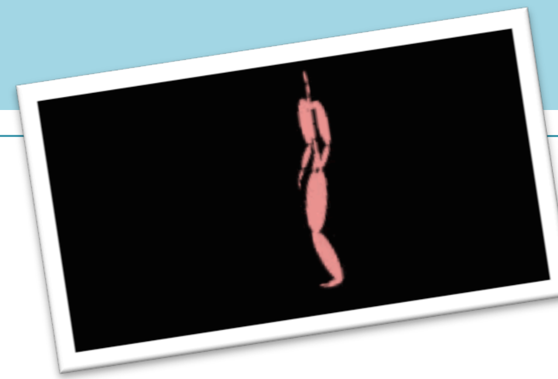
Fast and Exact Monitoring of Co-evolving Data Streams

Speaker: Yasushi Sakurai

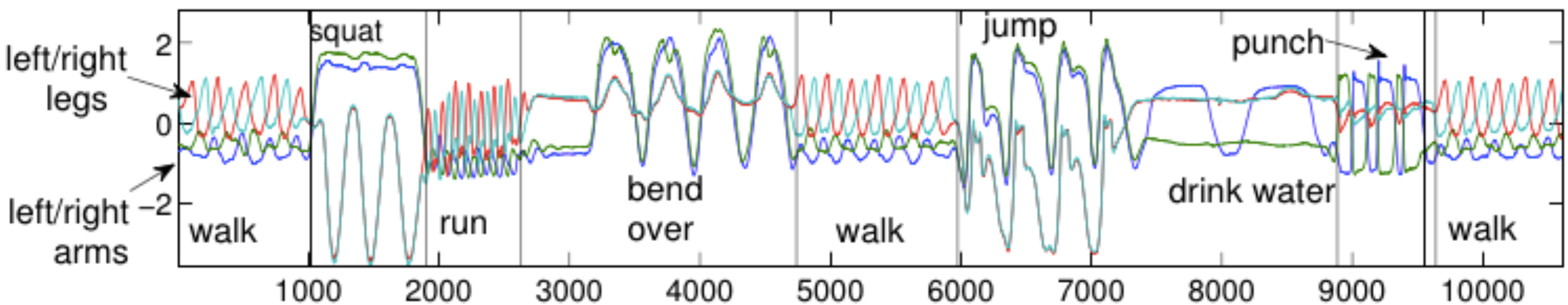
Yasuko Matsubara, Yasushi Sakurai (Kumamoto University)
Naonori Ueda (NTT)
Masatoshi Yoshikawa (Kyoto University)

Motivation



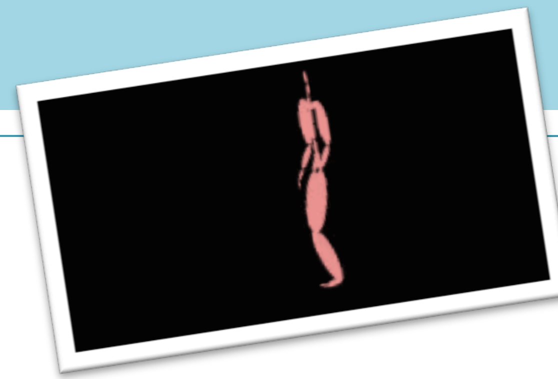
Given: co-evolving data streams

– e.g., MoCap (leg/arm sensors)



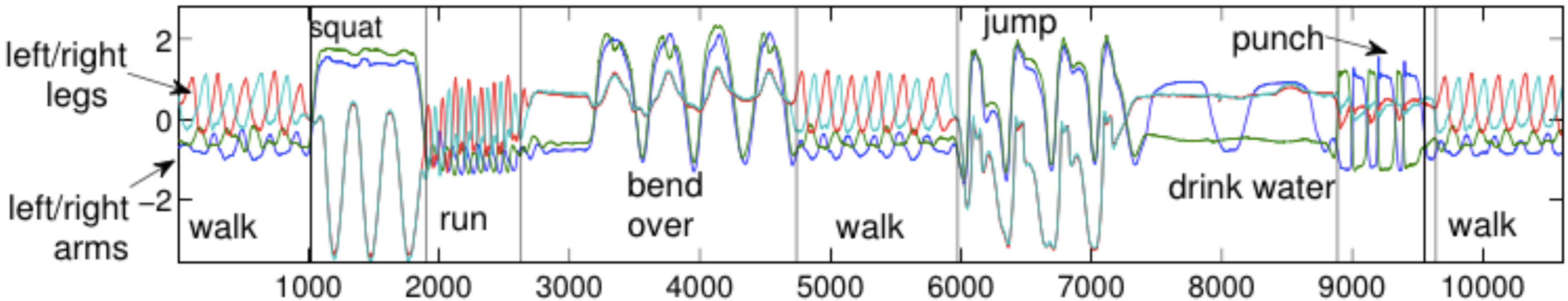
(b) Original *MoCap* stream

Motivation



Given: co-evolving data streams

– e.g., MoCap (leg/arm sensors)



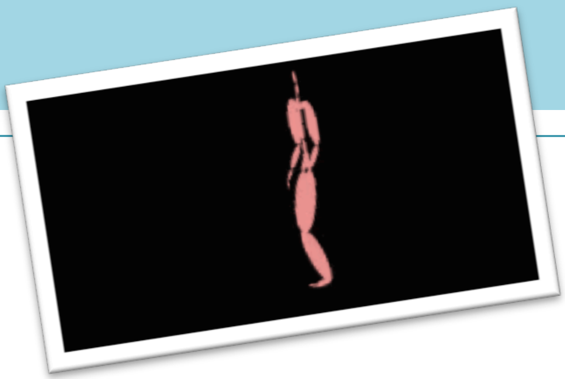
(b) Original *MoCa* stream

Multiple distinct patterns

e.g., Walking, punching

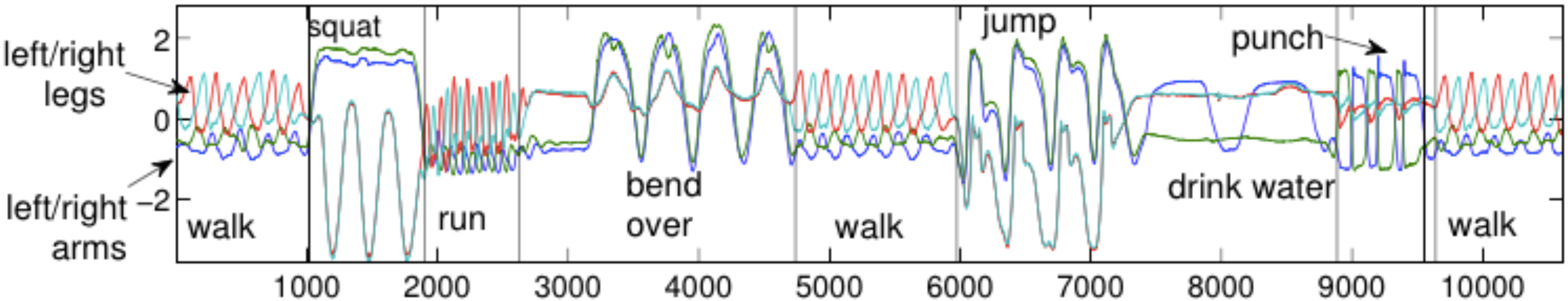
different durations

Motivation



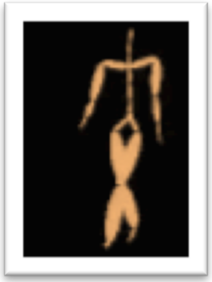
Given: co-evolving data streams

– e.g., MoCap (leg/arm sensors)



(b) Original *MoCap* stream

Queries:

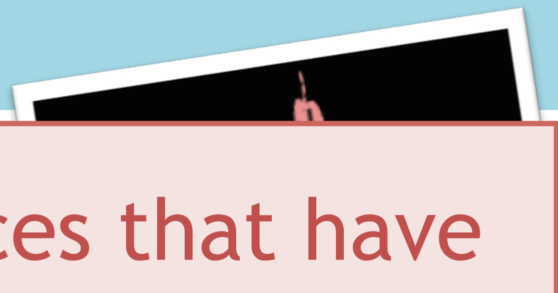


Q1: Walking

Q2: Running

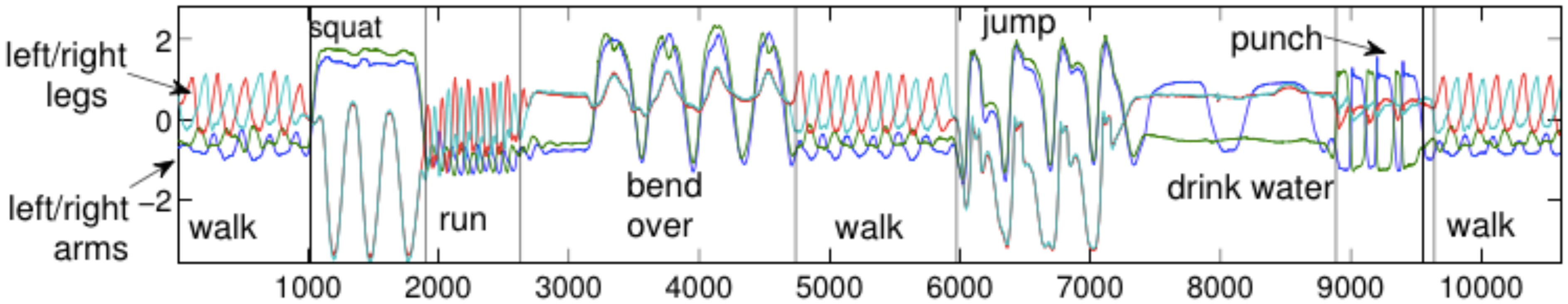
Q3: Punching

Motivation



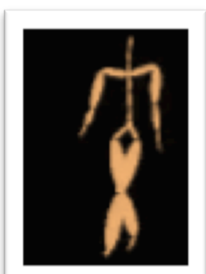
G

Q. Can we find subsequences that have the characteristics of query Θ ?

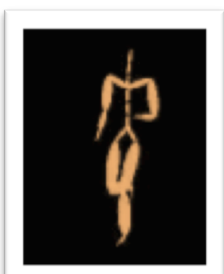


(b) Original *MoCap* stream

Queries:



Q1: Walking

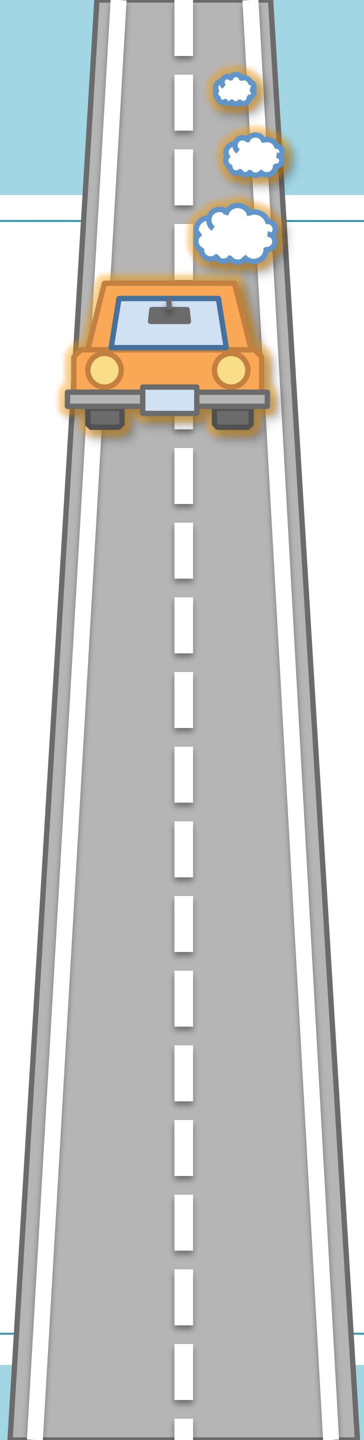


Q2: Running

Q3: Punching

Outline

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



Background

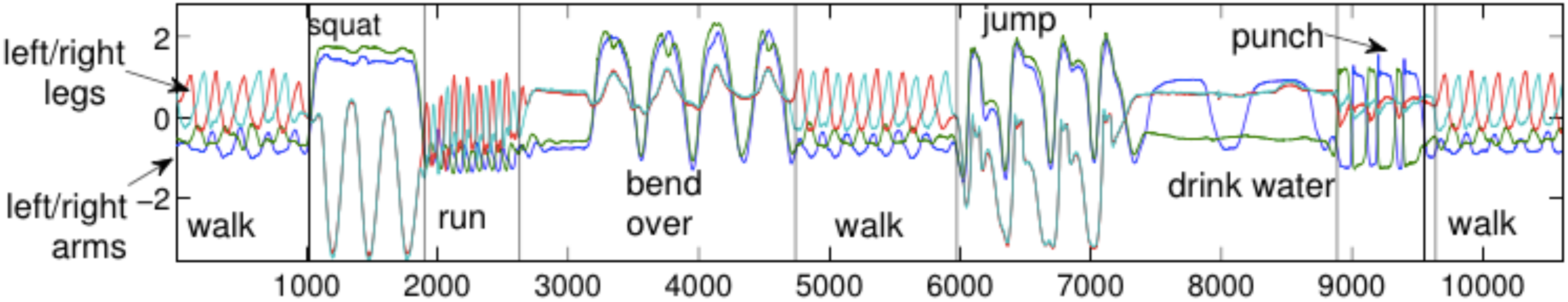
Goal: statistical monitoring of time-varying data streams

Background

Goal: statistical monitoring of time-varying data streams

Query (punch)

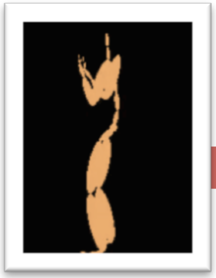
Data stream



Background

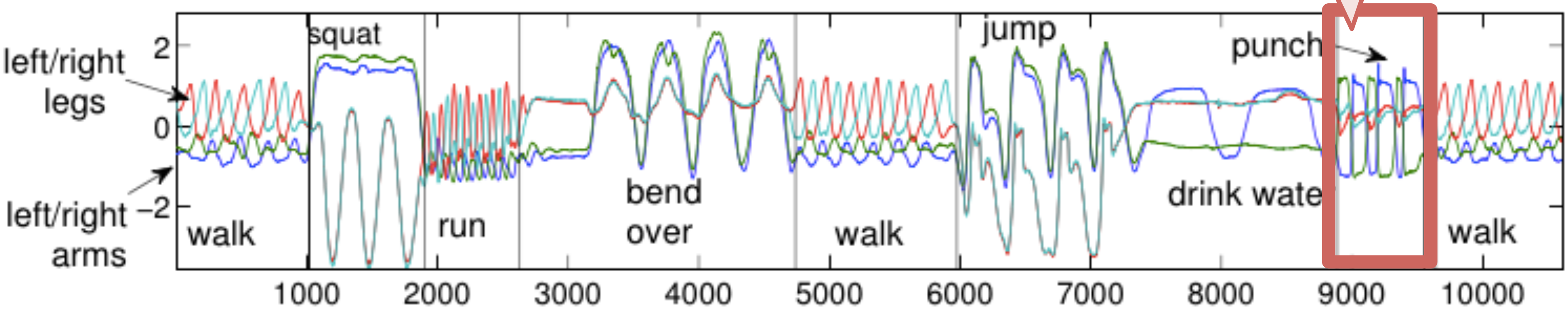
Goal: statistical monitoring of time-varying data streams

Query (punch)



Punch is here!

