Scalable Algorithms for Distribution Search

Yasuko Matsubara (Kyoto University)
Yasushi Sakurai (NTT Communication Science Labs)
Masatoshi Yoshikawa (Kyoto University)

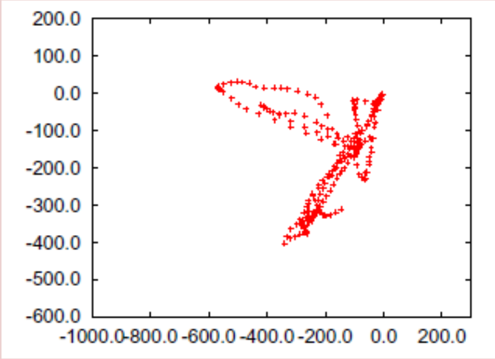
Introduction

Main intuition and motivation

Example: Motion capture

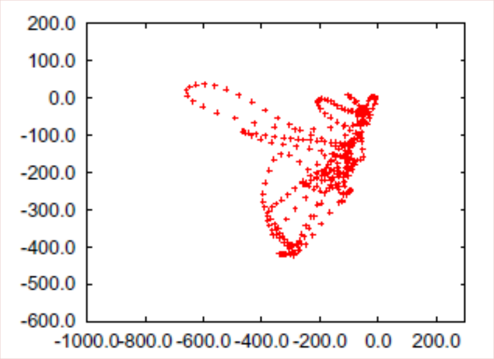
The scatter plots of foot kinetic energy values

#1 and #2 are similar and dissimilar distributions, respectively.



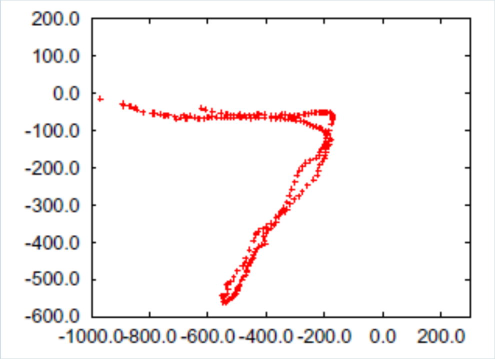
Query : Q

(running)



Distribution #1

(running)



Distribution #2

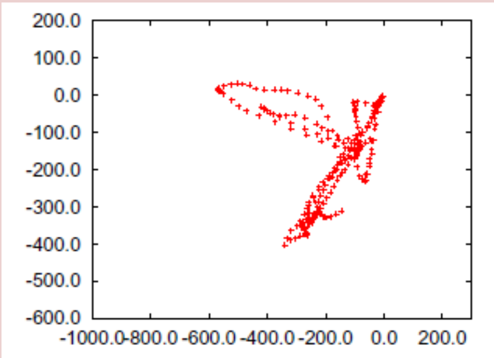
(squats)

- Our approach can identify Q and #1 as similar distributions

Problem definition

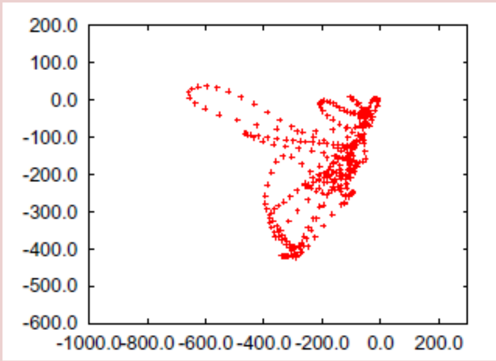
- Problem (Distribution search):

  "Given *n* distributions and query *Q*,
  Find similar distributions from the data set"



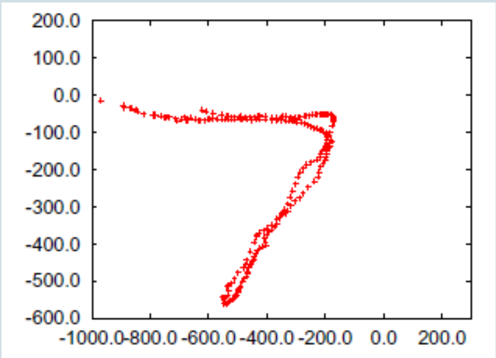
Query : Q

(running)



Distribution #1

(running)



Distribution #2

(squats)

Applications

- Distribution search application domains
  - Multimedia
  - Medical data analysis
  - Web service
  - E-commerce

Applications

- Multimedia

  Example: Motion capture datasets

  - Every motion can be represented as a cloud of hundreds of frames
  - For this collection of clouds, we can find similar motions without using annotations or other meta-data

Applications

- Web service

  Example: On demand TV

  - Discovering clusters and outliers in such data would help in tasks such as service design and content targeting

  (which groups or communities of users are associated with each other?)

Outline

- Introduction
- Background
- D-Search
- Time-series distribution mining
- Experiments
- Conclusions

Background

- Kullback-Leibler divergence

  Measures the natural distance difference
  from one probability distribution P
  to another arbitrary probability distribution Q.

$$d_{KL}(P,Q) = \int p_x \cdot \log\left(\frac{p_x}{q_x}\right) dx$$

* One undesirable property: $d_{KL}(P,Q) \neq d_{KL}(Q,P)$

- <u>Symmetric KL-divergence</u>

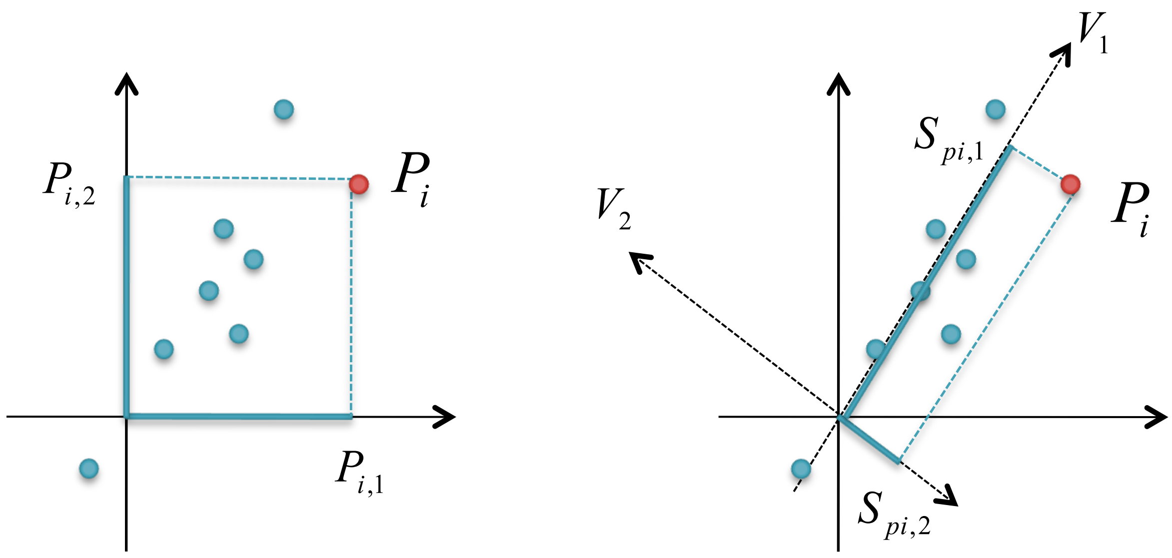
$$d_{SKL}(P,Q) = \int p_x \cdot \log\left(\frac{p_x}{q_x}\right) dx + \int q_x \cdot \log\left(\frac{q_x}{p_x}\right) dx = \int (p_x - q_x) \cdot \log\left(\frac{p_x}{q_x}\right) dx$$

Background

- Singular value decomposition (SVD)

  Every matrix $P \in \mathbf{R}^{m \times n}$ can be decomposed into

$$P = U \Sigma V^T$$



- The transformed data is given as:

$$S_p = U \Sigma$$

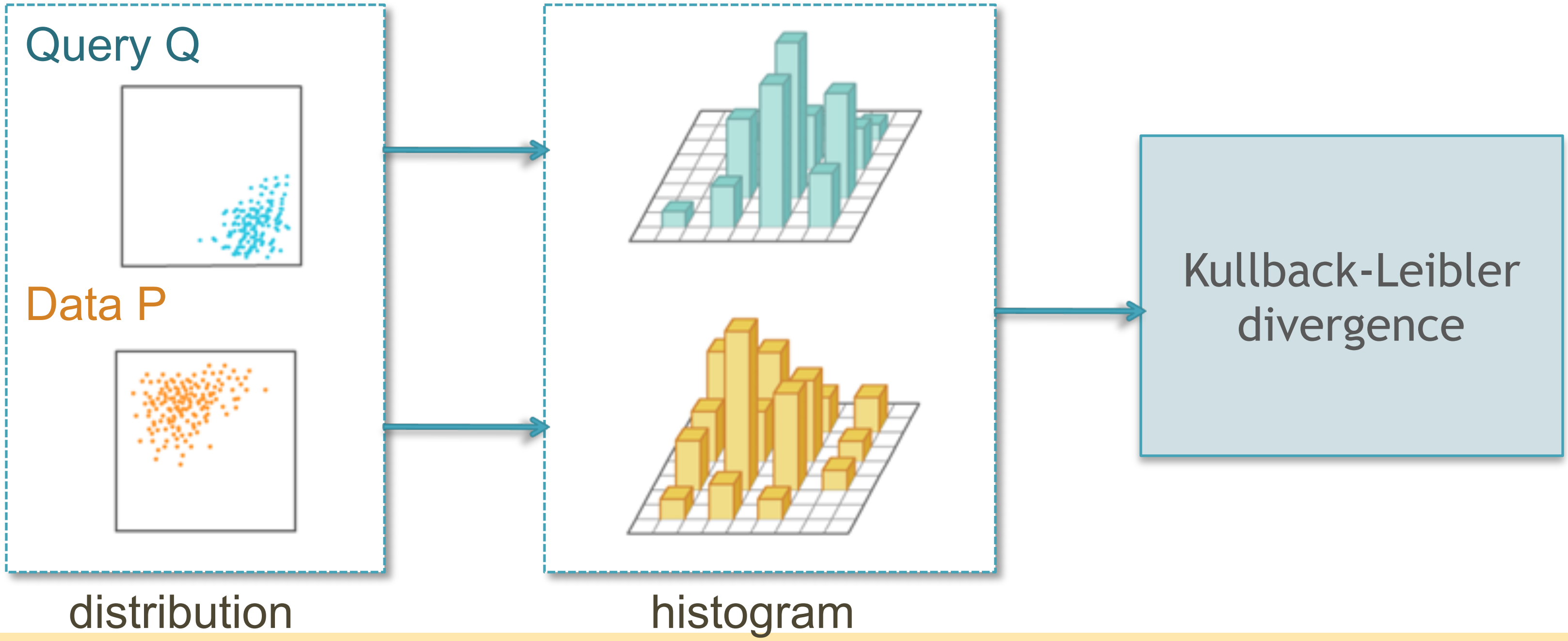
Outline

- Introduction
- Background
- D-Search
- Time-series distribution mining
- Experiments
- Conclusions

