Fast and Exact Monitoring of Co-evolving Data Streams

Speaker: Yasushi Sakurai

Yasuko Matsubara, Yasushi Sakurai (Kumamoto University)
Naonori Ueda (NTT)
Masatoshi Yoshikawa (Kyoto University)
Motivation

**Given:** co-evolving data streams – e.g., MoCap (leg/arm sensors)
Motivation

Given: co-evolving data streams – e.g., MoCap (leg/arm sensors)

Multiple distinct patterns

e.g., Walking, punching
different durations
Given: co-evolving data streams
– e.g., MoCap (leg/arm sensors)

Queries:
Q1: Walking
Q2: Running
Q3: Punching
Motivation

Q. Can we find subsequences that have the characteristics of query $\Theta$?

Queries:

Q1: Walking
Q2: Running
Q3: Punching

(b) Original MoCap stream
Outline

- Motivation
- Problem formulation
- Main ideas
- Experiments
- StreamScan at work
- Conclusions
Goal: statistical monitoring of time-varying data streams
**Background**

**Goal:** statistical monitoring of time-varying data streams

**Query (punch)**

**Data stream**
Background

Goal: statistical monitoring of time-varying data streams

Query (punch)
Goal: statistical monitoring of time-varying data streams

Query (punch)

Hidden Markov models (HMMs)

Data stream
Hidden Markov models (HMMs)

\[ \Theta = \{ \pi, A, B \} \]

- **Initial state probabilities**
  \[ \pi = \{ \pi_i \}_{i=1}^k, \]

- **State transition probabilities**
  \[ A = \{ a_{ij} \}_{i,j=1}^k, \]

- **Output probabilities**
  \[ B = \{ b_i(x) \}_{i=1}^k \]
Model: \( \Theta = \{ \pi, A, B \} \)

Sequence: \( X = (x_1, x_2, \ldots, x_n) \)

Likelihood: \( P(X, \Theta) \)
- **Model:** \( \Theta = \{\pi, A, B\} \)

- **Sequence:** \( X = (x_1, x_2, \ldots, x_n) \)

**Likelihood:** \( P(X, \Theta) \)

\[
P(X, \Theta) = \max_{1 \leq i \leq k} \{p_i(n)\}
\]

\[
p_i(t) = \begin{cases} 
\pi_i b_i(x_1) & (t = 1) \\
\max_{1 \leq j \leq k} \{p_j(t - 1)a_{ji}\} b_i(x_t) & (2 \leq t \leq n)
\end{cases}
\]
Given: a data stream $X$ and model $\Theta$

$$X = \{x_1, \ldots, x_n\}$$
**Given**: a data stream $X$ and model $\Theta$

$$X = \{x_1, \ldots, x_n\}$$

**Find**: subsequences that has a high likelihood value

$$X[t_s : t_e]$$

$$P(X[t_s : t_e], \Theta)$$
Requirements

1. **Exponential** threshold function
2. **Minimum length** of subsequences
3. **Non-overlapping** matches
Requirements

1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches
Requirements

1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches
1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches
Requirements

1. **Exponential threshold function**
2. **Minimum length of subsequences**
3. **Non-overlapping matches**

\[ P(X[t_s : t_e], \Theta) \geq \varepsilon^m \]

\[ m = t_e - t_s + 1 \]
Requirements

1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches

\[ P(X[t_s : t_e], \Theta) \geq \varepsilon^{m-\delta} \quad m \geq \delta \]
Requirements

1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches

\[ P(X[t_s : t_e], \Theta) \geq \varepsilon^{m-\delta} \]
Problem definition

Given:
- data stream $X$
- Model $\Theta$
- thresholds $\epsilon, \delta$
Problem definition

**Given:**
- data stream
- Model Θ
- thresholds ε, δ

**Report:** all subsequences $X[t_s : t_e]$ that satisfy:
1. $P(X[t_s : t_e], Θ) \geq ε^{m-δ}$
2. only the local maximum, among several overlapping matches
Outline

- Motivation
- Problem formulation
- Main ideas
- Experiments
- StreamScan at work
- Conclusions
Previous solution

Sliding model method (SMM), by Wilpon et al.
Compute likelihood starting from every time-tick
Previous solution

Sliding model method (SMM), by Wilpon et al.
Compute likelihood starting from every time-tick

\[
\Theta
\]

\[
X
\]

\[
x_t
\]

\[
x_1
\]

\[
\cdots
\]

\[
a_{ii}
\]

\[
a_{ij}
\]

\[
\cdots
\]

\[
O(n) \text{ trellis structures}
\]

