機械学習によるグラフベース近似最近傍探索の高速化

菅　　寧† 陸　　可鏡† 杉浦　健人† 石川　佳治†

† 東海国立大学機構名古屋大学大学院情報学研究科　〒 464–8603 愛知県名古屋市千種区不老町

Email: {jian, lu}@db.is.i.nagoya-u.ac.jp, {sugiura, ishikawa}@i.nagoya-u.ac.jp

あらまし　今画像分類や推薦システムなど，様々な分野で近似最近傍探索（approximate nearest neighbor，ANN）の手法がよく活用されている．グラフベース ANN 技術では，データセットをあらかじめグラフとして構築し，グラフ上の任意の点から貪欲に問合せ点にアプローチする．代表的な手法である HNSW（Hierarchical Navigable Small World）では，スキップリストのアイデアを参考に多層グラフを構築し，探索を高速化する．HNSW は全ての問合せ点に対して同じ数の検索エッジを使用するが，一部の検索しやすいノードに対して検索エッジの数を減らしても結果には影響がない．つまり，一部の問合せ点では，冗長なエッジを検索してしまう問題が発生する．本稿ではこの問題を解決するために，機械学習の手法を用いて，各問合せ点に応じてグラフ上の最適な検索エッジ数を予測する．また，様々なデータセットで実験により HNSW 手法と比較することで，提案手法の高速化の効果を検証する．

キーワード　近似最近傍探索，高次元データ，近傍グラフ．

1 はじめに

最近傍探索は画像検索，推薦システムなどの幅広い分野で用いられる基礎的な情報処理技術である．この技術は与えられた問合せデータに対して，データベースから問合せデータに最も近いデータ（最近傍点）を見つける．しかし，データの次元が大きくなると「次元の呪い」という問題を引き起こすのに加え，検索精度より検索速度が重要であるという要求が増えている．そのため，検索時間を大幅に削減できる**近似最近傍探索**の研究が盛んに行われている．

近似最近傍探索の一種であるグラフベースの手法は検索時間が短くかつ検索精度も高いため，現在広く研究されている．グラフベースの探索の基本的なアイデアは，データセットから近似近傍グラフを生成することである．問合せ点の最近傍点を検索する際には近傍グラフ上の任意のノードを開始点とし，そのノードの近傍点，つまりエッジの張られている各ノードと問合せ点との距離を計算する．最も問合せ点に近いノードへ移動しこの処理を繰り返すことで，問合せ点の最近傍点となるノードを探す．現在のグラフベース近似最近傍探索の主流手法は hierarchical navigable small world（HNSW）グラフ [1] や，navigating spreading-out graphs（NSG）[2],navigating satellite system graphs(NSSG)） [3] などを用いた手法である．また，近年では機械学習によるグラフベース近似最近傍探索の高速化手法も提案されている [4,5].

本研究ではグラフベースの近似最近傍探索を高速化するために，最先端の手法である HNSW グラフの改善点について検討する．特に，ユーザに要求された検索精度に対する機械学習によるより高速な近似最近傍探索の実現を図る．

本稿の構成は以下のとおりである．2 章では，グラフベースの近似最近傍探索および今の改善手法に関する先行研究について概説する．次に，3 章では先行研究の問題点を分析し，本研究の研究動機を説明する．また，4 章では提案手法の設計および具体的な手順についてそれぞれ説明する．そして，5 章では提案手法の有効性に関する評価実験について述べる．最後に，本研究のまとめと今後の課題を 6 章で述べる．

2 先行研究

2.1 NSW

データセットのデータをノードとし，各ノードを自身の k 近傍点を結んだグラフを k 近傍グラフという．しかし，あるデータの近傍点を検索するために，データが全てのデータとの距離を計算する必要がある．そのため，大規模なデータセットから k 近傍グラフを構築するには，膨大な時間が必要である．グラフの構築時間を短縮するために，近似的に k 近傍グラフを構成できる方法が提案された．このような方法で構築されるグラフの各ノードに接続されたノードは，必ずしも真の k 近傍点であるとは限らないので，近似 k 近傍グラフと呼ばれる．

近似近傍グラフ上での検索速度を向上させるために，Malkov らは navigable small world（NSW）グラフを提案した [6]．NSW はノードを 1 つずつグラフに追加し，ビームサーチによって現在のグラフ上で距離の近い k 個のノードへエッジを張り近似近傍グラフを生成する．つまり，後半に追加されるノードは近傍点へのエッジしか持たないが，早期に追加されたノードは遠くのノードへのエッジを持つ可能性がある．NSW ではそのような長いエッジを問合せ点への近道として利用できるため，検索速度が大幅に向上する．

しかし，このような構成方法は問題点がある．NSW では最初に追加された点だけがこれらの近道を持ち，後半に追加された点はその点に非常に近い点とのみ接続される．そのため，検索の早期段階（問合せ点への距離が大きい段階）に後半に追加されたノードを多数経由してしまった場合，細かな移動が重なり検索が悪化してしまう．