Proposed method

Naïve approach

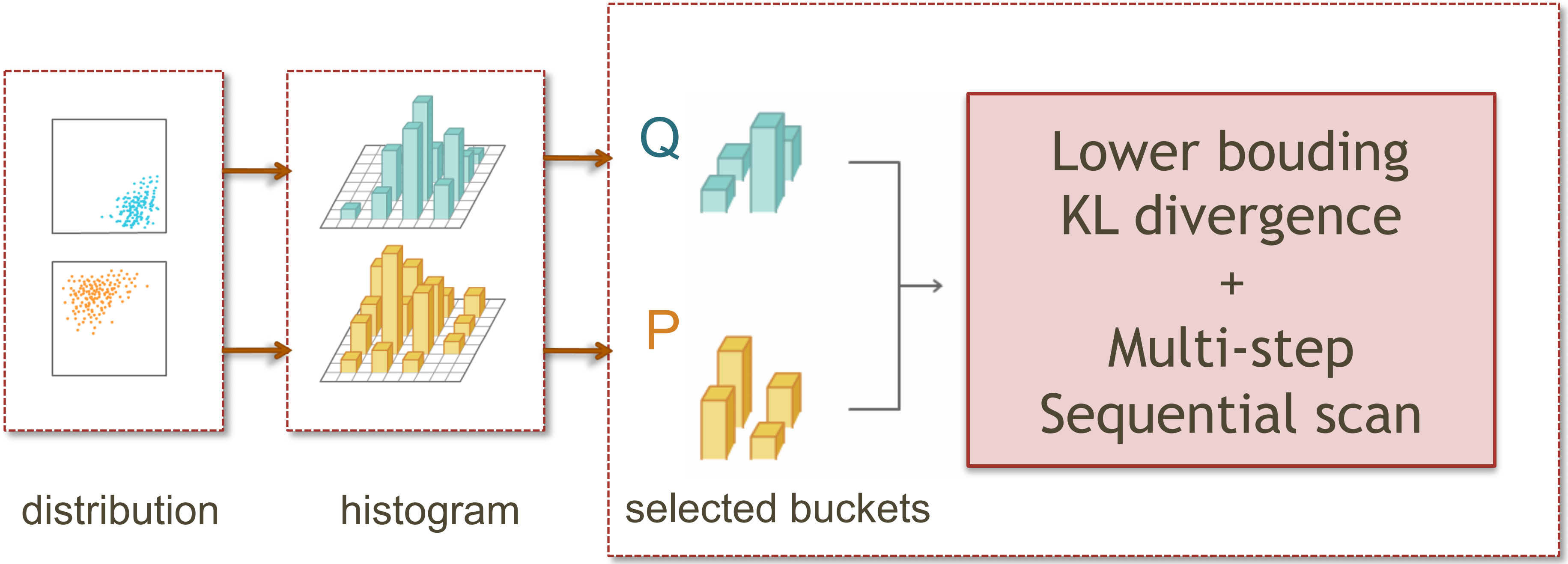
- Create histogram for each distribution of data
- Compute the KL divergence directly from histograms p_i and q_i
- Use any data mining method (k-nearest neighbor search)



Proposed method

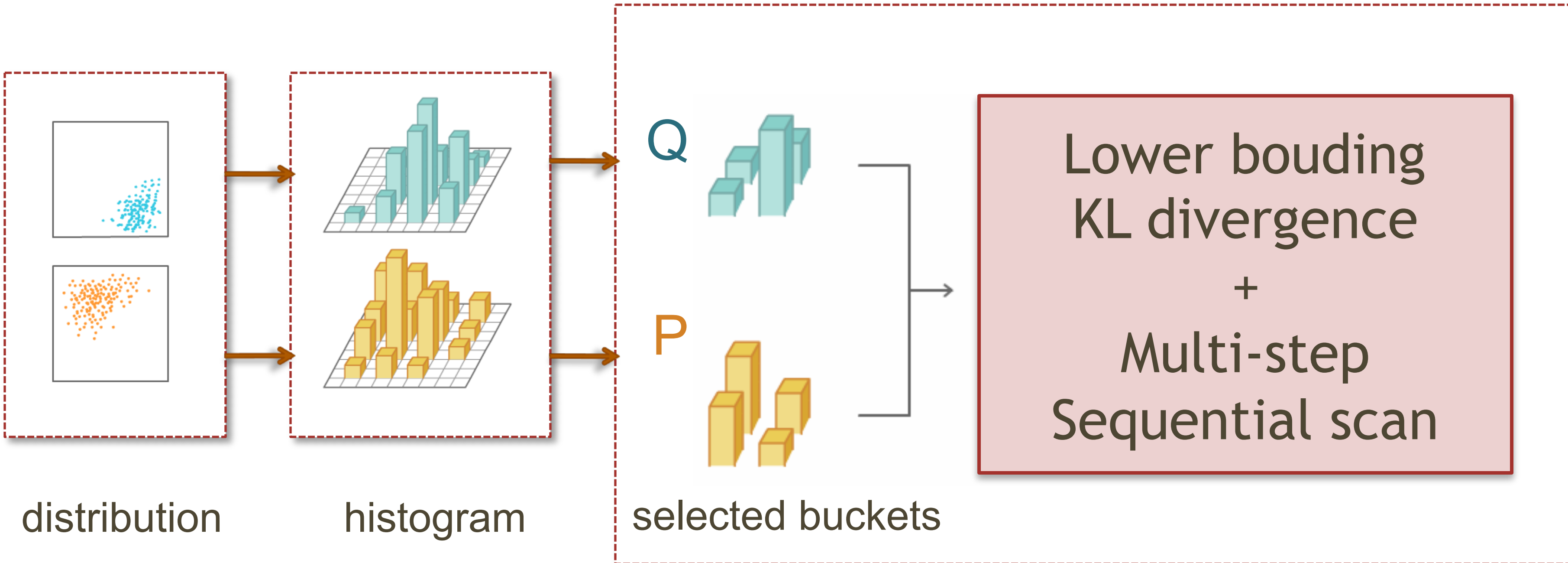
D-Search

- Compress histogram P and Q
- Compute the lower bounding KL divergence
- Prune the search candidates (Multi-step sequential scan)



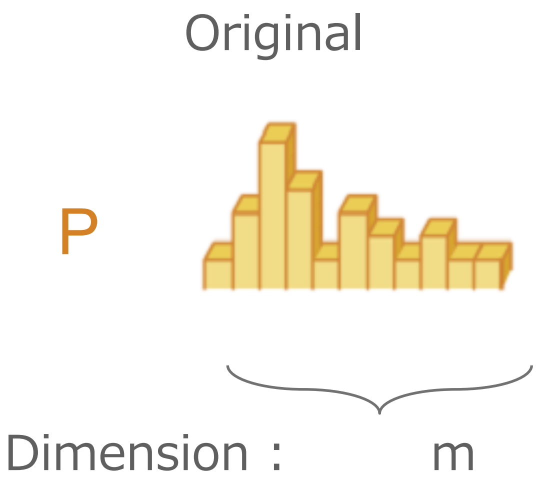
Proposed method

- **D-Search**
  - Compress histogram P and Q
  - Compute the lower bounding KL divergence
  - Prune the search candidates (Multi-step sequential scan)



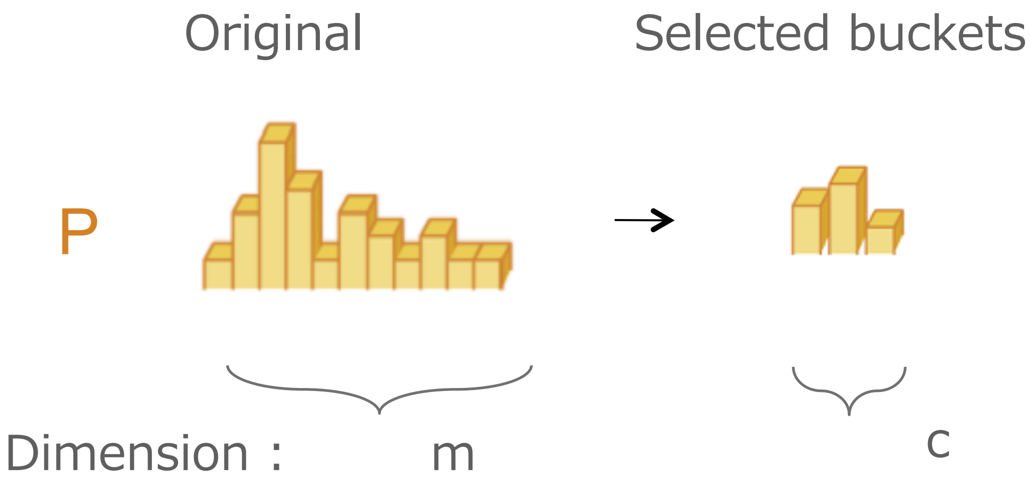
D-Search

- **Lower bounding KL divergence**
  - Create histogram for each distribution



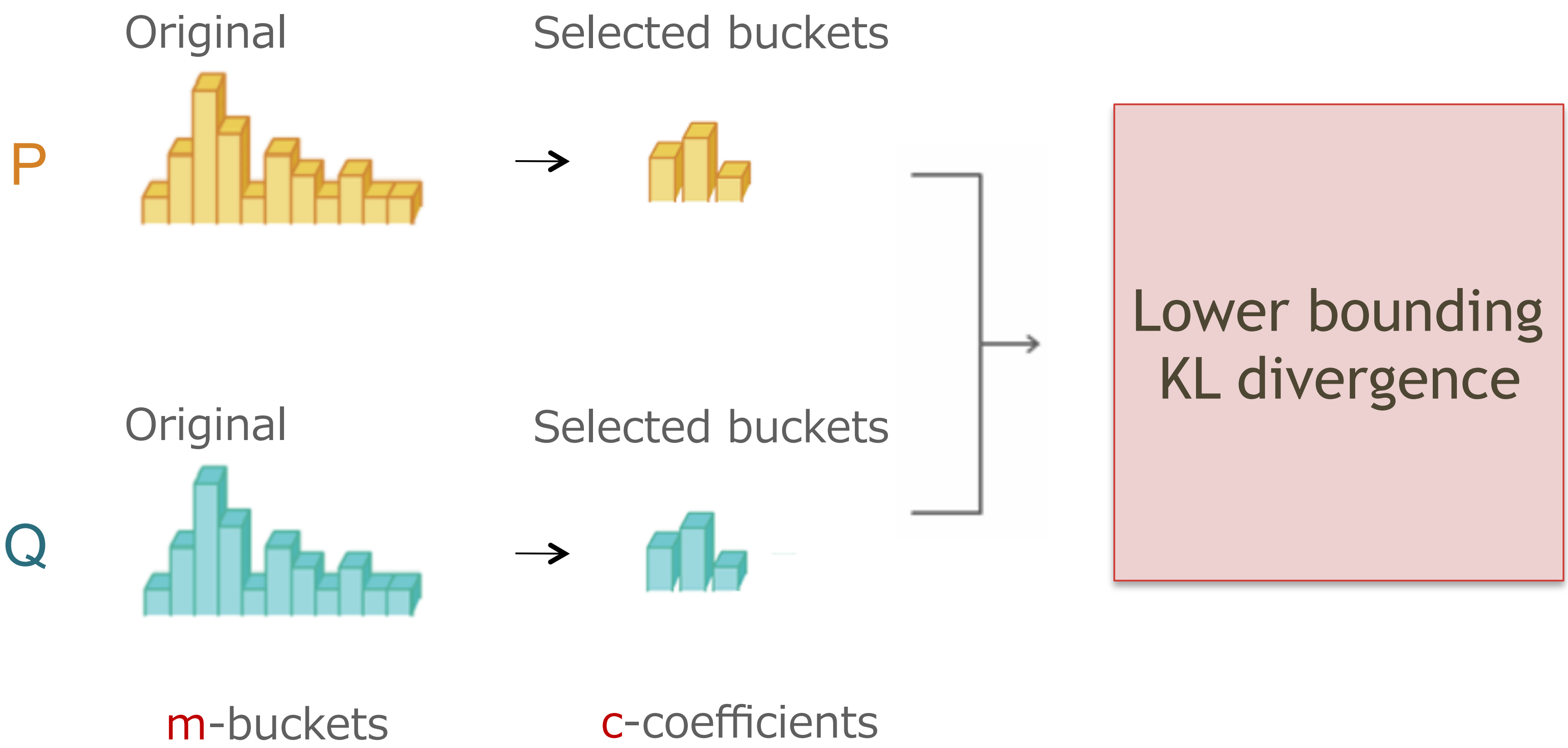
D-Search

- Lower bounding KL divergence
  - Create histogram for each distribution
  - Select the top c most populated buckets



