



Introduction to **Audio Content Analysis**

module 10.1: alignment — dynamic time warping

alexander lerch

introduction

overview

corresponding textbook section

section 10.1

■ lecture content

- Dynamic Time Warping (DTW):
synchronization of two sequences with similar content

■ learning objectives

- explain the standard DTW algorithm
- discuss disadvantages of and modifications to the standard DTW algorithm
- implement DTW



introduction

overview

corresponding textbook section

section 10.1

■ lecture content

- Dynamic Time Warping (DTW):
synchronization of two sequences with similar content

■ learning objectives

- explain the standard DTW algorithm
- discuss disadvantages of and modifications to the standard DTW algorithm
- implement DTW



dynamic time warping

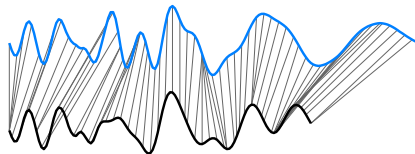
problem statement

■ synchronize two sequences

- *similar* musical content
- *different* tempo and timing

$$A(n_A) \quad n_A \in [0; \mathcal{N}_A - 1]$$

$$B(n_B) \quad n_B \in [0; \mathcal{N}_B - 1]$$



⇒ find alignment path

- ▶ minimizing pairwise distance between sequences
- ▶ covering whole sequence
- ▶ moving only forward in time

dynamic time warping

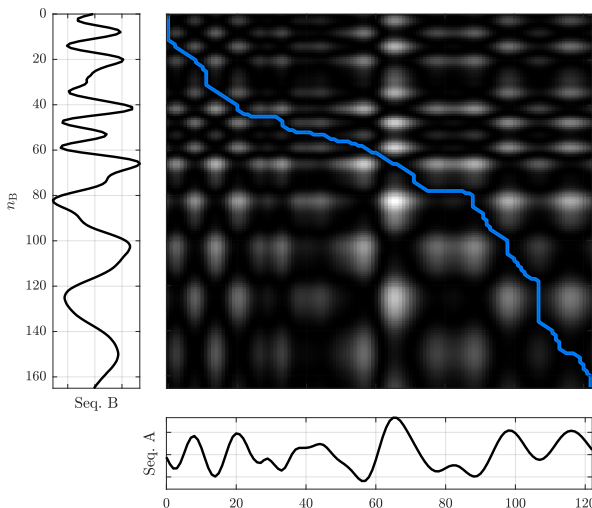
overview

- dynamic programming technique
- time is warped non-linearly to match sequences
- finds optimal match between two sequences given a cost function
- the overall cost indicates the overall distance between the sequences

dynamic time warping

processing steps

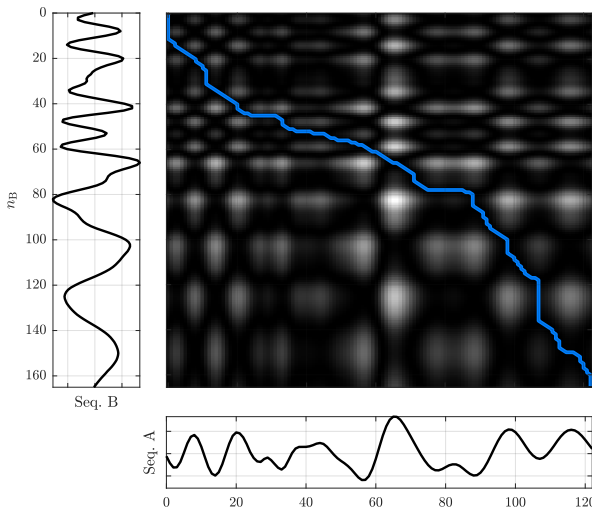
- 1 extract suitable **features**
⇒ two series of feature vectors
- 2 compute **distance matrix**
 $D_{AB}(n_A, n_B)$
- 3 compute **alignment path**
 $p(n_P)$ with $n_P \in [0; \mathcal{N}_P - 1]$
⇒ minimal *overall* distance
- 4 (align sequences using
dynamic time stretching)



dynamic time warping

distance matrix computation

- given 2 sequences of vectors, compute the distance between all pairs of observations
- compute distance matrix $D_{AB}(n_A, n_B)$
 - example $D_{AB}(1, n_B)$ is the distance of the first vector in Seq. A to all vectors in Seq. B



dynamic time warping

path properties 1/2

- **boundaries:** covers both A, B from beginning to end

$$\begin{aligned} \mathbf{p}(0) &= [0, 0] \\ \mathbf{p}(\mathcal{N}_P - 1) &= [\mathcal{N}_A - 1, \mathcal{N}_B - 1] \end{aligned}$$

