CD-001

高速・高精度な k 最近傍距離推定

新井 悠介, 天方 大地, 原 隆浩
大阪大学
{arai.yusuke,amagata.daichi,hara}@ist.osaka-u.ac.jp

藤田 澄男
ヤフー株式会社
sufujita@yahoo-corp.jp

アブストラクト

ビッグデータマイニングはデータサイエンスにおける重要なタスクであり，ビッグデータに隠れた新たな知識の獲得等が期待できる．特にデータ間の類似性を利用したデータ分析は多くのアプリケーションで利用されており，k 最近傍までの距離を利用するものが多い．この時，この距離を計算する操作が分析の際にボトルネックとなる．既存研究ではこの操作の効率性を向上する方法が提案されているが，多くのデータアクセスを要するため，大量のデータにスケールしない．

そこで本研究では，全結合ニューラルネットワークとピボットを用いた k 最近傍距離を高速かつ高精度で推定する機械学習モデルを提案する．このモデルは，最近傍から k 最近傍までの距離を同時かつ $O(1)$ 時間で推定することができるという特長がある．実データを用いた実験およびケーススタディから，提案モデルの有効性を示す．

1 はじめに

大量のデータを分析する際は，データ空間の局所的な構造に注目してデータ集合の特徴を理解することが多い [7, 10]．また，データ空間の局所的な疎密を把握する方法の一つとして，クエリ（データ）とその k 最近傍の間の距離を利用する方法がある．この方法は，幾何学的推論 [2]，外れ値検出 [14, 25]，異常検知 [3]，クラスタリング [5, 26]，および遺伝子解析 [27] といった幅広い分野で応用されており，k 最近傍までの距離を測ることの重要性は明らかである．以下では，k 最近傍までの距離を利用する具体的なアプリケーションを紹介する．

例 1（k 最近傍密度推定）． 密度推定は，データ分析で頻繁に利用される可視化のための手法の一つである [2, 18, 22]．k 最近傍密度推定 [2, 18] は k 最近傍までの距離を用いる密度推定法である．文献 [2] は，式 (1) に示す重みつき k 最近傍密度推定量を提案している．

$$\hat{f}(q)=\frac{1}{nV_d}\left(\frac{\sum_{j=1}^{k} j^{d/2}}{\sum_{j=1}^{k} dist(q,x_q^j)^d}\right)^{d/2} \tag{1}$$

ここで，q はクエリ，n はデータ集合の要素数，V_d は d 次元ユークリッド空間における単位超球の体積，x_q^j は q に最も近い j 番目のデータ，$dist(\cdot,\cdot)$ は 2 点間のユークリッド距離である．k 最近傍密度を可視化するためには，出力する画像の解像度を決定し，そのピクセルごとに式 (1) を計算して画面上に表示する．

例 2（外れ値検出）． 外れ値検出は異常検知およびデータクリーニングなどに応用される重要な技術の一つであり [7]，その方法の一つとして，k 最近傍までの距離が大きいデータを外れ値とみなす方法が知られている．文献 [14] は，k 最近傍までの距離が閾値よりも大きいデータを外れ値とみなす方法を提案している．また文献 [25] は，k 最近傍までの距離が最も大きい N 個のデータを外れ値とみなす方法を提案している．

以上で紹介したアプリケーションは実世界で広く利用されており [2, 3, 27]，大規模データにスケールする技術を求めているため，その高速化は重要である．よって，この要件を満たすためには k 最近傍までの距離を高速に計算する必要がある．さらに，例 1 で述べたようなアプリケーションでは，最も近い点との距離から k 番目に近い点との距離までを高速に推定できる方法を求めている．従来技術では，クエリ（データ）からその k 最近傍までの距離を計算するためには，k 最近傍検索が必要となる．しかし，距離のみを知りたい場合，検索操作は大量のデータにアクセスするため効率的でない．

そこで本研究では，高速かつ高精度な k 最近傍距離推定法である PivNet を提案する．PivNet は，クエリのベクトルに加え，データ空間内に配置するピボットの k 最近傍距離を特徴量として利用したニューラルネットワークにより，任意のクエリに対して高精度に k 最近傍距離を推定する．そして，推定にかかる時間計算量は $O(1)$ である．PivNet は，クエリに最も近い点との距離から k 番目に近い点との距離までを一度の順伝搬で推定できる．本研究の貢献を以下に述べる．

- 高速かつ高精度な k 最近傍距離推定法である PivNet を提案する．
- 実データを用いた実験により，PivNet は k 最近傍検索よりも高速に動作することを示す．
- PivNet を k 最近傍密度推定および外れ値検出に適用し，PivNet の有用性を示す．

