.2 LSH を用いたアルゴリズム

ALSH．文献 [1] では，ALSH（Asymmetric LSH）と呼ばれるアルゴリズムを提案している．これは，入力クエリベクトルとリポジトリ内のデータベクトルに非対称変換を適用することによって既存の LSH [2] を一般化したものであり，MIPS を NN（最近傍探索）に変換して次元を縮小することによる解決法である．

Simple-LSH．文献 [4] では，Simple-LSH と呼ばれるアルゴリズムを提案している．これは，パラメータを調整する必要がなく，従来の LSH アルゴリズムに比べてより単純な対称変換アルゴリズムである．Simple-LSH は，MIPS を MCS（Multilevel Coordinate Search）に縮小することにより問題を解決する．

H2-ALSH．文献 [16] では，H2-ALSH と呼ばれるアルゴリズムを提案している．n 個のデータベクトルからなるデータベース D があったとき，まず前処理フェーズとして，ノルムに従ってソートされたデータベクトルをいくつかの集合に分割し，ハッシュテーブルに格納する．次にクエリが与えられたらクエリフェーズとして，QNF（Query Normalized First）変換によりクエリ q をベクトル $Q_n(q)$ に変換して，枝刈りをしつつ最大ノルムに従って集合を順番に検索していく．

LSH の欠点．高次元データの場合，最大の内積はベクトルのノルムに比べて小さいことが多く，類似性が小さくなるため，これらのアルゴリズムでは最近傍探索のために多くのハッシュが必要となる．そのため LSH では，小さなデータセットであっても膨大なストレージを必要とする．本稿では，LSH と比較してストレージの消費が少ないサンプリングベースのアルゴリズムを提案している．

4. 提案アルゴリズム

前章で紹介した Wedge Sampling や Diamond Sampling といったアルゴリズムは，精度が低い．本章では，内積が大きいペアのみを効率的に取り出すための処理の詳細を紹介する．

4.1 方　　針

提案アルゴリズムの方針は，内積が大きいペアがより高い確率で選ばれるようにするというものである．既存手法である Wedge Sampling や Diamond Sampling では，各辺およびノードをサンプリングする際，確率の差が小さいため，値の低いエントリが選ばれる可能性が高くなる．

本稿では，Diamond Sampling を改良することで，より効率的なサンプリングを可能とするアルゴリズムを提案する．確率に関して再検討を行い，前処理として確率の再設定を行う．次節で示す処理を行うことにより，グラフにおけるペアのサンプル確率に差が生じる結果，内積の大きいベクトルのペアがより選ばれやすくなる．

4.2 アルゴリズム

提案アルゴリズムの概要を図 4 に示す．まず，Diamond Sampling と同様にサンプリングを行い，(i, j) のペアを選択する．このとき，2.2 節で示したステップそれぞれで辺 (k, i) およびノード j, k' が選ばれた回数を記録しておく．そして，ある一定の回数（ここでは p 回）サンプリングを行うごとに，選ばれた回数に基づいて重みを増加させる．具体的には，辺やノードがそれぞれ確率 p で x 回中 q 回選ばれた場合，その辺やノードの重みにあたる要素を $(1 + \alpha \cdot q \cdot p)$ 倍する．この操作により変化した重みにしたがって，グラフにおける全ての確率を設定し直す．この確率の再設定を繰り返した後に作られたグラフで，再度 Diamond Sampling を行うことにより，確率の差が大きくなり，大きな内積値を持つペアが多く選ばれるようになることで，精度の改善が期待できる．

サンプリングを行うたびに重みを更新し，確率を再設定した場合，多大な計算時間がかかり非効率的である．そのため，ある程度の回数サンプリングを行った後に，重みの更新および確率の再設定を行っている．

詳細をアルゴリズム 3 に示す．文献 [10] と同様，辺 (k, i)，ノード j, k' をサンプリングにより選択していく．その都度，選択された回数をそれぞれ配列 $c_{ki}, c_j, c_{k'}$ に記録しておく（8–13 行）．1000 回サンプリングが終わったところで，まず辺 (k, i) の重み w_{ki} を更新する（14 行）．同様に，要素 b_{kj} および $a_{k'i}$ を更新する（15–16 行）．そして更新された重みに基づき，確

