Introduction to Audio Content Analysis

Module 10.1: Alignment — Dynamic Time Warping

alexander lerch

Georgia Tech Center for Music Technology
College of Design

Georgia Tech Center for Music Technology College of Design

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

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

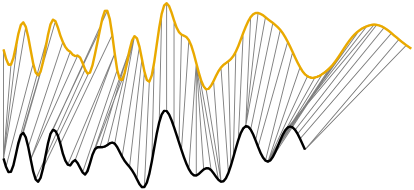
- **synchronize two sequences**
  - *similar* musical content
  - *different* tempo and timing

$$A(n_\mathrm{A}) \quad n_\mathrm{A} \in [0; \mathcal{N}_\mathrm{A} - 1]$$

$$B(n_\mathrm{B}) \quad n_\mathrm{B} \in [0; \mathcal{N}_\mathrm{B} - 1]$$

$\Rightarrow$ find alignment path

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



matlab source: plotSequenceAlignment.m

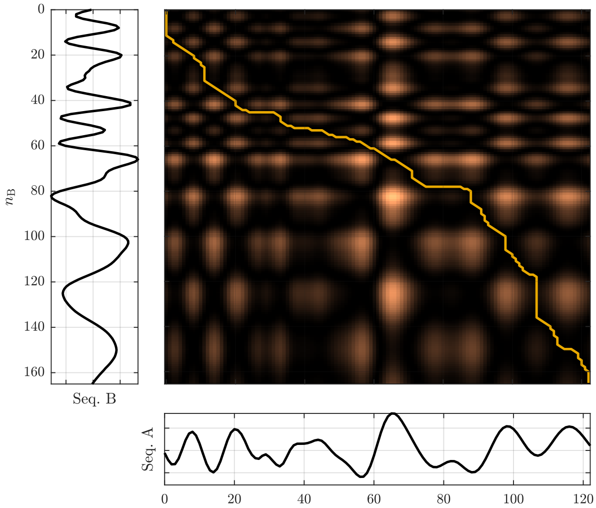
dynamic time warping
overview

- 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**
   $\Rightarrow$ two series of feature vectors
2. compute **distance matrix** $\boldsymbol{D}_{\mathrm{AB}}(n_{\mathrm{A}}, n_{\mathrm{B}})$
3. compute **alignment path** $\boldsymbol{p}(n_{\mathrm{P}})$ with $n_{\mathrm{P}} \in [0; \mathcal{N}_{\mathrm{P}} - 1]$
   $\Rightarrow$ minimal *overall* distance
4. (align sequences using dynamic time stretching)

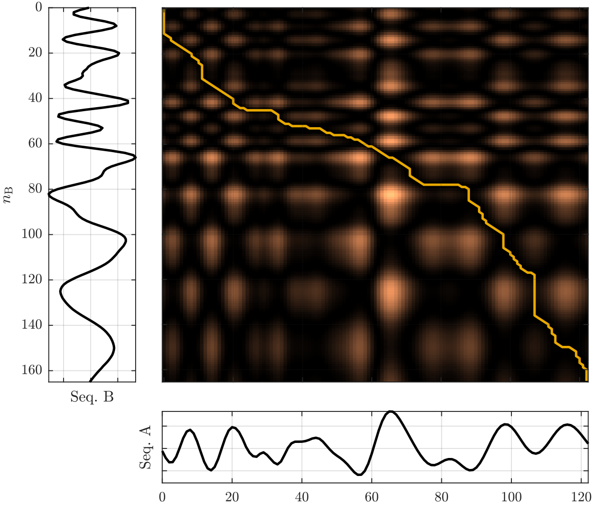


matlab source: plotDtwPath.m

dynamic time warping

distance matrix computation

- given 2 sequences of vectors, compute the distance between all pairs of observations
- compute distance matrix $\boldsymbol{D}_{\mathrm{AB}}(n_{\mathrm{A}}, n_{\mathrm{B}})$
  - example $\boldsymbol{D}_{\mathrm{AB}}(1, n_{\mathrm{B}})$ is the distance of the first vector in Seq. A to all vectors in Seq. B



matlab source: plotDtwPath.m

dynamic time warping

path properties 1/2

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

$$\begin{aligned} \boldsymbol{p}(0) &= [0, 0] \\ \boldsymbol{p}(\mathcal{N}_\mathrm{P} - 1) &= [\mathcal{N}_\mathrm{A} - 1, \mathcal{N}_\mathrm{B} - 1] \end{aligned}$$

