

ストリーミング時系列データの 効率的なモチーフモニタリングアルゴリズム

加藤 慎也¹ 天方 大地¹ 西尾 俊哉¹ 原 隆浩¹

概要：近年，多くの IoT 機器はストリーミング時系列データを生成しており，それらを環境モニタリングやイベント検知に応用することが考えられる．これを実現する一つの技術として，時系列データの中に最も多く現れるサブシーケンスであるレンジモチーフの発見が注目を集めている．本稿では，カウントベースのスライディングウィンドウ上でストリーミング時系列データのレンジモチーフをモニタリングする問題に取り組む．ウィンドウがスライドした際，新たにサブシーケンスが生成され，最も古いサブシーケンスが削除される．生成および削除されるサブシーケンスとウィンドウ内の全てのサブシーケンスを比較する単純な方法は，多くの類似度の計算を必要とするため効率的でない．そのため，効率的にモチーフを更新するアルゴリズム SRMM を提案する．SRMM の時間計算量は生成および削除されるサブシーケンスと類似するサブシーケンスの数にのみ依存し，効率的にレンジモチーフをモニタリングできる．4 つの実データを用いた実験により，SRMM の有効性を確認した．

1. 序論

モチーフ発見は時系列データを分析する最も重要な技術の一つである [1]．ある時系列データ t が与えられたとき， t のレンジモチーフとは， t の中で最も多く現れるサブシーケンスである [2], [3]．つまり，レンジモチーフは頻繁に発生するサブシーケンスを表す．例えば図 1 は，温室効果ガスの排出量のストリーミング時系列データ [4] の中で繰り返し現れているサブシーケンス（赤いサブシーケンス）を図示しており，最も左の赤いサブシーケンスが現在のレンジモチーフである．（本稿では，サブシーケンス間の類似度を $\approx$ 正規化ユークリッド距離を用いて計算するため，この図における値のスケールは問題ではない．）近年，多くの IoT 機器はストリーミング時系列データを生成するため [5]，本稿では，ストリーミング時系列データのレンジモチーフをモニタリングする問題に取り組む．今後，特に明記する必要がない場合，レンジモチーフを単にモチーフと呼ぶ．

アプリケーション例． センサ機器が定期的にデータを収集し，サーバに送信すると仮定する．さらに，専門家が時系列データをモニタリングすると仮定する．このデータをモニタリングし，時間の経過とともにモチーフが変化した場

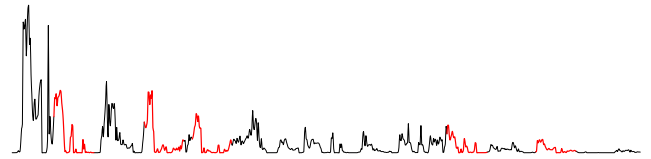


図 1: レンジモチーフの例

合，様々な潜在的な事象を分析したり，センサデータは環境的な要因だけでなく，時間的な要因と相関があるといった仮説を立てることができる．また，イベント検知への応用も考えられる．モチーフをモニタリングし，そのモチーフを 1 分ごとに保存すると仮定する．ある時刻におけるモチーフが前日の同時刻に得られたモチーフと大きく異なる場合やそれ以前のモチーフと著しく異なる場合，何らかの異常が発生していることが予測できる．

提案アルゴリズムの概要． 上記のようなアプリケーションでは，最新のデータのみを考慮し，それらのモチーフをリアルタイムにモニタリングする必要がある．そのため，カウントベースのスライディングウィンドウを用いて最新の w 個のデータのみを考慮し，それらのモチーフを効率的にモニタリングするアルゴリズム SRMM (Streaming Range Motif Monitoring) を提案する．ウィンドウがスライドした際，新たなデータがウィンドウに挿入され，最も古いデータがウィンドウから削除される．つまり，新たなデータを含むサブシーケンス s_n が生成され，最も古いデータを含むサブシーケンス s_e が削除される．このとき，モチー

¹ 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology,
Osaka University

フを更新する単純な手法として、 s_n および s_e とウィンドウ内の全てのサブシーケンスを比較する手法が考えられる。この手法は、 s_n および s_e と類似するサブシーケンスの数（類似サブシーケンス数）を正確に取得できるが、多大な計算コストがかかる。そこで、SRMM は、ウィンドウがスライドした際、モチーフになり得るサブシーケンスにのみ注目することにより、不要な計算を削減する。SRMM は PAA(Piecewise Aggregate Approximation)[6] および kd 木 [7] を用いて、 s_n の類似サブシーケンス数の上界値を高速に計算し、正確な類似サブシーケンス数を計算する回数を削減する。ここで、 s_n の類似サブシーケンス数の上界値は、 s_n と類似するサブシーケンスの候補の数である。そのため、正確な類似サブシーケンス数を計算する場合においても、全てのサブシーケンスとの比較を行うことなく正確な類似サブシーケンスの数を計算できる。

貢献. 以下に本研究の貢献を示す。

- カウントベースのスライディングウィンドウ上でストリーミング時系列データのレンジモチーフをモニタリングする問題に取り組む。筆者らの知る限り、この問題はこれまでに組み込まれていない。
- ウィンドウがスライドした際、モチーフを効率的に更新するアルゴリズム SRMM を提案する。SRMM は効率的にモチーフを更新でき、時間計算量は $O(\log(wl) + m_n + m_e)$ である。ここで、 l はモチーフ長、 m_n および m_e は新たに生成されるサブシーケンスおよび削除されるサブシーケンスの類似サブシーケンス数の上界値である。
- 4 つの実データを用いた実験により、SRMM の有効性を確認する。

本稿の構成. 2 章で本稿の問題を定義し、3 章で関連研究について述べる。4 章で SRMM について説明し、5 章で実データを用いた実験の結果を示す。最後に 6 章で本稿のまとめと今後の課題について述べる。

