情報処理学会論文誌　データベース　Vol.11　No.1　1–10 (Apr. 2018)

自動パターン検出のためのストリームアルゴリズム

川畑 光希[1,a)]　松原 靖子[2,b)]　櫻井 保志[2,c)]

受付日 2017年9月8日, 採録日 2018年1月7日

概要：本論文では，時系列データストリームを対象とした自動特徴抽出手法である StreamScope について述べる．StreamScope は IoT アプリケーションや Web アクセス履歴等の大規模なデータストリームから，(a) 自動的に時系列パターンを発見し，(b) それらの特徴を統計的に要約しながら，データストリームを構成するすべてのパターンを明らかにする．また，(c) 計算時間はデータストリームの長さに依存せず，ストリームマイニングに適した高速な処理を行う．実データを用いた実験では，提案手法がデータストリームに含まれる特徴的なパターンの種類と変化点を自動的かつ正確にとらえることを確認した．さらに，提案手法はオンライン処理でありながら高精度であり，計算時間について大幅な性能向上を達成していることを明らかにした．

キーワード：時系列解析，データストリーム処理，パターン検出

An Online Algorithm for Mining Data Streams

Kouki Kawabata[1,a)]　Yasuko Matsubara[2,b)]　Yasushi Sakurai[2,c)]

Received: September 8, 2017, Accepted: January 7, 2018

Abstract: In this paper, we propose a streaming algorithm, namely StreamScope, that is designed to find important patterns efficiently from data streams. Our proposed method has the following properties: (a) it is effective: it operates on semi-infinite collections of co-evolving streams and summarizes all the streams into a set of multiple discrete segments that agree with human intuition; (b) it is automatic: it incrementally and automatically recognizes such patterns and generates models for each of them if necessary; (c) it is scalable: the time complexity of our method does not depend on the length of the data streams. Our extensive experiments on real datasets demonstrate that StreamScope can find meaningful patterns and achieve great improvement in terms of computational time over its full batch method competitors.

Keywords: Time-series analysis, Data streams, Pattern discovery

1. まえがき

時系列データストリームは，センサネットワーク監視 [1], [2], Web アクティビティ [3], [4], 医療情報解析 [5], [6], 経済分析 [7] 等，様々な分野で絶え間なく生成されている．これらの応用の中で，時系列データを監視し，その中に含まれるトレンドや相関等の有用な時系列パターンをリアルタイムに発見することは非常に重要な課題である．

具体的な例として，Web 上の検索キーワードにおける時系列データでは，基本的に検索数が季節や時間帯によって変動する周期的な特徴を持つことが多い．これらを複数のキーワードに関して同時に解析することによって，ユーザの関心度の推移や急激な変化を細かくパターン化し，それらの特徴を知見として得ることができる．

本論文では，多次元時系列データストリームを対象とした自動パターン検出の手法として StreamScope を提案する．より具体的には，以下の問題を扱う．

d 次元ベクトルで構成されるデータストリーム $\boldsymbol{X} = \{\boldsymbol{x}(1), \boldsymbol{x}(2), \dots\}$ が与えられたとき，(1) 時系列パターンの

[1] 熊本大学大学院自然科学研究科情報電気電子工学専攻
Computer Science and Electrical Engineering, Graduate School of Science and Technology, Kumamoto University, Chuo, Kumamoto 860–8555, Japan
[2] 熊本大学大学院先端科学研究部
Advanced Science and Technology, Kumamoto University, Chuo, Kumamoto 860–8555, Japan
[a)] kouki@dm.cs.kumamoto-u.ac.jp
[b)] yasuko@cs.kumamoto-u.ac.jp
[c)] yasushi@cs.kumamoto-u.ac.jp

変化点検出と $\boldsymbol{X}$ のセグメント化，(2) 類似セグメントのグループ化とモデルパラメータの推定を行う．さらに重要な点として，これらの処理を自動かつオンラインで行う．

1.1　自動特徴抽出とストリーム処理の重要性

時系列データを対象とした研究課題は数多く存在し，パターン発見 [8]，情報要約 [9]，セグメンテーション [10]，クラスタリング [11] 等があげられる．これらの課題に取り組むにあたり，複雑化するデータの統合的な解析には高度なマイニングアルゴリズムが求められる一方で，実用においては大量に生成されるデータを高速に処理することが非常に重要となる．さらに，大規模なデータを解析する際にはユーザの手を介したパラメータ設定やチューニングによる時間的コストが問題となるだけでなく，出力結果にも影響を与える [12]．よって，つねに最新の情報に基づく知見獲得を目的とした，高速かつリアルタイムなモデリング技術，およびユーザの介入なく自動的に処理を行うストリームアルゴリズムは重要である．

1.2　本論文の貢献

本研究では，多次元データストリームのための自動特徴抽出手法として StreamScope を提案する．StreamScope は次の特徴を持つ．

(1) 大規模時系列ストリームに含まれる特徴的なパターンを自動的に発見し，その特徴をモデルパラメータとして抽出する．
(2) 過去に抽出したモデルパラメータを用いてインクリメンタルに時系列パターンの変化点を検出する．さらに，StreamScope はパターンの変化点が最適であることを保証する．
(3) 計算時間およびメモリ消費量はデータストリーム全体の長さに依存せず，高速に動作する．

2.　関連研究

時系列データからのパターン検出に関する研究は，時系列解析，機械学習，データベース等の分野でさかんに取り組まれている [13], [14], [15]．Wang ら [12] は文献 [5] を改良し，pHMM（pattern-based hidden Markov model）を提案した．pHMM はマルコフモデルを用いて時系列データをセグメントに分割し，クラスタリングを行う動的モデルである．しかし，モデル学習の際にはエラー値に関する閾値（すなわちパラメータ）を設定する必要がある．パラメータ設定やモデル構造の定義は出力結果に影響を与えるだけでなく，多くの時間的コストを要するため，大規模なデータを解析する際にはユーザの介入を必要としない手法が望ましい．著者らはこの問題を解決するため，多次元時系列データから自動的に特徴を抽出するための手法として AutoPlait を提案した [16]．AutoPlait は貪欲法に基づくアルゴリズムであり，与えられた時系列データのセグメント化とそれらの分類を同時に行う能力を持つ．しかし，上記のパターン検出手法はストリームマイニングを想定しておらず，大量に生成され続ける時系列データに対してリアルタイムにモデルを更新し，逐次的にパターンを検出する能力を有さない．

一方，ストリームデータ処理に関する研究もさかんである [17], [18], [19], [20], [21]．たとえば，Papadimitriou らはデータストリームから相関とトレンドに相当する隠れ値を検出する手法として SPIRIT [22] を提案した．文献 [23] では，データストリーム間の遅延相関を検出するアルゴリズムが提案されている．著者らの提案した StreamScan [24] は，データストリームから問合せモデルに類似した部分シーケンスを高速に検出することが可能だが，新たな時系列パターンを検出する能力を有さない．

まとめると，データストリームから時系列パターンを自動的かつ高速に検出するための手法は存在せず，本論文ではこの問題を解決するための手法を提案する．

3.　問題設定

本章では，本研究に必要な概念について整理し，具体的な問題設定について述べる．

d 次元のベクトルで構成される時系列データストリーム $\boldsymbol{X}$ が与えられたとき，本研究は $\boldsymbol{X}$ を m 個のセグメント s に分割し，類似セグメントをグループ化することを目的とする．