- **causality**: only forward movement

$$n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P})} \leq n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)}$$
$$n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P})} \leq n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)}$$

- **continuity**: no jumps

$$n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)} \leq (n_\mathrm{A} + 1)\big|_{\boldsymbol{p}(n_\mathrm{P})}$$
$$n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)} \leq (n_\mathrm{B} + 1)\big|_{\boldsymbol{p}(n_\mathrm{P})}$$

dynamic time warping
path properties 1/2

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

$$\begin{aligned}\boldsymbol{p}(0) &= [0, 0]\\ \boldsymbol{p}(\mathcal{N}_\mathrm{P}-1) &= [\mathcal{N}_\mathrm{A}-1, \mathcal{N}_\mathrm{B}-1]\end{aligned}$$

- **causality**: only forward movement

$$n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P})} \leq n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)}$$
$$n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P})} \leq n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)}$$

- **continuity**: no jumps

$$n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)} \leq (n_\mathrm{A}+1)\big|_{\boldsymbol{p}(n_\mathrm{P})}$$
$$n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)} \leq (n_\mathrm{B}+1)\big|_{\boldsymbol{p}(n_\mathrm{P})}$$

dynamic time warping

path properties 1/2

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

$$\begin{aligned}\boldsymbol{p}(0) &= [0, 0]\\ \boldsymbol{p}(\mathcal{N}_\mathrm{P}-1) &= [\mathcal{N}_\mathrm{A}-1, \mathcal{N}_\mathrm{B}-1]\end{aligned}$$

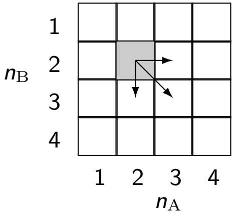
- **causality**: only forward movement

$$n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P})} \leq n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)}$$
$$n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P})} \leq n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)}$$

- **continuity**: no jumps

$$n_\mathrm{A}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)} \leq (n_\mathrm{A}+1)\big|_{\boldsymbol{p}(n_\mathrm{P})}$$
$$n_\mathrm{B}\big|_{\boldsymbol{p}(n_\mathrm{P}+1)} \leq (n_\mathrm{B}+1)\big|_{\boldsymbol{p}(n_\mathrm{P})}$$

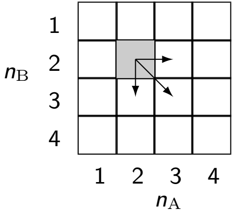
alignment
path properties 2/2



what is the minimum/maximum path length

alignment

path properties 2/2



what is the minimum/maximum path length

$$\mathcal{N}_{\mathrm{P,min}} = \max(\mathcal{N}_{\mathrm{A}}, \mathcal{N}_{\mathrm{B}})$$

$$\mathcal{N}_{\mathrm{P,max}} = \mathcal{N}_{\mathrm{A}} + \mathcal{N}_{\mathrm{B}} - 2$$

- every path has an *overall cost*

$$\mathfrak{C}_{\mathrm{AB}}(j) = \sum_{n_\mathrm{P}=0}^{\mathcal{N}_\mathrm{P}-1} \boldsymbol{D}\left(\boldsymbol{p}_j(n_\mathrm{P})\right)$$

- *optimal* path minimizes the overall cost

$$\mathfrak{C}_{\mathrm{AB},min} = \min_{\forall j}\left(\mathfrak{C}_{\mathrm{AB}}(j)\right)$$

$$j_{\mathrm{opt}} = \operatorname*{argmin}_{\forall j}\left(\mathfrak{C}_{\mathrm{AB}}(j)\right)$$