Update \(O(nk^2)\) for each time-tick
Main idea: StreamScan

(a) Cumulative likelihood function

$$V(X[t_s : t_e], \Theta)$$

(b) Subsequence trellis structure (STS)
Main idea (a)

(a) Cumulative likelihood function

\[ V(X[t_s : t_e], \Theta) = v_{best}(t_e) = \max_{1 \leq i \leq k} \{ v_i(t_e) \} \]

\[ v_i(t) = \max \left\{ \frac{\pi_i b_i(x_t) \cdot \epsilon^{-1}}{\max_{1 \leq j \leq k} \{ v_j(t - 1) a_{ji} \}} b_i(x_t) \cdot \epsilon^{-1} \right\} \]

\[ (t = 1, \ldots, n; i = 1, \ldots, k). \]

\[ P(X[t_s : t_e], \Theta) = V(X[t_s : t_e], \Theta) \cdot \epsilon^m, \]
Main idea (a)

(a) Cumulative likelihood function

\[ V(\mathbf{X}[t_0 : t_N], \Theta) = \max \{ \nu_i(t_s) \} \]

\[ \nu_i(t) = \frac{x_t}{\epsilon} \]

1, \ldots, n; i = 1, \ldots, k).

\[ P(\mathbf{X}[t_s : t_e], \Theta) = V(\mathbf{X}[t_s : t_e], \Theta) \cdot \epsilon^m, \]

It requires a **single** structure

It **guarantees** the **best** matches
Main idea (b)

(b) Subsequence trellis structure (STS)

Details
Main idea (b)

(b) Subsequence trellis structure (STS)

For each cell: $s_i(t)$ : Starting position
$v_i(t)$ : Cum. likelihood

\[ \Theta \]

Where $i = 1, 2, 3, \ldots$

$X_{i \colon s_e}$
Algorithm

Example,

\[ X = (3, 1, 1, 2, 3, 3, 3, 1). \]

\[ \varepsilon = 0.1, \delta = 3 \]

\[ \Theta = \{ \pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, A = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.25 & 0.25 \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 \\ 0.75 & 0.25 & 0 \\ 0 & 0 & 1 \end{bmatrix} \} \]
Algorithm

Example, (STS, k=3)

\[ X = (3, 1, 1, 2, 3, 3, 3, 1). \]

\[ \varepsilon = 0.1, \delta = 3 \]

<table>
<thead>
<tr>
<th>State 1</th>
<th>0 (1)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>State 2</td>
<td>0 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State 3</td>
<td>0 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_1=3</td>
<td>x_2=1</td>
<td>x_3=1</td>
<td>x_4=2</td>
<td>x_5=3</td>
<td>x_6=3</td>
<td>x_7=3</td>
<td>x_8=1</td>
<td></td>
</tr>
</tbody>
</table>

\( X \)

\( t = 1 \)
Algorithm

Example, (STS, k=3)

State 1 | 0 (1)
State 2 | 0 (1)
State 3 | 0 (1)

\[ x_1 = 3 \quad x_2 = 1 \quad x_3 = 1 \quad x_4 = 2 \quad x_5 = 3 \quad x_6 = 3 \quad x_7 = 3 \quad x_8 = 1 \]

\[ X = (3, 1, 1, 2, 3, 3, 3, 1). \]
\[ \varepsilon = 0.1, \delta = 3 \]

\[ v_i(t) \text{ : Cum. Likelihood} \]
\[ s_i(t) \text{ : Starting position} \]
**Algorithm**

**Example,**

(STS, k=3)

$$X = (3, 1, 1, 2, 3, 3, 3, 1).$$

$$\epsilon = 0.1, \delta = 3$$

<table>
<thead>
<tr>
<th>State 1</th>
<th>0 (1)</th>
<th>10 (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 2</td>
<td>0 (1)</td>
<td>0 (2)</td>
</tr>
<tr>
<td>State 3</td>
<td>0 (1)</td>
<td>0 (2)</td>
</tr>
</tbody>
</table>

$$x_1 = 3, x_2 = 1, x_3 = 1, x_4 = 2, x_5 = 3, x_6 = 3, x_7 = 3, x_8 = 1$$

**X**

$$t = 1 \quad t = 2$$
**Algorithm**

**Example,**

\[ X = (3, 1, 1, 2, 3, 3, 3, 1) \]

\[ \varepsilon = 0.1, \delta = 3 \]