定義1（セグメント） $\mathcal{S}$ を m 個のセグメント集合 $\mathcal{S} = \{s_1, \ldots, s_m\}$ とする．s_i は i 番目のセグメントの開始点，終了点で構成され（つまり，$s_i = \{t_s, t_e\}$），各セグメントは重複がないものとする．

定義2（レジーム） 類似セグメントのグループをレジーム（regime）と呼び，レジームの数を r とする．各レジームは統計モデル $\boldsymbol{\theta}_i$（$i = 1, \ldots, r$）で表現される．

つまり，各レジームは 1 つの時系列パターンを示しており，すべてのセグメント s はいずれか 1 つのレジームに属する．

定義3（セグメントメンバーシップ） $\mathcal{F}$ を m 個の整数列 $\mathcal{F} = \{f_1, \ldots, f_m\}$ とし，f_i を i 番目のセグメントが所属するレジームの番号（$1 \leq f_i \leq r$）とする．

本研究では各レジームを表現する統計モデルに HMM（hidden Markov model）[*1] を用いる．さらに，複数の時系列パターン間の遷移を表現するために，レジーム間の遷移確率を定義する．

定義4（レジーム遷移行列） $\boldsymbol{\Delta}_{r \times r}$ を r 個のレジームの遷移行列とする．$1 \leq i, j \leq r$, $i \neq j$ としたとき，i 番目のレジーム自身への遷移確率 $\delta_{ii} \in \boldsymbol{\Delta}$，$i$ 番目から j 番目のレジームへの遷移確率 $\delta_{ij} \in \boldsymbol{\Delta}$ はそれぞれ次の式で表される．

[*1] 本論文では出力確率 $\mathbf{B}$ に多次元ガウス分布を仮定する．これにより多次元ベクトルのシーケンスを確率モデルで表現する（つまり $\mathbf{B} = \{\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)\}_{i=1}^k$）．

$$\delta_{ii} = \frac{\sum_{s\in\mathcal{S}_i}|s| - \sum_{j=1}^{r} N_{ij}}{\sum_{s\in\mathcal{S}_i}|s|},\ \delta_{ij} = \frac{N_{ij}}{\sum_{s\in\mathcal{S}_i}|s|} \quad (1)$$

ここで，$\sum_{s\in\mathcal{S}_i}|s|$ はレジーム $\boldsymbol{\theta}_i$ に属するセグメント集合 $\mathcal{S}_i$（$\mathcal{S}_i \subset \mathcal{S}$）の総セグメント長を示す．また，$N_{ij}$ は $\boldsymbol{\theta}_i$ から $\boldsymbol{\theta}_j$ へのレジームの切替え回数を示す．

よって，データストリーム $\boldsymbol{X}$ に含まれるすべての時系列パターンは r 個のレジーム $\boldsymbol{\theta}$ と遷移行列 $\boldsymbol{\Delta}$ で表現される．まとめとして，本研究で求めたい $\boldsymbol{X}$ の特徴を次のように定義する．

定義5（候補解） $\mathcal{C} = \{m, r, \mathcal{S}, \mathcal{F}, \boldsymbol{\Theta}\}$ を $\boldsymbol{X}$ を表現する全パラメータ集合とし，候補解と呼ぶ．ここで，$\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, \boldsymbol{\Delta}\}$ は r 個のレジームを表現するモデルパラメータの集合とする．

本論文の目的は，データストリーム $\boldsymbol{X}$ の現在のパターンを監視し，その変化点検出とセグメント化，およびレジームの発見を自動かつオンラインで行い，$\boldsymbol{X}$ の要約情報である $\mathcal{C}$ を更新しながら継続的に出力することである．本論文で扱う問題を次のように定義する．

問題1　多次元データストリーム $\boldsymbol{X}$ が与えられたとき，$\boldsymbol{X}$ に含まれるすべての時系列パターンを表現するパラメータ集合 $\mathcal{C}$ を出力し続ける．

4. 提案モデル

本研究では，$\boldsymbol{X}$ を表現する最適な候補解 $\mathcal{C}$ を自動的に推定するために，最小記述長（Minimum description length: MDL）に基づく符号化スキームを適用する．MDL はモデル選択基準の 1 つであり，直感的には，データをより圧縮できればより良いモデルと見なす．MDL に従い，$\mathcal{C}$ を表現するためのモデル表現コストと $\mathcal{C}$ が与えられたときの $\boldsymbol{X}$ の符号化コストを定義する．これらの総和が最小となる $\mathcal{C}$ を推定することで，$\boldsymbol{X}$ を適切に表現するために必要なセグメント，およびレジームの数を決定する．

モデル表現コスト．本研究におけるモデル表現コストは次の要素から構成される：現在の時刻 t，および，$\boldsymbol{X}$ の次元数 d：$\log^*(t) + \log^*(d)$ ビット[*2]．セグメントとレジームの総数 m，r：$\log^*(m) + \log^*(r)$ ビット．各セグメントの長さ s：$Cost_M(\mathcal{S}) = \sum_{i=1}^{m-1} \log^* |s_i|$ ビット．セグメントメンバーシップ $\mathcal{F}$：$Cost_M(\mathcal{F}) = m\log(r)$ ビット．r 個のレジームのモデルパラメータ集合：

$$Cost_M(\boldsymbol{\Theta}) = \sum_{i=1}^{r} Cost_M(\boldsymbol{\theta}_i) + Cost_M(\boldsymbol{\Delta}) \quad (2)$$

を要する．ここで，浮動小数点のコストを C_F [*3]とすると，単一レジームのモデル $\boldsymbol{\theta}$ は k 個の状態それぞれについて初期確率，遷移確率，および d 次元ガウス分布の表現コスト：

$$Cost_M(\boldsymbol{\theta}) = \log^*(k) + C_F \cdot (k + k^2 + 2kd) \quad (3)$$

を要する．同様に，レジーム遷移行列は $Cost_M(\boldsymbol{\Delta}) = C_F \cdot r^2$ のコストを要する．

時系列データの符号化コスト．ハフマン符号を用いた情報圧縮では，モデルパラメータ $\boldsymbol{\theta}$ が与えられたときの $\boldsymbol{X}$ の符号化コストを次のように表現する：

$$Cost_C(\boldsymbol{X}|\boldsymbol{\theta}) = -\log_2 P(\boldsymbol{X}|\boldsymbol{\theta}) \quad (4)$$

ここで，$P(\boldsymbol{X}|\boldsymbol{\theta})$ は $\boldsymbol{X}$ の尤度を示す．したがって，$\boldsymbol{X}$ と r 個のレジームのモデルパラメータ $\boldsymbol{\Theta}$ が与えられたとき，符号化コストは次のように表される．

$$\begin{aligned} Cost_C(\boldsymbol{X}|\boldsymbol{\Theta}) &= \sum_{i=1}^{m} Cost_C(\boldsymbol{X}[s_i]|\boldsymbol{\Theta}) \\ &= \sum_{i=1}^{m} -\lg(\delta_{vu} \cdot (\delta_{uu})^{|s_i|-1} \cdot P(\boldsymbol{X}[s_i]|\boldsymbol{\theta}_u)) \end{aligned} \quad (5)$$

