# 大規模時系列データからの特徴自動抽出

松原　靖子†　　櫻井　保志†　　ChristosFaloutsos††

† 熊本大学
†† Carnegie Mellon University
E-mail: †{yasuko,yasushi}@cs.kumamoto-u.ac.jp, ††christos@cs.cmu.edu

あらまし　本論文では，大規模時系列データのための特徴自動抽出手法である AutoPlait について述べる．AutoPlait は，様々な時系列パターンを含む複雑なシーケンスが与えられたときに，そのシーケンスデータの中から重要な特徴を発見し，それらの情報を統計的に要約，表現する．実データを用いた実験では，AutoPlait が様々な時系列データの中から有用なパターンを正確に発見することを確認し，さらに，最新の既存手法と比較を行い提案手法が大幅な精度，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　時系列データ，特徴自動抽出

## 1. まえがき

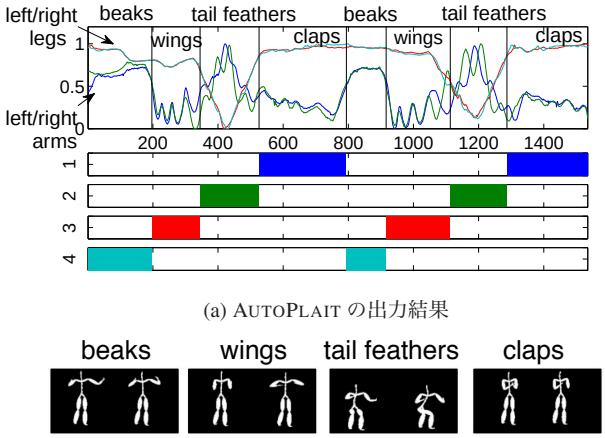
時系列シーケンスは，センサデータや Web アクセス履歴等，様々なアプリケーションにおいて大量に生成されている．これらの大規模な時系列シーケンスの中から，典型的なパターンや異常値を発見することは非常に重要な課題である．本論文では，大規模時系列データを対象とし，重要な時系列パターンの抽出を自動的に行なうことを目的とする．そして，大規模時系列データの中から，異なるトレンドを発見し，すべての時系列パターンを表現する手法として，AutoPlait を提案する．(注1) より具体的には，(a) $\boldsymbol{X}$ の中のパターンの変化点を発見し，部分シーケンス集合（セグメント）に分割し，(b) それらのセグメントをグループ化し，類似時系列パターン（本論文では「レジーム (regime)」と呼ぶ）を発見する．さらに重要な点として，これらの処理は (c) 高速かつ，自動で行う．

具体例．　図 1 (a) は，*MoCap* データにおける「チキンダンス (chicken dance)」(注2) の時系列シーケンスデータと，AutoPlait の出力結果例である．このモーションは，4 次元のシーケンスで構成され，それぞれの次元が，左右の腕と足の加速度を表現している．チキンダンスは，図 1 (b) に示す通り，beaks, wings, tail feathers, claps の 4 つの代表的なステップ から構成される．図 1 (a) の下の段は，AutoPlait が自動抽出した 4 つのレジームを示している．提案手法は，ダンスに含まれる 4 つのステップを抽出し，そして各ステップの切れ目も正しく発見することができる．ここで最も重要なこととして，AutoPlait は，これらの 4 つのステップに関する事前知識を必要とせず，適切な数のレジームとその位置を自動的に把握することができる．

### 1.1 関連研究と自動抽出手法の重要性

時系列データを対象とした研究課題は，数多く存在する．パターン発見 [12], [8], [11], [14]，情報要約 [1], [6], [7]，クラスタリング [4]，セグメンテーション [3], [15] や類似シーケンス探索 [2], [9], [13] 等は重要な課題である．しかし，これらの先行研究は，基本的にすべて，パラメータの設定やチューニングが必要である．例えば，文献 [3], [15] は，セグメントの個数や，エラーの閾値等，いくつかのユーザ指定のパラメータ入力が必要であり，これらのパラメータが出力結果に大きな影響を与える (5. 章を参照)．したがって，理想的にはこれらのパラメータ設定やユーザの介入を必要としない手法が望ましい．

また，さらに重要な問題がビックデータの解析である．時系列データは様々なシステムにおいて大量に発生している．大規模なデータを解析するにあたり，ユーザの手を介したパラメータ設定を行なうことは，多くの時間的コストが必要となるため，現実的ではない．すなわち，ビックデータの解析において，自動処理技術は必要不可欠な重要な要素である．




(a) AutoPlait の出力結果




(b) 「チキンダンス (chicken dance)」における 4 つの代表的なステップ

図 1　*MoCap* データにおける AutoPlait の出力例．

### 1.2 本論文の貢献

AutoPlait は以下の特長がある．

（1） 時系列パターン（レジーム） の個数と種類を把握し，それぞれの適切な変化点を発見する．さらに AutoPlait は，パターン変化点の最適解の検出を保証する．

（2） 3. 章で述べる提案モデル (MLCM) により，ユーザの

(注1)：ソースコード： http://www.cs.kumamoto-u.ac.jp/~yasuko/software.html
(注2)：Chicken dance: http://www.youtube.com/watch?v=6UV3kRV46Zs&t=49s

直感に合致した時系列パターンの抽出を行なう．

（3） AUTOPLAIT はパラメータ設定を必要としない．ユーザの介入を必要とせず，適切なレジームの数，変化点の数を，自動的に発見することができる．

（4） 計算コストは入力データの長さに対して線形である．

## 2. 問題設定

ここでは本論文で必要な概念について定義を行なう．$\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ を $d$ 次元の時系列シーケンスとし，$\boldsymbol{x}_t$ を時刻 $t$ における $d$ 次元ベクトルとする．シーケンス $\boldsymbol{X}$ が与えられたとき，本研究は $\boldsymbol{X}$ を $m$ 個のセグメント集合 $\mathcal{S}$= $\{s_1, \ldots, s_m\}$ に分割することを目的とする．$s_i$ 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．本研究ではさらに，発見したセグメント集合を類似セグメントのグループ（レジーム: regime ）に分類する．

［定義 **1**］（レジーム） $r$ を最適なセグメントグループの個数とする．それぞれのセグメント $s$ はセグメントグループの 1 つに割り当てられる．これらグループをレジーム (regime) と呼び，それぞれのレジームは統計モデル $\boldsymbol{\theta}_i$ $(i = 1, \ldots, r)$ として表現される．

例えば，図 1 において，モーションは $m = 8$ 個のセグメントから構成され，各々のセグメントは $r = 4$ 個のレジーム (beaks, winds, tail feathers, claps) のうちの 1 つに割り当てられる．

［定義 **2**］（セグメントメンバーシップ） $\mathcal{F} = \{f_1, \ldots, f_m\}$ を，$m$ 個の整数列とし，$f_i$ を $i$ 番目のセグメントが所属するレジームの番号とする $(1 \leqq f_i \leqq r)$.

