Lecture 09: Dynamic Time Warping for isolated word recognition

ELEC747 Speech Signal Processing
Gil-Jin Jang
Original slides from:
Elena Tsiporkova, Flanders Institute for Biotechnology / University of Gent
Bhiksha Raj, CMU
https://www.cs.cmu.edu/~bhiksha/courses/11-756.asr/spring2013/
Roger Jang, MIR Lab, Tsing Hua Univ.
http://www.cs.nthu.edu.tw/~jang

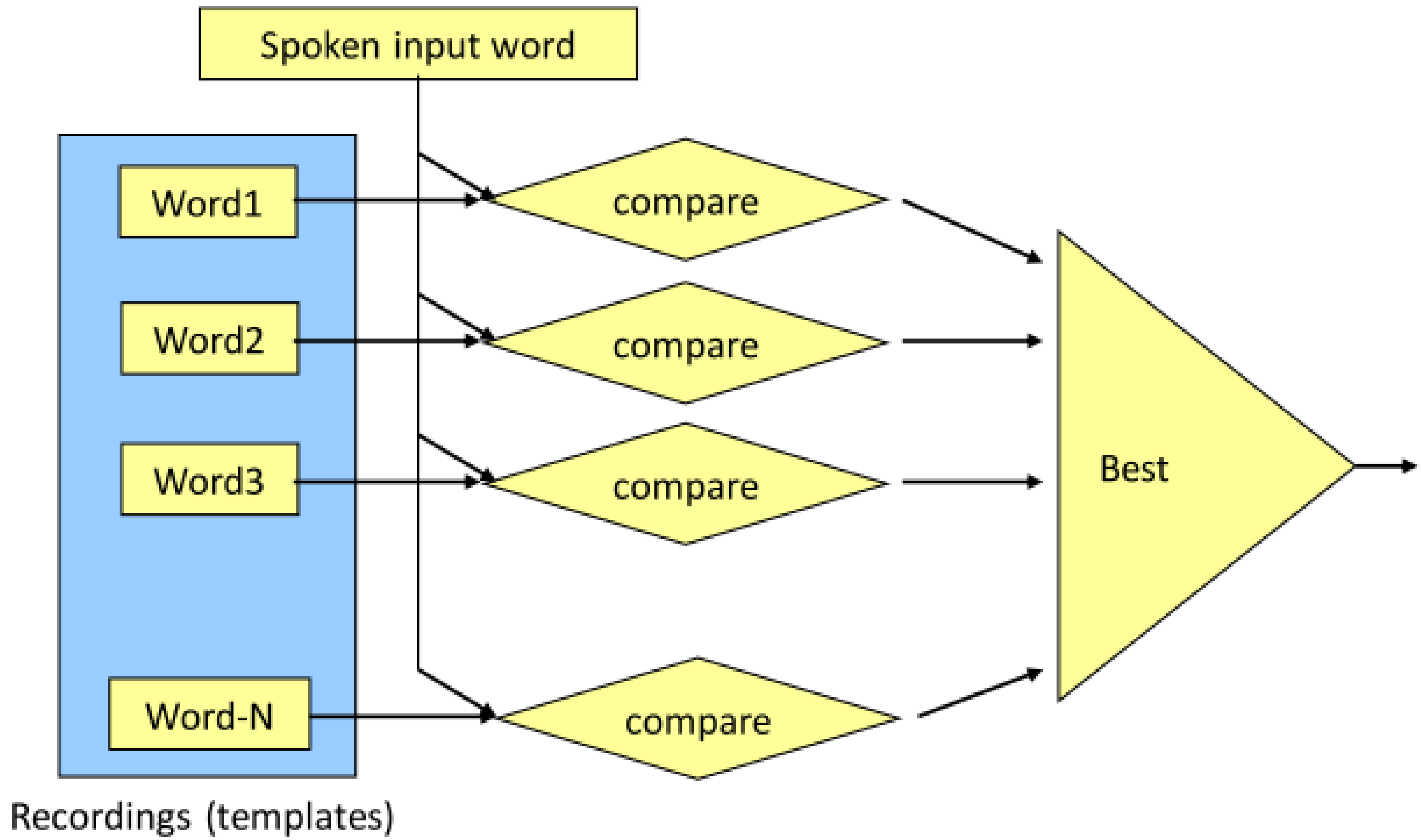
Topics

- Isolated word recognition
  - Template matching approach
- Dynamic time warping (DTW)
  - Why time warping is necessary
  - Basic algorithm definition
  - Local constraints
  - Global constraints
- Practical issues of DTW

Isolated word recognition (IWR)

- The distance or similarity of the short-time segments (frames) of speech signals can be defined by:
  - Euclidean distance between the frame features such as MFCC (mel-frequency cepstral coefficients)
- Given a problem of finding a word that are closest to the input speech signals, we may approach
  - Store a template (prototype) of each enrolled word
  - Compute distance between the template and input speech
  - Find the word of the template with minimum distance
- Issues
  - How to define the distance between the template and input speech signals with different lengths?
  - In other words, how to **WARP** one to the other?

IWR: template matching concepts



IWR: math

- Mathematical formulation of word recognition:

$$
\begin{aligned}
w^* &= \arg\min_w D(\mathbf{X}_i, \mathbf{X}_w) && \text{By distance} \\
&= \arg\max_w S(\mathbf{X}_i, \mathbf{X}_w) && \text{By similarity} \\
&= \arg\max_w P(\mathbf{X}_i, \mathbf{X}_w) && \text{By probability}
\end{aligned}
$$

$\mathbf{X}_i$: feature matrix of input speech

$\mathbf{X}_w$: template feature matrix of word w

- The similarity measure can be defined by inverse of the distance.
- Once a distance measure is given, one simple probability transformation can be obtained by $\exp(-\lambda D)$ with a proper choice of positive constant λ.
- "Distance" should be positive, so $0 \le \exp(-\lambda D) \le 1$, which satisfies the axiom of probability function. Sometimes, the probability function should be scaled to satisfy the condition that its integral over all real values should equal to 1.

- We can compute frame distance $d(\mathbf{x}_i(t_1), \mathbf{x}_w(t_2))$ by the Euclidean distance, but how to compute $D(\mathbf{X}_i, \mathbf{X}_w)$ from frame distance?
- Mere summation won't work
- $D(\mathbf{X}_i, \mathbf{X}_w) \neq \sum_{t_1=1}^{T} d(\mathbf{x}_i(t_1), \mathbf{x}_w(t_2))$

Metric Distances

- What properties should a distance have?
  - D(A,B) = D(B,A) Symmetry
  - D(A,A) = 0 Zero self-distance
  - D(A,B) >= 0 Positivity
  - D(A,B) ≤ D(A,C) + D(B,C) Triangular Inequality

- Generalized vector distance (norm)