2 問題定義

本論文で取り組む問題を定義する．

定義 1（k 最近傍距離推定）． d 次元実ベクトル空間 $\mathbb{R}^d$ において，データ集合 $X\subset\mathbb{R}^d$，クエリ $q\in\mathbb{R}^d$ が与えられ，集合 S が q の k 最近傍であるとする．本問題は，q と S の各要素との距離を推定することである．

本研究では，k の最大値（k_{max}）が存在すると仮定し，k_{max} 最近傍距離推定法を提案する．また，k_{max} はアプリケーションが事前に指定するものとする．

クエリ q に k 番目に近い点を x_q^k，$\hat{dist}(q,x_q^k)$ をその 2 点間の距離の推定値とする．1章で述べた例から，推定値の絶対誤差 $|dist(q,x_q^k)-\hat{dist}(q,x_q^k)|$ を最小化することが重要である．したがって，推定精度の評価指標として次に示す平均絶対誤差を用いる．

$$MAE_q=\frac{\sum_{k=1}^{k_{max}}|dist(q,x_q^k)-\hat{dist}(q,x_q^k)|}{k_{max}} \tag{2}$$

また1章で述べたように，k 最近傍距離のアプリケーションの多くは低次元空間を対象としているため，本研究では d が小さい場合を考える．

3 関連研究

k 最近傍検索 は機械学習 [12]，データマイニング [10]，および情報検索 [24] をはじめとする幅広い分野で利用されている．そのため，高速な k 最近傍検索アルゴリズムが数多く提案されており，その中でも木構造を用いたアルゴリズムが最も広く用いられている．木構造による k 最近傍検

第 2 分冊

索 [1, 6] では，あらかじめデータ空間を階層的に分割しておく．クエリが発行されると，まず k 最近傍距離の上界値を十分に大きな値に設定する．次に子ノードにアクセスするたびに k 最近傍距離の上界値を更新し，枝刈りしながら k 最近傍を検索する．

k 最近傍検索を k 最近傍距離推定に利用するには，検索によって得られた k 最近傍とクエリとの距離を調べる．しかし，距離のみを知りたい場合，検索操作は大量のデータにアクセスするため効率的でない．例えば，文献 [29] は，kd 木を用いた k 最近傍検索において $O(kn^{1-1/d})$ 回のデータアクセスが起こり得ることを報告している．

Learned Index. 近年，検索インデックスをモデルとして捉え，入力をクエリ，出力を検索結果（解となるデータのアドレス）とする機械学習モデルである Learned Index が提案されている．これまでに様々な Learned Index [15, 16, 21, 23, 34] が提案されており，最初に提案された Learned Index である RMI [15] は，1 次元空間を想定し，いくつかのニューラルネットワークを使用してクエリの位置を推定し，木構造よりも高速に検索できる．文献 [15, 16] は，多次元データにおける各次元の範囲検索に焦点を当てている．文献 [16, 21] で提案している方法は近似 k 最近傍検索に適用できるが，データをディスクに格納している環境を想定しており，ディスクの I/O を最小化するようにモデルが設計されている．これらの方法は，インメモリ環境での k 最近傍距離推定に適用できない．また，文献 [34] で提案している方法は 2 次元データ以外では利用できない．

カーディナリティ推定. カーディナリティ推定は，クエリとの距離が閾値未満のデータの数を推定する問題であり，クエリ最適化 [9] および機械学習 [32] などに応用されている．ヒストグラムによるアプローチ [8] では，あらかじめデータ空間に多次元ヒストグラムを構築しておく．推定の際は，クエリを中心，閾値を半径とする超球に外接するバケットを参照する．そして，バケットの中でデータが一様分布していると仮定し，外接バケットの体積に対するクエリを中心とする超球が占める体積の比とバケット内のデータ数を掛け合わせることにより，カーディナリティを推定する．ヒストグラムによるアプローチは，データがバケット内で一様分布していることを仮定しているため，そうでない場合の精度が低い．

ヒストグラムによるアプローチよりも柔軟にデータの分布を推定するため，機械学習を用いた方法が提案されている [4, 13, 17, 28, 30, 31, 33, 35]．文献 [4] は，クエリのベクトルに加え，データ空間内に構築した多次元ヒストグラムにおけるバケット内のデータ数を特徴量として利用することで，高精度なカーディナリティ推定を実現している．文献 [30, 31] は，単調性（閾値が大きくなればカーディナリティもまた大きくなる性質）を保証している．

カーディナリティ推定によって k 最近傍距離を推定するには，推定したカーディナリティが k となるような距離の閾値を探索する．しかし，カーディナリティの推定値は連続値であり，探索には無限回の試行を要するため，k 最近傍距離推定への応用は現実的でない．

