Automatic Sequential Pattern Mining in Data Streams

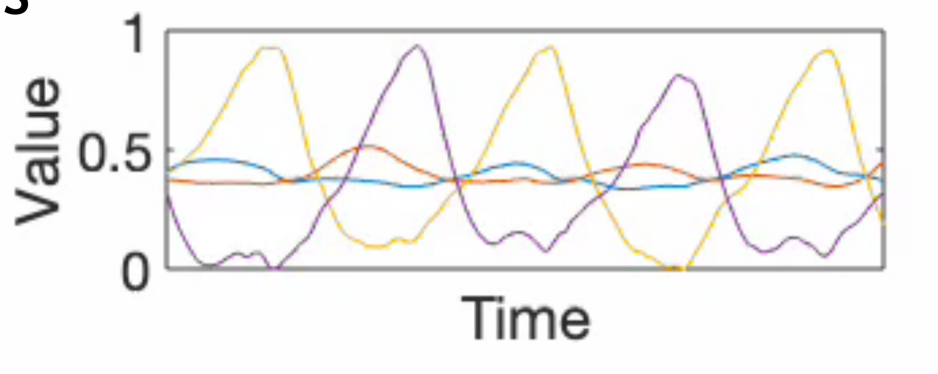
Koki Kawabata*, Yasuko Matsubara & Yasushi Sakurai
ISIR-AIRC, Osaka University

*Supported by SIGIR Student Travel Grants

Motivation

Given: time-evolving data streams

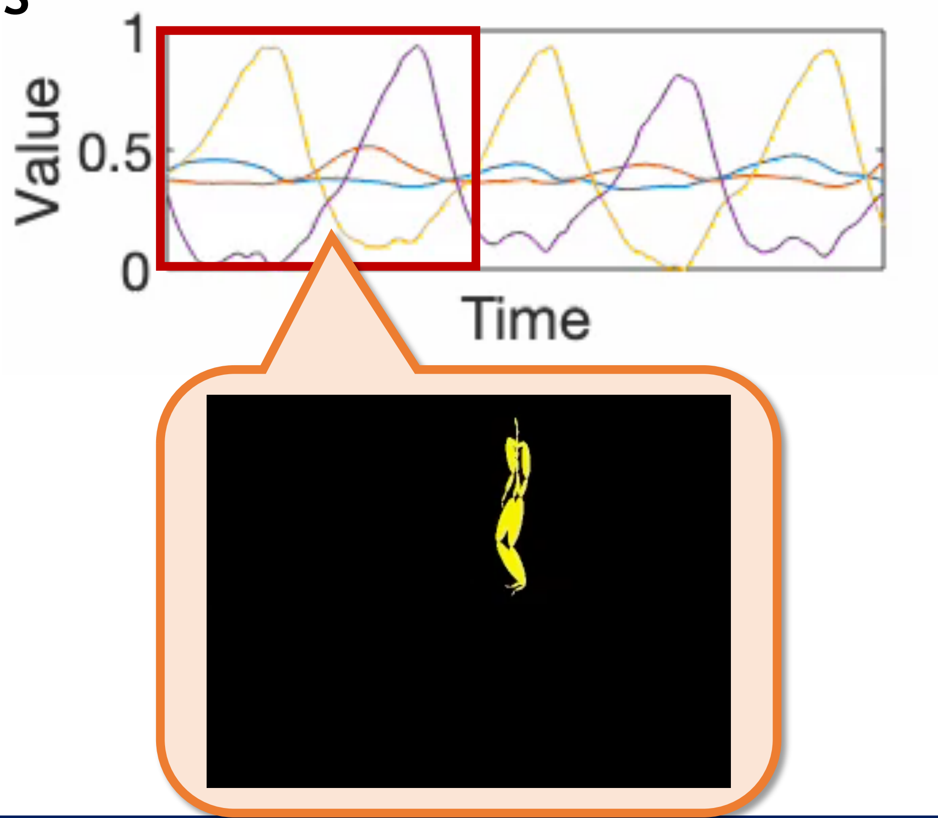
- e.g., IoT sensors/Web click logs
- contain multiple patterns



Motivation

Given: time-evolving data streams

- e.g., IoT sensors/Web click logs
- contain multiple patterns

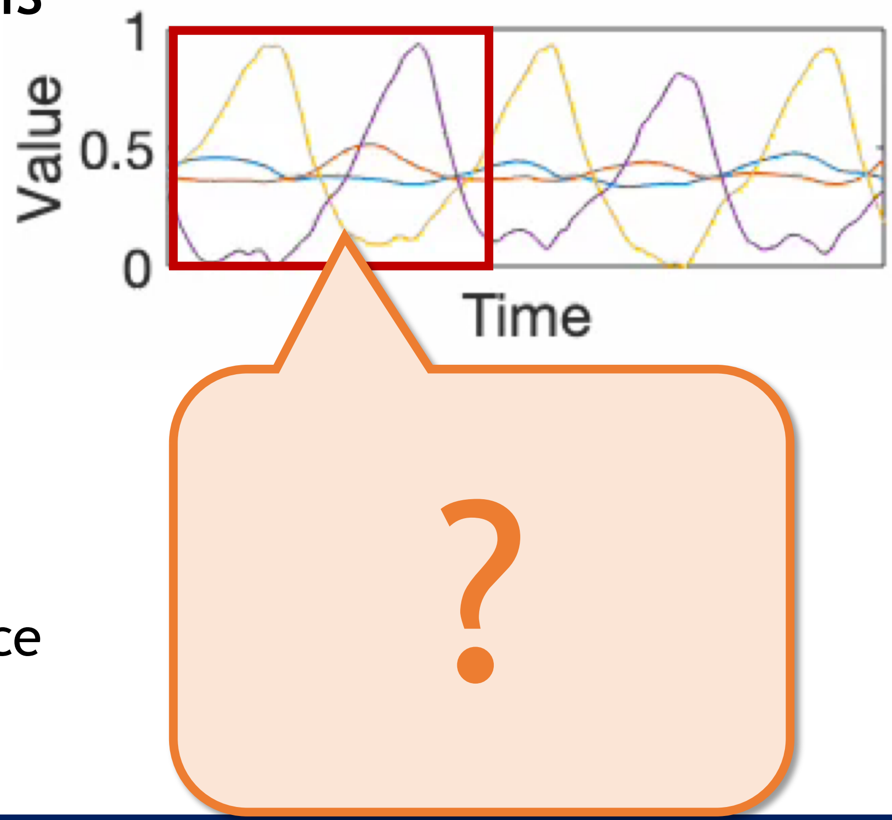


Answer: the following questions:

1. What kind of patterns?
2. How many patterns?
3. When do patterns change?

Motivation

Given: time-evolving data streams
- e.g., IoT sensors/Web click logs
- contain multiple patterns

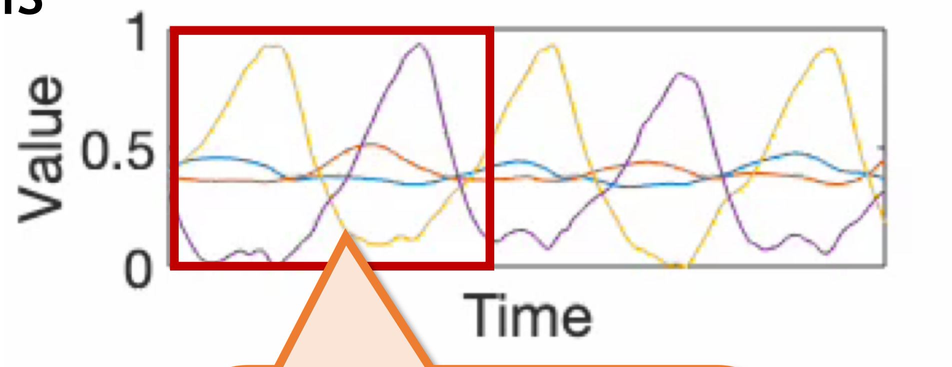


Requirements:
- **Incremental**
  - We cannot access all historical data
- **Automatic**
  - # of patterns are unknown in advance
  - without any parameter tunings

Motivation

Given: time-evolving data streams

- e.g., IoT sensors/Web click logs
- contain multiple patterns

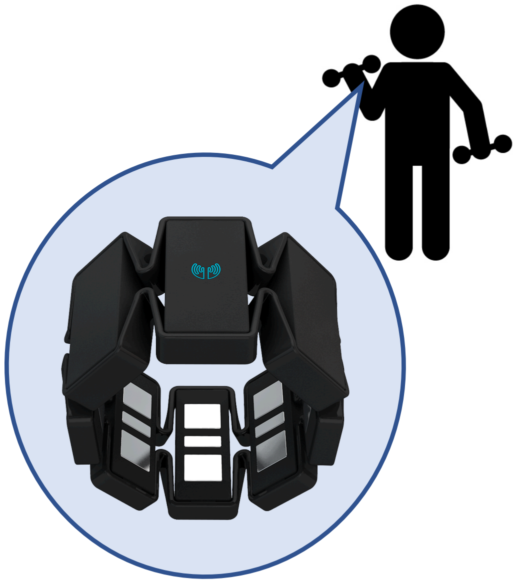


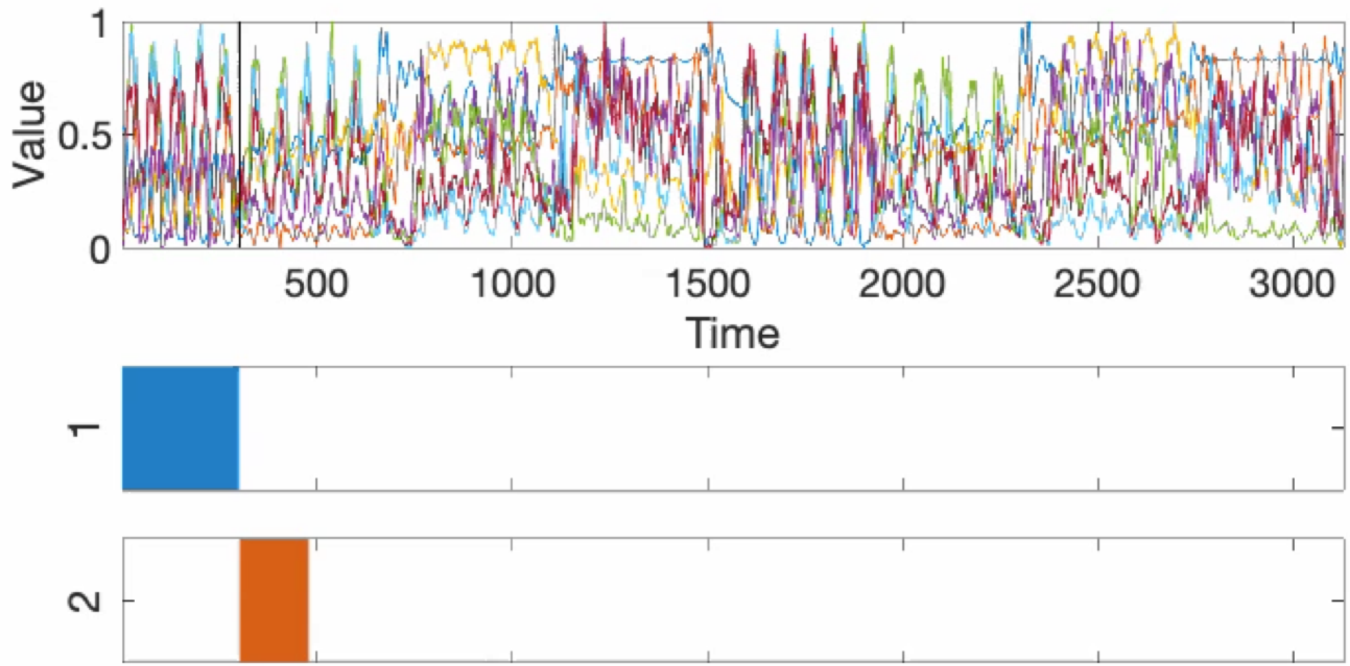
Requirements:

- **Incremental**
  - We cannot access all historical data

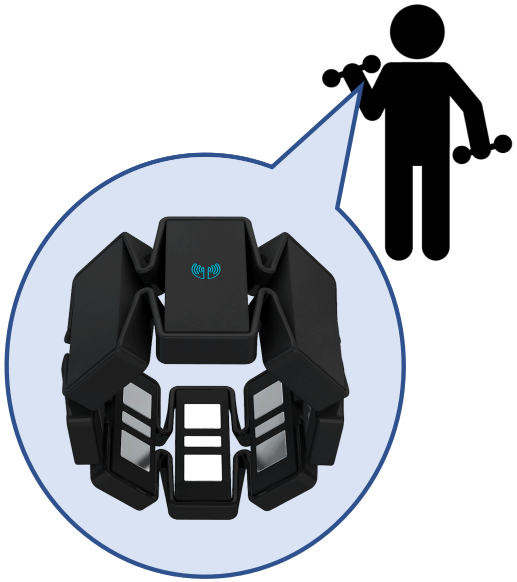
StreamScope: automatic & incremental approach

