DEIM Forum 2022 J24-5

学習型索引を用いた時系列データ検索の高速化

松本　和人† 　肖　　川† 　鬼塚　　真†

† 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

E-mail: †{matsumoto.kazuto,chuanx,onizuka}@ist.osaka-u.ac.jp

あらまし　時系列データの類似検索は経済，医学，薬学，生物学，天文学など幅広い分野で利用されているが，時系列データの類似検索にかかる計算時間はとても大きく，高速化が必要である．大規模データに対する検索を高速化する方法として索引を用いた手法が広く利用されており，近年機械学習ベースの索引である学習型索引が提案されている．しかし，学習型索引を時系列データ検索に応用するのには検索精度が悪いという問題点がある．この問題を解決するため，探索問題を荒い絞り込み部分と詳細な絞り込みの処理に分割し，前者に対して学習型索引を適用する方法を提案する．本手法では，時系列データを事前にクラスタリングすることで，クラスタ内の詳細な絞り込みのための検索は従来型の検索手法を適用し，クラスタを選択する荒い絞り込み処理を分類問題として捉えて索引を学習する．実験では，時系列データの類似検索の処理時間と精度の評価を行う．これにより，学習型索引を用いた時系列データ類似検索を行うことで検索時間が高速化することを示した．

キーワード　時系列データ, learned index, 検索, dynamic time warping

1　はじめに

IoT という言葉が広く使われるようになり，データを観測するセンサーの数は増え続け，観測されるデータ数は爆発的に増加している [1]．そこで観測されるほとんどのデータは時系列データである．センサーで観測されるデータのみでなく，DNA やものの輪郭，手描き文字などを変形して系列データとして扱うこともできる [2]．このような膨大な時系列データから知見を得ることは社会にとってとても有益である．時系列データを活用する方法の一つに類似検索がある．時系列データの類似検索は医学，バイオインフォマティクス，画像処理，音声処理，経済などさまざまな分野で必要とされている．例えば，バイオインフォマティックスでは，Aach と Church が RNA を時系列データとして扱って RNA の発現データを分析している [3]．また，歩行 [4]，署名 [5]，指紋 [6] などの生体データの適合判断に時系列データの類似検索が用いられている．しかし，時系列データの類似検索はとても時間がかかる．二つの時系列データの類似度を測るアルゴリズムに dynamic time warping (DTW) [2] というものがあるが，そのコストがとても大きい．二つの時系列データの比較を何回も行う時系列データの類似検索はデータセットの規模が大きくなると実現不可能なほど時間がかかるため，高速化が必要である．

時系列データの類似検索の高速化をするために主に二つの方法がある．一つ目は DTW の下界を高速で求める lowerbounding 関数を用いる方法 [2,7] である．DTW の値が小さいほど二つの時系列データが類似している．そのため，時系列データの類似検索ではクエリの時系列データと一番 DTW が小さくなるデータを探すのだが，計算コストの大きい DTW を計算するより前に lowerbounding 関数を計算して，その値が暫定で一番小さい DTW より小さい時のみ DTW を計算することで計算時間のかかる DTW を計算する回数を減らすことができる．この方法はデータを探索する順番によっては DTW と lowerbounding 関数を二重で計算する回数が大きくなり効率がよくないことがあるという短所がある．二つ目の方法は B+木などの木構造を用いて索引を作成する方法 [8–11] である．データベースの分野では木構造を用いた従来の索引より機械学習のモデルを用いた学習型索引 (learned index) の方が速度，メモリ効率ともに良いということが言われている [12]．

learned index はデータとそのデータがディスク内に存在しているブロック番号の関係を機械学習のモデルで学習し回帰分析をすることで索引構造を実現している．機械学習のモデルを用いており，索引を用いて探索するときにかかる時間計算量は $O(1)$ を達成している．従来の索引構造は木の高さ分計算が必要なため，探索するときにかかる時間計算量はデータの総数を n とすると $O(\log(n))$ となる．それゆえ learned index が高速であることがわかる． [13] の論文によると learned index を用いたことで一次元データに対する従来の索引構造と比べ 70 パーセント近くの速度向上を実現しているという．しかし，learned index を時系列データの類似検索に利用するにあたって困難な点が 2 つある．一つ目は精度が低い点である．データベースの中で，クエリに最も似たデータを機械学習のモデルのみで見つけると，十分な精度で類似検索できない (詳細は 4.5 を参照のこと) ため，工夫が必要である．二つ目は時系列長の長い時系列データの精度が特に低い点である．多次元データに対応した learned index はあるが，どれも多次元データを一次元化してから learned index を用いるため，時系列データのような超多次元データになりうるデータをそのまま learned index に入力すると，一次元化するときの情報損失が大きくなり，類似検索の精度が悪くなる [12]．