4 提案手法

アイデア. k 最近傍距離推定には，高速性と高精度が求められる．これらの要求に応えるための第一のアイデアとして，ピボットを用いる方法を提案する．この方法では，多くのピボットをデータ空間に配置する．具体的には，データ空間を細かいグリッドによって均等に分割し，各セルの中心にピボットを配置する（つまりあるピボットは該当セルのセントロイドである）．クエリが発行されると，クエリが属するセルのピボットを参照する．グリッド分割は固定長の配列を用いて実装できるため，クエリが属するセルは $O(1)$ 時間で計算できる．ここで，クエリ q，ピボット $p \in \mathbb{R}^d$，p と p に k 番目に近い点 $x_p^k \in X$ の間の距離 $dist(p, x_p^k)$ が与えられ，x_p^k が条件 $x_p^k \notin \{x_q^i \mid i = 1, \cdots, k-1\}$ を満たすとき，次が成り立つ．

$$dist(q, x_q^k) \leq dist(q, p) + dist(p, x_p^k) \tag{3}$$

これは，（i）上の仮定および k 最近傍距離の性質から $dist(q, x_q^k) \leq dist(q, x_p^k)$ であり，（ii）q，p，および x_p^k について，三角不等式より $dist(q, p) + dist(p, x_p^k) \geq dist(q, x_p^k) \geq dist(q, x_q^k)$ となるためである．

ここで，ピボットの k 最近傍距離は事前に計算可能であることに注意すると，上の条件が成り立つとき，クエリの k 最近傍距離の上界値が高速に得られる．セル数を十分に大きくとれば（グリッドの粒度が十分細かければ），クエリに近いピボットを選択でき，$dist(q, p) \approx 0$ となる．このとき，上界値はクエリの k 最近傍距離に近づくため，式 (3) の右辺によって高精度な推定値が得られる．しかし，クエリとピボットがこの条件を満たさない場合，その保証はできない．また，この方法で高精度に k 最近傍距離を推定するためには，細かいグリッド分割が必要である．各次元を g 個に分割すると空間計算量が $O(g^d)$ となることから，グリッド分割を細かくすると，特に次元数が大きい場合，メモリ消費量が急激に増加する．そこで本研究では，クエリとピボットの間の距離とピボットの k 最近傍距離を単純に足し合わせるのではなく，それらを特徴量とする機械学習モデルによって，k 最近傍距離を高精度に推定する関数を獲得する．これが本研究における第二のアイデアである．

ニューラルネットワークによるモデリング. 本問題は，クエリベクトル q，k，q が属するセルのピボット p との距離 $dist(q, p)$，および p から k 番目に近い点までの距離 $dist(p, x_p^k)$ を特徴量とし，k 最近傍距離を目的変数とする回帰問題に帰着できる．また，真の q の k 最近傍距離は，これまでに提案されてきた k 最近傍検索アルゴリズムによって得られるため，本問題は教師あり回帰問題として解ける．教師あり回帰問題を解くための方法は数多く提案されているが，本研究では，多くの分野で成功を収めている機械学習モデルであるニューラルネットワークを利用する．本研究で用いるニューラルネットワークは，5 層の全結合層で構成され，活性化関数には ReLU [20] を用いる．そして，k 最近傍距離の推定値を出力し，確率的勾配降下法によって式 (4) を最小化するよう訓練する．

$$\mathcal{L} = \frac{1}{k_{max}|Q_{train}|} \sum_{\langle q,k \rangle \in P} |dist(q, x_q^i) - \hat{dist}(q, x_q^k)| \tag{4}$$

ただし，Q_{train} は訓練クエリの集合，P は $q \in Q_{train}$ と $k \in [1, k_{max}]$ の全てのペアの集合，p は q が属するセルのピボットである．また，$\hat{dist}(q, x_q^k)$ は次に示すニューラルネットワーク f によって得られる．

$$\hat{dist}(q, x_q^k) = f(q, i, dist(q, p), dist(p, x_p^i)) \tag{5}$$

これにより，クエリとピボットの間の距離とピボットの k 最近傍距離の和を推定値とする場合に比べて，推定誤差を最小化するようにクエリとピボットの位置関係を考慮できるため，膨大な数のセルを用意することなく高精度に k 最近傍距離を推定できる．

最適化. k 最近傍距離を推定する際に，アプリケーションの要求に応じてオンラインで k を変更できる必要がある．また，どのようなクエリに対しても高精度に推定を行うためには，多くのクエリでモデルを訓練しなければならない．したがって，同じクエリに対して何度も k を変えて訓練する必要があるため，訓練に多くの時間がかかる．また，