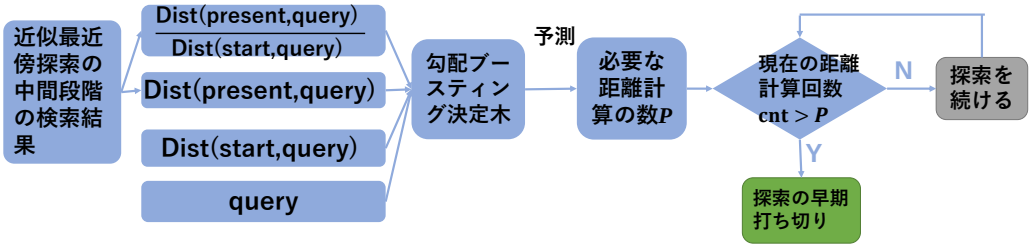

図 1　早期終了による HNSW の高速化手法の流れ

2.2　HNSW

近道となるエッジをより効果的に構築するために，Malkov らは NSW に階層構造を導入した HNSW を提案した．HNSW はスキップリストのアイデアを参考にしており，1 つ下位の層に存在するノードが現在の層にも存在する確率が p となるようにグラフを構築する．つまり，最下層を第 0 層とし全ノード数を N とする場合，第 i 層にあるノードの数はおよそ p^iN であり上層になるほどノード数が減少する．したがって，図 2 に示すように上位層のグラフを検索のための近道として利用できる．検索時には最上層から順に問合せ点の最近傍点を見つけ，ある層の最近傍点を次の層の開始点として繰り返すことで，最下層における問合せ点の最近傍を検索する．

こうして構築されたインデックスは，上層のノードが疎になり，その間のエッジは全ての近道と見なすことができる．上層のグラフは問合せ点のおおよその位置を推定できる．最下層のグラフ，すなわち全てのノードを含むグラフから，問合せ点の最近傍点を可能な限り正確に見つける．

HNSW では以下の 3 つのパラメータが使用される．

（1） M：各ノードの近傍点の最大数．

（2） $efConstruction$：グラフ作成のためのビームサイズ．

（3） $efSearch$：検索のためのビームサイズ．

既存研究では $efConstruction$ と M の増加に伴いグラフの作成時間とメモリの使用量も増加するが，検索精度は高くなると報告されている．

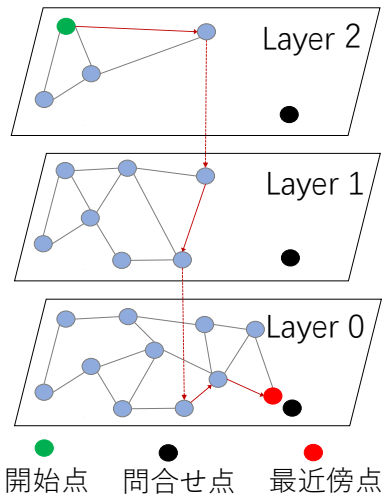

図 2　HNSW グラフ構造

2.3　早期終了による HNSW の高速化

HNSW を高速化するために，Li らは機械学習によって各問合せ点の終了条件を予測し，検索を早期に終了させる手法を提案した [4]．Li らは実験により，HNSW において 80 %の問合せ点はより小さな $efSearch$ によって最近傍点が見つけられることを示した．しかし，HNSW では全ての問合せ点に同じパラメータ $efSearch$ を使用する．つまり，一部の検索が難しい問合せ点の最近傍を見つけるためにより大きな $efSearch$ を設定する必要があり，その他多数の問合せ点に対して無駄なノードの検索が多く発生する．

この問題を解決するために，Li らは機械学習モデルを用いた各問合せ点の終了条件の予測を提案した．Li らの手法の処理手順を図 1 に示す．HNSW における検索のとき最も時間がかかるのは，複数のノードと問合せ点の距離計算である．そのため，距離計算の回数を検索時間として見なすことができる．また，同じ $efSearch$ であっても，問合せ点によって必要な距離計算の回数が異なる．したっがて，Li らは距離計算の最大回数を終了条件としている．元の HNSW では大きな $efSearch$ を用いるため，検索が容易な問合せ点において仮に全ての最近傍点が見つかった後でも，より近い結果がないことを確認するために多くのノードが検索される．一方，Li らの手法では各問合せ点の距離計算の最大回数を予測し検索処理を早期に終了させ無駄な検索を省く．

特徴量としては，開始点から問合せ点までの距離，問合せ点のベクトル，一定回数検索した後の結果から問合せ点までの距離などのデータが用いられる．以上の特徴量を使って距離計算の最大数を終了条件として予測し，距離計算の数が予測値より大きくなったら探索を終了し，現在の結果を返す．

3　研 究 動 機

本章では，まず先行研究の問題点を分析し，提案手法のアイデアを紹介する．次に，予備実験を行い，実験結果により提案手法の実現可能性を検証する．また，適切な検索エッジ数を予測するために，ユーザの精度要件を考慮する必要があると判断する．

3.1　先行研究の問題点

Li の方法は高い検索精度が要求される場合にのみ有効であり，検索精度が求められない場合は早期終了が上手く働かない．そして，Li の実験結果により再現率が 0.95 に達しない場合，Li

の手法では検索時間がほとんど短縮されない．また，検索を早期に終了しても，検索中の不要な検索を回避できないという問題点もある．

一方，HNSW はビームサーチでの検索時にノード全ての近傍点を走査する．問合せ点が開始点からまだ遠い場合，問合せ点の大まかな方向を推測するために各ノードの近傍点の一部だけを検索して，問合せ点に高速にアプローチすることが可能である．これらの問合せ点に対して，検索エッジの数を近傍点の最大数 M に設定することは明らかに不適切である．問合せ点が開始点に近い場合，開始点の近傍点はすでに問合せ点の最近傍である可能性が高い．そのため，開始点からより多くの近傍点を検索することで，素早く結果を得ることができる．以上の分析により，全ての問合せ点に対して同じ検索エッジ数で検索すると，HNSW の検索性能が制約されることが分かる．