2. 予備知識

2.1 定義

ストリーミング時系列データ t は実数値の系列であり、 $t = (t[1], t[2], \dots)$ と表現する。まず、 t の中に繰り返し現れる特徴的なパターンを発見するため、 t の一部を表すサブシーケンスを定義する。

定義 1 (サブシーケンス). t および長さ l が与えられたとき、 p を始点とするサブシーケンス s_p は式 (1) により定義される。

$$s_p = (t[p], t[p+1], \dots, t[p+l-1]) \quad (1)$$

ここで、 s_p の x 番目の値を $s_p[x]$ と表現する。つまり、

$s_p = (s_p[1], s_p[2], \dots, s_p[l])$ である。次に、 t の中で s_p と類似したサブシーケンスの数を計算するため、本研究では、時系列データ間の類似度を測る基本的な指標であるピアソン相関を用いる [8], [9]。

定義 2 (ピアソン相関). 長さ l の 2 つのサブシーケンス s_p および s_q が与えられたとき、これらのピアソン相関 $\rho(s_p, s_q)$ は式 (2) により定義される。

$$\rho(s_p, s_q) = 1 - \frac{\|\hat{s}_p, \hat{s}_q\|^2}{2l} \quad (2)$$

$\rho(s_p, s_q) \in [-1, 1]$ であり、 $\|\hat{s}_p, \hat{s}_q\|$ は $\hat{s}_p$ と $\hat{s}_q$ 間のユークリッド距離を計算したものである。また、 $\hat{s}_p$ は s_p を z 正規化したものであり、

$$\hat{s}_p[i] = \frac{s_p[i] - \mu(s_p)}{\sigma(s_p)}$$

である。ここで、 $\mu(s_p)$ および $\sigma(s_p)$ はそれぞれ $(s_p[1], s_p[2], \dots, s_p[l])$ の平均および標準偏差である。ピアソン相関は z 正規化ユークリッド距離 $d(\cdot, \cdot) = \|\cdot, \cdot\|$ に変換でき、式 (3) で与えられる。

$$d(\hat{s}_p, \hat{s}_q) = \sqrt{2l(1 - \rho(s_p, s_q))} \quad (3)$$

ピアソン相関の時間計算量は $O(l)$ である。次に、サブシーケンス s_p と類似するサブシーケンス（類似サブシーケンス）を定義する。

定義 3 (類似サブシーケンス). s_p , s_q , およびある閾値 θ が与えられたとき、 $s_p(s_q)$ が $s_q(s_p)$ と類似しているならば、次の条件を満たす。

$$\rho(s_p, s_q) \geq \theta \Leftrightarrow d(\hat{s}_p, \hat{s}_q) \leq \sqrt{2l(1 - \theta)} \quad (4)$$

s_p と s_{p+1} が互いに類似していることは自明であり、有用な結果を得るためには、このようなサブシーケンスを考慮すべきではない。そこで、互いに重なり合うサブシーケンスをトリビアルマッチと定義する [3], [10]。

定義 4 (トリビアルマッチ). s_p が与えられたとき、 s_p とトリビアルマッチであるサブシーケンスの集合 S_p は次の条件を満たす。

$$S_p = \{s_q | p-l+1 \leq q \leq p+l-1\} \quad (5)$$

次に、あるサブシーケンス s_p と類似したサブシーケンスの数をスコアと定義する。

定義 5 (スコア). t , l , および θ が与えられたとき、あるサブシーケンス $s_p \in t$ のスコアは式 (6) により定義される。

$$\text{score}(s_p) = |\{s_q | s_q \in t, \rho(s_p, s_q) \geq \theta, s_q \notin \overline{S_p}\}| \quad (6)$$

ここで、1 章で述べたアプリケーションを含む多くの

アプリケーションでは最新のデータのみを考慮している [11], [12]. そのため, ストリーミング時系列データに関する既存の研究 [13], [14] と同様に, 本研究においてもカウンタベースのスライディングウィンドウを用いて, 最新の w 個の値のみをモニタリングする. つまり, ウィンドウ内のストリーミング時系列データ t は $t = (t[i], t[i+1], \dots, t[i+w-1])$ のように表され, $t[i+w-1]$ が最新の値である. また, l が与えられたとき, ウィンドウ内には $w-l+1$ 個のサブシーケンスが含まれる. ウィンドウがスライドしたとき, 最新の l 個の値を含む新たなサブシーケンスが生成される. 同時に, 最も古い値がウィンドウから削除されるため, 最も古いサブシーケンスが削除される. 本研究では, このような環境においてスコアが最大となるサブシーケンスをモニタリングする. つまり, ウィンドウサイズ w のウィンドウに含まれる全てのサブシーケンスの集合を S としたとき, 本研究の問題は以下のように定義される.

問題定義. t, l, θ , および w が与えられたとき, 式 (7) で表されるレンジモチーフ s^* をモニタリングする.

$$s^* = \arg \max_{s \in S} \text{score}(s) \quad (7)$$

2.2 ベースラインアルゴリズム

本稿は, 本問題に初めて取り組むため, まず, ベースラインとなるアルゴリズムについて考える. ウィンドウがスライドした際, 新たに生成されるサブシーケンスおよび削除されるサブシーケンスに対して, ウィンドウ内の全てのサブシーケンスとのピアソン相関を計算し, スコアを更新する. そして, ウィンドウ内のスコアが最大のサブシーケンスをモチーフとする. 前述したように, ウィンドウ内には $w-l+1$ 個のサブシーケンスが含まれ, ピアソン相関の計算には $O(l)$ の時間計算量がかかる. そのため, ベースラインアルゴリズムの時間計算量は $O((w-l)l)$ である.

ここで, あるサブシーケンスのスコアに影響を与えるのは, そのサブシーケンスと類似したサブシーケンスのみであるため, 全てのサブシーケンスのスコアを更新する必要はない. そのため, ウィンドウがスライドした際, スコアを更新する必要があるサブシーケンスを効率的に特定するアルゴリズムを提案する.