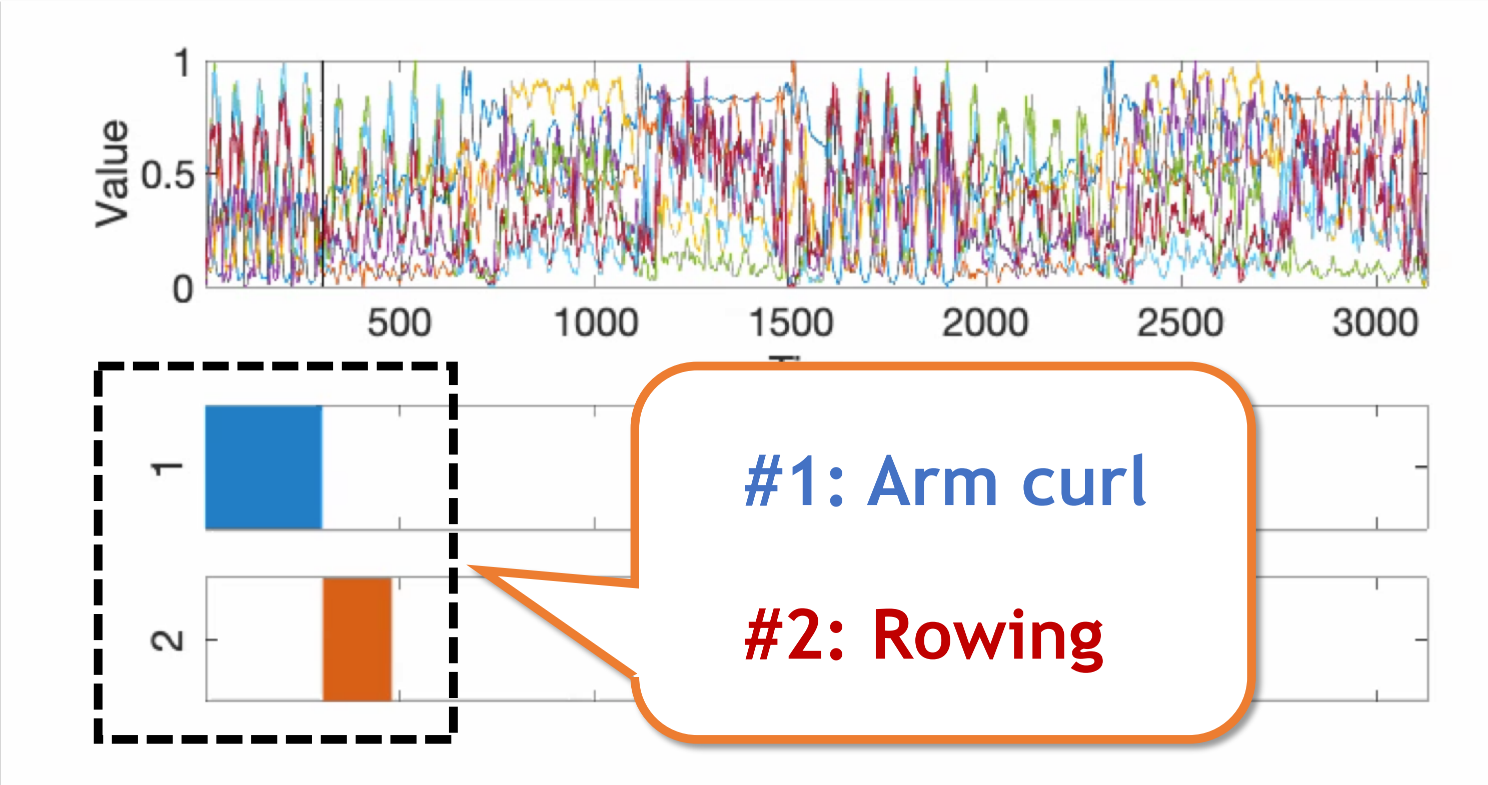
Demo movie



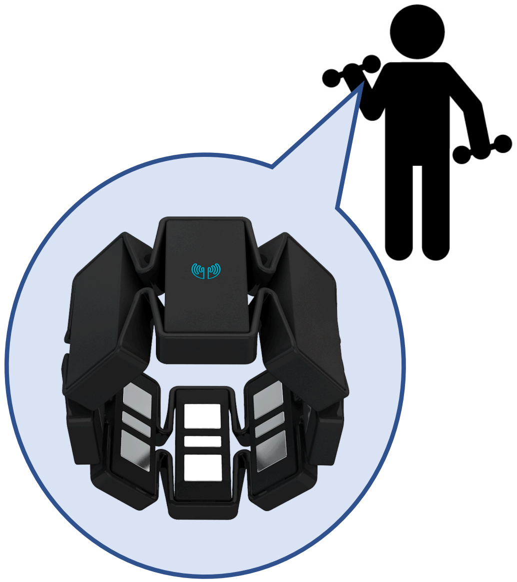


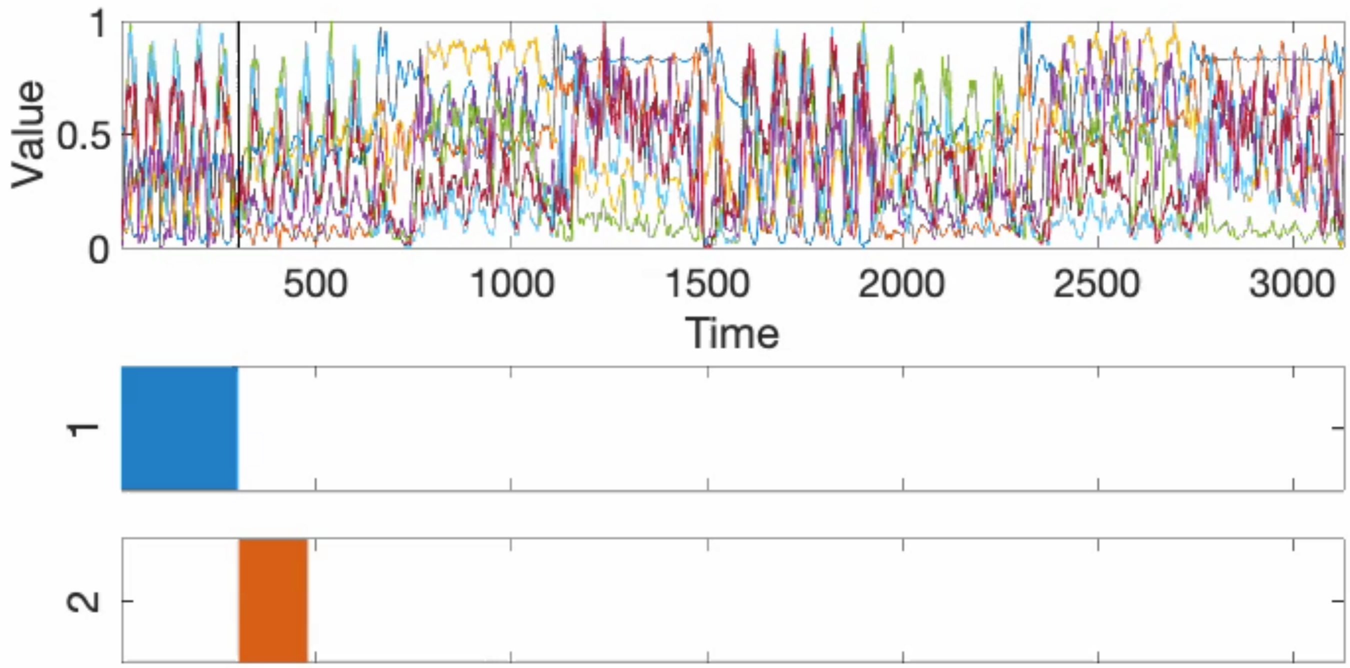
Demo movie





Demo movie





Demo movie

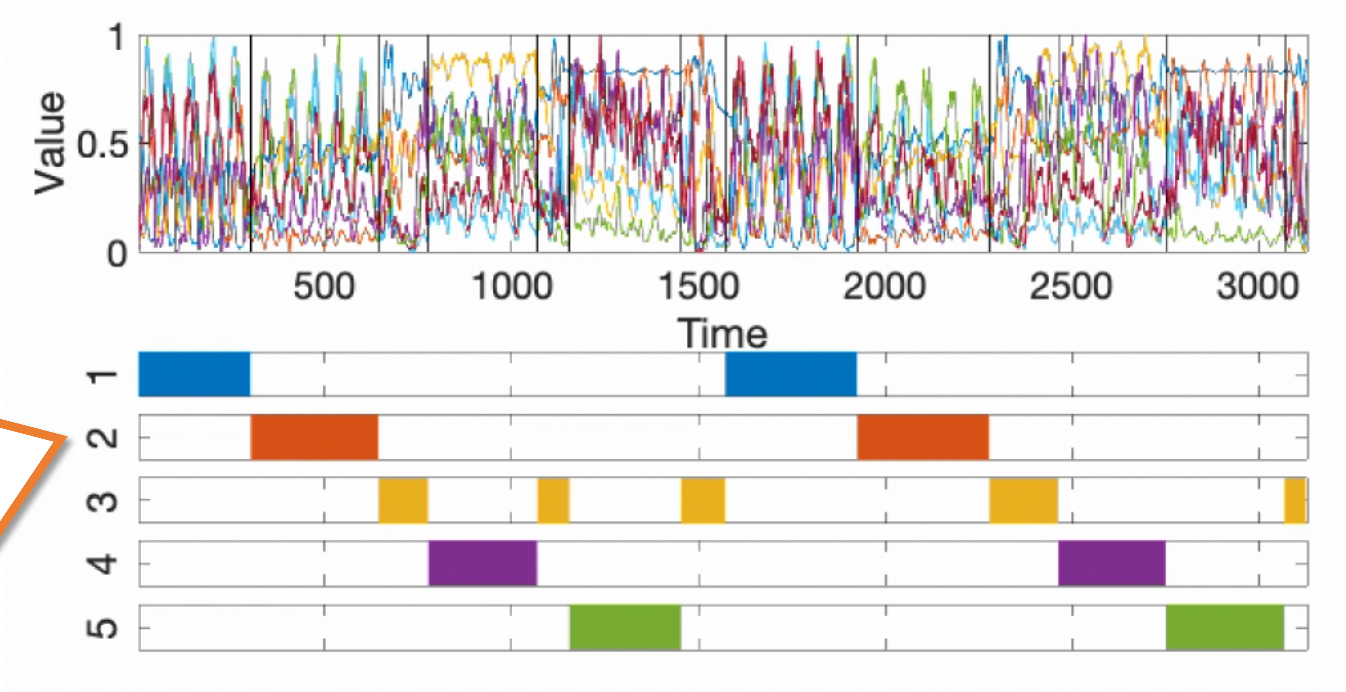
#1: Arm curl

#2: Rowing

#3: Intervals

#4: side raise

#5: Push up



Outline

1. ~~Motivation~~
2. **Problem definition**
3. Model
4. Streaming Algorithm
5. Experiments
6. Conclusions