$\Rightarrow$ stay in the 'valleys' of distance matrix

how to determine the optimal path

- every path has an *overall cost*

$$\mathfrak{C}_{\mathrm{AB}}(j) = \sum_{n_{\mathrm{P}}=0}^{\mathcal{N}_{\mathrm{P}}-1} \boldsymbol{D}\left(\boldsymbol{p}_j(n_{\mathrm{P}})\right)$$

- *optimal* path minimizes the overall cost

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

$\Rightarrow$ stay in the 'valleys' of distance matrix

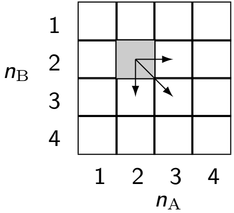
how to determine the optimal path

accumulated cost: *cost matrix*

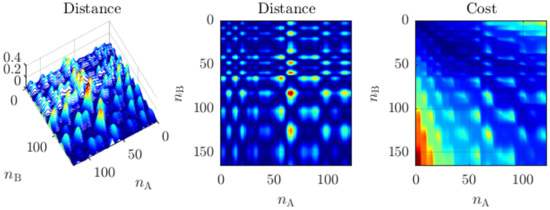
$$\mathbf{C}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B}) = \mathbf{D}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B}) + \min \begin{cases} \mathbf{C}_{\mathrm{AB}}(n_\mathrm{A} - 1, n_\mathrm{B} - 1) \\ \mathbf{C}_{\mathrm{AB}}(n_\mathrm{A} - 1, n_\mathrm{B}) \\ \mathbf{C}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B} - 1) \end{cases}$$

- initialization

$$\mathbf{C}_{\mathrm{AB}}(0, 0) = \mathbf{D}_{\mathrm{AB}}(0, 0)$$



alignment
DTW: accumulated cost 2/2



matlab source: plotDtwCost.m

alignment
DTW: algorithm description 1/2

- **initialization**:

$$C_{\mathrm{AB}}(0,0) = D_{\mathrm{AB}}(0,0),\ C_{\mathrm{AB}}(n_{\mathrm{A}},-1) = \infty,\ C_{\mathrm{AB}}(-1,n_{\mathrm{B}}) = \infty$$

- **recursion**:

$$\begin{aligned}
C_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}) &= D_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}) + \min\begin{cases} C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}}-1) \\ C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}}) \\ C_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}-1) \end{cases} \\
j &= \operatorname{argmin}\begin{cases} C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}}-1) \\ C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}}) \\ C_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}-1) \end{cases} \\
\Delta p(n_{\mathrm{A}},n_{\mathrm{B}}) &= \begin{cases} [-1,-1] & \text{if } j=0 \\ [-1,0] & \text{if } j=1 \\ [0,-1] & \text{if } j=2 \end{cases}
\end{aligned}$$

alignment
DTW: algorithm description 1/2

- **initialization**:

$$C_{\mathrm{AB}}(0,0) = D_{\mathrm{AB}}(0,0),\ C_{\mathrm{AB}}(n_{\mathrm{A}},-1) = \infty,\ C_{\mathrm{AB}}(-1,n_{\mathrm{B}}) = \infty$$

- **recursion**:

$$\begin{aligned}
C_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}) &= D_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}) + \min\begin{cases} C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}}-1)\\ C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}})\\ C_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}-1)\end{cases}\\
j &= \operatorname{argmin}\begin{cases} C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}}-1)\\ C_{\mathrm{AB}}(n_{\mathrm{A}}-1,n_{\mathrm{B}})\\ C_{\mathrm{AB}}(n_{\mathrm{A}},n_{\mathrm{B}}-1)\end{cases}\\
\Delta\boldsymbol{p}(n_{\mathrm{A}},n_{\mathrm{B}}) &= \begin{cases} [-1,-1] & \text{if } j=0\\ [-1,0] & \text{if } j=1\\ [0,-1] & \text{if } j=2\end{cases}
\end{aligned}$$

alignment
DTW: algorithm description 2/2

- **termination**:

$$n_\mathrm{A} = \mathcal{N}_\mathrm{A} - 1 \wedge n_\mathrm{B} = \mathcal{N}_\mathrm{B} - 1$$

