CD-003

近似最大内積探索問題を利用した効率的なアウトライア検出アルゴリズム

仲摩隼人 *　天方大地 *　原隆浩 *
Hayato Nakama　Daichi Amagata　Takahiro Hara

抄録

本研究では，2 つの高次元ベクトル集合および出力サイズ k が与えられたとき，内積が最も大きい k 個のベクトルペアを出力する問題に取り組む．この問題は，相関分析におけるアウトライアの検出や，出力サイズの制限付きジョイン操作といった応用がある．本問題を厳密に解く最も単純な方法は全ペアを列挙することであるが，これは非常に計算コストが大きい．本研究では，既存の state-of-the-art である ip-NSW を拡張した近接グラフを設計し，このグラフを利用して内積が大きくなりやすいベクトルペアを効率的に識別する近似アルゴリズムを提案する．実データを用いた実験により，提案アルゴリズムは高速に高精度な解を出力することを確認する．

1　はじめに

近年，Amazon[1]や YouTube[2]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 [12]．推薦システムにおいては，主に行列分解（MF：Matrix Factorization）を利用し，ユーザが個々のアイテムにつける評価を予測する [11, 17]．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．MF によってユーザおよびアイテムの属性がベクトルで表現され，評価行列の各成分はベクトルの内積と等しくなる．通常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数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．本研究では，2 つの高次元ベクトル集合が入力として与えられたとき，内積が最も大きくなるベクトルのペアを k 個検索する問題に取り組む．また，これらのペアを内積が異常に大きいものと考え，アウトライアと定義する．

*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
[1]https://www.amazon.co.jp
[2]https://www.youtube.com

この問題を厳密に解く最も単純な手法は，全ペアを列挙する方法だが，データのサイズが大きくなるにつれて計算コストが非常に大きくなる．そのため，近似計算を用いる手法が多く提案されている [1, 3, 14]．また，その中でも近接グラフを用いたアルゴリズムが最も高速・高精度であることが知られている [9]．本研究では，任意の近接グラフを用いて内積が大きいペアを効率的に探索する近似アルゴリズムを提案する．また，近似最大内積探索問題 [7] の state-of-the-art である近接グラフ ip-NSW[15] を拡張し，提案アルゴリズムの性能を向上する．

提案アルゴリズムでは，前処理としてグラフを構築するが，コサイン類似度を利用し，ノルムだけでなく角度を考慮したグラフを作成する．ip-NSW は，角度を考慮していないために，内積が大きくなる可能性の高いデータが検索の過程で候補に入らないという問題点が存在する．そこで，角度の類似度関数であるコサイン類似度を利用し，ノルムと角度の両方を考慮してグラフを作成することにより，より検索の精度が高くなることが期待できる．また，グラフ作成後には，ノルムの大きいデータと最も内積が大きくなる点を隣接点集合から検索し，解が更新されるまで幅優先探索を行う．さらに，コーシーシュワルツの不等式を利用して検索の対象範囲を狭めることにより，アウトライアを効率良く検出することが可能となる．

本研究の貢献は以下の通りである．

- 2 つの高次元ベクトル集合および出力サイズ k を入力として，ip-NSW を拡張した近接グラフにより，内積が大きくなりやすいベクトルペアを効率的に識別する近似アルゴリズムを提案する．
- 実世界の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．

以降では，2 章では予備知識を説明し，本研究の問題を定義する．3 章では，出力サイズの制限付きジョイン

操作などの関連研究を紹介する．4 章では，提案したアルゴリズムを詳細に説明する．5 章では，実データを用いて提案アルゴリズムと既存アルゴリズムとの比較実験を行い，その結果について報告する．最後に，6 章で本研究のまとめを行う．