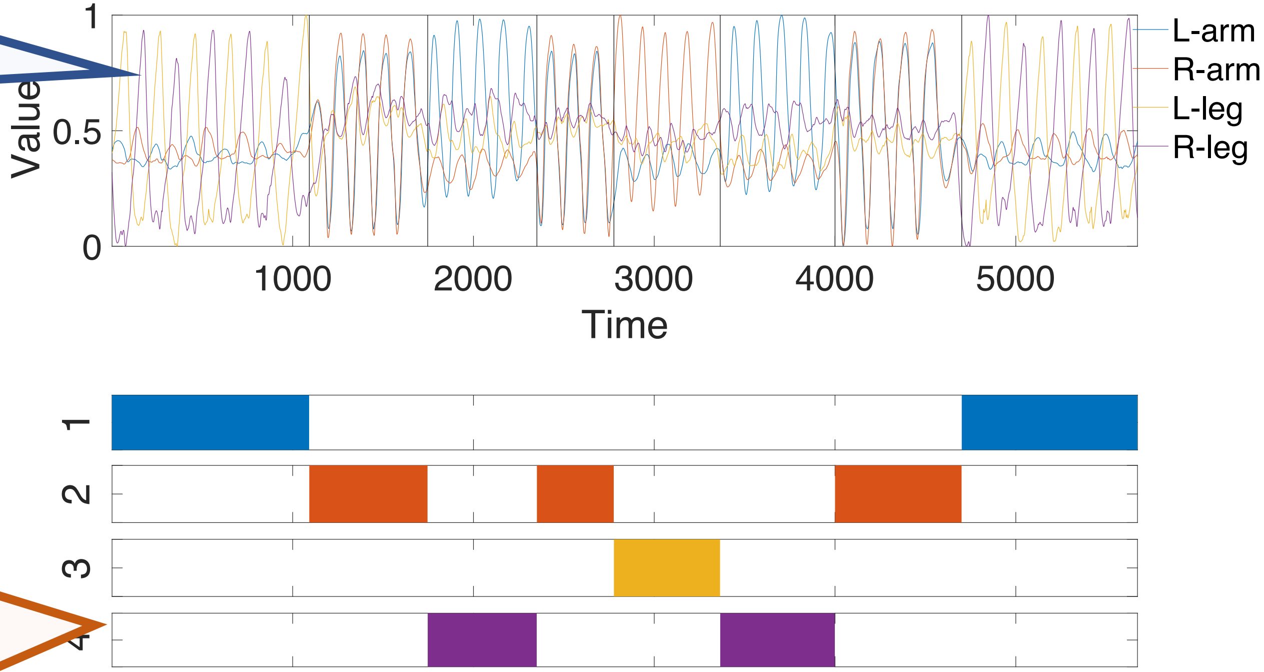
Problem definition

Given:

- Data stream X

Find:

1. Segment: $\mathcal{S}$
2. Regime: Θ
3. Segment-membership: $\mathcal{F}$

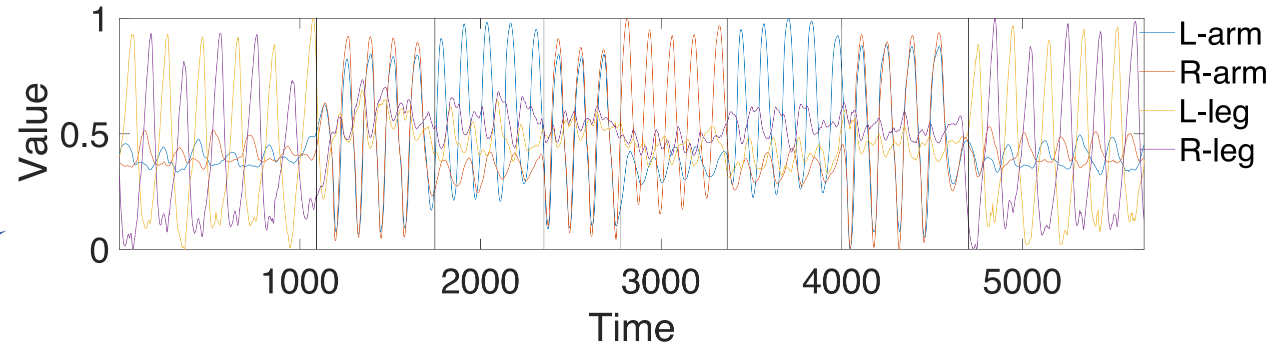


Problem definition

Given Data stream: set of d-dimensional vectors

$$X = \{x_1, \dots, x_t\}$$

$d = 4$

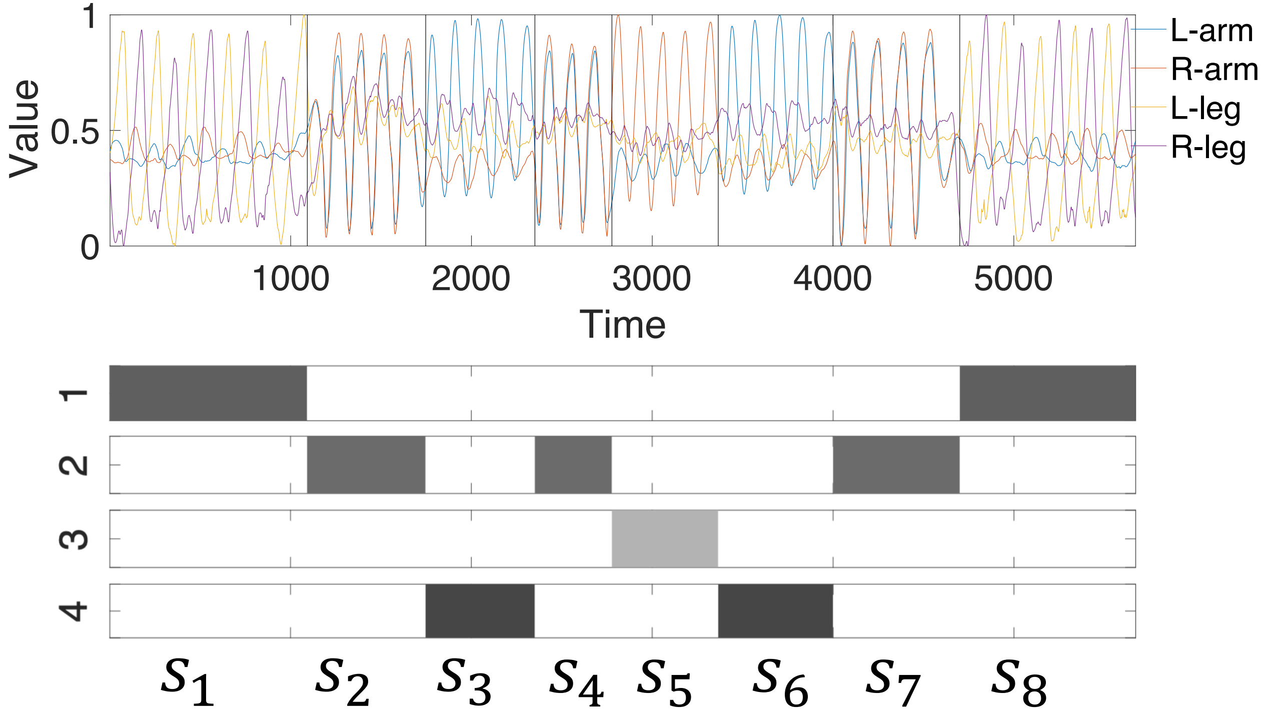


Problem definition

Hidden Segment: start/end positions of each pattern

$$\mathcal{S} = \{s_1, \dots, s_m\}$$

$m = 8$

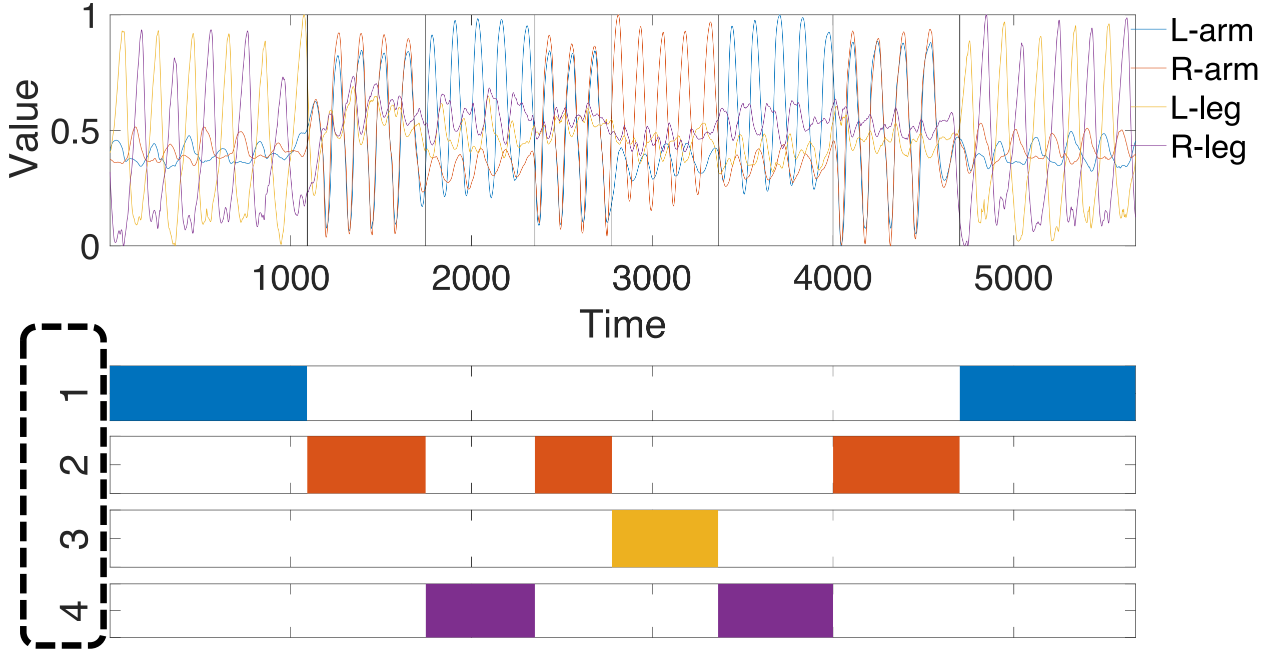


Problem definition

Hidden Regime: segment groups

$$\Theta = \{\theta_1, \ldots, \theta_r, \Phi\}$$

$r = 4$

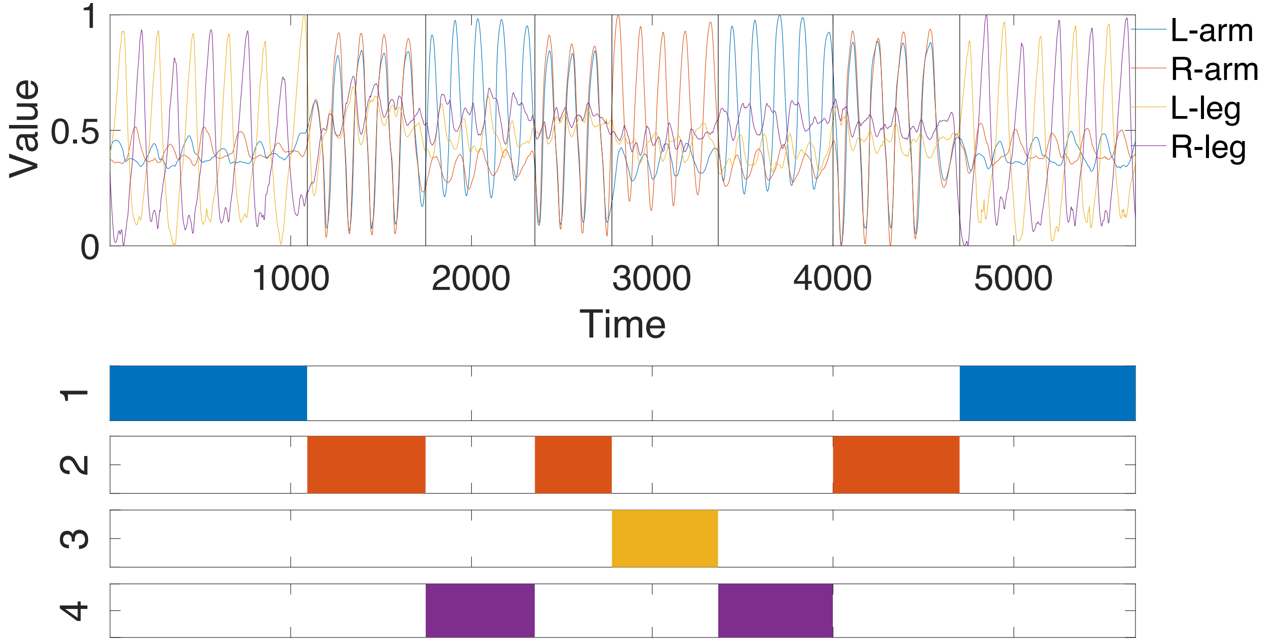


Problem definition

Hidden Segment-membership: regime-assignment

$$\mathcal{F} = \{f(1), \dots, f(m)\}$$

e.g., $f(3) = 4$



$$\mathcal{F} = \{ 1, 2, 4, 2, 3, 4, 2, 1, \}$$

Problem definition

Given: d-dimensional data stream $X = \{x_1, \dots, x_t\}$

Find: compact description $\mathcal{C} = \{m, r, \mathcal{S}, \Theta, \mathcal{F}\}$ of X

=

- m *segments* $\mathcal{S}$
- r *regimes* Θ
- *segment-membership* $\mathcal{F}$

Outline

1. ~~Motivation~~
2. ~~Problem definition~~
3. **Model**
4. Streaming Algorithm
5. Experiments
6. Conclusions

Proposed model

Goal: find compact description C in **a streaming setting**

Challenges:

Q1. How can we represent regimes?