図 1 では，1 番目のセグメントは 4 番目のレジーム (beaks) に，2 番目のセグメントは 3 番目のレジーム (wings) にそれぞれ所属する．つまり，この場合のセグメントメンバーシップは $\mathcal{F}$ =$\{4, 3, 2, 1, 4, 3, 2, 1\}$ となる．

本研究の目的は，大規模時系列シーケンスが与えられたときに，そのシーケンスのセグメント化と分割位置の検出及び，レジームの発見を高速かつ自動で行なうことである．本論文で取り組む問題を以下のように定義する．

［ 問題 1］ 多次元時系列シーケンス $\boldsymbol{X}$ が与えられたとき，$\boldsymbol{X}$ を表現するような以下の情報を抽出する．

（1） セグメントの総数 $m$ と各セグメントの位置：
$\mathcal{S} = \{s_1, \ldots, s_m\}$

（2） レジームの総数 $r$ とセグメントメンバーシップ：
$\mathcal{F} = \{f_1, \ldots, f_m\}$

（3） r 個のレジームを表現するモデルのパラメータ集合：
$\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, \boldsymbol{\Delta}_{r \times r}\}$

これらの情報はコスト関数（式 (5)）を最小化するものを選ぶ．

本論文では，レジームを表現するモデルパラメータ集合 $\boldsymbol{\Theta}$ を，$r$ 個の隠れマルコフモデル (HMM: hidden Markov model), $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r\}$，として表現する．(注3) ここでさらに，新たな概念として，レジーム遷移行列 $\boldsymbol{\Delta}_{r \times r}$ を導入する．レジーム遷移行列（定義 4）とコスト関数（式 (5)）についての詳細は，3. 章において示す．

問題 1 で示した通り，本論文の目的は，$\boldsymbol{X}$ の特徴を抽出し，すべての時系列パターンを表現するパラメータ集合 $\{m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F}\}$ を発見することである．ここで，この全パラメータ集合を候補解 $\mathcal{C}$ と呼ぶ．

［定義 **3**］ $\boldsymbol{X}$ を表現する全パラメータ集合 $\mathcal{C} = \{m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F}\}$ を候補解と呼ぶ．候補解 $\mathcal{C}$ は，セグメント集合，各セグメントのレジームへの割当て，レジームを表現する確率モデル，これらすべてを表現する．

結論として，本論文の目的は最適な解 $\mathcal{C}$ を発見することである．ここで非常に重要な課題は，(a) どのようにセグメントおよびレジームの数を推定するか，(b) どのようにレジームを表現し，セグメントの割当てを行なうかである．本研究では，ユーザによるパラメータ設定を介せず，自動処理によって最適解を求めるための新手法を提案する．

## 3. データ圧縮と情報要約

本章では，問題 1 を解決するためのモデルを提案する．

提案モデルは以下の 2 つのアイデアに基づく．

（1） 多階層連鎖モデル (**MLCM: multi-level chain model**): 複数のレジーム間の時系列パターンとその遷移を表現するために，多層的な連鎖モデル (MLCM) を提案する．図 2 は提案モデルの概念図である．青いセル (state 1,2,3) はレジーム 1 に所属する隠れマルコフモデルの状態遷移を表現し，赤いセルはレジーム 2 に該当する．各レジーム内部においてそれぞれ遷移行列 **A** を持つ．例えばレジームはそれぞれ図 1 のダンスにおける beaks と wings のステップに相当する．ここで，提案モデルの重要な要素として，それぞれのレジーム間のレジーム遷移確率 $(\boldsymbol{\Delta}_{r \times r})$ の概念を導入する．レジーム遷移確率は $r \times r$ の行列として表現され，$r$ 個のレジーム間の遷移を表現する（図 2 の例では，$r = 2$ となる）．

（2） モデル表現コスト: セグメントとレジームの発見のために，最小記述長 (MDL: minimum description length) の概念を用いる．MDL は情報理論に基づくモデル選択基準のひとつであり，可逆圧縮を行なうことができるが，そのものの概念だけでは本論文の目的を直接解決することはできない．そこで，与えられたシーケンス $\boldsymbol{X}$ を適切に表現するモデルを見つけるために，新しい符号体系を定義する．具体的には，(a) 新たな関数 (式 (5)) を用いて候補解 $\mathcal{C}$ のモデルコストを推定し，(b) 最適解を発見するための効果的なアルゴリズムを提案する．

### 3.1 MLCM: 多階層連鎖モデル

多階層連鎖モデル (MLCM: multi-level chain model) は，図 2 にあるように，隠れマルコフの状態遷移をレジームにグループ化し，階層的な時系列パターンの遷移を表現する．本論文では，これ以降，主に 2 層の遷移について焦点を当てるが，2 層以上の多層遷移を表現することも可能である．ここで，図 2 を用いて MLCM の説明を行なう．図は，総計 5 つの状態 (state) から構成される連鎖モデルであるが，ここでは従来の HMM のように，$5 \times 5$ の遷移行列を用いるのではなく，上位層の状態

(注3)：本論文で提案する枠組みは，HMM 以外の時系列モデルに適用することも可能である．

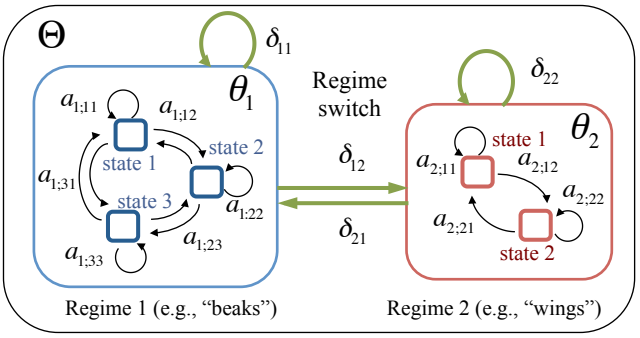



図 2　多階層連鎖モデル $\boldsymbol{\Theta}$ （ここでは $r = 2$）.

(super-state) の概念を導入することによって，パターンのグループ化を行なう．ここで，このグループを「レジーム」と呼ぶ．例えば，図 2 では，3 つの青い状態を用いて beaks のステップを表現し，残りの 2 つの赤い状態を用いて wings のステップを表現する．ここで，各レジームは内部に遷移行列 ($a_{1;ji} \in \mathbf{A}_1$, $a_{2;ji} \in \mathbf{A}_2$) を保持し，それらは黒い矢印で表される．さらに，上位層ではレジーム間の遷移行列 ($\delta_{vu} \in \boldsymbol{\Delta}_{2\times 2}$) を保有し，これらは緑の矢印で示されている．

[定義 4]（レジーム遷移行列） $\boldsymbol{\Delta}_{r\times r}$ を $r$ 個のレジーム群の遷移行列と呼ぶ．ここで，要素 $\delta_{ij} \in \boldsymbol{\Delta}$ は $i$ 番目のレジームから $j$ 番目のレジームへの遷移確率を示す．

行列 $\boldsymbol{\Delta}$ 内部の要素 $\delta_{i,j}$ は確率を表し，$0 \leqq \delta_{ij} \leqq 1$, $\sum_j \delta_{ij} = 1$ という条件を持つ．そして提案モデルは $r$ 個のレジーム集合 $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, \boldsymbol{\Delta}_{r\times r}\}$ で表現され，$\boldsymbol{\theta}_i$ は $i$ 番目のレジームのモデルパラメータを表現する．ここで，$\boldsymbol{\theta}_i$ は HMM に基づき，初期確率，遷移確率，出力確率の三つ組で次のように表現される：$\boldsymbol{\theta}_i = \{\boldsymbol{\pi}_i, \mathbf{A}_i, \mathbf{B}_i\}$. (注4)

### 3.2　特徴抽出とデータ圧縮

次に，大規模時系列データを表現するための符号化スキームを導入する．直感的には，データが与えられたときのモデルのよさは次の式で表現できる：$Cost_T = Cost(\mathcal{M}) + Cost(\boldsymbol{X}|\mathcal{M})$. ここで，$Cost(\mathcal{M})$ はモデル $\mathcal{M}$ を表現するためのコストを示し，$Cost(\boldsymbol{X}|\mathcal{M})$ は，$\mathcal{M}$ が与えられたときのデータ $\boldsymbol{X}$ の符号化のコストを示す．