Data stream

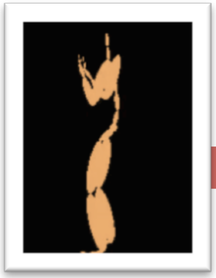


Background

Goal: statistical monitoring of

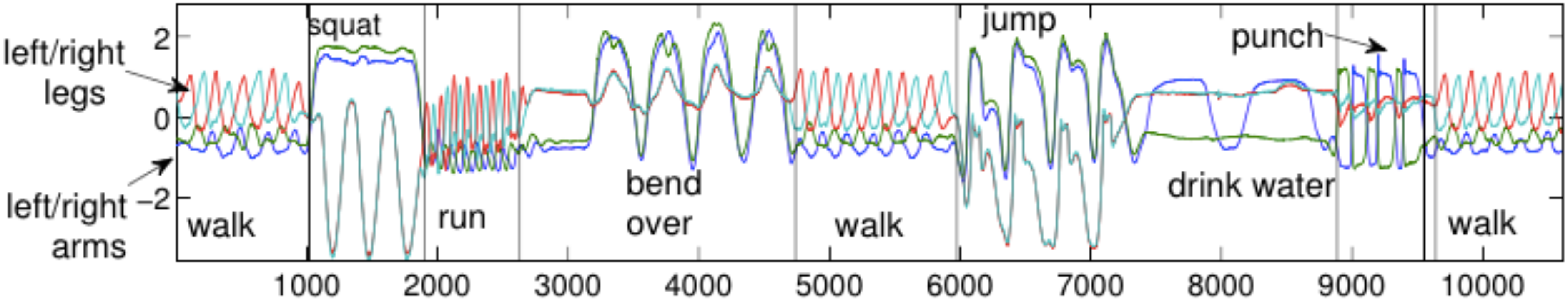
time-varying data streams

Query (punch)



Hidden Markov models (HMMs)

Data stream

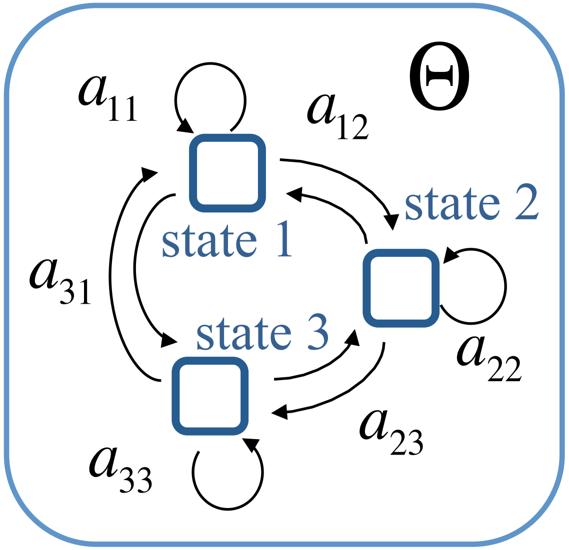


Background

Details

Hidden Markov models (HMMs)

$$\Theta = \{\pi, A, B\}$$



$k = 3$

Initial state probabilities $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^{k}$,

State transition probabilities $\mathbf{A} = \{a_{ij}\}_{i,j=1}^{k}$,

Output probabilities $\mathbf{B} = \{b_i(\boldsymbol{x})\}_{i=1}^{k}$

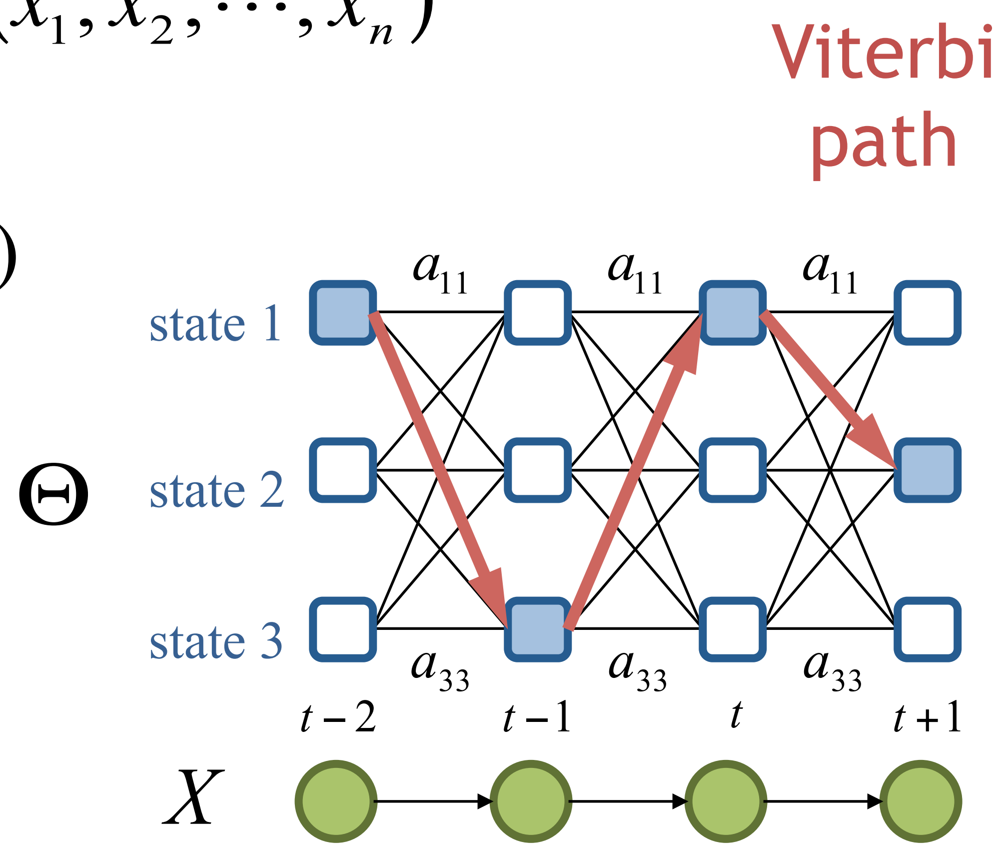
Background

Details

- Model: $\Theta = \{\pi, A, B\}$
- Sequence: $X = (x_1, x_2, \cdots, x_n)$

Likelihood: $P(X, \Theta)$

Trellis structure



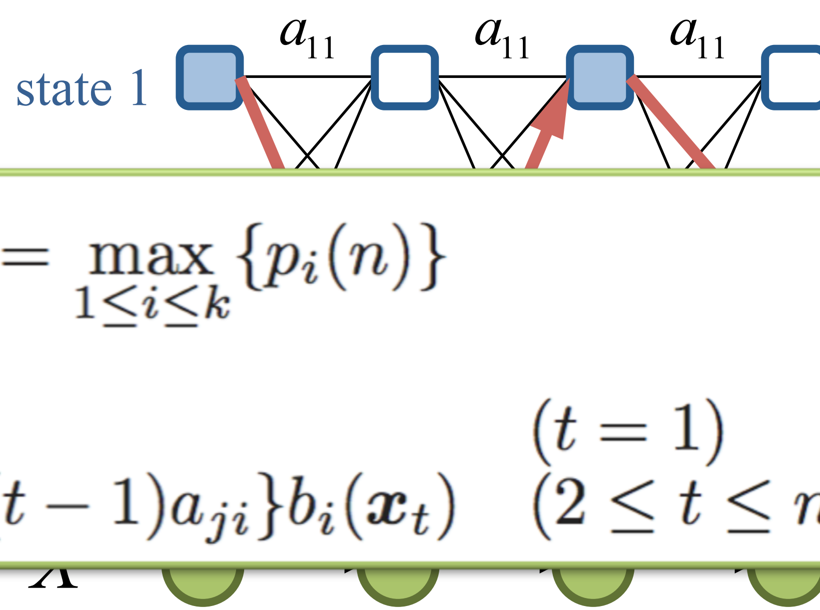
Background

Details

- Model: $\Theta = \{\pi, A, B\}$
- Sequence: $X = (x_1, x_2, \cdots, x_n)$

Viterbi path

Likelihood: $P(X, \Theta)$



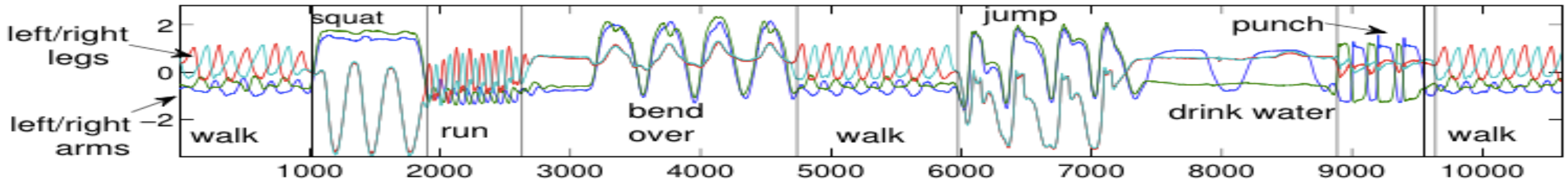
$$P(X, \boldsymbol{\Theta}) = \max_{1 \leq i \leq k} \{p_i(n)\}$$

$$p_i(t) = \begin{cases} \pi_i b_i(\boldsymbol{x}_1) & (t = 1) \\ \max_{1 \leq j \leq k}\{p_j(t-1)a_{ji}\} b_i(\boldsymbol{x}_t) & (2 \leq t \leq n) \end{cases}$$

Background

Given: a data stream X and model Θ

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

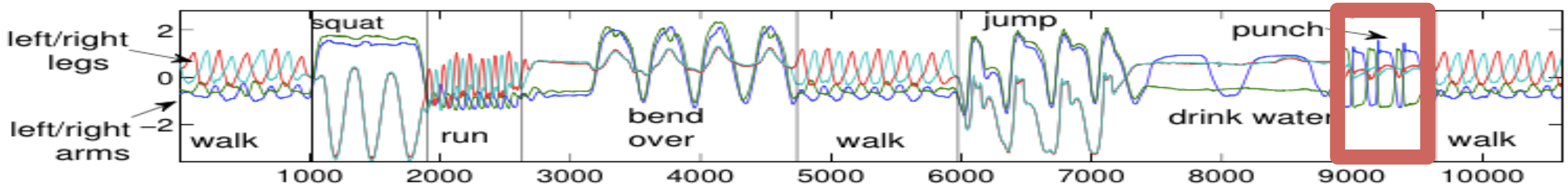


Θ_{punch}

Background

Given: a data stream X and model Θ

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



Θ_{punch}

$X[t_s : t_e]$

Find: subsequences that has a high likelihood value

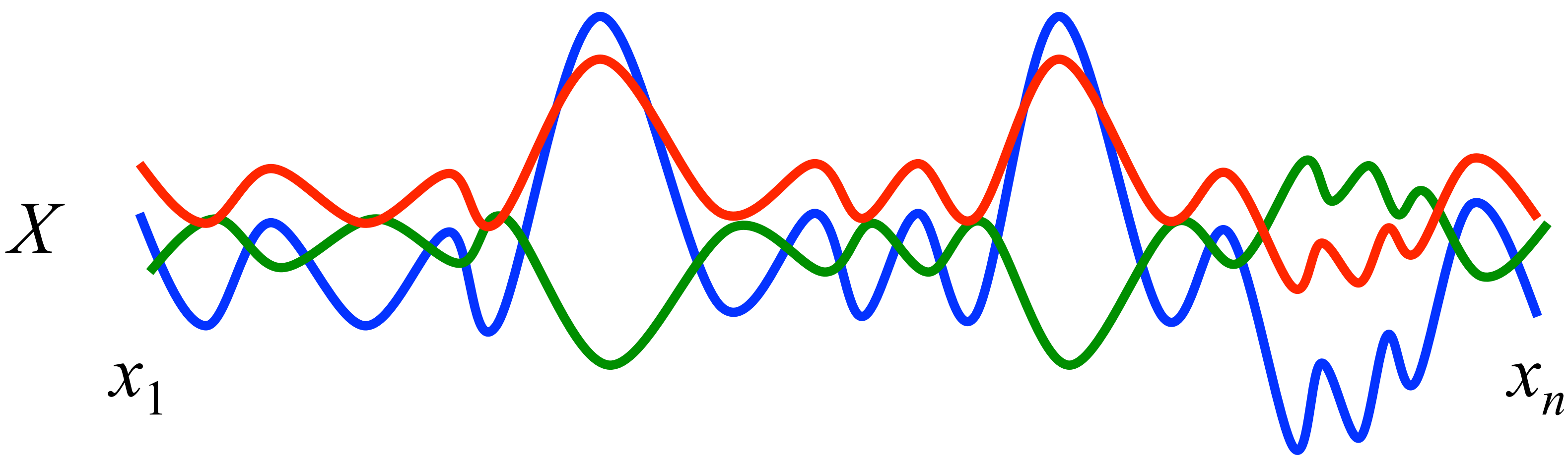
$$P(X[t_s : t_e], \Theta)$$

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

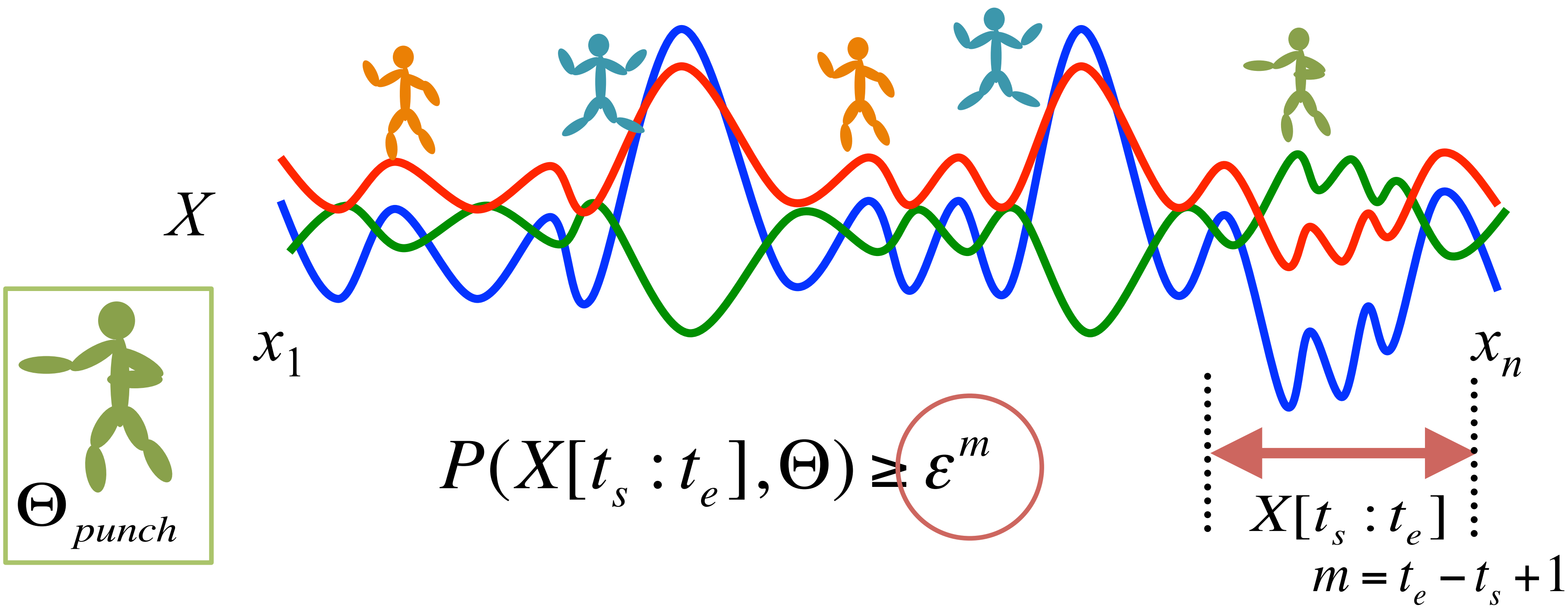
1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches