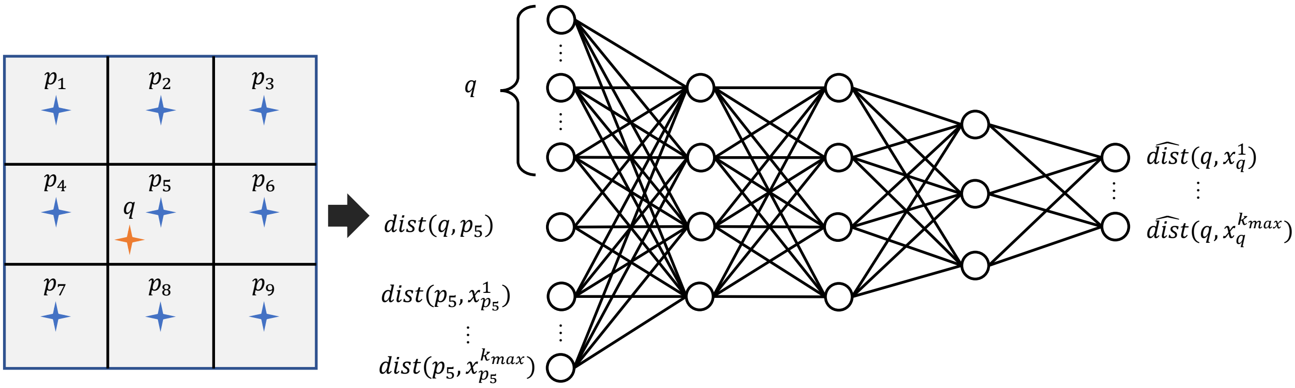


図 1: PivNet の推定パイプライン．$p_1, ..., p_9$ はピボットを表し，$\hat{dist}(\cdot,\cdot)$ は 2 点間の距離の推定値を表す．

k 最近傍距離推定のアプリケーションには，クエリに最も近い点との距離から k 番目に近い点との距離までを同時に要求するものがある [2]．クエリと k を入力してクエリに k 番目に近い点との距離を推定する方法では，k を何度も変えて推定を繰り返すため，アプリケーションの実行速度の低下を招く．

そこで提案手法では，図1に示すように，クエリのベクトル，クエリとピボットの間の距離，およびピボットの k_{max} 最近傍距離を特徴量とし，クエリに最も近い点との距離から k 番目に近い点との距離までの推定値を結合したベクトルを出力する．そして，式 (6) に示す損失関数を最小化するよう訓練する．

$$\mathcal{L}_{opt} = \frac{1}{|Q_{train}|} \sum_{q \in Q_{train}} \frac{||\mathbf{v}_q - \hat{\mathbf{v}}_q||_1}{k_{max}} \tag{6}$$

ただし，

$$\begin{aligned}\hat{\mathbf{v}}_q &= f_{PivNet}(q, dist(q,p), \mathbf{v}_p),\\ \mathbf{v}_p &= \langle dist(p, x_p^1), ..., dist(p, x_p^{k_{max}})\rangle,\\ \mathbf{v}_q &= \langle \hat{dist}(q, x_q^1), ..., \hat{dist}(q, x_q^{k_{max}})\rangle.\end{aligned}$$

である．また，p はピボット集合の中で q に最も近いピボットを表す．これにより，オンラインにおける k の変更が可能でありながら，訓練および k 最近傍距離推定の高速化が可能となる．

時間計算量． クエリの入るセルを計算するのに要する時間は $O(1)$ である．また，ニューラルネットワークにおいて，その層数，入力層のニューロン数，隠れ層のニューロン数はそれぞれ $O(1)$，出力層のニューロン数は $k_{max} = O(1)$ である．したがって，PivNet による k 最近傍距離推定に要する時間は $O(1)$ である．

訓練クエリ． モデルを訓練するためのクエリを準備するための最も単純な方法は，X からランダムサンプリングすることである．しかし，そのような訓練クエリを用いると X の分布に過剰適合し，例 1 で求められているようなデータ空間上の一様分布に従うクエリに対する高精度な予測ができない可能性がある．本研究では，この課題を解決するため，X からランダムにサンプリングした訓練クエリにデータ空間上の一様分布に従うクエリを加える．そして，各クエリ $q \in Q_{train}$ に対してデータ集合 $X \backslash Q_{train}$ の中から k 最近傍を計算する．

5 評価実験

本実験のプログラムは，Python 3.8 を用いて実装し，Ubuntu 18.04 LTS OS，Core i9-10980XE（18 コア，3.0GHz）および 128GB RAM を搭載した計算機上で実行した．

比較手法． 本実験では，PivNet と以下に示す手法を比較した．

- kd 木：kd 木 [1] による k 最近傍検索．Scipy[1]による実装を利用した．
- LightGBM [11]：勾配ブースティング決定木による回帰予測モデル．PivNet と同じ特徴量を入力することにより，PivNet のニューラルネットワークの有効性を確認する．
- Pivot：式 (3) の右辺を推定値として用いる方法．
- QueryNet：クエリのベクトルのみを特徴量として用いるニューラルネットワーク．
- PivNet-itr：式 (5) に示すニューラルネットワーク．各 $k \in [1, k_{max}]$ に対して繰り返し推定を行うことにより k 最近傍距離推定を行う．