#### 3.2.1　モデル表現コスト

提案モデルの表現コストは以下の要素から構成される．

- 多次元シーケンスデータの長さ $n$ と次元数 $d$：$\log^*(n) + \log^*(d)$ ビット (注5)
- セグメントとレジームの個数 $m, r$：$\log^*(m) + \log^*(r)$
- 各セグメントのレジームへの割当て（セグメントメンバーシップ）：$m \log(r)$ ビット
- 各セグメントの長さ $s$：$\sum_{i=1}^{m-1} \log^* |s_i|$ ビット
- $r$ 個のレジームのモデルパラメータ集合：$Cost_M(\boldsymbol{\Theta})$

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

(注4)：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 $\mathbf{B} = \{\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)\}_{i=1}^k$）.
(注5)：ここで， $\log^*$ は整数のユニバーサル符号長を表す：$\log^*(x) \approx \log_2(x) + \log_2 \log_2(x) + \ldots$. [10].

単一のレジームのモデル $\boldsymbol{\theta}$ は，状態数 $k$ （$\log^*(k)$）と確率モデル ($\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}\}$) の表現コストが必要となる．まとめると，

$$Cost_M(\boldsymbol{\theta}) = \log^*(k) + c_F \cdot (k + k^2 + 2kd). \tag{2}$$

ここで，$c_F$ は浮動小数点のコストを示す．(注6) 同様にして，レジーム遷移行列には，$Cost_M(\boldsymbol{\Delta}) = c_F \cdot r^2$ のコストを要する．

#### 3.2.2　時系列シーケンスの符号化コスト

先述の通り，本論文では MLCM モデルを用いてシーケンス $\boldsymbol{X}$ の時系列パターンを表現するが，ここで重要なのは，推定したモデルが $\boldsymbol{X}$ を正しく表現しているかを判断する指標の導入である．ハフマン符号を用いた情報圧縮では，モデル $\boldsymbol{\theta}$ が与えられた際の $\boldsymbol{X}$ の符号化コストを負の対数尤度を用いて次のように表現することができる．

$$Cost_C(\boldsymbol{X}|\boldsymbol{\theta}) = \log_2 \frac{1}{P(\boldsymbol{X}|\boldsymbol{\theta})} = -\ln P(\boldsymbol{X}|\boldsymbol{\theta}). \tag{3}$$

ここで，$P(\boldsymbol{X}|\boldsymbol{\theta})$ は $\boldsymbol{X}$ の尤度を示す．シーケンス $\boldsymbol{X}$ と $r$ 個のレジームのモデルパラメータ集合 $\boldsymbol{\Theta}$ が与えられたとき，データ圧縮のためのコストの総数は次の通りである．

$$\begin{aligned} Cost_C(\boldsymbol{X}|\boldsymbol{\Theta}) &= \sum_{i=1}^{m} Cost_C(\boldsymbol{X}[s_i]|\boldsymbol{\Theta}) \\ &= \sum_{i=1}^{m} -\ln(\delta_{vu} \cdot (\delta_{uu})^{|s_i|-1} \cdot P(\boldsymbol{X}[s_i]|\boldsymbol{\theta}_u)), \end{aligned} \tag{4}$$

ここで，$i$ と $(i-1)$ 番目のセグメントはそれぞれ $u$ と $v$ 番目のレジームに所属し，$f_i = u, f_{i-1} = v, f_0 = f_1$ とする．$\boldsymbol{X}[s_i]$ はセグメント $s_i$ の部分シーケンス，$P(\boldsymbol{X}[s_i]|\boldsymbol{\theta}_u)$ はセグメント $s_i$ の尤度，$\boldsymbol{\theta}_u$ はセグメント $s_i$ が所属するレジームである．

#### 3.2.3　符号化コスト関数

まとめると，候補解 $\mathcal{C} = \{m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F}\}$ が与えられたときの $\boldsymbol{X}$ の符号長は次のように表現される．

$$\begin{aligned} Cost_T(\boldsymbol{X}; \mathcal{C}) &= Cost_T(\boldsymbol{X}; m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F}) \\ &= \log^*(n) + \log^*(d) + \log^*(m) + \log^*(r) + m\log(r) \\ &\quad + \sum_{i=1}^{m-1} \log^* |s_i| + Cost_M(\boldsymbol{\Theta}) + Cost_C(\boldsymbol{X}|\boldsymbol{\Theta}) \end{aligned} \tag{5}$$

本論文の次の目標は，上記のコスト関数を最小化するようなセグメントおよびレジーム集合を発見することである．

## 4.　最適化アルゴリズム

前章では，候補解 $\mathcal{C} = \{m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F}\}$ が与えられた上でシーケンス $\boldsymbol{X}$ を表現するためのコスト関数として，式 (5) を示した．続いて本章では，式 (5) に基づき，最適な解 $\mathcal{C}$ を発見するためのアルゴリズムを提案する．

### 4.1　概　　要

本研究では，前章で述べたコストモデルに基づき，セグメントおよびレジームの個数を自動的に選択する．直感的には，データの圧縮率が高ければ，そのモデルはデータに含まれるパ

(注6)：本論文では $4 \times 8$ ビットとする．

表 1　主な記号と定義.

| 記号 | 定義 |
|---|---|
| シーケンス | |
| $n$ | 時系列の長さ |
| $d$ | 時系列の次元数 |
| $\boldsymbol{X}$ | $d$ 次元の時系列シーケンス |
| セグメント | |
| $m$ | $\boldsymbol{X}$ に含まれるセグメントの総数 |
| $\mathcal{S}$ | $\boldsymbol{X}$ に含まれるセグメント集合：$\mathcal{S}=\{s_1,\ldots,s_m\}$ |
| $\mathcal{F}$ | セグメントメンバーシップ：$\mathcal{F}=\{f_1,\ldots,f_m\}$ |
| レジーム | |
| $r$ | $\boldsymbol{X}$ に含まれるレジームの総数 |
| $\boldsymbol{\Theta}$ | $r$ 個のレジームのモデルパラメータ集合：$\boldsymbol{\Theta}=\{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_r,\boldsymbol{\Delta}_{r\times r}\}$ |
| $\boldsymbol{\theta}_i$ | $i$ 番目の レジームのモデルパラメータ |
| $k_i$ | $\boldsymbol{\theta}_i$ の状態数 |
| $\boldsymbol{\Delta}_{r\times r}$ | レジーム遷移行列：$\boldsymbol{\Delta}=\{\delta_{ij}\}_{i,j=1}^{r}$ |
| コスト関数 | |
| $\mathcal{C}$ | 候補解：$\mathcal{C}=\{m,r,\mathcal{S},\boldsymbol{\Theta},\mathcal{F}\}$ |
| $Cost_M(\boldsymbol{\Theta})$ | $\boldsymbol{\Theta}$ のモデル表現コスト |
| $Cost_C(\boldsymbol{X}\|\boldsymbol{\Theta})$ | $\boldsymbol{\Theta}$ による $\boldsymbol{X}$ の符号化コスト |
| $Cost_T(\boldsymbol{X};\mathcal{C})$ | $\mathcal{C}$ による $\boldsymbol{X}$ の総コスト |