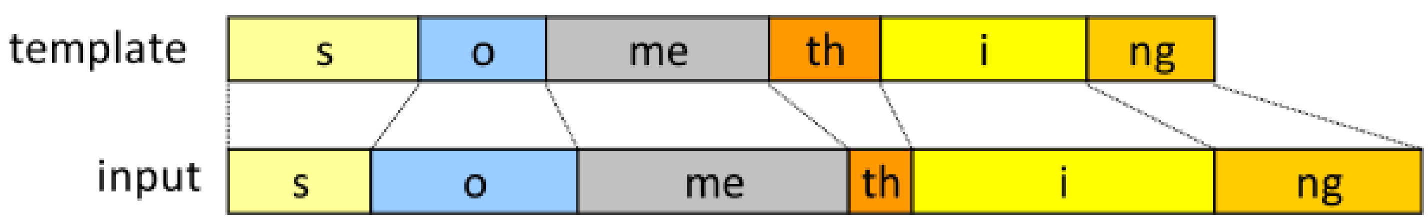
$$\boxed{L_q = \left(\sum_{k=1}^{n} |x_k - y_k|^q\right)^{1/q}}$$

q=1 Manhattan distance
q=2 Euclidean distance

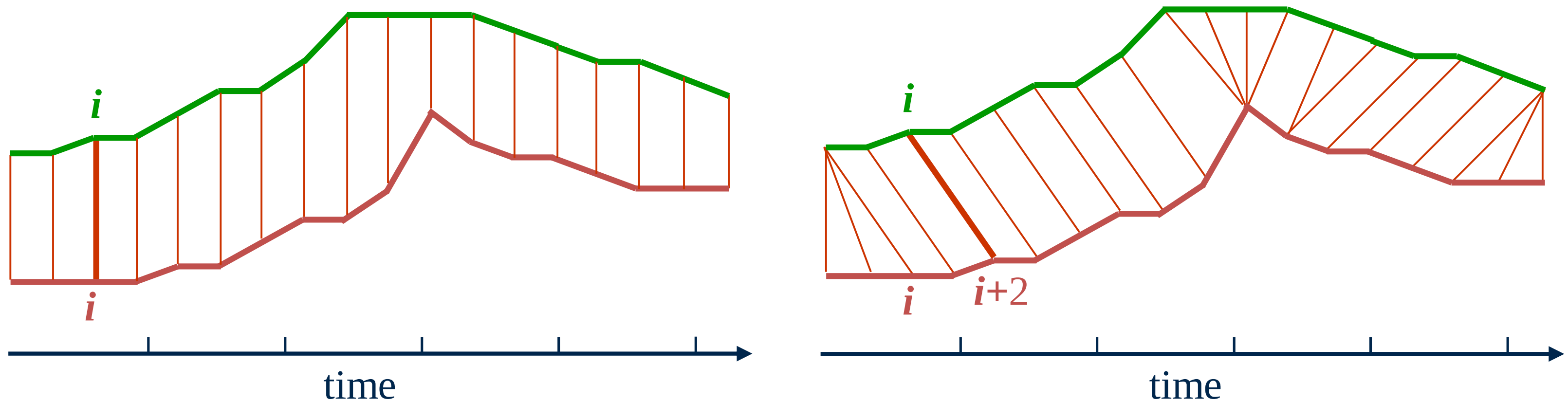
q=∞ Max magnitude

Template Matching

- Problems:
  - Input and template may be of different lengths
  - The change in matching lengths may not be uniform
    - Non-linear / non-uniform matching
  - Best alignment should be found to compute the optimal distance between two speech signals



Why Time Warping?

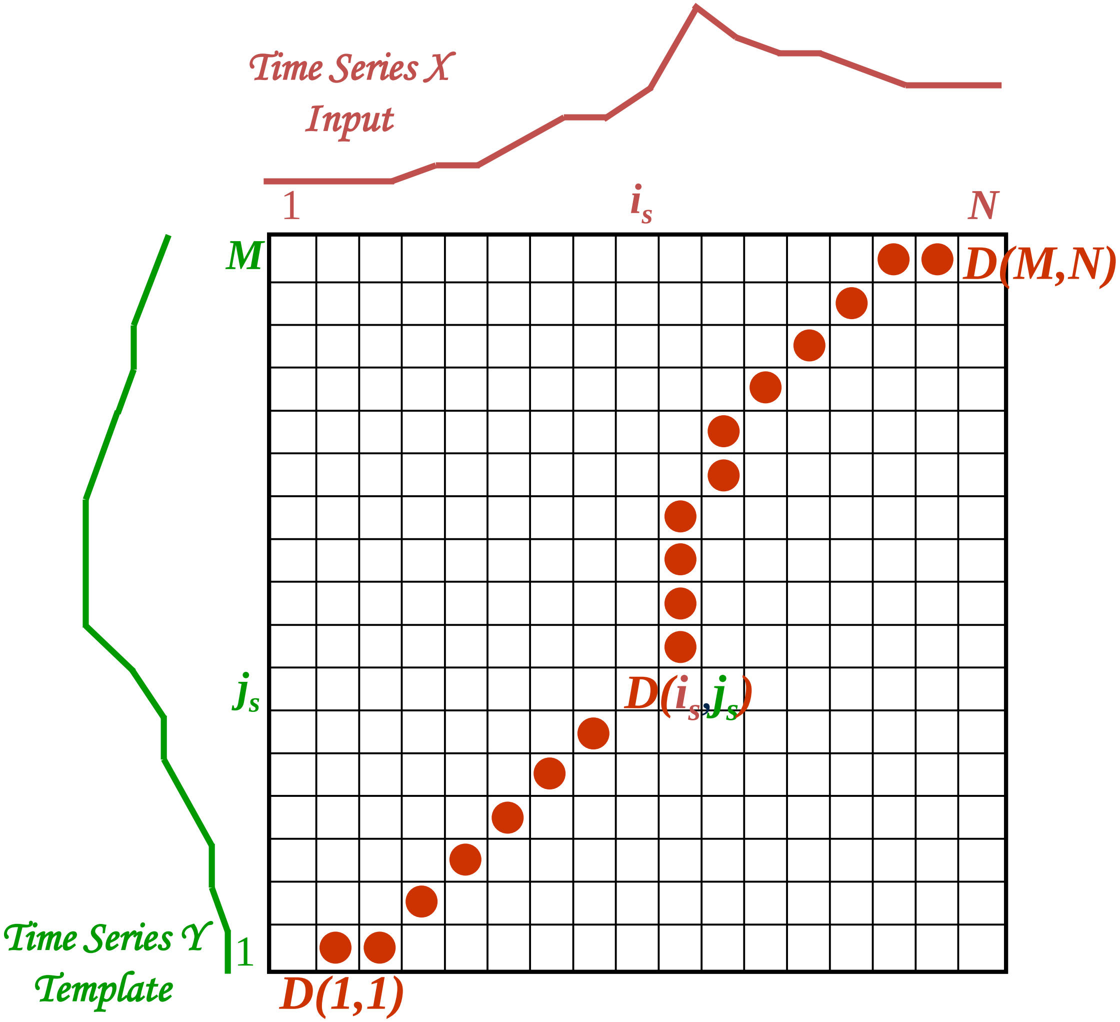


Any distance (Euclidean, Manhattan, …) which aligns the *i*-th point on one time series with the *i*-th point on the other will produce a poor similarity score.

A non-linear (elastic) alignment produces a more intuitive similarity measure, allowing similar shapes to match even if they are out of phase in the time axis.