本研究では HNSW と Li の方法の問題点を解決するために，終了条件の予測ではなく**各問合せ点における最適な検索エッジ数**の予測に着目する．つまり，問合せに応じて異なる検索エッジ数を設定することで，任意の検索精度で HNSW の高速化の実現を図る．

3.2 検索エッジ数の分析

このアイデアの実現可能性を検証するため，まず検索エッジ数が検索性能に与える影響を分析する予備実験を行った．HNSW は検索時にノード全ての近傍点を走査するため，各ノードの近傍点の最大数 M がそのまま検索エッジ数となる．そのため，ノードの最大近傍点の数が異なるグラフを構成し，その上で検索を行うことで検索エッジ数の異なる検索をシミュレートすることが可能である．

本研究は実データである GIST1M [7]，GLOVE1.2M, GLOVE2.2M [8] と正規分布に従う人工データ GAUSS を用いて実験を実施した．なお，各データセットの情報は後述する実験の章の表 1 にて詳細を紹介する．

各データセットに対して，$M \in \{16, 32, 64, 96, 128\}$ を用いて HNSW グラフを構築した．構築した各グラフに対して近似最近傍探索を実施し，異なる $efSearch$ に応じた検索時間と再現率を計測した．異なる検索エッジ数で得られた結果を図 3，図 4 に示す．図の横軸は問合せ点あたりの平均検索時間，縦軸がその平均検索精度を表す．

図 3 に示した GLOVE1.2M の実験結果により，検索精度が 0.95 以下の場合，1 ノードあたり 16 本のエッジのみを検索することで最良の結果が得られることが分かる．一方，要求精度が 0.95 より大きい場合，ノードあたりの検索エッジ数が 32 と 64 の方が良い結果となる．GLOVE2.2M の結果は GLOVE1.2M の結果と似ているが，検索エッジ数が 16 の場合の結果が他の結果よりはるかに優れていることが分かる．

図 4 の結果により GAUSS データセットでは，検索精度 0.90 以下の場合，検索するエッジの最適数は 64 個である．要求する検索精度が高くなるにつれて，検索エッジ数 96 の結果が最速で対応する精度に到達するようになる．また，検索エッジ数が 16 と 32 の結果は，他の検索エッジ数の結果に比べて一貫して劣っている．GIST1M データセットについては全ての検索エッジ数で得られた結果が近いが，検索エッジ数 32 と 64 の結果が常に他の検索エッジ数の結果を上回った．

以上の実験から次の結論が得られる．

- 適切な検索エッジ数を選択することで，HNSW の検索速度を効果的に向上させる．同じ問合せデータセットでも，検索エッジ数の違いによって得られる結果は大きく異なる．
- 最適な検索エッジ数は要求される検索精度によって異なる．低い検索精度でも十分な場合，ノードあたり少数の近傍点のみを検索することで，より高速な検索を実現できる．一方，高い検索精度が要求される場合は十分な数の近傍点を検索しなければ要求を満たせない．したがって，最適なエッジ数を予測する際には，ユーザに要求された検索精度を考慮する必要がある．
- 最適なエッジ数の選択はデータセットの分布に依存し，**単純にデータセットのサイズとデータの次元から最適なエッジ数を選択することはできない**．GIST1M データセットと GAUSS データセットの次元はそれぞれ 960 と 128 であり，データサイズは同じである．しかし，検索精度の設定が低い場合，GAUSS の最適なエッジ数は GIST1M の最適なエッジ数よりも大きい．一方，他の二つのデータセットは GAUSS よりもデータサイズと次元が大きいが，最適なエッジ数は GAUSS よりも小さい．そのため，**本研究では機械学習を用いて各問合せ点に対する最適な検索エッジ数を予測する**．

4 機械学習による検索エッジ数の予測

この章では提案手法のアイデアおよび具体的な処理手順について述べる．

4.1 ラベルの推測

3 章の実験により検索エッジ数は検索精度と検索時間についてトレードオフの関係を持つため，精度と時間の両方において支配的な検索エッジ数は見つけられない．そのため，本研究ではまずユーザが要求する検索精度を閾値として設定すし，閾値に最初に到達した検索エッジ数を学習データセットのラベルとして使用する．つまり，検索精度をパラメータとし，精度を満たす範囲で処理時間を最小化できるエッジ数を機械学習における正解ラベルとする．学習データセットにおけるラベル生成のアルゴリズムを図 5 に示す．閾値に最初に到達した検索エッジ数を学習データセットのラベルとして使用する．

まずノードの近傍点の数がラベルの最大値となるグラフを作成する．次に，学習データセット中の各問合せ点に対して，検索エッジ数を変えてグラフで検索し，検索時間と検索精度を記録する．検索精度が閾値に達した場合，このエッジ数をラベルとする．検索精度が閾値に達しない場合，より大きな検索エッジ数で検索する．検索エッジ数をノードの出次数まで増加させても検索精度が閾値に達しない場合，最も検索精度の高い検索エッジ数をラベルとする．ただし，学習データセットが大きい場合，各問合せ点の最適なエッジ数を正確に計算するには時間がかかりすぎる．そのため，各問合せ点に対して最小値から最