ターンをよく表現しているといえる．つまり，セグメントの個数 $m$，レジームの個数 $r$，モデルパラメータ集合 $\boldsymbol{\Theta}$，そしてメンバーシップ $\mathcal{F}$ から構成される候補解 $\mathcal{C}$ に対し，$\boldsymbol{X}$ の符号化コスト $Cost_T(\boldsymbol{X};m,r,\mathcal{S},\boldsymbol{\Theta},\mathcal{F})$ が最小となるとき，$\mathcal{C}$ は最適なモデルになる．

次に，具体的な最適化手法を示す．ここでは問題を簡略化するため，次に挙げる 3 つの部分問題に分割する．(1) レジームの個数を $r=2$ に固定し，各レジームのモデルパラメータも与えられる場合を考える．(2) $r=2$ を固定した状態で，モデルパラメータの推定を行う．(3) 最適なレジームの個数を推定する．より具体的には各部分問題に対応し，以下の 3 つのアルゴリズムを提案する．

（1） **CutPointSearch**: レジームの個数 ($r=2$) とモデルパラメータが与えられたときに，$\boldsymbol{X}$ を 2 つのレジームに分割し，それぞれのセグメントの分割位置を検出する．

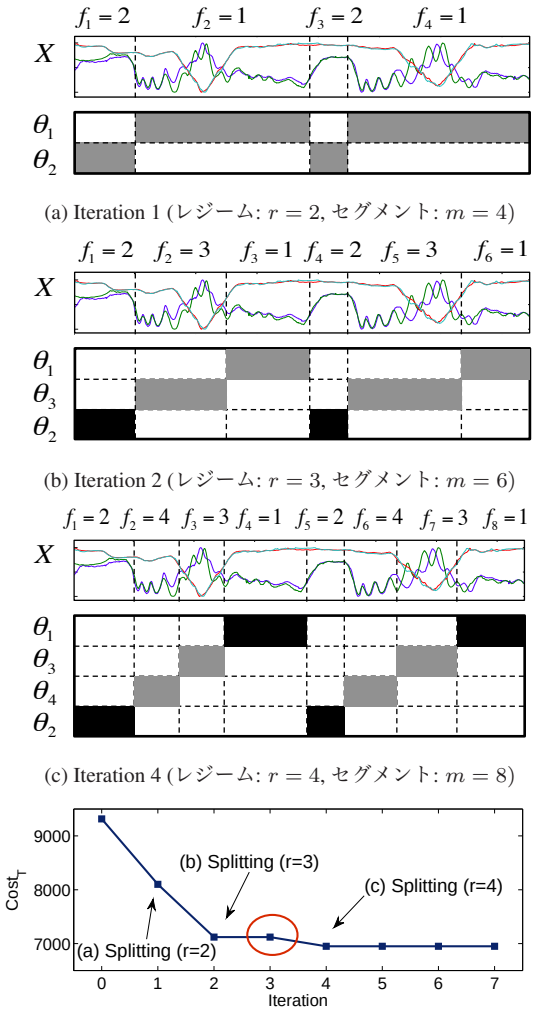
（2） **RegimeSplit**: レジームの個数 $r=2$ が与えられたときに，2 つのレジームを表現するモデルパラメータ $(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\Delta})$ を推定する．

（3） **AutoPlait**: 最適なレジームの個数 ($r=2,3,4,\ldots$) を求める．

図 3 は，AutoPlait の処理のながれである．AutoPlait は $r=2$ から開始し，セグメントとレジームを分割しながらシーケンス $\boldsymbol{X}$ を適切に表現する解 $\mathcal{C}$ を発見する．レジームとセグメントを分割するとき，コスト関数（図 3 (d)）が下がる．例外として，図中における赤丸の箇所 (Iteration 3) については，レジーム $\boldsymbol{\theta}_1$ が分割しないため，コストが下がらない．

### 4.2 Cut-point search

まず最も単純な部分問題として，シーケンス $\boldsymbol{X}$ と，2 つのレジームのモデルパラメータ $\{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\Delta}\}$ が与えられている場合を考える．CutPointSearch は レジームのモデルパラメータに基づき，$\boldsymbol{X}$ のパターンの変化点（つまりセグメントの分割位置）を検出することができる．ここで重要な点として，提案アルゴリズムは探索漏れがないことを保証しながらも，高速かつ単一の走査によって，最適なレジーム変化点の個数と位置を検出することができる．図 4 はレジームの変化点が 1 つの場合における CutPointSearch の処理の様子を示している．これは青い




(a) Iteration 1 (レジーム: $r=2$, セグメント: $m=4$)

(b) Iteration 2 (レジーム: $r=3$, セグメント: $m=6$)

(c) Iteration 4 (レジーム: $r=4$, セグメント: $m=8$)

(d) 総コストの変化.

図 3　AutoPlait の概要図：AutoPlait は $\boldsymbol{X}$ が与えられたとき，反復処理により適切なセグメント/レジームの個数を求める．

レジーム $\boldsymbol{\theta}_1$ から赤いレジーム $\boldsymbol{\theta}_2$ へ切り替わる例である．

より具体的には，ここでは $\boldsymbol{X}$ が与えられたとき，コスト関数（式 (4)）を最小とするような，複数のレジーム変化点を発見し，それらを 2 つのセグメント集合 $(\mathcal{S}_1,\mathcal{S}_2)$ に分割することを考える．ここで，2 つのセグメント集合は，偶数番目のセグメントは 1 つ目のレジームに，奇数番目は 2 つ目のレジームに所属する．例えば，図 3(a) では $\mathcal{S}_1=\{s_2,s_4\}$ は $\boldsymbol{\theta}_1$ に，$\mathcal{S}_2=\{s_1,s_3\}$ は $\boldsymbol{\theta}_2$ にそれぞれ所属する．

ここで，複数の変化点を検出するために，多階層連鎖モデル (MLCM) の概念を用いる．図 4 のように 2 つのレジーム $\boldsymbol{\theta}_1,\boldsymbol{\theta}_2$ が与えられているとする．ここで，この 2 つのレジームは相互に遷移確率を持つ．モデルが与えられた上での符号化コスト $Cost_C(\boldsymbol{X}|\boldsymbol{\Theta})=-\ln P(\boldsymbol{X}|\boldsymbol{\Theta})$ を計算するために，本論文では動的計画法に基づくアルゴリズムを提案する．

#### 4.2.1 アルゴリズム

シーケンス $\boldsymbol{X}$ と 2 つの レジーム $\boldsymbol{\theta}_1=\{\boldsymbol{\pi}_1,\mathbf{A}_1,\mathbf{B}_1\}$, $\boldsymbol{\theta}_2=\{\boldsymbol{\pi}_2,\mathbf{A}_2,\mathbf{B}_2\}$, およびレジーム遷移行列 $\boldsymbol{\Delta}=\{\delta_{11},\delta_{12},\delta_{21},\delta_{22}\}$ が与えられたとき，$\boldsymbol{X}$ の尤度 $P(\boldsymbol{X}|\boldsymbol{\Theta})$ は次のように計算される．