本稿では，learned index を用いて時系列データ検索を高速化するアルゴリズムを提案する．先ほど述べた二つの困難点を解

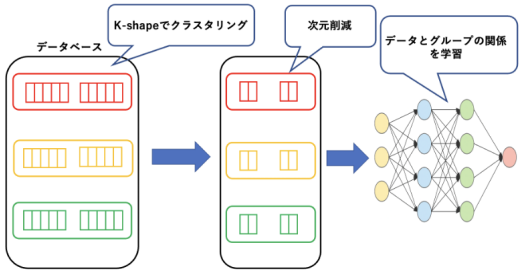

図 1　学習フェーズの概略図

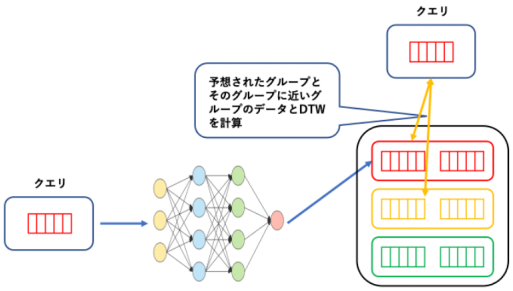

図 2　探索フェーズの概略図

決するために，探索問題を荒い絞り込み部分と詳細な絞り込みの処理に分割することをした．クラスタリングをすることで似たデータ同士でグループを作り，荒い絞り込み処理で learned index を用い，クエリに似たデータが存在するグループを予測する．詳細な絞り込み処理では，荒い絞り込み処理で出力したグループとそのグループに近いグループ内にあるデータのうちから最もクエリに類似したデータを探索する．クラスタリングを用いて探索問題を分割することで，機械学習が解く問題が簡単化し，精度向上を図った．また，時系列データの次元削減を行った．こうすることで，長い時系列長のデータの検索精度向上を図った．

実験では類似検索の結果の精度と探索時間を既存手法と比較する．これにより learned index を用いることでより十分高い精度で探索時間を大幅に短縮できることを確認した．

本稿の構成は以下の通りである．2 章にて事前知識を説明し，3 章にて提案手法について詳細に述べ，4 章にて評価実験の結果を示し，考察を行う．また，5 章では関連研究を紹介する．最後に，6 章で本稿をまとめ，今後の課題について論ずる．

2　前提知識

2.1　Dynamic Time Warping (DTW)

ダイナミックタイムワーピング (DTW) とは，2 つ時系列データの距離を最小化するように時間軸を伸長させる変換処理である．長さ n, N の時系列データ P, Q を考える．すなわち

$$P = p_1, ..., p_n.$$

$$Q = q_1, ..., q_N.$$

P, Q の DTW 距離は以下のように定義される．

$$D_{dtw}(P,Q) = f(n,N),$$

$$f(i,j) = ||p_i - q_j|| + min \begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1), \end{cases}$$

$$f(0,0) = 0, f(i,0) = f(0,j) = \infty$$

,

$$(i = 1, \ldots, n; j = 1, \ldots, N).$$

このように，P の各要素と Q の各要素を昇順にマッチングすることによって DTW 距離は得られる．動的計画法を用いることによって DTW 距離が得られるため，計算コストは $O(nN)$ となり，特に時系列長が長くなるほど大きな計算コストが発生する．

2.2　piecewise aggregate approximation (PAA)

piecewise aggregate approximation (PAA) [8] とは時系列データの次元を縮減する方法である．時系列長 n の時系列データ $P = p_1, ..., p_n$ が与えられたとき，PAA を用いて P を次元 N に変換したデータ C を考える．ただし $(1 \leqq N \leqq n)$ で n は N の倍数とする．C は以下のように定義される．

$$C = c_1, ..., c_N,$$

$$c_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} c_j.$$

すなわち，時系列長を n から N に縮小するために，P を N 個の同じ大きさのセグメントに分割する．セグメント内のデータの平均値が計算され，これらの値のベクトルがデータを縮小した表現になる．