以上の手法は全てシングルスレッドおよびインメモリ環境で実行した．QueryNet，PivNet-itr，および PivNet は PyTorch を用いて訓練した．

データセット． 評価に使用したデータセットを表1に示す．

訓練方法． データセットをランダムに分割し，10 万件を訓練クエリ，1 万件をテストクエリ，残りを X として扱った．さらに，データ空間上の一様分布から 10 万件および 1 万件のデータをサンプリングし，それぞれを訓練クエリおよびテストクエリに追加した．また，訓練データのうち 80% を訓練に使用し，残りの 20% を検証データとして用いた．

モデルの設定． 推定モデルのハイパーパラメータを表2に示す．ニューラルネットワークの損失関数は平均絶対誤差を使用し，確率的勾配降下法によりパラメータを最適化した．

評価指標． 推定精度の評価指標として，式2で定義した MAE_q の平均値および中央値を使用した．また，1 クエリあたりの k_{max} 最近傍距離の推定時間の平均値，および訓練時間を評価した．ただし，訓練時間はピボットの構築時間を含む．

5.1 結果

推定誤差を表3に示す．また，推定時間および訓練時間の結果を表4に示す．

ニューラルネットワークの効果． 表3より，Pivot，PivNet-itr，および PivNet を比較すると，PivNet-itr または PivNet の方が精度が高いことが分かる．これは，ニューラルネットワークによって，クエリとピボットの間の距離とピボットの k 最近傍距離を単純に足し合わせるだけでなく，これらの位置関係を考慮して柔軟にクエリの k 最近傍距離

[1]https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html

表 1: データセット

データセット名	データ数	次元数	備考
Crime[2]	259,809	2	アメリカのアトランタ州で発生した犯罪情報
HEPMASS[3]	10,049,359	5	高エネルギー物理の実験データ
Household[4]	1,771,612	4	電化製品の消費電力
PAMAP2[5]	2,643,873	5	行動中に取得したセンサデータ
Places[6]	9,033,486	2	アメリカにおける公共施設の位置情報
Wisdm[7]	4,476,481	3	行動中に取得したセンサデータ

表 2: ハイパーパラメータ

		Crime	HEPMASS	Household	PAMAP2	Places	Wisdm
グリッド数		2048	32	32	32	2048	256
第 2 層のニューロン数		64	128	128	128	64	128
第 3 層のニューロン数		64	128	128	128	64	128
第 4 層のニューロン数		32	32	32	32	32	32
バッチサイズ		500	500	500	500	500	500
学習率	QueryNet	0.1	0.1	0.1	0.2	0.2	0.1
	PivNet-itr	0.01	0.003	0.005	0.003	0.003	0.005
	PivNet	0.03	0.02	0.01	0.04	0.01	0.02

表 3: 推定誤差の平均値および中央値

		Crime	HEPMASS	Household	PAMAP2	Places	Wisdom
LightGBM	平均値	0.00057	0.05989	0.09785	0.09005	0.09216	0.03367
	中央値	0.00014	0.01714	0.02031	0.04586	0.00284	0.01168
Pivot	平均値	0.00049	0.12295	0.28569	0.38003	0.00811	0.05177
	中央値	0.00046	0.12486	0.28337	0.37429	0.00742	0.05077
QueryNet	平均値	0.00075	0.02225	0.02949	0.07335	0.07702	0.04213
	中央値	0.00048	0.01597	0.01691	0.04512	0.02336	0.02483
PivNet-itr	平均値	0.00018	0.00977	0.01024	0.03828	0.00711	0.01184
	中央値	0.00012	0.00578	0.00536	0.02499	0.00313	0.00806
PivNet	平均値	0.00019	0.01965	0.01852	0.04768	0.01113	0.01038
	中央値	0.00013	0.01117	0.00916	0.03528	0.00320	0.00748