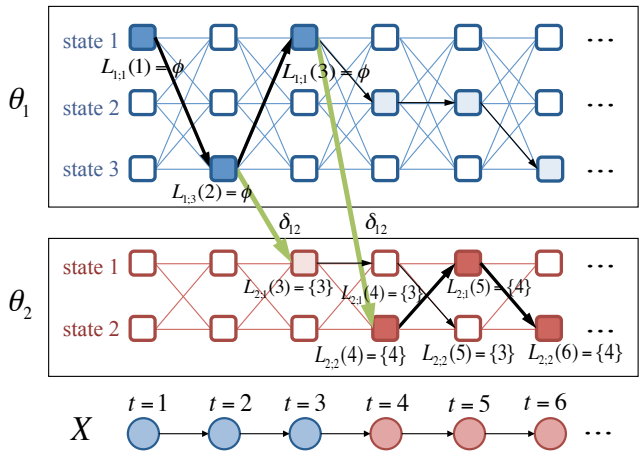



図 4　CutPointSearch の様子．$\boldsymbol{X}$ と $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ が与えられたとき，単一の走査でレジームの変化点 ($t = 4$) を検出する．

$$P(\boldsymbol{X}|\boldsymbol{\Theta}) = \max \begin{cases} \max_{1 \leqq i \leqq k_1} \{p_{1;i}(n)\} & // \text{ regime } \boldsymbol{\theta}_1 \\ \max_{1 \leqq u \leqq k_2} \{p_{2;u}(n)\} & // \text{ regime } \boldsymbol{\theta}_2 \end{cases} \tag{6}$$

$$p_{1;i}(t) = \max \begin{cases} \delta_{21} \cdot \max_v \{p_{2;v}(t-1)\} \cdot \pi_{1;i} \cdot b_{1;i}(\boldsymbol{x}_t) \\ \quad // \text{ regime switch from } \boldsymbol{\theta}_2 \text{ to } \boldsymbol{\theta}_1 \\ \delta_{11} \cdot \max_j \{p_{1;j}(t-1) \cdot a_{1;ji}\} \cdot b_{1;i}(\boldsymbol{x}_t) \\ \quad // \text{ staying at regime } \boldsymbol{\theta}_1 \end{cases} \tag{7}$$

$$p_{2;u}(t) = \max \begin{cases} \delta_{12} \cdot \max_j \{p_{1;j}(t-1)\} \cdot \pi_{2;u} \cdot b_{2;u}(\boldsymbol{x}_t) \\ \quad // \text{ regime switch from } \boldsymbol{\theta}_1 \text{ to } \boldsymbol{\theta}_2 \\ \delta_{22} \cdot \max_v \{p_{2;v}(t-1) \cdot a_{2;vu}\} \cdot b_{2;u}(\boldsymbol{x}_t) \\ \quad // \text{ staying at regime } \boldsymbol{\theta}_2 \end{cases} \tag{8}$$

ここで，$p_{1;i}(t)$ は，時刻 $t$ におけるレジーム $\boldsymbol{\theta}_1$ の状態 $i$ の確率の最大値であり，$p_{2;u}(t)$ は，時刻 $t$ における $\boldsymbol{\theta}_2$ の状態 $u$ の確率の最大値を示す．時刻 $t = 1$ においては，各レジームの確率は次のように計算する．

$$\begin{aligned} p_{1;i}(1) &= \delta_{11} \cdot \pi_{1;i} \cdot b_{1;i}(\boldsymbol{x}_1) \\ p_{2;u}(1) &= \delta_{22} \cdot \pi_{2;u} \cdot b_{2;u}(\boldsymbol{x}_1) \end{aligned} \tag{9}$$

式 (7) の上段は，レジーム間の切り替え（$\boldsymbol{\theta}_2$ から $\boldsymbol{\theta}_1$）の確率，下段はレジーム $\boldsymbol{\theta}_1$ の内部の状態遷移の確率を示す．$\pi_{1,i}, b_{1;i}(\boldsymbol{x}_t)$, $a_{1;ji}$ はそれぞれ $\boldsymbol{\theta}_1$ 内における状態 $i$ の初期確率，出力確率，状態 $j$ から状態 $i$ への遷移確率を示す．

アルゴリズム 1 は CutPointSearch の具体的な処理を示す．ここで $\mathcal{L}_{1;i}(t)$ は時刻 $t$ における $\boldsymbol{\theta}_1$ の状態 $i$ の変化点集合であり，$\mathcal{L}_{2;u}(t)$ は $\boldsymbol{\theta}_2$ の状態 $u$ の変化点集合である．これらは式 (7), (8) の尤度計算に基づき更新される．変化点は，レジームが他方に切り替わった場合，その時刻 $t$ を変化点の候補として集合に加える（詳細は文献 [5] 参照）．CutPointSearch は時刻 $t = n$ において $\mathcal{L}_{1;i}(n)$ と $\mathcal{L}_{2;u}(n)$ 内のすべての状態 $i$,$u$ の中から，尤度 $P(\boldsymbol{X}|\boldsymbol{\Theta})$ を最大化するような最適解 $\mathcal{L}_{best}$ を選出する．

［例 1］　ここでは図 4 を用いてアルゴリズムの具体的な例を示す．時刻 $t = 1$ と $t = 2$ において，$\boldsymbol{\theta}_1$ 内の確率 $p_{1;1}(1), p_{1;3}(2)$ はそれぞれ確率の最大値を持つ．時刻 $t = 3$ において CutPointSearch は $p_{1;3}(2)$ から $p_{2;1}(3)$ への変化点の候補を発見する（図では $\boldsymbol{\theta}_1$ から $\boldsymbol{\theta}_2$ への緑の矢印）．同時に，$\mathcal{L}_{2;1}(3) = \{3\}$ を候補集合として更新する．同様にして，2 番目の候補点として $\mathcal{L}_{2;2}(4) = \{4\}$ を保持する．時刻 $t = 6$ において，アルゴリズムは $p_{2;2}(6)$ が確率の最大値を持つことを明らかにし，$\mathcal{L}_{2;2}(6) = \{4\}$ を最適解として出力する．

**Algorithm 1** CutPointSearch ($\boldsymbol{X}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\Delta}$)

**Input:** Sequence $\boldsymbol{X}$, model parameters of two regimes $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\Delta}\}$
**Output:** (a) Number of segments assigned to each regime, $m_1, m_2$
　　(b) Segment sets of two regimes $\mathcal{S}_1, \mathcal{S}_2$
/* Compute $p_{1;i}(t)$ and $p_{2;u}(t)$ */
**for** $t = 1 : n$ **do**
　Compute $p_{1;i}(t)$ for state $i = 1, \ldots, k_1$;　/* Equations 7 and 9 */
　Compute $p_{2;u}(t)$ for state $u = 1, \ldots, k_2$;　/* Equations 8 and 9 */
　Update $\mathcal{L}_{1;i}(t)$ for state $i = 1, \ldots, k_1$;
　Update $\mathcal{L}_{2;u}(t)$ for state $u = 1, \ldots, k_2$;
**end for**
/* Divide into two sets of segments $\mathcal{S}_1, \mathcal{S}_2$ */
Choose the best cut-point set $\mathcal{L}_{best}$;
$t_s = 1$; /* Starting position of first segment */
**for** each cut point $l_i$ in $\mathcal{L}_{best}$ **do**
　Create segment $s_i = \{t_s, l_i\}$;
　**if** $i$ is odd **then**
　　Add $s_i$ into $\mathcal{S}_1$;　$m_1 = m_1 + 1$;
　**else**
　　Add $s_i$ into $\mathcal{S}_2$;　$m_2 = m_2 + 1$;
　**end if**
　$t_s = l_i$;
**end for**
**return** $\{m_1, m_2, \mathcal{S}_1, \mathcal{S}_2\}$;

#### 4.2.2　理論的な分析

［補助定理 1］　CutPointSearch は $O(ndk^2)$ の計算量を要する．

［補助定理 2］　与えられたモデル $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, \boldsymbol{\Delta}\}$ に対し，CutPointSearch は最適な変化点集合を検出することを保証する．