- **path backtracking**:

$$\boldsymbol{p}(n_\mathrm{P}) = \boldsymbol{p}(n_\mathrm{P}+1) + \Delta\boldsymbol{p}\big(\boldsymbol{p}(n_\mathrm{P}+1)\big), \;\; n_\mathrm{P} = \mathcal{N}_\mathrm{P}-2, \mathcal{N}_\mathrm{P}-3, \ldots, 0$$

alignment
DTW: algorithm description 2/2

- **termination**:

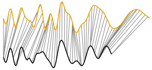
$$n_\mathrm{A} = \mathcal{N}_\mathrm{A} - 1 \wedge n_\mathrm{B} = \mathcal{N}_\mathrm{B} - 1$$

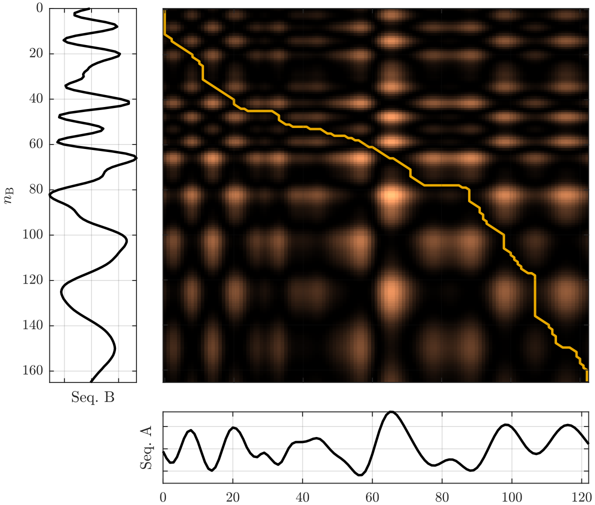
- **path backtracking**:

$$\boldsymbol{p}(n_\mathrm{P}) = \boldsymbol{p}(n_\mathrm{P}+1) + \Delta\boldsymbol{p}\big(\boldsymbol{p}(n_\mathrm{P}+1)\big),\ n_\mathrm{P} = \mathcal{N}_\mathrm{P} - 2, \mathcal{N}_\mathrm{P} - 3, \ldots, 0$$

dynamic time warping

DTW: example





matlab source: plotDtwPath.m

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

compute distance matrix, cost matrix, and DTW path

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

compute distance matrix, cost matrix, and DTW path

$$\boldsymbol{D}_{\mathrm{AB}} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix}$$

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

compute distance matrix, cost matrix, and DTW path

$$
\boldsymbol{D}_{\mathrm{AB}} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix}
\qquad
\boldsymbol{C}_{\mathrm{AB}} = \begin{bmatrix} 0 & \leftarrow 1 & \leftarrow 3 & \leftarrow 4 \\ \uparrow 1 & \nwarrow 2 & \nwarrow 4 & \nwarrow 3 \\ \uparrow 2 & \nwarrow 1 & \leftarrow 2 & \leftarrow 4 \\ \uparrow 4 & \uparrow 2 & \nwarrow 1 & \leftarrow 4 \\ \uparrow 4 & \uparrow 3 & \uparrow 3 & \nwarrow 2 \end{bmatrix}
$$

dynamic time warping

example

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

compute distance matrix, cost matrix, and DTW path

$$\boldsymbol{D}_{\mathrm{AB}} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \end{bmatrix} \qquad \boldsymbol{C}_{\mathrm{AB}} = \begin{bmatrix} 0 & \leftarrow 1 & \leftarrow 3 & \leftarrow 4 \\ \uparrow 1 & \nwarrow 2 & \nwarrow 4 & \nwarrow 3 \\ \uparrow 2 & \nwarrow 1 & \leftarrow 2 & \leftarrow 4 \\ \uparrow 4 & \uparrow 2 & \nwarrow 1 & \leftarrow 4 \\ \uparrow 4 & \uparrow 3 & \uparrow 3 & \nwarrow 2 \end{bmatrix}$$