- In naïve implementation, finding the optimal warping path (alignment) is $O(M^N)$, where M and N are the lengths of the two time sequences
- The exponential function grows rapidly with M and N, so it is inapplicable in real situations

Formulation of DTW algorithm



Let $D(i,j)$ refer to the dynamic time warping distance between the subsequences

$$x_1, x_2, \ldots, x_i$$
$$y_1, y_2, \ldots, y_j$$

It is calculated by the following recurrence relation:

$$D(i,j) = d(\mathbf{x}_i, \mathbf{y}_j) + \min\left\{\begin{matrix} D(i,j-1), \\ D(i-1,j-1), \\ D(i-1,j) \end{matrix}\right\}$$

$D(\mathbf{X},\mathbf{Y})$ is then $D(N,M)$

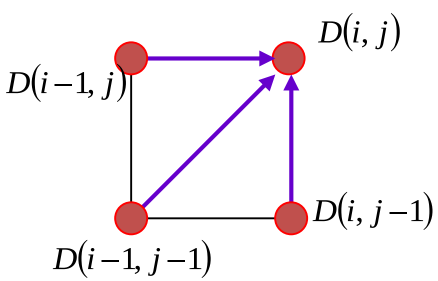
➔ Filling in NxM matrix

➔ O(NM)

- In most other matrix notations are row-major, that is, row index first.
- Don’t know why
- However, most DTW notations use column-major, which makes implementation with MATLAB quite confusing.

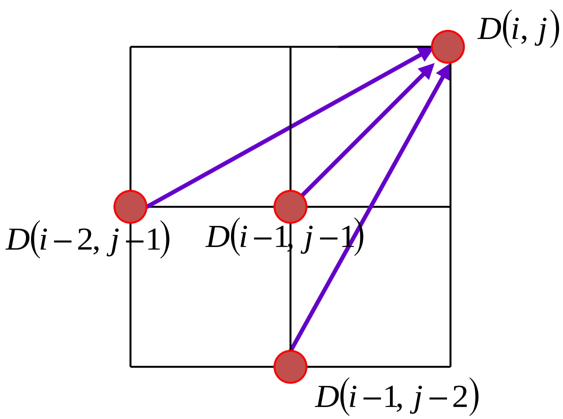
Local Transition Constraints

- Type 1 (Levenshtein)
  - 0-45-90 local paths



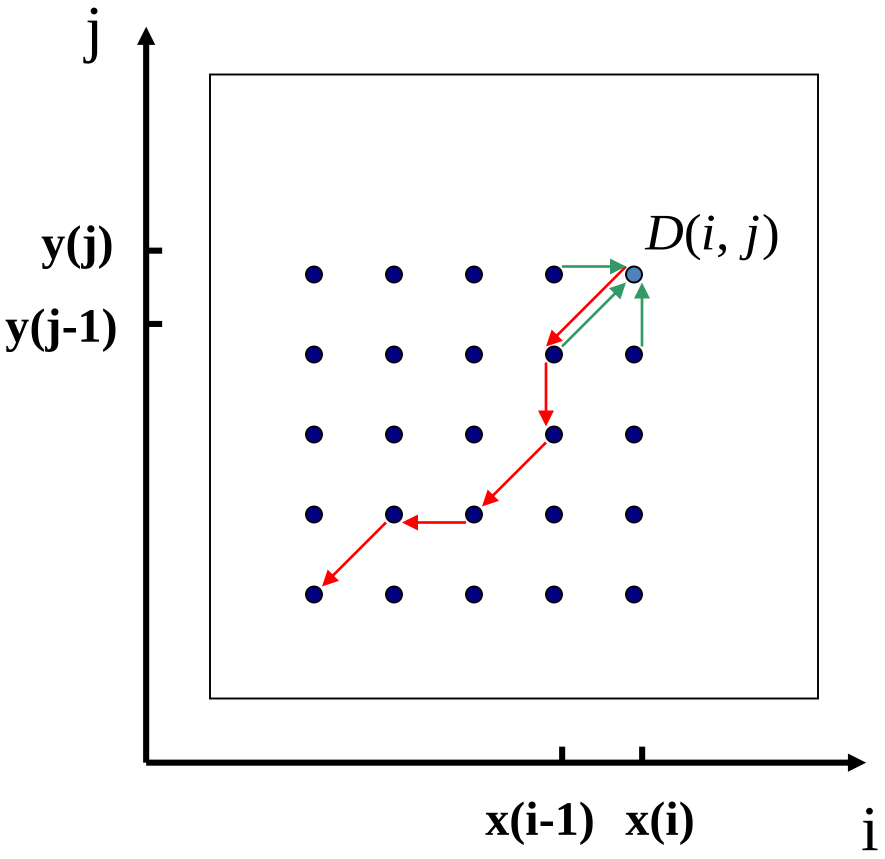
$$D(i, j) = \left\| \mathbf{x}(i) - \mathbf{y}(j) \right\| + \min \begin{Bmatrix} D(i, j-1) \\ D(i-1, j-1) \\ D(i-1, j) \end{Bmatrix}$$

- Type 2
  - 27-45-63 local paths



$$D(i, j) = \left\| \mathbf{x}(i) - \mathbf{y}(j) \right\| + \min \begin{Bmatrix} D(i-1, j-2) \\ D(i-1, j-1) \\ D(i-2, j-1) \end{Bmatrix}$$

Dynamic Time Warping: Type 1

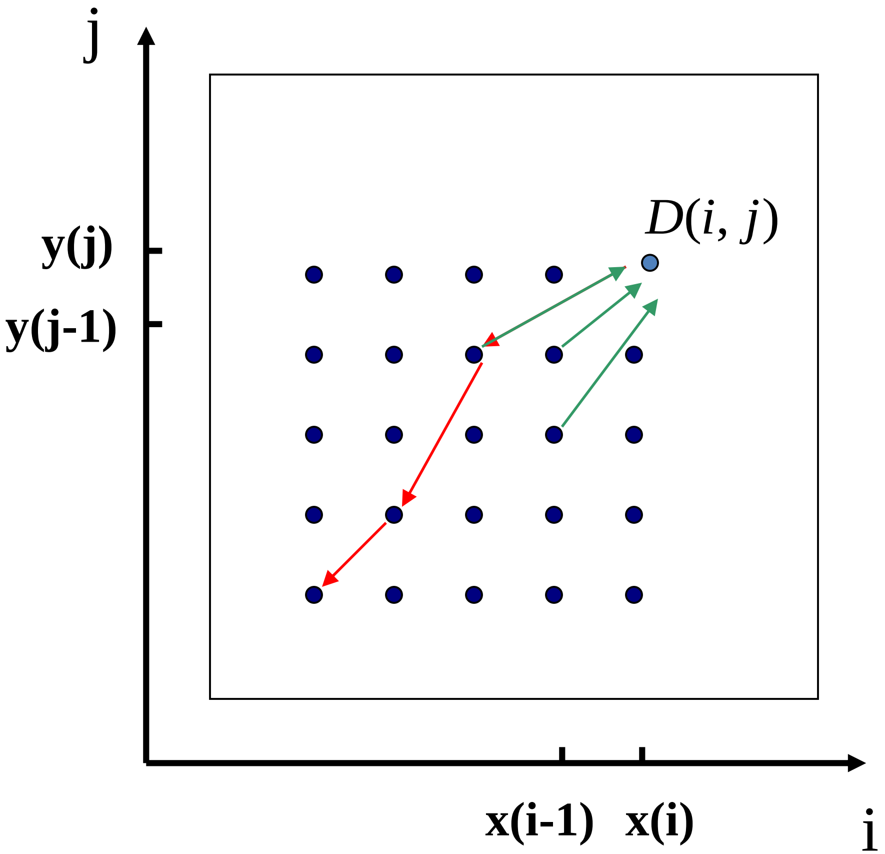


t: input MFCC matrix
(Each row is a frame's feature.)
r: reference MFCC matrix
Local paths: 0-45-90 degrees