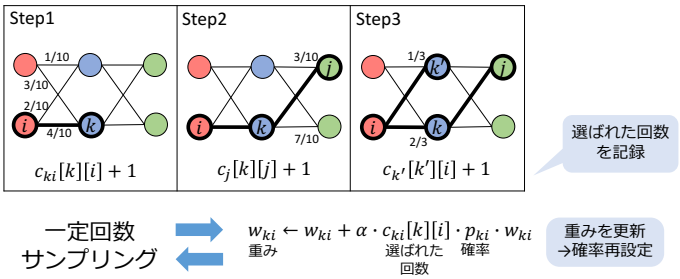


図 4: 提案アルゴリズムの概要

Algorithm 3: UpdateDiamond

Input: $A \in \mathbb{R}^{m \times d}$, $B \in \mathbb{R}^{n \times d}$, s, and r

for $\forall a_{ki} \neq 0$ **do**
  $w_{ki} \leftarrow |a_{ki}| \|a_{*i}\| \|b_{k*}\|$
$c_{ki} \leftarrow$ all-zero matrix of size $d \times m$
$c_j \leftarrow$ all-zero matrix of size $d \times n$
$c_{k'} \leftarrow$ all-zero matrix of size $m \times d$
for $l = 1, ..., r$ **do**
  for $l = 1, ..., x$ **do**
    Sample (k, i) with probality $w_{ki}/\|W\|$
    $c_{ki}[k][i] \leftarrow c_{ki}[k][i] + 1$
    Sample j with probality $|b_{kj}|/\|b_{k*}\|$
    $c_j[k][j] \leftarrow c_j[k][j] + 1$
    Sample k' with probality $|a_{k'i}|/\|a_{*i}\|$
    $c_{k'}[i][k'] \leftarrow c_{k'}[i][k'] + 1$
  $w_{ki} \leftarrow w_{ki} + \alpha \cdot c_{ki}[k][i] \cdot p_{ki} \cdot w_{ki}$
  $b_{kj} \leftarrow b_{kj} + \alpha \cdot c_j[k][j] \cdot p_j \cdot b_{kj}$
  $a_{k'i} \leftarrow a_{k'i} + \alpha \cdot c_{k'}[i][k'] \cdot p_{k'} \cdot a_{k'i}$
  reset probability

率を設定し直し，グラフを更新する（17 行）．この確率の再設定を r 回繰り返すまでを前処理として行う．（ユーザからの問い合わせ時には，更新が完了したグラフで通常の Diamond Sampling（アルゴリズム 1）を行う．）

上で紹介した前処理を繰り返し行うことにより，開始時点で高い確率で選ばれるペアと低い確率で選ばれるペアとの確率の差がさらに大きくなり，次章の実験結果でも示す通り，より精度の高く正確なサンプリングを行うことが可能となる．

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紹介する．本実験は，Windows 10 Home，2.50GHz Intel Core i5，および 8GB RAM を搭載した計算機で行い，すべてのアルゴリズムは C++で実装した．

5.1 セッティング

本実験では，提案アルゴリズムと Diamond の比較を行った．

データセット． 本実験では，データセットとして Movielens-1M[注4]および MovieTweetings[注5]を用いた．表 1 に，データセットの詳細を示す．

表 1: データセットの詳細

	Movielens	MovieTweetings
ユーザ数 / アイテム数	6,040/3,952	60,868/34,665
評価件数の合計	1,000,209	822,938

表 2: パラメータの設定

パラメータの設定	値
サンプル回数，s	5,000, **10,000**, 25,000, 50,000
k	10, 100
確率再設定回数，r	5, 10, 25, **50**, 100

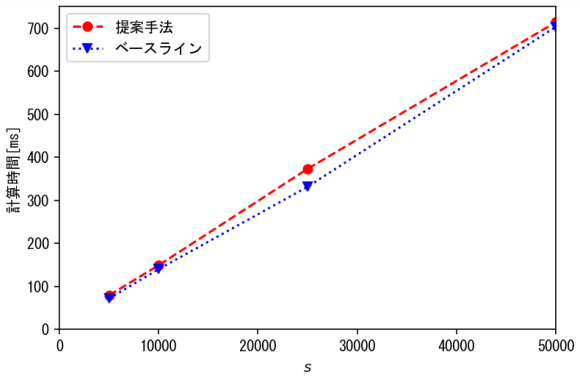


図 5: 提案手法と Diamond の計算時間の比較

パラメータ． 表 2 に，本実験で用いたパラメータを示す．太字で表されている値はデフォルトの値である．また，次元 $d = 50$ とし，予算 $t' = s$ と設定する．

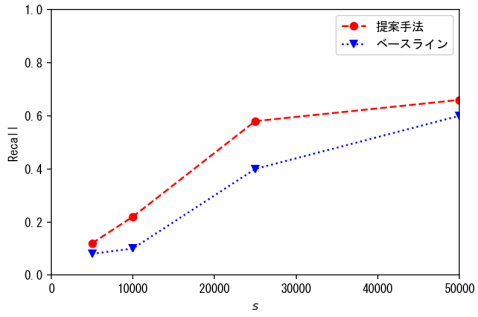
5.2 評価結果

提案手法と Diamond の計算時間の比較． 図 5 に提案手法と Diamond の平均計算時間の結果（Movielens の場合）を示す．提案手法は，前処理後は Diamond と同じサンプリングをしているため，計算時間も Diamond と同程度となった．