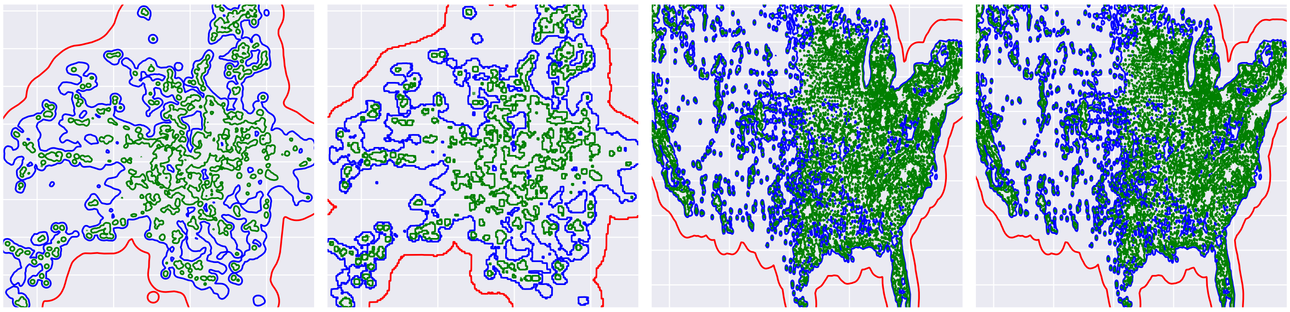
表 4: 推定時間および訓練時間

		Crime	HEPMASS	Household	PAMAP2	Places	Wisdom
kd 木	検索時間 [μsec]	56.04	148.02	104.68	154.18	54.46	65.77
LightGBM	推定時間 [μsec]	13.20	11.35	11.44	11.36	13.67	13.47
	訓練時間 [min]	0.32	0.34	0.32	0.34	0.32	0.31
Pivot	推定時間 [μsec]	1.36	1.60	1.15	1.48	1.62	1.67
QueryNet	推定時間 [μsec]	1.50	2.02	2.23	2.63	1.05	2.14
	訓練時間 [min]	6	2	12	5	7	7
PivNet-itr	推定時間 [μsec]	191.49	193.84	192.37	193.37	192.38	196.04
	訓練時間 [min]	119	575	217	746	47	232
PivNet	推定時間 [μsec]	3.24	3.94	5.65	5.13	3.58	4.81
	訓練時間 [min]	15	6	9	3	10	38

表 5: k を変化させたときの推定精度の変化

	Crime	HEMPASS	Household	PAMAP2	Places	Wisdom
$k \in [1, 10]$	0.00023	0.02214	0.02001	0.05660	0.02167	0.01315
$k \in [11, 20]$	0.00019	0.01930	0.01790	0.04582	0.00995	0.01029
$k \in [21, 30]$	0.00019	0.01888	0.01775	0.04497	0.00648	0.00961
$k \in [31, 40]$	0.00017	0.01888	0.01827	0.04525	0.00733	0.00958
$k \in [41, 50]$	0.00019	0.01926	0.01890	0.04665	0.01097	0.00951

を推定できているためだと考えられる．この影響は，次元数が大きくなるほど顕著に表れている．これは，次元が大きいほどデータ空間が疎になりやすいため，ピボットとクエリの距離が大きくなったことが原因だと考えられる．さらに PivNet は，Places における中央値を除いて，LightGBM よりも高精度に推定できていることが分かる．このことか



(a) 厳密解（Crime） (b) PivNet による近似解（Crime） (c) 厳密解（Places） (d) PivNet による近似解（Places）

図 2: k-NN 密度の可視化

ら，決定木による推定器よりもニューラルネットによる推定器の方が優れていることが分かる．

ピボット特徴量の効果． 表3より，QueryNet と PivNet-itr および PivNet を比較すると，PivNet-itr または PivNet の方が常に精度が高いことが分かる．この結果は，ピボット特徴量が推定精度の向上に寄与していることを示している．

PivNet-itr との比較． 表3より，PivNet-itr と PivNet を比較すると両者は同等の精度となっている．しかし表4より，PivNet-itr の訓練時間および推定時間は PivNet に比べて大きく，また推定時間は kd 木による検索時間よりも大きくなっている．これは，PivNet-itr によって 50 最近傍距離を推定する際に，k を変えて繰り返しニューラルネットワークの順伝搬が必要なことが原因である．それに対し PivNet は，1 回の順伝搬で 50 最近傍距離を同時に出力できるため，kd 木および PivNet-itr よりも高速に動作する．以上より，推定時間，推定精度，および訓練時間を考慮すると PivNet が最も優れていることが分かる．

k の影響． 区間 [1, 50] を 5 分割し，それぞれの区間における MAE の推定値を表5に示す．表5より，$k \in [1, 10]$ の場合はそれ以外の場合に比べて MAE が大きいことが分かる．これは，k が小さいときの k 最近傍距離は，クエリ周辺の点の密度の影響を強く受けることが原因だと考えられる．データ空間の疎な部分にある点の数は密な部分にある点の数よりも小さいため，ニューラルネットの特性上，疎な部分に発行されたクエリに対する推定が難しい可能性がある．しかし，PivNet はそのような状況でも十分に小さい誤差で推定できている．

6 ケーススタディ

本章では，PivNet を k 最近傍密度推定および外れ値検出に利用し，その有効性を確認する．

6.1 k 最近傍密度推定の高速化

設定． 本実験には，2 次元のデータセットとして Crime および Places を用いた．まずデータ空間を $1,000 \times 1,000$ に等分割した．次に，各セルの中央の座標をクエリとして，その k 最近傍密度推定量を計算した．その計算には，式 (1) に示す重みつき k 最近傍密度推定量 [2] を使用した．本実験では，$k = 100$ に設定した．そしてその結果を等高線によって可視化し，可視化にかかった時間を評価した．等高線の間隔は，20 パーセンタイル値，60 パーセンタイル値，90 パーセンタイル値とした．また，次に示す方法を用いた．

- kd 木 [1]：kd 木によって厳密に求めた 100 最近傍距離を用いる方法.
- PivNet：PivNet により求めた 100 最近傍距離の推定値を用いる方法.

