Sequences and Time Series

Time Series Matching

K. Selçuk Candan, Professor of Computer Science and Engineering

CASCADE
ASU Center for Assured and SCAlable Data Engineering

Ira A. Fulton Schools of Engineering
Arizona State University

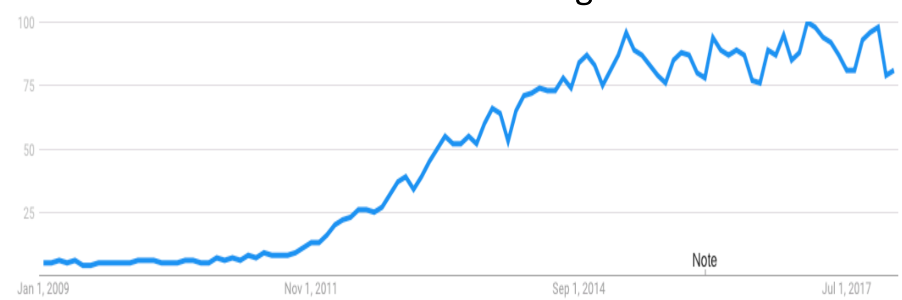
Strings, sequences, time series

- A *string* or *sequence*, $S = (c_1, c_2, \ldots, c_N)$, is a finite sequence of symbols.

abcbbbaabbaabcbbbaaabbc

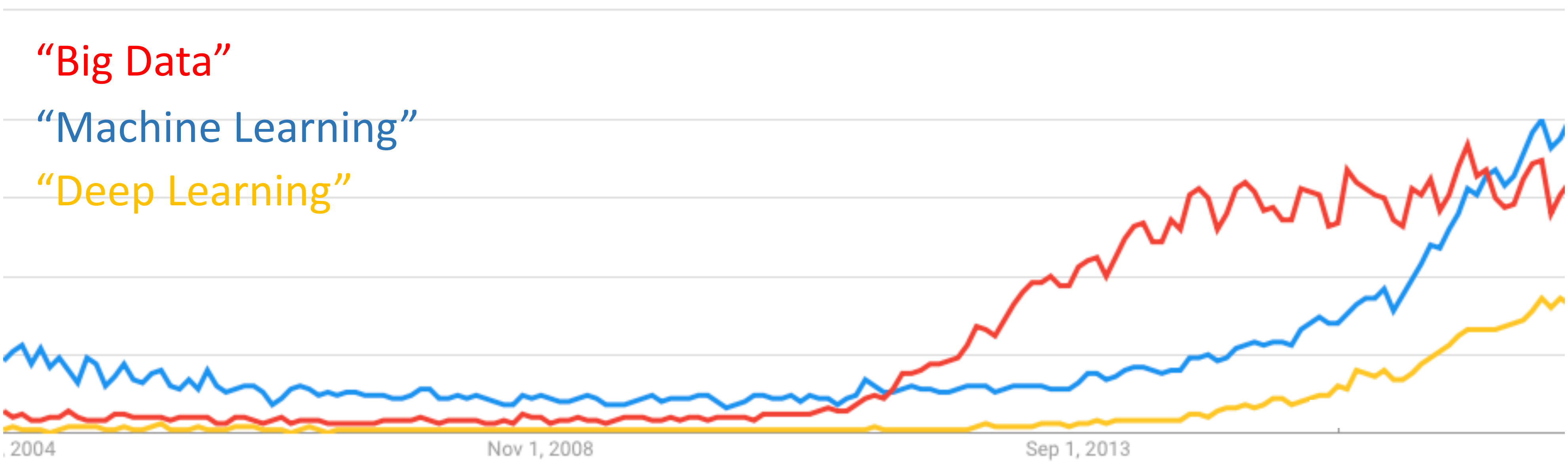
- A *time series*, $T = (d_1, d_2, \ldots, d_N)$, is a finite sequence of data values.

Interest in the term “big data”

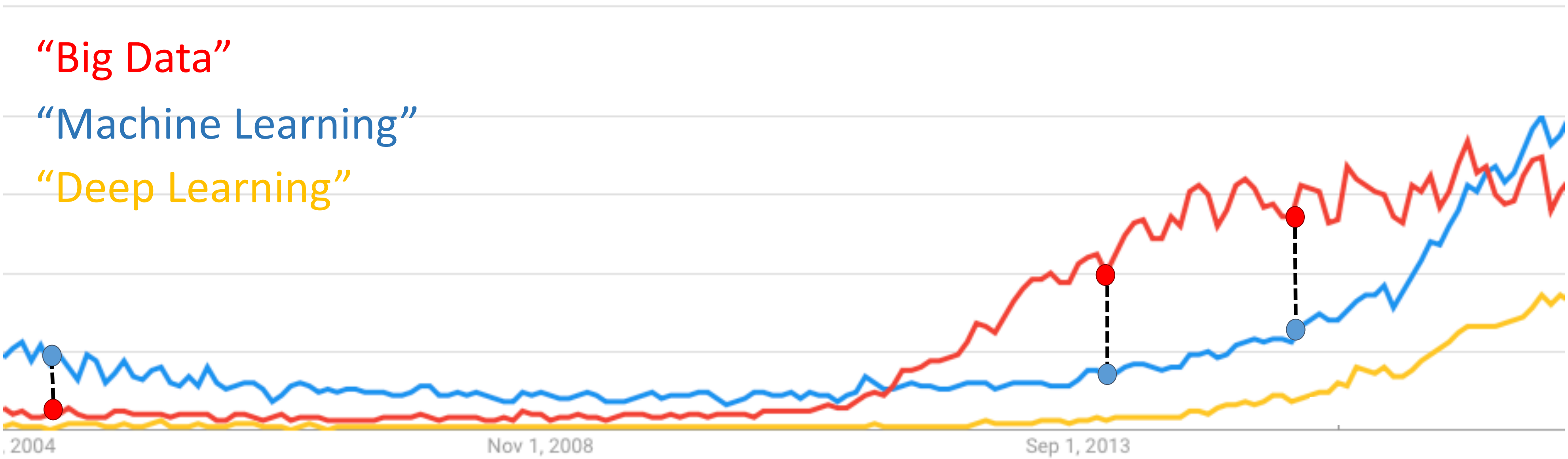


https://trends.google.com/trends/explore?date=2008-12-19%202018-01-19&q=big%20data