Requirements

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

Requirements

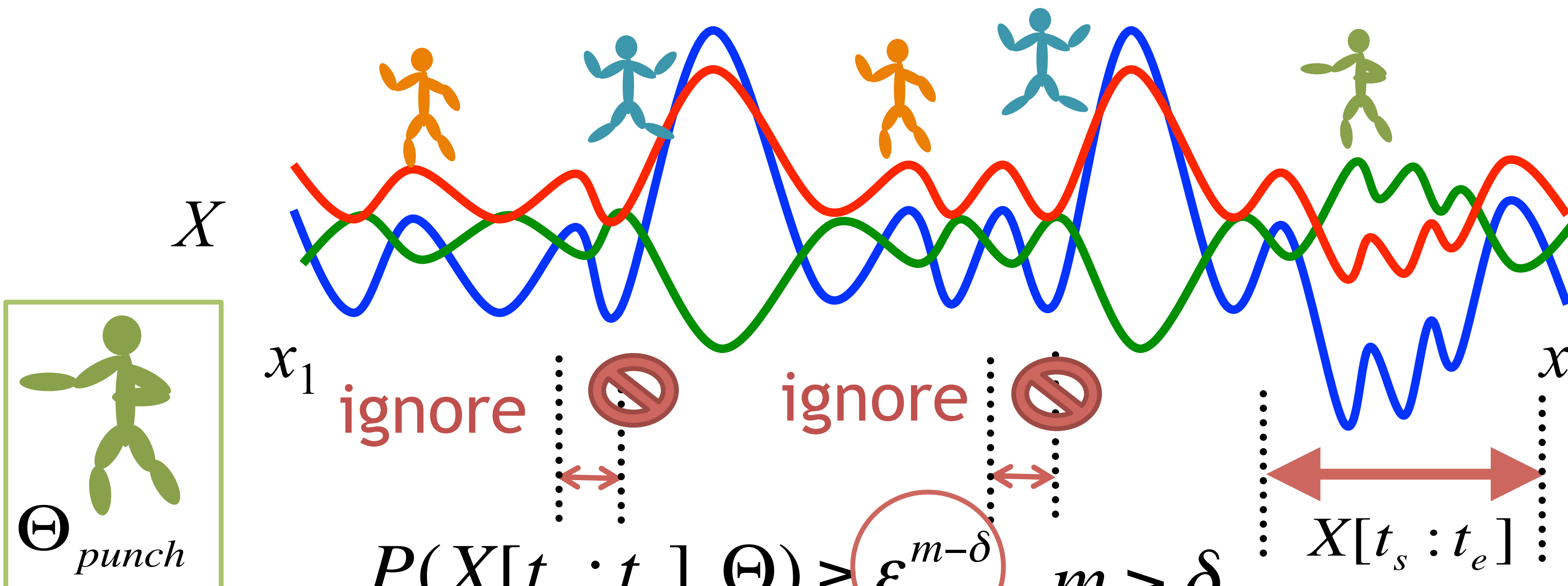
1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches



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

Requirements

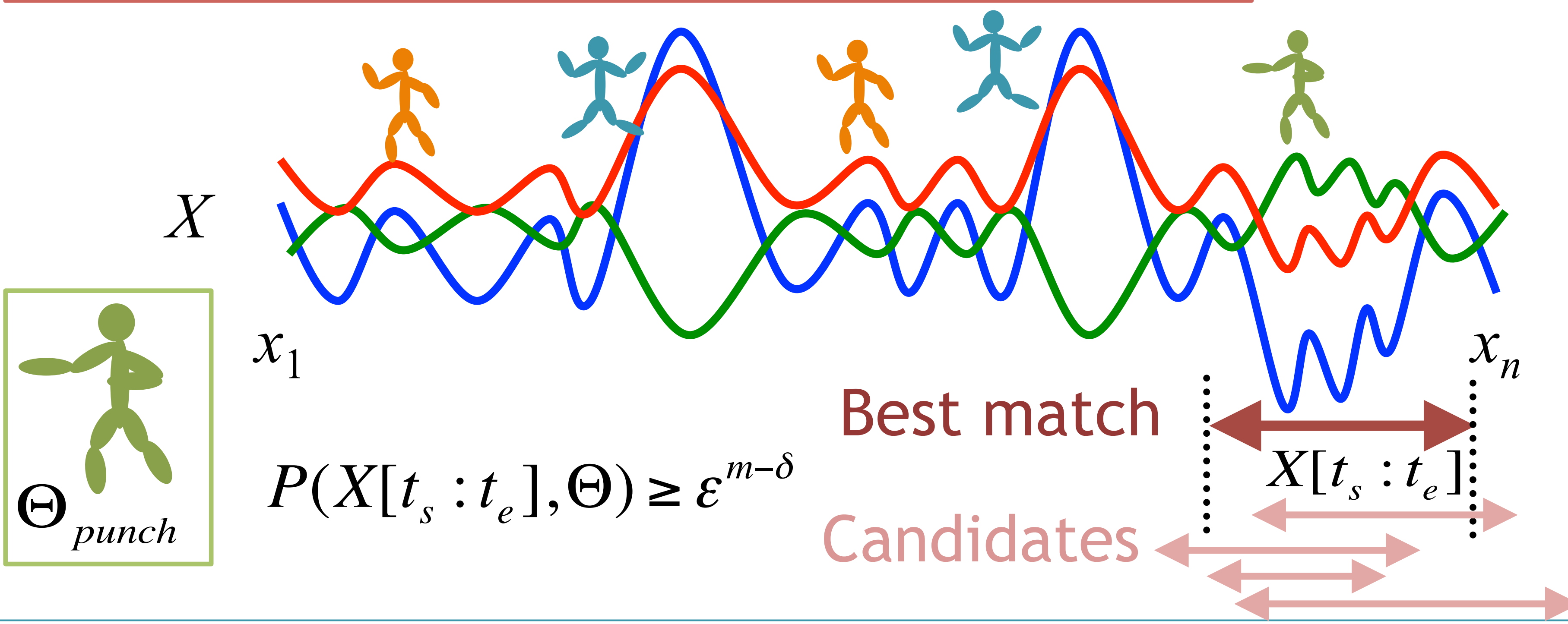
1. Exponential threshold function
2. Minimum length of subsequences
3. Non-overlapping matches



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

Requirements

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

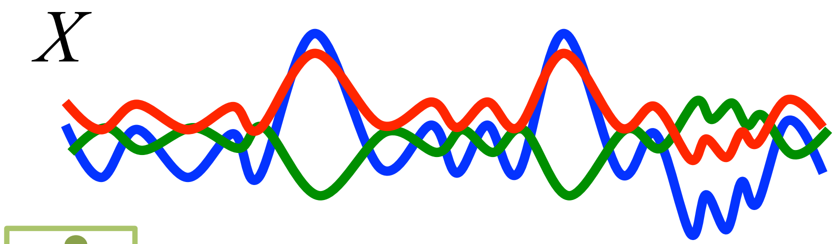


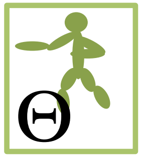
$$P(X[t_s : t_e], \Theta) \geq \varepsilon^{m-\delta}$$

Problem definition

Given:

- data stream X
- Model Θ
- thresholds ε, δ

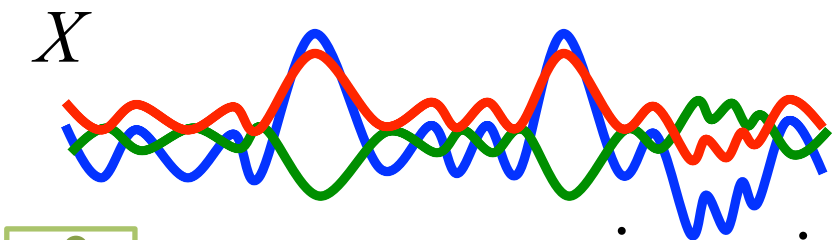


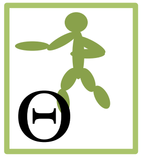


Problem definition

Given:

- data stream X
- Model Θ
- thresholds ε, δ



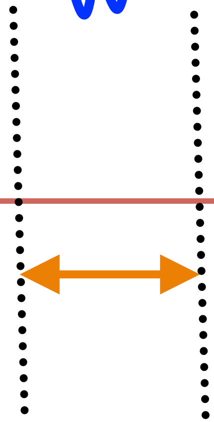


Report: all subsequences $X[t_s : t_e]$

that satisfy:

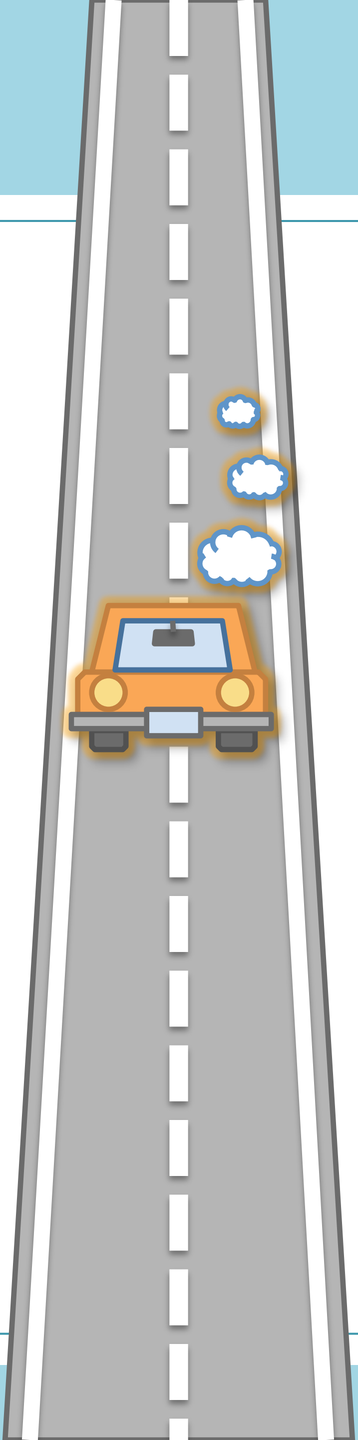
1. $P(X[t_s : t_e], \Theta) \geq \varepsilon^{m-\delta}$
2. only the local maximum,

   among several overlapping matches



Outline

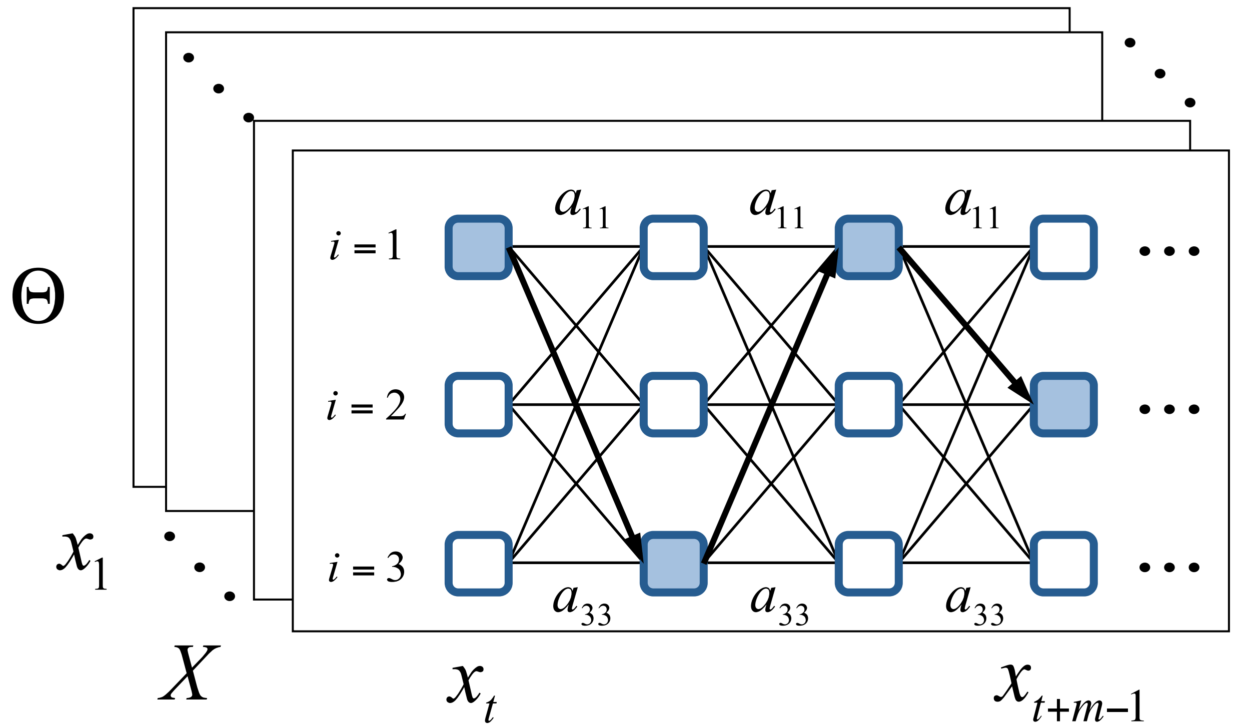
- Motivation
- Problem formulation
- **Main ideas**
- Experiments
- StreamScan at work
- Conclusions



Previous solution

Sliding model method (SMM), by Wilpon et al.