本論文ではスペースの都合で証明は省略する（文献 [5] 参照）．

### 4.3　RegimeSplit

次にモデルパラメータの推定について述べる．具体的には，(a) 2 つのレジームのモデルパラメータを推定し，同時に，(b) すべてのレジーム変化点を検出する．本研究では，式 (5) を用いてシーケンス $\boldsymbol{X}$ の表現コストを最小にするようなモデルパラメータの推定を行なう．アルゴリズム 2 は RegimeSplit の処理を示す．提案アルゴリズムは以下に示す 2 つのステップから構成される反復処理によって，モデルパラメータを求める．

- ステップ *1:*　CutPointSearch （アルゴリズム 1）を利用し，符号化コストが最小となるレジーム変化点を検出し，セグメント集合を 2 つのグループ $\{\mathcal{S}_1, \mathcal{S}_2\}$ に分割する．
- ステップ *2:*　ステップ 1 で得られたセグメント集合に基づき，2 つのレジームのモデルパラメータ $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\Delta}\}$ を推定する．ここで，HMM のパラメータの学習には，Baum-Welch アルゴリズムを用いる．

**モデルパラメータの推定．** HMM のモデルパラメータの推定については，モデル $\boldsymbol{\theta}$ に対し，隠れ状態の数 $k$ を与える必要があるが，この $k$ を手動で設定するのは非常に難しい．もし $k$ の値を小さくすれば，データの表現能力が低くなり，適切なセグメントおよびレジームを求めることが困難となる．一方で，もし $k$ を大幅に上げてしまうと，オーバーフィッティングを招く．

**Algorithm 2** RegimeSplit ($\boldsymbol{X}$)

**Input:** Sequence $\boldsymbol{X}$
**Output:** (a) Number of segments assigned to each regime, $m_1, m_2$
(b) Segment sets of two regimes, $\mathcal{S}_1, \mathcal{S}_2$
(c) Model parameters of two regimes $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\Delta}\}$
Initialize models $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$;
**while** improving the cost **do**
  /* Find segments (phase 1) */
  $\{m_1, m_2, \mathcal{S}_1, \mathcal{S}_2\}$ = CutPointSearch ($\boldsymbol{X}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\Delta}$);
  /* Update model parameters (phase 2) */
  $\boldsymbol{\theta}_1$ = BaumWelch ($\boldsymbol{X}[\mathcal{S}_1]$);
  $\boldsymbol{\theta}_2$ = BaumWelch ($\boldsymbol{X}[\mathcal{S}_2]$);
  Update regime transitions $\boldsymbol{\Delta}$;  /* Equation 10 */
**end while**
**return** $\{m_1, m_2, \mathcal{S}_1, \mathcal{S}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\Delta}\}$;

そこで本研究では，隠れ状態の個数を $k = 1, 2, 3, \ldots$ のように変化させながら，コスト関数 $Cost_M(\boldsymbol{\theta}) + Cost_C(\boldsymbol{X}[\mathcal{S}]|\boldsymbol{\theta})$ が最小となるような $k$ を求める．

さらに，本研究ではレジーム遷移確率 $\boldsymbol{\Delta}$ についても符号化コストを最小にする必要がある．そこで，セグメントの変化点集合 $\{\mathcal{S}_1, \mathcal{S}_2\}$ とモデルパラメータ $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ が与えられたときのレジーム遷移確率 $\boldsymbol{\Delta} = \{\delta_{11}, \delta_{12}, \delta_{21}, \delta_{22}\}$ をラグランジュ乗数法に基づき次のように計算する．

$$\delta_{11} = \frac{\sum_{s \in \mathcal{S}_1} |s| - N_{12}}{\sum_{s \in \mathcal{S}_1} |s|}, \quad \delta_{12} = \frac{N_{12}}{\sum_{s \in \mathcal{S}_1} |s|}, \tag{10}$$

ここで $\sum_{s \in \mathcal{S}_1} |s|$ はレジーム $\boldsymbol{\theta}_1$ に所属するセグメントの長さの総和を示し，$N_{12}$ は $\boldsymbol{\theta}_1$ から $\boldsymbol{\theta}_2$ へのレジームの切替回数を示す．$\delta_{21}, \delta_{22}$ についても同様に計算できる．

また本研究ではモデルパラメータの適切な初期値を求めるために，サンプリングに基づく手法 [5] を用いている．

### 4.4 AutoPlait

本論文の最終目標は，問題 1 で述べた通り，大規模時系列データ の中から任意の数のパターンを自動的に抽出することである．ここで解決すべき問題は，(a) 最適なセグメントおよびレジームの個数 $m, r$ はどのように決定すればよいか，(b) それぞれのセグメントを適切なレジームに割り当てるにはどうしたらいいか，の 2 点である．

#### 4.4.1 アルゴリズム

大規模時系列シーケンスの中から自動的にパターンを取り出す手法として AUTOPLAIT を提案する．AUTOPLAIT はスタックを用いた手法であり．貪欲法に基づくアルゴリズムである．AUTOPLAIT は与えられたシーケンスをセグメントに分割し，新たなレジームを生成し，コスト関数である式 (5) を減少させていく．アルゴリズム 3 は AUTOPLAIT の処理の流れを示している．各ステップにおいて，AUTOPLAIT はスタック $\mathcal{Q}$ の中からエントリ $\{\boldsymbol{\theta}_0, m_0, \mathcal{S}_0\}$ を取り出す．続いて，現在のレジーム $\boldsymbol{\theta}_0$ の分割を試み，新たなレジームの候補ペア $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ とそのセグメント集合 $\{\mathcal{S}_1, \mathcal{S}_2\}$ を生成する．もし新しいレジームの候補のコストが現在のレジームのコストより低い場合は（つまり，レジームの候補ペアが勝った場合），AUTOPLAIT は候補ペアをスタック $\mathcal{Q}$ に追加する．もし現在のレジームのコストのほうが低ければ，エントリをスタックから取り除き，$\{\boldsymbol{\theta}_0, m_0, \mathcal{S}_0\}$ を出力する．これらの処理をスタックが空になるまで繰り返す．

**Algorithm 3** AUTOPLAIT ($\boldsymbol{X}$)

**Input:** Sequence $\boldsymbol{X}$
**Output:** Complete set of parameters $\mathcal{C}$, i.e.,
(a) Number of segments, $m$
(b) Number of regimes, $r$
(c) Segment set $\mathcal{S} = \{s_1, \ldots, s_m\}$
(d) Model parameters of regimes $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r; \boldsymbol{\Delta}\}$
(e) Segment membership $\mathcal{F} = \{f_1, \ldots, f_m\}$
$\mathcal{Q} = \emptyset$;  /* $\mathcal{Q}$: stack for number of segments, segment set, regime */
$\mathcal{S} = \emptyset$;  $m = 0$;  $r = 0$;  $\mathcal{S}_0 = \{1, n\}$;  $m_0 = 1$;
$\boldsymbol{\theta}_0$ = BaumWelch ($\boldsymbol{X}[\mathcal{S}_0]$);  /* Estimate model $\boldsymbol{\theta}_0$ of $\mathcal{S}_0$ */
Push an entry $\{m_0, \mathcal{S}_0, \boldsymbol{\theta}_0\}$ into $\mathcal{Q}$;
**while** stack $\mathcal{Q} \neq \emptyset$ **do**
  Pop an entry $\{\boldsymbol{\theta}_0, m_0, \mathcal{S}_0\}$ from $\mathcal{Q}$;
  /* Try to refine a regime */
  $\{m_1, m_2, \mathcal{S}_1, \mathcal{S}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\Delta}\}$ = RegimeSplit ($\boldsymbol{X}[\mathcal{S}_0]$);
  /* Compare single regime $\boldsymbol{\theta}_0$ v.s. regime pair $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ */
  **if** $Cost_T(\boldsymbol{X}; \mathcal{S}_0, \boldsymbol{\theta}_0) > Cost_T(\boldsymbol{X}; \mathcal{S}_1, \mathcal{S}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ **then**
    /* Regime pair win - split regime */
    Push entries $\{m_1, \mathcal{S}_1, \boldsymbol{\theta}_1\}, \{m_2, \mathcal{S}_2, \boldsymbol{\theta}_2\}$ into $\mathcal{Q}$;
  **else**
    /* Single regime win - no more split, leave it out of the stack */
    $\mathcal{S} = \mathcal{S} \cup \mathcal{S}_0$;  $\boldsymbol{\Theta} = \boldsymbol{\Theta} \cup \boldsymbol{\theta}_0$;  $r = r + 1$;
    Update $\boldsymbol{\Delta}_{r \times r}$;  /* 10 */
    $f_i = r \quad (i = m + 1, \ldots, m_0)$;  $m = m + m_0$;
  **end if**
**end while**
**return** $\mathcal{C} = \{m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F}\}$;