D-Search

- Lower bounding KL divergence
  - Compute the KL divergence from the selected buckets



D-Search

- Lower bounding KL divergence
  - Compute the KL divergence from the selected buckets

$$d_c(P,Q) = \sum_{i \in I_{pq}} (p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right)$$

$i \in I_{pq}$: Positions of the top c values

D-Search

Lower bounding KL divergence

- Compute the KL divergence from the selected buckets

$$d_c(P,Q) = \sum_{i \in I_{pq}} (p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right)$$

$i \in I_{pq}$: Positions of the top c values

Lemma 1

$$d_{SKL}(P,Q) \geq d_c(P,Q)$$

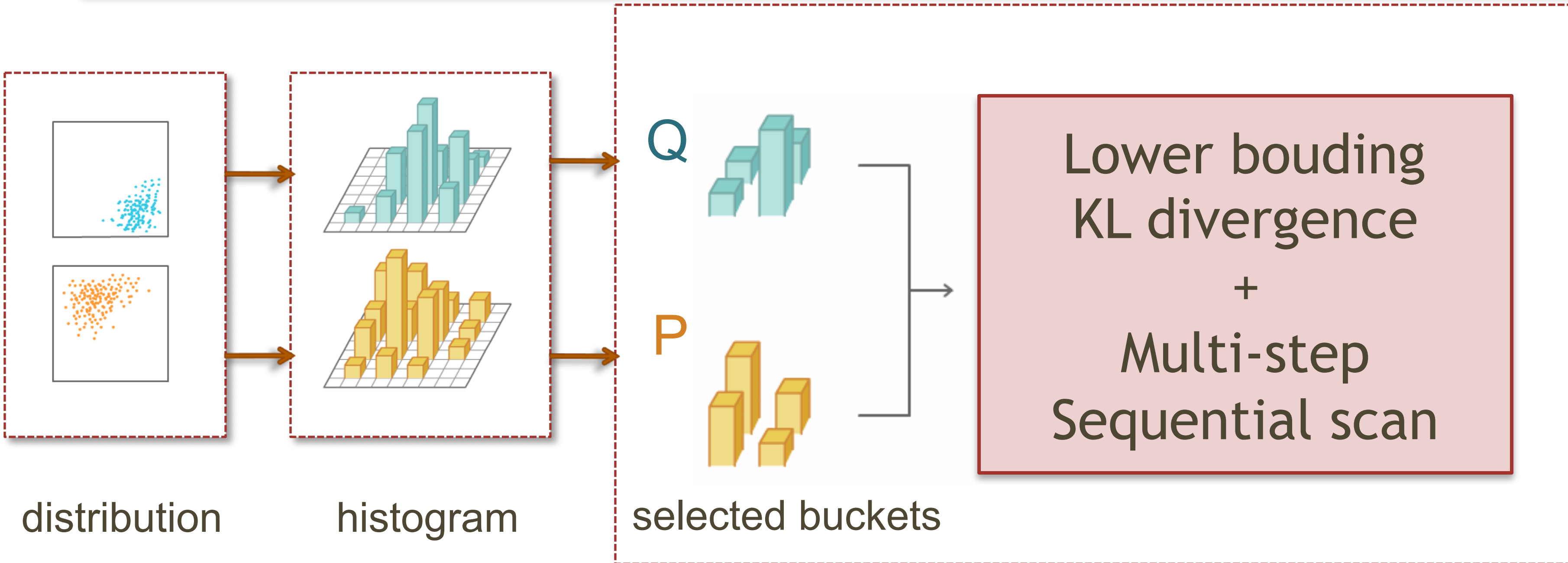
For any distributions, lower bounding KL divergence can be computed

$$\because \forall i, (p_i - q_i)(\log p_i - \log q_i) \geq 0$$

Proposed method

D-Search

- Compress histogram P and Q
- Compute the lower bounding KL divergence
- Prune the search candidates (Multi-step sequential scan)



D-Search

- Multi-Step Sequential Scan
  - KNN-search approach based on the lower bounding distance

    - Prune a significant number of search candidates
    - Lead to a direct reduction in the search cost

  - Guarantee no false dismissals

    (i.e., guarantee the exactness of search results)

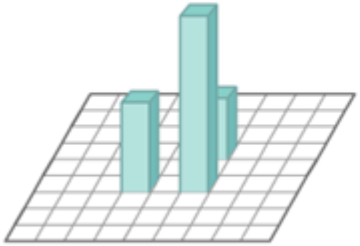
D-Search

- For the first step,
  - Compute the lower bounding distance from the coarsest version of P ((a) c=4)
  - If the distance is greater than D_{cb} (the current k-th nearest neighbor distance), we can prune P

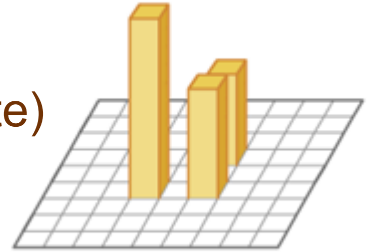
Step 1

(a) c = 4 (b) c = 8 (c) Original data

Q (query)

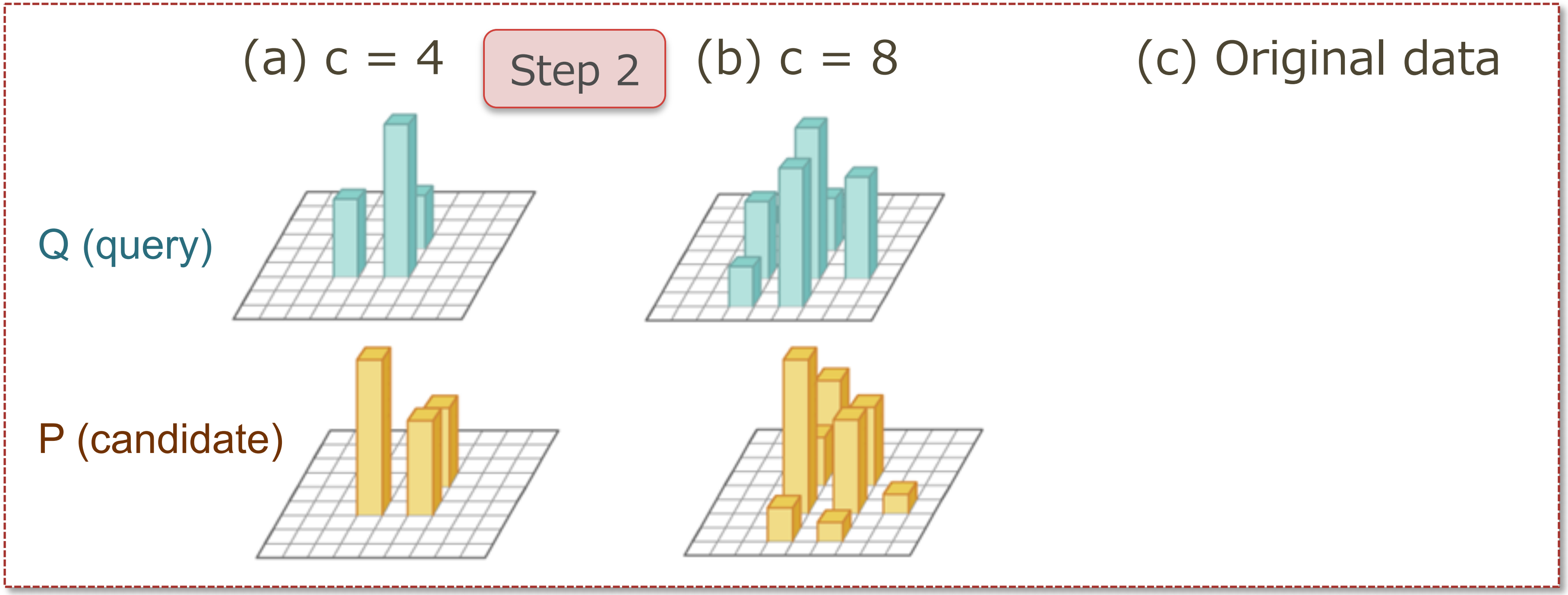


P (candidate)



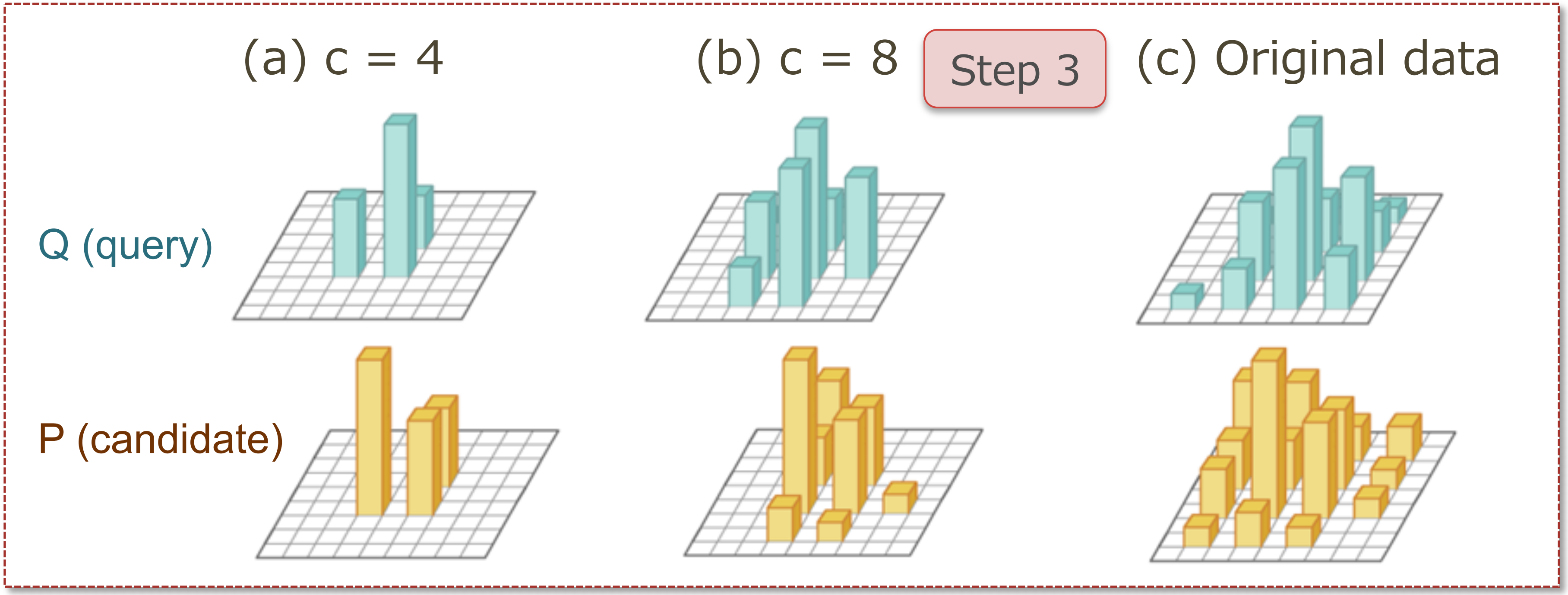
D-Search

- Otherwise,
  for the second step,
  - Compute the lower bounding distance from the more accurate version of P ((b) c=8)