Comparing time series

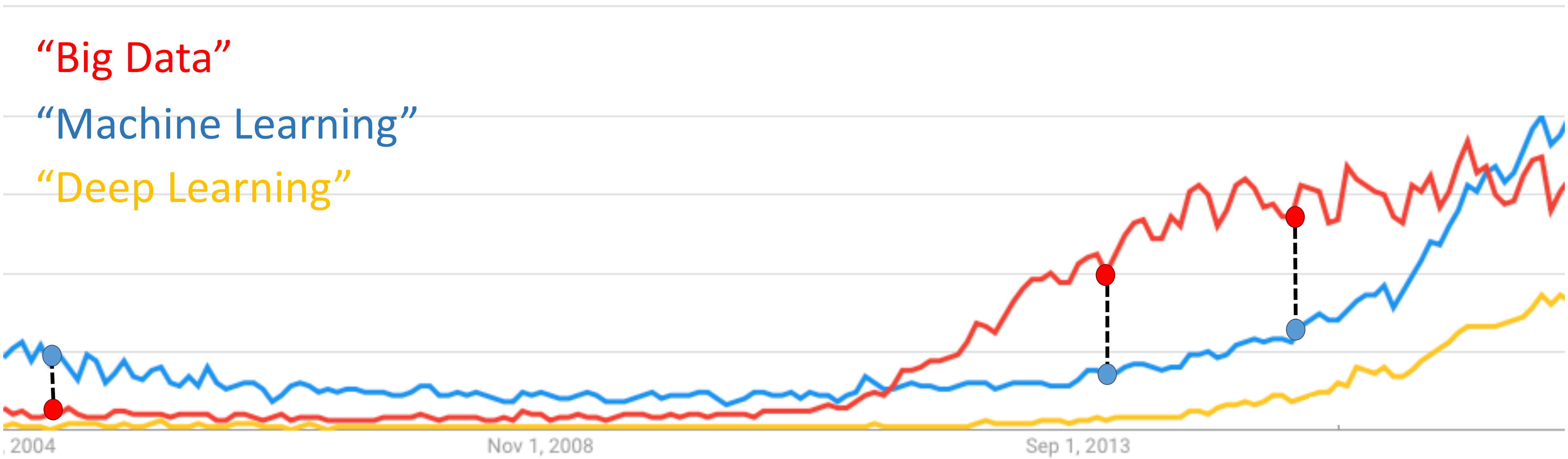


Euclidean Distance



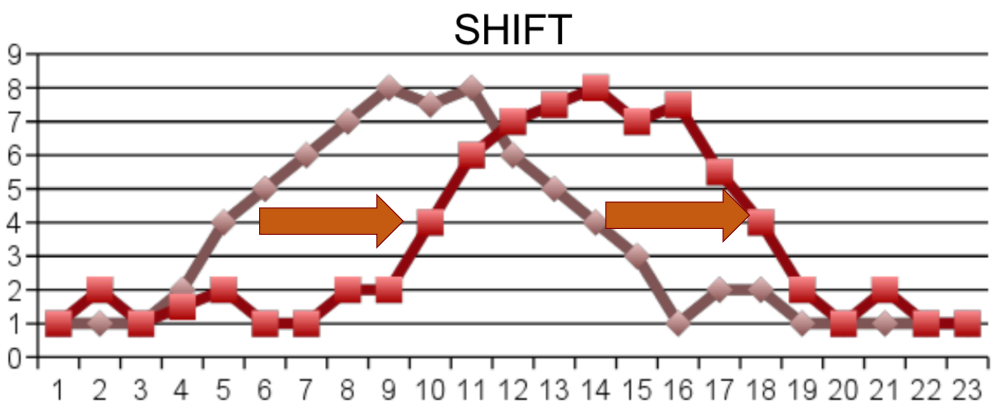
$$\Delta_{Euc}(B,M) = \sqrt{\sum_{i=1..N} \left(B_i - M_i\right)^2}$$

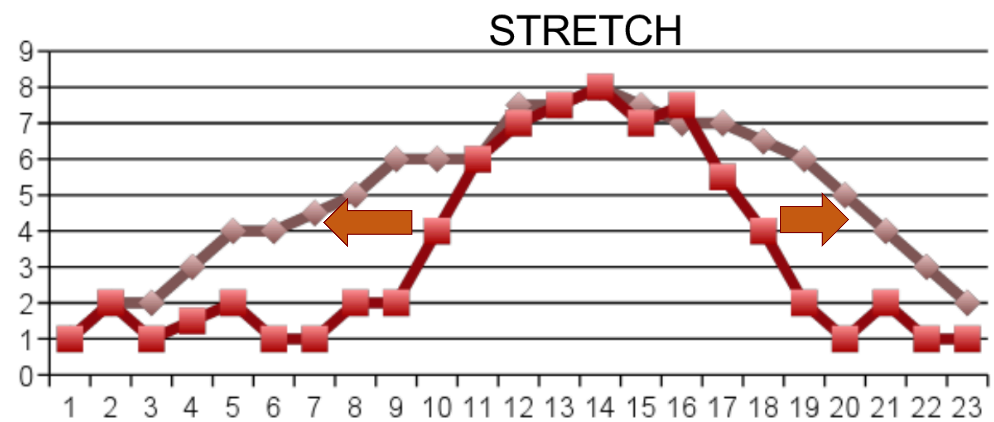
Correlation Similarity



$$\mathrm{Sim}_{correl}(B,M) = \frac{E\left[\left(B-\mu_B\right)\left(M-\mu_M\right)\right]}{\sigma_B \sigma_M}$$

Issues with Synchronized Measures





Reminder: Edit cost

- Let E be a sequence of edit operations to convert one string to another
- Let us associate a cost, C, to each edit operation

> - Costs of edit operations can be different from each other
>   - Type of the operation (replace, delete, insert)
>   - Symbols involved in the operation
>   - Position of the edit operation