DTW recurrence:

$$D(i,j) = \left\| \mathbf{x}(i) - \mathbf{y}(j) \right\| + \min\left\{ \begin{array}{c} D(i, j-1) \\ D(i-1, j-1) \\ D(i-1, j) \end{array} \right\}$$

Dynamic Time Warping: Type 2



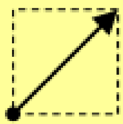
t: input MFCC matrix
(Each column is a frame's feature.)
r: reference MFCC matrix
Local paths: 27-45-63 degrees

DTW recurrence:

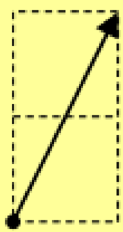
$$D(i,j) = \|\mathbf{x}(i) - \mathbf{y}(j)\| + \min\begin{Bmatrix} D(i-1, j-2) \\ D(i-1, j-1) \\ D(i-2, j-1) \end{Bmatrix}$$

Local Transition Constraints: Type 3

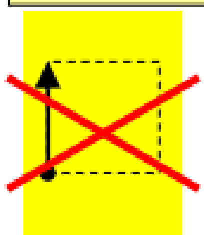
The next input frame aligns to the same template frame as the previous one. (Allows a template segment to be arbitrarily stretched to match some input segment)



The next input frame aligns to the next template frame. No stretching or shrinking occurs in this region



The next input frame skips the next template frame and aligns to the one after that. Allows a template segment to be shrunk (by at most ½) to match some input segment

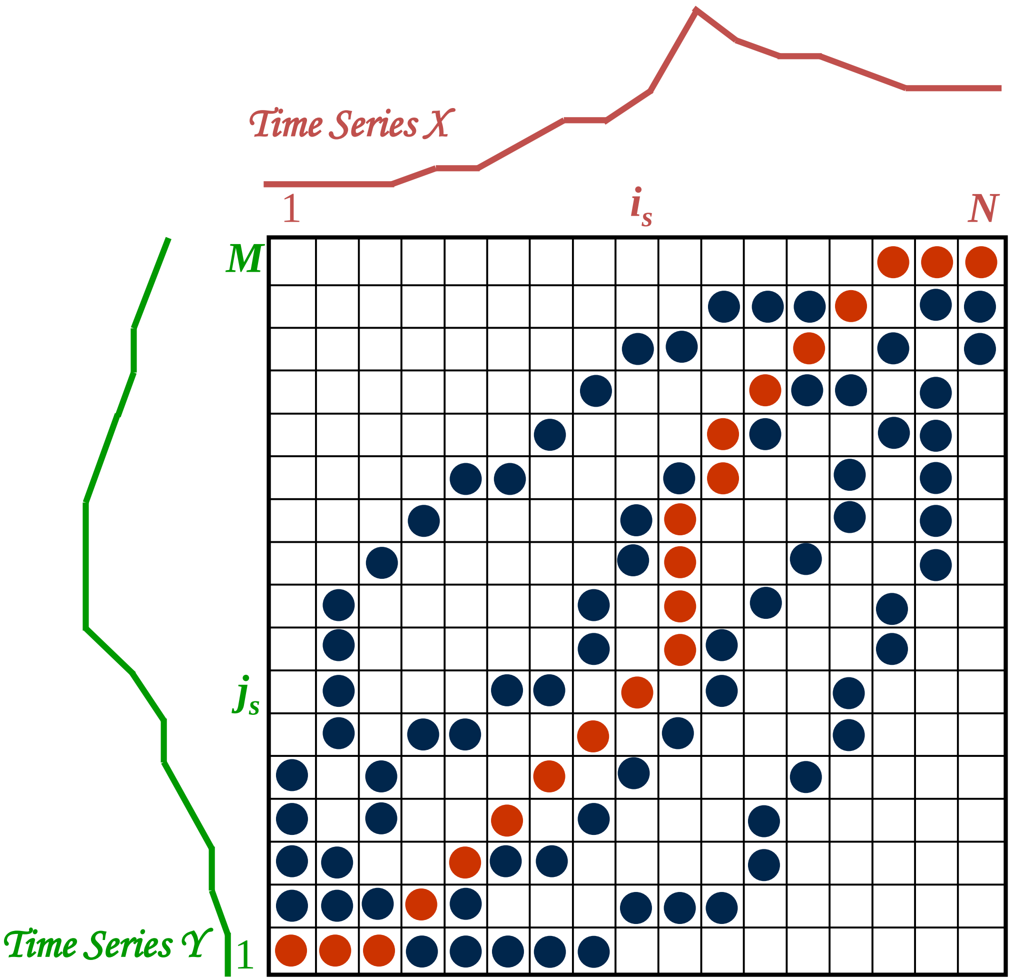


– Vertical: Disallowed

$$D(i, j) = \|\mathbf{x}(i) - \mathbf{y}(j)\| + \min\begin{Bmatrix} D(i-1, j) \\ D(i-1, j-1) \\ D(i-1, j-2) \end{Bmatrix}$$

- Typically used in speech recognition
- **No skipping of any template frame** is allowed
- All transitions move one step to the right, ensuring that each input frame gets used exactly once along any path

Other Path Restrictions for DTW



The number of possible warping paths through the grid is exponentially explosive!

reduction of the search space

Restrictions on the warping function:

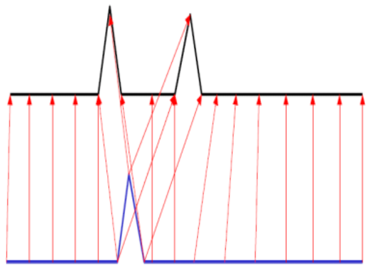
- monotonicity
- continuity
- boundary conditions
- warping window
- slope constraint.

Monotonicity and Continuity

Monotonicity: $i_{s-1} \leq i_s$ and $j_{s-1} \leq j_s$.

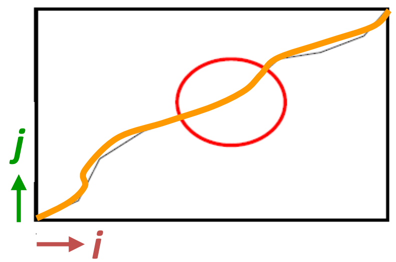
The alignment path does not go back in "*time*" index.

Guarantees that features are not repeated in the alignment.

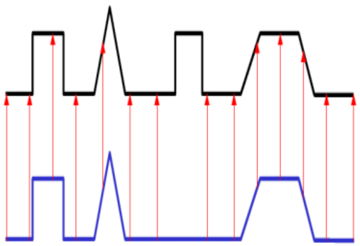


Continuity: $i_s - i_{s-1} \leq 1$ and $j_s - j_{s-1} \leq 1$.

The alignment path does not jump in "*time*" index.



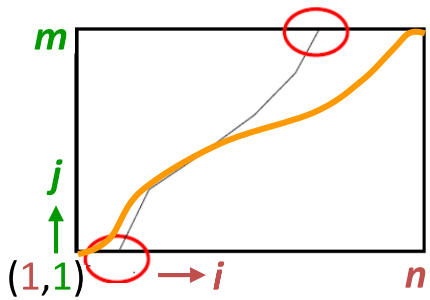
Guarantees that the alignment does not omit important features.