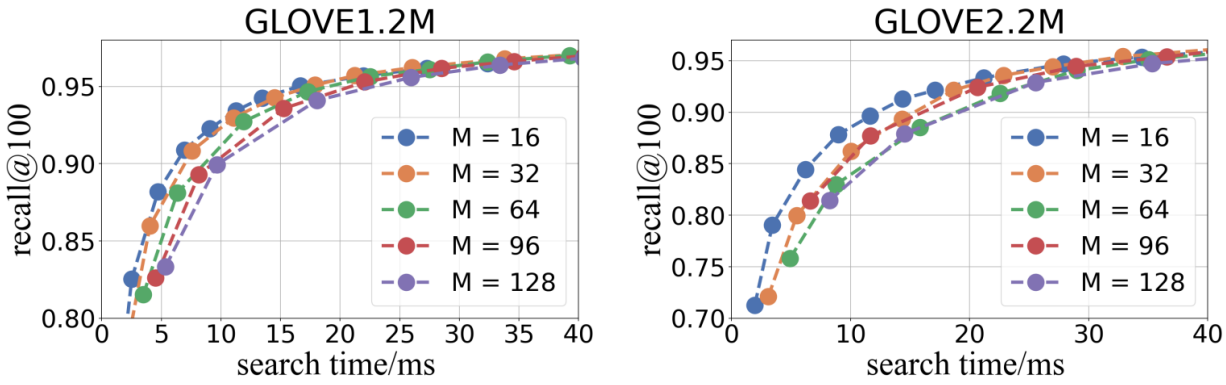

図 3　GLOVE1.2M と GLOVE2.2M における検索結果

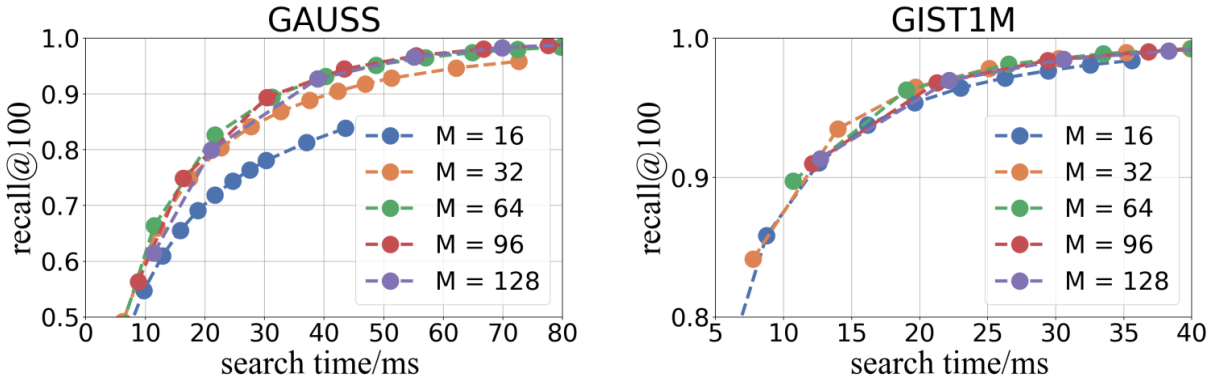

図 4　GAUSS と GIST1M における検索結果

Input: 検索精度の閾値 R，問合せ点セット Q，問合せ点の近傍点の集合 G，検索エッジ数の最大値 M，ラベルの間隔 t
Output: 最適な検索エッジ数

```
best_recall := 0, best_edges := 0
m := t
for each q ∈ Q do
    while m <= M do
        results := SearchbyHNSW(q, m, efSeach);
        temp_recall := Calculate_Recall(results, G);
        if temp_recall > R then
            best_edges := m
            break
        if temp_recall > best_recall then
            best_edges := m
            best_recall := temp_recall
        m := m + t
return best_edges
```

図 5　学習データのラベルの生成

大値までの等間隔で数値を取り，検索エッジ数の候補とする．

4.2　特徴量の選択

本研究では，以下の 2 つの要因が，問合せ点に対する最適な検索エッジの数に影響を与えると考える．

1 つ目は，開始点から問合せ点までの距離である．開始点が問合せ点からまだ遠い場合，問合せ点への大まかな方向さえ正しければ距離を縮められる．つまり，確認するエッジ数が少なくても妥当な次の検索ノードが見つけられると考えられる．一方，検索が進み問合せ点に近づくにつれて真の最近傍点を見つけるには微調整が必要となり，確認するエッジ数も増加すると考えられる．

2 つ目は開始点と問合せ点周辺の点密度である．ある点の周囲の点の密度が高い場合，それらは同じ近傍点を多数共有している可能性が高い．したがって，検索エッジ数を減らし周囲の点の一部を検索するだけで次の最近傍点を発見できると考える．

問合せ点や開始点周辺の点の密度を計算することは困難であるため，開始点と問合せ点のベクトルを直接特徴量として利用する．ある問合せ点の最適なエッジ数が小さい場合，その周辺の他の問合せ点の最適なエッジ数も小さくなると考える．そのため，問合せ点の位置を最適なエッジ数を決定するための特徴量として利用する．また，開始点と問合せ点のベクトルには既に両者の距離に関する情報が暗黙に含まれるため，両者の距離をあらためて特徴量としては扱わない．したがって，本研究では問合せ点と開始点のベクトルを特徴量として利用する．