- Given a sequence of edit operations, E

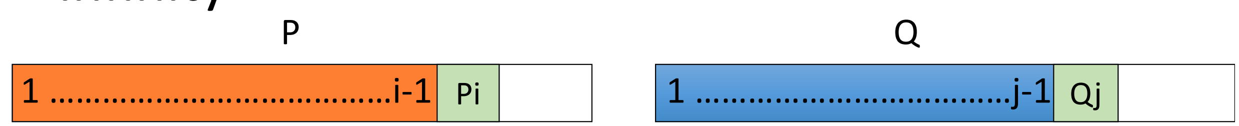
$$C(E) = \sum_{e_i \in E} C(e_i)$$

Reminder: Edit distance…

- Let us be given two strings, P and Q, of lengths N and M

D[i,j] = # of edits from length-i prefix of P to length-j prefix of Q

- D[-1,j] = infinity; D[i,-1] = infinity
- D[0,0] = 0



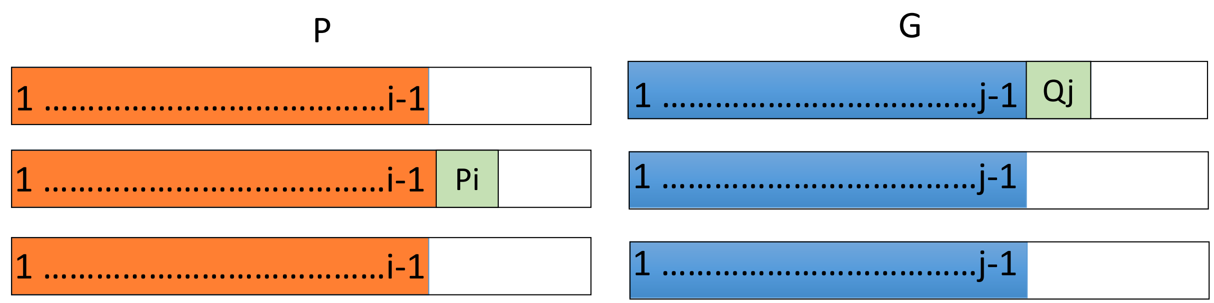
- if(P_i=Q_j) D[i,j] = D[i-1,j-1]

  else D[i,j] = min{

insert Qj $C_{ins}(Q_j)$ + D[i-1,j] ,

delete Pi $C_{del}(P_i)$ + D[i,j-1] ,

replace Pi with Qj $C_{rep}(P_{i,}\ Q_j)$ + D[i-1,j-1] }



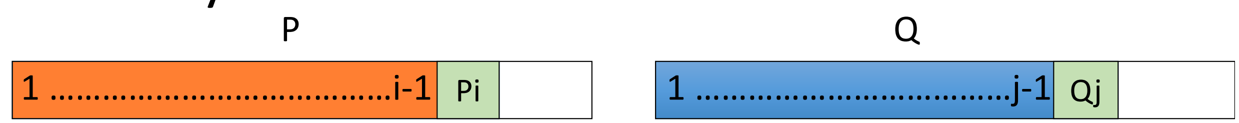
Reminder: Edit distance…

- Let us be given two time series, P and Q, of lengths N and M

D[i,j] = # of edits from length-i prefix of P to length-j prefix of Q

- D[-1,j] = infinity; D[i,-1] = infinity
- D[0,0] = 0
- if(P_i=Q_j) D[i,j] = D[i-1,j-1]

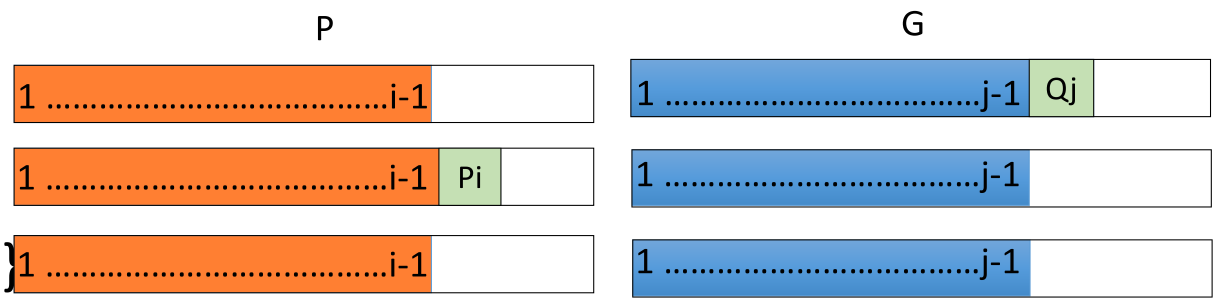
  else D[i,j] = min{



insert Qj $abs(Q_j)$ + D[i-1,j] ,

delete Pi $abs(P_i)$ + D[i,j-1] ,

replace Pi with Qj $abs(P_i - Q_j)$ + D[i-1,j-1] }



Dynamic Time Warping

- Let us be given two time series, P and Q, of lengths N and M

D[i,j] = # of edits from length-i prefix of P to length-j prefix of Q

- D[0,j] = infinity; D[i,0] = infinity
- D[0,0] = 0

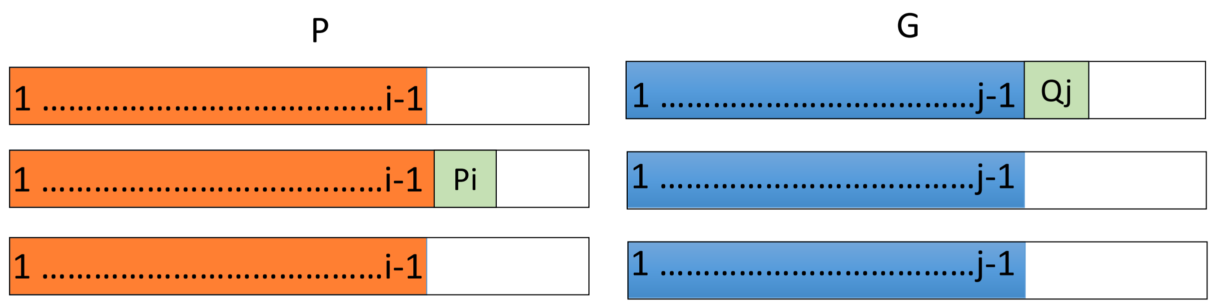
else D[i,j] = $abs(P_{i,} - Q_j)$ + min{