ここで，i と $(i-1)$ 番目のセグメントはそれぞれ u と v 番目のレジームに属し，$f_i = u$，$f_{i-1} = v$，$f_0 = f_1$ とする．また，$\boldsymbol{X}[s_i]$ はセグメント s_i の部分シーケンス，$P(\boldsymbol{X}[s_i]|\boldsymbol{\theta}_u)$ は s_i の尤度を示し，$\boldsymbol{\theta}_u$ は s_i が属するレジームである．以上より，$\boldsymbol{X}$ の総符号長は次式で与えられる．

$$\begin{aligned} Cost_T(\boldsymbol{X};\mathcal{C}) &= Cost_T(\boldsymbol{X}; m, r, \mathcal{S}, \mathcal{F}, \boldsymbol{\Theta}) \\ &= \log^*(t) + \log^*(d) + \log^*(m) + \log^*(r) + Cost_M(\mathcal{S}) \\ &\quad + Cost_M(\mathcal{F}) + Cost_M(\boldsymbol{\Theta}) + Cost_C(\boldsymbol{X}|\boldsymbol{\Theta}) \end{aligned} \quad (6)$$

しかし，本研究で対象とする時系列データは半無限長のシーケンスであり，データ全体の符号長を計算することは困難である．よって，新たに生成されたデータに対して動的にパラメータを最適化する．より具体的には，$\boldsymbol{X}$ の最新の部分シーケンスを $\boldsymbol{X}_c$ としたとき，$\boldsymbol{X}_c$ の圧縮に必要となる総コストの増加量を最小化するように $\mathcal{C}$ 内の各パラメータを更新する．今，候補解 $\mathcal{C}$ 内のセグメント数が m から m' へ，レジーム数が r から r' へ変化した場合を考える．このとき，追加されたセグメントおよびレジームに関する情報が集合 $\{\mathcal{S}, \mathcal{F}, \boldsymbol{\Theta}\}$ の各要素として追加され，以下のようにモデルコストが増加する．

- $\Delta Cost_M(\mathcal{S}) = \sum_{i=m}^{m'-1} \log^* |s_i|$
- $\Delta Cost_M(\mathcal{F}) = m'\log(r') - m\log(r)$
- $\Delta Cost_M(\boldsymbol{\Theta}) = \sum_{i=r+1}^{r'} Cost_M(\boldsymbol{\theta}_i) + C_F \cdot r'^2 - C_F \cdot r^2$

まとめると，総コストの増加量 $\Delta Cost_T(\boldsymbol{X}_c;\mathcal{C})$ は次の式で表される．

$$\begin{aligned} \Delta Cost_T(\boldsymbol{X}_c;\mathcal{C}) = &\log^*(m') - \log^*(m) + \log^*(r') \\ &-\log^*(r) + \Delta Cost_M(\mathcal{S}) + \Delta Cost_M(\mathcal{F}) \\ &+\Delta Cost_M(\boldsymbol{\Theta}) + Cost_C(\boldsymbol{X}_c|\boldsymbol{\Theta}) \end{aligned} \quad (7)$$

特に，$\boldsymbol{X}_c$ をすべて既存レジームに割り当てた場合は

[*2] $\log^*$ は整数のユニバーサル符号長を示す [25].
[*3] 本論文では 4 × 8 ビットとする．

$r' = r$ であり，$\log^*(r') - \log^*(r) = 0$，$\Delta Cost_M(\boldsymbol{\Theta}) = 0$ である．

つまり，本研究の具体的な目標は最新のデータ $\boldsymbol{X}_c$ に含まれる時系列パターンをリアルタイムにとらえ，$\Delta Cost_T(\boldsymbol{X}_c; \mathcal{C})$ を最小化するような $\mathcal{C}$ の更新，すなわち，

- セグメント数 m とレジーム数 r の更新
- セグメント s とセグメントメンバーシップ f の追加
- $\boldsymbol{\Theta}$ 内のパラメータ更新，モデルパラメータ $\boldsymbol{\theta}$ の追加

をインクリメンタルに行うことである．

5. ストリームアルゴリズム

本章では，データストリームから自動的に解 $\mathcal{C}$ を求めるためのアルゴリズムである StreamScope について述べる．

5.1 概要

StreamScope は $\boldsymbol{X}$ の最新の部分シーケンスをカレントウィンドウ $\boldsymbol{X}_c$ とし，$\boldsymbol{X}_c$ から時系列パターンの発見とそれらの変化点検出を行う．より具体的には，以下の手順で処理を行う．

(1) レジーム変化の検出：過去に検出したレジーム $\boldsymbol{\Theta}$ に基づき $\boldsymbol{X}_c$ を監視することで，時系列パターン（レジーム）の変化点を検出する．
(2) レジームの推定：レジーム変化が検出された場合に，$\boldsymbol{X}_c$ から新たなレジームを推定する．また，$\boldsymbol{X}_c$ を推定したレジームに割り当てた場合のレジーム変化点を発見する．
(3) 候補解 $\mathcal{C}$ の更新：4 章で述べたコスト関数（式 (7)）に基づき $\mathcal{C}$ を更新する．すなわち，新たに推定したレジームの必要性を判断したうえで，新たに得られたセグメントに関する情報を $\mathcal{C}$ に反映させる．$\mathcal{C}$ の更新処理を終えた後，レジーム変化の監視を再開する．

5.2 SegmentAssignment

ここでは，各時刻 t における入力 $\boldsymbol{x}(t)$ とモデルパラメータ集合 $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, \boldsymbol{\Delta}_{r\times r}\}$ が与えられたとき，$\boldsymbol{X}$ のパターン変化をリアルタイムにとらえ，セグメントとして適切なレジームへ割り当てるためのアルゴリズムである SegmentAssignment について述べる．Algorithm 1 に SegmentAssignment の具体的な処理の流れを示す．まずはじめに，符号化コスト（式 (5)）を最小化するレジーム変化点を発見するため，$\boldsymbol{X}$ の尤度 $P(\boldsymbol{X}|\boldsymbol{\Theta})$ を計算する．

時刻 t における尤度 P は次式で表される．

$$P(\boldsymbol{x}(t)|\boldsymbol{\Theta}) = \max_{1\le i\le r}\Big\{\max_{1\le u\le k_i}\{p_{i;u}(t)\}\Big\} \tag{8}$$

$$p_{i;u}(t) = \max\begin{cases}\delta_{ji}\cdot \max_v\{p_{j;v}(t-1)\}\cdot \pi_{i;u}\cdot b_{i;u}(\boldsymbol{x}(t)) \\ \delta_{ii}\cdot \max_w\{p_{i;w}(t-1)\cdot a_{i;wu}\}\cdot b_{i;u}(\boldsymbol{x}(t))\end{cases} \tag{9}$$

Algorithm 1 SegmentAssignment $(\boldsymbol{x}(t), \boldsymbol{\Theta}, \gamma)$

Input: (a) New vector $\boldsymbol{x}(t)$ at time tick t, (b) Regime parameter set $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r\}$ (c) regime transition matrix $\boldsymbol{\Delta}$ and (d) γ
Output: (a) Number of segments m, (b) segment set $\mathcal{S}$, (c) segment membership $\mathcal{F}$ and (d) γ

```
m ← 0; S ← ∅; F ← ∅;
/* Find cut-points with Equations (9) and (10) */
for i = 1 : r do
    Compute p_{i;j}(t) for state j = 1 to k_i;
    Update L_{i;j}(t) for state j = 1 to k_i;
end for
if find the first regime transition then
    Compute γ; /* Equation (11) */
end if
/* Add guaranteed segments into optimal regime */
if γ = 0 then
    L_best = L_{i;j}(t) | arg max_{i∈r, j∈k_i} P(x(t)|Θ);
    t_s ← Starting point of s_m;
    for each cut point l = {t_e, i} ∈ L_best do
        Create a new segment s = {t_s, t_e};
        Add s into S; f_m ← i; m ← m + 1;
        t_s ← t_e + 1;
    end for
    Initialize L;
end if
return {m, S, F, γ};
```