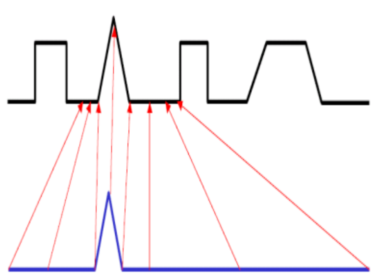
Boundary and Window Conditions

Boundary Conditions: $i_1 = 1$, $i_k = n$ and $j_1 = 1$, $j_k = m$.

The alignment path starts at the bottom left and ends at the top right.

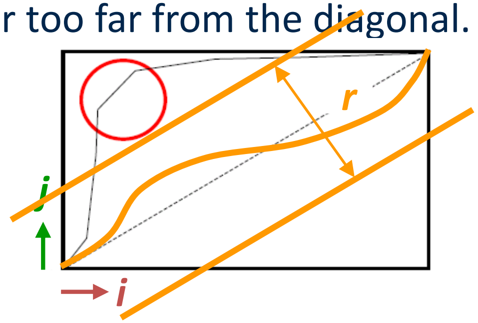


Guarantees that the alignment does not consider partially one of the sequences.

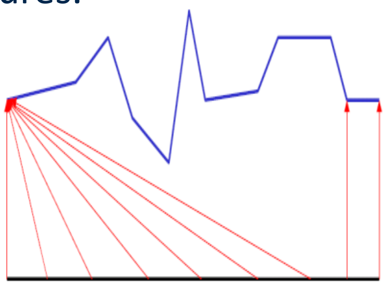


*Warping Window (**Beam Search**)*:
$|i_s - j_s| \le r$, where $r > 0$ is the window length.

A good alignment path is unlikely to wander too far from the diagonal.



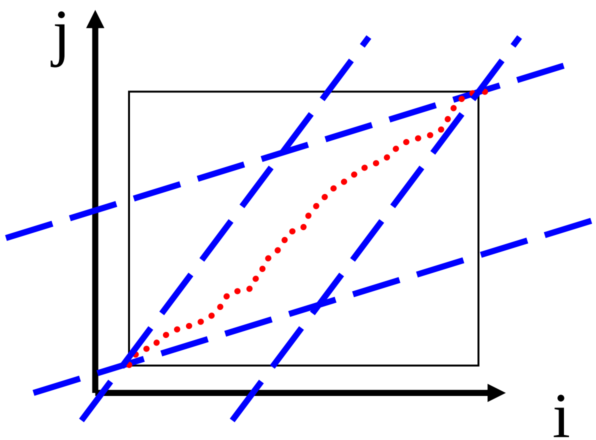
Guarantees that the alignment does not try to skip different features and gets stuck at similar features.

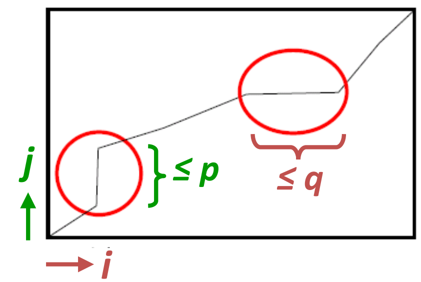


Slope Constraints

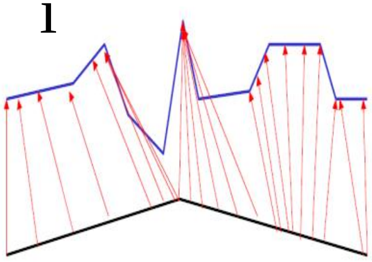
Slope Constraint: $(j_{s_p} - j_{s_0}) \,/\, (i_{s_p} - i_{s_0}) \le p$ and $(i_{s_q} - i_{s_0}) \,/\, (j_{s_q} - j_{s_0}) \le q$, where $q \ge 0$ is the number of steps in the x-direction and $p \ge 0$ is the number of steps in the y-direction.

After q steps in x one must step in y and vice versa: $S = p \,/\, q \in [0, \propto]$.





The alignment path should not be too steep or too shallow.



Prevents that very short parts of the sequences are matched to very long ones.

Path Weighting

Time-normalized distance between X and Y :

$$D(X, Y) = \min_{P}\left[\frac{\sum_{s=1}^{k} d(p_s)\cdot w_s}{\sum_{s=1}^{k} w_s}\right].$$

complicates optimisation

Seeking a weighting coefficient function which guarantees that:

$$C = \sum_{s=1}^{k} w_s$$

is independent of the warping function. Thus

$$D(X, Y) = \frac{1}{C}\min_{P}\left[\sum_{s=1}^{k} d(p_s)\cdot w_s\right]$$

can be solved by use of dynamic programming.

Weighting Coefficient Definitions

- Symmetric form

  $w_s = (i_s - i_{s-1}) + (j_s - j_{s-1})$,

  then $C = n + m$.

- Asymmetric form

  $w_s = (i_s - i_{s-1})$,

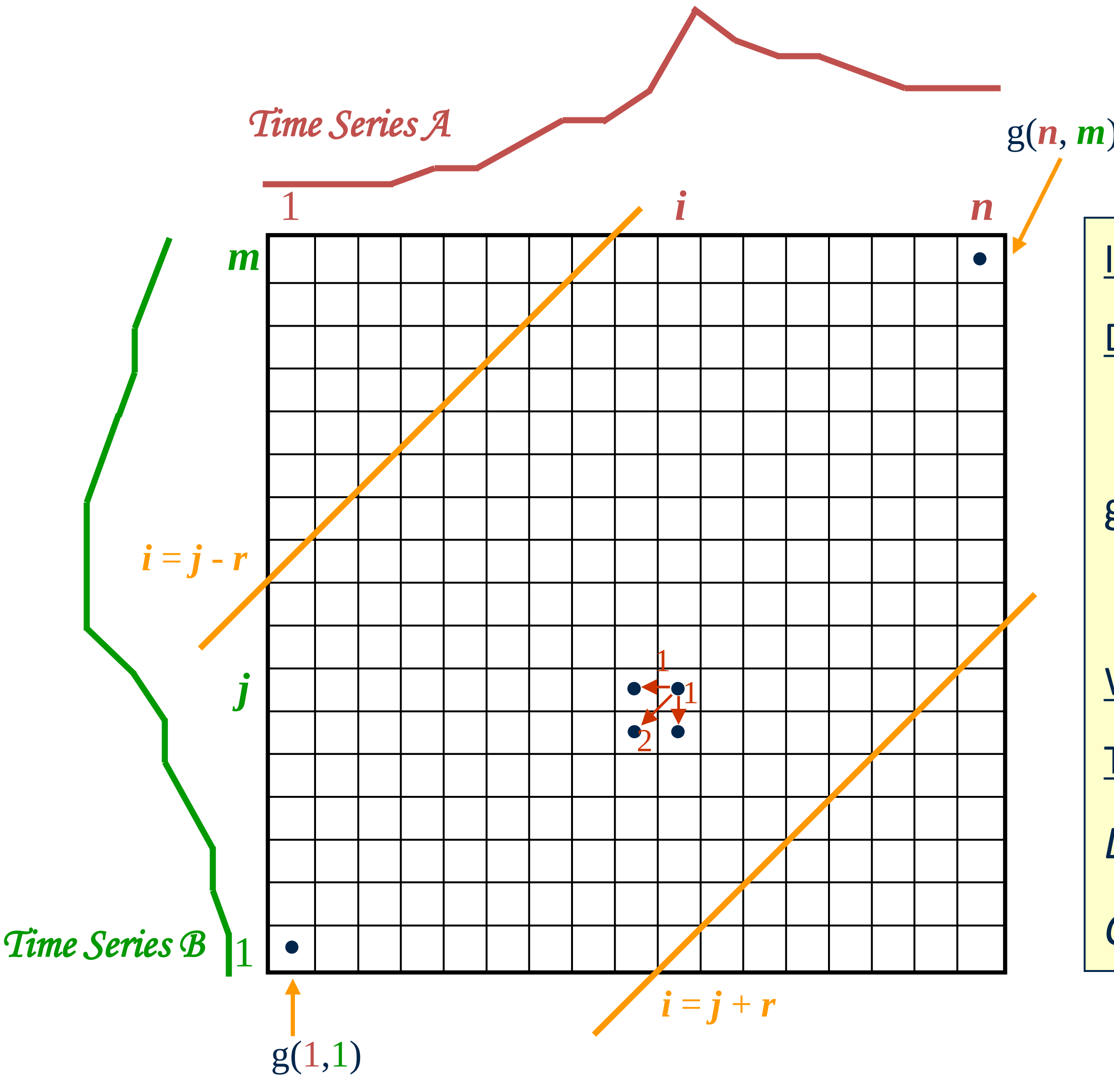
  then $C = n$.