insertion D[i-1,j] ,

deletion D[i,j-1] ,

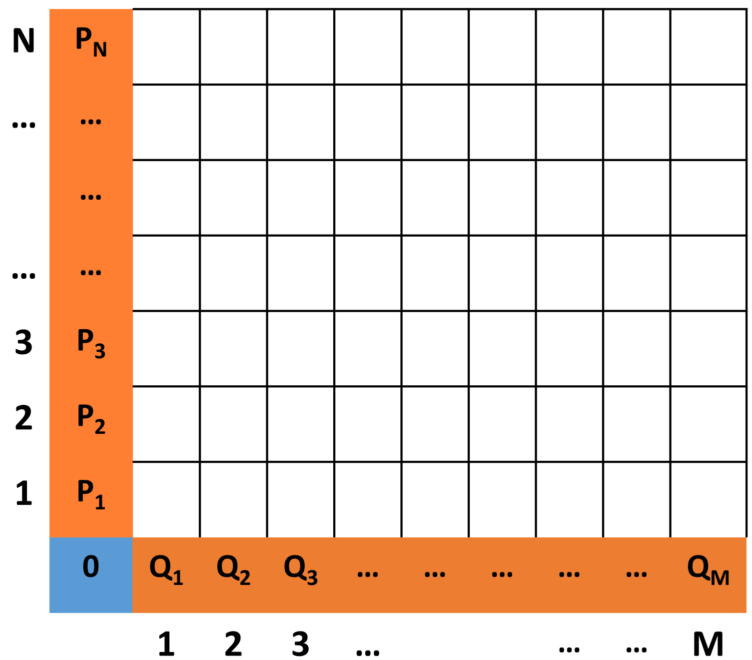
replacement D[i-1,j-1]

}



Dynamic Time Warping (DTW)

- Complexity: O(M,N)
- Dynamic Programming based implementation

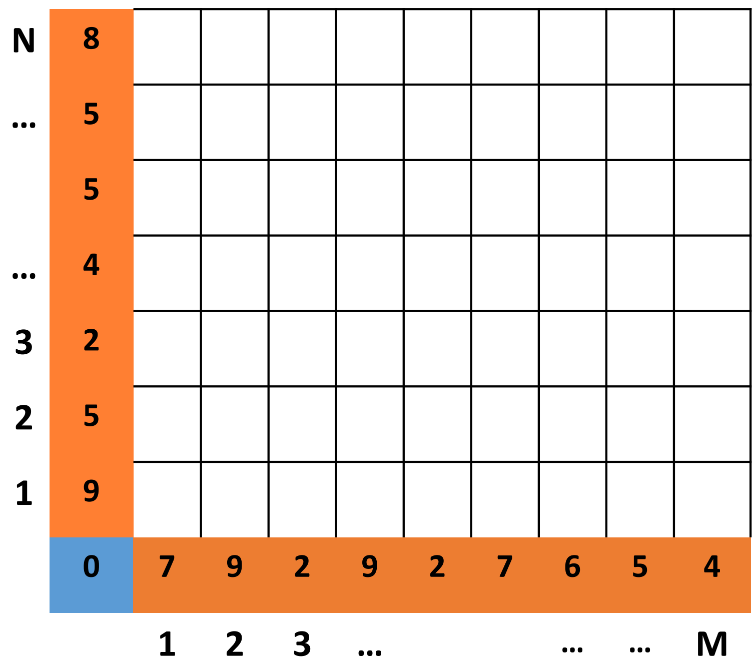


Dynamic Time Warping (DTW)

- Complexity: O(M,N)
- Dynamic Programming based implementation

7	9	2	9	2	7	6	5	4

9	5	2	4	5	5	8



Dynamic Time Warping (DTW)

7	9	2	9	2	7	6	5	4

9	5	2	4	5	5	8

- Complexity: O(M,N)
- Dynamic Programming based implementation

N	**8**	1	1	6	1	6	1	2	3	4
…	**5**	2	4	3	4	3	2	1	0	1
	5	2	4	3	4	3	2	1	0	1
…	**4**	3	5	2	5	2	3	2	1	0
3	**2**	5	7	0	6	0	5	4	3	2
2	**5**	2	4	3	4	3	2	1	0	1
1	**9**	2	0	7	0	7	2	3	4	5
	0	**7**	**9**	**2**	**9**	**2**	**7**	**6**	**5**	**4**
		1	**2**	**3**	**…**			**…**	**…**	**M**

Dynamic Time Warping (DTW)

- Complexity: O(M,N)
- Dynamic Programming based implementation

7	9	2	9	2	7	6	5	4

9	5	2	4	5	5	8

N	**8**	17	1	6	1	6	1	2	3	4
…	**5**	16	4	3	4	3	2	1	0	1
	5	14	4	3	4	3	2	1	0	1
…	**4**	12	5	2	5	2	3	2	1	0
3	**2**	9	7	0	6	0	5	4	3	2
2	**5**	4	4	3	4	3	2	1	0	1
1	**9**	2	2	9	9	16	18	21	25	30
	0	**7**	**9**	**2**	**9**	**2**	**7**	**6**	**5**	**4**
		1	**2**	**3**	**…**			**…**	**…**	**M**

Dynamic Time Warping (DTW)

- Complexity: O(M,N)
- Dynamic Programming based implementation

7	9	2	9	2	7	6	5	4

9	5	2	4	5	5	8

N	**8**	17	1	6	1	6	1	2	3	4
…	**5**	16	4	3	4	3	2	1	0	1
	5	14	4	3	4	3	2	1	0	1
…	**4**	12	5	2	5	2	3	2	1	0
3	**2**	9	7	0	6	0	5	4	3	2
2	**5**	4	6	3	4	3	2	1	0	1
1	**9**	2	2	9	9	16	18	21	25	30
	0	**7**	**9**	**2**	**9**	**2**	**7**	**6**	**5**	**4**
		1	**2**	**3**	**…**			**…**	**…**	**M**