ここで，$p_{i;u}(t)$ は時刻 t におけるレジーム $\boldsymbol{\theta}_i$ の状態 u の確率の最大値を示し，式 (9) の上段は，レジーム間（$\boldsymbol{\theta}_j$ から $\boldsymbol{\theta}_i$）に遷移したときの確率，下段はレジーム $\boldsymbol{\theta}_i$ 内部において状態遷移したときの確率を示す．また，$\max_v\{p_{j;v}(t-1)\}$ は時刻 $t-1$ における $\boldsymbol{\theta}_j$ 内の確率の最大値，$\pi_{i,u}$，$b_{i;u}(\boldsymbol{x}(t))$，$a_{i;wu}$ はそれぞれ $\boldsymbol{\theta}_i$ 内における状態 u の初期確率，出力確率，状態 w から状態 u への遷移確率を示す．本研究では，動的計画法に基づき遷移確率 $p_{i;u}(t)$ を時刻 $t-1$ の遷移確率から累積的に計算する．

次に，$\mathcal{L} = \{l_1, l_2, \ldots\}$ をレジーム変化点 l の集合とする．時刻 t において，レジーム $\boldsymbol{\theta}_i$ への変化点を検出した場合，変化点 $l = \{t, i\}$ を集合に加える．次の式に示すように，検出したすべての変化点候補を集合として各レジームの各状態について保持し，その中から最適な変化点集合を決定する．

$$\mathcal{L}_{i;u}(t) = \begin{cases}\mathcal{L}_{j;v}(t-1)\cup l = \{t, i\} & \text{// if switch} \\ \mathcal{L}_{i;u}(t-1) & \text{// else}\end{cases} \tag{10}$$

ここで，$\mathcal{L}_{i;u}(t)$ は時刻 t におけるレジーム $\boldsymbol{\theta}_i$，状態 u の変化点集合を示す．これらは式 (9) の尤度計算に基づき更新される．

補助定理1　モデル $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, \boldsymbol{\Delta}\}$ および長さ n のシーケンス $\boldsymbol{X}[1:n]$ が与えられたとき，SegmentAssignment は探索漏れを発生させないことを保証する．

証明1　動的計画法に基づき，SegmentAssignment は各時刻における尤度を累積的に更新する．このとき，変化点集合 $\mathcal{L}$ は各レジーム内の各状態についての変化点候補をすべて保有しているため，時刻 n において尤度 $P(\boldsymbol{x}(n)|\boldsymbol{\Theta})$ を最大にする変化点集合は最適 Viterbi パスに等しい．よって，SegmentAssignment は最適な変化点集合を検出することができる．　□

続いて，候補集合の中から尤度を最大にする最適解 $\mathcal{L}_{best}$ を選出したい．しかし，本研究で扱うデータストリームは半無限長のシーケンスであり，最大尤度を与える変化点集合は刻々と変化する．そこで，本手法では最大尤度を与えるレジームが初めて他のレジームに変化した時刻を t としたとき，時刻 $t+\gamma$ において $\mathcal{L}_{best}$ を決定する．

補助定理2　モデル $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, \boldsymbol{\Delta}\}$ およびシーケンス $\boldsymbol{X}[1:t+\gamma]$ が与えられたとき，時刻 1 から t までの変化点集合は時刻 $t+\gamma$ において最適である．

証明2　尤度 $P(\boldsymbol{x}(t+\gamma)|\boldsymbol{\Theta})$ は時刻 $t+\gamma$ における確率の最大値を示す．補助定理 1 より，時刻 1 から $t+\gamma$ までの変化点集合は最適であるため，時刻 1 から t における部分パスは時刻 $t+\gamma$ において最適である．　□

すなわち，γ は遷移先のセグメント長に基づくものである．変化点の候補を検出した場合，ただちにその変化点によるセグメントの割当てを行わず，γ 保証された変化点集合をもとに割り当てる．

定義6（γ–保証）　本研究では，時刻 1 から t における部分パスが補助定理 2 を満たすことを γ 保証と呼ぶ．時刻 t において最大尤度 $P(x(t)|\boldsymbol{\Theta})$ を出力するレジームがレジーム $\boldsymbol{\theta}_i$ への変化した場合の γ を次のように定義する．

$$\gamma \propto \left\lceil \frac{1}{1-\delta_{ii}} \right\rceil \quad (1 \leq i \leq r) \tag{11}$$

直感的には，時刻 t においてレジーム $\boldsymbol{\theta}_i$ への変化点候補を検出したとき，過去にレジーム $\boldsymbol{\theta}_i$ に割り当てられたセグメントの長さを考慮し，γ 時刻後において尤もらしいことが保証された変化点集合に基づきセグメントの分割/割当てを行う．

補助定理3　レジーム $\{\boldsymbol{\theta}_i\}_{i=1}^{r}$ 中の隠れ状態の数の最大値 $\max\{k_1, \ldots, k_r\}$ を k とする．SegmentAssignment は $O(drk + rk^2 + r^2)$ のメモリ量と単位時間あたり $O(drk^2 + dr^2k)$ の計算時間を要する．

証明3　各時刻において，SegmentAssignment は動的計画法に基づき d 次元データに対する尤度を更新する．すなわち，トレリス構造を 1 つしか保持せず，各レジームについて，内部で遷移した確率を dk^2 個，他のレジームから遷移した確率を $(r-1) \times dk$ 個計算する．よって，計算量は合計 $O(r \times (dk^2 + (r-1)dk)) = O(drk^2 + dr^2k)$ である．また，SegmentAssignment は単一のトレリス構造を保持するために $r \times dk$ 個の配列を 2 つ，$\boldsymbol{\Theta}$ 内の各モデルパラメータを合計 $r \times (k + k^2 + 2kd) + r^2$ 個保持する．よって，全体で $O(drk + rk^2 + r^2))$ のメモリ空間を必要とする．　□

Algorithm 2 StreamScope $(\boldsymbol{x}(t), \mathcal{C})$

Input: new vector $\boldsymbol{x}(t)$ at time tick t and current parameter set $\mathcal{C} = \{m, r, \mathcal{S}, \mathcal{F}, \boldsymbol{\Theta}\}$
Output: Updated parameter set $\mathcal{C}$

```
/* Assign x into existing regimes */
Decrement γ;
{m', S', F', γ} ← SegmentAssignment (x(t), Θ, γ);
/* Estimate new regimes in X_c */
if γ = 0 then
    t_s ← t_s ∈ s_m;   c = f_m;   X_c ← X[t_s : t];
    Remove s_m, f_m;  m ← m − 1;
    /* R: list for number of segments, segment set, regime */
    R ← RegimeGeneration (X_c);
    /* Compare ΔCost_T to decide optimal r */
    if ΔCost_T(X_c; m', r, S', F', Θ) > ΔCost_T(X_c; R) then
        for each {m*, S*, θ*} ∈ R do
            if t_s = t*_s ∈ s*_1 then
                θ_c ← ModelUpdate (S*);
                f_i ← c  (i = m + 1, ..., m + m*);
            else
                Θ ← Θ ∪ θ*;   r ← r + 1;
                f_i ← r  (i = m + 1, ..., m + m*);
            end if
            S ← S ∪ S*;   m ← m + m*;
        end for
    else
        for i = 1 : m' do
            θ_{f'_i} ← ModelUpdate (s'_i);
            m ← m + 1;   S ← S ∪ s'_i;   f_m ← f'_i;
        end for
    end if
    Update Δ;  // Equation (1)
    Compute γ;  // Equation (11)
end if
return {C, γ};
```