3. 関連研究

本章では, モチーフ発見に関する既存研究について紹介する. モチーフという用語は, 文献によって 2 つの異なる意味で用いられている [2]. 1 つ目のモチーフの定義は, 本稿におけるモチーフの定義と同様である. 一方, 文献 [8], [9], [12] では, 時系列データの中で最も類似するサブシーケンスのペアをモチーフと定義している. 本章では, 最も類似するサブシーケンスのペアを発見する問題をペアモチーフ発見と呼ぶ.

3.1 ペアモチーフ発見

ペアモチーフ発見はサブシーケンスの数の二乗の時間計算量がかかるため, 文献 [9] では, 三角不等式を用いて実践的な計算時間を削減するアルゴリズム MK を提案している. MK は, いくつかのサブシーケンスを基準点とし, あるサブシーケンスのペアが与えられたときに, それらの距離の上界値を取得する. しかし, 時間計算量は依然としてサブシーケンスの数の二乗の時間計算量がかかる. より性能を向上させるため, 文献 [8] では, Quick-Motif と呼ばれるアルゴリズムを提案している. Quick-Motif はオンラインでサブシーケンスのインデックスを作成することで, サブシーケンス同士の比較回数を削減する. 評価実験では, Quick-Motif は MK よりも優れていることを示している. 文献 [15], [16] では, Matrix Profile と呼ばれるオフラインでインデックスを作成するアプローチを提案している. このインデックスは, 全てのサブシーケンスに対してそのサブシーケンスと最も類似するサブシーケンスとの距離を保持する. このインデックスを用いることでペアモチーフを高速に発見できる.

これらの研究は, 静的な時系列データを対象としている. 一方, 文献 [12] では, ペアモチーフをモニタリングする問題に初めて取り組んでいる. 文献 [12] の提案アルゴリズムでは, ペアモチーフを高速に更新するため, 各サブシーケンスは最近傍と逆最近傍のサブシーケンスのリストを保持する. また, 文献 [11] では, ペアモチーフをモニタリングするためにデータ構造を最適化したアルゴリズムを提案しており, [12] のアルゴリズムより優れた性能を示している.

3.2 レンジモチーフ発見

Patel らはレンジモチーフを効率的に発見するための近似アルゴリズムを提案している [3]. このアルゴリズムでは, 各サブシーケンスを SAX[17] を用いて記号列に変換する. このアルゴリズムと同様に, Castro と Azevedo は iSAX[18] を用いてレンジモチーフを発見するアルゴリズムを提案している [19]. SAX および iSAX は時系列データを記号列に近似するため, 発見されたモチーフが正確であることが保証されない. また, いくつかの確率的アルゴリズムが提案されているが [1], [10], これらのアルゴリズムも発見されたモチーフが正確であることは保証されない. 文献 [2] では, 学習ベースのモチーフ発見アルゴリズムを提案している. しかし, このアルゴリズムは前処理を行う必要があるため, ストリーミング時系列データに適用できない. また, これらの研究は, 静的な時系列データを対象としている.

文献 [20] では, ストリーミング時系列データを対象としているが, 短期間ではモチーフが出現しない場合に取り組んでいる. さらに, 文献 [20] で提案されたアルゴリズムも近似手法 (SAX および Bloom filter) を用いている. 文

献 [21] も、ストリーミング時系列データを対象としているが、時系列データを SAX で記号化したサブシーケンス間の距離に基づいてモチーフをモニタリングする。このように、既存の研究では近似的なモチーフをモニタリングする問題に取り組んでいる。本稿では、正確なモチーフをモニタリングするための効率的なアルゴリズムを提案する。

4. SRMM: Streaming Range Motif Monitoring

ウィンドウがスライドした際、ウィンドウ内の各サブシーケンスのスコアは最大で 1 増加する。これは、定義 5 およびカウントベースのスライディングウィンドウの特性から明らかである。そのため、モチーフは頻繁に変化せず、新たに生成されるサブシーケンスのスコアが頻繁に $score(s^*)$ を超えることは非常にまれである。

ここで、新たに生成されるサブシーケンスを s_n としたとき、高速に $score(s_n) < score(s^*)$ であることが分かれば、正確なモチーフを効率的にモニタリングできる。これを実現するため、4.1 節において $score(s_n)$ の上界値を効率的に取得し、不要なスコアの計算を枝刈りするアルゴリズムを提案する。また、ウィンドウがスライドした際、ウィンドウからサブシーケンスが削除され、そのサブシーケンスと類似したサブシーケンスのスコアが 1 だけ減少するため、 s^* が変化する可能性がある。そこで、4.2 節において、スコアが減少する可能性のあるサブシーケンスを効率的に特定するアルゴリズムを紹介する。最後に、4.3 節において SRMM の全体的なアルゴリズムを紹介し、SRMM の時間計算量について述べる。

4.1 スコアの上界値の取得

まず、 s_n と $s \in S$ のピアソン相関の上界値、つまり z 正規化ユークリッド距離の下界値を取得するため、次元削減アルゴリズム PAA[6] を用いる。このとき、長さ l のサブシーケンス $s_p = (s_p[1], s_p[2], \dots, s_p[l])$ は l 次元上の点として表現できる。

ある次元 $\phi < l$ が与えられたとき、PAA により l 次元上の点を ϕ 次元上の点に変換できる。このとき、 $\hat{s}_p$ を ϕ 次元上の点に変換したものを $\hat{s}_p^\phi$ とすると、 $\hat{s}_p^\phi$ の各値は以下の式で表される。

$$\hat{s}_p^\phi[i] = \frac{\phi}{l} \sum_{j=\frac{l}{\phi}i}^{\frac{l}{\phi}(i+1)-1} \hat{s}_p[j]$$

また、PAA では以下の補題が成り立つ [6]。

補題 1. 2 つのサブシーケンス $\hat{s}_p$ および $\hat{s}_q$ が与えられたとき、以下の関係が成り立つ。

$$\sqrt{\frac{l}{\phi}} \text{dist}(\hat{s}_p^\phi, \hat{s}_q^\phi) \leq \text{dist}(\hat{s}_p, \hat{s}_q) \quad (8)$$