Dynamic Time Warping (DTW)

- Complexity: O(M,N)
- Dynamic Programming based implementation

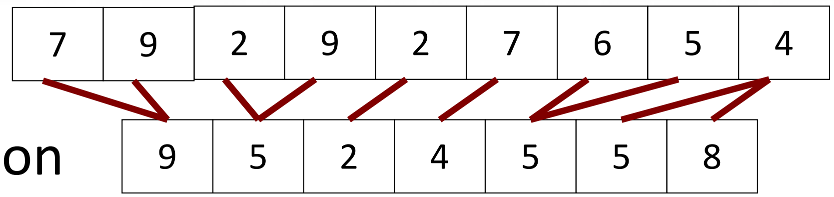
7	9	2	9	2	7	6	5	4

9	5	2	4	5	5	8

N	**8**	17	17	19	14	20	15	16	17	18
…	**5**	16	18	13	14	14	15	14	14	14
	5	14	16	10	11	13	13	13	13	14
…	**4**	12	14	7	10	11	12	14	15	15
3	**2**	9	11	5	11	9	14	18	18	20
2	**5**	4	6	5	9	12	14	15	15	16
1	**9**	2	2	9	9	16	18	21	25	30
	0	**7**	**9**	**2**	**9**	**2**	**7**	**6**	**5**	**4**
		1	**2**	**3**	**…**			**…**	**…**	**M**

Dynamic Time Warping (DTW)

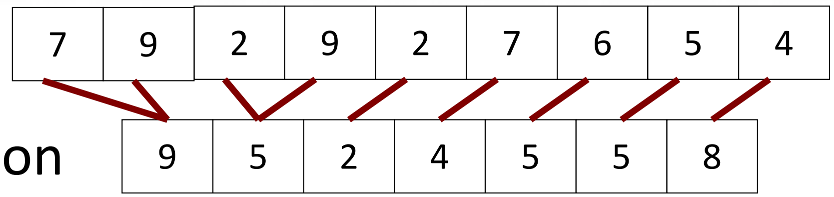
- Complexity: O(M,N)
- Dynamic Programming based implementation



N	**8**	17	17	19	14	20	15	16	17	18
…	**5**	16	18	13	14	14	15	14	14	14
	5	14	16	10	11	13	13	13	13	14
…	**4**	12	14	7	10	11	12	14	15	15
3	**2**	9	11	5	11	9	14	18	18	20
2	**5**	4	6	5	9	12	14	15	15	16
1	**9**	2	2	9	9	16	18	21	25	30
	0	**7**	**9**	**2**	**9**	**2**	**7**	**6**	**5**	**4**
		1	**2**	**3**	**…**			**…**	**…**	**M**

Dynamic Time Warping (DTW)

- Complexity: O(M,N)
- Dynamic Programming based implementation

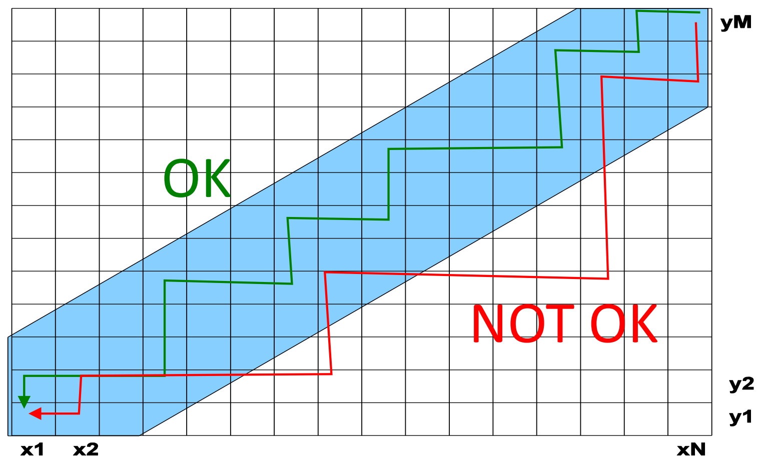


N	**8**	17	17	19	14	20	15	16	17	18
...	**5**	16	18	13	14	14	15	14	14	14
	5	14	16	10	11	13	13	13	13	14
...	**4**	12	14	7	10	11	12	14	15	15
3	**2**	9	11	5	11	9	14	18	18	20
2	**5**	4	6	5	9	12	14	15	15	16
1	**9**	2	2	9	9	16	18	21	25	30
	0	**7**	**9**	**2**	**9**	**2**	**7**	**6**	**5**	**4**
		1	**2**	**3**	**...**			**...**	**...**	**M**

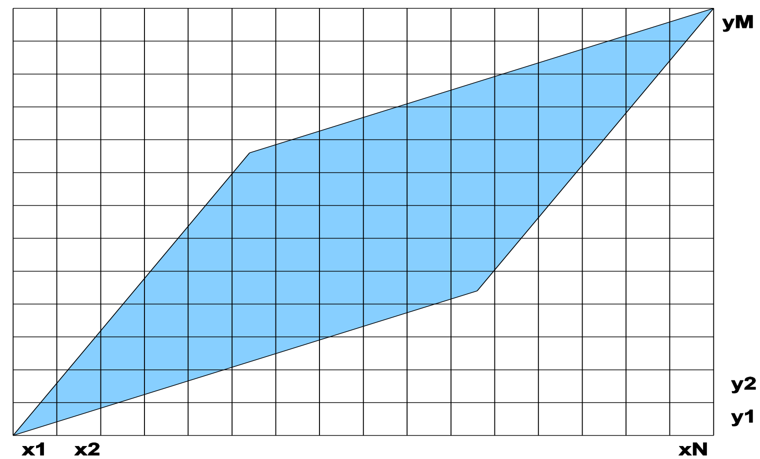
Reducing the Cost of DTW

To reduce the O(NM) cost of filling the grid various heuristics impose constraints on the grid regions through which the warp paths can pass.