- **causality:** only forward movement

$$\begin{aligned} n_A|_{\mathbf{p}(n_P)} &\leq n_A|_{\mathbf{p}(n_P+1)} \\ n_B|_{\mathbf{p}(n_P)} &\leq n_B|_{\mathbf{p}(n_P+1)} \end{aligned}$$

- **continuity:** no jumps

$$\begin{aligned} n_A|_{\mathbf{p}(n_P+1)} &\leq (n_A + 1)|_{\mathbf{p}(n_P)} \\ n_B|_{\mathbf{p}(n_P+1)} &\leq (n_B + 1)|_{\mathbf{p}(n_P)} \end{aligned}$$

dynamic time warping

path properties 1/2

- **boundaries:** covers both A, B from beginning to end

$$\begin{aligned} \mathbf{p}(0) &= [0, 0] \\ \mathbf{p}(\mathcal{N}_P - 1) &= [\mathcal{N}_A - 1, \mathcal{N}_B - 1] \end{aligned}$$

- **causality:** only forward movement

$$\begin{aligned} n_A|_{\mathbf{p}(n_P)} &\leq n_A|_{\mathbf{p}(n_P+1)} \\ n_B|_{\mathbf{p}(n_P)} &\leq n_B|_{\mathbf{p}(n_P+1)} \end{aligned}$$

- **continuity:** no jumps

$$\begin{aligned} n_A|_{\mathbf{p}(n_P+1)} &\leq (n_A + 1)|_{\mathbf{p}(n_P)} \\ n_B|_{\mathbf{p}(n_P+1)} &\leq (n_B + 1)|_{\mathbf{p}(n_P)} \end{aligned}$$

dynamic time warping

path properties 1/2

- **boundaries:** covers both A, B from beginning to end

$$\begin{aligned} \mathbf{p}(0) &= [0, 0] \\ \mathbf{p}(\mathcal{N}_P - 1) &= [\mathcal{N}_A - 1, \mathcal{N}_B - 1] \end{aligned}$$

- **causality:** only forward movement

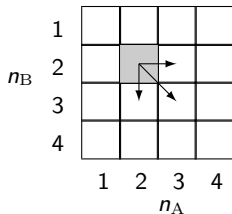
$$\begin{aligned} n_A|_{\mathbf{p}(n_P)} &\leq n_A|_{\mathbf{p}(n_P+1)} \\ n_B|_{\mathbf{p}(n_P)} &\leq n_B|_{\mathbf{p}(n_P+1)} \end{aligned}$$

- **continuity:** no jumps

$$\begin{aligned} n_A|_{\mathbf{p}(n_P+1)} &\leq (n_A + 1)|_{\mathbf{p}(n_P)} \\ n_B|_{\mathbf{p}(n_P+1)} &\leq (n_B + 1)|_{\mathbf{p}(n_P)} \end{aligned}$$

alignment

path properties 2/2

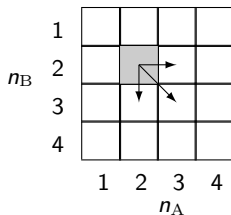


what is the minimum/maximum path length



alignment

path properties 2/2



what is the minimum/maximum path length

$$\mathcal{N}_{P,\min} = \max(\mathcal{N}_A, \mathcal{N}_B)$$

$$\mathcal{N}_{P,\max} = \mathcal{N}_A + \mathcal{N}_B - 2$$



alignment

DTW: overall cost

- every path has an *overall cost*

$$\mathfrak{C}_{AB}(j) = \sum_{n_P=0}^{\mathcal{N}_P-1} D(\mathbf{p}_j(n_P))$$

- *optimal* path minimizes the overall cost

$$\begin{aligned}\mathfrak{C}_{AB,min} &= \min_{\forall j} (\mathfrak{C}_{AB}(j)) \\ j_{opt} &= \operatorname{argmin}_{\forall j} (\mathfrak{C}_{AB}(j))\end{aligned}$$