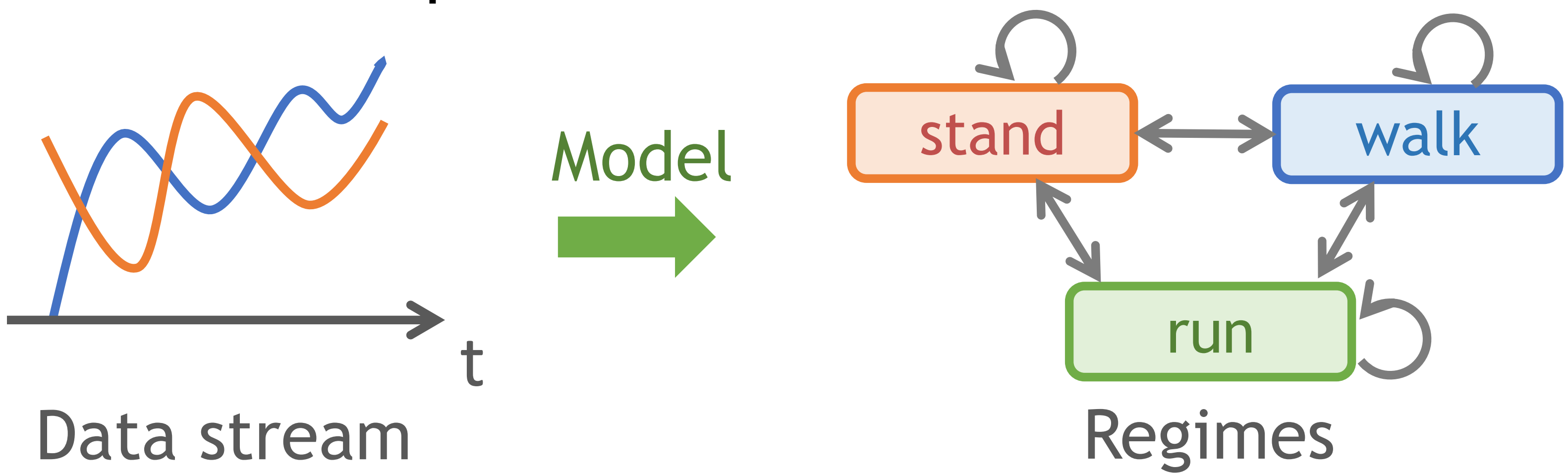
Idea (1): Hierarchical probabilistic model

Q2. How can we decide # of segments/regimes?

Idea (2): Model description cost

Idea (1): hierarchical probabilistic model

Q. How to describe patterns?

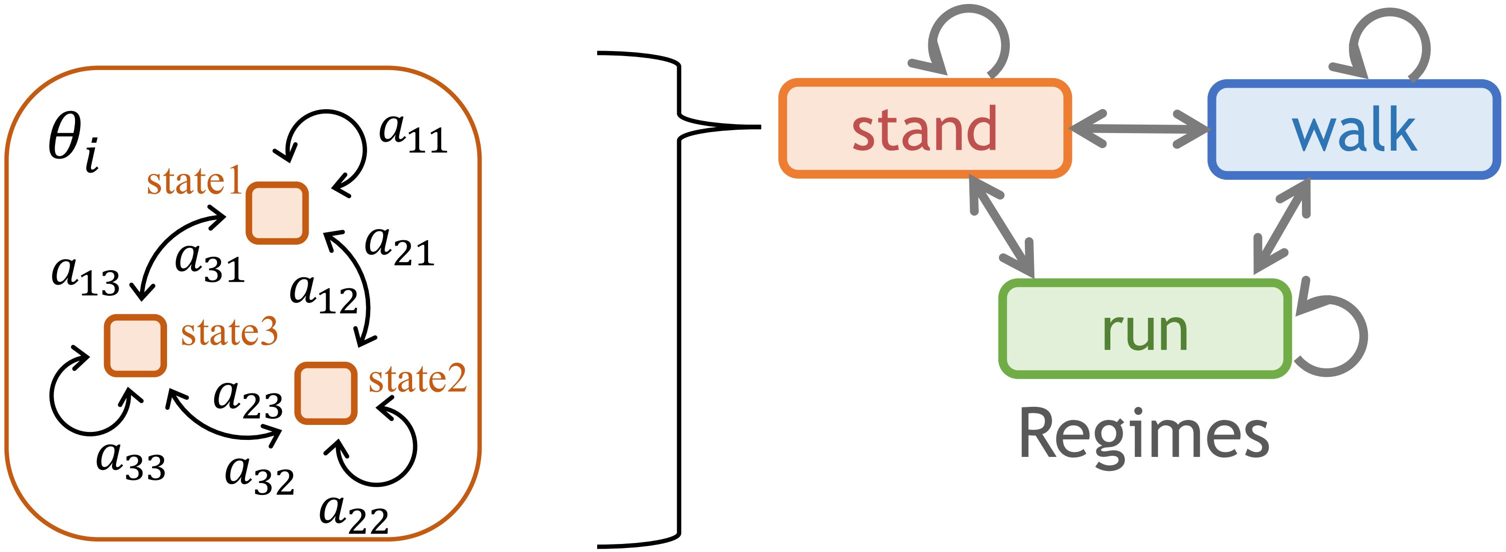


Idea: HMM-based probabilistic model

- **'within-regime'** transitions: A hidden Markov model $\theta = \{\pi, A, B\}$
- **'across-regime'** transitions: Regime transition matrix $\Phi = \{\phi_{ij}\}_{i,j=1}^{r}$

Idea (1): hierarchical probabilistic model

Full model $\Theta = \{\theta_1, \ldots, \theta_r, \Phi\}$

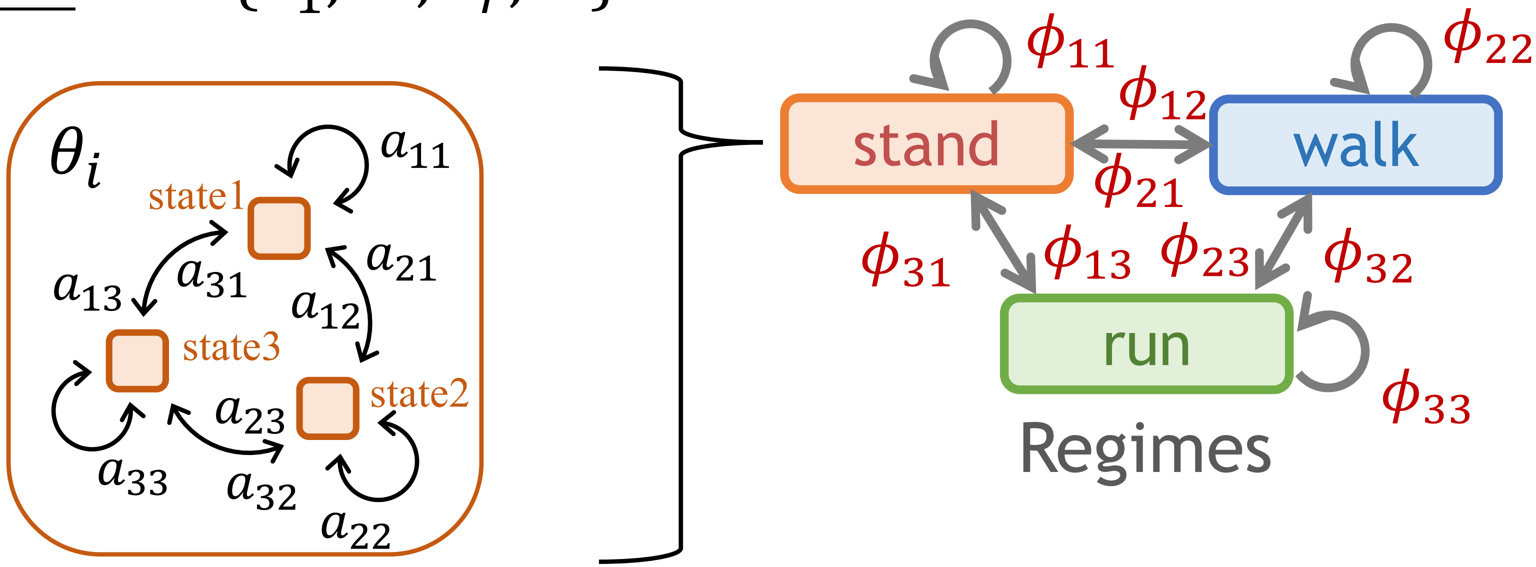


Single HMM parameters:

$$\theta_i = \{\pi_i, A_i, B_i\}$$

Idea (1): hierarchical probabilistic model

Full model $\Theta = \{\theta_1, \ldots, \theta_r, \Phi\}$



Single HMM parameters:
$\theta_i = \{\pi_i, A_i, B_i\}$

Regime transition matrix:
$\Phi = \{\phi_{ij}\}_{i,j=1}^{r}$

Idea (2): Incremental encoding scheme

Q. How to decide # of segments/regimes?

Idea: Minimum description length (MDL)

- Minimize the total description cost of a data stream
- Update 'optimal' # of segments/regimes