Compute likelihood starting from every time-tick

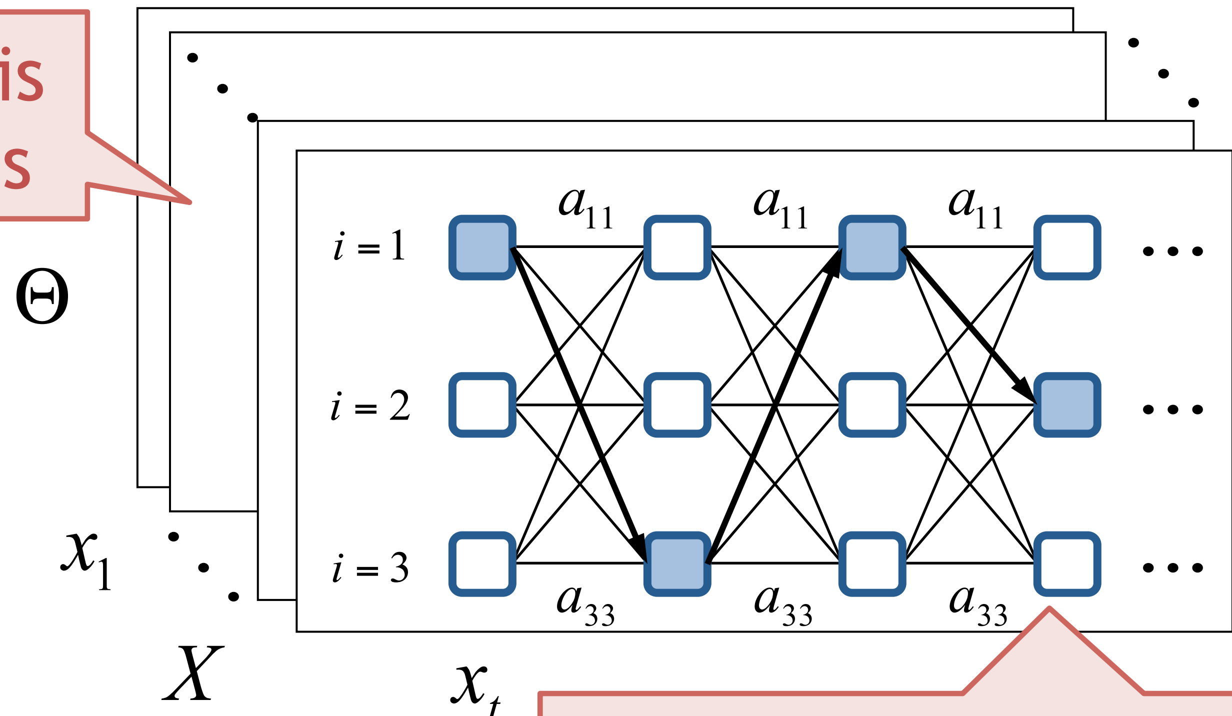


Previous solution

Sliding model method (SMM), by Wilpon et al.

Compute likelihood starting from every time-tick

O(n) trellis structures



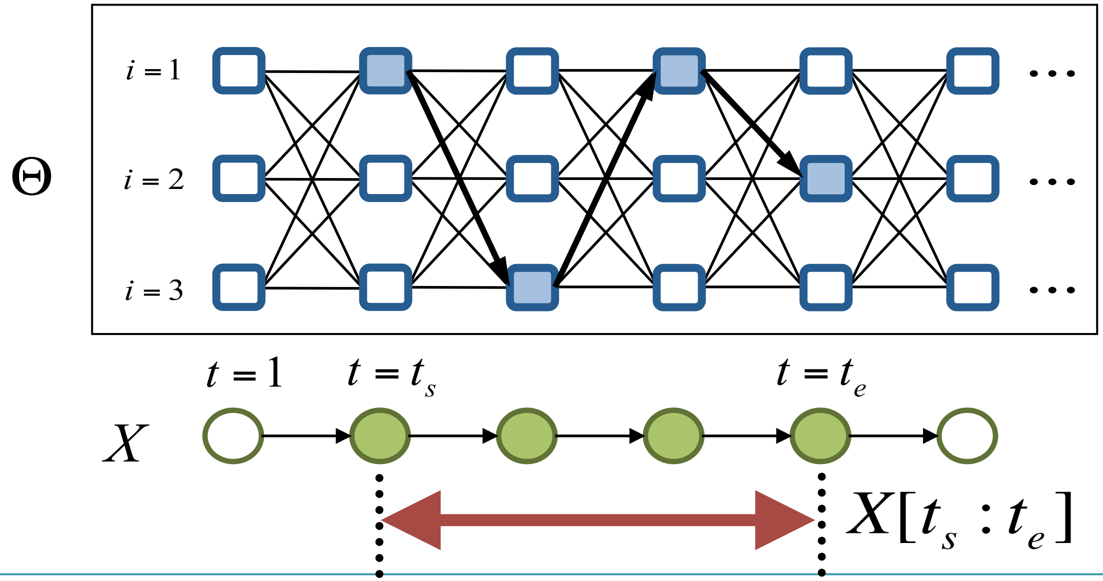
Update $O(nk^2)$ for each time-tick

Main idea: StreamScan

(a) Cumulative likelihood function

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

(b) Subsequence trellis structure (STS)



Main idea (a)

Details

(a) Cumulative likelihood function

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

$$v_i(t) = \max \begin{cases} \pi_i b_i(x_t) \cdot \epsilon^{-1} \\ \max_{1 \le j \le k} \{v_j(t-1) a_{ji}\} b_i(x_t) \cdot \epsilon^{-1} \end{cases}$$

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

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

Main idea (a)

Details

(a) Cumulative likelihood function

$v_i(t)$ $\quad x_t) \cdot \epsilon^{-1}$

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

It requires a single structure

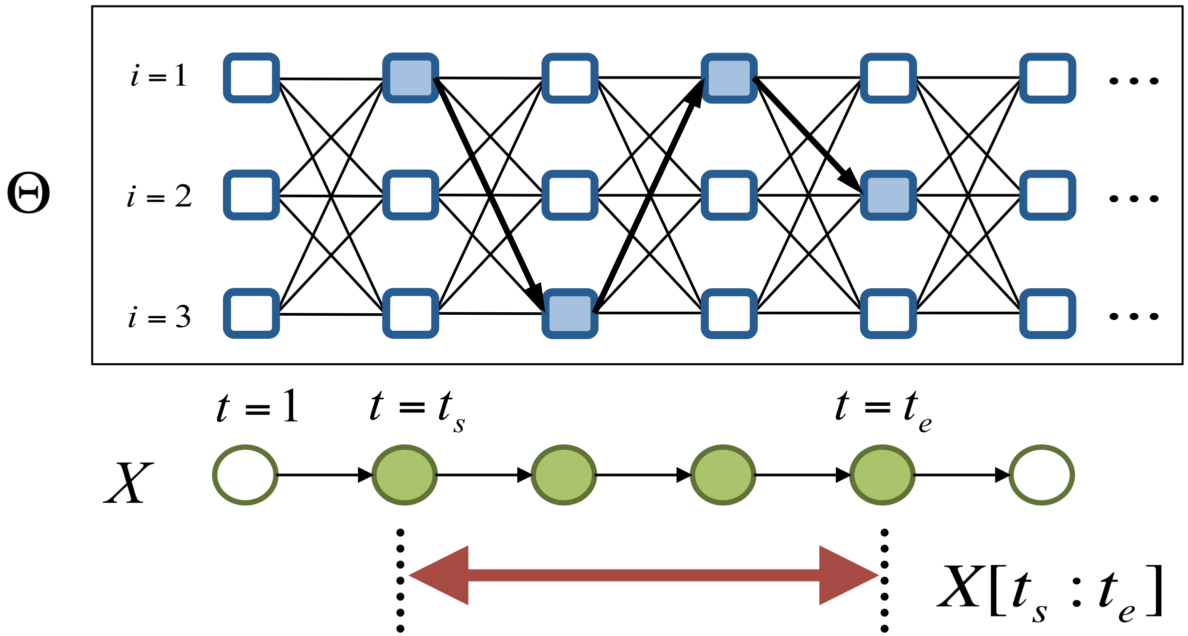
It guarantees the best matches

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

Main idea (b)

Details

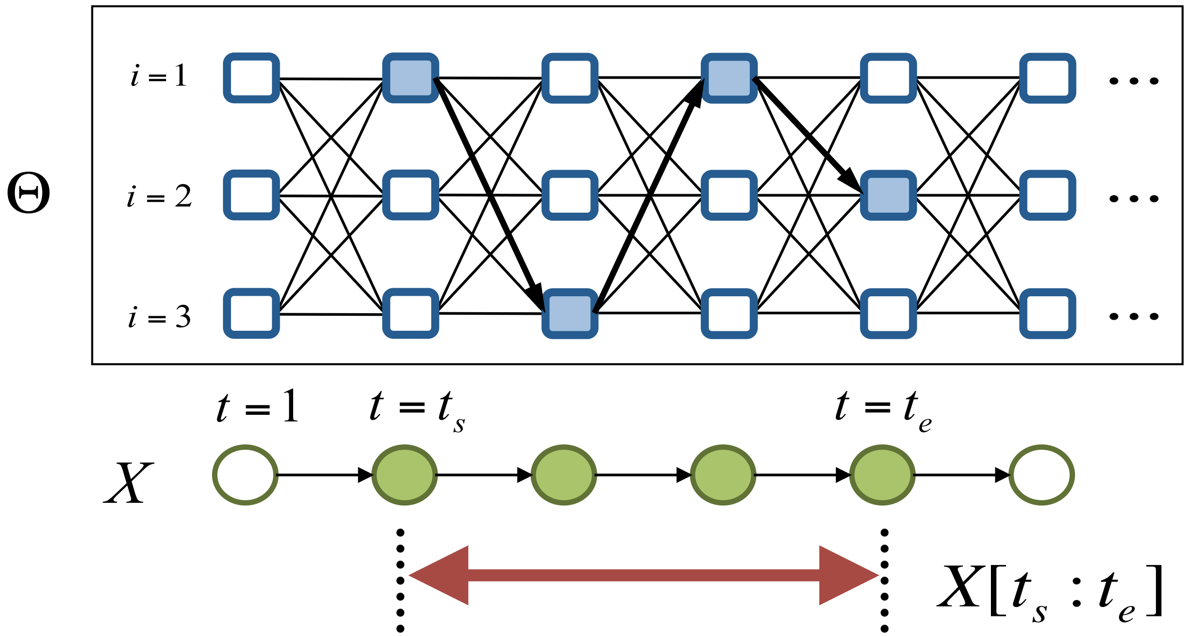
(b) Subsequence trellis structure (STS)



Main idea (b)

Details

(b) Subsequence trellis structure (STS)



For each cell: □

$s_i(t)$: Starting position

$v_i(t)$: Cum. likelihood

Algorithm

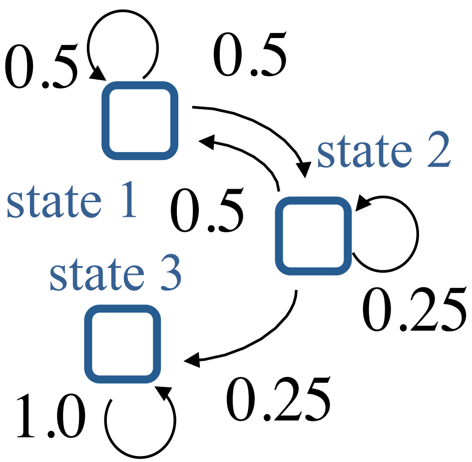
Details

Example,

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

$$\varepsilon = 0.1, \delta = 3$$

$$\boldsymbol{\Theta} = \left\{ \pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.25 & 0.25 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0.75 & 0.25 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right\}$$



Algorithm

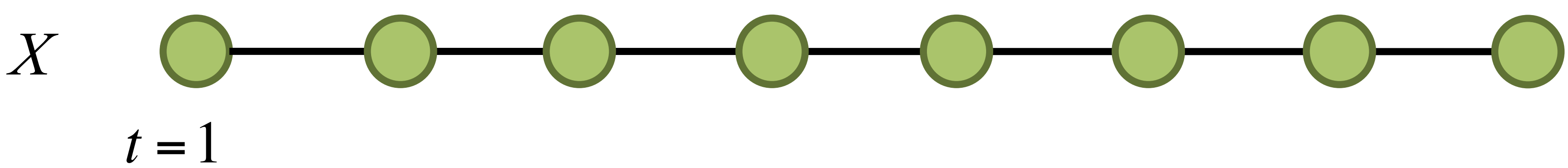
Details

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

$\varepsilon = 0.1, \delta = 3$

(STS, k=3)

State 1	0 (1)							
State 2	0 (1)							
State 3	0 (1)							
	x_1=3	x_2=1	x_3=1	x_4=2	x_5=3	x_6=3	x_7=3	x_8=1



Algorithm

Details

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

$\varepsilon = 0.1, \delta = 3$

(STS, k=3)

$v_i(t)$: Cum. Likelihood

$s_i(t)$: Starting position

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

$t = 1$

Algorithm

Details

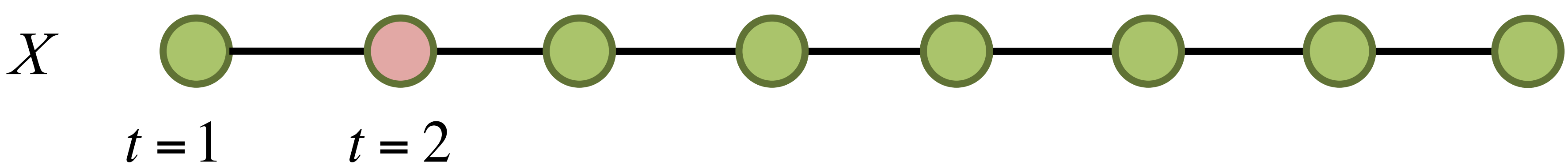
Example, $X = (3, 1, 1, 2, 3, 3, 3, 1).$

$\varepsilon = 0.1, \delta = 3$

(STS, k=3)

State 1	0 (1)	10 (2)						
State 2	0 (1)	0 (2)						
State 3	0 (1)	0 (2)						
	x_1=3	x_2=1	x_3=1	x_4=2	x_5=3	x_6=3	x_7=3	x_8=1

candidate!