  Or equivalently,

  $w_s = (j_s - j_{s-1})$,

  then $C = m$.

Symmetric DTW Algorithm (warping window, no slope constraint)



Initial condition: g(1,1) = 2d(1,1).

DP-equation:

$$g(i, j) = \min \begin{pmatrix} g(i, j-1) + d(i, j) \\ g(i-1, j-1) + 2d(i, j) \\ g(i-1, j) + d(i, j) \end{pmatrix}.$$

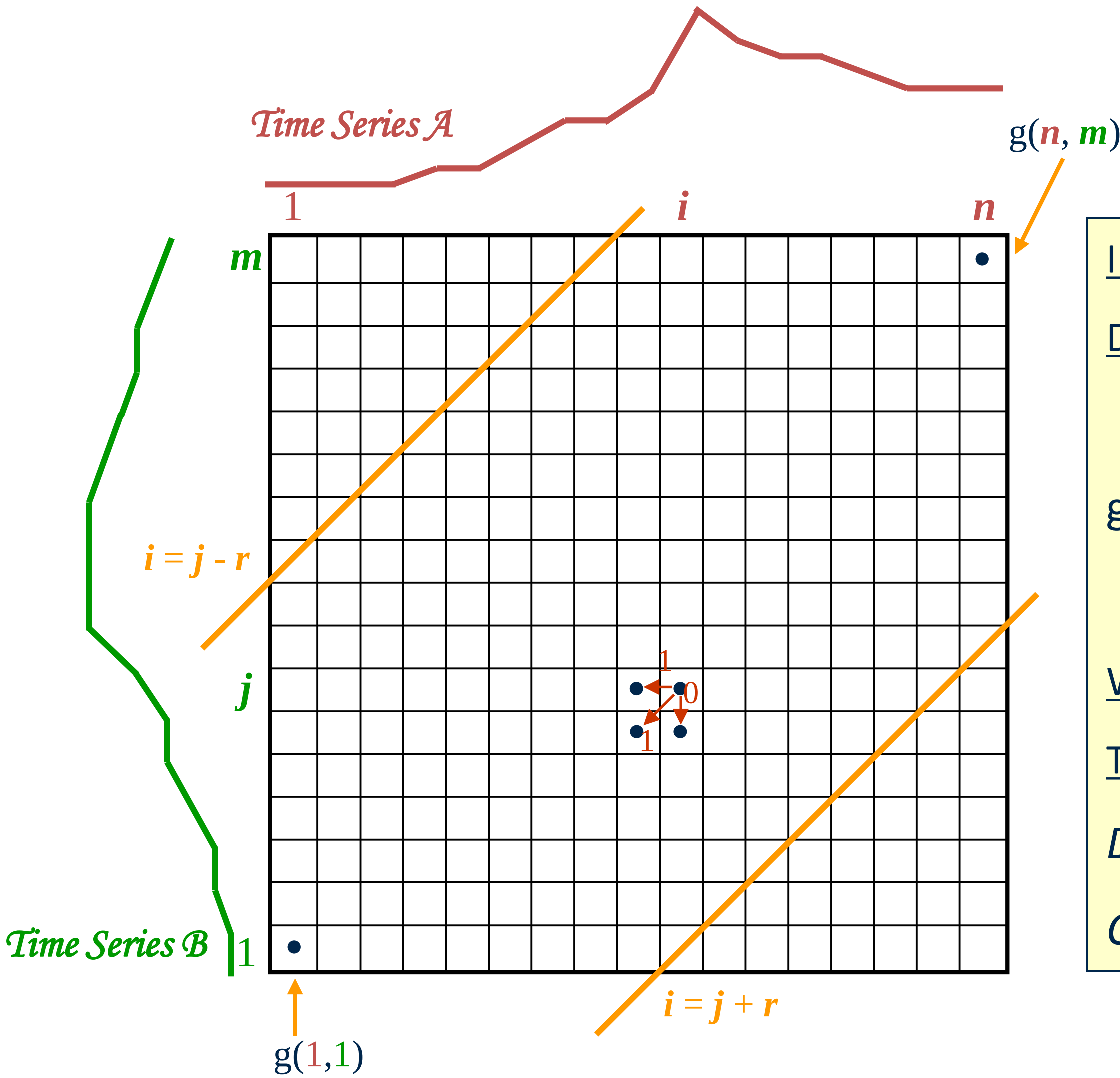
Warping window: $j - r \le i \le j + r$.

Time-normalized distance:

$D(A, B) = g(n, m) / C$

$C = n + m$.

Asymmetric DTW Algorithm (warping window, no slope constraint)



Initial condition: $g(1,1) = d(1,1)$.

DP-equation:

$$g(i, j) = \min \begin{pmatrix} g(i, j-1) \\ g(i-1, j-1) + d(i, j) \\ g(i-1, j) + d(i, j) \end{pmatrix}.$$

