社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－UBI－4 (2)
2004／4／23

センサーデータにおける周期性と異常の検出

倉 光君 郎†

ユビキタスコンピューティングではセンサーデータの処理が重要になる。最も関心が高いのは、異常の検出である。我々は、イベント発生のパターンから周期性と周期に沿わない異常の検出を行うアルゴリズムを開発し、メモリ使用の上限を議論したい。

Discovering Periodic Unusualness in Sensor Data Stream

KIMIO KURAMITSU †

We are given a continuous data stream of ubiquitous sensors, where each sensors consists of time and event. We address the problem of discovering periodicity of event patterns occurs in the sensor stream.

1. はじめに

センサーデータの処理は、ユビキタスコンピューティングの重要な課題である。環境上では、様々なセンサーがアクティブであり、読み取られるストリームデータは膨大な量になる。アプリケーション開発者は、常に監視し続けるのも、全てを記録するのも好まない。データストリームから有益な情報を取り出して提供する必要がある。

我々は、センサーデータの周期性/periodicity に注目する。世の中のイベントの多くは何らかの周期性を持って発生し、その周期から外れた現象は関心度の高い非日常/unusual である。ひとつの例は、朝コーヒーである。毎朝、コーヒーメーカーの温度計はコーヒーを入れたことを教えてくれる。もし温度センサーが働かなければ、何か起きたかも知れない。少なくともその可能性に注目すべきである。

簡単な解決策は、人手でルールを与えることである。しかし、センサーの数が増えると、ルールを設定するのも問題である。ルールがアプリケーションに依存するのもスケールと複雑さの問題を備える。

我々は、もう少しスマートなセンサーを考える。自動的にイベントの周期性を学習し、発生頻度の低いイベントに対しては自律的に注意を喚起してくれるセンサーである。イベントの種類を問わず、このような一般的なアルゴリズムを考えるのは非常に利用範囲が大きい。本研究では、そのようなセンサーを作成するためのアルゴリズムを報告する。

センサーデバイスの処理能力[9] は限られている。そこで発生する特別な技術課題もある。周期パターンの表現は十分にシンプルでなければならない。非日常の判定も簡単でなければならない。もちろん利用するメモリの上限もある。本論文の貢献は以下のとおりである。

- 周期パターンの簡単な表現と非日常性の検出を可能にした。
- データマイニングのアイディアの解析法を取り入れて、複数の周期性を分離することを可能にした。
- データストリームに対して、限られたメモリ上で実行を行うアルゴリズムを開発した。

本論文の構成は次の通り構成される。第 2 節では、問題の形式化を行う。第 3 節では、ストリーム上のアルゴリズムを述べる。第 4 節では、関連研究をまとめる。第 5 節では、本論文をまとめる。

2. 問題の定式化

2.1 準　　備

センサーは、$E = \{e_1, e_2, .., e_i, ..\}$ で識別できるイベントを検出するデバイスである。あるイベント e の発生は、発生時刻 t とあわせて、$\langle e, t\rangle$ で記述できる。これを出現イベントと呼ぶ。我々は、出現イベントの量に関しては考慮にいれない。センサーが読み取った出現イベントは、次のようなデータストリームとなる。

$$\langle e_1, t_1\rangle, \langle e_3, t_2\rangle, \langle e_1, t_3\rangle, \langle e_2, t_4\rangle, \langle e_2, t_5\rangle, ..$$

これをイベント列/event sequence と呼ぶ。

† 工学院大学 CPD センター
CPDC, Kogakuin University

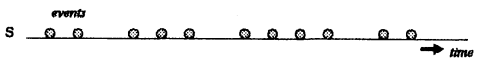


図 1 A Pictorical Notation of A Event Sequence S

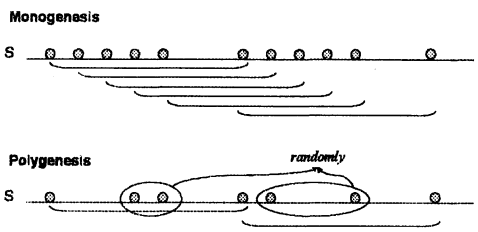


図 2 Monogenesis vs. Polygenesis

定義 2.1 イベント列 S は、出現イベント $\langle e,t\rangle$ の集合で、時刻 t によってソートされた全順序集合である。便宜上、我々は S を有限集合とみなし、S 上の最終時刻を $t_{|S|}$ と書く。