2 予備知識

$[n] = 1, ..., n$ と表記する．入力として 2 つの d 次元ベクトル集合 $A = (a_1, ..., a_m), B = (b_1, ..., b_n)$ が与えられたとき，各ベクトルペアの内積 $\langle a_i, b_j \rangle (i \in [m], j \in [n])$ は次のように表される．

$$\langle a_i, b_j \rangle = \|a_i\| \|b_j\| \cos\beta$$

ここで β は a_i, b_j のなす角である．

問題定義．本問題は，ベクトル集合 A, B および k が与えられたとき，$A \times B$ において内積が大きい上位 k 個のベクトルペアを検索することである．

3 関連研究

3.1 Top-k MIP Join アルゴリズム

Diamond Sampling．内積の 2 乗が大きいペアの列挙手法として，文献 [7] は，Diamond Sampling と呼ばれるサンプリングアルゴリズムを提案している．まず与えられた 2 つのベクトル集合を 3 部グラフとして表現する．左ノード，右ノードがそれぞれ A, B のベクトル，中央ノードが次元数 d に対応し，行列成分を辺の重みとする．このアルゴリズムの目標は，グラフから 3 経路を重みによる確率によりサンプリングすることである．サンプリングの結果で得られた経路が繋がってサイクルを形成するなら,「Diamond」としてベクトルペアを解の候補として加える．この手法の問題点として，内積の大きいペアを高確率でサンプルできないこと，および，本論文の問題に対応するためには非負ベクトルという条件が必要になることが挙げられる．

LEMP．文献 [4, 5] では，分割統治アプローチを利用した 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ノルムでソートし，各バケットにノルムがほぼ等しい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コサイン類似度による閾値を用いて結果に含まれないバケットを枝刈りし，枝刈りされなかったバケットに対してそれぞれ適した検索アルゴリズムを動的に選択する．

3.2 MIPS のための厳密解アルゴリズム

Cone Tree．文献 [13] は，分枝限定法によるアルゴリズム Dual-Tree および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る．Cone Tree は既存の Ball Tree[16] 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，ベクトルの方向に基づいてインデックスを作成する．

FEXIPRO．文献 [8] 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，より厳密に枝刈りする範囲を決定することができる．

MAXIMUS．文献 [6] 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．

3.3 MIPS のための近似解アルゴリズム

ALSH．文献 [1] 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）に変換する解決法である．

Simple-LSH．文献 [3] 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）に変換することにより問題を解決する．

H2-ALSH．文献 [14] は，H2-ALSH と呼ばれるアルゴリズムを提案している．H2-ALSH では，データセットをノルムの大きさに応じていくつかのグループ（サブセット）に分割する．続いて，各サブセット S_i でオブジェクトを $d+1$ 次元空間にマッピングすることで，マッピング空間内においてクエリ q との内積が大きくなるオブジェクトと q との距離が短くなる．クエリが入力されると，大きいノルムを持つサブセットから近似最近傍探索を繰り返す．

ip-NSW．文献 [15] は，近接グラフを用いたアルゴリズムである ip-NSW を提案している．ip-NSW は，KNN グラフの作成・検索手法である NSW（グラフの頂点を 1 つずつ追加していき，グラフをたどっていくことで検索を行う手法）[10] を MIPS に応用したものであり，類似度関数に内積を利用し，クエリとの内積が大きくなるデータ点を探索する．ip-NSW の問題点として，角度を考慮していないということが挙げられる．例えば図 1 において，あるベクトル x と y がグラフ上の隣接点であり，y と z も隣接点であるとする．x がクエリで，グラフで y を探索していると仮定すると，x は z の MIP であると判断されやすくなるが，x, z 間の角度 $\psi° + \omega°$ が大きい場合．実際 x, z 間の内積は小さい可能性があり，角度を考慮する必要があるといえる．

ip-NSW+．文献 [9] は，ip-NSW の問題点を改良したアルゴリズムを提案している．ip-NSW+は，角度の類似度関数であるコサイン類似度を利用し，内積に関するグラフ G_s の他に角度に関するグラフ A_s を作成し，2 つのグラフを用いることにより検索を行う．この手法の問題点として，角度の影響が大きく，ノルムが大きくクエリとの内積も大きい解が候補から除外される可能性が挙げられる．またグラフを 2 つ作成する必要があるため，前処理に時間がかかるという課題もある．