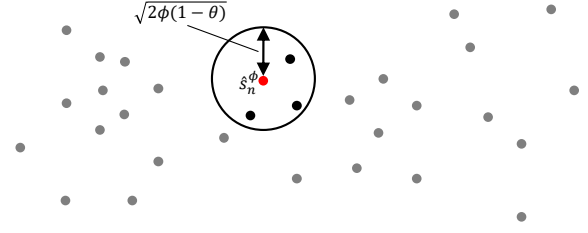


図 2: $score(s_n)$ の上界値を求める例

PAA により、 $\hat{s}_p$ と $\hat{s}_q$ の z 正規化ユークリッド距離の下界値、つまり $\rho(s_p, s_q)$ の上界値を $O(\phi)$ で得ることができる。 $\sqrt{\frac{l}{\phi}} \text{dist}(\hat{s}_p^\phi, \hat{s}_q^\phi) > \sqrt{2l(1-\theta)}$ ならば、定義 3 より s_q は s_p と類似しないため、正確性を失うことなく s_p と s_q の正確な距離の計算を枝刈りできる。 $\hat{s}_n$ が与えられたとき、 $\forall s_p \in S \setminus \bar{S}_n$ に対して、 $\sqrt{\frac{l}{\phi}} \text{dist}(\hat{s}_n^\phi, \hat{s}_p^\phi)$ を計算することで、 $\hat{s}_n$ のスコアの上界値を取得できるが、これには $O(\phi(w-l))$ の時間計算量がかかる。ここで、 s_n のスコアの上界値を求めるためには、 $\sqrt{\frac{l}{\phi}} \text{dist}(\hat{s}_n^\phi, \hat{s}_p^\phi) \leq \sqrt{2l(1-\theta)}$ を満たす s_p のみに注目すればよい。そこで、このような s_p を効率的に取得するために kd 木 [7] を用いる。kd 木は k 次元上の点を二分木で管理するデータ構造であり、データの削除、挿入および範囲検索を効率的に実行できる。

ウィンドウ内の PAA により変換されたサブシーケンスを kd 木で管理した場合、 $\sqrt{\frac{l}{\phi}} \text{dist}(\hat{s}_n^\phi, \hat{s}_p^\phi) \leq \sqrt{2l(1-\theta)}$ を満たす s_p は、 $\hat{s}_n^\phi$ を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索により取得できる。このとき、以下の定理が成り立つ。

定理 1. 新たに生成されるサブシーケンス s_n 、距離の閾値 $\sqrt{2l(1-\theta)}$ 、および最新の l 個のサブシーケンスを除くウィンドウ内の全てのサブシーケンスを管理する kd 木が与えられたとする。このとき、 $\hat{s}_n^\phi$ を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索は、 $\sqrt{\frac{l}{\phi}} \text{dist}(\hat{s}_n^\phi, \hat{s}_p^\phi) \leq \sqrt{2l(1-\theta)}$ を満たす s_p の集合 S_n^{in} を返し、 $|S_n^{\text{in}}| = m_n$ とすると、 $m_n \geq score(s_n)$ である。

証明. 補題 1 より、 s_p は $\sqrt{\frac{l}{\phi}} \text{dist}(\hat{s}_n^\phi, \hat{s}_p^\phi) \leq \sqrt{2l(1-\theta)}$ を満たす必要がある。この不等式から、 $\text{dist}(\hat{s}_n^\phi, \hat{s}_p^\phi) \leq \sqrt{2\phi(1-\theta)}$ が導かれる。次に、最新の l 個のサブシーケンスは s_n とトリビアルマッチであるため $score(s_n)$ には影響しない。以上より、定理 1 が成り立つ。 $\square$

例 1. 図 2 は、2 次元に次元削減されたサブシーケンスを 2 次元上の点として表現している。 $score(s_n)$ の上界値を求める場合、 $\hat{s}_n^\phi$ (赤い点) を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索を実行すると、範囲内に 3 個のサブシーケンスが存在するため、 $m_n = 3$ となり、これが $score(s_n)$ の上界値となる。

定理 1 より、以下の系が成り立つ。

系 1. $s \in S \setminus \{s_n\}$ に対して $score(s) \geq m_n$ ならば、 s_n は

モチーフになり得ない。つまり、正確な $score(s_n)$ の計算を枝刈りできる。

定理 1 を用いるため、SRMM では最新の l 個のサブシーケンスは kd 木で管理しない。ここで、 n を kd 木で管理するデータの数、 m を範囲内に存在するデータの数としたとき、kd 木による範囲検索の時間計算量は $O(\log n + m)$ である。つまり、スコアの上界値を取得する時間計算量は $O(\log(w-l) + m_n)$ であり、さらに、 $\log(w-l) + m_n \ll w$ である。

4.2 スコアが減少するサブシーケンスの特定

ウィンドウがスライドした際、サブシーケンスが削除され、そのサブシーケンスと類似したサブシーケンスのスコアが減少する。スコアが減少するサブシーケンスを特定するため、削除されるサブシーケンスに対して範囲検索を実行することが考えられるが、これは効率的ではない。そこで、スコアが減少するサブシーケンスを効率的に特定するため、2つのリスト SL および PL を定義する。

定義 6 (SL). s_p の SL SL_p は、サブシーケンスの識別子 q と $\rho(s_p, s_q)$ のタプルの集合であり、以下の条件を満たす。

$$SL_p = \{\langle q, \rho(s_p, s_q) \rangle \mid s_q \in S \setminus S_p, \rho(s_p, s_q) \geq \theta\} \quad (9)$$