3　提案手法

learned index を用いた時系列データの類似検索の高速化手法を提案する．提案手法では探索精度向上を図るため，探索問題を荒い絞り込み部分と詳細な絞り込み部分に分割することを行った．クラスタリングをすることで似たデータ同士でグループを作り，荒い絞り込み処理でクエリに似たデータが存在するグループを予測する．詳細な絞り込み処理では，荒い絞り込み処理で出力したグループとそのグループに近いグループ内にあるデータのうちから最もクエリに類似したデータを探索する．クラスタリングを用いて探索問題を分割することで，機械学習が解く問題が簡単化し，精度向上を図った．また，時系列データの次元削減を行った．次元削減することで，長い時系列長のデータの精度向上を図った．

提案手法は深層学習の学習をする学習フェーズと実際に探索を行う探索フェーズの二つの段階がある．学習フェーズの概略を図 1 に示す．学習フェーズでは，まず似た時系列データ同士をクラスタリング手法でグループ化する (3.2 節)．次に，時系

列データの次元削減をする (3.3 節)．そして，深層学習を用いて各時系列データが属するグループの関係を学習する (3.4 節)．探索フェーズの概略図を図 2 に示す．探索フェーズでは，深層学習のモデルが出力したグループとそのグループに近いグループ内のデータとクエリとの DTW を計算しその値が最も小さいデータを出力する (3.5 節)．

3.1 問題定義

はじめに，提案手法が扱う問題を定義する．N 個の時系列データが入った時系列データセット

$$C = \{c_1, c_2, ..., c_N\}$$

とクエリ q が与えられたときにクエリ q に最も類似している C の要素を top-k 件見つける．ただしデータセット C の各時系列データの時系列長とクエリ q の時系列長は同じと仮定する [1]．

3.2 時系列データのクラスタリング

探索問題を荒い絞り込み処理と詳細な絞り込み処理とに分割するために，データセット C の時系列データを似たデータ同士でクラスタリングする．時系列データをクラスタリングするアルゴリズムに K-shape [14] を用いる．K-shape は，軸に対して縮小または拡大した際にデータ同士の性質が似ているかどうかを表す指標である scaling と，時間軸に対してずらしたときにデータ同士が似ているかに着目した指標である shifting の二つの指標を鑑みて，データ同士が似ているのかを決定する．そのため，時系列データの位相が多少ずれていても形状が似ていれば似ているデータとしてクラスタリングする shape-based クラスタリングであると言える．DTW も位相が少しずれていても，形が似ていたら似ているデータと判断するので，DTW と相性が良いと考えたため，クラスタリングアルゴリズムに K-shape を採用した．K-shape はクラスタ数 k を事前に決めなければならないため，提案手法では，データセット C の中からランダムに取ってきた 100 個のデータを k を 3 から 10 まで増やして K-shape する．シルエットスコア [15] が一番小さい k をクラスタ数とすることで適切なクラスタ数を決定している．決定したクラスタ数 k を用いて全データで K-shape を行い，グループ番号の配列 G を求める．

$$G = g_1, g_2, .., g_N,$$

但し, g_iは c_iのグループ番号を表す.

3.3 時系列データの次元削減

時系列長の長い時系列データでも高速に，精度よく探索するために時系列データの次元削減を行う．また，時系列長がとても長い時系列データをそのまま深層学習のモデルに入力すると深層学習の学習時間がとてもかかってしまうため，深層学習の学習時間短縮にもつながる．精度よく時系列データの次元削減を行うために，piecewise aggregate approximation (2.2 節) という方法でデータセットの各データの次元削減を行う．すなわちデータセット C の PAA 表現 A を求める．

$$A = a_1, a_2, ..., a_N, (a_i\text{は } c_i\text{の } PAA \text{ 表現}).$$

3.4 深層学習

深層学習技術が learned index の設計に広く使われており，他のモデルより速度，精度ともに良いということが言われている [12]．クエリが属するグループの判定を高速化するため，深層学習を用いてグループの予測をする．具体的には，深層学習のモデルでデータセットのデータとそのデータが属するグループ番号の関係を学習する．すなわち，元のデータセット C を PAA を用いて次元削減した A を入力とし，G を出力として深層学習のモデルを学習する．