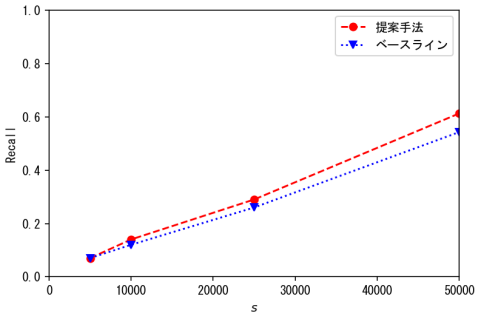
s の影響． 図 6 および 7 にサンプル回数 s を変化させたときの結果を示す．いずれのデータセットの場合も，提案アルゴリズムは Diamond と比較して，高い精度で top-k を識別できていることが分かる．サンプル回数の少ない場合においては，$k = 100$ の場合は精度はあまり変わらないが，$k = 10$ の場合は提案アルゴリズムの精度が高く，その有効性が確認できる．これは 4 章で述べた通り，確率の再設定を繰り返し行うことにより確率の差が大きくなり，大きな内積値を持つペアが多く選ばれるようになったことが理由であると考えられる．また s の増加に伴い，精度も上がっていることも確認できる．

r の影響． 図 8 に再設定回数 r を変化させたときの提案アルゴリズムの結果（$s = 10000$ の場合）を示す．いずれのデータセットの場合も，r の増加に伴い，高い精度で top-k を識別できる傾向があることが分かる．この結果は，提案アルゴリズムにおける確率の再設定の重要性を示しており，内積の大きなペアが選ばれる確率が高くなるような更新ができていることが考察できる．図 8 から，確率の再設定回数を増やしていくと，$k = 100$ の場合においては精度は 50–100 回程度で収束してい

(注4)：https://grouplens.org/datasets/movielens/1m/
(注5)：https://github.com/sidooms/MovieTweetings

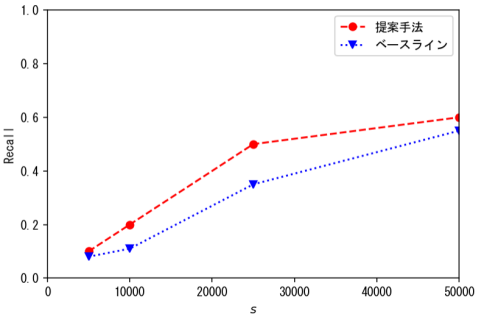


(a) $k = 10$

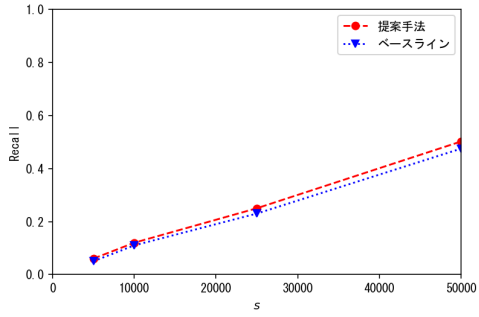


(b) $k = 100$

図 6: s の影響 (Movielens)



(a) $k = 10$



(b) $k = 100$

図 7: s の影響 (MovieTweetings)

るが，$k = 10$ の場合は 100 回以上再設定を行っても精度が高くなっていくことが予想される．

k の影響． 図 9 に k を変化させたときの結果（$s = 10000, r = 50$ の場合）を示す．提案アルゴリズム，Diamond のいずれにおいても，k の増加に伴い精度が低くなっていくことが分かる．これは，k が大きくなるにつれて，top-k ペアの内積の差が小さくなり，正確な top-k の出力が難しくなるため，精度が落ちていくと考えられる．Diamond と比較して，提案アルゴリズムはいずれのデータセットにおいても高い精度で top-k を出力できていることが分かり、確率の再設定の有効性が改めて確認できる．