定義 7 (PL). s_p の PL PL_p は、以下の条件を満たすサブシーケンス s_q の集合である。

$$dist(\hat{s}_p^\phi, \hat{s}_q^\phi) \leq \sqrt{2\phi(1-\theta)}, s_q \notin S_p, \langle q, \cdot \rangle \notin SL_p \quad (10)$$

つまり、範囲検索により s_p のスコアの上界値を求めるとき、 $dist(\hat{s}_p^\phi, \hat{s}_q^\phi) \leq \sqrt{2\phi(1-\theta)}$ を満たす q を PL_p に、 p を PL_q に追加する。さらに、 $\rho(s_p, s_q)$ を計算するとき、 q (p) を PL_p (PL_q) から取り除き、 $\rho(s_p, s_q) \geq \theta$ ならば q (p) を SL_p (SL_q) に追加する。ここで、2つの補題が成り立つ。

補題 2. $|SL_p| + |PL_p| \geq score(s_p)$

補題 3. サブシーケンス s_e の削除によりスコアが減少する可能性のあるサブシーケンス s_q は、 $q \in PL_e$ または $\langle q, \cdot \rangle \in SL_e$ を満たす。

補題 2 および 3 は、定義 7 および 8 から導かれる。また、補題 3 より、 SL_p および PL_p の更新の時間計算量は $O(1)$ であるため、サブシーケンスの削除に対する時間計算量の合計は $O(|SL_e| + |PL_e|)$ である。

4.3 アルゴリズム

本節では SRMM の詳細を紹介する。ウィンドウがスライドした際、まず、削除されるサブシーケンス s_e に対する処理を行い、暫定のモチーフ s_{temp}^* を取得する。その後、新たに生成されるサブシーケンス s_n に対する処理を行い、

Algorithm 1: SRMM (expiration case)

Input: s_e : the expired subsequence

Output: s_{temp}^* : a temporal motif

```

1 Delete  $\hat{s}_e^\phi$  from kd-tree,  $f \leftarrow 0$ 
2 for  $\forall p \in SL_e$  do
3    $SL_p \leftarrow SL_p \setminus \langle e, \cdot \rangle$ 
4   if  $s_p = s^*$  then
5      $f \leftarrow 1$ 
6 for  $\forall p \in PL_e$  do
7    $PL_p \leftarrow PL_p \setminus \{e\}$ 
8 if  $s^* = s_e$  then
9    $f \leftarrow 1, s^* \leftarrow \emptyset$ 
10  $s_{temp}^* \leftarrow s^*$ 
11 if  $f = 1$  then
12   for  $\forall s_p \in S$  such that  $|SL_p| + |PL_p| \geq score(s_{temp}^*)$ 
13     do
14        $s_{temp}^* \leftarrow \text{Motif-Update}(s_p, s_{temp}^*)$ 

```

s^* を取得する。

s_e に対する処理. アルゴリズム 1 は、SRMM の s_e に対する処理を行うアルゴリズムを示している。 s_e が与えられたとき、kd 木から $\hat{s}_e^\phi$ を削除し、フラグ f を 0 とする (1 行)。このときの時間計算量は $O(\log(w-l))$ である。次に、補題 3 より、 $p \in PL_e$ または $\langle p, \cdot \rangle \in SL_e$ を満たす全ての SL_p および PL_p から e を削除する (2-9 行)。ただし、 $score(s^*)$ が減少する場合、または $s^* = s_e$ である場合、 $f = 1$ とする。最後に、 $f = 1$ の場合、モチーフが変化する可能性がある。補題 2 より、 $|SL_p| + |PL_p| \geq score(s_{temp}^*)$ を満たす場合、 s_p がモチーフになる可能性があるため、 $\text{Motif-Update}(s_p, s_{temp}^*)$ を用いて、 s_p の正確なスコアを計算し、 s_{temp}^* を得る (13 行)。 $\text{Motif-Update}(s_p, s_{temp}^*)$ の詳細は後述する。

次に、 s_{temp}^* が現在のモチーフであるか、または新たに生成されるサブシーケンス s_n がモチーフとなるかを確認する。

s_n に対する処理. アルゴリズム 2 は、SRMM の s_n に対する処理を行うアルゴリズムを示している。まず、PAA より $\hat{s}_n^\phi$ を計算し、 $\hat{s}_{n-l}^\phi$ を kd 木に挿入する (1-2 行)。ここで、 s_{n-l} は s_n とトリピアルマッチでない最新のサブシーケンスである。(前述した通り、最新の l 個のサブシーケンスは kd 木で管理しない。) 次に、 $SL_n = \emptyset$ とし、 $\hat{s}_n^\phi$ を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索を実行することにより PL_n を取得する (3-4 行)。その後、 $\forall p \in PL_n$ に対して、 PL_p を更新する。 $s_p = s_{temp}^*$ の場合、 $\rho(s_p, s_n)$ を計算して、 $score(s_p)$ を取得し、 SL_p 、 SL_n 、および PL_n を更新する (6-10 行)。一方、 $s_p \neq s_{temp}^*$ の場合、 PL_p を

Algorithm 2: SRMM (insertion case)

Input: s_n : the new subsequence, s_{temp}^* : a temporal motif

Output: s^* : the current motif

```

1 Compute  $\hat{s}_n^\phi$  by PAA
2 Insert  $\hat{s}_{n-l}^\phi$  to kd-tree
3  $SL_n \leftarrow \emptyset$ 
4  $PL_n \leftarrow \text{Range-Search}(\hat{s}_n^\phi, \sqrt{2\phi(1-\theta)})$ 
5 for  $\forall p \in PL_n$  do
6   if  $s_p = s_{temp}^*$  then
7     Compute  $\rho(s_p, s_n)$ 
8     if  $\rho(s_p, s_n) \geq \theta$  then
9        $SL_p \leftarrow SL_p \cup \langle n, \rho(s_p, s_n) \rangle$ ,
        $SL_n \leftarrow SL_n \cup \langle p, \rho(s_p, s_n) \rangle$ 
10     $PL_n \leftarrow PL_n \setminus \{p\}$ 
11  else
12     $PL_p \leftarrow PL_p \cup \{n\}$ 
13    if  $|SL_p| + |PL_p| \geq \text{score}(s_{temp}^*)$  then
14       $s_{temp}^* \leftarrow \text{Motif-Update}(s_p, s_{temp}^*)$ 
15 if  $|SL_n| + |PL_n| \geq \text{score}(s_{temp}^*)$  then
16    $s^* \leftarrow \text{Motif-Update}(s_p, s_{temp}^*)$ 
17 else
18    $s^* = s_{temp}^*$ 