5.3　StreamScope

次に，自動的に新たなレジームを検出し，解 $\mathcal{C}$ を求めるためのアルゴリズムである StreamScope について述べる．

StreamScope の詳細を Algorithm 2 に示す．時刻 t においてベクトル $\boldsymbol{x}(t)$ が与えられたとき，提案手法は SegmentAssignment によって現在の時系列パターン（レジーム）の変化を監視し，過去に検出したレジームへ遷移した場合のセグメントの情報 $\{m', \mathcal{S}', \mathcal{F}'\}$ を得る．また，新たなレジームを検出するため，変化点の候補が γ–保証されるとき（すなわち，$\gamma = 0$ のとき），RegimeGeneration を用いて $\boldsymbol{X}_c$ から新たなレジームを推定し，生成したレジームに

属するセグメントとモデルパラメータの集合 $\{m^*, \mathcal{S}^*, \boldsymbol{\theta}^*\}$ から構成される集合 $\mathcal{R}$ を得る．このとき，最後に検出した時系列パターンと，現在のパターンの変化点を正確に検出するために，提案手法が保持する最新の部分シーケンスを次のように定義する．

定義7（カレントウィンドウ：$\boldsymbol{X}_c$） t_s を最新セグメント s_m の開始点，t を現時刻としたとき，データストリーム $\boldsymbol{X}$ の最新の部分シーケンスを $\boldsymbol{X}_c = \boldsymbol{X}[t_s : t]$ とする．

続いて，レジームを追加した場合とそうでない場合のそれぞれに対する符号化コスト（式 (7)）を比較し，コストがより小さくなるように解 $\mathcal{C}$ の各パラメータを更新する．具体的には，

- レジームを追加したときの符号化コストが小さい場合，新たに推定した $\boldsymbol{\theta}^*$ を $\boldsymbol{\Theta}$ に追加し，r を更新する．ここで，$\boldsymbol{X}_c$ の開始点から始まるセグメントが属するレジームは，更新前の s_m が属していたレジーム $\boldsymbol{\theta}_c$ と重複するため，$\boldsymbol{\theta}_c$ のパラメータを更新する．また，生成したセグメントの分割位置およびセグメントメンバーシップをそれぞれ $\mathcal{S}$，$\mathcal{F}$ に追加し，セグメント数 m を更新する．
- 既存レジームに割り当てたときの符号化コストが小さい場合，上記と同様に，割り当てたセグメントに関する $\{m, \mathcal{S}, \mathcal{F}\}$ をそれぞれ更新する．また，分割された部分シーケンスを用いて割当て先のパラメータ $\boldsymbol{\theta}$ を更新する．

以上のいずれかの処理を行い，セグメントおよびレジームの総数に基づきレジーム遷移行列 $\boldsymbol{\Delta}$ を更新する．最後に，現在のレジーム ID で γ を設定する．これにより，レジーム変化が検出されない場合にも γ 時刻ごとに $\mathcal{C}$ を更新し，現在のレジームパターンに類似した新たな時系列パターンを検出することができる．

レジームパラメータ $\boldsymbol{\theta}$ の推定． 時刻 t におけるカレントウィンドウ $\boldsymbol{X}_c$ が与えられたとき，RegimeGeneration は $\boldsymbol{X}_c$ から新たな時系列パターンを発見する．具体的には，$Cost_T(\boldsymbol{X}_c; \mathcal{R})$ を最小化するレジーム集合 $\mathcal{R}$ を出力する [16]．まず，$\boldsymbol{X}_c = [t_s : t]$ からモデルパラメータ $\boldsymbol{\theta}$ を推定する．次に，$\boldsymbol{X}_c$ をセグメントとした 1 つのレジームと仮定し，$\{m = 1, s_1 = \{t_s : t\}, \boldsymbol{\theta}\}$ をスタック $\mathcal{Q}$ に追加する．その後，$\mathcal{Q}$ が空になるまで以下の 3 つのステップを反復する．

(1) $\mathcal{Q}$ からエントリ $\{m, \mathcal{S}, \boldsymbol{\theta}\}$ を取り出す．$\boldsymbol{X}[\mathcal{S}]$ から複数のセグメント s をサンプリングし，それぞれモデルパラメータ $\boldsymbol{\theta}_s$ を推定する．それらのすべての組 $\{\boldsymbol{\theta}_{s_1}, \boldsymbol{\theta}_{s_2}\}$ に対する $Cost_C(\boldsymbol{X}_c[\mathcal{S}]|\boldsymbol{\theta}_{s_1}, \boldsymbol{\theta}_{s_2})$ を計算し，コストを最小化するレジーム候補の組 $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ を得る．

(2) $Cost_C(\boldsymbol{X}_c[\mathcal{S}]|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ を最小化する Viterbi パスに基づき，セグメント $\{\mathcal{S}_1, \mathcal{S}_2\}$ を生成する．得られた 2 つのセグメント集合より，モデルパラメータ $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ をそれぞれ再推定する．

(3) ステップ (2) を $Cost_C(\boldsymbol{X}_c[\mathcal{S}]|\boldsymbol{\theta}_{s_1}, \boldsymbol{\theta}_{s_2})$ が下がらなくなるまで繰り返し，最終的に $Cost_T(\boldsymbol{X}_c; \mathcal{S}, \boldsymbol{\theta})$ と $Cost_T(\boldsymbol{X}_c; \mathcal{S}_1, \mathcal{S}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ を比較する．分割後のコストが小さい場合は $\{m_1, \mathcal{S}_1, \boldsymbol{\theta}_1\}$, $\{m_2, \mathcal{S}_2, \boldsymbol{\theta}_2\}$ を $\mathcal{Q}$ に追加する．そうでない場合は $\mathcal{R}$ に $\{m, \mathcal{S}, \boldsymbol{\theta}\}$ を追加する．

HMM のパラメータ推定には Baum-Welch アルゴリズムを使用する．Baum-Welch アルゴリズムは，モデル $\boldsymbol{\theta}$ に対して隠れ状態の数 k を与える必要があるが，本研究ではコスト関数 $Cost_M(\boldsymbol{\theta}) + Cost_C(\boldsymbol{X}_c[\mathcal{S}]|\boldsymbol{\theta})$ が下がらなくなるまで隠れ状態の個数を $k = 1, 2, 3, \ldots$ と増加させ，各レジームの k を自動的に決定する．

理論的な分析．

補助定理4 カレントウィンドウ $\boldsymbol{X}_c$ の長さを l とする．各時刻において，StreamScope は $O(dl + drk + rk^2 + r^2)$ のメモリ量と単位時間あたり最小 $O(drk^2 + dr^2k)$，最大 $O(drk^2 + dr^2k + ldk^2)$ の計算時間を要する．