⇒ stay in the 'valleys' of distance matrix

how to determine the optimal path



alignment

DTW: overall cost

- every path has an *overall cost*

$$\mathfrak{C}_{AB}(j) = \sum_{n_P=0}^{\mathcal{N}_P-1} D(\mathbf{p}_j(n_P))$$

- *optimal* path minimizes the overall cost

$$\begin{aligned}\mathfrak{C}_{AB,min} &= \min_{\forall j} (\mathfrak{C}_{AB}(j)) \\ j_{opt} &= \operatorname{argmin}_{\forall j} (\mathfrak{C}_{AB}(j))\end{aligned}$$

⇒ stay in the 'valleys' of distance matrix

how to determine the optimal path



alignment

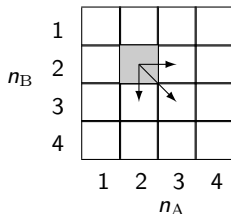
DTW: accumulated cost 1/2

accumulated cost: *cost matrix*

$$C_{AB}(n_A, n_B) = D_{AB}(n_A, n_B) + \min \begin{cases} C_{AB}(n_A - 1, n_B - 1) \\ C_{AB}(n_A - 1, n_B) \\ C_{AB}(n_A, n_B - 1) \end{cases}$$

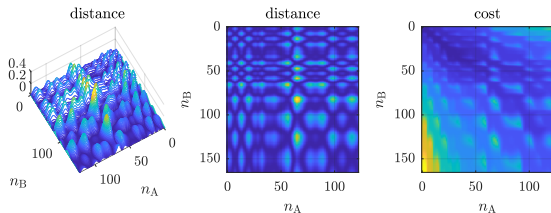
■ initialization

$$C_{AB}(0, 0) = D_{AB}(0, 0)$$



alignment

DTW: accumulated cost 2/2



alignment

DTW: algorithm description 1/2

■ initialization:

$$\mathbf{C}_{AB}(0, 0) = \mathbf{D}_{AB}(0, 0), \mathbf{C}_{AB}(n_A, -1) = \infty, \mathbf{C}_{AB}(-1, n_B) = \infty$$

■ recursion:

$$\mathbf{C}_{AB}(n_A, n_B) = \mathbf{D}_{AB}(n_A, n_B) + \min \begin{cases} \mathbf{C}_{AB}(n_A - 1, n_B - 1) \\ \mathbf{C}_{AB}(n_A - 1, n_B) \\ \mathbf{C}_{AB}(n_A, n_B - 1) \end{cases}$$

$$j = \operatorname{argmin} \begin{cases} \mathbf{C}_{AB}(n_A - 1, n_B - 1) \\ \mathbf{C}_{AB}(n_A - 1, n_B) \\ \mathbf{C}_{AB}(n_A, n_B - 1) \end{cases}$$

$$\Delta p(n_A, n_B) = \begin{cases} [-1, -1] & \text{if } j = 0 \\ [-1, 0] & \text{if } j = 1 \\ [0, -1] & \text{if } j = 2 \end{cases}$$

alignment

DTW: algorithm description 1/2

■ initialization:

$$\mathcal{C}_{AB}(0, 0) = \mathcal{D}_{AB}(0, 0), \mathcal{C}_{AB}(n_A, -1) = \infty, \mathcal{C}_{AB}(-1, n_B) = \infty$$

■ recursion:

$$\mathcal{C}_{AB}(n_A, n_B) = \mathcal{D}_{AB}(n_A, n_B) + \min \begin{cases} \mathcal{C}_{AB}(n_A - 1, n_B - 1) \\ \mathcal{C}_{AB}(n_A - 1, n_B) \\ \mathcal{C}_{AB}(n_A, n_B - 1) \end{cases}$$

$$j = \operatorname{argmin} \begin{cases} \mathcal{C}_{AB}(n_A - 1, n_B - 1) \\ \mathcal{C}_{AB}(n_A - 1, n_B) \\ \mathcal{C}_{AB}(n_A, n_B - 1) \end{cases}$$

$$\Delta p(n_A, n_B) = \begin{cases} [-1, -1] & \text{if } j = 0 \\ [-1, 0] & \text{if } j = 1 \\ [0, -1] & \text{if } j = 2 \end{cases}$$

alignment

DTW: algorithm description 2/2

■ termination:

$$n_A = \mathcal{N}_A - 1 \wedge n_B = \mathcal{N}_B - 1$$

■ path backtracking:

$$p(n_P) = p(n_P + 1) + \Delta p(p(n_P + 1)), \quad n_P = \mathcal{N}_P - 2, \mathcal{N}_P - 3, \dots, 0$$

alignment

DTW: algorithm description 2/2

■ termination:

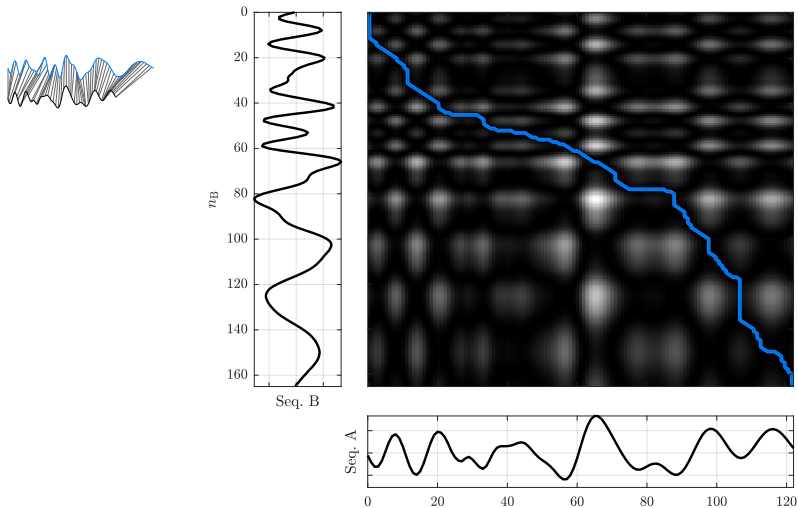
$$n_A = \mathcal{N}_A - 1 \wedge n_B = \mathcal{N}_B - 1$$

■ path backtracking:

$$\mathbf{p}(n_P) = \mathbf{p}(n_P + 1) + \Delta \mathbf{p}(\mathbf{p}(n_P + 1)), \quad n_P = \mathcal{N}_P - 2, \mathcal{N}_P - 3, \dots, 0$$

dynamic time warping

DTW: example



dynamic time warping

example

$$A = [1, 2, 3, 0],$$
$$B = [1, 0, 2, 3, 1],$$

compute distance matrix, cost matrix, and DTW path



dynamic time warping

example

$$\begin{aligned} A &= [1, 2, 3, 0], \\ B &= [1, 0, 2, 3, 1], \end{aligned}$$

compute distance matrix, cost matrix, and DTW path

$$D_{AB} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix}$$



dynamic time warping

example

$$A = [1, 2, 3, 0],$$

$$B = [1, 0, 2, 3, 1],$$

compute distance matrix, cost matrix, and DTW path

$$D_{AB} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix}$$

$$C_{AB} = \begin{bmatrix} 0 & \leftarrow 1 & \leftarrow 3 & \leftarrow 4 \\ \uparrow 1 & \swarrow 2 & \swarrow 4 & \swarrow 3 \\ \uparrow 2 & \swarrow 1 & \leftarrow 2 & \leftarrow 4 \\ \uparrow 4 & \uparrow 2 & \swarrow 1 & \leftarrow 4 \\ \uparrow 4 & \uparrow 3 & \uparrow 3 & \swarrow 2 \end{bmatrix}$$



dynamic time warping

example

$$A = [1, 2, 3, 0],$$

$$B = [1, 0, 2, 3, 1],$$

compute distance matrix, cost matrix, and DTW path

$$D_{AB} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix}$$

$$C_{AB} = \begin{bmatrix} 0 & \leftarrow 1 & \leftarrow 3 & \leftarrow 4 \\ \uparrow 1 & \swarrow 2 & \swarrow 4 & \swarrow 3 \\ \uparrow 2 & \swarrow 1 & \leftarrow 2 & \leftarrow 4 \\ \uparrow 4 & \uparrow 2 & \swarrow 1 & \leftarrow 4 \\ \uparrow 4 & \uparrow 3 & \uparrow 3 & \swarrow 2 \end{bmatrix}$$



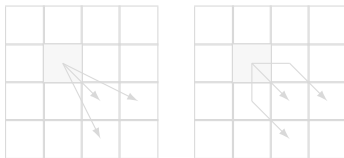
dynamic time warping

variants

- transition weights: favor specific path directions

$$C_{AB}(n_A, n_B) = \min \begin{cases} C_{AB}(n_A - 1, n_B - 1) & + & \lambda_d \cdot D_{AB}(n_A, n_B) \\ C_{AB}(n_A - 1, n_B) & + & \lambda_v \cdot D_{AB}(n_A, n_B) \\ C_{AB}(n_A, n_B - 1) & + & \lambda_h \cdot D_{AB}(n_A, n_B) \end{cases}$$