```

更新し, $|SL_p| + |PL_p| \geq \text{score}(s_{temp}^*)$ を満たすかどうかを確認する. $|SL_p| + |PL_p| \geq \text{score}(s_{temp}^*)$ を満たす場合, $\text{Motif-Update}(s_p, s_{temp}^*)$ を実行し, 必要ならばモチーフを更新する (14 行). 最後に, $|SL_n| + |PL_n| \geq \text{score}(s_{temp}^*)$ を満たす場合, $\text{Motif-Update}(s_n, s_{temp}^*)$ を実行し, s_n がモチーフになるかを確認する (15–16 行). そうでない場合, s_{temp}^* が現在のモチーフであることが保証される (18 行).

三角不等式による高速化. $\text{Motif-Update}(s_n, s_{temp}^*)$ において, $\rho(s_n, s_{temp}^*) \geq \theta$ を満たすかどうかを確認し, SL および PL を更新する. また, 必要に応じて s_{temp}^* を更新する. SL および PL の更新の時間計算量は $O(1)$ であるが, $\rho(s_n, s_{temp}^*) \geq \theta$ の確認には $O(l)$ 時間必要であるため, 以下の定理を用いて高速化を実現する.

定理 2. $s_n, s_p (p \in PL_n), s_q (q \in PL_n \wedge \langle q, \rho(s_p, s_q) \rangle \in SL_p)$, および閾値 θ が与えられたとき, $\text{dist}(\hat{s}_n, \hat{s}_p) + \text{dist}(\hat{s}_p, \hat{s}_q) \leq \sqrt{2l(1-\theta)}$ ならば, $\rho(s_n, s_q) \geq \theta$ である.

証明. $\text{dist}(\cdot, \cdot)$ は z 正規化ユークリッド距離であるため, 三角不等式および式 (4) が成り立つ. 以上より, 定理 2 が成り立つ. $\square$

$|SL_n| + |PL_n| \geq \text{score}(s_{temp}^*)$ であるとき, $\text{score}(s_n)$ を計算する必要がある. 定理 2 を用いて, これを高速化する. 三角不等式の基準となるサブシーケンスとして, $p' \in PL_n$ および $SL_{p'} \neq \emptyset$ を満たす $s_{p'}$ の集合の

表 1: パラメータ設定

パラメータ	値
モチーフ長, l	50, 100 , 150, 200
ウィンドウサイズ, $w[\times 1000]$	5, 10 , 15, 20
閾値, θ	0.75, 0.8, 0.85, 0.9 , 0.95

中で, ϕ 次元上で s_n の最近傍である s_p を用いる. s_p は $\text{Range-Search}(\hat{s}_n^\phi, \sqrt{2\phi(1-\theta)})$ の実行と同時に取得できる. まず, $\text{dist}(\hat{s}_n, \hat{s}_p)$ を計算する. 次に, $\forall q \in PL_n$ に対して, $\langle q, \cdot \rangle \in SL_p$ ならば $\text{dist}(\hat{s}_n, \hat{s}_p) + \text{dist}(\hat{s}_p, \hat{s}_q)$ を計算する. そして, $\text{dist}(\hat{s}_n, \hat{s}_p) + \text{dist}(\hat{s}_p, \hat{s}_q) \leq \sqrt{2l(1-\theta)}$ が成り立つ場合, $\text{dist}(\hat{s}_n, \hat{s}_q)$ を計算する必要はないため, $\text{dist}(\hat{s}_n, \hat{s}_p) + \text{dist}(\hat{s}_p, \hat{s}_q) > \sqrt{2l(1-\theta)}$ または $\langle q, \cdot \rangle \notin SL_p$ を満たす場合のみ $\text{dist}(\hat{s}_n, \hat{s}_q)$ を計算する.

時間計算量. 前述したとおり, kd 木へのサブシーケンスの挿入および削除には $O(\log(w-l))$ かかる. アルゴリズム 1 には, $m_e = |SL_e| + |PL_e|$ としたとき, 少なくとも $O(\log(w-l) + m_e)$ 時間かかる. アルゴリズム 2 には, 少なくとも $O(\log(w-l) + m_n)$ 時間かかる. ここで, m_n は $\text{Range-Search}(\hat{s}_n^\phi, \sqrt{2\phi(1-\theta)})$ によって返されるサブシーケンスの数である. s_p のスコアを正確に計算する場合, $|PL_p|$ 回ピアソン相関の計算が必要となるため, $O(l|PL_p|)$ 時間かかる. よって, ウィンドウがスライドした際, 正確なスコア計算が必要なサブシーケンスの集合を S' としたとき, SRMM の時間計算量は, $O(\log(w-l) + m_e + m_n + \sum_{S'} l|PL_p|)$ となる. 実践的には, $|S'|$ は非常に小さい値であり, 本稿で行った実験では平均で $|S'| \leq 1$ であった. $\log(w-l)$ は定数とみなすと, SRMM の時間計算量は, 削除および生成されるサブシーケンスのスコアの上界値にのみ依存する.

5. 性能評価

本章では, SRMM およびベースライン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.