今、非日常発見問題/unusual discovering problem は次のとおりである。

定義 2.2 (非日常発見問題) 今、あるイベント列 S が与えられたとき、時刻 $t(>t_{|S|})$ にイベントが発生する（発生しない）のは、日常的かどうか？(注目すべき発生頻度の低い現象か？)

2.2 Assumptions

我々は、いくつかの仮定を導入して問題に取り組む。イベントは通常相関的に発生すると考えられる。つまりあるイベントが他のイベントを引き起こす。ただし、S 上の出現イベントがお互いに独立であると仮定しても一般性を失わない。我々はまず、S のイベントは、外部の何らかの周期的な変化によってのみ引き起こされると仮定する。

この仮定は、S 上の特定の種類のイベントに議論を集中できるようにする。我々は、ここからあるイベント e に関するサブセットのことを S と書く。図 1は、S の図形表記である。

周期の期限に関する仮説 – S 上の出現イベントを引き起こす変化はいくつあるか – は、我々のアルゴリズムの中核である。次の 2 つの仮説 (図 2) を考えられる。

- 単一起源説/monogenesis. S 上の出現イベントは、全てある単一な周期的変化に依存している。例えば、月曜日から金曜日までは毎朝コーヒーを飲むが、土日は飲まない場合が考えられる。このとき、コーヒーの出現は 1 日ではなく、7 日周期になる。
- 多元起源説/polygenesis. S 上の出現イベントは、複数の外部の周期的変化による、もしくは時には乱数的でもある。例えば、毎朝コーヒーを定期的に飲むが、午後からは来客に合わせて飲む場合が考えられる。このとき、来客の周期性があるか、午後コーヒーの出現はランダムである。

2 つの事実に注意したい。ひとつは、単一起源なイベント列の方が扱いやすい点である。もうひとつ、実環境ではたぶん多くの場合、複数起源的なイベント列を扱う必要がある点である。ただし、イベント列がどちらの起源なのかあらかじめ予測するのは難しい。一見ランダムに見える午後の来客もはやり何らかの外部の周期的な変化があるかも知れない。我々のアルゴリズムの戦略は、単一起源を扱うアルゴリズムを組み立てたのち、任意のイベント列を単一起源なシーケンスに変換することである。

3. A Data-Mining Approach

我々は、まず単一起源の立場をとる。単一起源なシーケンス S は、周期/*period* を持つ。我々は、これを S の単一起源な周期 T と呼ぶ。出現イベントの立場からみれば、周期は次のとおり定義できる。

定義 3.1 (period) S 上において、ある出現イベント $\langle e,t\rangle$ が周期 T を満たすとは、出現イベント $\langle e,t-\mathrm{T}\rangle$ が存在することである。

3.1 サポート率

我々は、S の T を決定する方法から述べる。読者は、S を単一起源であると想定してもかまわないが、本手法は複数起源のときも同様に利用できる。

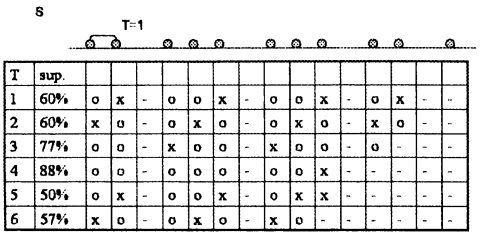
まず、候補セット $\mathbf{T}=\{T_1,T_2,T_3,..\}$ を考える。(候補セットを求める具体的な手法はアルゴリズム節で述べる。)S の周期 T は、このセットの中から選ばれる。$|S|$ を S 上の出現イベントの総数、$T(S)$ を周期 T を満たす出現イベントによる S の部分集合とする。データマイニング用語を用いれば、

"S は T を $\frac{|T(S)|}{|S|}$ でサポートする"

といえる。S の周期 T は、候補セットにおいて、最大のサポートを提供する候補周期である。最大サポート率を α% と書くすると、S は周期 T を α% サポートするといえる。