4.3　予測モデルの作成

本研究では，勾配ブースティング決定木を予測モデルとして利用する．勾配ブースティング決定木はアンサンブル学習を参考にして，複数の決定木を組み合わせることで予測を行う．まず，最適なエッジ数を予測するために単一の決定木を構築し，予測誤差を計算する．そして，最初の決定木の予測誤差を予測するために，もう一つの決定木を生成する．つまり，前の決定木の予測値の誤差を，新しい決定木が引継いで小さくしていく．学習ラウンドごとに新しい決定木が追加される．最後は全

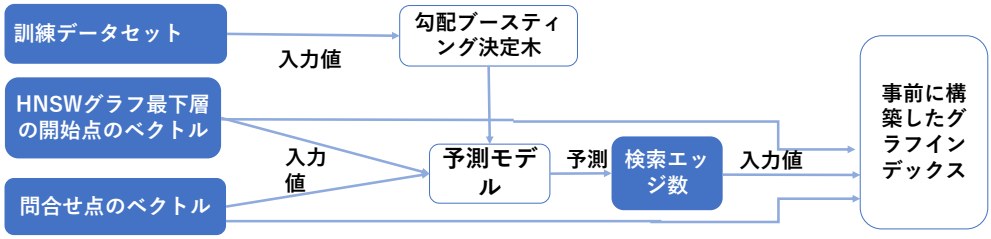

図 6　提案手法の流れ

Input: 問合せ点 Q，返すべき近傍点の数 k，HNSW グラフ G
Output: 検索された結果
while G のベースレイヤに到達していない **do**
    $s \leftarrow$ BeamSearch()
    次のレイヤーに移動します
// 予測モデルを呼び出し，最適なサーチエッジの数を予測する
$features \leftarrow s, q$
$best\_edges \leftarrow$ Predict($features$)
$results \leftarrow$ SearchbyHNSW($best\_edges$)
return $results$

図 7　予測モデルへの HNSW の組み込み

ての決定木の予測値を足し合わせて，最終的な予測値とする．勾配ブースティング決定木は今まで多くのライブラリがリリースされており，マイクロソフトが開発した LightGBM ライブラリ [9] は高速な学習速度と少ないメモリオーバーヘッドにより，企業で広く利用されている．本研究では，LightGBM ライブラリの関数を直接利用して予測モデルを作成する．

また，この勾配ブースティング決定木を選択した主な理由は以下の 3 つである．一つ目はメモリのオーバーヘッドの小ささである．決定木のサイズは数千 KB であり，グラフのサイズに比べてはるかに小さい．二つ目は時間のオーバーヘッドも少ない点である．勾配ブースティング決定木を用いた各問合せ点の予測時間はわずか数十マイクロ秒であり，HNSW の検索時間と比べても短い．三つ目はこのモデルが特徴量の重要性を定量化する点で，特徴量選択などに用いられる関数も持つ．本研究で利用する特徴量が最適な検索エッジの数に与える影響を分析するのに役立つ．

4.4　勾配ブースティング決定木による HNSW

提案手法の処理手順を図 6 に示す．HNSW では近傍点全てを検索するためその検索順序は結果に影響を与えないが，提案手法により検索エッジ数を削減した場合は検索順序も重要となる．そのため，近傍点の検索順が固定されるように各ノードの近傍点はそのノードから近い順に格納する．検索エッジ数が X の場合，近傍点の配列から最初の X 個の近傍点を探索する．

また，最適なエッジ数で検索できるように HNSW の検索関数を変更する．まず関連する特徴量を収集し，最適なエッジ数を予測した上で探索を行えるようにする．

変更した HNSW を用いて，まず近傍点の数がラベルの最大値となるグラフを作成する．次に，データベースから一部のデータ点を取り出し，問合せ点として使用する．これらの問合せ点について 4.1 章の方法で対応するラベルを生成する．また，これらの問合せ点について，4.2 章で説明した特徴量を収集する．そして，学習データセットに基づいて 4.3 章で紹介した勾配ブースティング決定木予測モデルを作る．

問合せ点が与えられると，まず事前に構築されたグラフに対して検索を行い，HNSW グラフにおける最下層の開始点を探す．次に，開始点と問合せ点のベクトルを特徴量とする勾配ブースティング決定木を用いて，最適な検索エッジ数を予測する．最後に，このエッジ数を用いて，結果が返されるまで検索を実行する．

5　評価実験

本章では，提案手法の高速化の有効性を実験により評価する．

5.1　実験設定

まずは，本実験で使用するデータセット，および評価方法についてそれぞれ説明する．

5.1.1　使用データセット

提案手法の評価のために，表 1 に示すように GIST1M, GAUSS，GLOVE1.2M，GLOVE2.2M の 4 つのデータセットを用いる．GIST1M は画像から求まる特徴ベクトルを集めたデータセットである．GLOVE1.2M と GLOVE2.2M は単語ベクトルからなるデータセットである．GAUSS は正規分布に従う生成した人工データである．