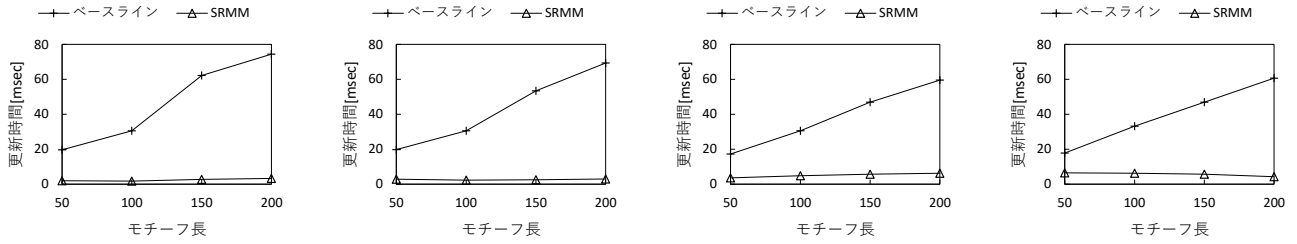
5.1 実験環境

本実験は, Windows10, 3.40GHz Core i7 および 16GB RAM を搭載した計算機で行い, 全てのアルゴリズムを C++ で実装した.

データセット. 以下の 4 つの実データを用いた.

- Google-CPU [22]: Google のデータセンタの CPU 使用率の時系列データ (長さ 133,902)
- Google-Memory [22]: Google のデータセンタのメモリ使用率の時系列データ (長さ 133,269)
- GreenHouseGas [4]: カリフォルニア州の温室効果ガスの排出量の時系列データ (長さ 100,062)
- RefrigerationDevices^{*1}: 冷蔵庫の 2 分ごとの消費電力

^{*1} <http://timeseriesclassification.com/index.php>

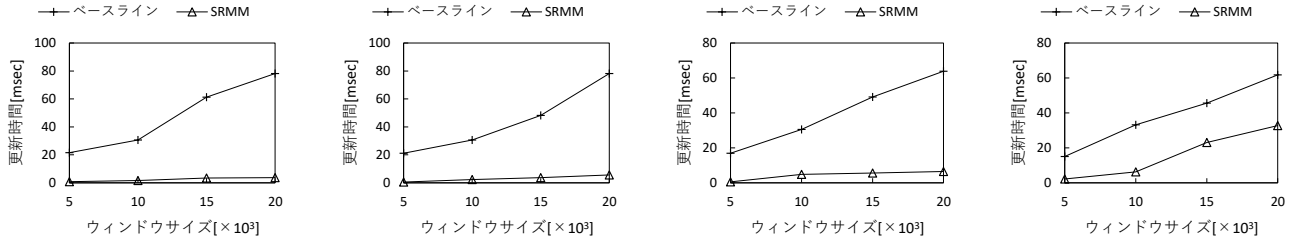


(a) Google-CPU

(b) Google-Memory

(c) GreenHouseGas

(d) RefrigerationDevices

図 3: l の影響

(a) Google-CPU

(b) Google-Memory

(c) GreenHouseGas

(d) RefrigerationDevices

図 4: w の影響

の時系列データ（長さ 270,000）

パラメータ. 本実験で用いたパラメータを表 1 に示す. 太字で表されている値はデフォルトの値である. また, $\phi = \frac{l}{2}$ とし, あるパラメータの影響を調べるとき, 他のパラメータは固定する.

5.2 評価結果

本節では, 各アルゴリズムにおける 1 スライド当たりの平均更新時間の結果を示す.

l の影響. 図 3 に l を変化させたときの結果を示す. ベースラインアルゴリズムにおける更新時間は, l の増加に伴い, 線形に増加する. これは, ベースラインアルゴリズムの時間計算量が $O((w-l)l)$ であるからである. 一方, SRMM は l によらずほぼ一定値となる. ここで, l の増加に伴い, ピアソン相関の計算時間は増加する. しかし, θ を固定した場合, l の増加に伴い, 2 つのサブシーケンス間の距離が大きくなる傾向があり, サブシーケンス間のピアソン相関は小さくなりやすいため, m_e および m_n が減少する. 以上の理由から, SRMM は l によらず高速にモチーフを更新できる. SRMM はベースラインアルゴリズムよりも最大で 24.5 倍高速である.

w の影響. 図 4 に w を変化させたときの結果を示す. 図 4 に示す通り, 図 3 と同様の結果が得られていることがわかる. ベースラインアルゴリズムの時間計算量は w に対して線形であるため, この結果は妥当である. 一方, l を変化さ

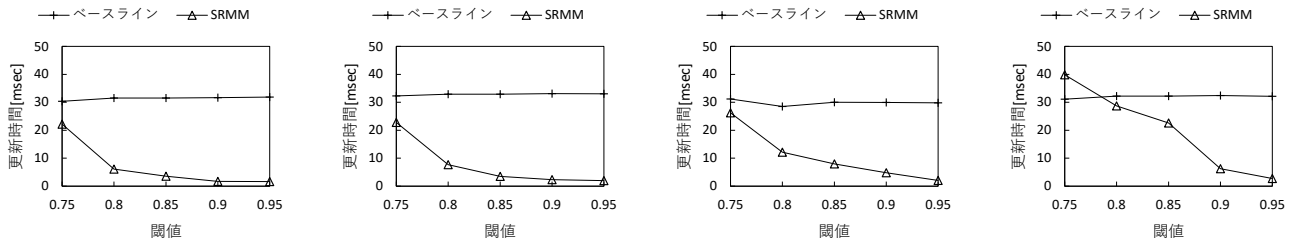
せたときの結果と異なり, w の増加に伴い SRMM の更新時間が増加する. これは, w の増加に伴って, 各サブシーケンスのスコアが大きくなりやすく, m_e および m_n が大きくなりやすいためである.

θ の影響. 図 5 に θ を変化させたときの結果を示す. ベースラインアルゴリズムでは, がスライドした際, 新しく生成されるサブシーケンスおよび削除されるサブシーケンスとウィンドウ内の全てのサブシーケンスとのピアソン相関を計算するため, 更新時間は θ によらず一定である. 一方で, SRMM は θ の増加に伴って更新時間は減少する. 式 (4) より, θ が大きくなるに伴って距離の閾値は小さくなり, SRMM において範囲検索を実行する際, 範囲内に存在するサブシーケンス数が減少する. つまり, m_e および m_n が減少するため, SRMM の更新時間は減少する.