3.5 探索

深層学習のモデルが出力したグループ内のデータを探索することで最もクエリに似た top-k 件のデータを求める．クエリ q を学習された深層学習のモデルに入力すると，クエリ q に似たデータが属すると予測されたグループ番号を深層学習のモデルが出力する．そのグループとそのグループに近いグループ内にあるデータとクエリ q で DTW を計算し，最も DTW 距離が小さい上位 n 件を出力する

Algorithm 1 索引構築フェーズ

Input: C
Output: $model$

```
1: min_silhouette_score ← ∞
2: for i ← 3 to 10 do
3:     C' ← C からランダムに 100 個データを選択
4:     G' ← kshape(C', i)
5:     tmp_silhouette_score ← calc_silhouette_score(G')
6:     if min_silhouette_score > tmp_silhouette_score then
7:         min_silhouette_score = tmp_silhouette_score
8:         k = i
9:     end if
10: end for
11: G ← kshape(C, k)
12: A ← PAA(C, n)     n は次元削減後の次元数
13: model ← NeuralNetwork(A, G)
14: return model
```

3.6 アルゴリズム

索引構築フェーズのアルゴリズムを Algorithm 1 に示し，探索フェーズのアルゴリズムを Algorithm 2 に示す．索引構築フェーズでは，始めにデータをランダムにサンプリングした後にシルエットスコアを用いて，高速にクラスタ数 k を決める．(2-10 行目)．そして，得られた k をクラスタ数として，C を K-shape を用いてクラスタリングし，結果を G に入れる (11 行目)．つぎに，データベース C を PAA を用いて次元 n に次元削減し，結果を A に入れる (12 行目)．そして，入力データを A，正解ラベルを G として NeuralNetwork で学習する (13 行

1：時系列長が異なる場合は今後の課題である

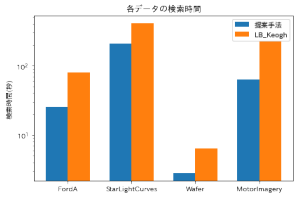

図 3　検索時間の比較

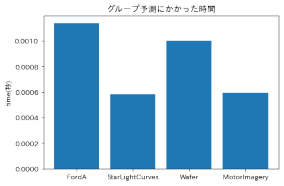

図 4　荒い絞り込みの所要時間

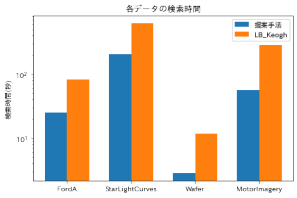

図 5　top-3 の検索時間の比較

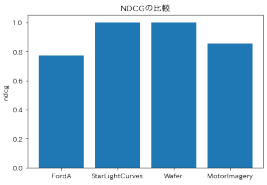

図 6　各データセットの NDCG

Algorithm 2 探索フェーズ

Input: $C, q, model$

Output: $answer$

1: $predicted\_group \leftarrow model(q)$
2: $min\_dtw \leftarrow \infty$
3: **foreach** $data \in$ predicted_group 内にある全データ **do**
4: 　$heap\_data = []$
5: 　$tmp\_dtw \leftarrow DTW(data, q)$
6: 　$heappush(heap\_data, (tmp\_dtw, data))$
7: **end for**
8: $answer \leftarrow heap\_data$ の上位 n 件のデータ
9: **return** $answer$

目)．探索フェーズでは，まず，学習したモデルにクエリを入力し，モデルから予測されたグループ番号を predicted_group に代入する (1 行目)．predicted_group に属する全データのうちで最も DTW 距離が小さい上位 n 件のデータを探す (2-9 行目)．

4　実　　験

本章では 4 つの実際のデータセットを用いて時系列データの類似検索を行うことで提案手法の探索精度と探索速度を検証する．類似している上位一つのデータを出力する top-1 と，上位三つのデータを出力する top-3 の二つの実験を行った．また，クラスタリングを用いずに learned index のみで top-1 を行う実験も行った．

4.1　実験設定

4.1.1　データセット

本実験で用いる 4 つのデータセットについての説明を以下に示す．4 つのデータセットは全て UEA & UCR Time Series Classification Repository[2]で公開されているものを使用する．各データセットの統計情報は表 1 に記す．