- transition weights: favor specific path directions

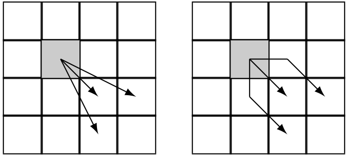
$$\mathbf{C}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B}) = \min \begin{cases} \mathbf{C}_{\mathrm{AB}}(n_\mathrm{A}-1, n_\mathrm{B}-1) & + \quad \lambda_\mathrm{d} \cdot \mathbf{D}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B}) \\ \mathbf{C}_{\mathrm{AB}}(n_\mathrm{A}-1, n_\mathrm{B}) & + \quad \lambda_\mathrm{v} \cdot \mathbf{D}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B}) \\ \mathbf{C}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B}-1) & + \quad \lambda_\mathrm{h} \cdot \mathbf{D}_{\mathrm{AB}}(n_\mathrm{A}, n_\mathrm{B}) \end{cases}$$

- step types

- transition weights: favor specific path directions

$$C_{\mathrm{AB}}(n_{\mathrm{A}}, n_{\mathrm{B}}) = \min \begin{cases} C_{\mathrm{AB}}(n_{\mathrm{A}}-1, n_{\mathrm{B}}-1) & + \quad \lambda_{\mathrm{d}} \cdot D_{\mathrm{AB}}(n_{\mathrm{A}}, n_{\mathrm{B}}) \\ C_{\mathrm{AB}}(n_{\mathrm{A}}-1, n_{\mathrm{B}}) & + \quad \lambda_{\mathrm{v}} \cdot D_{\mathrm{AB}}(n_{\mathrm{A}}, n_{\mathrm{B}}) \\ C_{\mathrm{AB}}(n_{\mathrm{A}}, n_{\mathrm{B}}-1) & + \quad \lambda_{\mathrm{h}} \cdot D_{\mathrm{AB}}(n_{\mathrm{A}}, n_{\mathrm{B}}) \end{cases}$$

- step types



dynamic time warping

optimization

- **challenge**: distance matrix dimensions $\mathcal{N}_\mathrm{A} \cdot \mathcal{N}_\mathrm{B}$

$\Rightarrow$ 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

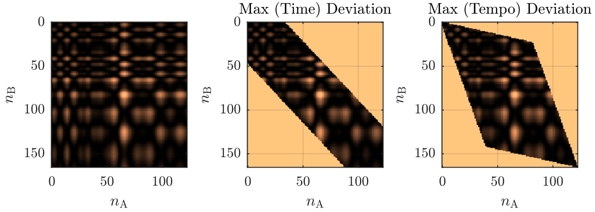
optimization

- **challenge**: distance matrix dimensions $\mathcal{N}_\mathrm{A} \cdot \mathcal{N}_\mathrm{B}$

$\Rightarrow$ 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)



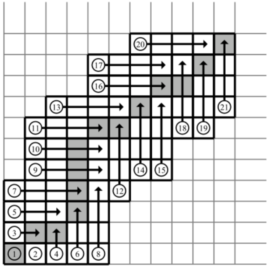
matlab source: plotDtwConstraints.m

- **challenge**: distance matrix dimensions $\mathcal{N}_\mathrm{A} \cdot \mathcal{N}_\mathrm{B}$

$\Rightarrow$ 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)

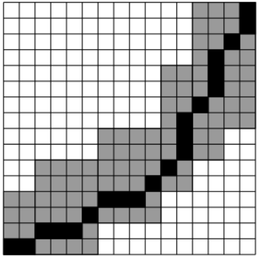


1

[1] S. 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.

- **challenge**: distance matrix dimensions $\mathcal{N}_\mathrm{A} \cdot \mathcal{N}_\mathrm{B}$

$\Rightarrow$ 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

[1] M. 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

Georgia Tech Center for Music Technology College of Design

similarities and differences of DTW and the Viterbi algorithm

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