**(STS, k=3)**

<table>
<thead>
<tr>
<th>State 1</th>
<th>0</th>
<th>10</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(2)</td>
<td></td>
</tr>
<tr>
<td>State 2</td>
<td>0</td>
<td>0</td>
<td>37.5</td>
</tr>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(2)</td>
<td></td>
</tr>
<tr>
<td>State 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td></td>
</tr>
</tbody>
</table>

\[ x_1 = 3 \quad x_2 = 1 \quad x_3 = 1 \quad x_4 = 2 \quad x_5 = 3 \quad x_6 = 3 \quad x_7 = 3 \quad x_8 = 1 \]

\[ t = 1 \quad t = 2 \quad t = 3 \]
Example, (STS, $k=3$)

<table>
<thead>
<tr>
<th>State 1</th>
<th>0</th>
<th>10</th>
<th>50</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State 2</th>
<th>0</th>
<th>0</th>
<th>37.5</th>
<th>62.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State 3</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$x_1 = 3$, $x_2 = 1$, $x_3 = 1$, $x_4 = 2$, $x_5 = 3$, $x_6 = 3$, $x_7 = 3$, $x_8 = 1$

$X = (3, 1, 1, 2, 3, 3, 3, 1)$.

$\varepsilon = 0.1, \delta = 3$

$X_t = 2, t = 3 \times 0.5 \times 0.25 \times 10$

$X_t = 4$
Algorithm

Example, \( X = (3, 1, 1, 2, 3, 3, 3, 1) \).
\( \epsilon = 0.1, \delta = 3 \)

(STS, \( k=3 \))

<table>
<thead>
<tr>
<th>State 1</th>
<th>0</th>
<th>10</th>
<th>50</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(2)</td>
<td>(4)</td>
<td>(5)</td>
<td>(6)</td>
<td>(7)</td>
<td></td>
</tr>
<tr>
<td>State 2</td>
<td>0</td>
<td>0</td>
<td>37.5</td>
<td>62.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(5)</td>
<td>(6)</td>
<td>(7)</td>
</tr>
<tr>
<td>State 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>156.25</td>
<td>1562.5</td>
<td>15625</td>
</tr>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td></td>
</tr>
</tbody>
</table>

\[
X = \begin{pmatrix}
    x_1 = 3 \\
    x_2 = 1 \\
    x_3 = 1 \\
    x_4 = 2 \\
    x_5 = 3 \\
    x_6 = 3 \\
    x_7 = 3 \\
    x_8 = 1
\end{pmatrix}
\]

\[
x_1(t) = 3, x_2(t) = 1, x_3(t) = 1, x_4(t) = 2, x_5(t) = 3, x_6(t) = 3, x_7(t) = 3, x_8(t) = 1
\]

\[
X = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)
\]

\[
x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)
\]

\[
t = 1, 2, 3, 4, 5, 6, 7
\]
Algorithm

Example, (STS, k=3)

\[ X = (3, 1, 1, 2, 3, 3, 3, 1). \]
\[ \epsilon = 0.1, \delta = 3 \]

<table>
<thead>
<tr>
<th>State 1</th>
<th>0 (1)</th>
<th>10 (2)</th>
<th>50 (2)</th>
<th>0 (4)</th>
<th>0 (5)</th>
<th>0 (6)</th>
<th>0 (7)</th>
<th>10 (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 2</td>
<td>0 (1)</td>
<td>0 (2)</td>
<td>37.5 (2)</td>
<td>62.5 (2)</td>
<td>0 (5)</td>
<td>0 (6)</td>
<td>0 (7)</td>
<td>0 (8)</td>
</tr>
<tr>
<td>State 3</td>
<td>0 (1)</td>
<td>0 (2)</td>
<td>0 (3)</td>
<td>0 (4)</td>
<td>156.25 (2)</td>
<td>1562.5 (2)</td>
<td>15625 (2)</td>
<td>0 (8)</td>
</tr>
</tbody>
</table>

\[ x_1 = 3 \]
\[ x_2 = 1 \]
\[ x_3 = 1 \]
\[ x_4 = 2 \]
\[ x_5 = 3 \]
\[ x_6 = 3 \]
\[ x_7 = 3 \]
\[ x_8 = 1 \]

\[ X \]

\[ t = 1 \]
\[ t = 2 \]
\[ t = 3 \]
\[ t = 4 \]
\[ t = 5 \]
\[ t = 6 \]
\[ t = 7 \]
\[ t = 8 \]
StreamScan guarantees
- no false dismissals
- $O(1)$ space and time per time-tick

(Details in paper!)
Outline

- Motivation
- Problem formulation
- Main ideas
- Experiments
- StreamScan at work
- Conclusions
Experiments

We answer the following questions...