証明4 補助定理 3 より，StreamScope は各時刻において $O(drk^2 + dr^2k)$ の計算時間を要する．加えて，新たにレジームを推定する場合，$\boldsymbol{X}_c$ に対して符号化コストの計算とモデルパラメータの推定を繰り返すため $O(\#iter \times ldk^2)$ の計算時間を要する．ここで，反復回数 $\#iter$ は非常に小さい定数であるため無視する．よって，提案手法の計算時間は最小 $O(drk^2 + dr^2k)$，最大 $O(drk^2 + dr^2k + ldk^2)$ である．また，提案手法は新たなレジームを推定するために，$\boldsymbol{X}_c$ に含まれる dl 個のデータを保持する．よって補助定理 3 より，提案手法は合計 $O(dl + drk + rk^2 + r^2)$ のメモリ空間を必要とする． □

各レジームの状態数 k およびレジームの総数 r は，コストモデルに基づき $\boldsymbol{X}$ を効率良く圧縮するように決定されるパラメータであるため，比較的小さくなると想定される．よって，6.3 節に示すように，提案手法はストリームマイニングに適した高速な処理を行うことができる．

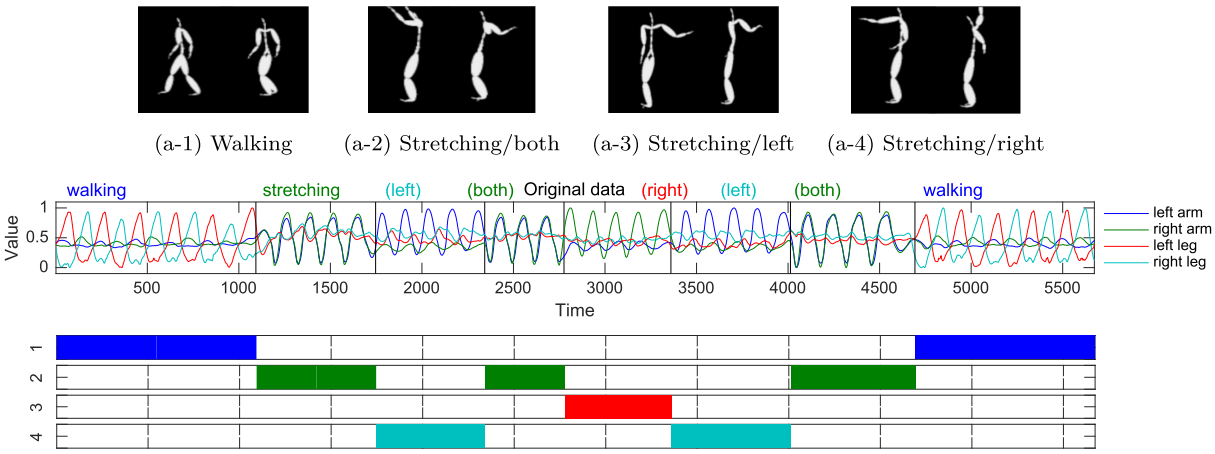
6. 評価実験

本論文では StreamScope の有効性を検証するため，以下の項目について実データを用いた実験を行った．

- Q1 時系列データストリームを対象としたパターン検出および変化点抽出に対する提案手法の有効性
- Q2 検出したセグメントおよびレジームに対する提案手法の精度
- Q3 時系列データストリームに対する提案手法の計算時間とメモリ使用量

実験は 16 GB のメモリ，Intel Core i5 3.3 GHz の CPU を搭載した iMac 上で実施し，データセットとして *MoCap*[*4] を使用した．*MoCap* は，1 秒 120 フレームでヒトの

[*4] http://mocap.cs.cmu.edu/



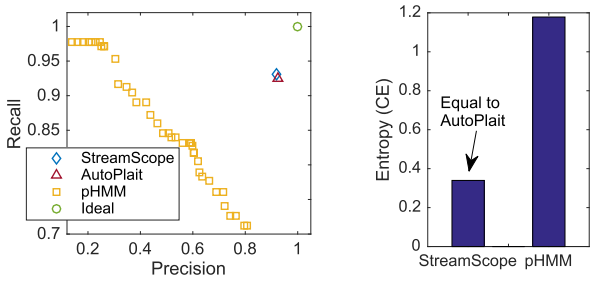
(b) オリジナルデータストリーム（上段）とセグメンテーション結果（下段）

図 1　*MoCap* データストリームに対する StreamScope の出力結果
Fig. 1　Simulation results of StreamScope.

動きを計測したモーションキャプチャのデータセットである．本実験ではデータの中から左右の腕と足の 4 次元から構成される加速度の値を選出し，平均値と分散値で正規化（z-normalization）して使用した．

6.1　Q1．提案手法の有効性

本節では，StreamScope のパターン検出の能力を検証した結果を示す[*5]．**図 1** は *MoCap* データストリームに対する StreamScope の出力結果を示している．図 1 (a) に示すように，オリジナルデータは人が運動する様子をとらえたものであり，歩く動作（“Walking”，図 1 (a-1)）および腕を回す動作（“Stretching”，図 1 (a-2)〜(a-4)）の合計 4 種類の動作で構成される．図 1 (b) 下段の各番号はレジームの ID，長方形の両端は各セグメントの開始点と終了点を示す．StreamScope は $t = 1418$ において “Stretching (both)” を検出し，レジーム#2 を生成した．その後，カレントウィンドウの開始点をセグメント#2 の開始点に移して処理を進める．3 種類の腕を回す動作は非常に類似した特徴を持ち，はじめは 1 つの時系列パターンとして表現されるが，やがてそのすべてを効率的にモデルパラメータとして圧縮することができなくなる．そのため，提案手法は $t = 3938$ において腕の動きが異なる動作を表すレジーム#3，レジーム#4 を新たに生成した．本データストリームにおけるすべての時系列パターンを検出した提案手法は，セグメント#7，セグメント#8 に関しても逐次的に適切なレジームへ割り当てることに成功した．図に示すように，提案手法はデータストリームに含まれる動作に関する事前知識を必要とせず，時間経過とともにモデルパラメータを更新しながらすべての時系列パターンに該当する部分シーケンスの時間情報，すなわち 4 個のレジームと 8 個のセグメントをすべて正確に検出した．



(a) 適合率と再現率　　(b) CE スコア

図 2　StreamScope の精度
Fig. 2　Accuracy of StreamScope: (a) Precision and recall. (b) CE score.