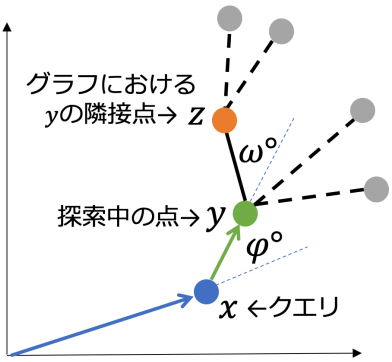


図 1: ip-NSW の問題点：$\psi° + \omega°$ が大きい $\rightarrow \langle x, z \rangle$ は小さい可能性が高い

4 提案アルゴリズム

4.1 アイデア

本問題を厳密に解くアルゴリズムは $O(nm)$ 時間が必要なため，本研究では近似アルゴリズムを考える．具体的には，近接グラフを用いた MIPS の近似解アプローチである ip-NSW および ip-NSW+を応用し，内積が大きくなりやすいベクトルペアを効率的に識別するアルゴリズムを考える．角度の類似度関数であるコサイン類似度を利用することにより，3 章で示したようなノルムは大きいが内積は小さいベクトルを解から除外することができる．提案アルゴリズムはグラフ構築の際にコサイン類似度を利用し，ノルムと角度の両方を考慮したグラフをオフラインで構築する．ip-NSW+では 2 つ必要だったグラフを 1 つにまとめることにより，計算時間の高速化が期待できる．またオンラインで内積が大きいペアを検索するフェーズにおいては，コーシーシュワルツの不等式や幅優先探索を用いて，高速に内積の大きい（近似）上位 k 個のベクトルペアを列挙する．

4.2 アルゴリズム

提案アルゴリズムはグラフ構築（オフラインフェーズ）と検索フェーズ（オンラインフェーズ）に分かれている．まず，グラフ構築の手順は以下の通りである．

1. ベクトル集合 A, B 内のベクトルをノルムによりソートし，ノルムの大きな順にそれぞれ $a_1, a_2, ..., b_1, b_2, ...$ とおく．次に，A に対し以下の操作をソート順に実行する．

2. ベクトル $a \in A$ に対し，その時点でのグラフ G 中で a との内積が大きい（近似）上位 rM 個の頂点を Greedy Walk[15] により見つける．

3. 2 で見つかった頂点 q のうち，a に対するコサイン類似度 $\beta = \frac{\langle a,q \rangle}{\|a\|\|q\|}$ の大きな順に上位 M 個の頂点を見つける．

4. a を 3 で得られた各頂点に接続する．

上の操作により，ノルムと角度の両方を考慮した精度の高いグラフを構築できる．

また，検索フェーズの手順は以下の通りである．

1. ベクトル $b_1, ..., b_k \in B$ をクエリとし，A においてこれらのベクトルとの内積が大きい上位 k 個の頂点を探索する．具体的には，グラフ G の a_1 から幅優先探索の要領で，現在アクセスしている頂点（ベクトル）との内積が閾値（暫定の上位 k 番目の内積）を超える場合に，その頂点の隣接頂点を展開する．また，この操作によって得られた暫定の閾値を u_k とする．

2. ベクトル $b_{k+1}, b_{k+2}, ..., b_j, ...$ に対し，コーシーシュワルツの不等式 $\langle b_j, a_1 \rangle \leq \|b_j\|\|a_1\|$ により，$\|b_j\|\|a_1\| \leq u_k$ であれば b_j は top-k の内積を持ち得ないため，その時点で検索を終了する．そうでなければ，解が更新されるまで 1 と同様の方法でグラフ探索を行う．

以上の操作により，内積が大きくなりやすいベクトルペアを効率的に識別することが可能となる．

5 評価実験

本章では，提案アルゴリズムの性能を評価するための実験とその結果を示す．

5.1 設定

本実験では以下のアルゴリズムの評価を行った．

- ip-NSW [15] を用いて，4.2 節の提案アルゴリズムを用いた手法.
- 4 章で提案したグラフを用いて，Naive k-MIPS を行った手法（PGN と表記）.
- ip-NSW+ [9].