Algorithm

Details

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

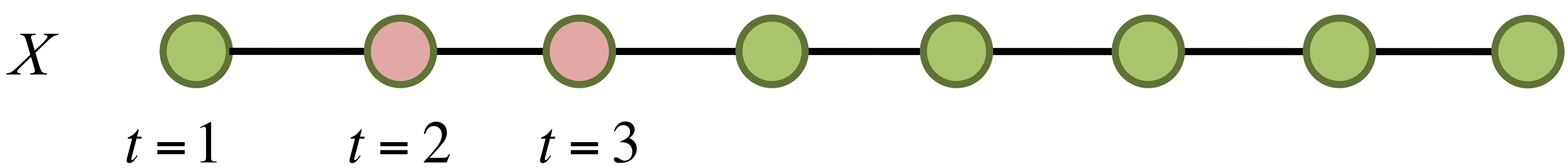
$\varepsilon = 0.1, \delta = 3$

(STS, k=3)

State 1	0 (1)	10 (2)	$*^1$ 50 (2)					
State 2	0 (1)	0 (2)	$*^2$ 37.5 (2)					
State 3	0 (1)	0 (2)	0 (3)					
	x_1=3	x_2=1	x_3=1	x_4=2	x_5=3	x_6=3	x_7=3	x_8=1

$*^1 \times 0.5 \times 1.0 \times 10$

$*^2 \times 0.5 \times 0.75 \times 10$



Algorithm

Details

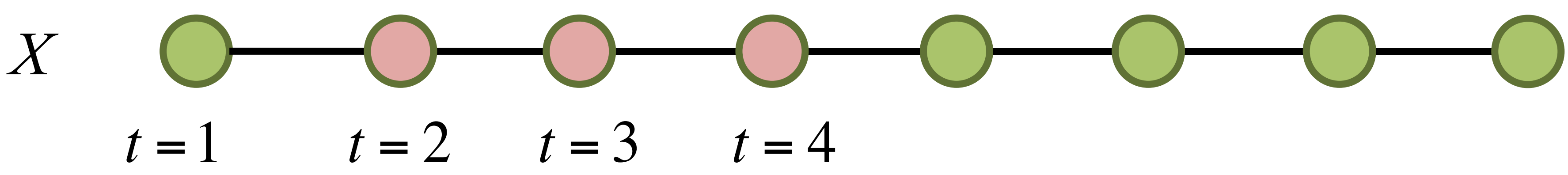
Example, $X = (3, 1, 1, 2, 3, 3, 3, 1).$

$\varepsilon = 0.1, \delta = 3$

(STS, k=3)

State 1	0 (1)	10 (2)	50 (2)	0 (4)				
State 2	0 (1)	0 (2)	37.5 (2)	62.5 $*^3$ (2)				
State 3	0 (1)	0 (2)	0 (3)	0 (4)				
	x_1=3	x_2=1	x_3=1	x_4=2	x_5=3	x_6=3	x_7=3	x_8=1

$*^3 \times 0.5 \times 0.25 \times 10$



Algorithm

Details

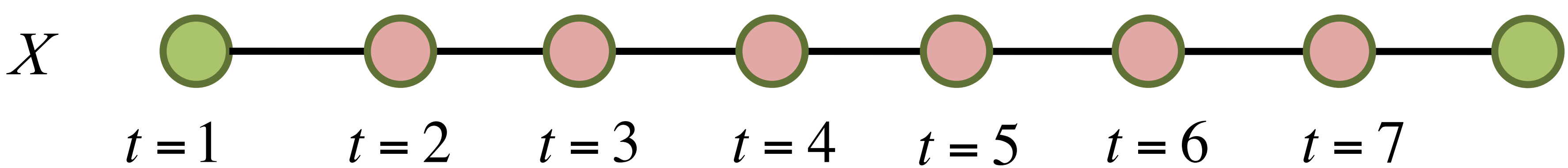
Example,

(STS, k=3)

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

$$\varepsilon = 0.1, \delta = 3$$

State 1	0 (1)	10 (2)	50 (2)	0 (4)	0 (5)	0 (6)	0 (7)	
State 2	0 (1)	0 (2)	37.5 (2)	62.5 (2)	0 (5)	0 (6)	0 (7)	
State 3	0 (1)	0 (2)	0 (3)	0 (4)	156.25 (2)	1562.5 (2)	15625 (2)	
	$x_1=3$	$x_2=1$	$x_3=1$	$x_4=2$	$x_5=3$	$x_6=3$	$x_7=3$	$x_8=1$



Algorithm

Details

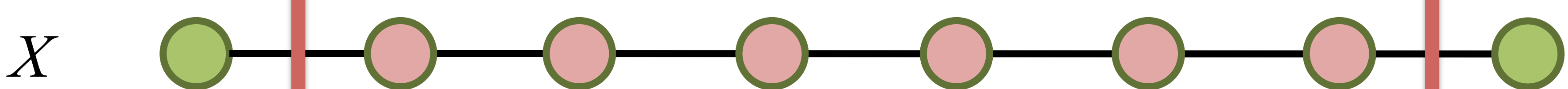
Example, $X = (3, 1, 1, 2, 3, 3, 3, 1).$

$\varepsilon = 0.1, \delta = 3$

(STS, k=3)

Report X[2:7]

State 1	0 (1)	10 (2)	50 (2)	0 (4)	0 (5)	0 (6)	0 (7)	10 (8)
State 2	0 (1)	0 (2)	37.5 (2)	62.5 (2)	0 (5)	0 (6)	0 (7)	0 (8)
State 3	0 (1)	0 (2)	0 (3)	0 (4)	156.25 (2)	1562.5 (2)	15625 (2)	0 (8)
	x_1=3	x_2=1	x_3=1	x_4=2	x_5=3	x_6=3	x_7=3	x_8=1



$t=1$ $t=2$ $t=3$ $t=4$ $t=5$ $t=6$ $t=7$ $t=8$

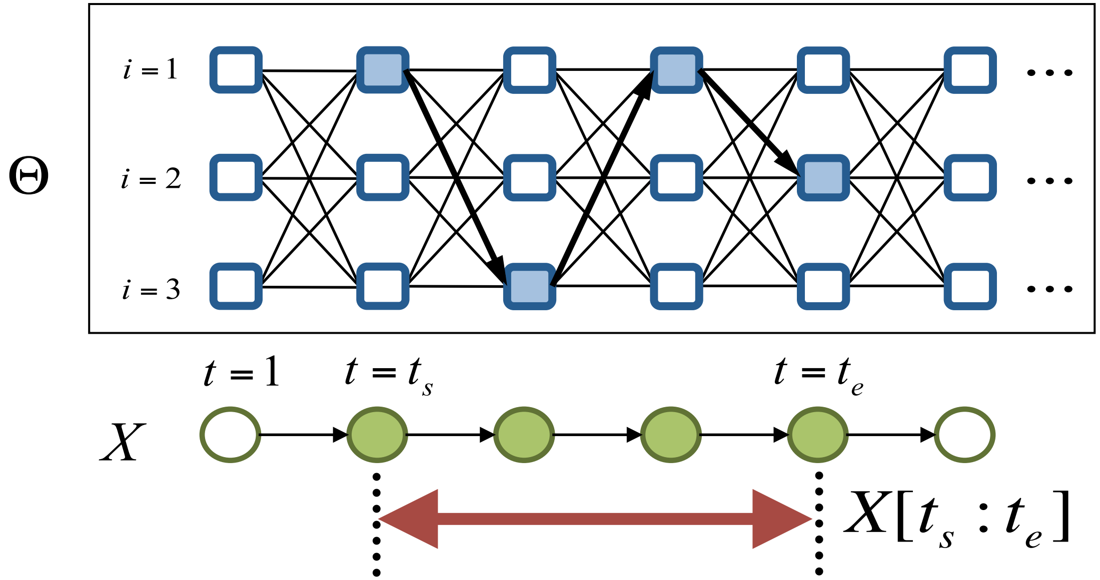
Theoretical analysis

Details

StreamScan guarantees

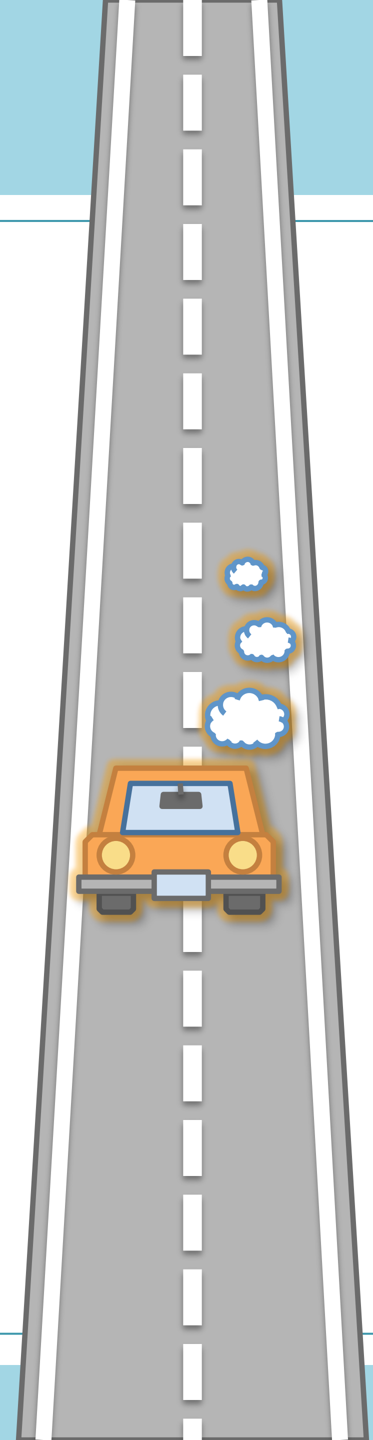
- no false dismissals
- O(1) space and time per time-tick

(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)

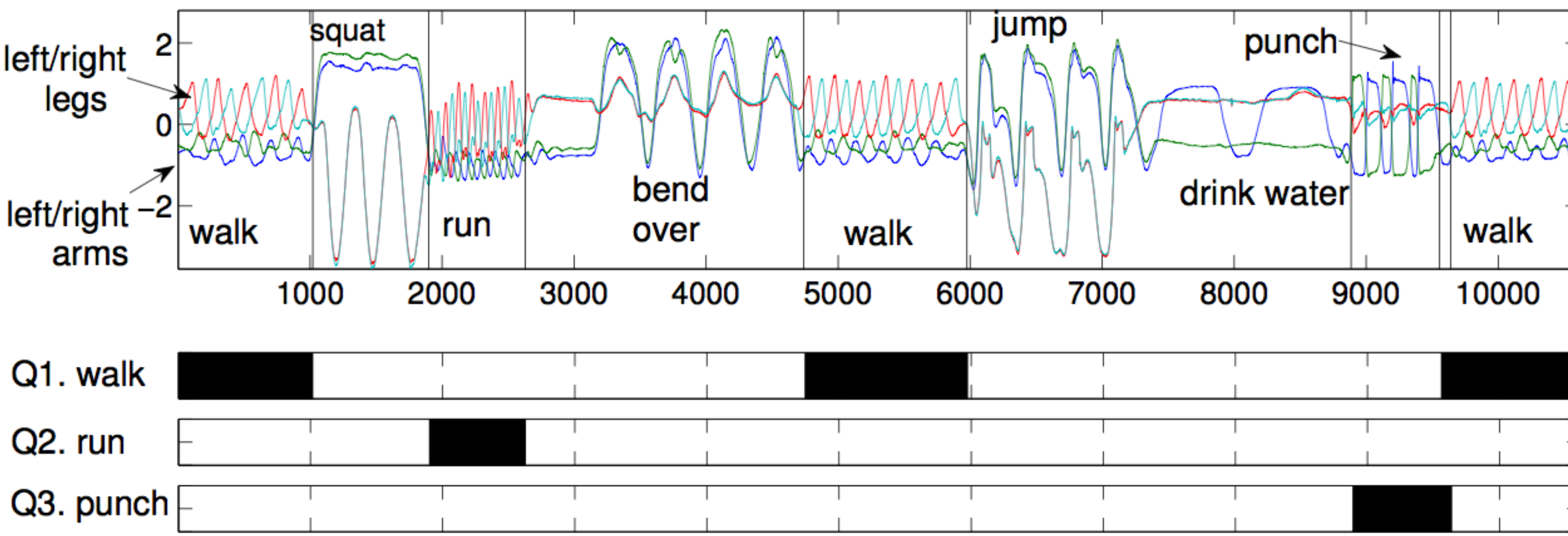
(Query)

(a-1) Query #1 : walking

(a-2) Query #2 : running

(a-3) Query #3 : punching

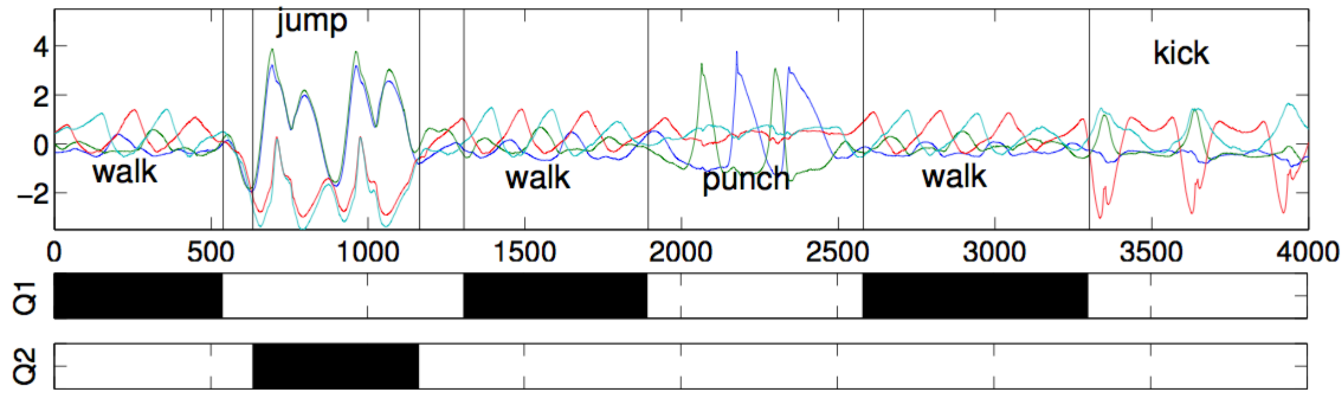
(Output)