表 1　用いたデータセット

データセット	データ数	問合せ点数	次元数	データタイプ
GAUSS	1,000,000	1000	128	人工データ
GLOVE1.2M	1,193,514	1000	200	テキストデータ
GLOVE2.2M	2,196,017	1000	300	テキストデータ
GIST1M	1,000,000	1000	960	画像データ

5.1.2　パラメータ設定と比較手法

GLOVE1.2M と GLOVE2.2M のデータセットの探索難易度が低いため，オリジナルの HNSW と提案手法では，$M = 128$ と $efConstruction = 500$ を用いてグラフインデックスを構築した．他の二つのデータセットに対して，オリジナルの HNSW と提案手法では，$M = 128$ と $efConstruction = 1000$ を用いてグラフインデックスを構築した．

また，各問合せ点に対して学習データのラベルを 16 から 256

まで 16 の倍数に設定し，再現率 0.95 を閾値として使用する．以下，各実験結果では以下の略称で提案手法と比較手法を区別する．

- **Vanilla HNSW**：以上のパラメータ設定でオリジナルの HNSW グラフ構造を構築し，このグラフ上で得られた結果を示す．
- **Optimal HNSW**：3 章の実験から，最大近傍数の異なるグラフに対する同じ問合せデータセットで得られた結果の検索精度は，同じ検索時間でも異なることが分かる．そこで，同じ検索時間で異なるグラフで得られた結果の精度を比較し，その中で最も優れた検索精度を求める．このような操作をいくつかの異なる検索時間に対して行い，各検索時間におけるオリジナルの HNSW の最良の検索精度を近似的に実現する．これらの点を結ぶことで optimal HNSW の結果とする．
- **Our approach**：提案手法の検索結果を示す．

5.1.3 実験方法

本研究では，提案手法の有効性を確認するため，提案手法と HNSW との比較実験を行った．まずは，四つのデータセットにおいて，各問合せ点に対する 100 個の近似最近傍を見つける探索を行う．検索時の $efSearch$ が大きいほど，結果の検索精度が高く，検索時間が長くなる．これらの異なる $efSearch$ の結果を連結することにより，検索精度と検索時間の関係が分かる．検索時間の評価においては，グラフインデックスの生成などの時間は含まず，問合せ点を与えてから近傍点が出力されるまでを計測対象としている．最後に，問合せ点ごとの平均検索時間を結果とする．検索精度の評価においては再現率を用いる．再現率の定義は以下のとおりである．

- **M**：検索された最近傍点の集合．
- **G**：正解となる最近傍点の集合．

$$\text{再現率} = \frac{M \cap G}{G} \tag{1}$$

以上の設定のもと，各手法における処理結果を比較評価する．全ての手法は，同じ問合せデータセットを使って実験された．また，本実験では全てのプログラムを 1 台のサーバ上で実行する．実験に使用したマシンの構成を表 2 に示す．

表 2　実験用サーバの構成

OS	Ubuntu 20.04.5 LTS
CPU	Intel(R) Xeon(R) Gold 6262V (two sockets)
メモリ	DIMM DDR4 (Registered) 2933 MHz (16GB ×14)

5.2 実験結果

5.2.1 予測モデルのトレーニング

本研究では，各データセットについてそれぞれ予測モデルを作成し，各問合せ点に対応する検索エッジ数を予測する．

また，各予測モデルのトレーニング時間および予測時間を集計した．関連情報は表 3 のとおりである．この結果から分かるように，各データセットに対する予測モデルに必要な予測時間は，わずか数十マイクロ秒である．一方，3 章の実験で得られた結果は，0.90 以上の検索精度に到達するためには，問合せ点あたり少なくとも 5ms が必要であることが分る．

表 3　学習データセットに対する実験結果

データセット	サイズ	トレーニング時間 (s)	予測時間 (μs)
GAUSS	100,000	3.21	59.36
GLOVE1.2M	108,502	2.94	31.85
GLOVE2.2M	109,801	2.81	17.51
GIST1M	100,000	10.99	22.43

5.2.2 索引の構築時間とサイズ

表 4　グラフの構築について提案手法と HNSW の比較

データ	アルゴリズム	構築時間 (s)	索引サイズ (GB)
GIST1M	HNSW	2291.88	4.54
GIST1M	提案手法	2558.64	4.54
GAUSS	HNSW	3142.86	1.44
GAUSS	提案手法	3878.36	1.44
GLOVE1.2M	HNSW	2421.69	2.04
GLOVE1.2M	提案手法	2462.48	2.04
GLOVE2.2M	HNSW	6752.86	4.58
GLOVE2.2M	提案手法	6828.34	4.58

表 4 からもわかるように，データセットによって構築時間の増加率が異なることがわかる．提案技術は，全てのノードの近傍点をそのノードから近い順に並べる．オリジナルの HNSW は，近傍点数が M より大きい場合にのみ近傍点を並べ替える．そのため，提案手法の構築時間はオリジナル HNSW よりも長くなる．しかし，提案手法により HNSW の高速化を実現するため，構築時間の増加は許容できると考える．GIST1M,GAUSS,GLOVE1.2M,GLOVE2.2M の四つのデータセットの構築時間の増加率は，それぞれ 11.64%，23.40%，1.68%，1.12% である．時間の増加率は，データセットの分布にも関係していると考える．また，提案手法と HNSW のパラメータが同じであるため，グラフの索引サイズが大きくならないことがわかる．予測モデルのトレーニング時間は，グラフの構築時間に比べてごくわずかである．そのため，予測モデルの導入は HNSW にあまり影響を及ぼさない．