表 1: データセット

データセット	アイテム数 (A)	ユーザ数 (B)	d
Yahoo! Music	$136,736$	$100,900$	300
WordVector	$1,000,000$	$10,000$	300
ImageNet	$2,340,373$	$80,000$	150
deep1M	$1,000,000$	$10,000$	256

表 2: パラメータ

パラメータ	値
r	$\mathbf{1.25}, 1.5, 2, 2.5, 3$
M	50
k	$1000, \mathbf{5000}, 10000$

- LEMP [5]（厳密解アルゴリズム）.
- 本研究における提案アルゴリズム (Proposed Algorithm：PA と表記).

全ての手法は C++で実装されており，g++ 7.4.0 の-O3 最適化オプションを付けてコンパイルされている．すべての実験を Windows10 が搭載された 2.50GHz Intel Core i9 および 8GB RAM で構成される計算機上で行った．

5.1.1 評価指標

本実験では，ユーザから入力データとして k が与えられてから解を特定するまでの時間および Recall（精度）を評価した．

5.1.2 データセット

実験で用いるデータは Yahoo! Music, WordVector, ImageNet，および deep1M[3]である．表 1 にデータセットの詳細を示す．

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．それぞれのパラメータにおける太字は本実験におけるデフォルト値である．

5.2 実験結果

5.2.1 r の影響

r を変化させたときの計算時間および Recall は図 2 の結果になった．r が小さいときは計算時間が短く，また

[3]http://www.cse.cuhk.edu.hk/systems/hash/gqr/datasets.html

表 3: PGN の結果

データセット	計算時間 (s)	Recall
Yahoo! Music	64.58	0.809
WordVector	380.63	0.731
ImageNet	617.25	0.716
deep1M	365.20	0.669

表 4: LEMP の結果

データセット	計算時間 (s)
Yahoo! Music	41.25
WordVector	62.50
ImageNet	208.75
deep1M	68.70

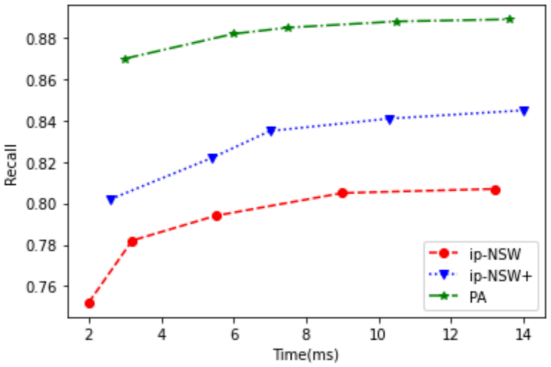
r が大きいときは Recall が高くなった．PA，ip-NSW および ip-NSW+を比較した結果，どのデータセットにおいても PA が高い Recall を達成しており，提案したグラフの有効性が示された．また，PGN の結果は表 3 のようになった．PGN はコーシーシュワルツの不等式を使用していないため，他の手法と比較して計算時間が長い．

5.2.2 出力サイズ k の影響

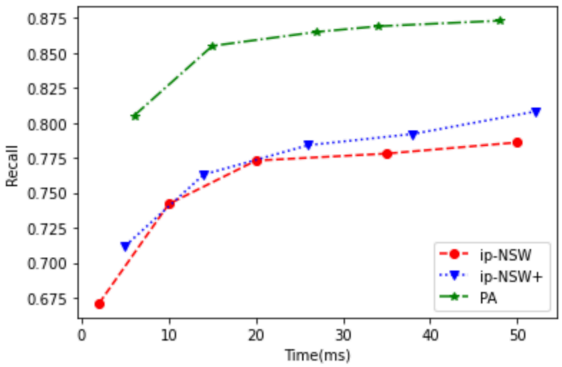
出力サイズ k を 1000 から 10000 まで変化させたときの Recall は図 3 の結果になった．いずれの手法においても，k の増加に伴い精度が低くなっていくことが分かる．これは，k が大きくなるにつれて，top-k ペアの内積の差が小さくなり，正確な top-k の出力が難しくなるため，精度が落ちていくと考えられる．また LEMP の計算時間は表 4 のようになり，提案アルゴリズムが LEMP と比較して大幅に高速化できていることが分かる．