Q1. Effectiveness

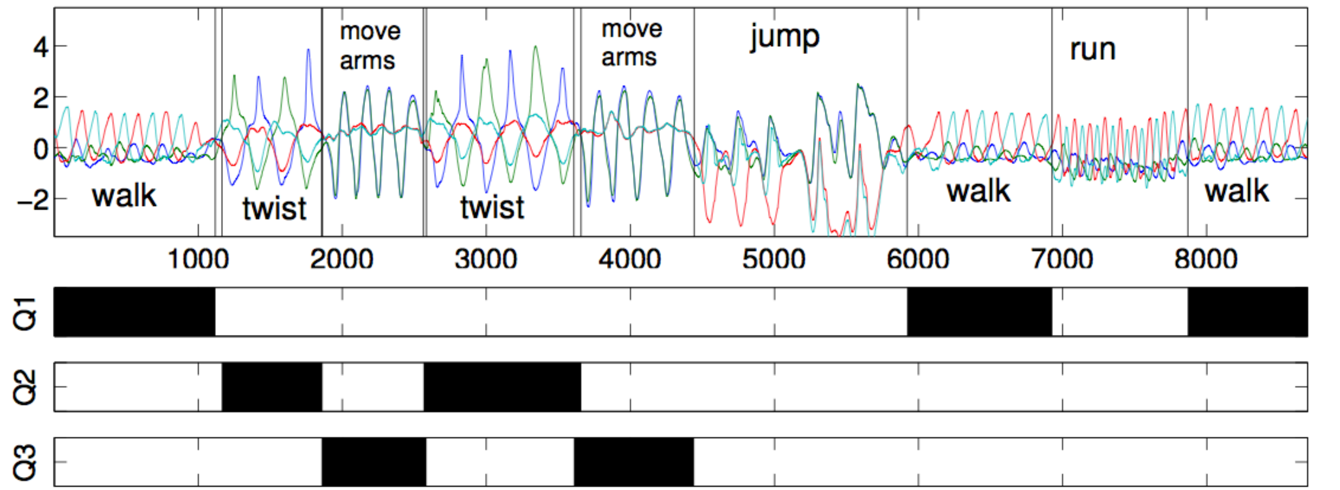
MoCap

Q1. Walk
Q2. Jump



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

Q1. Walk
Q2. Jump
Q3. Moving arms

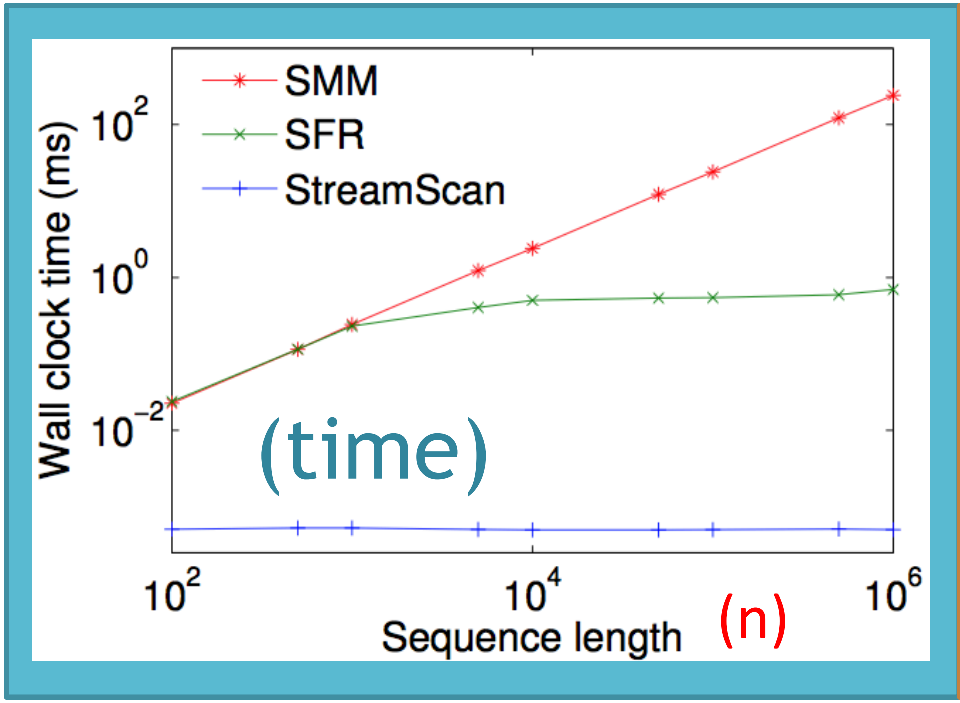


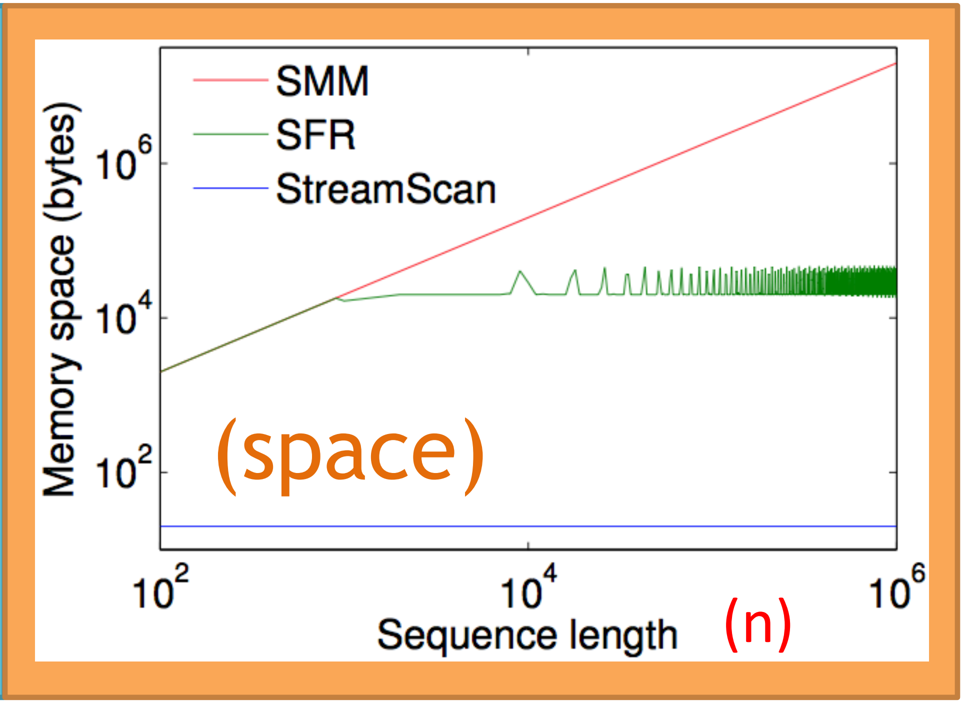
(b) *MoCap* #3 (Query #1: walking, Query #2: twisting, Query #3: moving arms)

Q2. Scalability

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

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

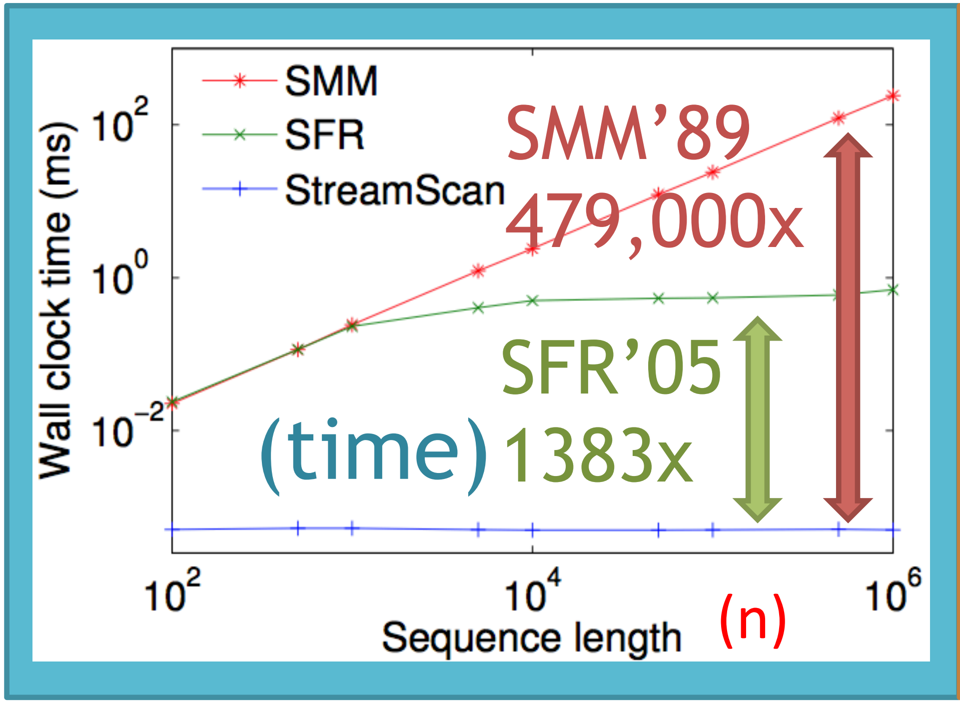


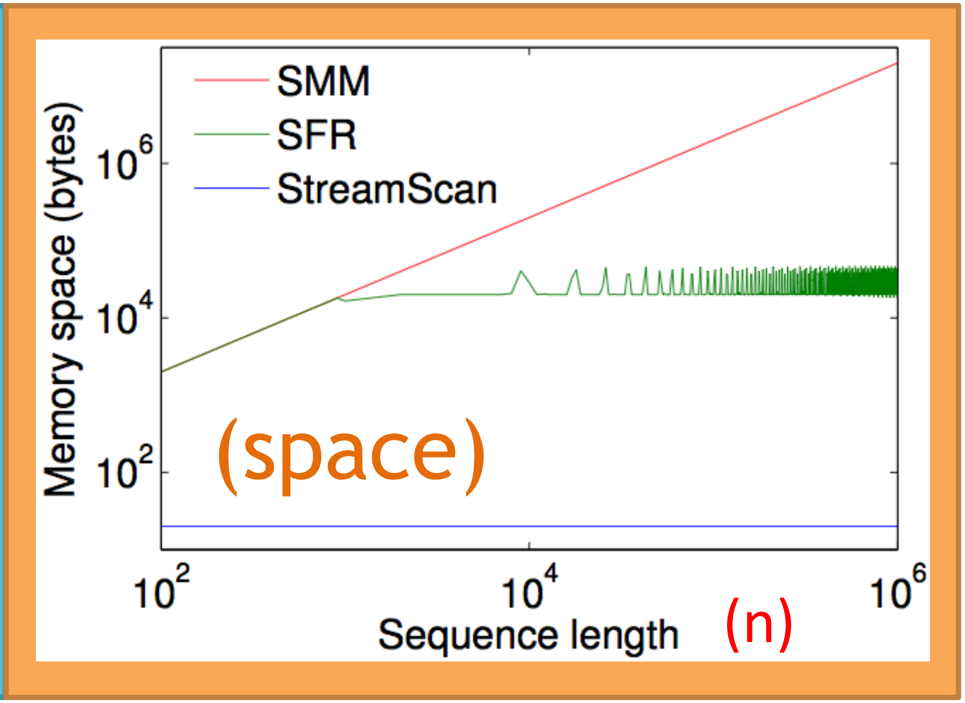


Q2. Scalability

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

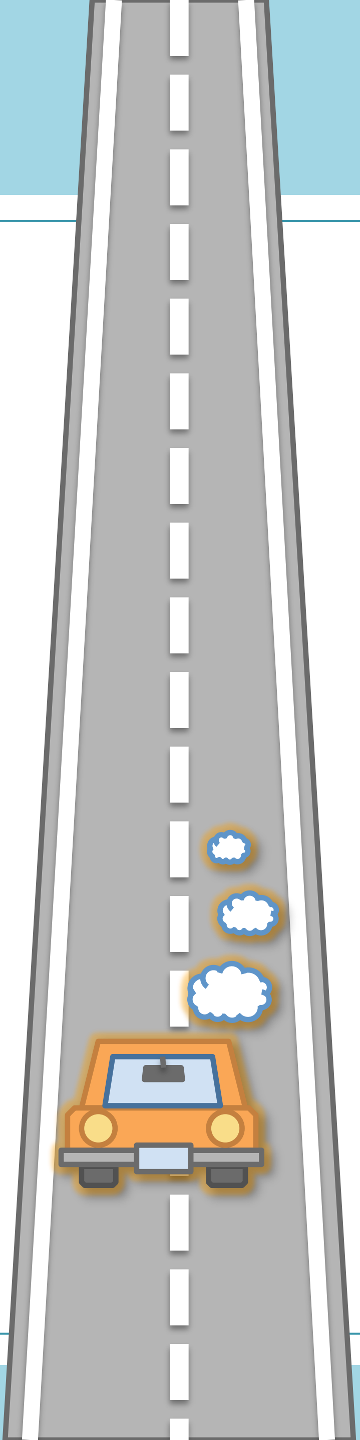
StreamScan requires
constant time/space, i.e., O(1)





Outline

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



StreamScan at work

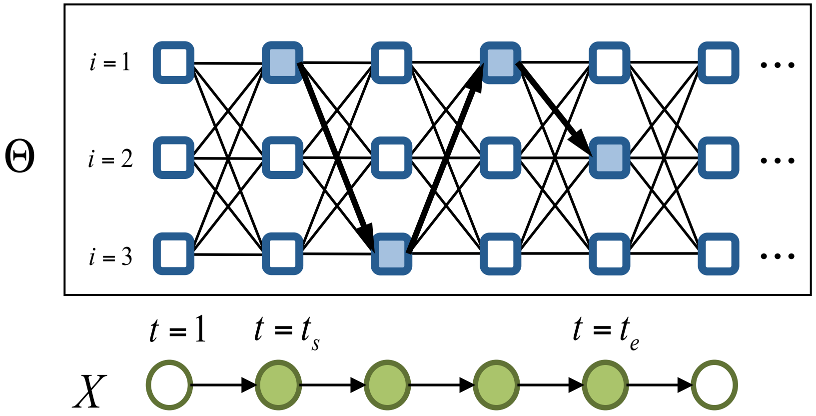
StreamScan is capable of various applications, e.g.,

App1. Social activity monitoring

- *Web-click* streams

App2. Extreme detection in keyword stream

- *GoogleTrend* streams



StreamScan at work

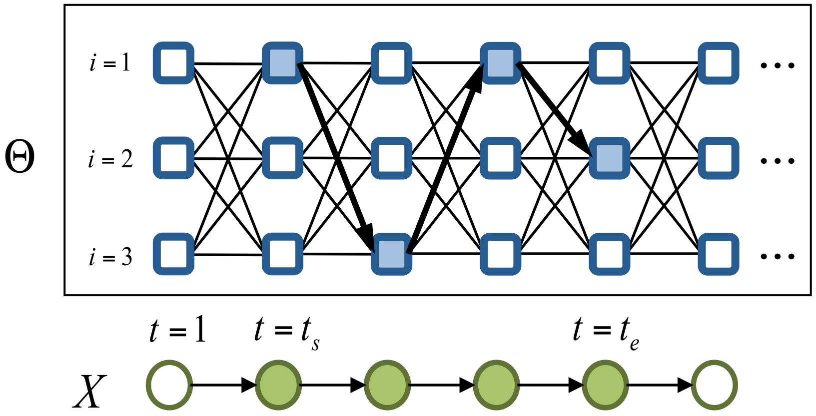
StreamScan is capable of various applications, e.g.,

App1. Social activity monitoring

- *Web-click* streams

App2. Extreme detection in keyword stream

- *GoogleTrend* streams

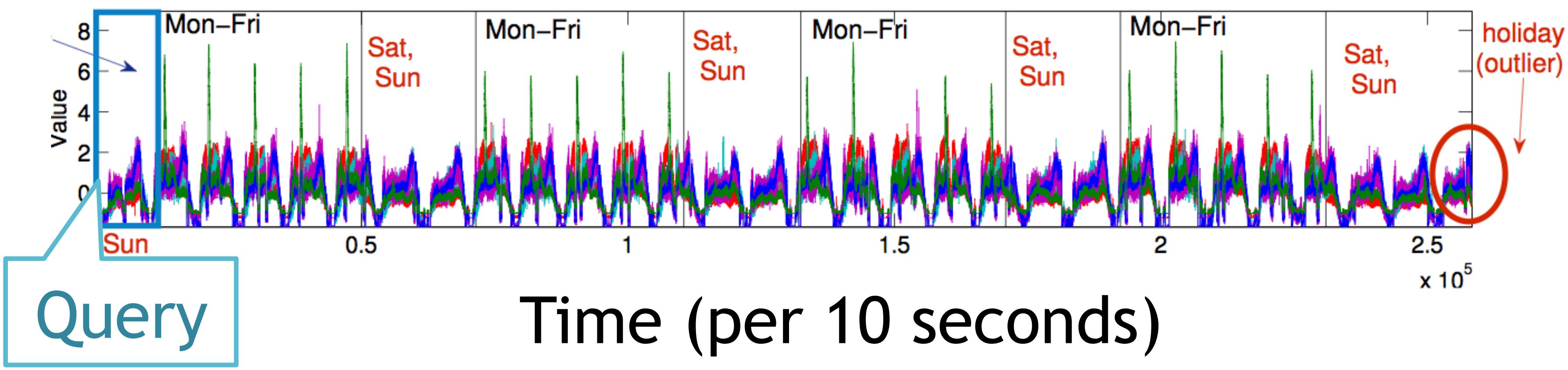


Application (1)