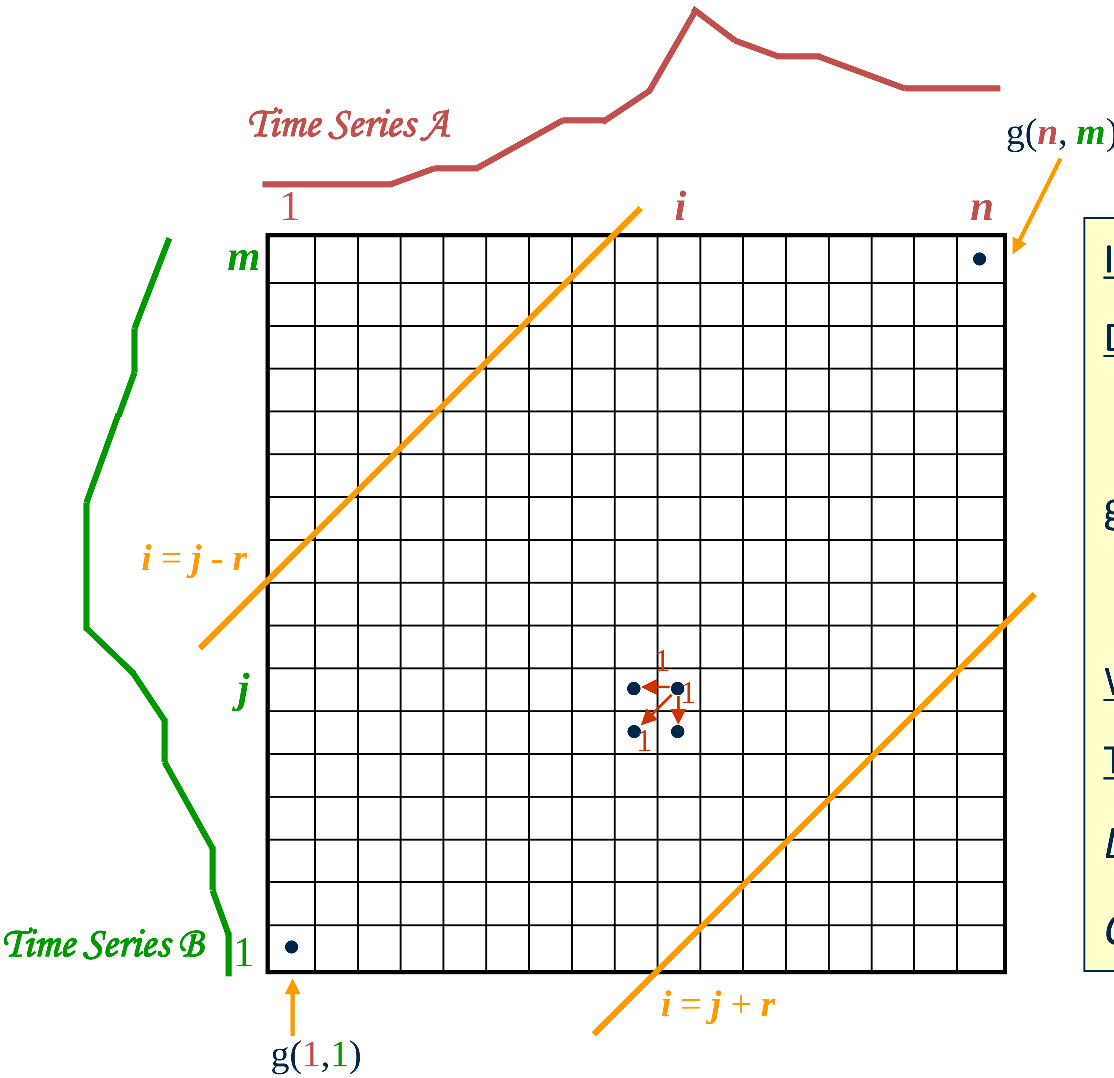
Warping window: $j - r \le i \le j + r$.

Time-normalized distance:

$$D(A, B) = g(n, m) / C$$

$C = n$.

Quazi-symmetric DTW Algorithm (warping window, no slope constraint)



Initial condition: g(1,1) = d(1,1).

DP-equation:

$$g(i, j) = \min \begin{pmatrix} g(i, j-1) + d(i, j) \\ g(i-1, j-1) + d(i, j) \\ g(i-1, j) + d(i, j) \end{pmatrix}.$$

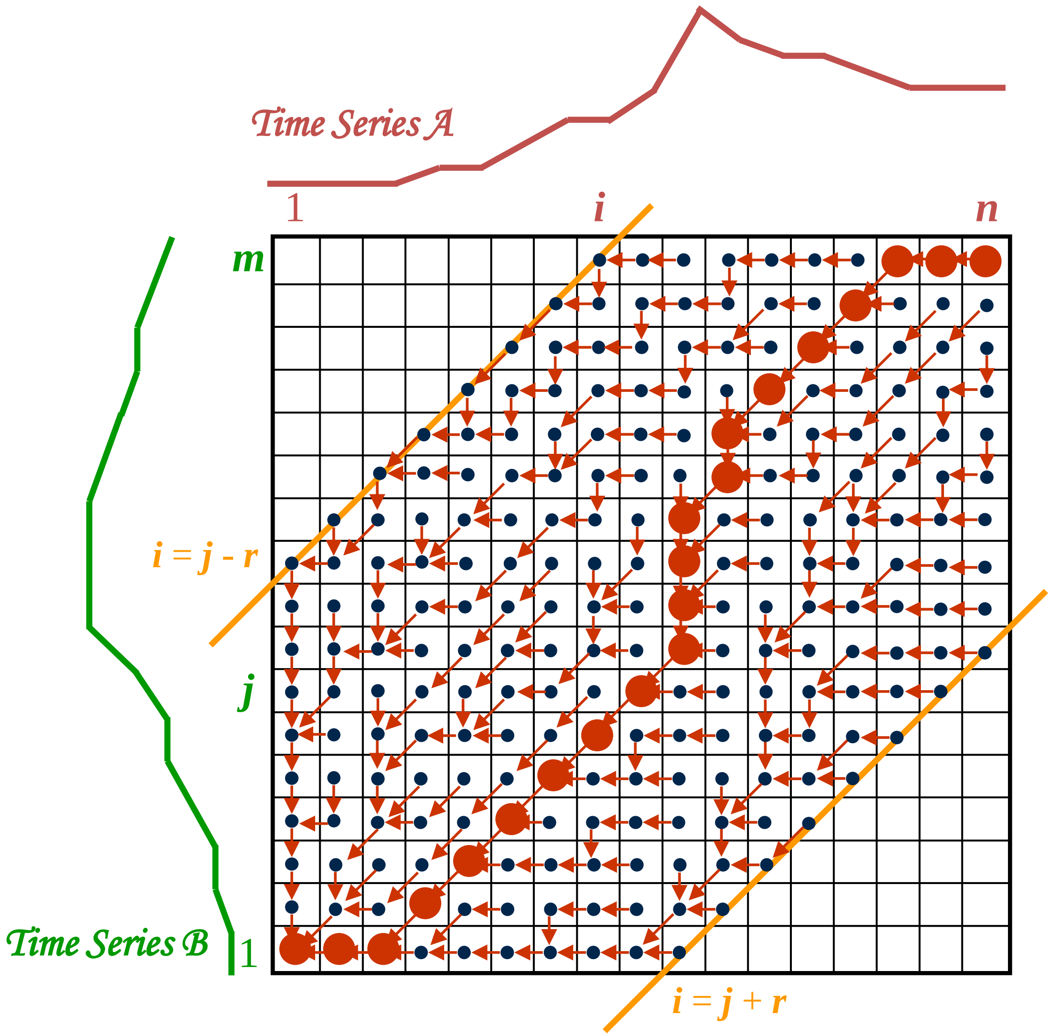
Warping window: $j - r \leq i \leq j + r$.

Time-normalized distance:

$D(A, B) = g(n, m) / C$

$C = n + m$.

DTW Algorithm at Work



Start with the calculation of g(1,1) = d(1,1).

Calculate the first row g(***i***, 1) = g(***i***–1, 1) + d(***i***, 1).

Calculate the first column g(1, ***j***) = g(1, ***j***) + d(1, ***j***).

Move to the second row g(***i***, 2) = min(g(***i***, 1), g(***i***–1, 1), g(***i*** – 1, 2)) + d(***i***, 2). Book keep for each cell the index of this neighboring cell, which contributes the minimum score (red arrows).

Carry on from left to right and from bottom to top with the rest of the grid g(***i***, ***j***) = min(g(***i***, ***j***–1), g(***i***–1, ***j***–1), g(***i*** – 1, ***j***)) + d(***i***, ***j***).