5.2.3 探索時間の比較

前述したデータセットにおいて異なる検索精度を達成するために必要な検索時間を図 8，9 にそれぞれ示す．赤い破線は本研究の提案手法による結果である．オレンジ色の破線は，最大近傍数の異なるグラフを用いた実験から得られた最良の結果である．また，提案手法で用いたグラフと同じ数の最大最近傍を持つ HNSW グラフを構築し，その結果を青い破線で示す．

図 8 の左の図は，GLOVE1.2M データセットにおける提案手法と HNSW の結果を示す．この結果から，提案手法の結果は一貫して HNSW の結果を上回っていることが分かる．つまり，ある検索精度を得るためには，提案手法はオリジナルの HNSW よりも短い検索時間しか必要としない．閾値として 0.95 の検索精度を達成するために，提案手法は約 13ms の検索時間が必要である．同じパラメータで構築した HNSW グラフがこの精度

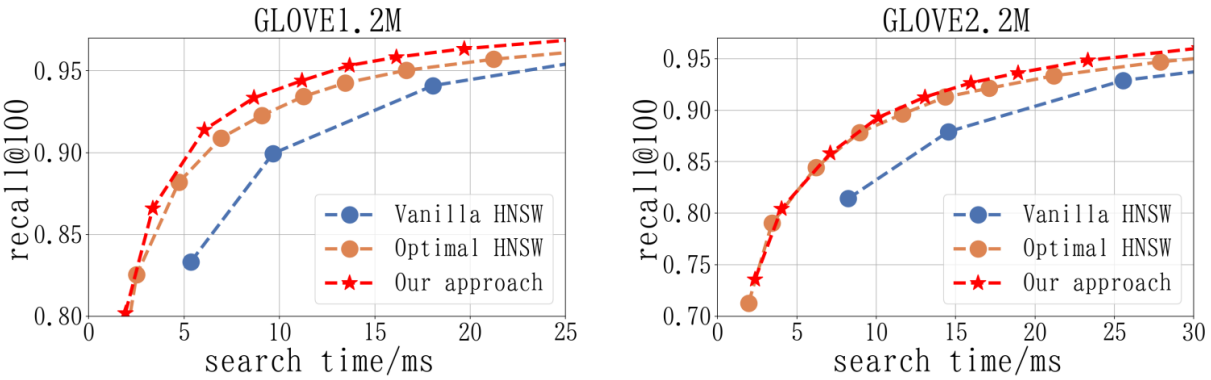

図 8　GLOVE1.2M と GLOVE2.2M における検索結果

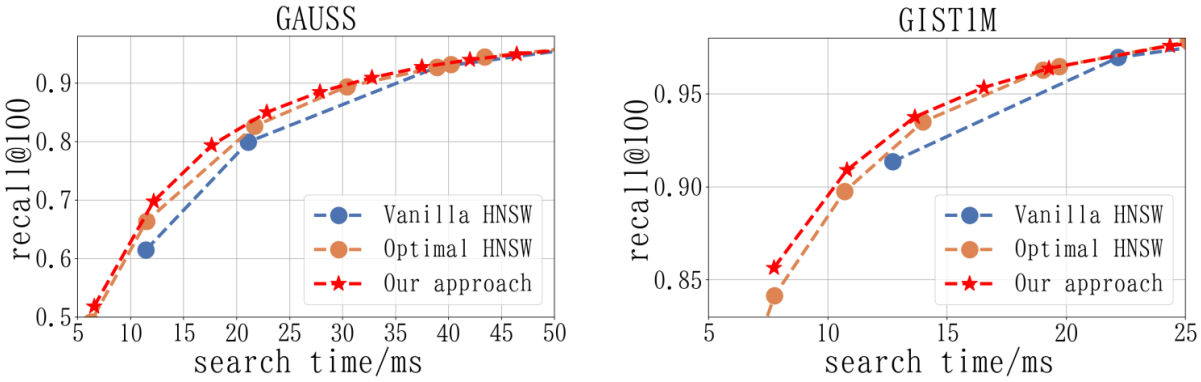

図 9　GAUSS と GIST1M における検索結果

を達成するのには 23ms かかり，検索時間を 30% 短縮できている．また，実験で得られた最も性能の良いグラフと比較しても 23% 短縮されている．以上の結果より，提案手法の高速化の有効性が検証された．

図 8 の右の図は GLOVE2.2M データセットに対する提案手法と HNSW の結果である．この結果から，検索精度が 0.90 以下の場合，提案手法は Optimal HNSW の結果とほぼ同じ時間を要することが分かる．しかし，検索精度が 0.90 以上の場合，提案手法の検索速度は一貫して Optimal HNSW の検索速度より速いことが分かる．また，精度がある閾値を超えた後も，Optimal HNSW の結果より優れていることが分かる．GLOVE2.2M の問合せ点がバラバラに散らばっており，同じ検索エッジ数で全ての問合せ点に対し良い結果を得られないためと考えられる．一方，提案手法は問合せ点ごとに異なる数の検索エッジ数を用いるため，このような問合せデータセットを扱うのに適している．