D-Search

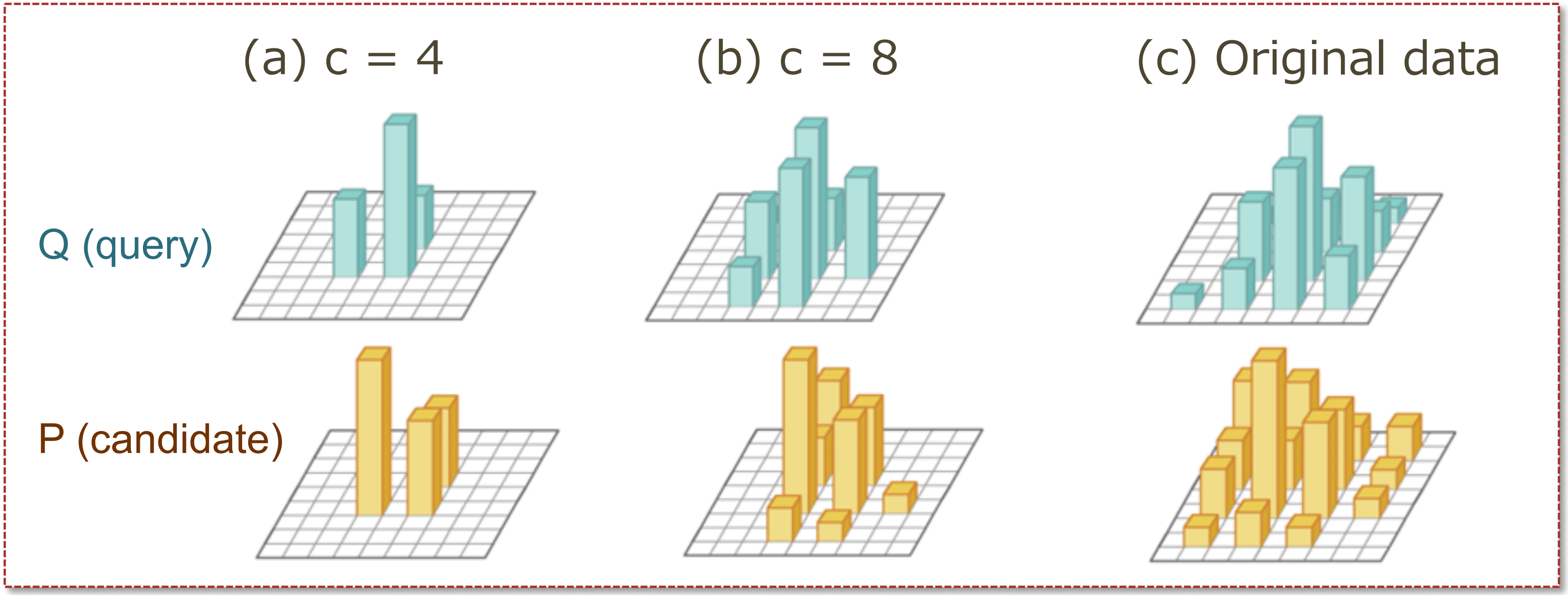
- If the lower bounding distance does not exceed D_{cb} for the third step,
  - Compute the exact distance of P ((c) Original data)



D-Search

- For the final step,
  If the exact distance does not exceed D_{cb}
  - Update the answer candidate and D_{cb}

- Repeat this procedure for every distribution



D-Search

- ## Enhanced D-Search

  More efficient solution without a theoretical guarantee

D-Search

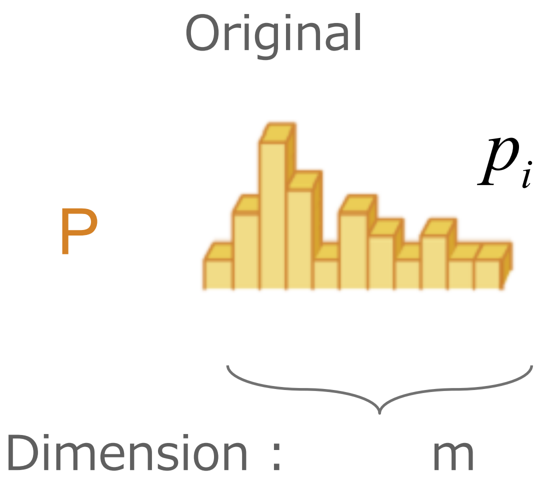
Enhanced D-Search

More efficient solution without a theoretical guarantee

- Compute the SVD coefficients of histogram P and Q
- Approximate the KL divergence

Enhanced D-Search

- Approximate KL divergence
  - Create histogram for each distribution



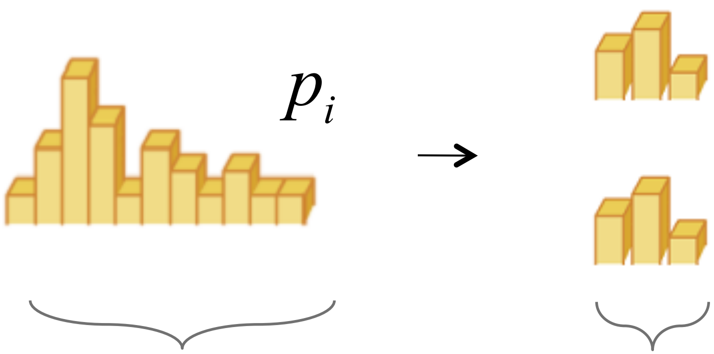
Enhanced D-Search

Approximate KL divergence

- Create histogram for each distribution
- Represent each histogram p_i and $\log(p_i)$ as sp_i and $s\hat{p}_i$ using SVDs
- Reduce the number of SVDs by selecting top *c*

Original SVD coeffs

P



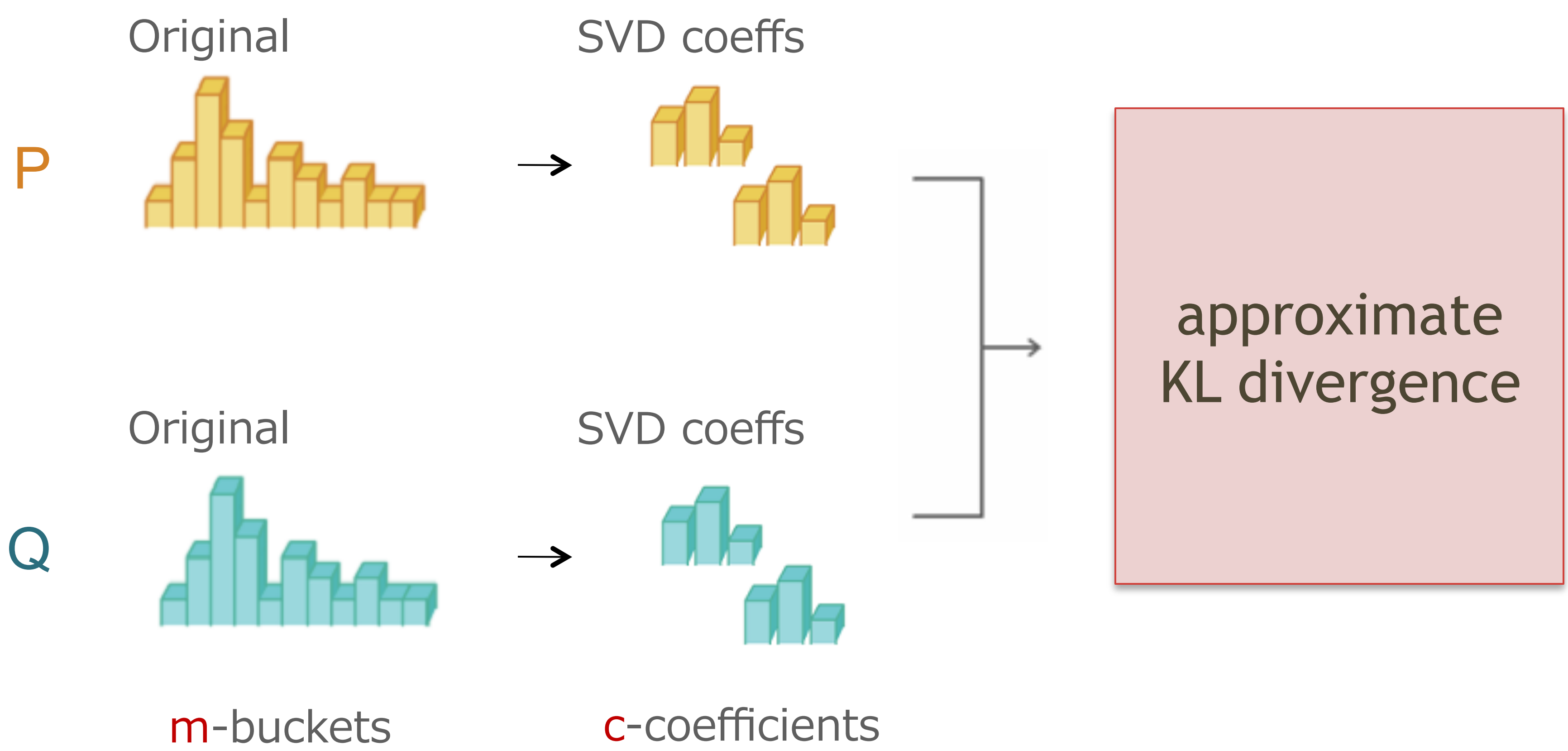
sp_i : The SVD of p_i

$s\hat{p}_i$: The SVD of $\log(p_i)$

Dimension : m c

Enhanced D-Search

- Approximate KL divergence
  - Compute the KL divergence from the SVDs



Enhanced D-Search

m: # of buckets of a histogram
c: # of SVD coefficients
(m >> c)

Theorem 1

Let

sp_i and sq_i be the SVD of p_i and q_i resp.

$s\hat{p}_i$ and $\hat{s}q_i$ be the SVD of $\log p_i$ and $\log q_i$ resp.

We have

$$d_{SKL}(P,Q) = \sum_{i=1}^{m} (p_i - q_i) \cdot \log\left(\frac{p_i}{q_i}\right)$$

$$= \sum_{i=1}^{m} (p_i - q_i) \cdot (\log p_i - \log q_i)$$

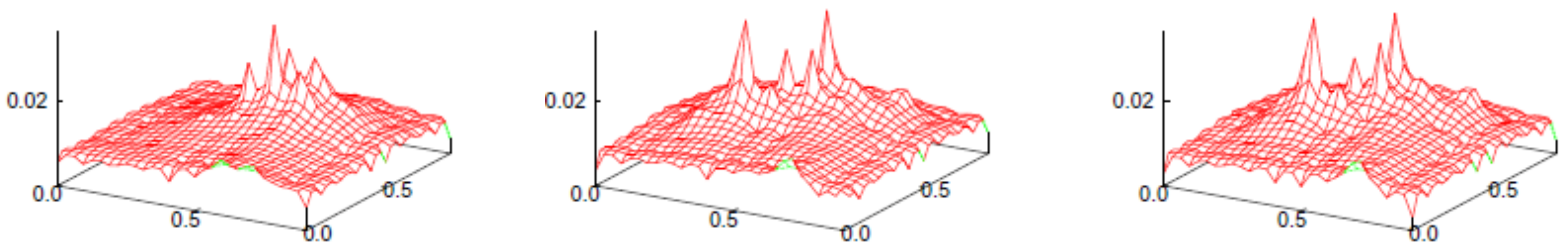
$$\approx \frac{1}{2} \cdot \sum_{i=1}^{c} \begin{pmatrix} (sp_i - \hat{s}q_i)^2 + (sq_i - \hat{s}p_i)^2 \\ -(sp_i - \hat{s}p_i)^2 - (sq_i - \hat{s}q_i)^2 \end{pmatrix}$$

Approx. KL divergence can be computed from SVD coefficients

Enhanced D-Search

- Multi-Step Sequential Scan
  - SVD-based approx. of distribution from MoCap
  - Represented by a 10*10 bucketized histogram
  - (Full coefficients c = m = 100)