図 5(d) において $\theta = 0.75$ のとき, SRMM の更新時間はベースラインアルゴリズムよりも大きい. ここで, RefrigerationDevices では, θ が小さいとき, 多くのサブシーケンスが互いに類似している. このような場合, 正確なスコアの計算回数が削減できず, 更新時間が大きくなる. しかし, 多くのアプリケーションでは, 大きく相関したサブシーケンスを発見することが求められる. 図 5 で示す通り, SRMM は θ が大きいとき, 非常に高速にモチーフを更新できる.

6. 結論

近年注目されている IoT 機器では, ストリーミング時系



(a) Google-CPU

(b) Google-Memory

(c) GreenHouseGas

(d) RefrigerationDevices

図 5: θ の影響

列データを生成するため、時系列データをリアルタイムに解析することがより重要になっている。本稿では、レンジモチーフ（時系列データの中で最も多く現れるサブシーケンス）をモニタリングする問題に初めて取り組んだ。効率的にモチーフをモニタリングするため、SRMM を提案した。SRMM は PAA および kd 木を用いることにより、不要なスコアの計算を削減することができる。4 つの実データを用いた実験により、SRMM の有効性を確認した。

本稿では、1 次元の時系列データのみを対象とした。しかし、近年、機器は複数のセンサをもち、多次元の時系列データを生成する。今後は、多次元のストリーミング時系列データのレンジモチーフを効率的にモニタリングする手法を検討している。

謝辞. 本研究の一部は、基盤研究 (A)(JP26240013), 基盤研究 (B)(JP17KT0082), および若手研究 (B)(JP16K16056) の研究助成によるものである。ここに記して謝意を表す。

参考文献

- [1] Yankov, D., Keogh, E., Medina, J., Chiu, B. and Zordan, V.: Detecting time series motifs under uniform scaling, *KDD*, pp. 844–853 (2007).
- [2] Grabocka, J., Schilling, N. and Schmidt-Thieme, L.: Latent time-series motifs, *TKDD*, Vol. 11, No. 1, p. 6 (2016).
- [3] Patel, P., Keogh, E., Lin, J. and Lonardi, S.: Mining motifs in massive time series databases, *ICDM*, pp. 370–377 (2002).
- [4] Lucas, D., Kwok, C. Y., Cameron-Smith, P., Graven, H., Bergmann, D., Guilderson, T., Weiss, R. and Keeling, R.: Designing optimal greenhouse gas observing networks that consider performance and cost, *Geoscientific Instrumentation, Methods and Data Systems*, Vol. 4, No. 1, p. 121 (2015).
- [5] Moshtaghi, M., Leckie, C. and Bezdek, J. C.: Online Clustering of Multivariate Time-series, *SDM*, pp. 360–368 (2016).
- [6] Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases, *KIS*, Vol. 3, No. 3, pp. 263–286 (2001).
- [7] Bentley, J. L.: Multidimensional binary search trees used for associative searching, *Communications of the ACM*, Vol. 18, No. 9, pp. 509–517 (1975).
- [8] Li, Y., Yiu, M. L., Gong, Z. et al.: Quick-motif: An efficient and scalable framework for exact motif discovery, *ICDE*, pp. 579–590 (2015).
- [9] Mueen, A., Keogh, E., Zhu, Q., Cash, S. and Westover, B.: Exact discovery of time series motifs, *SDM*, pp. 473–484 (2009).
- [10] Chiu, B., Keogh, E. and Lonardi, S.: Probabilistic discovery of time series motifs, *KDD*, pp. 493–498 (2003).
- [11] Lam, H. T., Pham, N. D. and Calders, T.: Online discovery of top-k similar motifs in time series data, *SDM*, pp. 1004–1015 (2011).
- [12] Mueen, A. and Keogh, E.: Online discovery and maintenance of time series motifs, *KDD*, pp. 1089–1098 (2010).
- [13] Chen, Y., Nascimento, M. A., Ooi, B. C. and Tung, A. K.: Spade: On shape-based pattern detection in streaming time series, *ICDE*, pp. 786–795 (2007).
- [14] Li, Y., Zou, L., Zhang, H. and Zhao, D.: Computing longest increasing subsequences over sequential data streams, *PVLDB*, Vol. 10, No. 3, pp. 181–192 (2016).
- [15] Yeh, C.-C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., Silva, D. F., Mueen, A. and Keogh, E.: Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets, *ICDM*, pp. 1317–1322 (2016).
- [16] Zhu, Y., Zimmerman, Z., Senobari, N. S., Yeh, C.-C. M., Funning, G., Mueen, A., Brisk, P. and Keogh, E.: Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins, *ICDM*, pp. 739–748 (2016).
- [17] Lin, J., Keogh, E., Wei, L. and Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series, *Data Mining and knowledge discovery*, Vol. 15, No. 2, pp. 107–144 (2007).
- [18] Shieh, J. and Keogh, E.: i SAX: indexing and mining terabyte sized time series, *KDD*, pp. 623–631 (2008).
- [19] Castro, N. and Azevedo, P.: Multiresolution motif discovery in time series, *SDM*, pp. 665–676 (2010).
- [20] Begum, N. and Keogh, E.: Rare time series motif discovery from unbounded streams, *PVLDB*, Vol. 8, No. 2, pp. 149–160 (2014).
- [21] Nguyen, H.-L., Ng, W.-K. and Woon, Y.-K.: Closed motifs for streaming time series classification, *KIS*, Vol. 41, No. 1, pp. 101–125 (2014).
- [22] Reiss, C., Wilkes, J. and Hellerstein, J. L.: Google cluster-usage traces: format+ schema, *Google Inc., White Paper*, pp. 1–14 (2011).