Trace back the best path through the grid starting from g(***n***, ***m***) and moving towards g(1,1) by following the red arrows.

DTW Algorithm: Example

Time Series A →

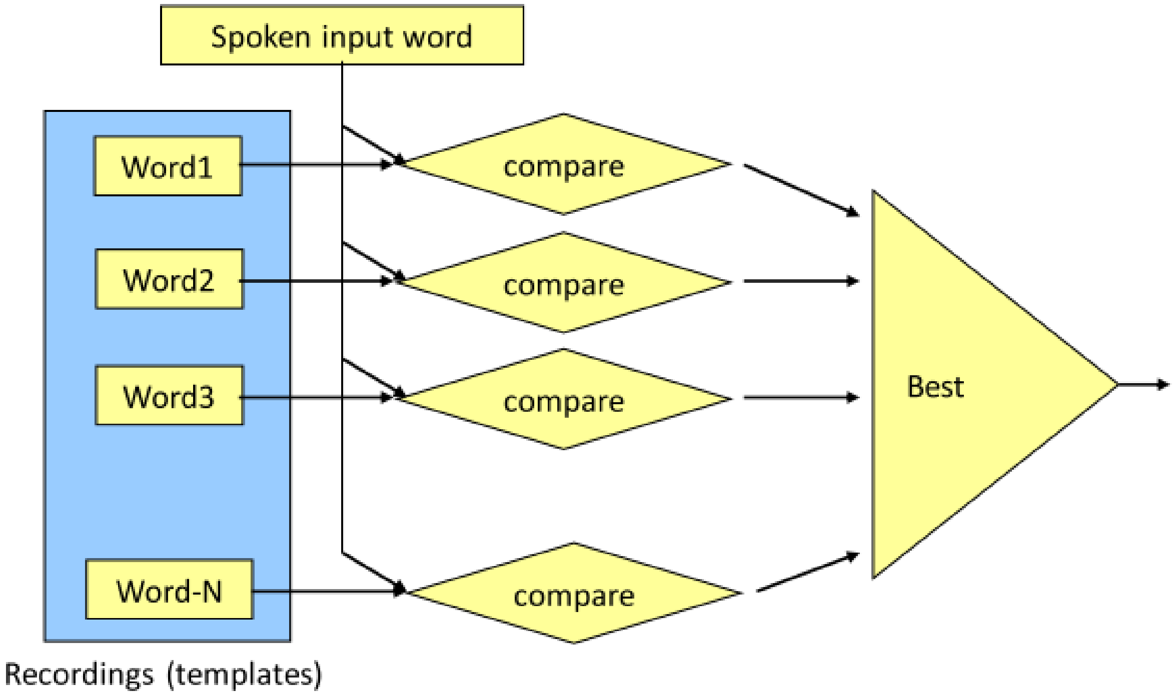
Time Series B ↑

Time Series B \ Time Series A	-0.87, -0.88	-0.84, -0.91	-0.85, -0.84	-0.82, -0.82	-0.23, -0.24	1.95, 1.92	1.36, 1.41	0.60, 0.51	0.0, 0.03	-0.29, -0.18
1.94, 1.97	0.51	0.51	0.49	0.49	0.35	0.17	0.21	0.33	0.41	**0.49**
0.77, 0.74	0.27	0.27	0.26	0.25	0.16	**0.18**	**0.23**	**0.25**	**0.31**	0.68
-0.17, -0.32	0.13	0.13	0.13	0.12	**0.08**	0.26	0.40	0.47	0.49	0.49
-0.58, -0.63	0.08	0.08	0.08	**0.08**	0.10	0.31	0.47	0.57	0.62	0.65
-0.71, -0.68	0.06	0.06	**0.06**	0.07	0.11	0.32	0.50	0.60	0.65	0.68
-0.65, -0.62	0.04	**0.04**	0.06	0.08	0.11	0.32	0.49	0.59	0.64	0.66
-0.60, -0.46	**0.02**	0.05	0.08	0.11	0.13	0.34	0.49	0.58	0.63	0.66

Euclidean distance between vectors

Isolated Word Recognition Using DTW

- TRAINING: for each word in the vocabulary, pre-record a spoken example (its template)
- RECOGNITION of a given recording:
  - for each word in the vocabulary
  - Measure distance of recording to template using DTW
  - Select word whose template has smallest distance

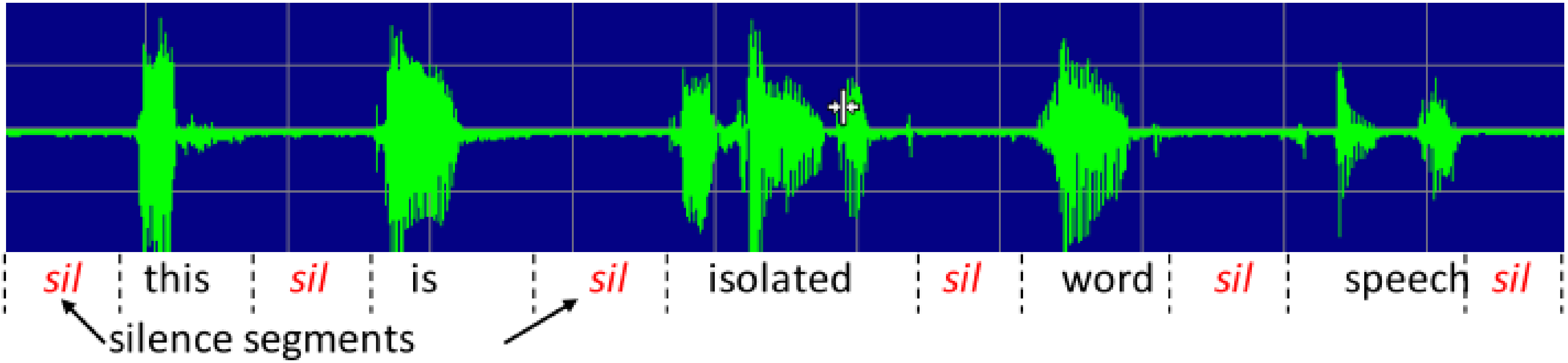


3

Example: Isolated Speech Based Dictation

- We could, in principle, almost build a large vocabulary isolated-word dictation application using DTW
- Training : Record templates (i.e. record one or more instances) of each word in the vocabulary
- Recognition
  - Each word is spoken in isolation, i.e. silence after every word
  - Each isolated word compared to all templates
- Problem: How to detect when a word is spoken?
  - Need a speech/silence detector!

Endpointing: A Revision



- Goal: automatically detect pauses between words
  - to segment the speech stream into isolated words
- Such a speech/silence detector is called an end-point detector (EPD) or VAD (voice activity detector)
  - Detects speech/silence boundaries
- Most speech applications use such an EPD to relieve the user of having to indicate start and end of speech
  - Without Explicit “click-to-speak”, “click-to-stop” button clicks from user, for every word?
    - Obviously extremely tedious

Dealing with Recognition Errors

- Confidence estimation
  - Many methods exist based on statistics
  - One simple solution:
    - Generate N-best list
    - Normalize Top1 distance by (N-1) distances, such as (Top1 – Top2), (Top1-sum(Top2, … , TopN)), etc.
- Improving accuracy by multiple templates
  - Compute DTW distances with multiple templates
  - Use the minimum (best) distance

DEEE725 Speech Signal Processing Lab

Gil-Jin Jang

END OF DYNAMIC TIME WARPING