(a) c = 1 (b) c = 16 (c) Original data



Enhanced D-Search

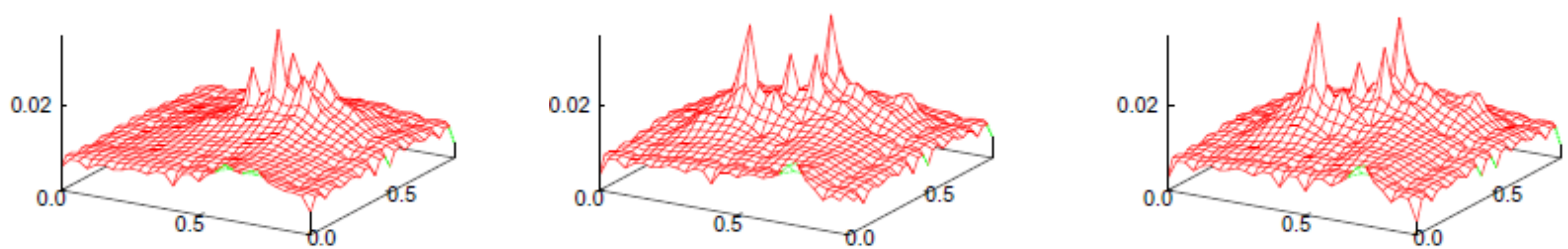
- <u>Gradual refinement of the approximation:</u>

  For the first step,

  - Compute the approx. distance from the coarsest version of the distribution ((a) c=1)
  - If the distance is greater than D_{cb} , we can prune it

Step 1

(a) c = 1 (b) c = 16 (c) Original data



Enhanced D-Search

- <u>Gradual refinement of the approximation:</u>

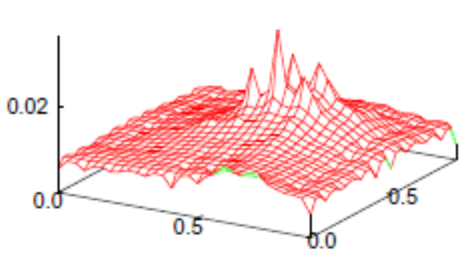
  Otherwise,

  for the second step,

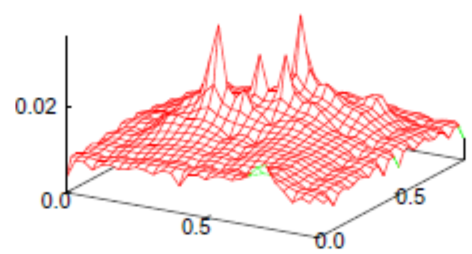
  - Compute the approx. distance from the more accurate version of the distribution ((b) c=16)

Step 2

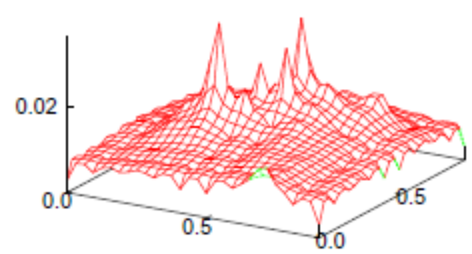
(a) c = 1



(b) c = 16



(c) Original data



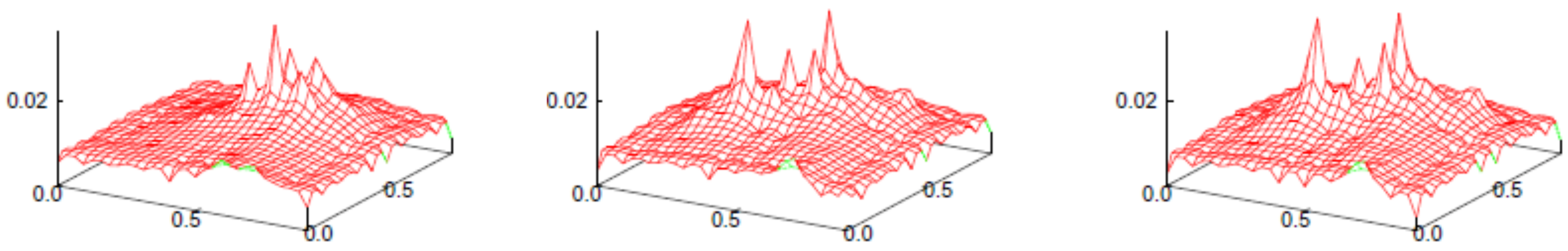
Enhanced D-Search

- <u>Gradual refinement of the approximation:</u>

  If the approx. distance does not exceed D_{cb}
  for the third step,
  - Compute the exact distance
    from the original distribution ((c) original data)

Step 3

(a) c = 1 (b) c = 16 (c) Original data



Enhanced D-Search

- Gradual refinement of the approximation:

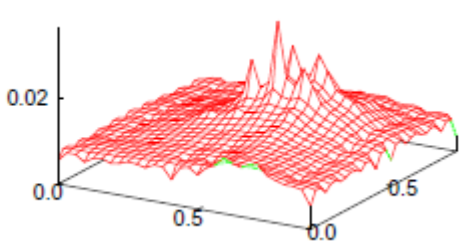
  For the final step,

  If the exact distance does not exceed D_{cb}

  - Update the answer candidate and D_{cb}

- Repeat this procedure for every distribution

(a) c = 1 (b) c = 16 (c) Original data



Time complexity

- Computation for KL divergence

Naïve method	D-Search
$O(mn)$	$O(n)$

n: # of input distributions
m: # of buckets of histogram
c : # of SVD coefficients we use

D-Search :

- requires O(cn)

• c is a small constant and negligible

Space complexity

- Space for our method

Naïve method	D-Search
$O(mn)$	$O(m+n)$

D-Search :

- allocates space to store histogram of m buckets
- allocates O(cn) space for computing the criterion
- We obtain O(m + cn)

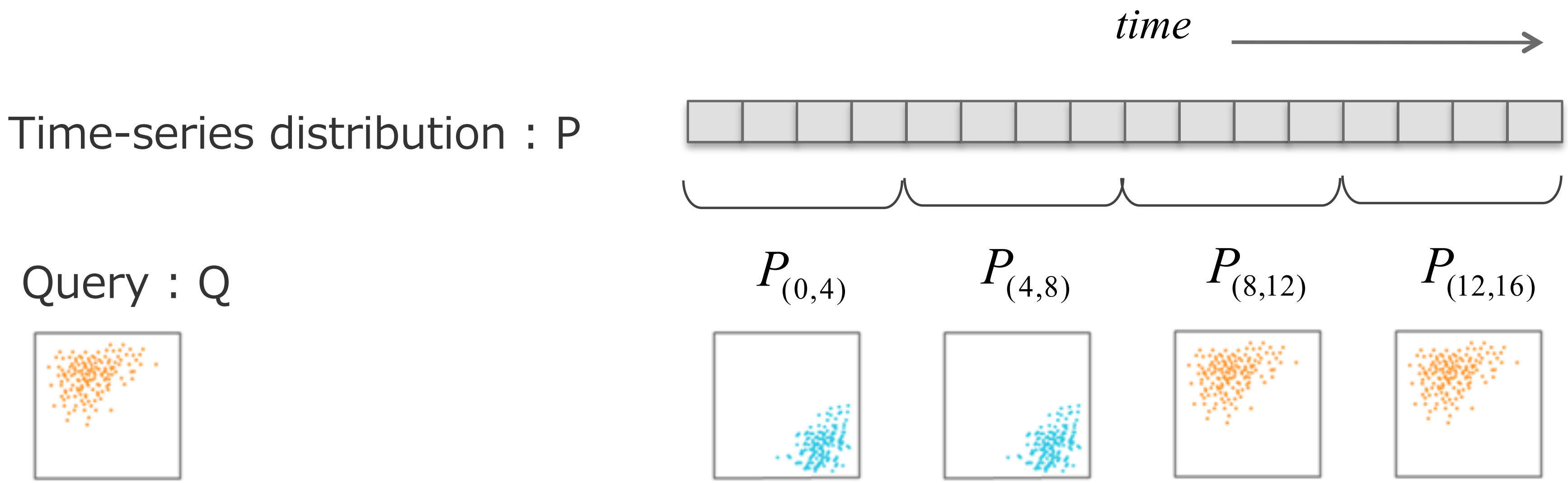
* c is a small constant and negligible

Outline

- Introduction
- Background
- D-Search
- **Time-series distribution mining**
- Experiments
- Conclusions

Time-series distribution mining

- Problem:
  - Given time-series distribution P and query Q,
  - Finds similar subsequences



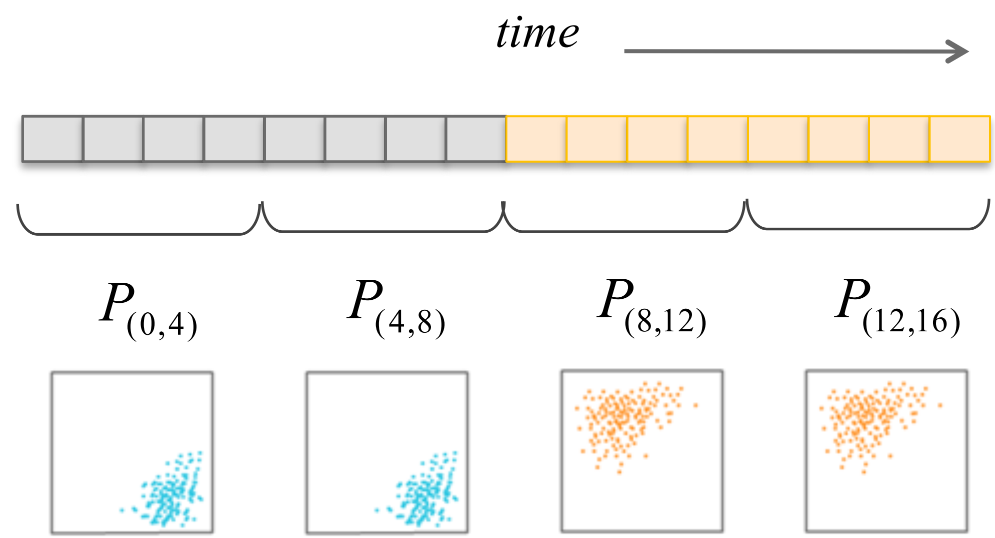
Time-series distribution mining

Problem:

Example:

We want to find three subsequences : 8-12sec., 12-16sec., 8-16sec.

Time-series distribution : P



Query : Q

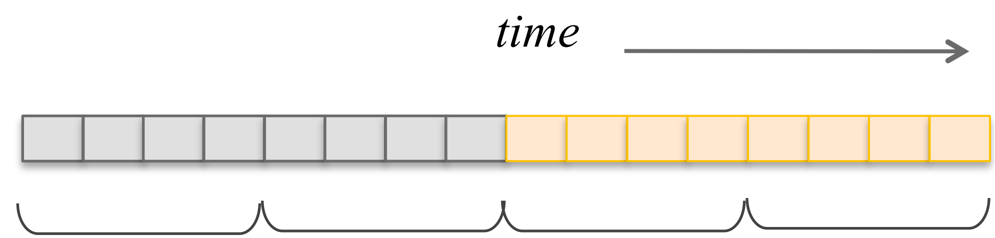
Time-series distribution mining

Problem:

Example:

We want to find three subsequences : 8-12sec., 12-16sec., 8-16sec.