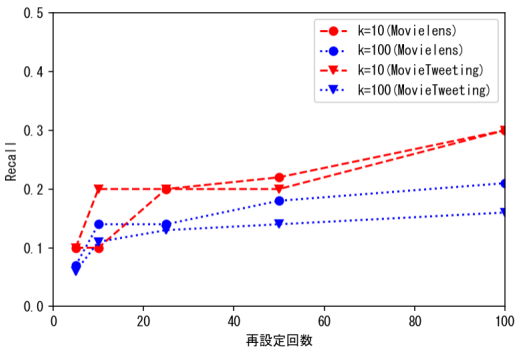


図 8: r の影響

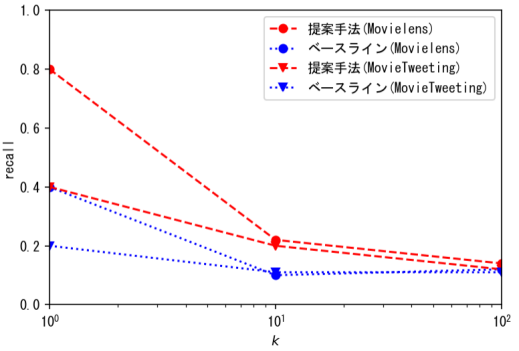


図 9: k の影響

6. ま と め

近年，ユーザの嗜好を予測して適切なアイテムを推薦する，推薦システムへの関心が高まっている．本研究では，NMF により生成される 2 つの行列において，内積値の大きなベクトルのペアをサンプリングする問題に取り組んだ．高い値をもつ成分のみを効率的にサンプリングするため，事前にサンプリングを行いつつ，サンプルされた辺とノードの重みを選ばれた回数の分だけ増加させ，確率の再設定を行うことで大きな値をもつ成分のみを効率的に取り出すことのできるアルゴリズムを提案した．また評価実験の結果から，提案手法の MIPS に対する有効性を確認した．

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） および JST さきがけ（JPMJPR1931）の支援を受けたものである．ここに記して謝意を表す．

文 献

[1] Anshumali, S., Ping, L.: Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In: NIPS. pp. 2321–2329 (2014)

[2] Aristides, G., Piotr, I., Rajeev, M.: Similarity search in high dimensions via hashing. In: Proc. Int'l Conf. on Very Large Data Bases. pp. 518–529 (1999)

[3] Badrul, M.S., George, K., Joseph, A.K., John, R.: Item-based collaborative filtering recommendation algorithms. In: WWW. pp. 285–295 (2001)

[4] Behnam, N., Nathan, S.: On symmetric and asymmetric

lshs for inner product search. In: ICML. pp. 1926–1934 (2015)
[5] Christina, T., Rainer, G.: Exact and approximate maximum inner product search with lemp. TODS 42(1), 1–49 (2017)
[6] Christina, T., Rainer, G., Olga, M.: Lemp:fast retrieval of large entries in a matrix product. In: SIGMOD. pp. 107–122 (2015)
[7] Daniel, D.L., Sebastian, S.H.: Algorithms for non-negative matrix factorization. In: Proc. Int'l Conf. on Neural Information Processing Systems. pp. 556–562 (2000)
[8] Edith, C., David, D.L.: Approximating matrix multiplication for pattern recognition tasks. In: SODA. pp. 682–691 (1997)
[9] Firas, A., Geet, S., Peter, B., Matei, Z.: To index or not to index: Optimizing exact maximum inner product search. In: ICDE. pp. 1250–1261 (2019)
[10] Grey, B., Ali, P., Tamara, G.K., Seshadhri, C.: Diamond sampling for approximate maximum all-pairs dot-product (mad) search. In: ICDM. pp. 11–20 (2015)
[11] Hsiang-Fu, Y., Cho-Jui, H., Qi, L., Inderjit, S.D.: A greedy approach for budgeted maximum inner product search. In: NIPS. pp. 5453–5462 (2017)
[12] Hui, L., Tsz, N.C., Man, L.Y., Nikos, M.: Fexipro: Fast and exact inner product retrieval in recommender systems. In: SIGMOD. pp. 835–850 (2017)
[13] Noam, K., Parikshit, R., Yuval, S.: Efficient retrieval of recommendations in a matrix factorization framework. In: CIKM. pp. 535–544 (2012)
[14] Paolo, C., Yehuda, K., Roberto, T.: Performance of recommender algorithms on top-n recommendation tasks. In: RecSys. pp. 39–46 (2010)
[15] Parikshit, R., Alexander, G.G.: Maximum inner-product search using cone trees. In: KDD. pp. 931–939 (2012)
[16] Qiang, H., Guihong, M., Jianlin, F., Qiong, F., Anthony, K.H.T.: Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In: KDD. pp. 1561–1570 (2018)
[17] Ryan, R.C., Alexander, G.G., Parikshit, R.: Fast exact max-kernel search. In: SDM. pp. 1–9 (2013)
[18] Stanislav, M., Artem, B.: Non-metric similarity graphs for maximum inner product search. In: NeurIIPS. pp. 4726–4735 (2018)
[19] Stephen, M.O.: Five Balltree Construction Algorithms. International Computer Science Institute, Berkeley, U.S.A. (1989)
[20] Thomas, D.A., Rasmus, P., Ilya, P.R., Francesco, S.: On the complexity of inner product similarity join. In: PODS. pp. 151–164 (2016)
[21] Yehuda, K., Robert, M.B., Chris, V.: Matrix factorization techniques for recommender systems. In: IEEE Computer. pp. 30–37 (2009)
[22] Yoram, B., Yehuda, F., Ran, G.B., Liran, K., Noam, K., Nir, N., Ulrich, P.: Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In: RecSys. pp. 257–264 (2014)