6.2　Q2．提案手法の精度

次に，StreamScope の変化点抽出とクラスタリング精度を検証するため，*MoCap* 中の合計 15 個のデータを用い，オフラインにセグメンテーションを行う手法である AutoPlait [16]，および pHMM [12] との比較実験を行った．

図 2 (a) は，各データに付与されたラベルの開始点・終了点を正解値としたときの各手法の適合率，再現率を示している．適合率は検出した変化点の合計数とそのうち正解であった点の合計数の割合を示しており，再現率はすべての変化点の正解値の数と検出された点の中で正解した合計数の割合を示す．pHMM はパラメータを必要とするため，本実験ではセグメンテーション結果に影響するパラメータ ϵ_r を $0.1, 0.2, \ldots, 10.0$ と変化させながら検証を行った．提案手法と AutoPlait はパラメータを持たず，出力結果が 1 つに定まるため，図 2 (a) において 1 点のみで表現される．一方，pHMM はパラメータにより結果が異なるため，複数の点で表現される．それらのどの点と比較した場合にお

[*5] 各実験において，データの前半 1/2 を静的にセグメンテーションし，その平均長を γ の初期値としたうえで，データ全体に対して提案アルゴリズムを適用した．

いても，提案手法の精度は高く，オンライン手法であるにもかかわらず，適合率 93%，再現率 91%と高い精度を示している．

図 2 (b) は各手法のクラスタリング精度を示す．本実験ではレジームの正解ラベルおよび推定したラベルに関する混同行列（CM: confusion matrix）を作成し，条件付きエントロピー（CE: conditional entropy）のスコア：$CE = -\sum_{i,j} \frac{CM_{ij}}{\sum_{ij} CM_{ij}} \log \frac{CM_{ij}}{\sum_{j} CM_{ij}}$．を計算した．正確にラベルを求めることができた場合，混同行列は対角行列となり，$CE = 0$ となる．pHMM のパラメータは平均 10 個程度のセグメントが得られる $\epsilon_r = 0.1$，$\epsilon_c = 1.0$ を使用した．図より，オンラインにセグメンテーションを行った場合にも高い精度を示すことが分かる．

6.3　Q3. 提案手法の計算時間とメモリ使用量

最後に，StreamScope の性能を評価するための実験を行った．**図 3** は，StreamScope と比較手法である AutoPlait [16]，pHMM [12] との各時刻 t における計算時間の比較である．データ集合には *MoCap* を用い，pHMM のパラメータは図 2 (b) と同じものを使用した．実験結果から，StreamScope は比較手法と比べて高速に動作することが分かる．AutoPlait と pHMM はオフライン手法であるため計算コストがデータ長に依存し，各時刻においてそれぞれ $O(t)$, $O(t^2)$ の計算コストを要する．これは，いずれ比較手法がデータストリームを処理できなくなることを意味する．一方，StreamScope は $\boldsymbol{X}_c$ に対してのみ処理を行うため，計算時間はデータストリーム全体の長さ t に依存することなく高速に動作する．図 3 において，StreamScope の計算時間が上昇している点は新たなレジーム $\boldsymbol{\theta}$ を推定した時刻を示すが，その場合においてもオフライン手法である AutoPlait と比較して約 100 倍の性能向上を達成した．

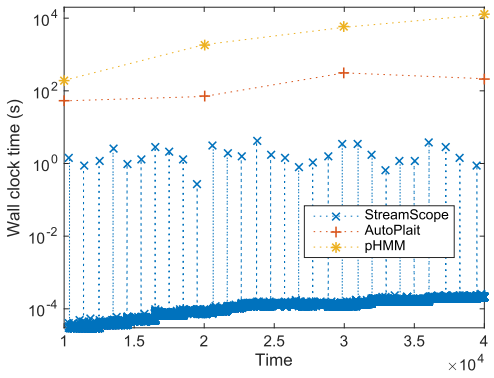


図 3　各時刻における StreamScope の計算時間
Fig. 3　Wall clock time vs. sequence length t.

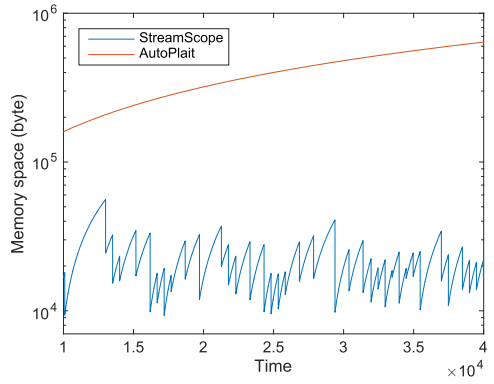


図 4　各時刻における StreamScope のメモリ使用量
Fig. 4　Memory space consumption vs. sequence length t.

図 4 は，同データセットに対する StreamScope と AutoPlait のメモリ使用量を示したものである．計算時間と同様に，提案手法のメモリ使用量はストリームの長さに依存せず一定のメモリ使用量を示しており，データストリームを効率的に処理可能であることが分かる．

7.　アプリケーション

本章では，StreamScope の実用的なアプリケーションとして，*GoogleTrend*[*6] を用いた Web におけるユーザ動向の特徴抽出を行う．*GoogleTrend* は，Google による週ごとの検索クエリの頻度を 13 年間にわたり集計したものである．本実験では，2 年分のデータ長を γ の初期値とした．

- イベント検出：**図 5** (a) は，雨具に関するキーワード 4 つ（“umbrella”，“rain coat” 等）の検索数を示している．このデータは年単位の周期性を持っており，降水量が増加する時期にピークを迎える．しかし，2007 年にキーワード “umbrella” の検索数が急激に増加している．これは，2007 年 3 月に “umbrella” という楽曲がリリースされたことによるものであり，StreamScope はこのイベントをレジーム#2 として抽出することに成功した．このように，StreamScope は検索キーワードに関する特徴的なイベントをレジームとして検

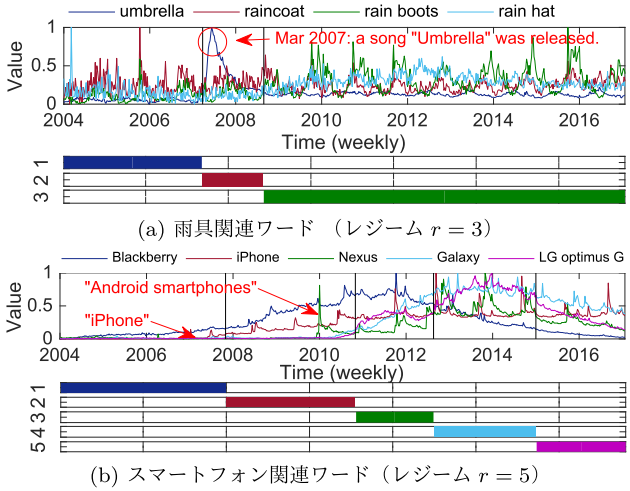


図 5　*GoogleTrend* におけるトレンドの変化点抽出例
Fig. 5　Application examples: StreamScope discovers significant discontinuities.

[*6] http://www.google.com/insights/search/

出することが可能である．

- トレンド発見：図 5 (b) はスマートフォンに関連するキーワード（“iPhone”，“Galaxy” 等）を示す時系列ストリームであり，各キーワードは新機種の発表や発売，クリスマスのように，様々なイベントが検索数の上昇/下降に影響を与える複雑な特徴を持つ．StreamScope は，過去 13 年間のスマートフォンのシェア拡大の様子を 5 つのレジームとして抽出することに成功した．具体的には，(1) スマートフォンの先駆けとなった Blackberry が，欧米のビジネスマンの間で流行．(2) iPhone の発売にともない，スマートフォンが世界で注目される．(3) Nexus，Galaxy 等の Android 搭載端末が市場参入，シェア拡大．(4) iOS 端末と Android 端末の競合．(5) 現在の各スマートフォンの世界シェアの動向，といった 5 つのトレンドをレジームとして抽出した．

図 5 に示すように，StreamScope は Web アクティビティデータを効率的に処理し，ユーザ活動に潜む様々な特徴をオンラインに抽出することが可能である．