Time-series distribution : P

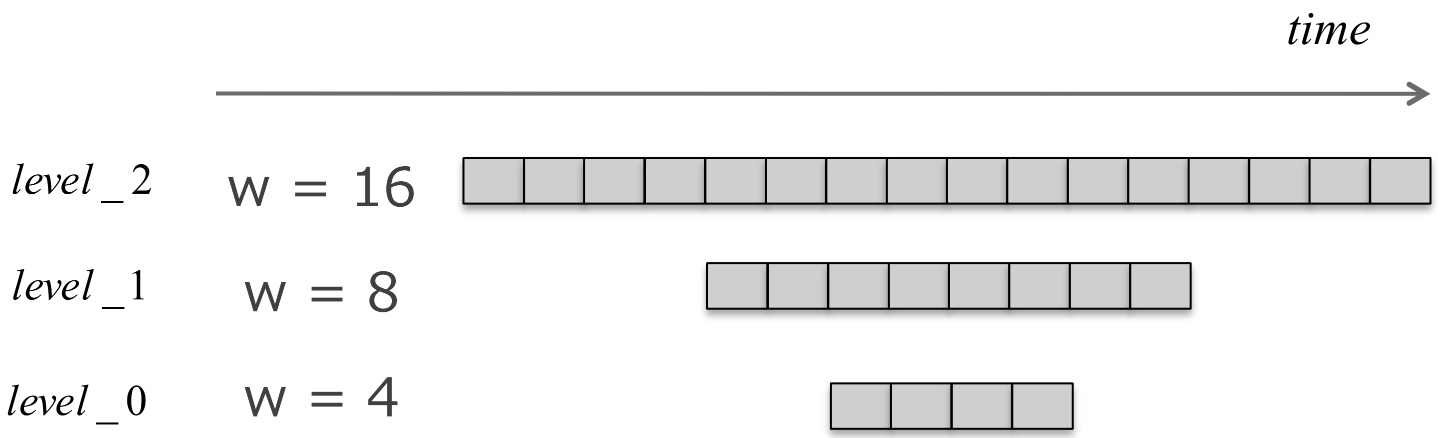


Q: How do we efficiently find the similar subsequences for multiple lengths?