Social activity monitoring

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

Query: first Sunday



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

Application (1)

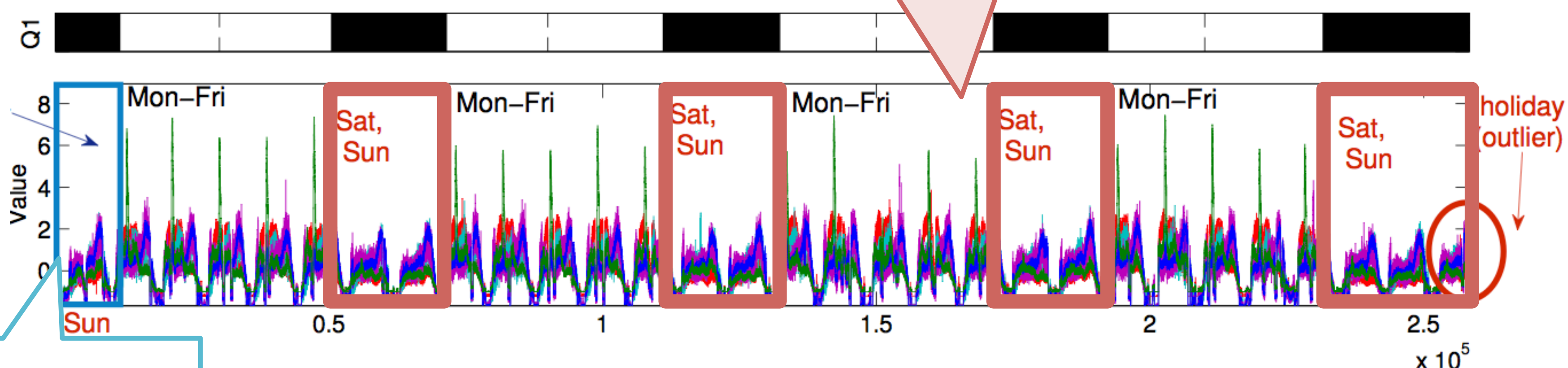
Social ac … oring

Detected!

Given: Web-click seq … s)

StreamScan identifies all weekends + holiday

Query: first Sunday



Query

Time (per 10 seconds)

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

StreamScan at work

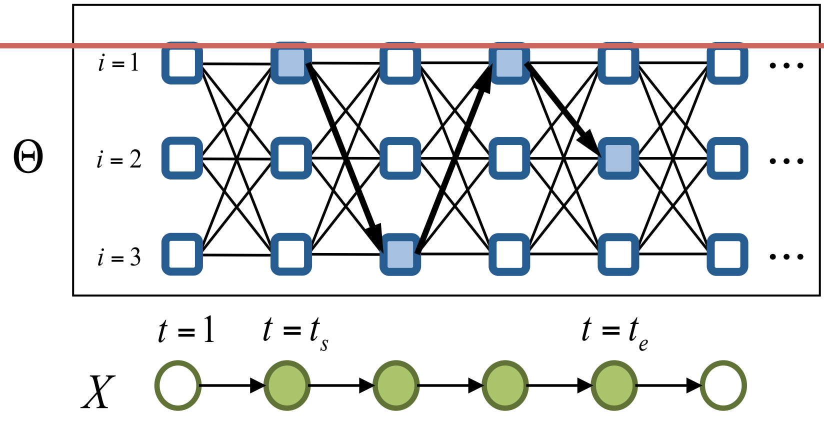
StreamScan is capable of various applications, e.g.,

App1. Social activity monitoring

- *Web-click* streams

App2. Extreme detection in keyword stream

- *GoogleTrend* streams



Application (2)

"Extreme monitoring" in keyword stream

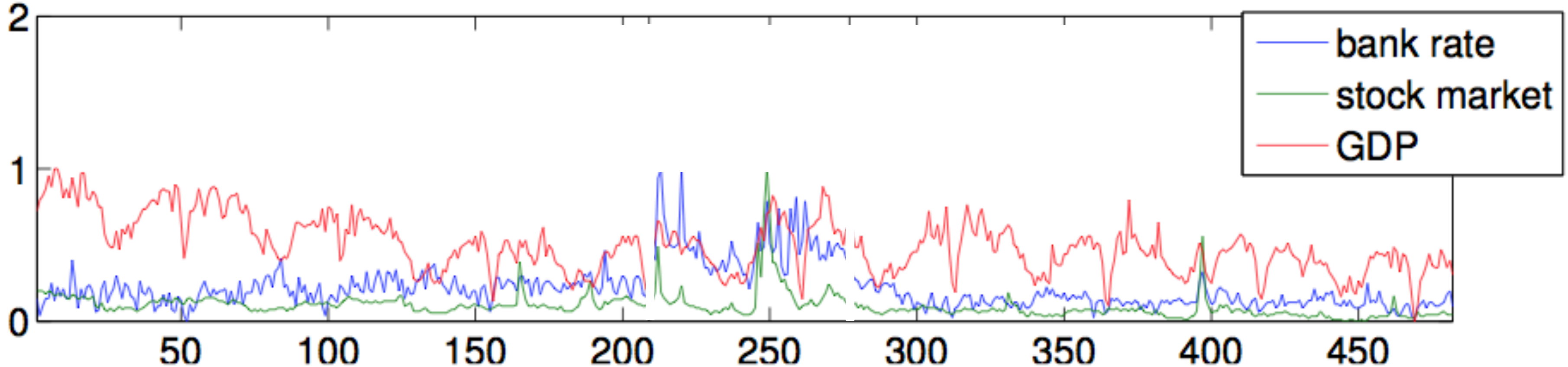
Given: GoogleTrend streams (10 years, weekly)

Application (2)

"Extreme monitoring" in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Finance-related keywords



Time (weekly, 10 years)

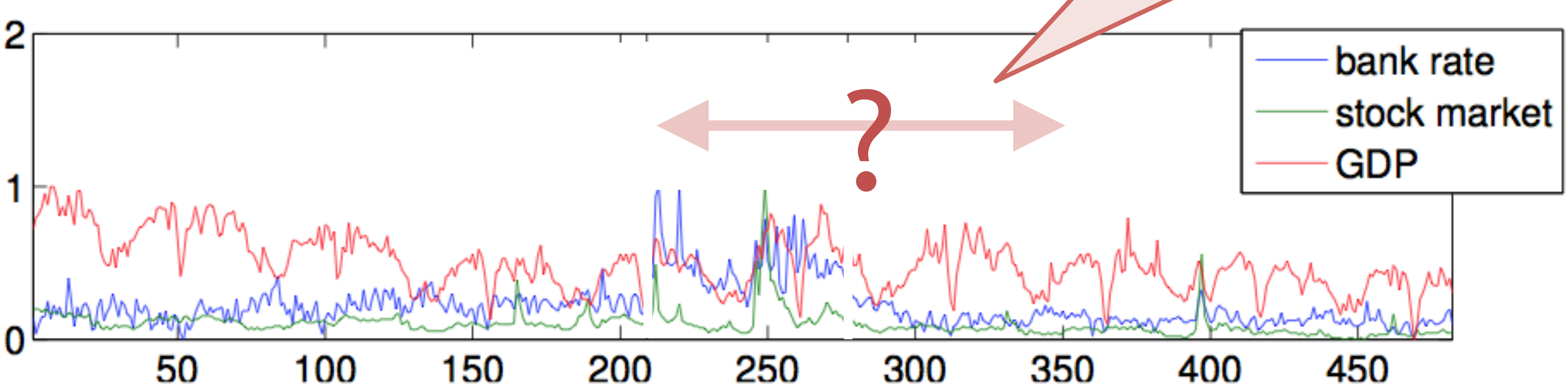
Application (2)

"Extreme monitoring" in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Finance-related keywords

Any extreme behavior?



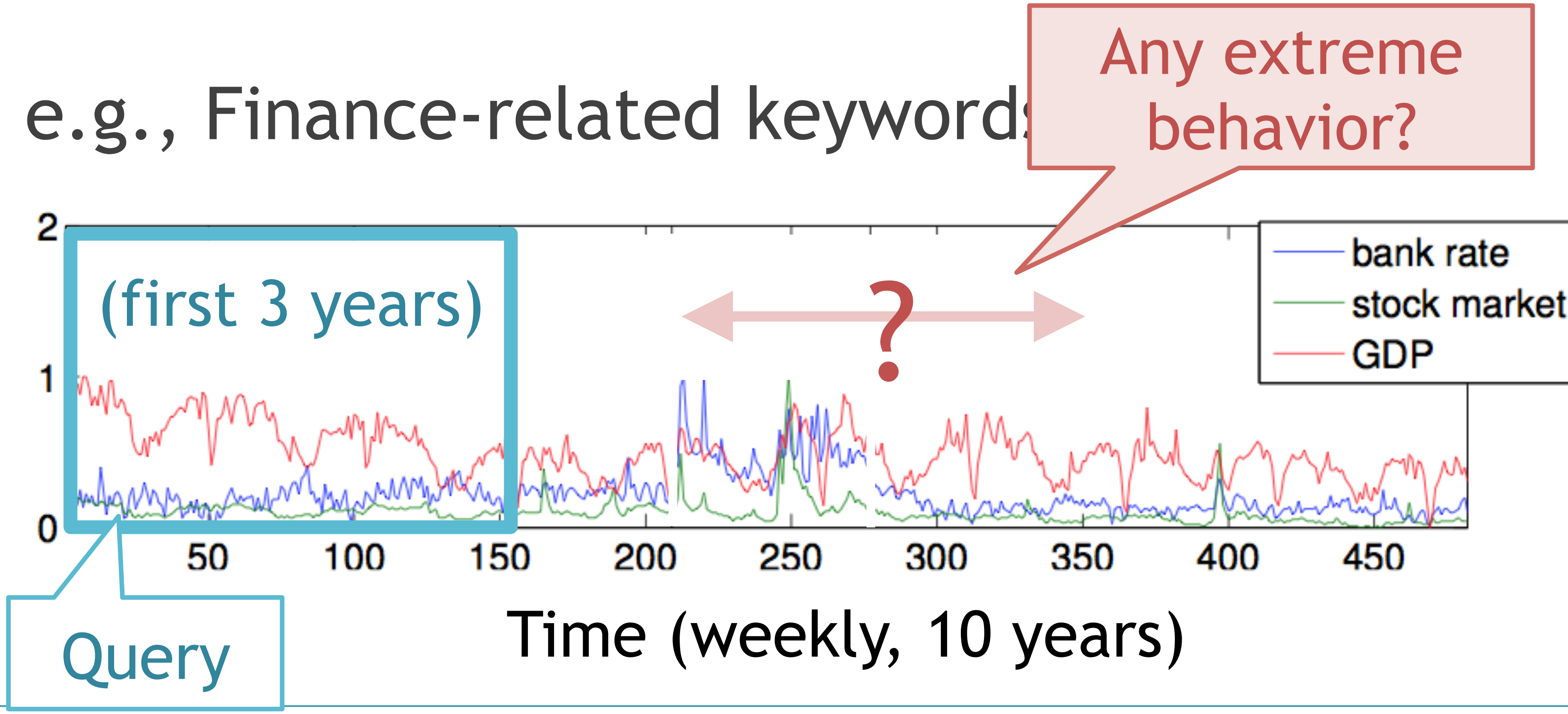
Time (weekly, 10 years)

Application (2)

"Extreme monitoring" in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Finance-related keywords



Application (2)

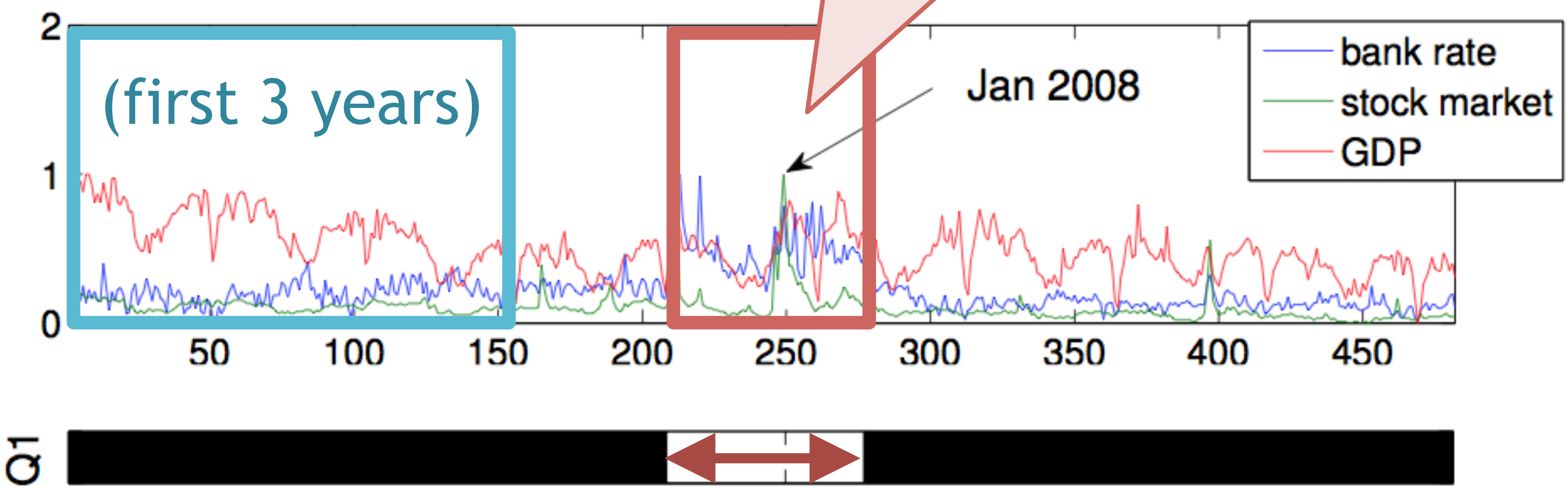
"Extreme monitoring" in keyword stream

Given: G streams (10 years, weekly)

e.g., Finance-related

Detected!

Global financial crisis in 2008!