6 結論

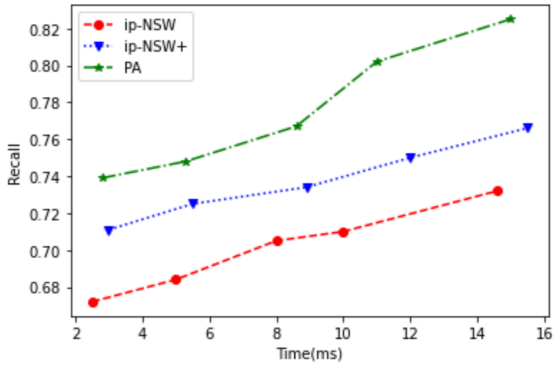
本研究では，2 つのベクトル集合から内積が最も大きい k 個のベクトルペアを高精度かつ効率的に検索できるアルゴリズムを提案した．提案アルゴリズムはコサイン類似度やコーシーシュワルツの不等式を用いることにより，効率よく内積が大きい k 個のベクトルペアを列挙することができる．実データを用いた評価実験により，提案アルゴリズムが比較手法に比べて高速かつ高精度に解を列挙することを確認した．



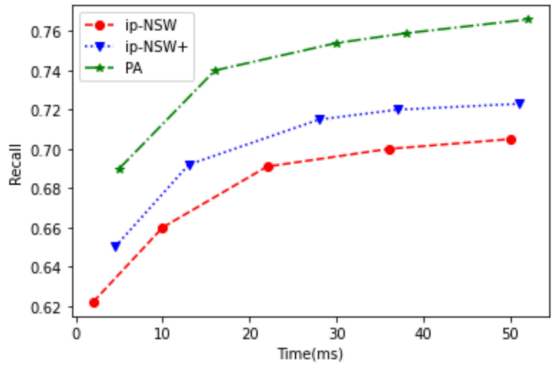
(a) Yahoo! Music



(b) WordVector

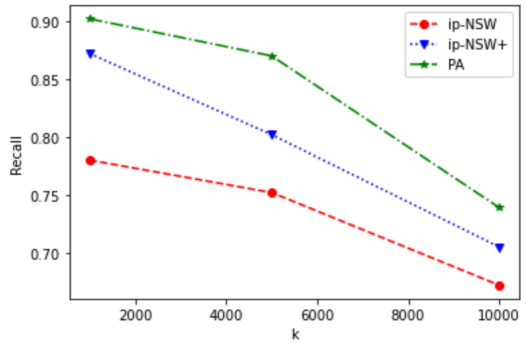


(c) ImageNet

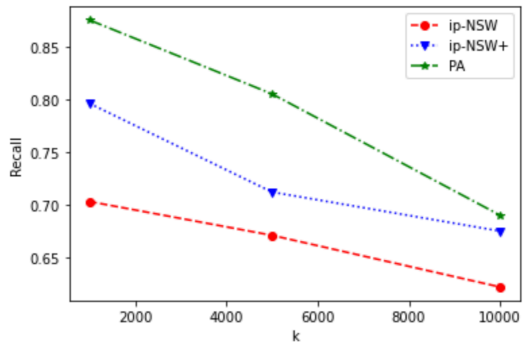


(d) deep1M

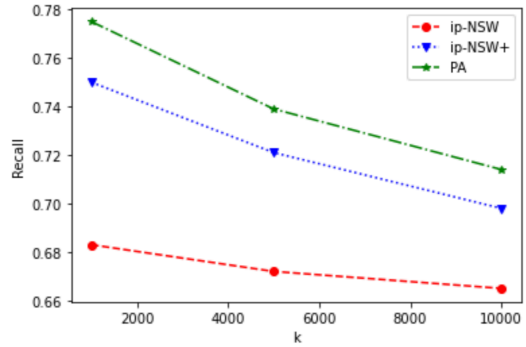
図 2: r の影響



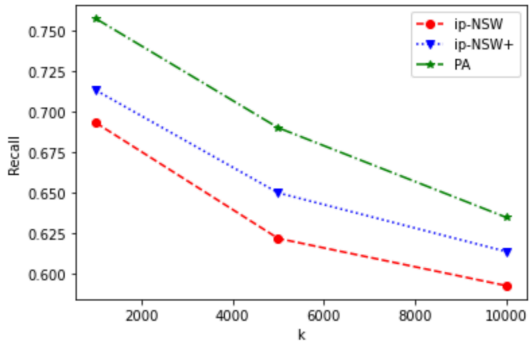
(a) Yahoo! Music



(b) WordVector



(c) ImageNet



(d) deep1M

図 3: k の影響

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] S. Anshumali and L. Ping, “Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips),” Neural Information Processing Systems, pp.2321–2329, 2014.

[2] G. Aristides, I. Piotr, and M. Rajeev, “Similarity search in high dimensions via hashing,” Proc. Int’l Conf. on Very Large Data Bases, pp.518–529, 1999.

[3] N. Behnam and S. Nathan, “On symmetric and asymmetric lshs for inner product search,” International Conference on Machine Learning, pp.1926–1934, 2015.

[4] T. Christina and G. Rainer, “Exact and approximate maximum inner product search with lemp,” ACM Transactions on Database Systems, vol.42, no.1, pp.1–49, 2017.

[5] T. Christina, G. Rainer, and M. Olga, “Lemp:fast retrieval of large entries in a matrix product,” Special Interest Group on Management of Data, pp.107–122, 2015.

[6] A. Firas, S. Geet, B. Peter, and Z. Matei, “To index or not to index: Optimizing exact maximum inner product search,” IEEE International Conference on Data Engineering, pp.1250–1261, 2019.