Time-series distribution mining

A: Use hierarchical window sizes

Main idea: Geometric progression of windows sizes



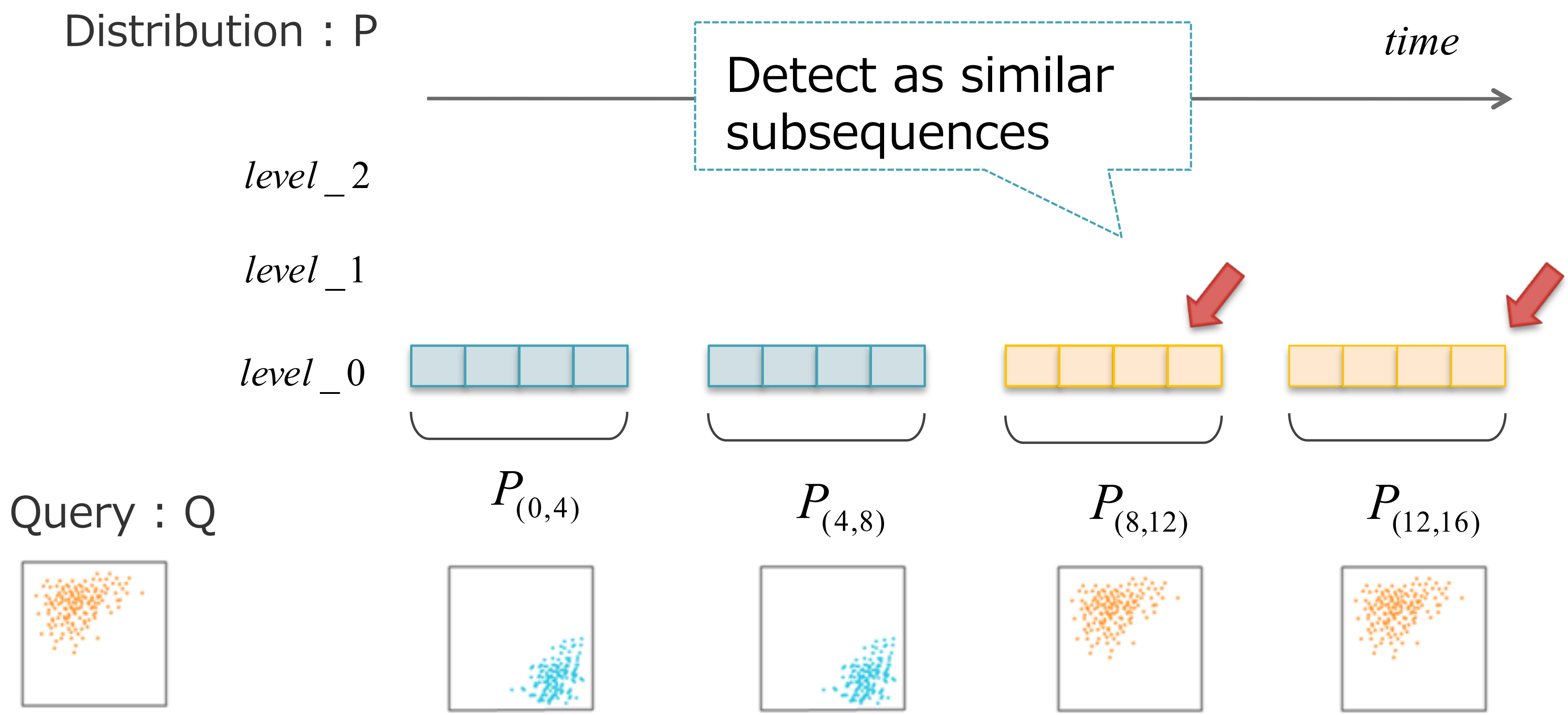
Query : Q

$$w = w_0 \cdot 2^l \qquad l = \{0,1,2,3,\dots\}$$

- The size of the window set can be reduced

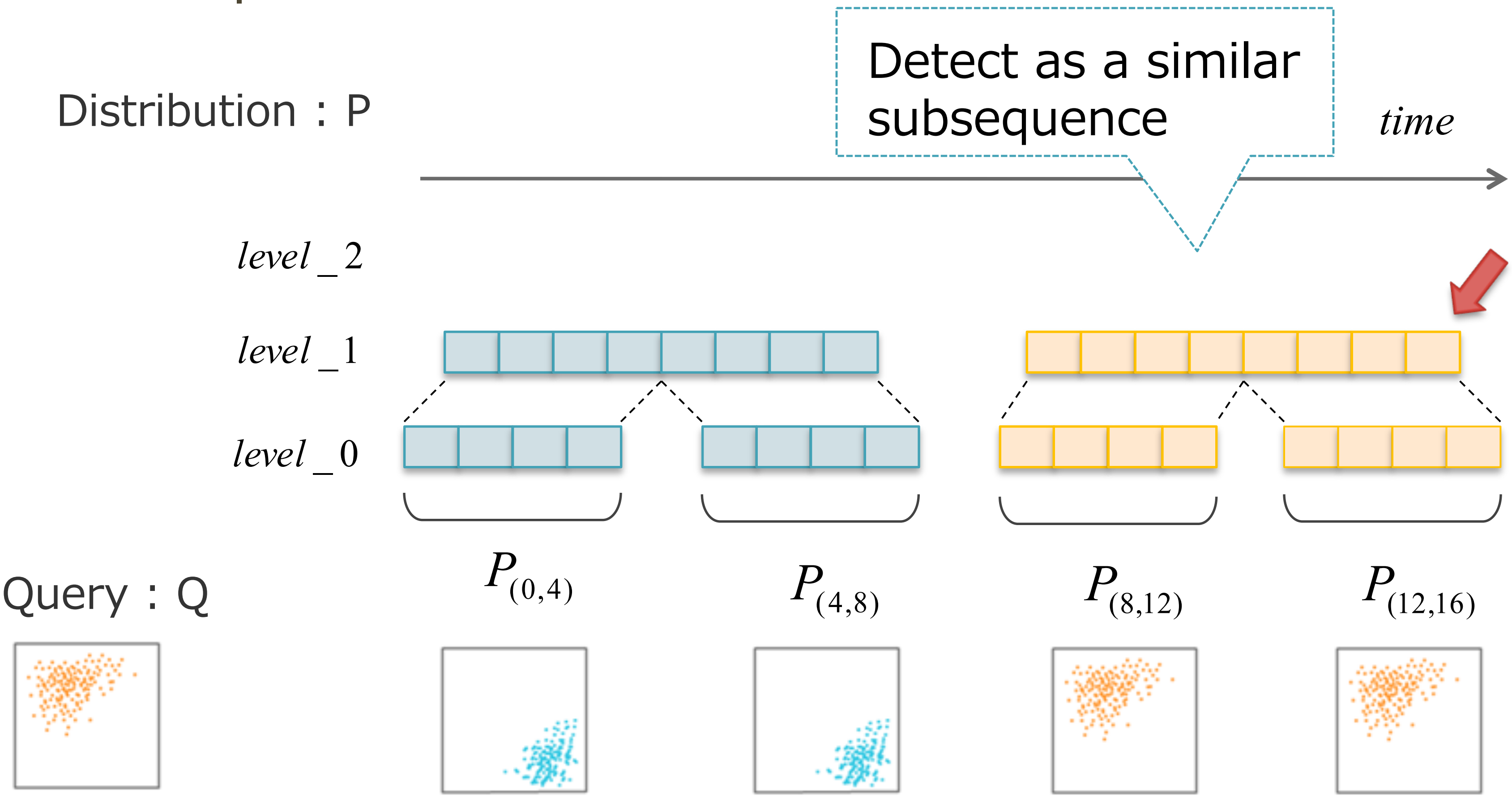
Time-series distribution mining

- How to detect similar subsequences
  - Example: at the level 0



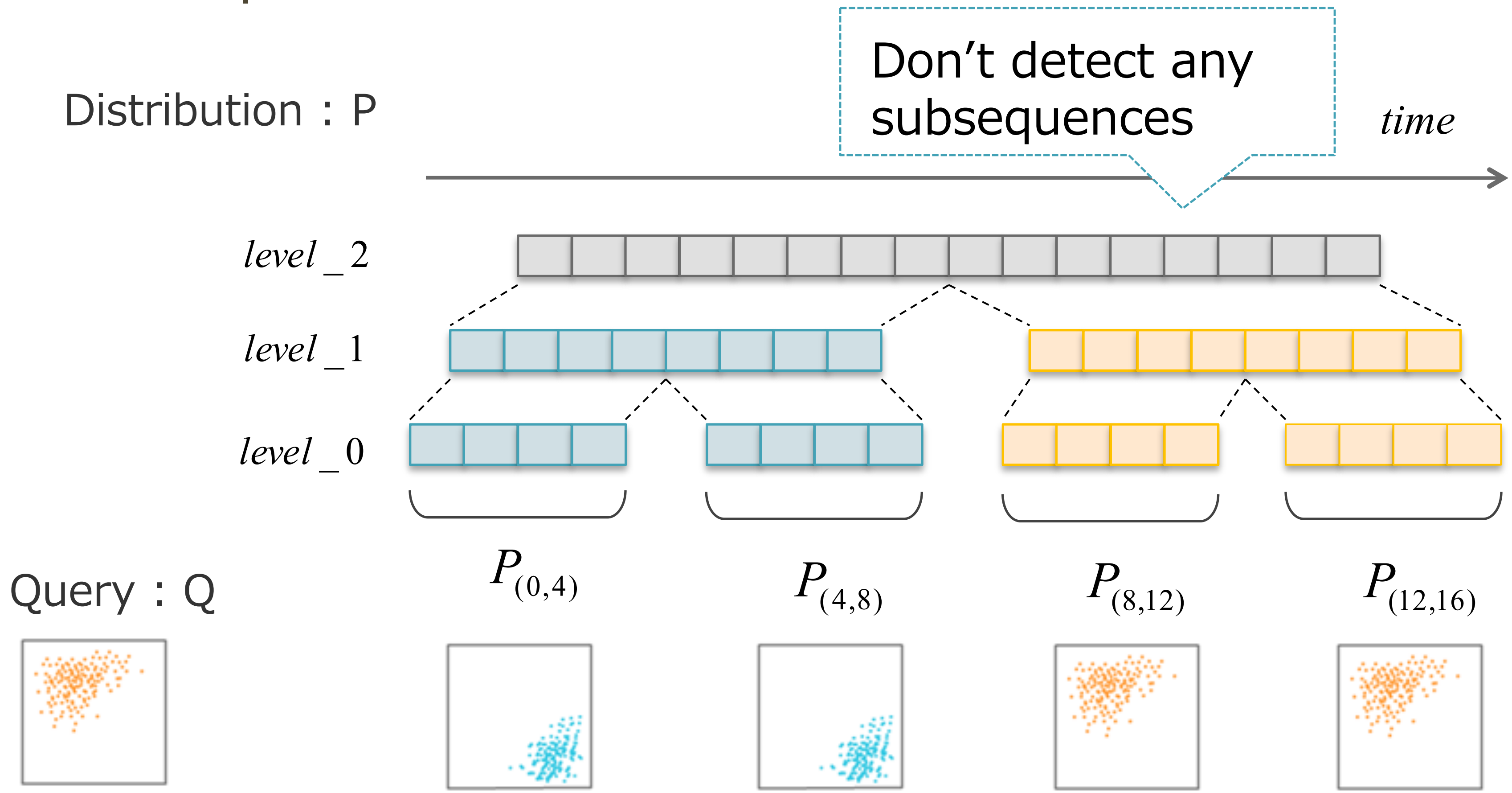
Time-series distribution mining

- How to detect similar subsequences
  - Example: at the level 1



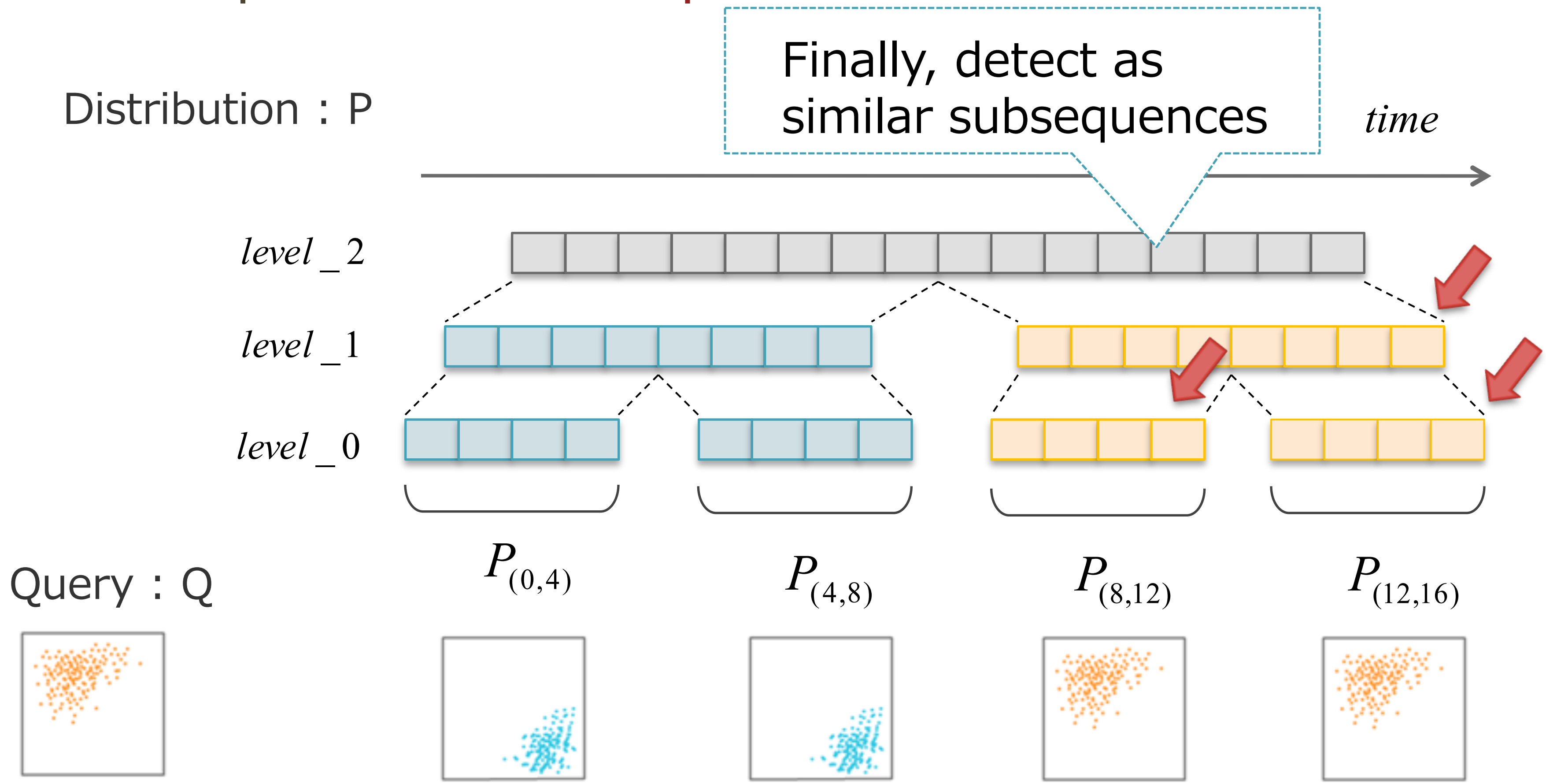
Time-series distribution mining

- How to detect similar subsequences
  - Example: at the level 2



Time-series distribution mining

- How to detect similar subsequences
  - Example: at the multiple levels



Outline

- Introduction
- Background
- D-Search
- Time-series distribution mining
- Experiments
- Conclusions

Experimental evaluation

- The experiments were designed to answer the three questions:

1. Effectiveness

How successful is **D-Search (enhanced)** in capturing time-series distribution patterns?

2. Speed

How does **D-Search** scale with the sequence lengths n in terms of the computational time?

3. Quality

How well does **D-Search** approximate the KL divergence?

Experimental evaluation

- We carried out experiments on real datasets:

Numerical data

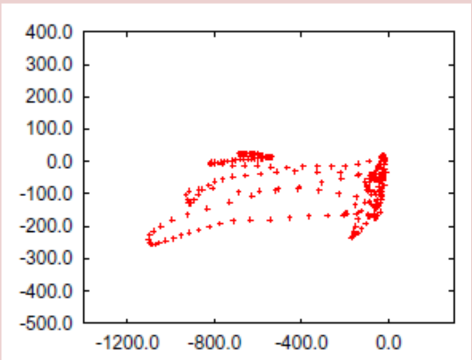
- **Motion capture**
  - It contains 26 sequences, each of which is a series of simple motions such as walking, running, jumping
- **EEG**
  - It is from a large study that examined the EEG correlates of alcoholism. There were two subject groups: alcoholic and control

Categorical data

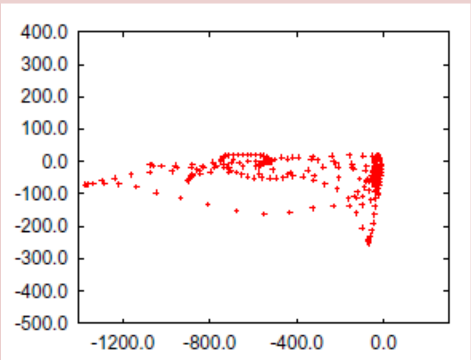
- **On-demand TV**
  - Dataset from the on-demand TV service. It contains a list of content ID, Date, user ID
- **Music store**
  - This dataset consists of the purchasing records from an on-line music store obtained over 16 months

Case studies

(1) Motion capture

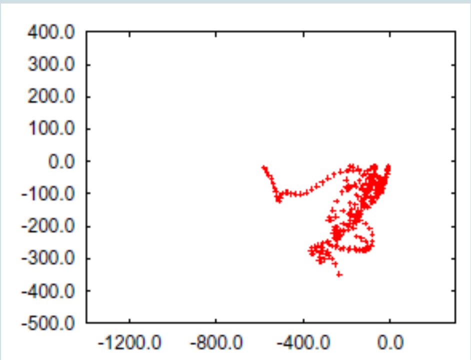


Query : jumping



#1 : jumping

ID:14_14 (2-4sec.)



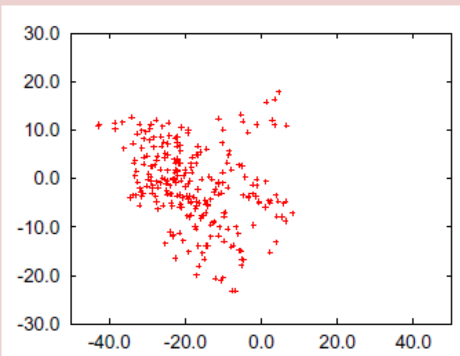
#2 : jogging

ID:14_14 (4-6sec.)

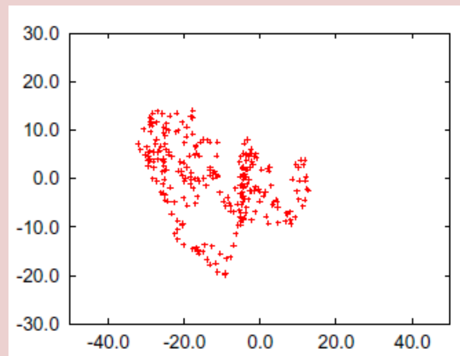
- D-Search can identify similar subsequences
  - Query and #1 both correspond to a **jumping** motion
  - #2 corresponds to a **jogging** motion

Case studies

(2) EEG (Alcohol or Control)

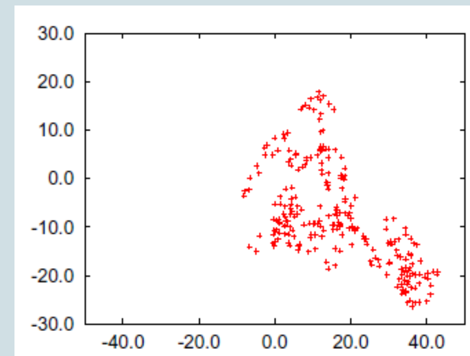


Query : alcohol



#1 : alcohol

co3a (55-56sec.)



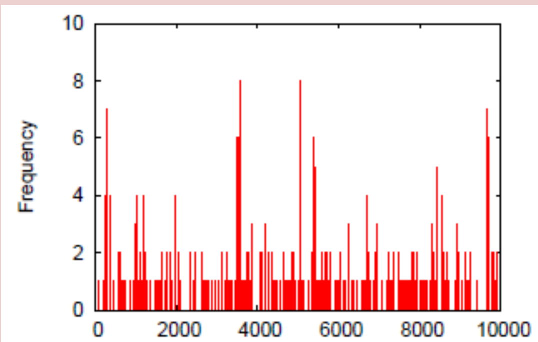
#2 : control

co2c (74-75sec.)