Sakoe-Chiba band [1]



Itakura parallelogram [2]

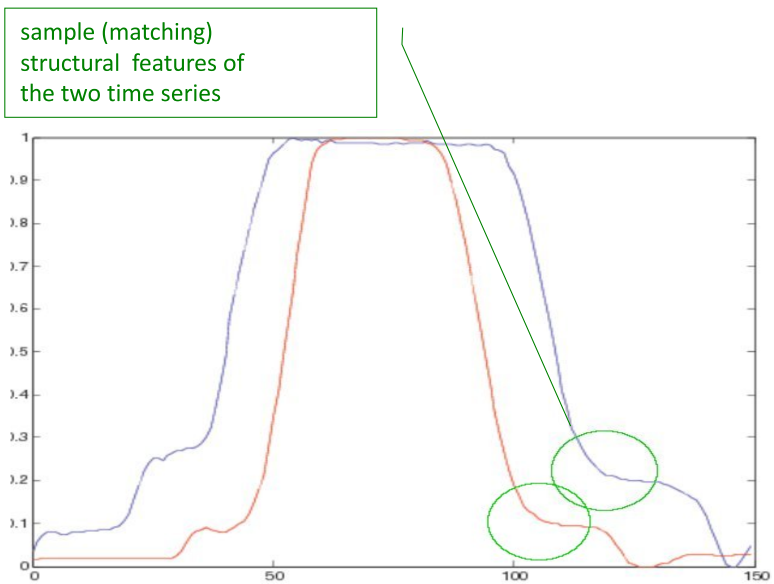


[1] Dynamic Programming Algorithm Optimisation for Spoken Word Recognition, 1978
[2] F. Itakura. Minimum prediction residual principle applied to speech recognition, 1975

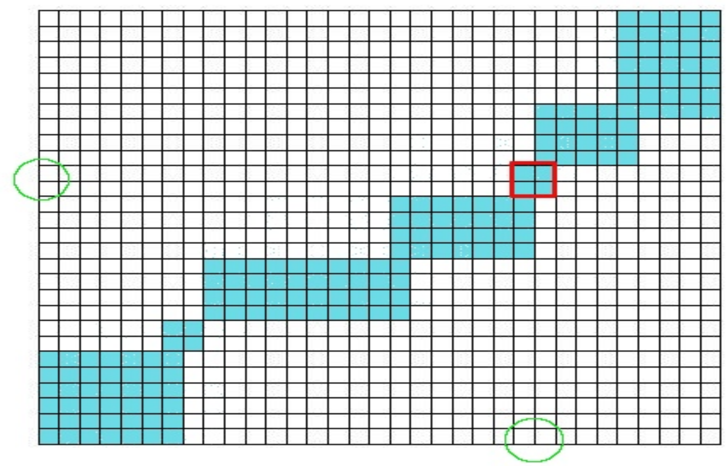
sDTW

K.S. Candan, R. Rossini, X. Wang, M.L. Sapino, sDTW: computing DTW distances using locally relevant constraints based on salient feature alignments, Proc. VLDB Endowment 5 (11) (2012) 1519–1530.

Time series often carry temporal features that can be used for identifying locally relevant constraints to eliminate redundant work in an adaptive manner.



Two time series


Adaptive constraints on the DTW grid

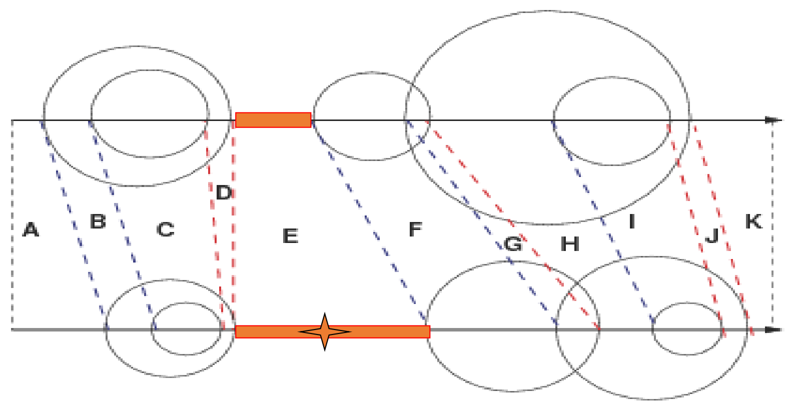
Overview of the sDTW Process

- **Step 1:** Search for salient temporal features of the input time series.

- **Step 2:** Find consistent alignments of a given pair of time series by matching the "descriptors" of the salient features.

- **Step 3:** Use these alignments to compute locally relevant constraints to prune the warp path search.