8. むすび

本論文では自動パターン検出のためのストリームアルゴリズムとして StreamScope を提案した．StreamScope は，与えられた時系列ストリームに関する事前知識を必要とせず，その中に含まれる特徴的なパターンやトレンド（レジーム）とその変化点を正確に発見することができる．様々な種類の実データを用いて実験を行い，StreamScope はオンライン処理でありながら，その有用性を確認することができた．また，計算コストはデータストリームの長さに依存せず，従来の手法と比較して大幅に性能が向上していることを示した．

謝辞　本研究の一部は JSPS 科研費 JP15H02705, JP17H04681, JP16K12430, JST さきがけ, 総務省 SCOPE（受付番号 162110003），厚生労働科学研究費補助金（H29-ICT-一般-007）および国立研究開発法人日本医療研究開発機構（AMED）の臨床研究等 ICT 基盤構築研究事業の助成を受けたものです．

参考文献

[1] Letchner, J., Ré, C., Balazinska, M. and Philipose, M.: Access Methods for Markovian Streams, *ICDE*, pp.246–257 (2009).
[2] Papadimitriou, S. and Yu, P.S.: Optimal multi-scale patterns in time series streams, *SIGMOD*, pp.647–658 (2006).
[3] Matsubara, Y., Sakurai, Y. and Faloutsos, C.: The Web as a Jungle: Non-Linear Dynamical Systems for Co-evolving Online Activities, *WWW*, pp.721–731 (2015).
[4] Matsubara, Y., Sakurai, Y. and Faloutsos, C.: Non-Linear Mining of Competing Local Activities, *WWW*, pp.737–747 (2016).
[5] Keogh, E.J., Chu, S., Hart, D. and Pazzani, M.J.: An Online Algorithm for Segmenting Time Series, *ICDM*, pp.289–296 (2001).
[6] Matsubara, Y., Sakurai, Y., van Panhuis, W.G. and Faloutsos, C.: FUNNEL: Automatic mining of spatially coevolving epidemics, *KDD*, pp.105–114 (2014).
[7] Zhao, Y., Sundaresan, N., Shen, Z. and Yu, P.S.: Anatomy of a web-scale resale market: A data mining approach, *WWW*, pp.1533–1544 (2013).
[8] Toyoda, M., Sakurai, Y. and Ishikawa, Y.: Pattern discovery in data streams under the time warping distance, *VLDB J.*, Vol.22, No.3, pp.295–318 (2013).
[9] Fox, E.B., Sudderth, E.B., Jordan, M.I. and Willsky, A.S.: Sharing Features among Dynamical Systems with Beta Processes, *NIPS*, pp.549–557 (2009).
[10] Li, L., McCann, J., Pollard, N.S. and Faloutsos, C.: DynaMMo: Mining and summarization of coevolving sequences with missing values, *KDD*, pp.507–516 (2009).
[11] Li, L., Prakash, B.A. and Faloutsos, C.: Parsimonious Linear Fingerprinting for Time Series, *PVLDB*, Vol.3, No.1, pp.385–396 (2010).
[12] Wang, P., Wang, H. and Wang, W.: Finding semantics in time series, *SIGMOD*, pp.385–396 (2011).
[13] Box, G.E., Jenkins, G.M. and Reinsel, G.C.: *Time Series Analysis: Forecasting and Control, 3rd edition*, Prentice Hall, Englewood Cliffs, NJ (1994).
[14] Fujiwara, Y., Sakurai, Y. and Yamamuro, M.: SPIRAL: Efficient and Exact Model Identification for Hidden Markov Models, *KDD*, pp.247–255 (2008).
[15] Sakurai, Y., Matsubara, Y. and Faloutsos, C.: Mining Big Time-series Data on the Web, *WWW, Tutorial*, pp.1029–1032 (2016).
[16] Matsubara, Y., Sakurai, Y. and Faloutsos, C.: AutoPlait: Automatic mining of co-evolving time sequences, *SIGMOD*, pp.193–204 (2014).
[17] Jain, A., Chang, E.Y. and Wang, Y.-F.: Adaptive stream resource management using Kalman Filters, *SIGMOD*, pp.11–22 (2004).
[18] Morales, G.D.F., Bifet, A., Khan, L., Gama, J. and Fan, W.: IoT Big Data Stream Mining, *KDD, Tutorial*, pp.2119–2120 (2016).
[19] Yi, B.-K., Sidiropoulos, N., Johnson, T., Jagadish, H., Faloutsos, C. and Biliris, A.: Online Data Mining for Co-Evolving Time Sequences, *ICDE*, pp.13–22 (2000).
[20] Zhu, Y. and Shasha, D.: StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time, *VLDB*, pp.358–369 (2002).
[21] Iwata, T., Yamada, T., Sakurai, Y. and Ueda, N.: Online Multiscale Dynamic Topic Models, *KDD*, pp.663–672 (2010).
[22] Papadimitriou, S., Sun, J. and Faloutsos, C.: Streaming Pattern Discovery in Multiple Time-Series, *VLDB*, pp.697–708 (2005).
[23] Sakurai, Y., Papadimitriou, S. and Faloutsos, C.: BRAID: Stream Mining through Group Lag Correlations, *Proc. ACM SIGMOD*, Baltimore, Maryland, pp.599–610 (2005).
[24] Matsubara, Y., Sakurai, Y., Ueda, N. and Yoshikawa, M.: Fast and Exact Monitoring of Co-Evolving Data Streams, *ICDM*, pp.390–399 (2014).
[25] Rissanen, J.: A universal prior for integers and estimation by minimum description length, *The Annals of statistics*, pp.416–431 (1983).

川畑 光希

2016 年熊本大学工学部情報電気電子工学科卒業．2016 年より同大学院自然科学研究科情報電気電子工学専攻博士前期課程に在籍．第 8 回データ工学と情報マネジメントに関するフォーラム（DEIM2016）最優秀論文賞，学生プレゼンテーション賞受賞．時系列データストリーム解析の研究に従事．日本データベース学会学生会員．

松原 靖子　（正会員）

2006 年お茶の水女子大学理学部情報科学科卒業．2009 年同大学院博士前期課程修了．2012 年京都大学大学院情報学研究科社会情報学専攻博士後期課程修了．博士（情報学）．2012 年 NTT コミュニケーション科学基礎研究所 RA．2013 年熊本大学大学院自然科学研究科日本学術振興会特別研究員（PD）．2014 年より同大学院助教．この間，カーネギーメロン大学客員研究員．2016 年 12 月より国立研究開発法人科学技術振興機構さきがけ研究員．2016 年度日本データベース学会上林奨励賞，山下記念研究賞受賞．大規模時系列データマイニングに関する研究に従事．ACM，電子情報通信学会，日本データベース学会各会員．

櫻井 保志　（正会員）

1991 年同志社大学工学部電気工学科卒業．1991 年日本電信電話（株）入社．1999 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．博士（工学）．2004～2005 年カーネギーメロン大学客員研究員．2013 年熊本大学大学院自然科学研究科教授．本会平成 18 年度長尾真記念特別賞，平成 16 年度および平成 19 年度論文賞，電子情報通信学会平成 19 年度論文賞，日本データベース学会上林奨励賞，ACM KDD best paper awards（2008，2010）等受賞．データマイニング，データストリーム処理，センサデータ処理，Web 情報解析技術の研究に従事．ACM，電子情報通信学会，日本データベース学会各会員．

（担当編集委員　渡辺 陽介）