- FordA：自動車のサブシステムに問題があるか調べるために，エンジン音を 500 回計測した時系列データセット．
- StarLightCurves：天体の明るさを時間の関数として表した時系列データセット．
- Wafer：半導体製造用のシリコンウェーハの加工中に様々なセンサーから記録された時系列データセット．
- MotorImagery：運動野の右側に設置された 8 × 8 の ECoG 白金電極グリッドから得られた脳波．記録はすべてサンプリングレート 1000Hz で行われた．

2：https://www.timeseriesclassification.com/dataset.php

表 1　データセット

データセット名	データ数	時系列長
FordA	4921	500
StarLightCurves	9236	1024
Wafer	7164	152
MotorImagery	378	3000

表 2　比較手法に対する検索時間短縮率の比較

データセット名	短縮率 (%)
FordA	67.8
StarLightCurves	49.9
Wafer	56.1
MotorImagery	71.3
平均	61.3

4.1.2　ニューラルネットワーク

今回は，ドロップアウト，全結合層 2 層で構成されたモデルを使用した．一層目のノード数は時系列データを PAA で次元削減した次元数とし，二層目のノード数は 8 とした．ドロップアウト層のドロップアウト率を 0.2 とし，正則化パラメーターを 0.01 として l2 正則を行った．バッチサイズを 10，エポック数を 300 とした．

4.1.3　比 較 手 法

我々は提案手法の類似検索精度を以下の手法と比較する．

- LB_Keogh：lowerbounding 関数を用いて DTW の計算回数を減らすことで高速に類似検索する手法．lowerbounding 関数である LB_Yi や LB_Kim よりもより厳密に DTW を近似するため，より多く DTW の計算回数を削減できる．

4.2　top-1 検索

提案手法と比較手法それぞれで，各データセット 10 回ずつ類似検索を行い，検索時間と精度を調べた．

4.2.1　検 索 時 間

top-1 の平均検索時間のグラフを図 3 に示す．全てのデータセットで提案手法の方が比較手法より検索時間が短いことがわかる．top-1 の提案手法の検索時間のうち，モデルの出力を求める荒い絞り込みにかかった時間のグラフを図 4 に示す．提案手法の検索時間に対する荒い絞り込みにかかった時間の割合は平均約 0.01%ほどであり，詳細な絞り込みにかかった時間に比べると無視できるほど小さい．提案手法の所要時間が比較手法のものより何パーセント短縮したかを表 2 に示す．平均で提案手法の所要時間が比較手法のより約 61%短縮しており，提案手法が時系列データの類似検索の高速化に有用であることがわ

表 3　正 答 回 数

データセット名	正答回数
FordA	8/10
StarLightCurves	10/10
Wafer	10/10
MotorImagery	7/10

かる．

4.2.2　精　　度

各データセットで，提案手法で類似検索を 10 回行って，何回正答したかを表 3 に示す．二つのデータセットでは正答回数が 10 回であるが，FordA では 8 回，MotorImagery では 7 回であり，提案手法の精度がとても高いとは言えない．しかし，FordA の提案手法が出した答えと，正答との DTW 距離の比の平均は約 99%になっている．また，MotorImagery の提案手法が出した誤答 3 つ全て，全データの中で二番目にクエリに似ている時系列データであった．このことより，提案手法の出力は正答を外すことがあるが，的外れな誤答はせず，正答と似た，クエリに十分類似した答えを出力すると言える．

4.3　top-3 検索

提案手法と比較手法それぞれで，各データセット 10 回ずつ類似検索を行い，その検索時間と精度を調べた．提案手法は深層学習が出力したグループ内で最も DTW の値が小さい 3 つのデータを出力することで top-3 を実現している．

4.3.1　検 索 時 間

top-3 の平均検索時間のグラフを図 5 に示す．全てのデータセットで提案手法の方が比較手法より検索時間が短いことがわかる．提案手法の検索時間が比較手法のものより何パーセント短縮したかを表 4 に示す．平均で提案手法の検索時間が比較手法のよりも約 73%短縮しており，top-1 より比較手法の検索時間に対する提案手法の検索時間の短縮率が大きくなり，提案手法がより高速化できていることがわかる．

表 4　top-3 の比較手法に対する検索時間短縮率の比較

データセット名	短縮率 (%)
FordA	69.9
StarLightCurves	67.0
Wafer	76.0
MotorImagery	80.0
平均	73.2