#### 4.4.2 理論的な分析

［補助定理 3］ AUTOPLAIT の計算量はシーケンスの長さ $n$ に対し線形である．

## 5. 評 価 実 験

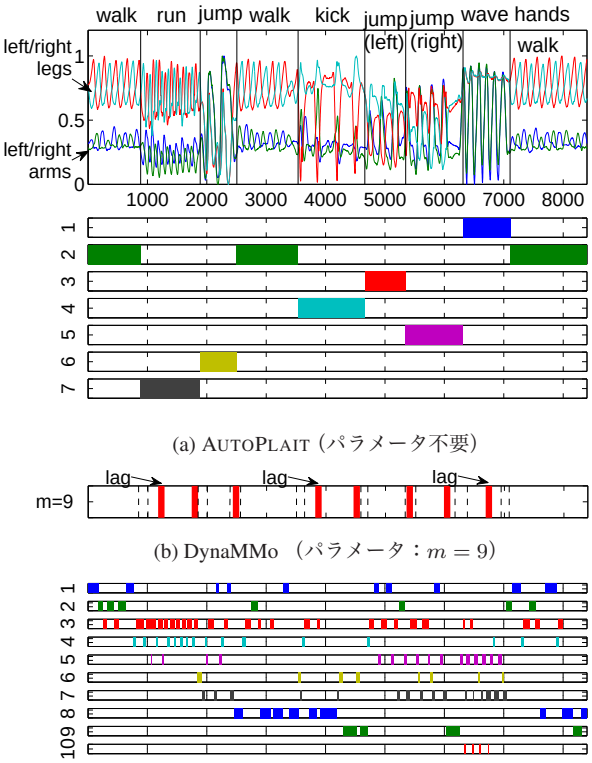
本論文では AUTOPLAIT の有効性を検証するため，実データを用いた実験を行なった．実験は 32GB のメモリ，Intel Core 2 Duo 1.86GHz の CPU を搭載した Linux のマシン上で実施した．実験で用いたデータは以下の通りであり，各々は平均値と分散値で正規化 (z-normalization) して使用した．

- *MoCap*: *MoCap* は，1 秒 120 フレームでヒトの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．(注7) 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使用した．
- *GoogleTrend*(注8): このデータセットは，Google による検索クエリの頻度を週毎に 9 年間に渡り集計したものである．各シーケンスは各クエリの出現頻度を表す．

### 5.1 時系列データからの特徴抽出

提案手法である AUTOPLAIT の情報抽出の効果を検証するため，大規模時系列データの解析のための最新の手法として DynaMMo [3], pHMM [15] と比較した．図 5 は *MoCap* データ

(注7)：http://mocap.cs.cmu.edu/
(注8)：http://www.google.com/insights/search/




(a) AUTOPLAIT（パラメータ不要）

(b) DynaMMo （パラメータ：$m = 9$）

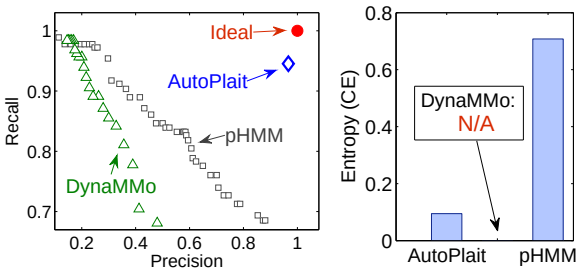
(c) pHMM （パラメータ：$\epsilon_r = 0.1, \epsilon_c = 0.8$）

図 5　*MoCap* における AUTOPLAIT の出力結果のようす．

におけるモーションの特徴抽出の結果を示している．図 1 においても既にモーションの解析結果を示しているが，ここで使用したデータは，異なるユーザによる，より複雑な動きから生成したデータである．このシーケンスは walking, jumping, kicking 等の複数のパターンを含み，walking 以外のパターンは一度しか出現しない．従来のクラスタリング手法にはない AUTOPLAIT の強みの 1 つとして，重複のないトレンドの検出が挙げられる．実際に，図 5 (a) において AUTOPLAIT は running, jumping のような，繰り返し出現しないモーションを抽出し，$m = 9$ 個のセグメントと $r = 7$ 個のレジームの抽出に成功している．

一方，図 5 (b) は DynaMMo によるセグメントの分割結果である．DynaMMo はユーザの指定するパラメータとして，セグメントの個数 $m$ を与える必要があるため，ここでは $m = 9$ を用いて処理を行っている．図 5 (b) では，点線が正解値を示し，色のついた実線が DynaMMo によって得られた値である．DynaMMo は正しいセグメントの個数として $m = 9$ を与えた場合であっても，変化点の位置を正確に特定することができない．さらに重要なこととして，DynaMMo はクラスタリングの能力がないため，レジームを発見することができない．

図 5 (c) は pHMM の出力結果である．pHMM は，モデル学習の際のエラー値に関連する閾値（すなわちパラメータ）として $\epsilon_r$ と $\epsilon_c$ の 2 つを設定しなくてはいけない．文献 [15] を参考にして，数多くの閾値を試し，それらの中から最も適したものを選び，図 5 (c) に示している．図に示すように，pHMM は線形のパターンをモデル化する手法であるため，複雑な時系列パターンや長期的なトレンドを抽出することができない．




(a) 適合率と再現率　　(b) CE スコア

図 6　AUTOPLAIT の精度．

### 5.2 精　　度

続いて，与えられたシーケンスに対する提案手法の変化点抽出とクラスタリングの精度について検証する．

**変化点抽出の精度．** 図 6 (a) は，提案手法と比較手法におけるセグメントの分割点の抽出の精度を適合率と再現率に基づき比較したものである．ここでは *MoCap* データから合計 20 個のシーケンスを選び使用した．各シーケンスにはそれぞれ平均して 10 個程度のセグメントが含まれており，本実験で使用したデータはおよそ $n = 20 \times 10,000$ のモーションフレームを含んでいる．図 6 (a) では，AUTOPLAIT は 1 点のみで表現されている．これは，提案手法がパラメータを持たず，出力結果が 1 つに定まるためである．図のように，提案手法は，95% 以上の正解率を示している．