図 9 の左の図は，GAUSS データセットに対する提案手法と HNSW の結果である．結果から分かるように，要求される検索精度が小さい場合，提案手法の結果は HNSW の最良の結果よりも常に優れている．GAUSS データセットは検索が難しい分布となっているが，検索精度の要求が低い場合，提案手法による検索時間の削減が可能だとわかる．一方，検索精度が 0.95 を超えると，条件を満たすために検索するノード数を減らすことが難しくなる．その結果，検索時間は元の HNSW が必要とする検索時間に近づいている．

図 9 の右の図は，GIST1M データセットに対する提案手法と HNSW の結果である．検索精度は 0.95 以下の場合，提案手法が最も検索性能が高いという予想通りの結果となった．

5.2.4　考　　察

以上の分析から使用するグラフの最大近傍点の数が同じであっても，提案手法によって得られる結果はオリジナルの HNSW によって得られる結果よりもはるかに優れていることが分かる．また，本実験の GIST1M，GLOVE1.2M，GLOVE2.2M 三つのデータセットにおいて，提案手法は必要な検索精度をより速く達成することが分かる．検索が難しいデータセットに対しては，提案手法により要求される検索精度の設定が低い場合の検索時間を短縮することができる．高い要求される精度を達成するのにかかる時間も，オリジナルの HNSW と同じ程度になることが保証される．

4.2 章の分析により，問合せ点の検索のしやすさが主に開始点からの距離と，問合せ点が位置する空間のデータ密度に依存すると考える．オリジナルの HNSW では，異なる問合せ点に応じてパラメータを自動的に設定することを考慮していない．その結果，より大きな検索精度を得るために，検索しやすい問合せ点に対して不要なエッジを大量に検索することになる．

本研究ではこの問題に対処するため，問合せ点と開始点の位置に応じて異なる検索エッジ数を設定し，HNSW の高速化を実現する．最適なエッジ数を予測するのに必要な時間が数十マイクロ秒であり，予測時間は検索時間に比べてほとんど無視できるほど短い．そのため，データセット検索が特に困難な場合，提案手法の検索速度がオリジナル HNSW に劣らないことを保証できる．

また，本研究の提案手法には以下の利点がある．

- 実際の応用では，HNSW の性能を向上させるために，パ

ラメータ M を調整する必要がある．しかし，データセットのサイズが大きいため，グラフ索引の構築には長い時間がかかる．データセットに対して複数の異なるグラフをを構成し，最も性能の良い M を探すことは不可能である．そのため，オリジナルの HNSW は今回の実験における Optimal HNSW と同じ検索性能を実現することは困難である．一方，本研究で提案する手法はより大きなパラメータ M を設定し，一つのグラフを構成するだけで各問合せ点に応じて最適な検索エッジ数で検索できる．これにより，HNSW の性能を向上させる．

- 従来の HNSW ではユーザの異なる精度要求に対応するために，異なるグラフを構築する必要がある．精度の要求が高い場合，より多くの近傍点を持つグラフを構築する必要があるが，そのようなグラフは小さな $efSearch$ の設定でも検索に長い時間を要する．一方，近傍点の数が少ないグラフは高速に結果を出せるが，$efSearch$ を上げても高い検索精度は達成できない．これに対し，提案手法はより多くの最近傍点でグラフを構成するものの，短い検索時間で結果を得ることができる．また，$efSearch$ を上げることで，より高い精度の要求に対応可能である．

6 おわりに

本稿では，HNSW の概要を述べ，HNSW に対する改善点を分析した．予備実験により検索エッジ数の選択が HNSW の性能に影響を与えることを示し，HNSW の問題点を解決するために，機械学習による HNSW の高速化手法を提案した．また，HNSW に学習した予測モデルを取り込み，4 つのデータセットを用いて評価実験を行った．本実験により，要求される検索精度に対し，提案手法はオリジナルの手法よりも高速に近似最近傍点が検索できることを検証した．

今後の課題としては，検索の進行段階に応じた最適なエッジ数予測への拡張が挙げられる．また，この手法を他のグラフベース近似最近傍検索手法に拡張する予定である．

謝　　辞

本研究は JSPS 科研費 JP20K19804，JP21H03555，JP22H03594，JP22H03903 の助成を受けたものである．

文　　献

[1] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *IEEE TPAMI*, vol. 42, no. 4, pp. 824–836, 2018.

[2] C. Fu, C. Wang, and D. Cai, “Fast approximate nearest neighbor search with navigating spreading-out graphs,” *CoRR*, vol. abs/1707.00143, 2017.

[3] C. Fu, C. Wang, and D. Cai, “High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[4] C. Li, M. Zhang, D. G. Andersen, and Y. He, “Improving approximate nearest neighbor search through learned adaptive early termination,” in *Proc. SIGMOD*, 2020.

[5] D. Baranchuk, D. Persiyanov, A. Sinitsin, and A. Babenko, “Learning to route in similarity graphs,” *CoRR*, vol. abs/1905.10987, 2019.

[6] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, “Approximate nearest neighbor algorithm based on navigable small world graphs,” *Information Systems*, vol. 45, pp. 61–68, 01 2013.

[7] GIST1M: `http://corpus-texmex.irisa.fr/` (accessed: Jan 8, 2023).

[8] GloVe: `https://nlp.stanford.edu/projects/glove/` (accessed: Jan 8, 2023).

[9] LightGBM Documentation: `https://lightgbm.readthedocs.io/en/latest/index.html` (accessed: Jan 8, 2023).