Step 3: Width Adaptation

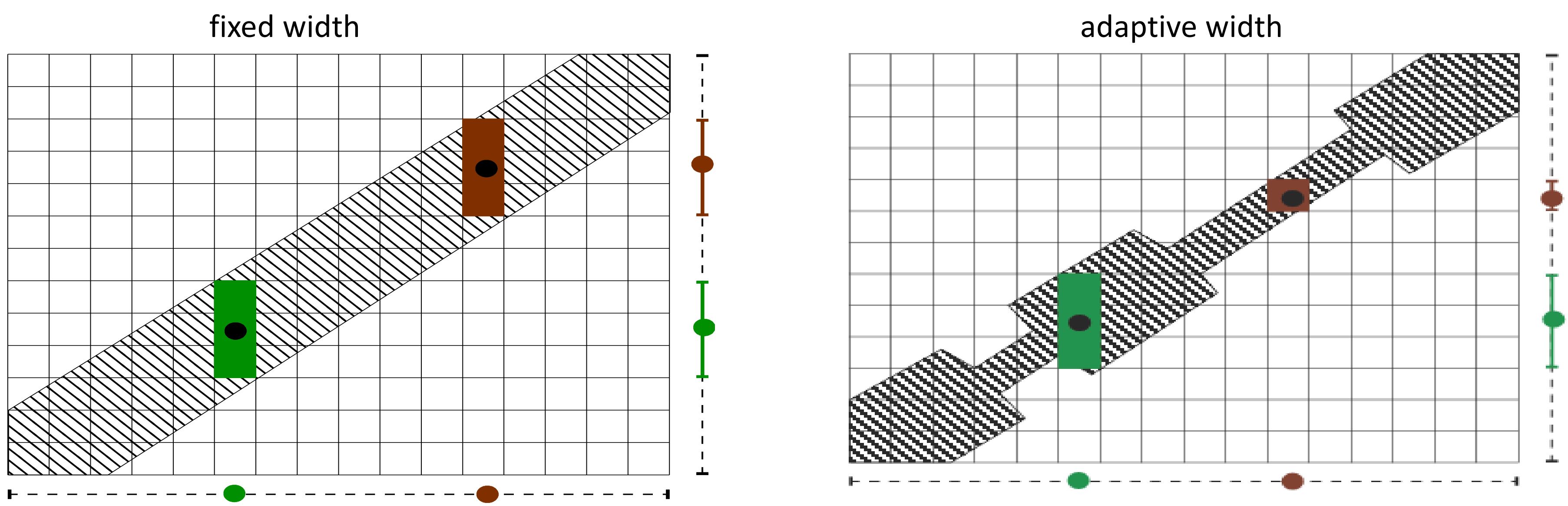


Consistently aligned features partition two time series into intervals

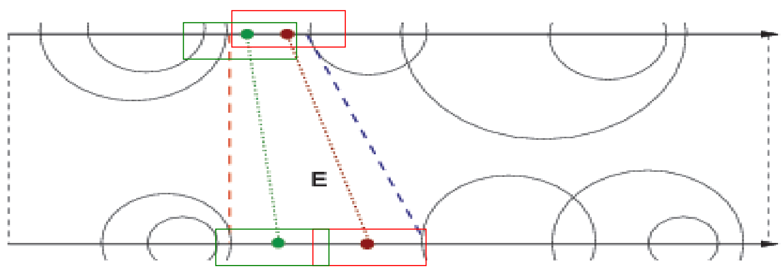
For each time instance, the width of the DTW band can be adapted based on the lengths of the corresponding intervals

Adaptive Width Constraints

Adaptive width constraints use the widths of the resulting intervals to choose a different locally relevant width for each time instance



Step 3: Core Adaptation

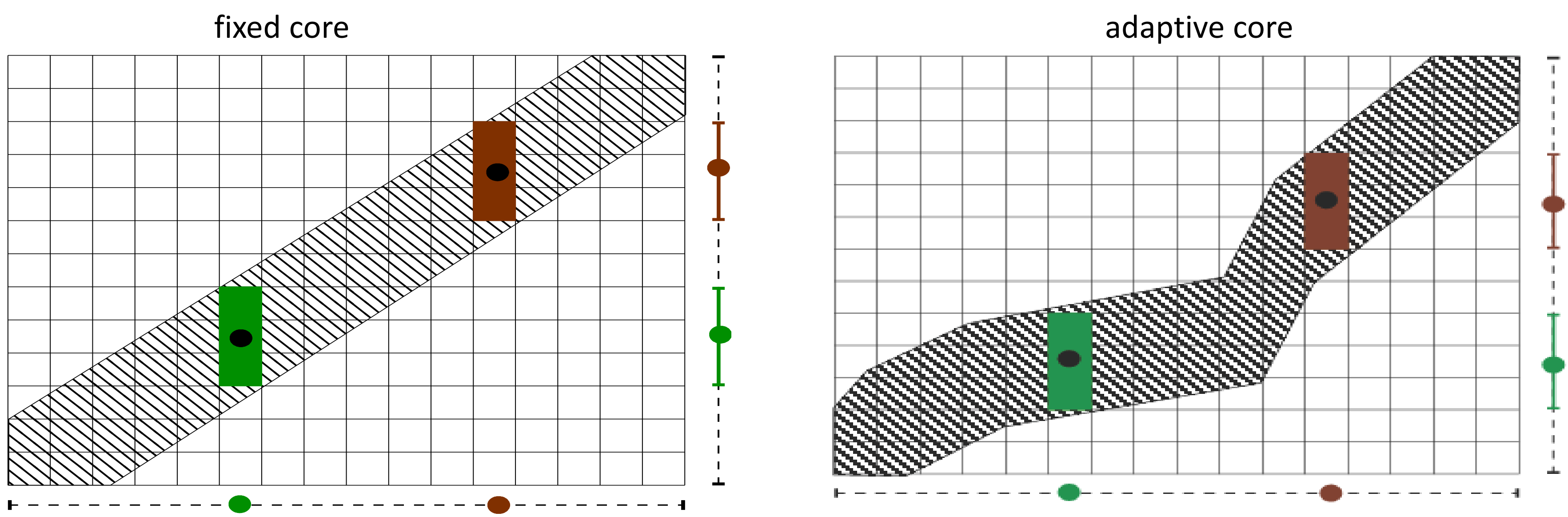


Each point on one time series has a roughly corresponding point on the other time series.

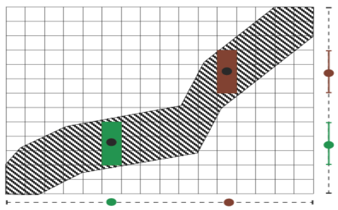
Thus, we can center the search band around these candidate points.

Adaptive Core Constraints

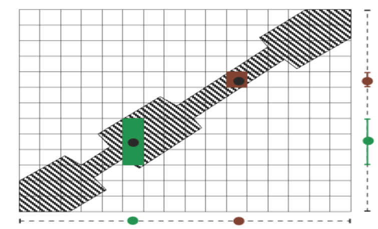
The core follows a path that reflects the candidate alignments implied by the salient features.