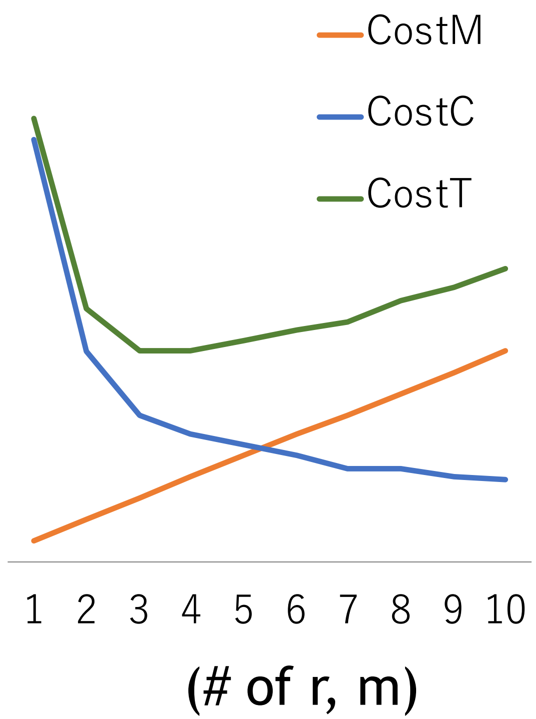
Idea (2): Incremental encoding scheme

Idea: Minimize total encoding cost

$$\min \left(\mathrm{Cost}_M(C) + \mathrm{Cost}_C(X|C) \right)$$

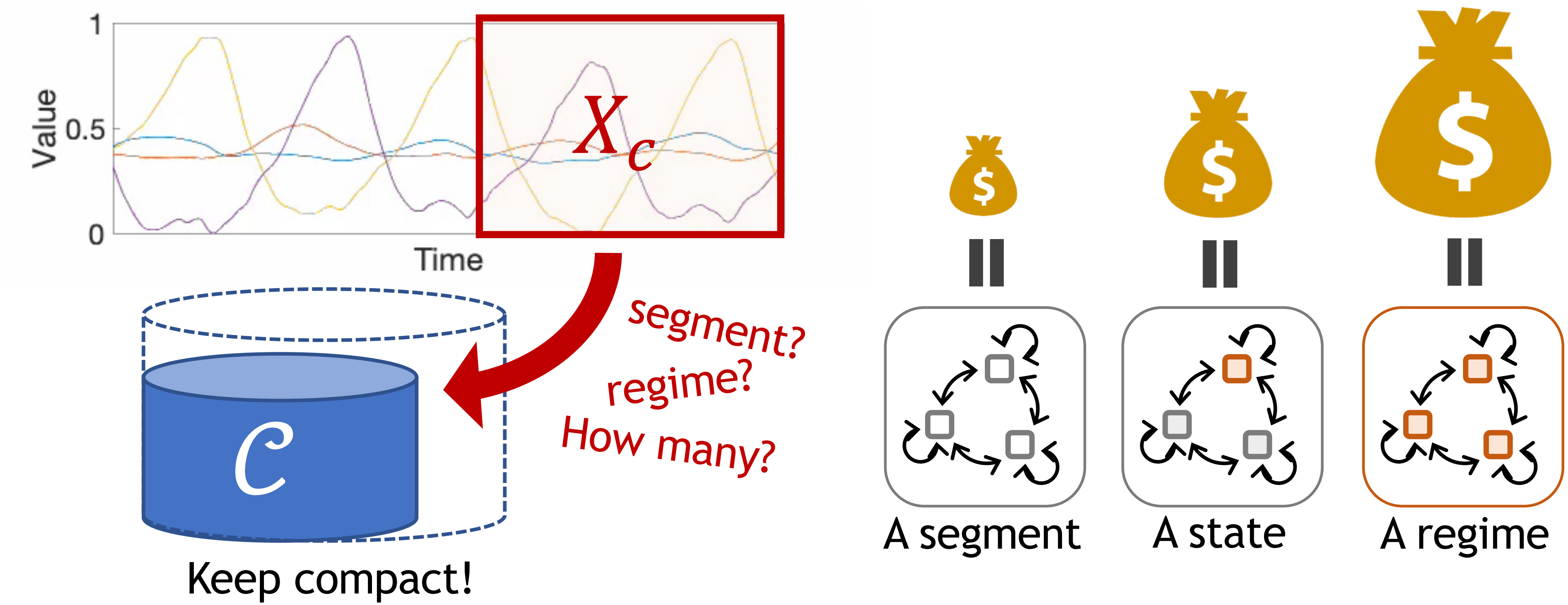
Model cost　　Coding cost

Good compression ⟲ Good description



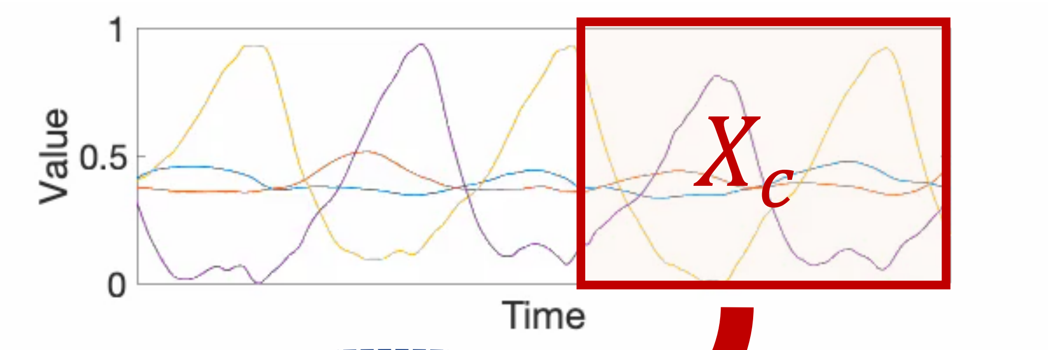
Idea (2): Incremental encoding scheme

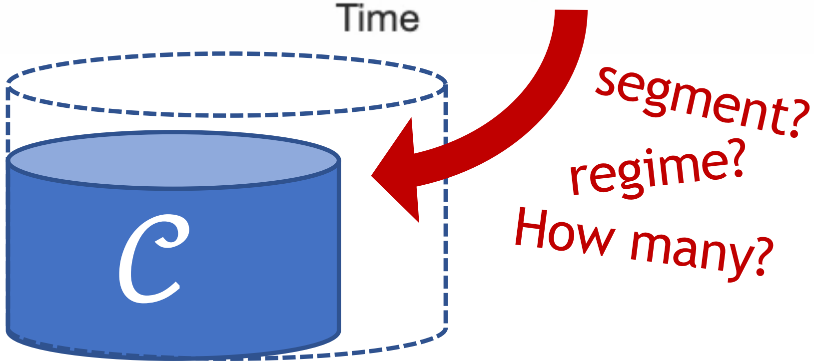
Q. How many new components does $\mathcal{C}$ need?



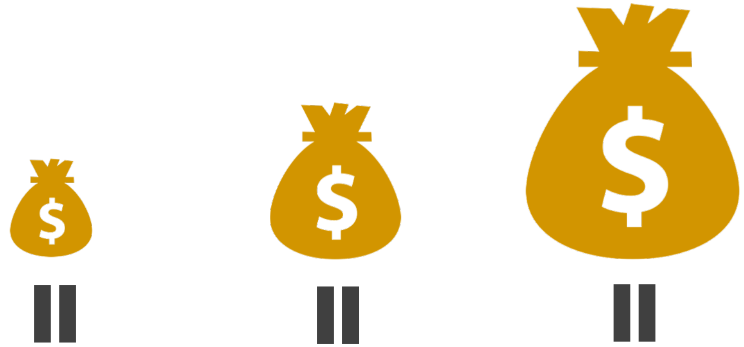
Idea (2): Incremental encoding scheme

Q. How many new components does $\mathcal{C}$ need?





Keep compact!



$$\Delta Cost_T(X_c; C) = \log^*(m_+) - \log^*(m) + \log^*(r_+) - \log^*(r) + \Delta Cost_M(\mathcal{S}) + \Delta Cost_M(\Theta) + \Delta Cost_M(\mathcal{F}) + Cost_C(X_c|\Theta).$$

Details in paper

Outline

1. ~~Motivation~~
2. ~~Problem definition~~
3. ~~Model~~
4. **Streaming Algorithm**
5. Experiments
6. Conclusions

Streaming algorithms

- Algorithms

Main

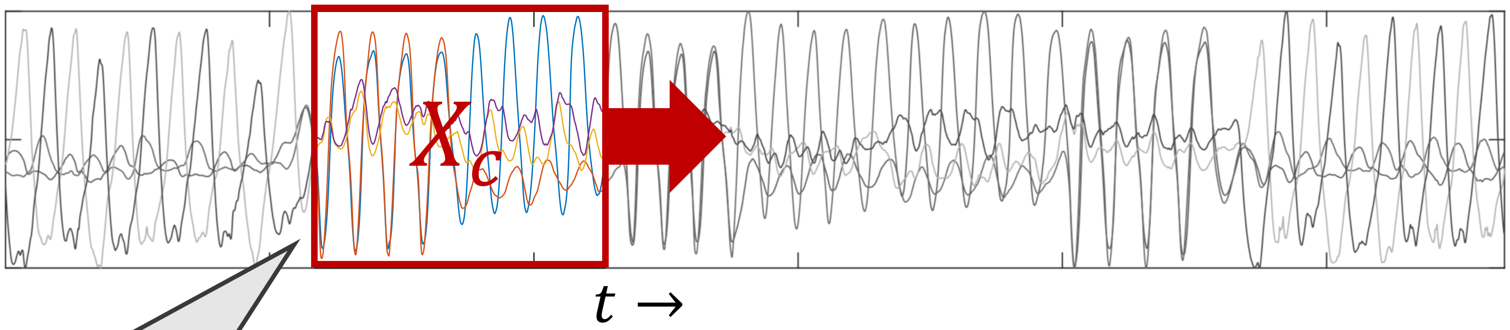
StreamScope
Optimize/update parameter set $\mathcal{C}$

1. **SegmentAssignment**
   Identify regime transitions & segments
2. **RegimeGeneration**
   Estimate new regimes θ

StreamScope

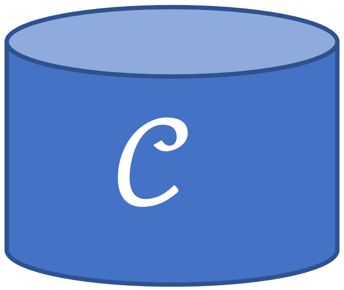
- <u>Overview</u>

Data stream X



1. Keep current window:
- The latest segment, s_m
- New observations, $x_t, \ldots$

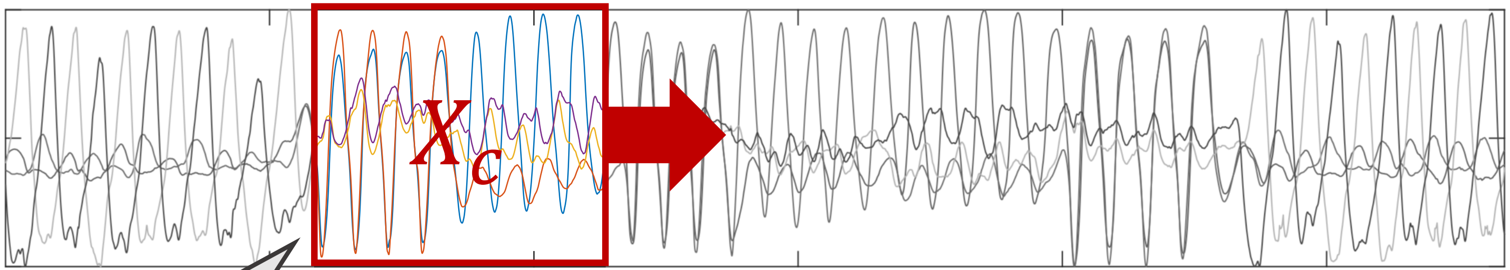
$$X_c = s_m \cup x_t$$



StreamScope

- <u>Overview</u>

Data stream X

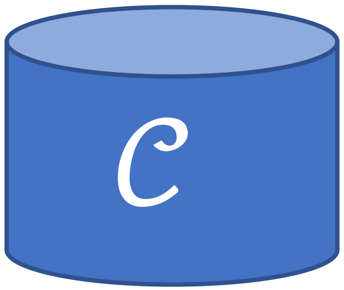


$t \rightarrow$

1. Keep current window:

- The latest segment, s_m
- New observations, $x_t, \ldots$

$$X_c = s_m \cup x_t$$



2. Update model set $\mathcal{C}$

- Minimize $\Delta Cost_T(X_c|\mathcal{C})$

Increase segments?
(SegmentAssignment)
vs.
Increase states/regimes?
(RegimeGeneration)

StreamScope

- <u>Overview</u>

Data stream X

$t \rightarrow$

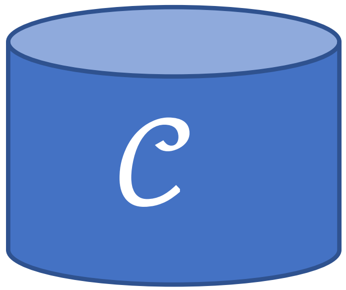
3. Update X_c

- If pattern has changed

1. Keep current window:

- The latest segment, s_m
- New observations, $x_t, \dots$

$$X_c = s_m \cup x_t$$



2. Update model set $\mathcal{C}$

- Minimize $\Delta Cost_T(X_c|\mathcal{C})$
  - Increase segments? (SegmentAssignment)
  - vs.
  - Increase states/regimes? (RegimeGeneration)

1. SegmentAssignment

Given:

- Observation $\boldsymbol{x_t}$
- Model parameter set $\Theta = \{\theta_1, \ldots, \theta_r, \Phi\}$

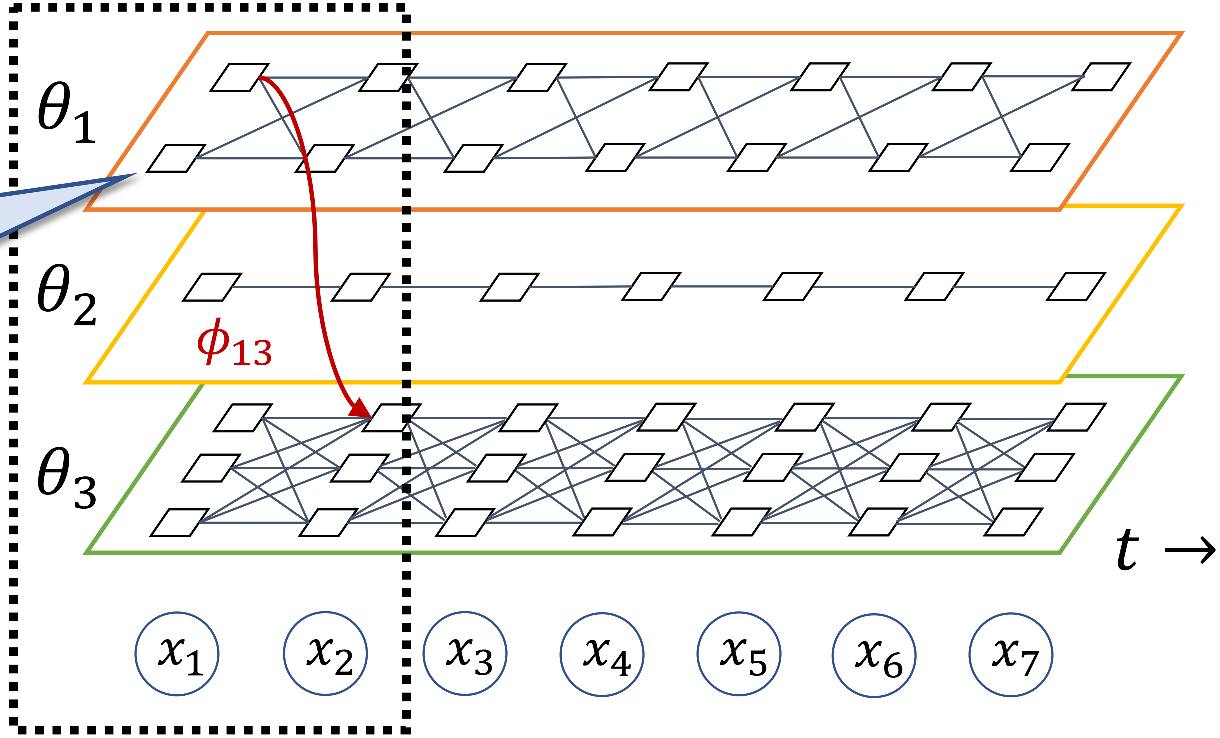
Find:

- Optimal cut point between regimes: $\{m, \mathcal{S}, \mathcal{F}\}$

1. SegmentAssignment

Overview

Dynamic programing algorithm to compute $P(x_t|\Theta)$

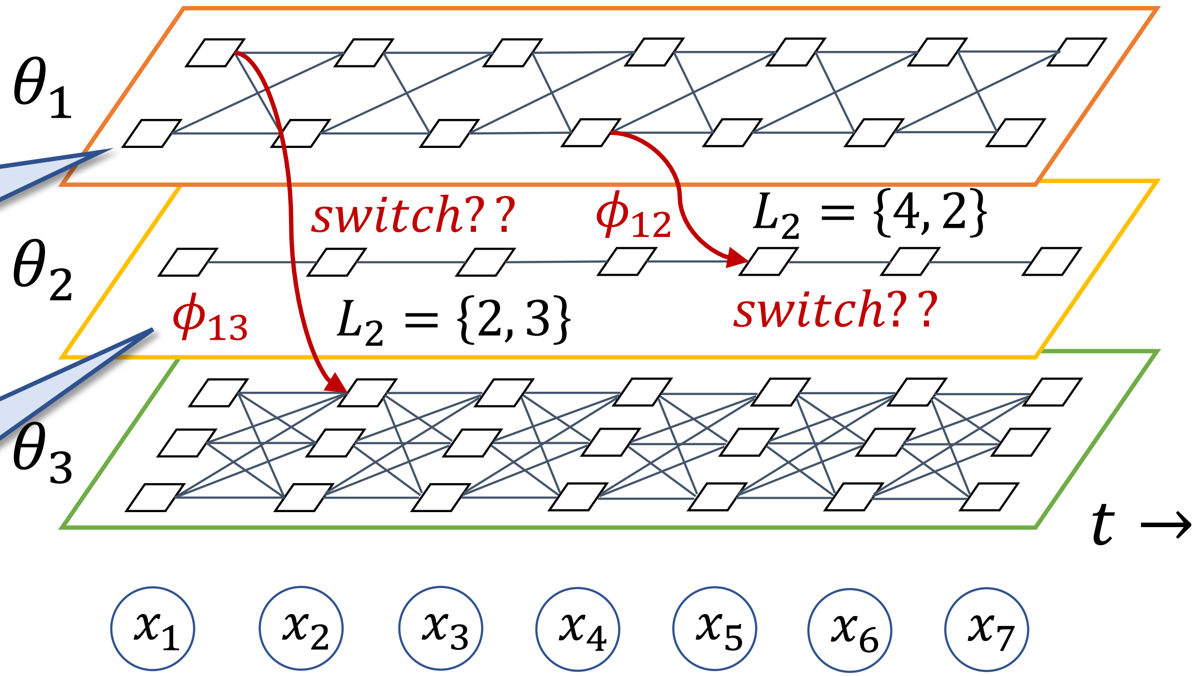


1. SegmentAssignment

Overview

Dynamic programing algorithm to compute $P(x_t|\Theta)$

Keep all candidate cut points $\mathcal{L} = \{L_1, L_2, \ldots\}$



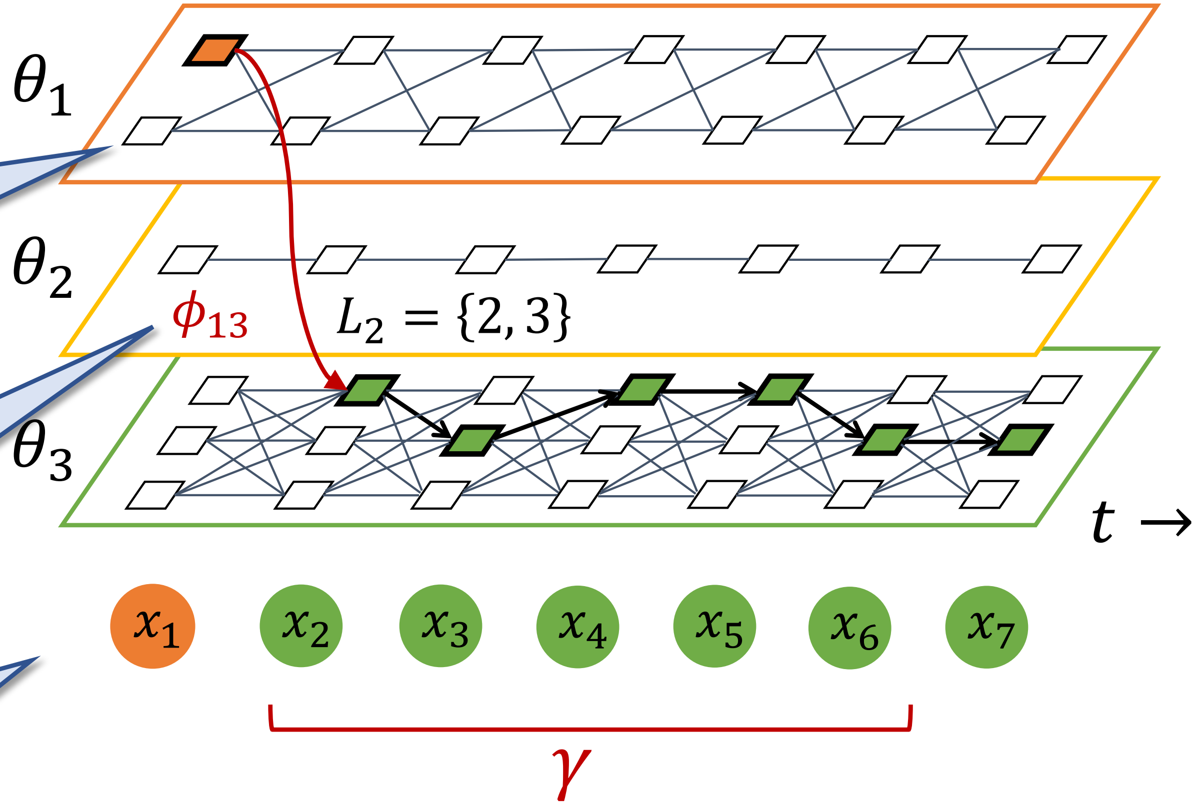
1. SegmentAssignment

Overview

Dynamic programing algorithm to compute $P(x_t|\Theta)$

Keep all candidate cut points $\mathcal{L} = \{L_1, L_2, \ldots\}$

$\gamma -$ guarantee: $\gamma \propto mean(|s|)$



2. RegimeGeneration

Given:

- Current window X_c

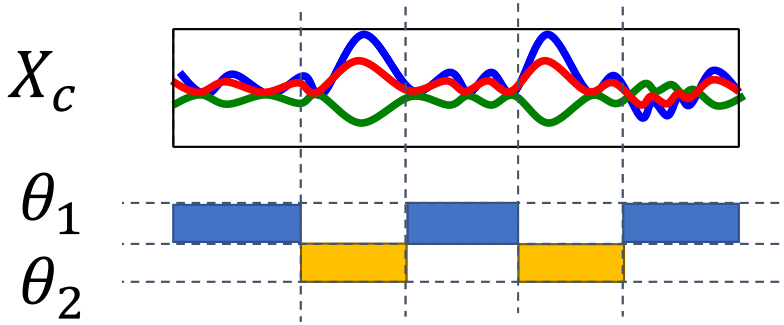
Find:

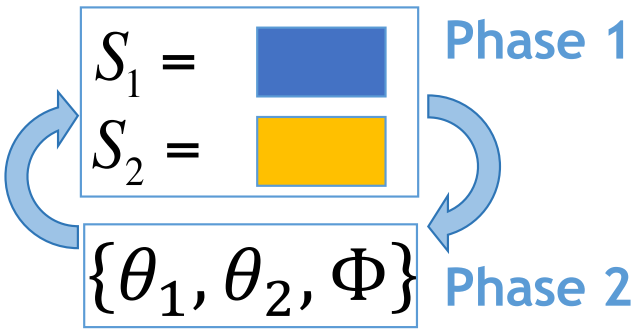
- New regimes: parameter set $\{m,\ r,\ \mathcal{S},\ \Theta,\ \mathcal{F}\}$ for X_c

2. RegimeGeneration

1. Two phase iterative approach

- **Phase1:** split segments into 2 groups
- **Phase2:** update 2 model parameters





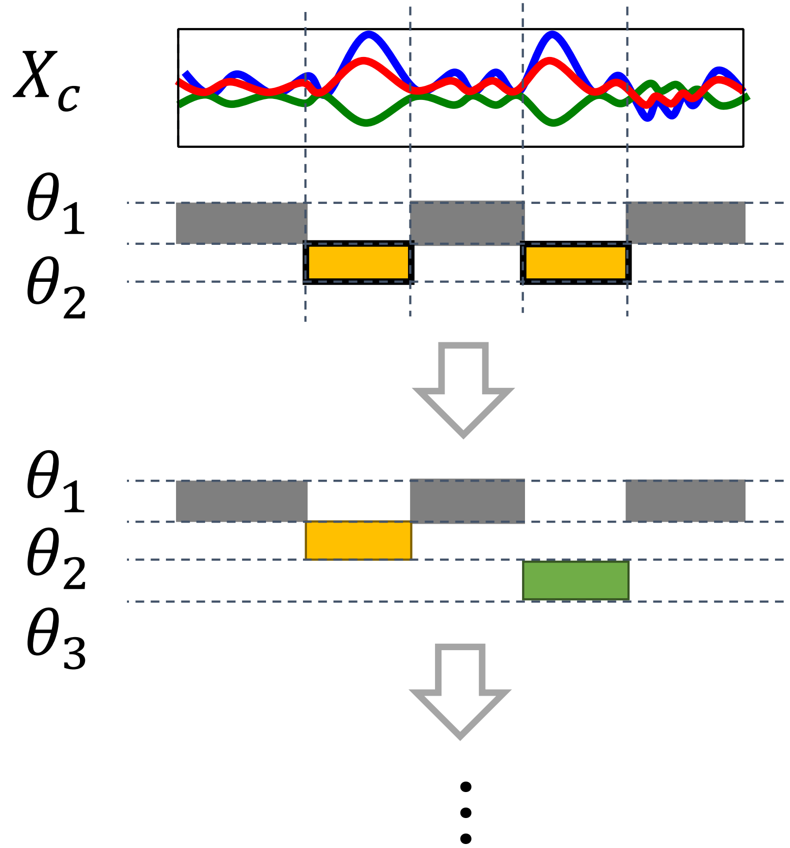
2. RegimeGeneration

1. Two phase iterative approach

- **Phase1:** split segments into 2 groups
- **Phase2:** update 2 model parameters

2. Recursively split new regimes

- While total cost can be reduced



Outline

1. ~~Motivation~~
2. ~~Problem definition~~
3. ~~Model~~
4. ~~Streaming Algorithm~~
5. **Experiments**
6. Conclusions

Experiments

- We answer the following questions:

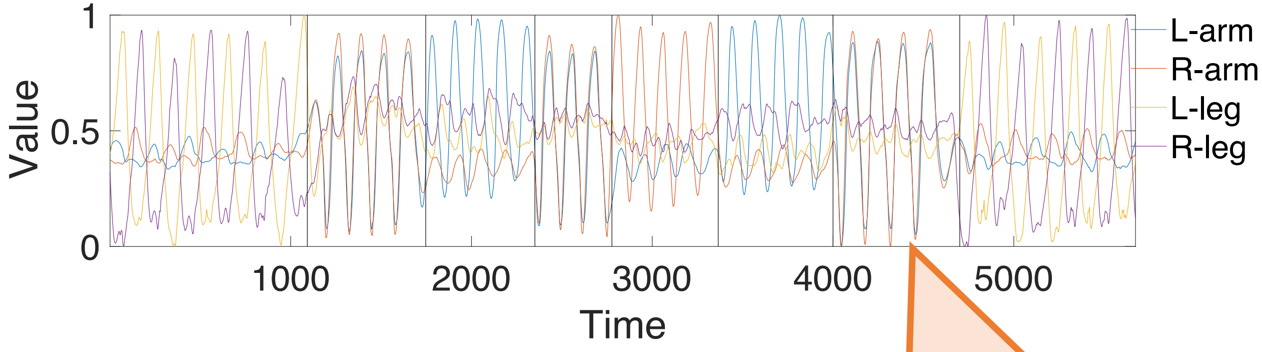
Q1. Effectiveness:
How successful is it in discovering patterns?