表 6: 重みつき k 最近傍密度推定に要した時間 [sec]

	Crime	Places
kd 木	16.2	35.8
PivNet	3.1	3.1

結果． 可視化結果を図2に，推定にかかった時間を表6に示す．図2より，両者はほぼ同じ可視化結果を生成していることが分かる．また表6より，PivNet による近似により，Crime では約 5 倍，Places では約 11 倍の高速化を達成している．

6.2 外れ値検出の高速化

本実験では，次に定義する外れ値検出について考える．

定義 2（**(r, k)-distance-based outlier detection [14]**）**.** 点集合 X, k, および距離の閾値 r が与えられたとする．(r, k)-distance-based outlier detection は，$dist(x, x') \le r$ を満たす点 $x' \in X$ の数が k 未満である点 $x \in X$ を求める操作である．

定義 3（**(N, k)-distance-based outlier detection [25]**）**.** 点集合 X，k，および N が与えられたとする．(N, k)-distance-based outlier detection は，k 番目に近い点との距離が最も大きい N 個の点 $x \in X$ を求める操作である．

本実験では，点集合 X の各要素に対して k 番目に近い点との距離を推定することにより，(r, k)-distance-based outlier detection（(r, k)-DOD）および (N, k)-distance-based outlier detection（(N, k)-DOD）を行う．

設定． 本実験では，$k = 50$ に設定した．(r, k)-DOD における r を，外れ値の数が 1,000 になるように設定した．同様に，(N, k)-DOD における N を 1,000 に設定した．それぞれの検出方法について，kd 木 [1] および PivNet を用いたときの検出時間を比較した．(r, k)-DOD における PivNet の評価指標には Precision と Recall を，(N, k)-DOD における評価指標には Recall を使用した．Precision および Recall の定義

[2] http://opendata.atlantapd.org
[3] https://archive.ics.uci.edu/ml/datasets/HEPMASS
[4] https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption
[5] https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring
[6] https://archive.org/details/2011-08-SimpleGeo-CC0-Public-Spaces
[7] https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+

表 7: 外れ値検出に要した時間 [sec]

	HEPMASS	Household	PAMAP2	Wisdm
kd 木	964	93	248	102
PivNet	30	5	7	13

表 8: PivNet による外れ値検出の Precision および Recall

DOD		HEPMASS	Household	PAMAP2	Wisdm
(r,k)	Prec.	0.85	0.85	0.86	0.94
	Recall	0.93	0.88	0.86	0.95
(N,k)	Recall	0.90	0.87	0.86	0.95

式を式 (7) および (8) に示す．

$$Precision = \frac{\text{検出した外れ値に含まれる真の外れ値の数}}{\text{検出した外れ値の数}} \quad (7)$$

$$Recall = \frac{\text{検出した外れ値に含まれる真の外れ値の数}}{\text{真の外れ値の数}} \quad (8)$$

本実験では, HEPMASS, Household, PAMAP2, および Wisdm を用いて評価した．

結果． 外れ値検出に要した時間を表7に，PivNet による外れ値検出の Recall を表8に示す．表7より，PivNet による近似によって 7 倍から 35 倍の高速化を実現している．PivNet による方法は kd 木による方法と異なり，$O(1)$ の時間計算量で推定が可能なため，データ数が増えても高速性が保持される．表8より，PivNet による検出方法によって，真の外れ値のうち大部分を検出できていることが分かる．

7 まとめ

本章では，k 最近傍距離推定法である PivNet を提案した．PivNet は，クエリのベクトルに加え，データ空間内に配置するピボットの k 最近傍距離を利用することにより，任意のクエリに対して高精度に k 最近傍距離を推定する．そして，推定にかかる時間計算量は $O(1)$ である．実データを用いた実験により，PivNet は，単純な方法に比べて高速かつ高精度な推定ができることを示した．また，訓練した PivNet を k 最近傍密度推定および外れ値検出に利用し，その有効性を確認した．

本章では低次元データに対する k 最近傍距離推定について考えた．高次元データに本研究を適用するための方法として，主成分分析および t-SNE [19] などの次元削減があるが，次元削減では大域的な構造を捉えきれない．高次元空間における k 最近傍距離推定は今後の課題とする．

謝辞

本研究の一部は，JST さきがけ（JPMJPR1931），JST CREST（JPMJCR21F2），および文部科学省科学研究費補助金・基盤研究 (A)(18H04095) の支援を受けたものである．