一方，DynaMMo は，図 5 で示した通り，ユーザの設定するパラメータ $m$ を要する．ここではパラメータを $m = 2, 4, 6, \ldots, 30$ のように変化させてそれぞれ精度を比較している．同様に，pHMM はモデルの学習精度に関連する閾値のパラメータを要する．本実験では $\epsilon_r$ を 0.1 から 10.0 に変化させながら精度を検証した．DynaMMo と pHMM はパラメータによって適合率と再現率が大きく左右されることがわかる．

**クラスタリング精度．** AUTOPLAIT は与えられたシーケンスの中からレジームを発見すると同時に，各クラスタがどのレジームに所属するかの情報（つまり，セグメントメンバーシップ）を抽出することができる．図 6 (b) は提案手法と比較手法におけるクラスタリングの精度を示している．より具体的には，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合行列 (CM: confusion matrix) を作成し，条件付きエントロピー (CE: conditional entropy) [5] を計算した．図 6 (b) は，AUTOPLAIT と pHMM における CE の平均スコアを示している．pHMM と異なり，提案手法はほぼすべてのレジームを正しく抽出している．なお，DynaMMo はクラスタ発見の能力を持たない．

### 5.3 計算コスト

図 7 はシーケンスの長さ $n$ を変化させた際の AUTOPLAIT と比較手法における計算コストを示している．ここでは *MoCap* データを用いた．DynaMMo はパラメータとしてモデルの隠れ状態の個数を必要とするため，本実験では $k = 4$ とした．pHMM については $\epsilon_r = 0.1, \epsilon_c = 0.8$ とした．AUTOPLAIT と DynaMMo はデータの長さに対し，線形 $O(n)$ である（対数スケールにおいて傾きは $slope = 1.0$ である）．一方，pHMM

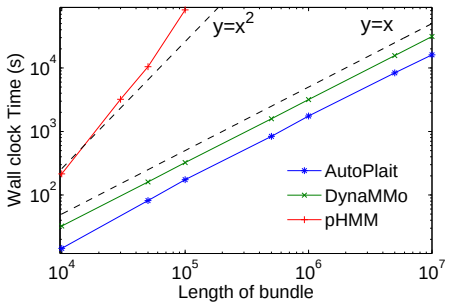



図 7　AutoPlait の計算コスト.

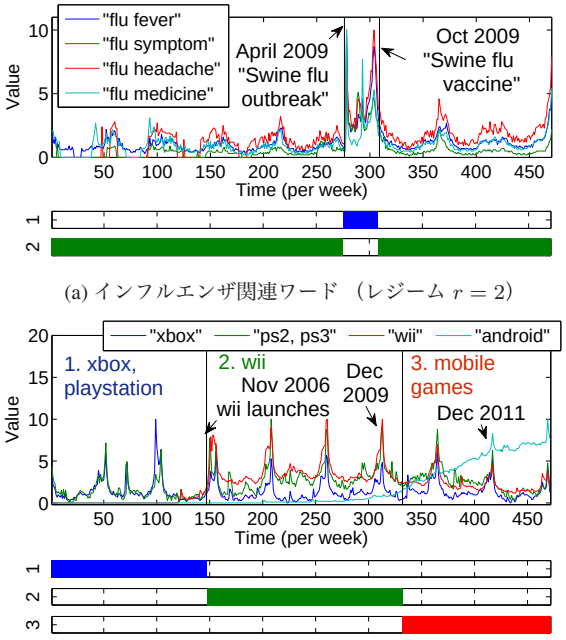
は $O(n^2)$ の計算量を要する（$slope \approx 2.0$）．AutoPlait は pHMM と比較し，$n = 100,000$ において 472 倍の性能向上を達成している．

## 6. アプリケーション

本章では，AutoPlait の実用的なアプリケーションとして，*GoogleTrend* データを使用して，Web 上のユーザ動向の情報を自動検出する例を紹介する．AutoPlait は時系列データの中から，未知のパターンと任意の数のレジームを自動的に発見することができる．

**異常検出.** 図 8 (a) は，インフルエンザに関連するキーワード 4 つ（”flu fever”, ”flu symptom”等）の 9 年間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ち，毎年 10 月から 2 月にかけて検索数が増加し，次第に春，夏にかけて減少していく．この傾向は毎年同様であるが，2009 年は異なる傾向を持つ．これは，豚インフルエンザが世界的に大流行したことが原因である．AutoPlait はこの例外的なパターンをレジーム#1 として検出することに成功している．

**ドレンド発見.** 図 8 (b) はゲーム産業に関連するキーワード (“xbox”, “wii” 等）の時系列データである．毎年 12 月のクリスマスにかけてピークを持つ．近年，ゲーム産業界は様々な企業の参入により，競争的になりつつある．AutoPlait は過去 9 年間のゲーム機戦争における 3 つのレジームを発見している．具体的には，(1) Xbox と Playstation が主流である時代から，(2) Wii が 2006 年に販売開始して以来の大幅なシェアの獲得，そして (3) 高性能携帯端末の普及によるモバイルゲームやソーシャルゲームの出現への流れを捉えている．




(a) インフルエンザ関連ワード （レジーム $r = 2$）




(b) ゲーム関連ワード （レジーム $r = 3$）

図 8　*GoogleTrend* データにおけるトレンドの変化点抽出例.

## 7. むすび

本論文では大規模時系列データのための特徴自動抽出手法として AutoPlait を提案した．AutoPlait は，ユーザによるパラメータ設定や事前知識を要すること無く，与えられた大規模シーケンスに対し，ユーザの直感に合う複雑な時系列パターン（レジーム）とその変化点を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，AutoPlait は最新の時系列解析手法と比べてより高い精度と性能を持つことを示した．

## 文　献

[1] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Sharing features among dynamical systems with beta processes. In *NIPS*, pages 549–557, 2009.

[2] Y. Fujiwara, Y. Sakurai, and M. Yamamuro. Spiral: efficient and exact model identification for hidden markov models. In *KDD*, pages 247–255, 2008.

[3] L. Li, J. McCann, N. Pollard, and C. Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD*, 2009.

[4] L. Li, B. A. Prakash, and C. Faloutsos. Parsimonious linear fingerprinting for time series. *PVLDB*, 3(1):385–396, 2010.

[5] Y. Matsubara, Y. Sakurai, and C. Faloutsos. Autoplait: Automatic mining of co-evolving time sequences. In *SIGMOD*, 2014 (to appear, http://www.cs.kumamoto-u.ac.jp/~yasuko/PUBLICATIONS/sigmod14-autoplait.pdf).

[6] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa. Fast mining and forecasting of complex time-stamped events. In *KDD*, pages 271–279, 2012.

[7] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and fall patterns of information diffusion: model and implications. In *KDD*, pages 6–14, 2012.

[8] A. Mueen and E. J. Keogh. Online discovery and maintenance of time series motifs. In *KDD*, pages 1089–1098, 2010.

[9] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. J. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*, pages 262–270, 2012.

[10] J. Rissanen. A Universal Prior for Integers and Estimation by Minimum Description Length. *Ann. of Statist.*, 11(2):416–431, 1983.

[11] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *ICDE*, pages 1046–1055, 2007.

[12] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *SIGMOD*, pages 599–610, 2005.

[13] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: Fast similarity search under the time warping distance. In *PODS*, pages 326–337, 2005.

[14] M. Toyoda, Y. Sakurai, and Y. Ishikawa. Pattern discovery in data streams under the time warping distance. *VLDB J.*, 22(3):295–318, 2013.

[15] P. Wang, H. Wang, and W. Wang. Finding semantics in time series. In *SIGMOD Conference*, pages 385–396, 2011.