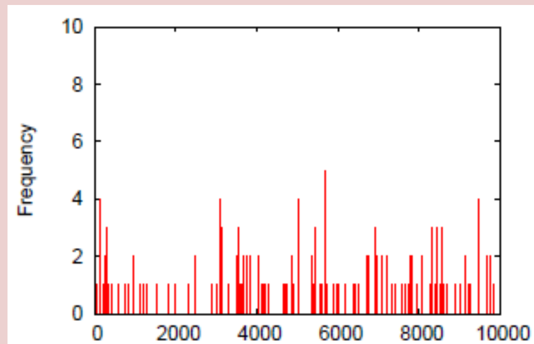
- Our approach is also useful for classification
  - Query and #1 are classified into the same group
  - #2 goes to another group (it belongs to “control”)

Case studies

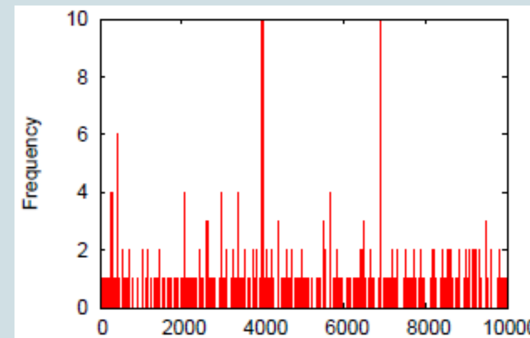
(3) On-demand TV (distribution of users)



Query : tennis final



#1 : tennis semifinal

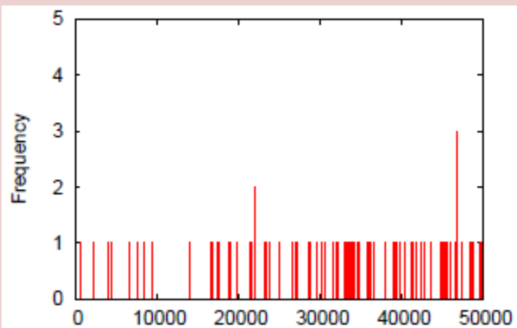


#2 : cooking

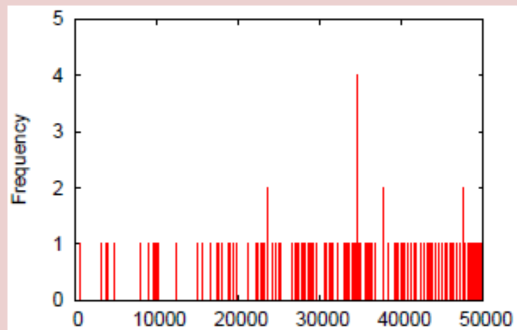
- D-Search can find similar trends in Q and #1
  - “Australian Open Tennis final” and “Australian Open Tennis semi-final”

Case studies

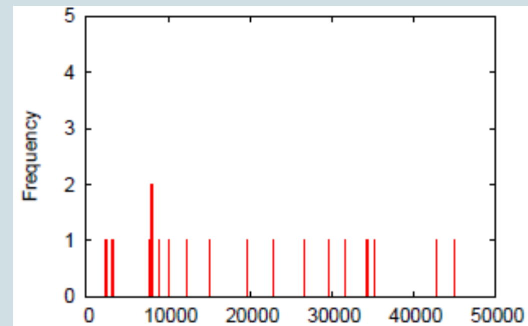
(4) Music Store (distribution of purchasers)



Query : John Lennon



#1 : John Lennon



#2 : Nat King Cole

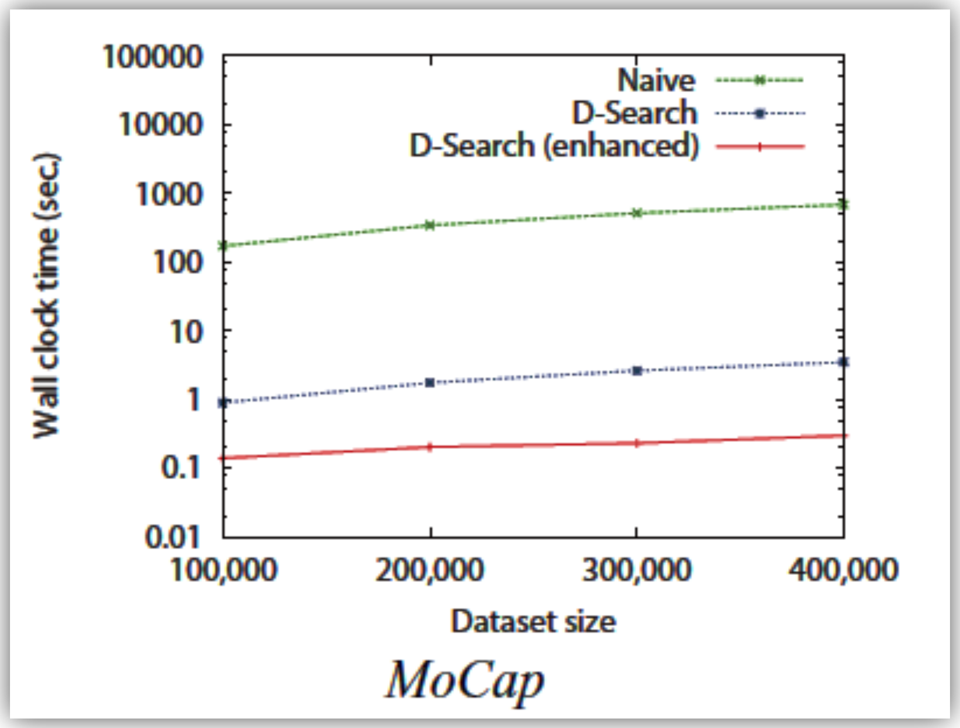
- D-Search can find similar purchasers groups
  - the songs of the same artist are identified as the same purchasers' group

Computation cost

- Compared with the naïve approach:
  - **Naïve**
    use all histogram buckets
  - **D-Search (basic)**
    use only selected buckets (largest values),
    and use the multi-step sequential scan
  - **D-Search (enhanced)**
    use the SVD coefficients,
    and use the multi-step sequential scan

Computation cost

- Compared with the naïve approach:

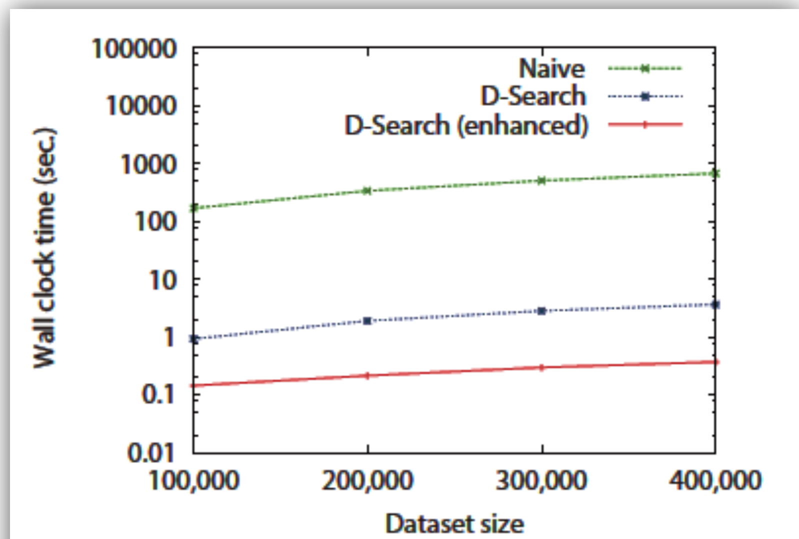


D-Search provides a reduction in computation time

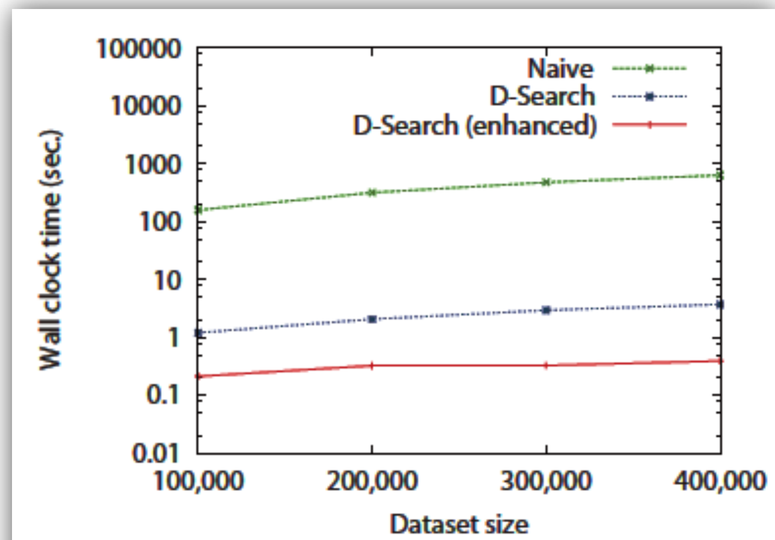
(up to 2,300 times faster than **Naïve**, 10 times faster than **D-Search (basic)**)

Computation cost

- Compared with the naïve approach:



EEG



Ondemand TV

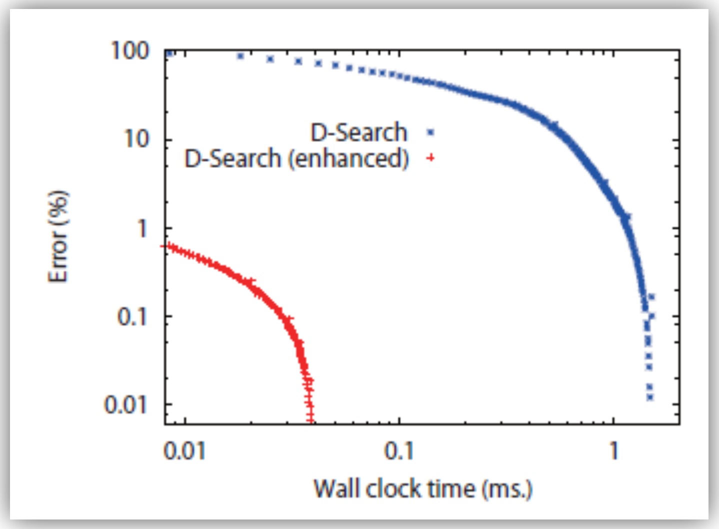
D-Search provides a reduction in computation time

(up to 2,300 times faster than **Naïve**, 10 times faster than **D-Search (basic)**)

Quality of proposed methods

Approximation Quality of SVD

- Trade-off between quality and cost



Ex. On demand TV

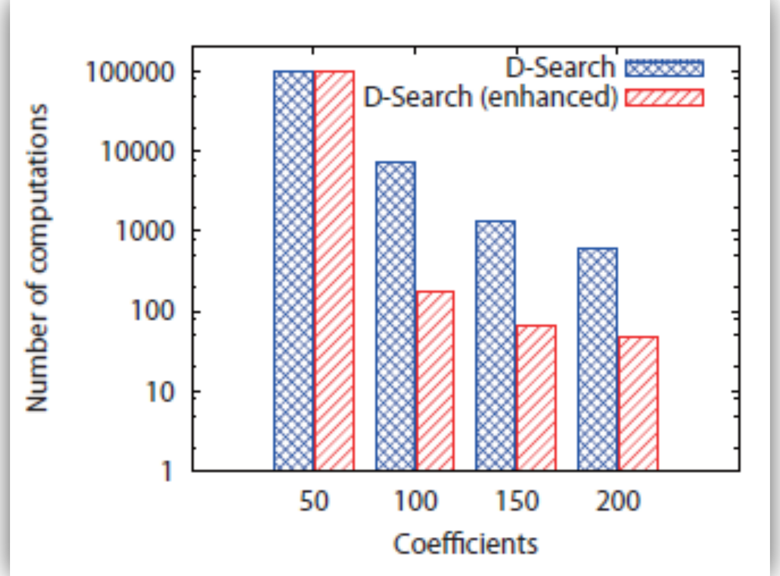
- Scatter plot of computation cost vs. approx. quality
- vary the number of c for each approx. technique

- SVD gives significantly lower approximation error, for the same computation time

Quality of proposed methods

Effect of the multi-step sequential scan

- How often each approximation was used?



Ex. On demand TV

D-Search efficiently prunes a large number of candidates, which leads to a significant reduction in the search cost

Outline

- Introduction
- Background
- D-Search
- Time-series distribution mining
- Experiments
- Conclusions

Conclusions

- Addressed the problem of distribution search
- Proposed a fast and effective method to solve it
  - Lower bounding KL divergence
  - Multi-step sequential scan
  - SVD-based approximate KL divergence
- Extended to time-series distribution mining
- Experiments show that our approach is faster than naïve implementation (up to 2,300 times)

Scalable Algorithms for Distribution Search

Yasuko Matsubara (Kyoto University)
Yasushi Sakurai (NTT Communication Science Labs)
Masatoshi Yoshikawa (Kyoto University)