4.3.2　精　　度

ランキング予測評価の指標として NDCG [16, 17] を用いて評価する．各データセットの平均の NDCG の値のグラフを図 6 に示す．StarLightCurves と Wafer はそれぞれ平均 NDCG の値が 1 であり，10 回の実験全てで top-3 を正答している．4 つのデータセットの NDCG の平均は約 0.91 であり，十分高い精度であると言える．

4.4　学 習 時 間

深層学習の平均学習時間のグラフを図 7 に示す．データ数に

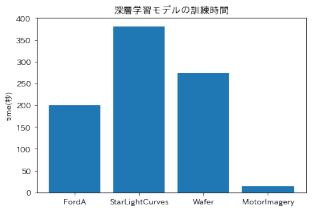

図 7　学習時間の比較

ほぼ比例して学習時間がかかっている．データ数が約 10000 件だと学習時間が約 5 分ほどかかってしまい，短いとはいえない．現在深層学習のエポック数が 300 であるのでその値を小さくすると学習時間が減少すると考える．

4.5　クラスタリングなし top-1

深層学習のみを用いて top-1 の探索問題を 10 回解いた時の精度を測った．すなわち，データとデータ番号の関係を深層学習で学習し，学習した深層学習のモデルにクエリを入力して，モデルが出力した値がデータ番号であるデータを出力する．その出力がデータベース内のデータで何番目にクエリに類似しているかの平均を表 5 に示す．FordA と MotorImagery の深層学習のモデルの出力が何番目にクエリに似ているかの平均の値は全データの上位半分に入っていない．また，StarLightCurves と Wafer も探索精度はとても悪く，深層学習のみを用いて探索問題を解くと探索精度が悪く実用的ではない．このことから，時系列データの検索を深層学習を用いて高速化するには，ただ用いるだけでは十分な精度が得られず，クラスタリングや次元削減などの工夫が必要であることがわかる．

表 5　クラスタリングなし top-1

データセット名	平均順値
FordA	2559.7/4921
StarLightCurves	2068.6/9236
Wafer	1539.9/7164
MotorImagery	206.7/378

5　関 連 手 法

二つの時系列データの類似度を計算する DTW というアルゴリズムは時間計算量が $O(n^2)$ で計算コストが大きい (ただし時系列長を n とする)．DTW の値が小さいほど二つの時系列データが似ているため，類似検索のクエリと検索対象のデータセットの全データとで DTW をし，DTW の値が一番小さいデータを探すことで，クエリと一番類似しているデータを見つけることができる．しかし，データセットのデータ数が大きくなったり，時系列長の長さが大きくなると計算時間が実行不可能なほどかかる．DTW の計算回数を減らして高速化するために，主に DTW の下界を DTW より高速で求める lowerbounding 関数用いる方法と，索引を用いる方法がある．

5.1　lowerbounding 関数を用いた手法

lowerbounding 関数は，時系列長を n とすると時間計算量

$O(n)$ で DTW の下界を求める関数である．クエリとデータセットの各データの DTW を計算する前に lowerbounding 関数を計算し，lowerbounding 関数が返す値が暫定で一番小さい DTW の値（best_so_far と呼ぶ）より大きいときは，そのデータとクエリとの DTW の値は best_so_far より大きくなり，DTW を計算せずとも，最も類似していることはないことがわかる．このことを利用して DTW を計算する回数を減らすことができる．lowerbounding 関数には LB_Kim，LB_Yi，LB_Keogh などがある [8, 18, 19]．LB_Kim は二つの時系列データの二つの端点とそれぞれの最大値，最小値の点の値の差の和を返す．LB_Yi は一つの時系列データの最小値と，それより小さい全ての点との値の差の和と，最大値とそれより大きい全ての点との値の差の和の和を返す，LB_Keogh は LB_Yi を動的に応用したもので，LB_Yi より精度の良い DTW の下界の値を返す．

5.2 索引を用いた手法

索引を用いた手法は B+木 [20] や R 木 [21] などの索引構造を時系列データに応用することで時系列データの類似検索を高速化する [8, 22]．B+木や R 木など既存の索引構造よりも高速でメモリ効率も良いものに learned index [12, 13] というものがある．

learned index は既存の索引構造を機械学習のモデルに置き換えたものである．学習フェーズでは，データとそのデータがディスク内に存在しているブロック番号の関係を機械学習のモデルで学習する．既存の索引構造は探索時に索引構造の木の高さ分計算するため索引内に存在するデータ数を n とすると探索の時間計算量は $O(\log(n))$ となるが，学習されたモデルが出力するのにかかる時間計算量は $O(1)$ なので learned index で探索するときにかかる時間計算量は $O(1)$ となり，learned index が既存の索引構造よりも高速であることがわかる．多次元データに対応した learned index [23] は存在するが，多次元データを一次元化するため，長い時系列長の時系列データのような超多次元データに learned index を適用すると，一次元化した時の情報損失が大きくなり，精度が悪くなるという問題がある．