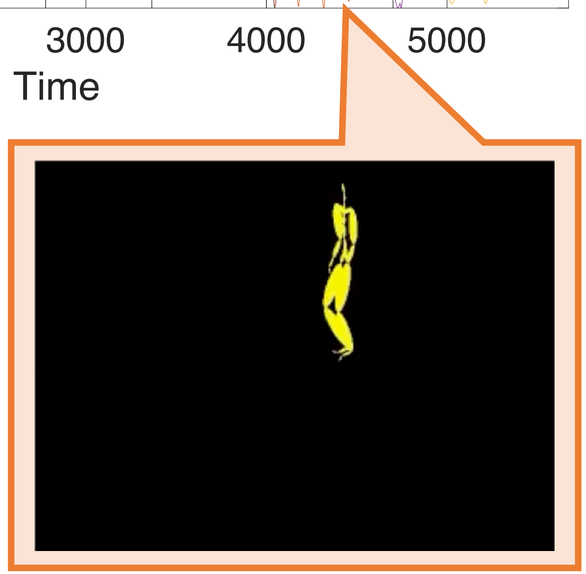
Q2. Accuracy:
How well does it find cut-points & regimes?

Q3. Scalability:
How does it scale in terms of time & memory consumption?

Q1. Effectiveness - #Mocap

- MoCap sensor stream





Q1. Effectiveness - #Mocap

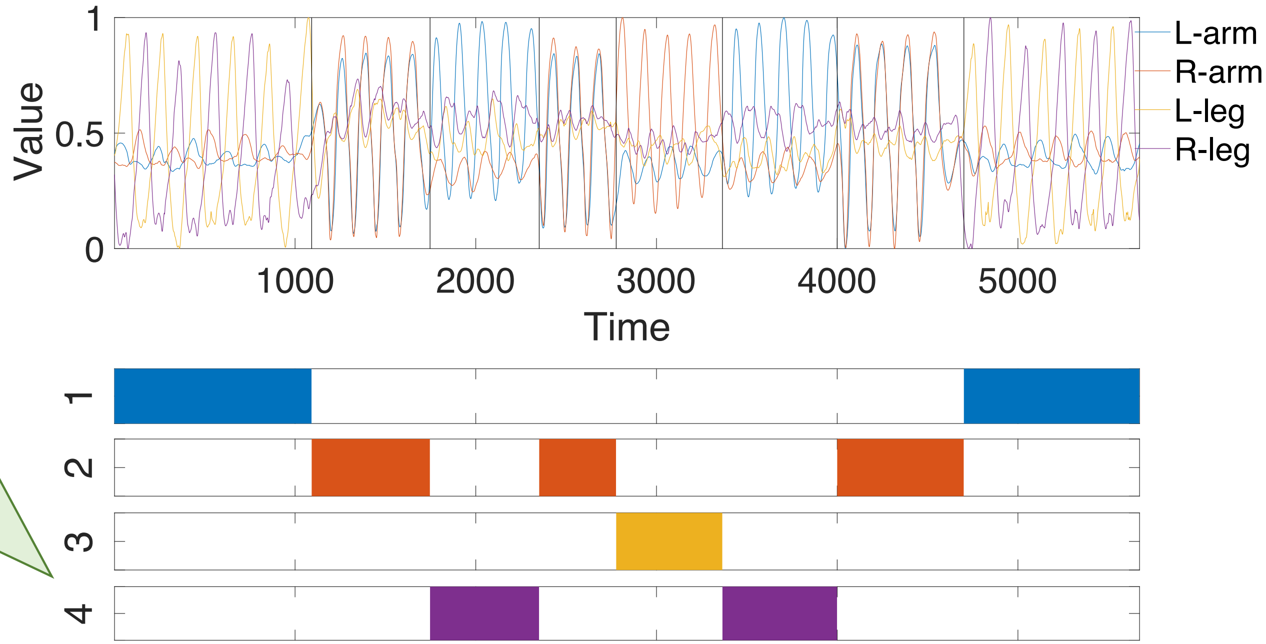
- MoCap sensor stream

#1 Going straight

#2 Stretching arms

#3 Stretching right arm

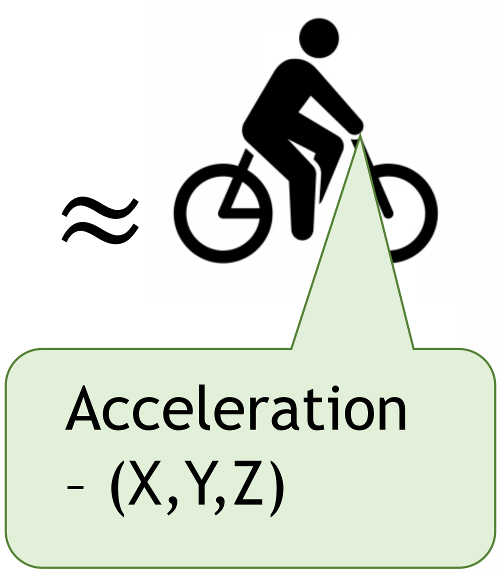
#4 Stretching left arm

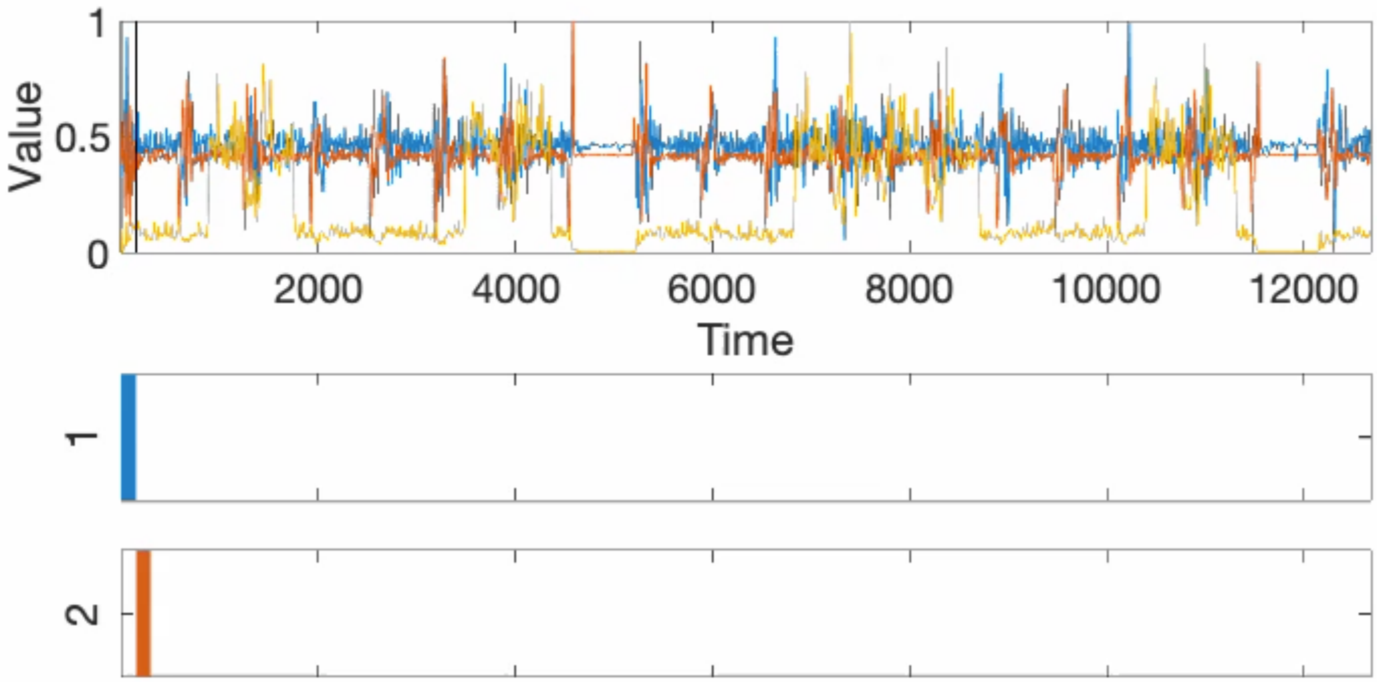


StreamScope can find intuitive patterns **automatically**

Q1. Effectiveness - #Bicycle

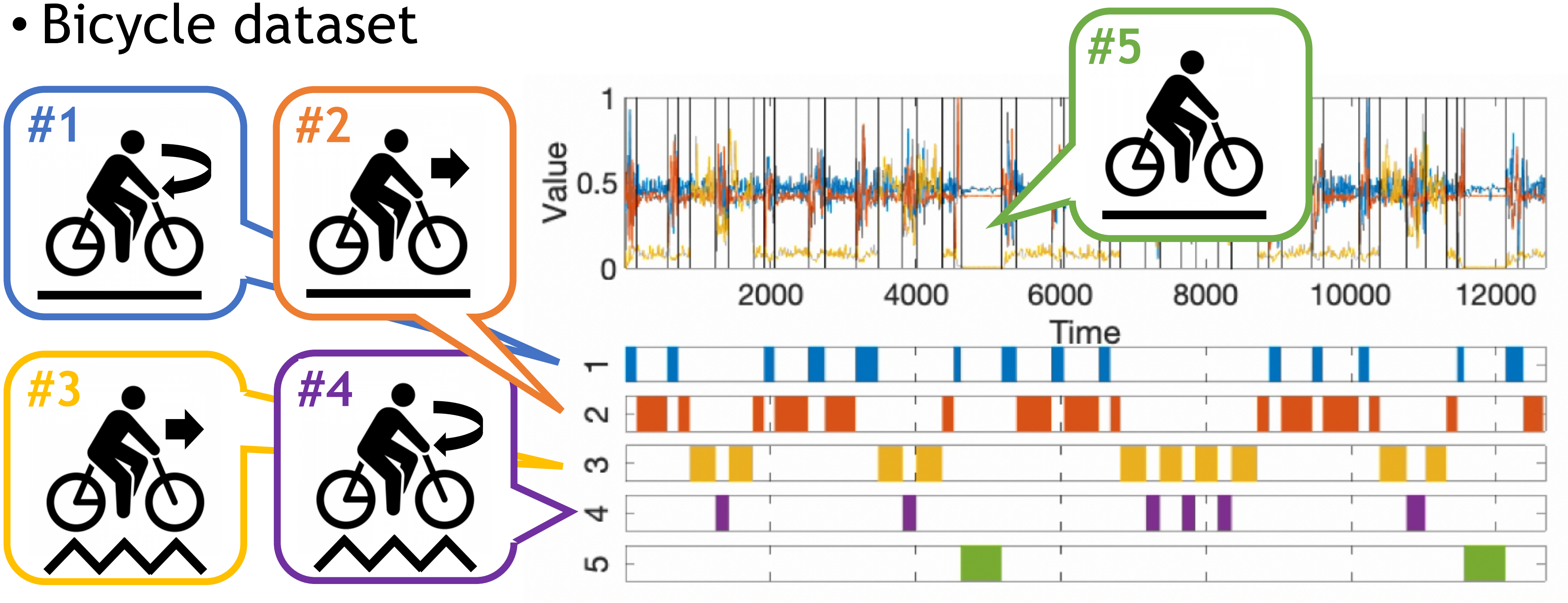
- Bicycle dataset





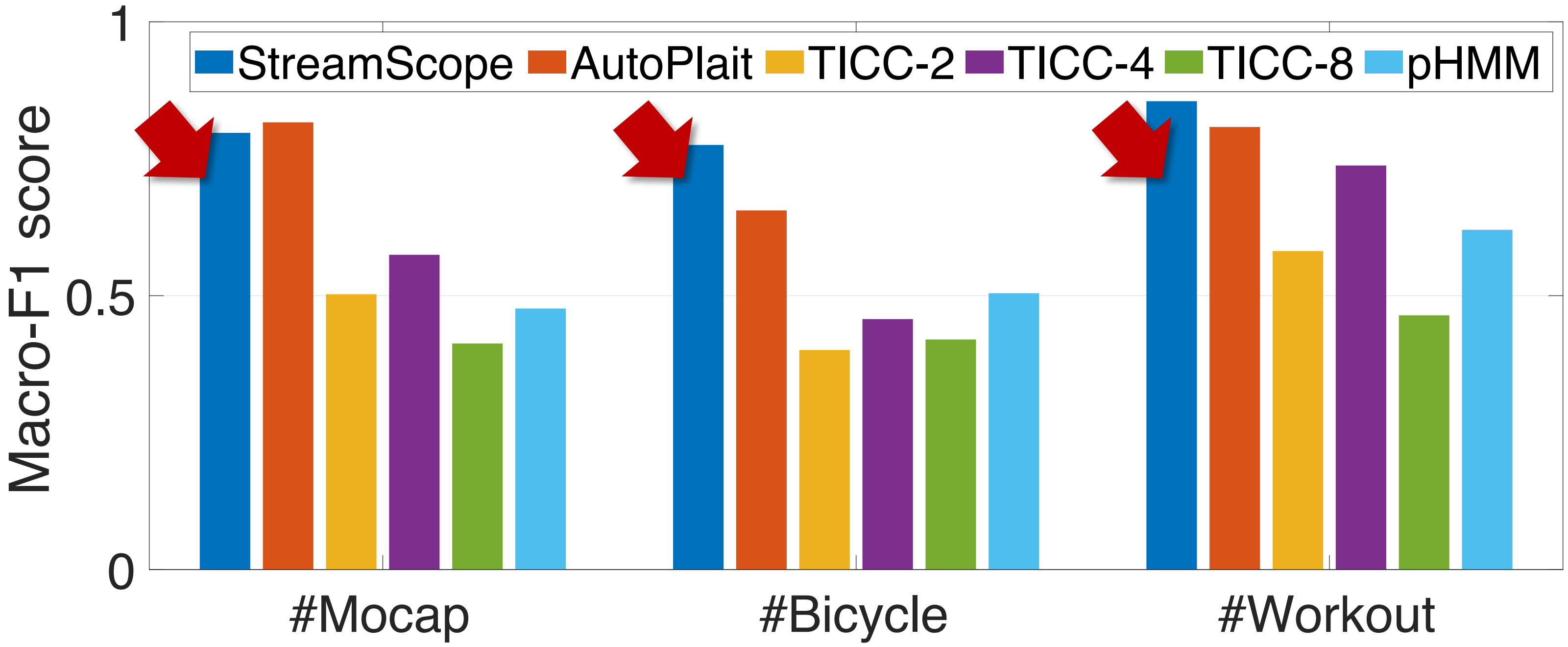
Q1. Effectiveness - #Bicycle

- Bicycle dataset



Q2. Accuracy