[7] B. Grey, G.K. Tamara, and P. Ali, “Diamond sampling for approximate maximum all-pairs dot-product (mad) search,” IEEE International Conference on Data Mining, pp.11–20, 2015.

[8] L. Hui, N.C. Tsz, L.Y. Man, and M. Nikos, “Fexipro: Fast and exact inner product retrieval in recommender systems,” Special Interest Group on Management of Data, pp.835–850, 2017.

[9] L. Jie, Y. Xiao, D. Xinyan, L. Zhirong, C. James, and Y. Ming-Chang, “Understanding and improving proximity graph based maximum inner product search,” AAAI Conference on Artificial Intelligence, pp.139–146, 2020.

[10] Y.A. Malkov and D.A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.824–836, 2018.

[11] K. Noam, R. Parikshit, and S. Yuval, “Efficient retrieval of recommendations in a matrix factorization framework,” ACM Conference on Information and Knowledge Management, pp.535–544, 2012.

[12] C. Paolo, K. Yehuda, and T. Roberto, “Performance of recommender algorithms on top-n recommendation tasks,” The ACM Recommender Systems conference, pp.39–46, 2010.

[13] R. Parikshit and G.G. Alexander, “Maximum inner-product search using cone trees,” Knowledge Discovery and Data Mining, pp.931–939, 2012.

[14] H. Qiang, M. Guihong, F. Jianlin, F. Qiong, and K.H.T. Anthony, “Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search,” Knowledge Discovery and Data Mining, pp.1561–1570, 2018.

[15] M. Stanislav and A. Babenko, “Non-metric similarity graphs for maximum inner product search,” Neural Information Processing Systems, pp.4726–4735, 2018.

[16] M.O. Stephen, Five Balltree Construction Algorithms, International Computer Science Institute, Berkeley, U.S.A., 1989.

[17] K. Yehuda, M.B. Robert, and V. Chris, “Matrix factorization techniques for recommender systems,” IEEE Computer, pp.30–37, 2009.