- step types



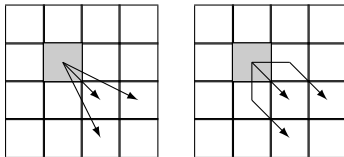
dynamic time warping

variants

- transition weights: favor specific path directions

$$C_{AB}(n_A, n_B) = \min \begin{cases} C_{AB}(n_A - 1, n_B - 1) & + & \lambda_d \cdot D_{AB}(n_A, n_B) \\ C_{AB}(n_A - 1, n_B) & + & \lambda_v \cdot D_{AB}(n_A, n_B) \\ C_{AB}(n_A, n_B - 1) & + & \lambda_h \cdot D_{AB}(n_A, n_B) \end{cases}$$

- step types



dynamic time warping optimization

■ **challenge:** distance matrix dimensions $\mathcal{N}_A \cdot \mathcal{N}_B$

⇒ DTW *inefficient* for long sequences

- high memory requirements
- large number of operations

optimizations:

- 1 maximum time and tempo deviation
- 2 sliding window
- 3 multi-scale DTW (several downsampled iterations)

dynamic time warping optimization

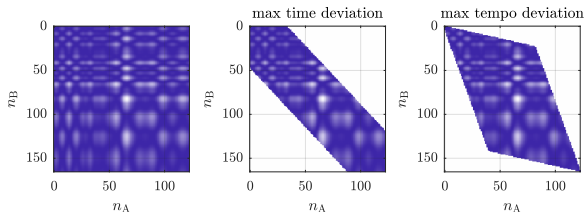
■ **challenge:** distance matrix dimensions $\mathcal{N}_A \cdot \mathcal{N}_B$

⇒ DTW *inefficient* for long sequences

- high memory requirements
- large number of operations

optimizations:

- 1 maximum time and tempo deviation
- 2 sliding window
- 3 multi-scale DTW (several downsampled iterations)



dynamic time warping optimization

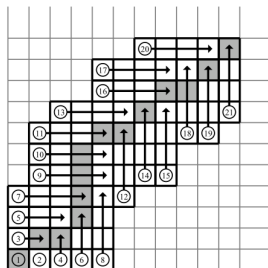
■ **challenge:** distance matrix dimensions $\mathcal{N}_A \cdot \mathcal{N}_B$

⇒ DTW *inefficient* for long sequences

- high memory requirements
- large number of operations

optimizations:

- 1 maximum time and tempo deviation
- 2 sliding window
- 3 multi-scale DTW (several downsampled iterations)



1

¹S. Dixon and G. Widmer, "MATCH: A Music Alignment Tool Chest," in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, Sep. 2005.

dynamic time warping

optimization

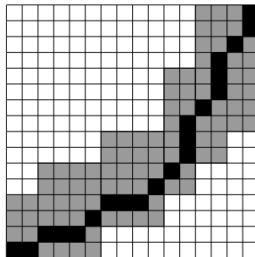
■ **challenge:** distance matrix dimensions $\mathcal{N}_A \cdot \mathcal{N}_B$

⇒ DTW *inefficient* for long sequences

- high memory requirements
- large number of operations

optimizations:

- 1 maximum time and tempo deviation
- 2 sliding window
- 3 multi-scale DTW (several downsampled iterations)



1

¹M. Müller, H. Mattes, and F. Kurth, "An Efficient Multiscale Approach to Audio Synchronization," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Victoria, 2006.

dynamic time warping

DTW vs. Viterbi

similarities and differences of DTW and the Viterbi algorithm



dynamic time warping

DTW vs. Viterbi



similarities and differences of DTW and the Viterbi algorithm

■ commonalities

- find path through matrix
- maximizes overall probability/minimizes overall cost
- based on dynamic programming principles

■ differences

- DTW has more constraints: start/end in corner, move only to neighbor
- DTW is not usually parametrized by training data (transition probs, construction of distance/emission prob matrix)
- Viterbi path length is predefined, DTW path length is not

summary

lecture content

■ dynamic time warping

- find globally optimal alignment path between two sequences

■ processing steps

- 1 compute distance matrix
- 2 compute cost matrix
- 3 back-track path