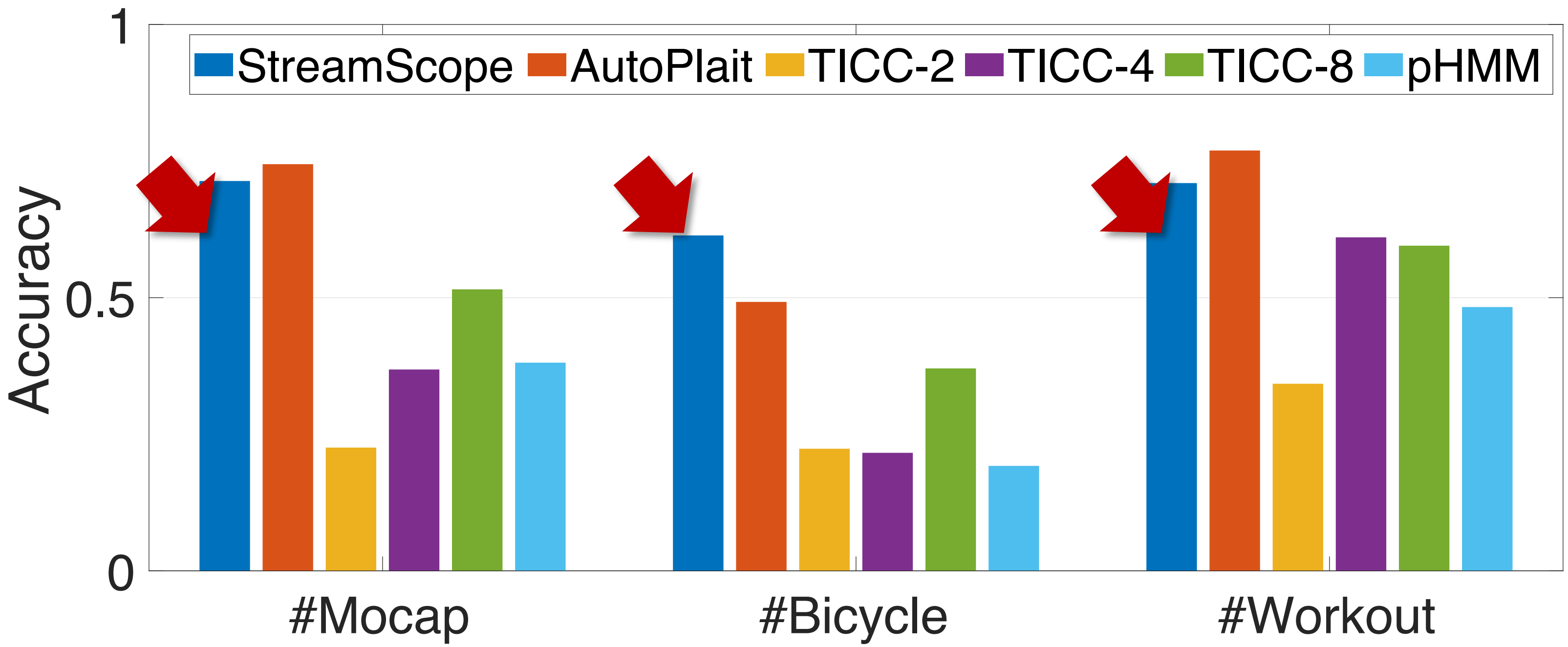
- **Segmentation** accuracy (**higher is better**)



Good accuracy compared with other methods

Q2. Accuracy

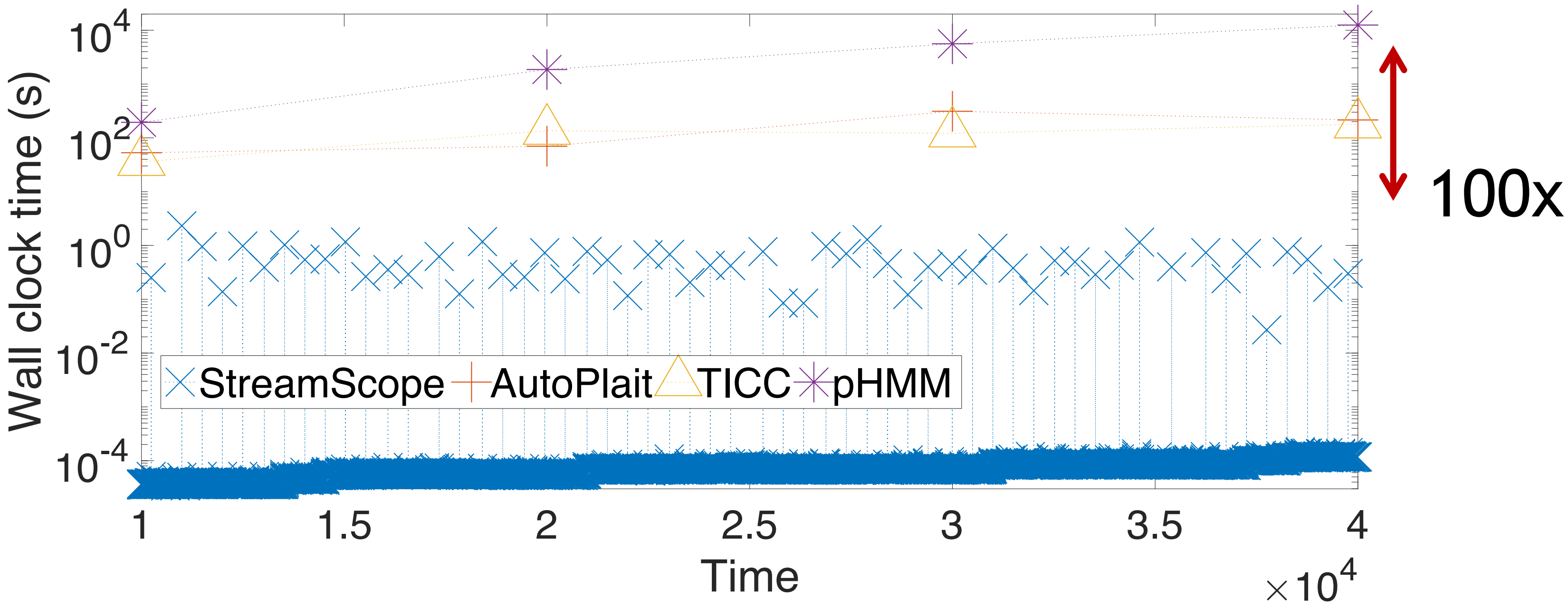
- **Clustering** accuracy (**higher is better**)



Good accuracy compared with other methods

Q3. Scalability

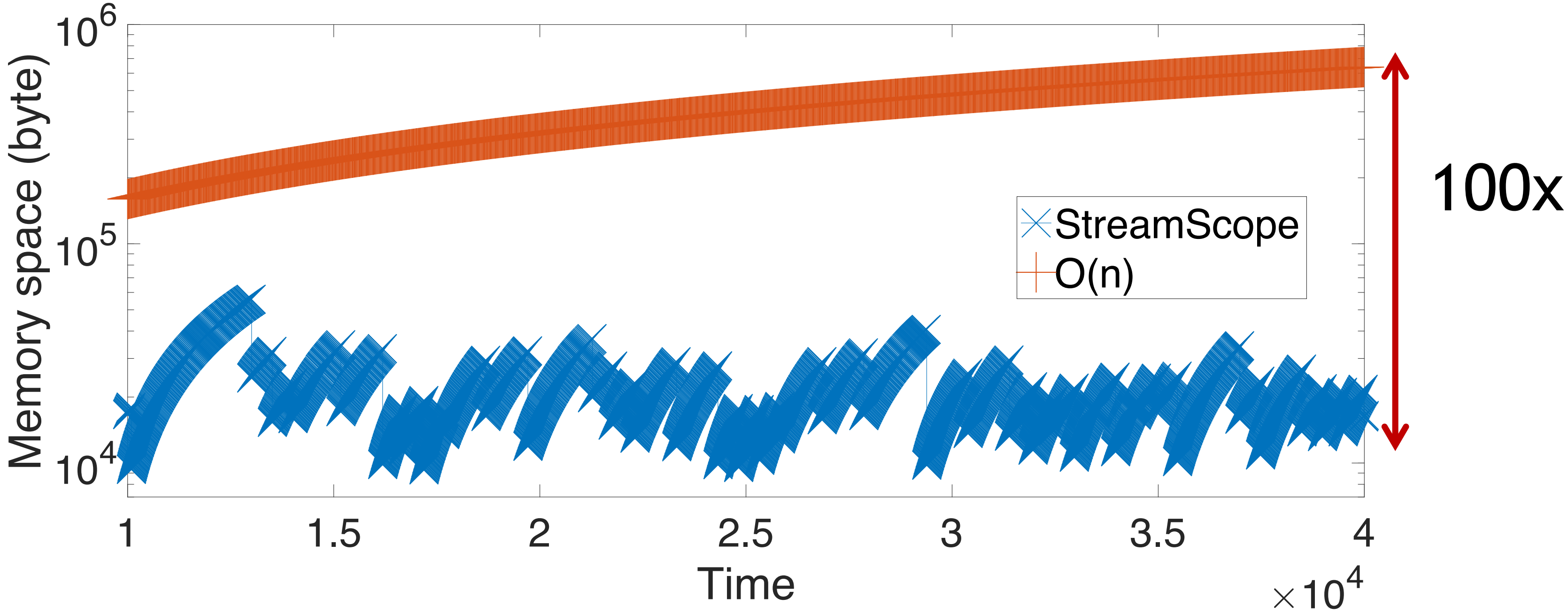
- Wall clock time vs. stream length



The complexity is **independent of the data length**

Q3. Scalability

- Memory space vs. stream length



The complexity is **independent of the data length**

Conclusions

StreamScope has the following advantages:

- **Effective:**
  Find optimal segments/regimes
- **Adaptive:**
  Automatic and incremental
- **Scalable:**
  It does not depend on data length

Thank you !