6 まとめ

本稿では，学習型索引を用いた時系列データの類似検索の高速化手法を提案した．提案手法では，探索問題を荒い絞り込み部分と詳細な絞り込みの処理に分割し，前者に対して学習型索引を適用することで，精度を保ちつつ検索速度の向上を図った．これにより，提案手法が十分な精度を保ちつつ既存手法より検索時間を高速化することが可能であることを確認した．今後の展望としては，より効果的な設計および学習を行うことで類似検索をさらに高速化していきたい．データセットをクラスタリングする際のクラスタ数を増やすことで詳細な絞り込み処理の回数を減らし高速化できるが，荒い絞り込み処理の精度が落ちることが予想される．そのため，深層学習の学習精度を上げるなどして，精度を保ったままできるだけ早く類似検索ができるよう提案手法を改良していきたい．

謝　辞

本研究は日本学術振興会科研費 19K11979 の支援によって行われた．

文　献

[1] Keiichi Yasumoto, Hirozumi Yamaguchi, and Hiroshi Shigeno. Survey of real-time processing technologies of IoT data streams. *Journal of Information Processing*, Vol. 24, pp. 195–202, 2016.

[2] Abdullah Mueen and Eamonn J. Keogh. Extracting optimal performance from dynamic time warping. In *KDD*, pp. 2129–2130, 2016.

[3] John Aach and George Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, Vol. 17, pp. 495–508, 2001.

[4] D.M. Gavrila and L.S. Davis. 3-D model-based tracking of humans in action: a multi-view approach. In *CVPR*, pp. 73–80, 1996.

[5] M.E. Munich and P. Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *ICCV*, pp. 108–115, 1999.

[6] Zsolt Kovács-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 1266 – 1276, 2000.

[7] Hailin Li and Libin Yang. Extensions and relationships of some existing lower-bound functions for dynamic time warping. *Journal of Intelligent Information Systems*, Vol. 43, No. 1, pp. 59–79, 2014.

[8] Eamonn Keogh and Chotirat Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, Vol. 7, pp. 358–386, 2005.

[9] 櫻井保志, 吉川正俊. ダイナミックタイムワーピングのための類似探索手法. 情報処理学会論文誌データベース（TOD）, Vol. 45, No. SIG04(TOD21), pp. 23–36, 2004.

[10] Zhengxin Li. Exact indexing of time series under dynamic time warping. *CoRR*, Vol. abs/2002.04187, , 2020.

[11] Yunyue Zhu and Dennis E. Shasha. Warping indexes with envelope transforms for query by humming. In *SIGMOD*, pp. 181–192, 2003.

[12] Abdullah Al-Mamun, Hao Wu, and Walid G. Aref. A tutorial on learned multi-dimensional indexes. In *SIGSPATIAL*, pp. 1–4, 2020.

[13] Ani Kristo, Kapil Vaidya, Ugur Çetintemel, Sanchit Misra, and Tim Kraska. The case for a learned sorting algorithm. In *SIGMOD*, pp. 1001–1016, 2020.

[14] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*, pp. 1855–1870, 2015.

[15] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, Vol. 20, pp. 53–65, 1987.

[16] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *ICML*, pp. 89–96, 2005.

[17] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, Vol. 20, No. 4, pp. 422–446, 2002.

[18] Sang-Wook Kim, Sanghyun Park, and W.W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, pp. 607–614,

2001.

[19] Byoung-Kee Yi, H.V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pp. 201–208, 1998.

[20] Rudolf Bayer and Edward M. McCreight. Organization and maintenance of large ordered indices. *Acta Informatica*, Vol. 1, pp. 173–189, 1972.

[21] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pp. 47–57, 1984.

[22] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pp. 419–429, 1994.

[23] Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H. Chi, Ani Kristo, Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. SageDB: A learned database system. In *CIDR*, 2019.