Adaptive Core & Adaptive Width Constraints

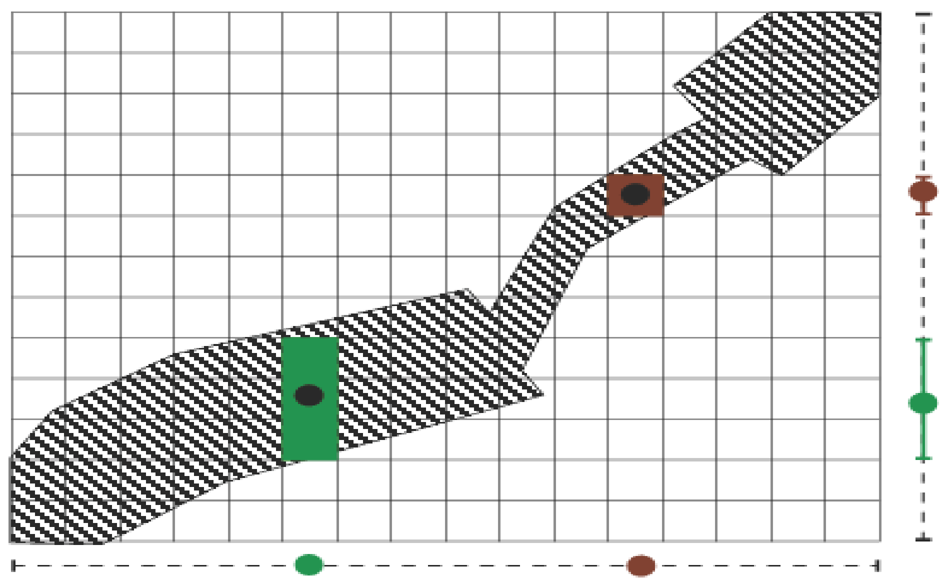


adaptive core



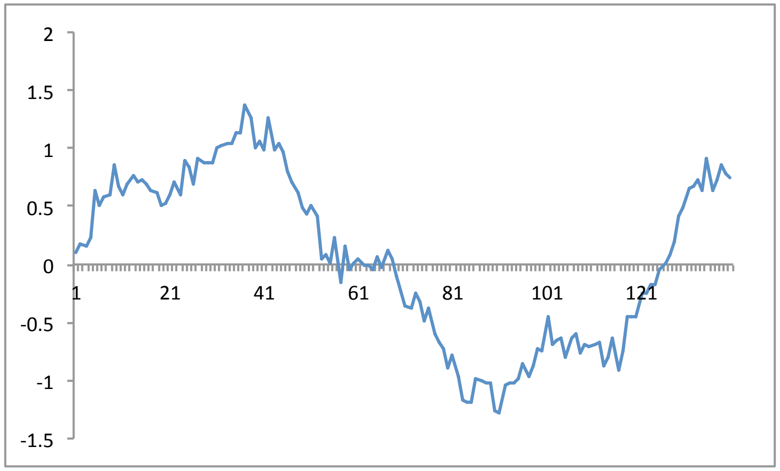
adaptive width

adaptive core and adaptive width



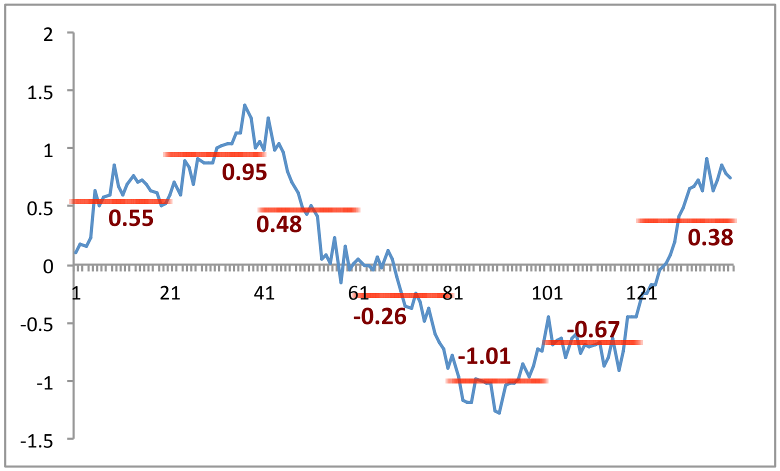
SAX (Symbolic Aggregate ApproXimation)

- Time series are similar to sequences



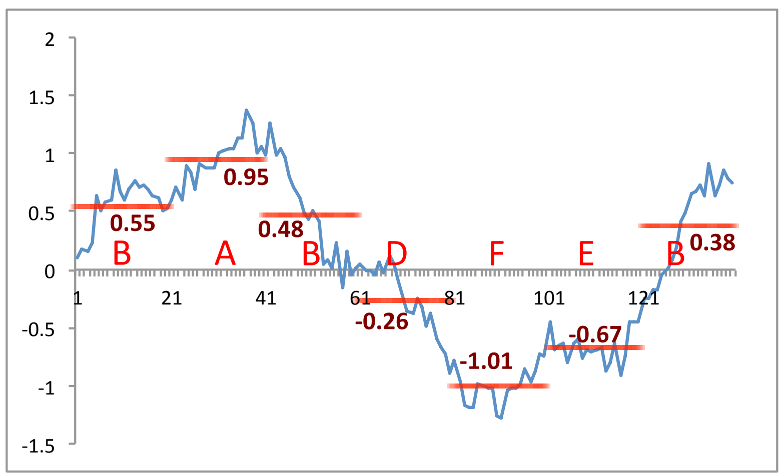
SAX (Symbolic Aggregate ApproXimation)

- Time series are similar to sequences
- Transform a time series into a compact sequence representation
  - Divide the time series into w-length (non-overlapping) windows
  - For each window,
    - compute the average amplitude



SAX (Symbolic Aggregate ApproXimation)

- Time series are similar to sequences
- Transform a time series into a <u>compact</u> sequence representation
  - Divide the time series into w-length (non-overlapping) windows
  - For each window,
    - compute the average amplitude
    - assign a symbol from a dictionary with s symbols representing this average amplitude



Range	Symbol
[0.9,2]	A
[0.3,0.9)	B
[0.0,0.3)	C
(0.0, -0.3)	D
(-0.3,-0.9]	E
(-0.9,-2]	F

SAX (Symbolic Aggregate ApproXimation)

- Time series are similar to sequences
- Transform a time series into a <u>compact</u> sequence representation
  - Divide the time series into w-length (non-overlapping) windows
  - For each window,
    - compute the average amplitude
    - assign a symbol from a dictionary with s symbols representing this average amplitude

- Note that, SAX reduces
  - temporal resolution, by dividing the string into windows (length ~ N/w)
  - amplitude resolution, by using only one of the s symbols per window