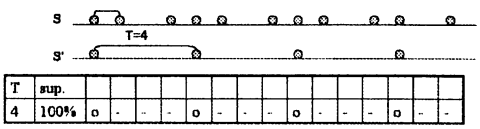
図 3 は、周期 T を決定する例である。簡単さのため、候補キーは全て $T=1$ の倍数とする。最大サポート率は、数え上げにより、$T=3$ のときになる。つまり、S の周期は $\mathrm{T}=3$ である。注意. 我々は $|S|$ の代わりに有効な出現イベントのカウントを用いているが、$|S|$ が大きくなればこれは無視できる。

我々は、α が極めて高く 100% にほぼ近いとき、S は単一起源であると呼べる。もちろん α が高いことが望ましいが、低い場合は複数起源であると考えればよ



T	sup.														
1	60%	o	x	-	o	o	x	-	o	o	x	-	o	x	-	-
2	60%	x	o	-	o	x	o	-	o	x	o	-	x	o	-	-
3	77%	o	o	-	x	o	o	-	x	o	o	-	o	-	-	-
4	88%	o	o	-	o	o	o	-	o	o	x	-	-	-	-	-
5	50%	o	x	-	o	o	x	-	o	x	x	-	-	-	-	-
6	57%	x	o	-	o	x	o	-	x	o	-	-	-	-	-	-

図 3 Decision of Maximum Support



T	sup.														
4	100%	o	-	-	-	o	-	-	-	o	-	-	-	o	-	-

図 4 A subset of S with 36% con. dence

い。(その閾値はアプリケーションに依存する。)

3.2 確信度/Con. dence

実際のイベント列は、必ずしも高サポート率が保証されるとは限らない。我々は、サポート率の低いイベント列からサポート率の高いイベント列を取り出す手法に注目する。そのための準備として、まず確信度を導入する。

定義 3.2 (con. dence) S の部分集合 S' を考える。S' が周期 T を α'% サポートするとき、確信度は、$\frac{|S'|}{|S|}$ (%) である。

図 4は、S の部分イベント列 S' の例を示している。S' の確信度は 36% であるが、$T = 4$ を 100% サポートする。

確信度は、単一起源説と複数起源説を区別するのに便利である。単一起源なイベント列は、確信度を下げてもサポート率はほとんど向上しない。これに対し、複数起源なイベント列は確信度を下げればサポート率が大きく向上する。後者の場合、サポート率が大きくなる部分イベントを探すことは、単一起源なイベント列の抽出になる。

3.3 非日常の発見

最後に、イベント列 S 上で発見された T と α% から非日常性を検出する解法を考える。大きな戦略として、2 種類の解が考えられる。

- **確実な解.** 新しく読み取られたイベントを S に加える。もし S のサポート率が事前にセットされた値を下回ったら、非日常的である。
- **不確実な解.**
  - **(起こるはずのないイベントが発生する場合)** $\langle e, t\rangle$ が出現し、かつ $\langle e, t - \mathrm{T}\rangle$ が存在しないとき、α% 非日常的である。

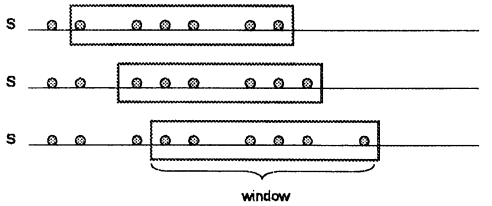


図 5 A Sliding Window for Incoming Events

  - **(起こるはずのイベントが発生しない場合)** 時刻 t にイベント e が出現せず、かつ $\langle e, t - \mathrm{T}\rangle$ が存在するとき、α% 非日常的である。

どちらの解も有意義な答えを出してくれる。ただし、スマートセンサーでの実装を考えた場合、後者の方がより判定が簡単である。結果は、統計的な不確実さが含まれるが、この複雑さは単体のセンサーが判断する内容とはいえない。より上位のミドルウェアやアプリケーションレベルにおいて、複数のセンサーや知識ベースと統合して判断すべきである。

4. ストリーム・アルゴリズム

S を有界なデータベース D と見なせば、周期 T やサポート率を最大化する部分イベント列 S を探すのは難しくない。実際は、S はサイズの上限がないデータストリームであり、それを全て蓄積することはできない。

4.1 スライディングウィンドウ

我々は、ストリームデータ処理の基本テクニックであるスライディングウィンドウ/sliding window[3),5)] 用いる。基本的なアイディアは、図 5に示すとおりである。出現イベントにあわせ、ストリームデータを有界な領域 (window) に分割する。