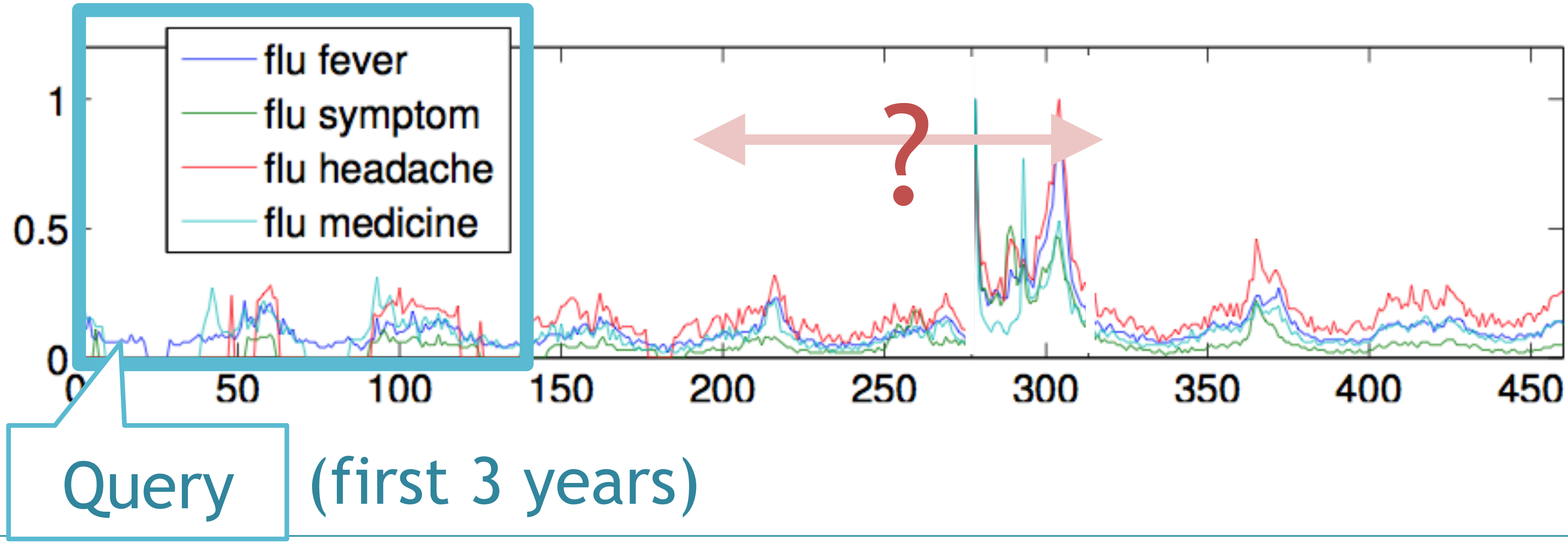


Application (2)

"Extreme monitoring" in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Flu-related keywords



(first 3 years)

Application (2)

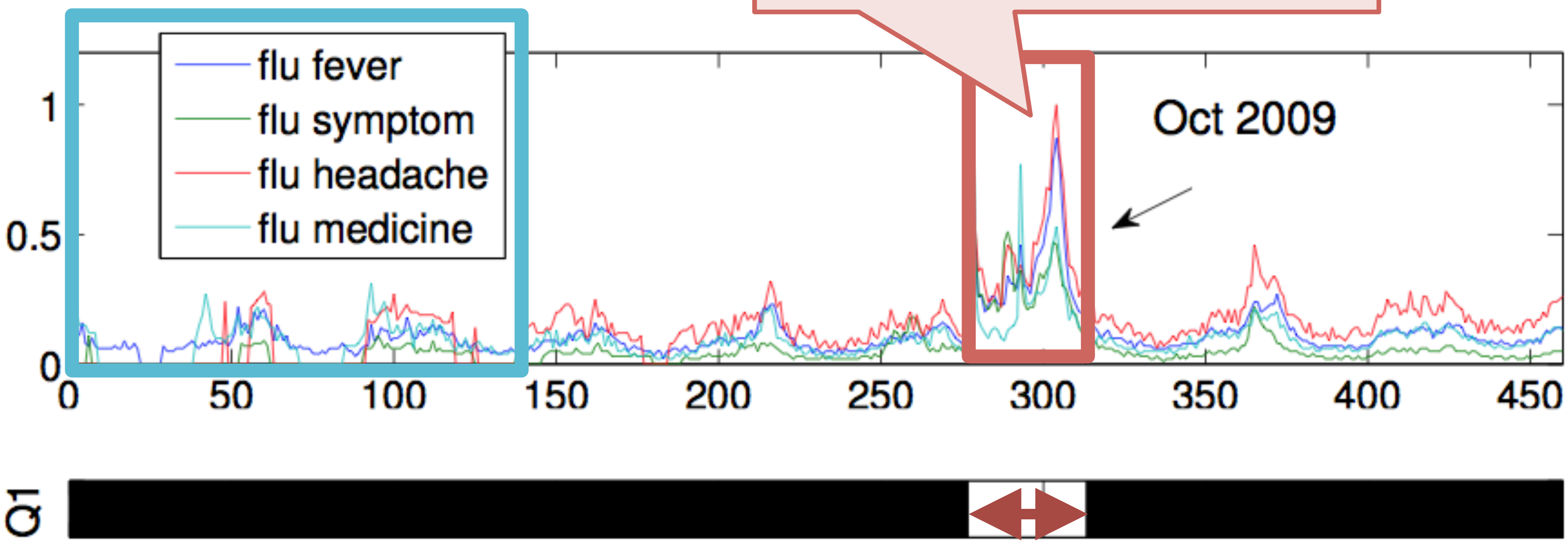
"Extreme monitoring" in keyword stream

Given: G streams (10 years, weekly)

Detected!

Swine flu pandemic in 2009

e.g., Flu-related keyw

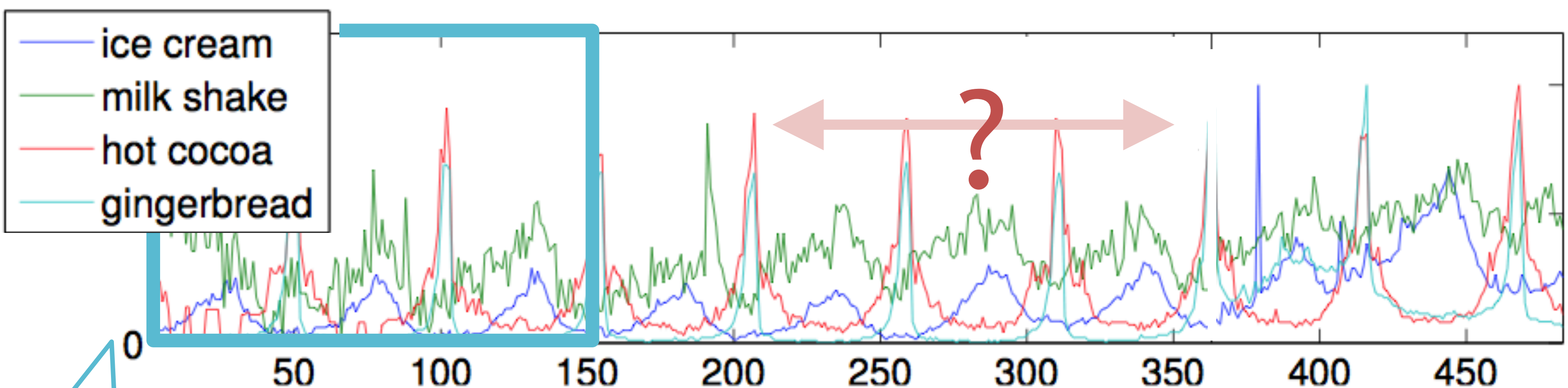


Application (2)

"Extreme monitoring" in keyword stream

Given: GoogleTrend streams (10 years, weekly)

e.g., Seasonal sweets-related keywords



Query (first 3 years)

Application (2)

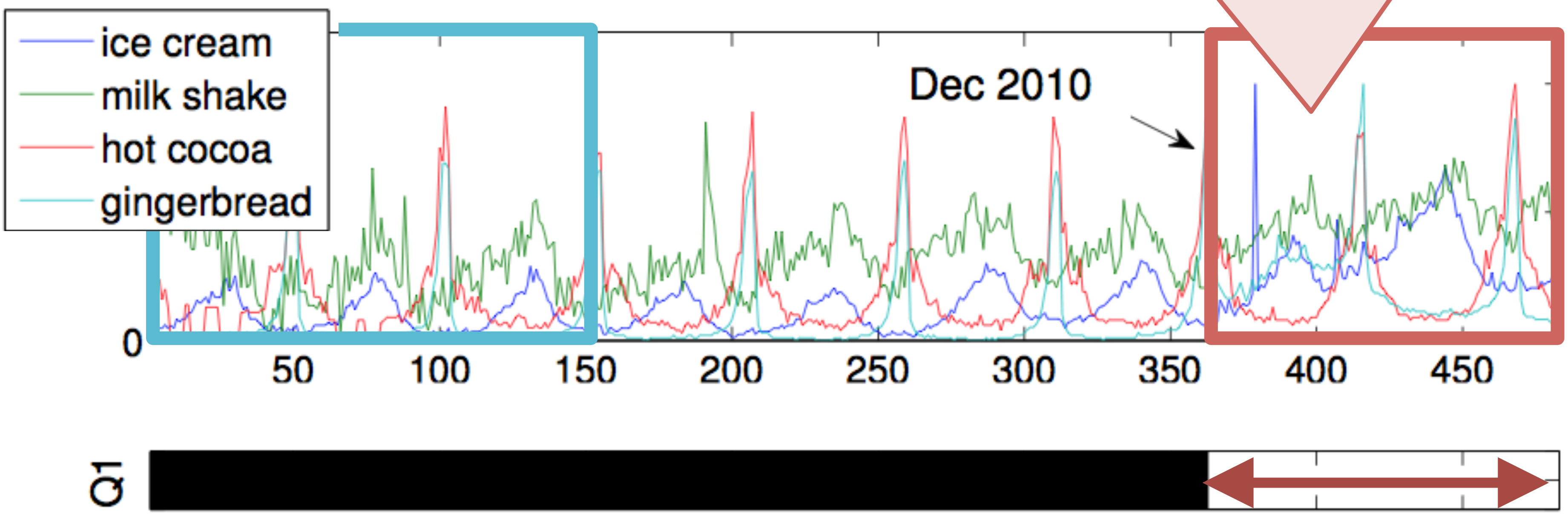
“Extreme m g” in keyword stream

Detected!

Given: Go r)

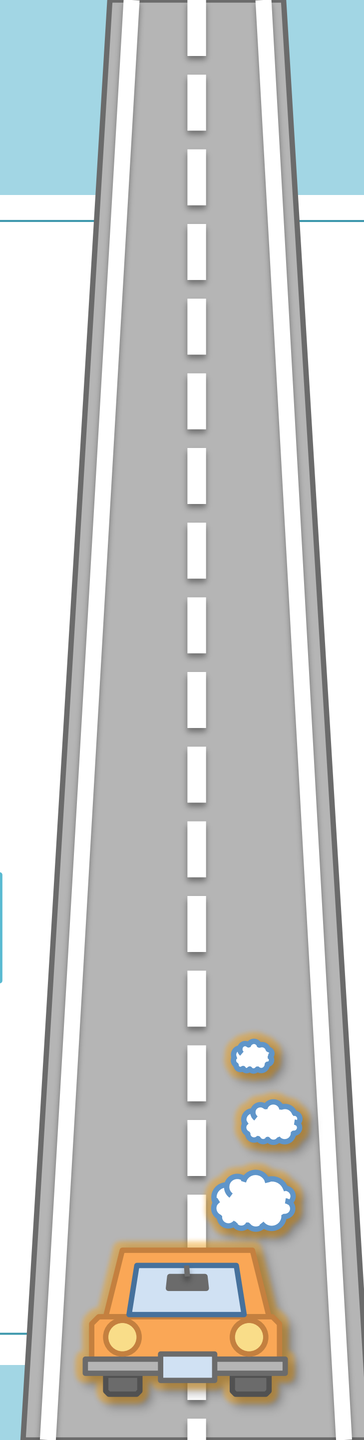
Release of Android OS,
“Gingerbread”
“Ice Cream Sandwich”

e.g., Seasonal sweets-r



Outline

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



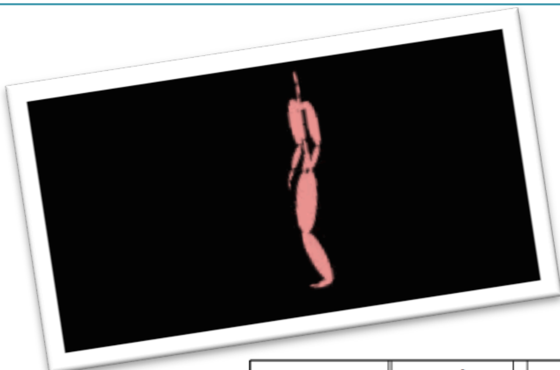
Conclusions

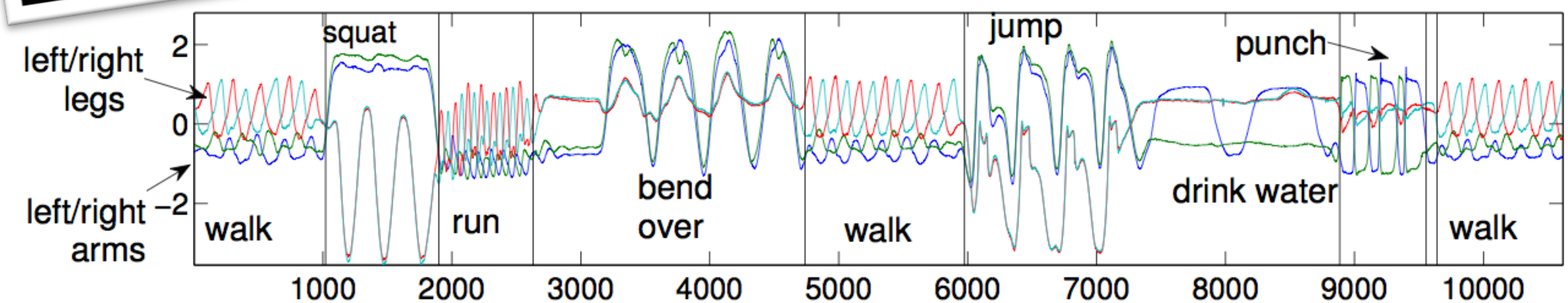
StreamScan has the following properties

- Effective ✔
  Find fruitful patterns from diverse data streams
- Exact ✔
  It guarantees exactness
- Fast and nimble ✔
  It requires single scan, O(1) space and time

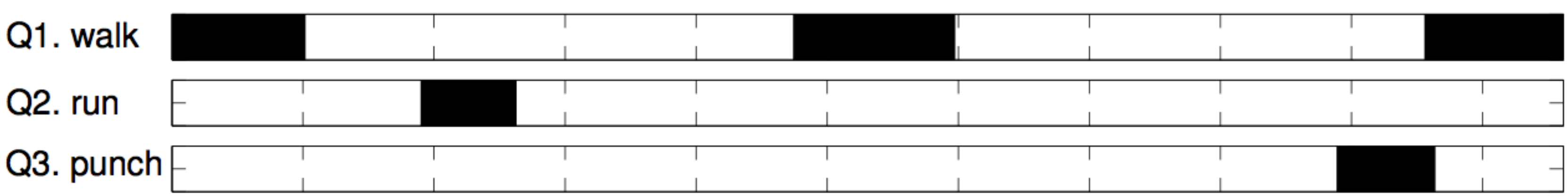
Thank you!





(b) Original *MoCap* stream



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

yasuko@cs.kumamoto-u.ac.jp