**Q1. Effectiveness**
How successful is StreamScan in capturing sequence patterns?

**Q2. Scalability**
How does it scale with the sequence lengths $n$ in terms of time and space?
Q1. Effectiveness (MoCap)

(Query)

(a-1) Query #1: walking
(a-2) Query #2: running
(a-3) Query #3: punching

(Output)

Diagram showing the results of the queries with timelines and graphs representing different movements. The diagram includes timelines for walking, running, and punching, with corresponding graphs showing data for left/right legs and left/right arms.
Q1. Effectiveness

**MoCap**

Q1. Walk
Q2. Jump

Q1. Walk
Q2. Jump
Q3. Moving arms

(a) MoCap #2 (Query #1: walking, Query #2: jumping)

(b) MoCap #3 (Query #1: walking, Query #2: twisting, Query #3: moving arms)
Q2. Scalability

Time/space vs. data size (length) : n

StreamScan requires constant time/space, i.e., $O(1)$
Q2. Scalability

Time/space vs. data size (length) : n

StreamScan requires constant time/space, i.e., $O(1)$
Outline

- Motivation
- Problem formulation
- Main ideas
- Experiments
- StreamScan at work
- Conclusions
StreamScan at work

StreamScan is capable of various applications, e.g.,

**App1. Social activity monitoring**
- *Web-click* streams

**App2. Extreme detection in keyword stream**
- *GoogleTrend* streams
StreamScan at work

StreamScan is capable of various applications, e.g.,

**App1. Social activity monitoring**
- **Web-click streams**

**App2. Extreme detection in keyword stream**
- **GoogleTrend streams**
Application (1)

Social activity monitoring

Given: Web-click sequences (1 month, 5 urls)

Query: first Sunday

(5 urls: blog, news, dictionary, Q&A, mail)
Application (1)

Social activity monitoring

Given: Web-click sequences (1 month, 5 urls)

Query: first Sunday

(5 urls: blog, news, dictionary, Q&A, mail)

StreamScan identifies all weekends + holiday

Detected!
StreamScan at work

StreamScan is capable of various applications, e.g.,

**App1. Social activity monitoring**
- Web-click streams

**App2. Extreme detection in keyword stream**
- GoogleTrend streams

\[ \Theta \]

\[ X \]
“Extreme monitoring” in keyword stream

Given: GoogleTrend streams (10 years, weekly)
"Extreme monitoring" in keyword stream

**Given:** GoogleTrend streams (10 years, weekly)

e.g., Finance-related keywords
Application (2)

“Extreme monitoring” in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Finance-related keywords

Any extreme behavior?

Time (weekly, 10 years)
Application (2)

“Extreme monitoring” in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Finance-related keywords

Any extreme behavior?

Query

Time (weekly, 10 years)
“Extreme monitoring” in keyword stream

Given: Google Trend streams (10 years, weekly)
e.g., Finance-related

Detected!

Global financial crisis in 2008!

(first 3 years)
"Extreme monitoring" in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Flu-related keywords

Query (first 3 years)
Application (2)

“Extreme monitoring” in keyword stream

Given: Google Trend streams (10 years, weekly)

e.g., Flu-related keywords

Detected!

Swine flu pandemic in 2009
“Extreme monitoring” in keyword stream

**Given:** GoogleTrend streams (10 years, weekly)

e.g., Seasonal sweets-related keywords

Query (first 3 years)
Application (2)

“Extreme monitoring” in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Seasonal sweets-related keywords

Detected!

Release of Android OS, “Gingerbread”
“Ice Cream Sandwich”
Outline

- Motivation
- Problem formulation
- Main ideas
- Experiments
- StreamScan at work
- Conclusions
Conclusions

StreamScan has the following properties

- **Effective** ✔
  Find fruitful patterns from diverse data streams

- **Exact** ✔
  It guarantees exactness

- **Fast and nimble** ✔
  It requires single scan, O(1) space and time
Thank you!

Q1. walk
Q2. run
Q3. punch

http://www.cs.kumamoto-u.ac.jp/~yasuko/yasuko@cs.kumamoto-u.ac.jp