参考文献

[1] Jon Louis Bentley. 1979. Multidimensional Binary Search Trees in Database Applications. *IEEE Trans. Software Engineering* 5, 4 (1979), 333–340.
[2] Gérard Biau, Frédéric Chazal, David Cohen-Steiner, Luc Devroye, and Carlos Rodriguez. 2011. A Weighted k-Nearest Neighbor Density Estimate for Geometric Inference. *Electronic Journal of Statistics* 5 (2011), 204–237.
[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Compututing Survey* 41, 3 (2009), 15:1–15:58.
[4] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity Estimation for Range Predicates Using Lightweight Models. *PVLDB* 12, 9 (2019), 1044–1057.
[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD.* 226–231.
[6] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD.* 47–57.
[7] Victoria J. Hodge and Jim Austin. 2004. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review* 22, 2 (2004), 85–126.
[8] Yannis Ioannidis. 2003. The History of Histograms (Abridged). In *VLDB.* 19–30.
[9] Yannis E. Ioannidis. 1996. Query Optimization. *ACM Computing Survey* 28, 1 (1996), 121–123.
[10] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. 1999. Data Clustering: A Review. *ACM Computing Survey* 31, 3 (1999), 264–323.
[11] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In *NIPS.* 3149–3157.
[12] James M. Keller, Michael R. Gray, and James A. Givens. 1985. A fuzzy K-nearest neighbor algorithm. *IEEE Trans. Systems, Man, and Cybernetics* SMC-15, 4 (1985), 580–585.
[13] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *CIDR.*
[14] Edwin M. Knorr and Raymond T. Ng. 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB.* 392–403.
[15] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The Case for Learned Index Structures. In *SIGMOD.* 489–504.
[16] Pengfei Li, Hua Lu, Qian Zheng, Long Yang, and Gang Pan. 2020. LISA: A Learned Index Structure for Spatial Data. In *SIGMOD.* 2119–2133.
[17] Qiyu Liu, Yanyan Shen, and Lei Chen. 2021. LHist: Towards Learning Multi-dimensional Histogram for Massive Spatial Data. In *ICDE.* 1188–1199.
[18] D.O. Loftsgaarden and C.P. Quesenberry. 1965. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics* 36 (1965), 1049–1051.
[19] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[20] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML.* 807–814.
[21] Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. Learning Multi-dimensional Indexes. In *SIGMOD.* 985–1000.
[22] Emanuel Parzen. 1962. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics* 33, 3 (1962), 1065–1076.
[23] Jianzhong Qi, Guanli Liu, Christian S Jensen, and Lars Kulik. 2020. Effectively Learning Spatial Indices. *PVLDB* 13, 12 (2020), 2341–2354.
[24] Prabhakar Raghavan. 1997. Information Retrieval Algorithms: A Survey. In *SDM.* 11–18.
[25] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. In *SIGMOD.* 427–438.
[26] Alex Rodriguez and Alessandro Laio. 2014. Clustering by fast search and find of density peaks. *Science* 344, 6191 (2014), 1492–1496.
[27] Nikolay Samusik, Zinaida Good, Matthew H. Spitzer, Kara L. Davis, and Garry P. Nolan. 2016. Automated mapping of phenotype space with single-cell data. *Nature Methods* 13, 6 (2016), 493–496.
[28] Ji Sun, Guoliang Li, and Nan Tang. 2021. Learned Cardinality Estimation for Similarity Queries. In *SIGMOD.* 1745–1757.
[29] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. 2017. Handbook of Discrete and Computational Geometry. 1065.
[30] Yaoshu Wang, Chuan Xiao, Jianbin Qin, Xin Cao, Yifang Sun, Wei Wang, and Makoto Onizuka. 2020. Monotonic Cardinality Estimation of Similarity Selection: A Deep Learning Approach. In *SIGMOD.* 1197–1212.
[31] Yaoshu Wang, Chuan Xiao, Jianbin Qin, Rui Mao, Makoto Onizuka, Wei Wang, Rui Zhang, and Yoshiharu Ishikawa. 2021. Consistent and Flexible Selectivity Estimation for High-Dimensional Data. In *SIGMOD.* 2319–2327.
[32] Xian Wu, Moses Charikar, and Vishnu Natchu. 2018. Local Density Estimation in High Dimensions. In *ICML*, Vol. 80. 5296–5305.
[33] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. *PVLDB* 13, 3 (2019), 279–292.
[34] Songnian Zhang, Suprio Ray, Rongxing Lu, and Yandong Zheng. 2021. SPRIG: A Learned Spatial Index for Range and kNN Queries. In *SSTD.* 96–105.
[35] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation. *PVLDB* 14, 9 (2021), 1489–1502.