図 6は、ウィンドウのデータ構造である。ウィンドウサイズは、(理想的には、時間で切るのが望ましいが、) メモリ使用量の上限を見積もるため、固定数 N とする。我々は実世界の時刻を扱うため、許容時間差 Δ を導入する。つまり、$|t_{i+1} - t_i| < \Delta$ なら、$t_i \equiv t_{i+1}$ である。

4.2 候補セットとサポート率

今、新たにイベントが検出されたとき、出現時刻はイベントイベントウィンドウの最終列に追加される。つまり、出現時刻は t_N である。

```
for(i = 1; i < N − 1; i + +){
    T = t_N − t_i;
    count[T] + +; // count
}
```

	t	T=20	T=30	T=40
1	1400	**33**	*25*	*18*
2	1440	**34**	*22*	**19**
3	1450	*11*	**18**	*16*
4	1480	**35**	**19**	**20**
5	1500	**36**	*23*	*14*
6	1530	*12*	*25*	*17*
7	1560	**37**	*26*	**21**
8	1580	**38**	*26*	*15*
9	1600	**39**	**20**	**22**

N=9, S, sub(1600,20), sub(1600,30) sub(1600,40)

図 6 Data Structure of Window

$count[T]$ は候補周期 T の出現数を数えるカウンタである。このカウンタを用いると、サポート率の計算ができる。センサー起動から現在までに、合計 n 回イベントが出現したとする。このとき、候補 T のサポート率が 100% なら、$count[T] = n - 1$ のはずである。したがって、

S は、T を $\frac{count[T]}{n-1}(\%)$ でサポートする。

4.3 部分イベント列

$sub[t_N, T]$ は、時刻 t_N から周期 T ごとに発生する部分イベント列のカウンタである。候補セットに対して、次のように数える。

```
foreach T in T{
  if (sub[t_N − T, T] exists) them
    sub[t_N, T] = sub[t_N − T, T] + 1;
  else
    sub[t_N, T] = 0: sub[t_N − T ∗ n, T];
}
```

ここで、センサーの起動時間を t_0、現在の時刻を t_N とする。$sub[t_N, T]$ のイベント列は、最大 $(t_N - t_0)/T$ 回発生する。したがって、$sub[t_N, T]$ の周期 T のサポート率は、$\frac{sub[t_N,T]}{(t_N-t_0)/T}(\%)$ である。ちなみに、確信度は $(sub[t_N, T])/n$ である。

5. 関連研究

シーケンシャルなデータ列からパターンを抽出する関連研究に関してまとめる。

FFT や DWT は、パターンを抽出するアルゴリズムであるが、パターンから予測したり異常を発見することに利用できない。時系列解析は、統計学の分野で大きな仕事が行われてきたが、同様に異常の検出には適さない。[6]

我々は、データマイニングの概念と手法[1] をイベント列の解析に適用した。従来の研究分野では、タプル間のマイニングを行うという点で、mining sequencial patterns[2] の研究が盛んに行われてきた。これは、"Star Wars", "Empire Strikes Back", を見たら、次は "Return of Jedi"を観るというようなイベント間の相関ルールを探すアルゴリズムである。周期性に着目したものではない。

近年では、ストリームデータ処理[3),5)] は盛んである。クエリー言語 (SQL の変形) による簡単な統計処理が可能であるが、本研究のような処理を直接記述することはできない。

6. ま と め

センサーデータの処理は、重要な課題である。我々は、イベント発生の周期性と非日常発見問題に取り組み、メモリの上限のある中で解くアルゴリズムを開発した。これにより、簡単なプロセッサを搭載したセンサーに載せることも可能になった。

今後は、実データを用いて実験を行う予定である。さらに将来の課題は、複数のセンサーや他のアルゴリズム（イベント間相関など）で検出した異常を統合する確率的なインターフェースの開発を行う予定である。

参 考 文 献

1) R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. of ACM SIGMOD93*, pages 207–216, 26–28 1993.
2) R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of ICDE'95*, pages 3–14, 1995.
3) B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of PODS2002*, 2002.
4) J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. of ASPLOS-IX*, 2000.
5) R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, and J. R. andR. Varma. Query processing, resource management, and approximation in a data stream management system. In *Proc. of CIDR2003*, January 2003.
6) S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, hands-off stream mining. In *Proc. of VLDB2004*, pages 560–571, 2004.