INTERNATIONAL AUDIO LABORATORIES ERLANGEN
A joint institution of Fraunhofer IIS and Universität Erlangen Nürnberg

ISMIR Tutorial
Daejeon, Korea, September 21, 2025

Differentiable Alignment Techniques for Music Processing: Techniques and Applications

Part 2: Theoretical Foundations

Meinard Müller, Johannes Zeitler
International Audio Laboratories Erlangen
{meinard.mueller, johannes.zeitler}@audiolabs-erlangen.de

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg
Fraunhofer IIS

Overview

Part 0: Overview

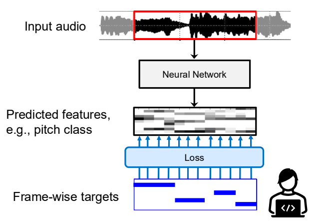
Part 1: Introduction to Alignment Techniques

Coffee Break

Part 2: Theoretical Foundations & Implementation

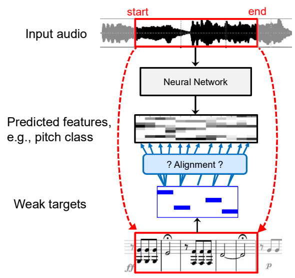
Introduction: Training with Strongly Aligned Targets

- Train DNN-based feature extractor from audio
- Frame-wise annotations (strong targets) are very costly

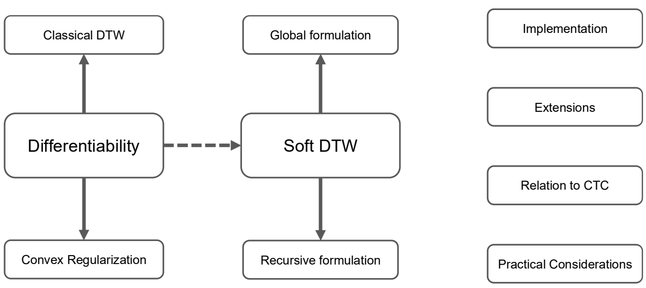


Introduction: Training with Weakly Aligned Targets

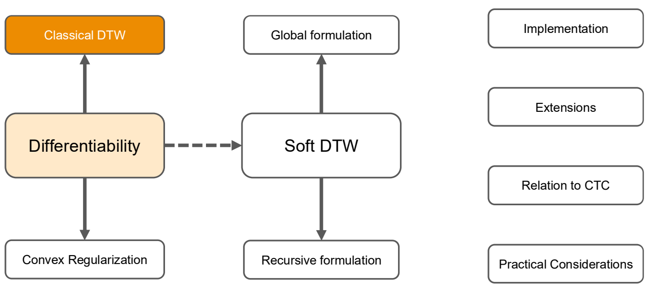
- Train DNN-based feature extractor from audio
- Frame-wise annotations (strong targets) are very costly
- Only annotate start & end of audio segments
- Retrieve note events from musical score
- Weak targets Y provide information about note event order, but not duration
- Use alignment techniques to train DNN on weakly aligned data



Overview

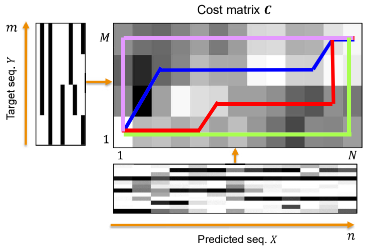


Overview



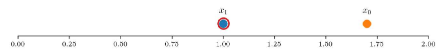
Recap: Dynamic Time Warping

- Compute cost matrix $\boldsymbol{C} \in \mathbb{R}^{N \times M}$
- $\boldsymbol{C}(n,m) = c(x_n, y_m)$ with cost function $c: \mathcal{F}_X \times \mathcal{F}_Y \to \mathbb{R}$
- Goal: compute minimum cost over the cost matrix, taking valid paths $P \in \mathcal{P} = \{$ $\}$
- $\mathrm{DTW}(\boldsymbol{C}) = \min\left(\left\{\sum_{(n,m)\in P} \boldsymbol{C}(n,m) \mid P \in \mathcal{P}\right\}\right)$
- Problem: min function does not have a continuous derivative!



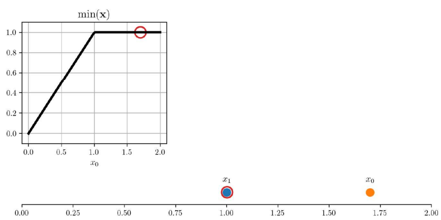
Differentiable Minimum Functions

- Investigate mimimum function over $\boldsymbol{x} = [x_0, x_1]$
- Argmin ○ changes when $x_0 > x_1$



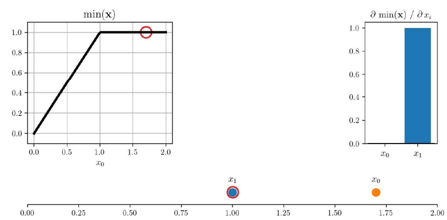
Differentiable Minimum Functions

- Investigate mimimum function over $\boldsymbol{x} = [x_0, x_1]$
- Argmin ○ changes when $x_0 > x_1$
- Minimum function: "edge" at $x_0 = 1.0$



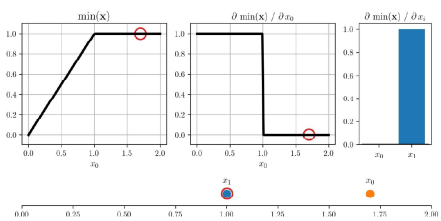
Differentiable Minimum Functions

- Investigate mimimum function over $\boldsymbol{x} = [x_0, x_1]$
- Argmin ○ changes when $x_0 > x_1$
- Minimum function: "edge" at $x_0 = 1.0$
- Argmin (derivative): hard decision for x_0 or x_1



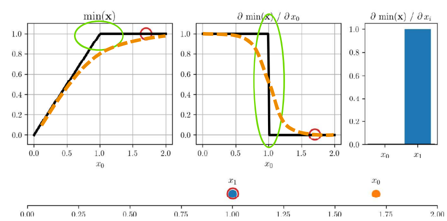
Differentiable Minimum Functions

- Investigate mimimum function over $\boldsymbol{x} = [x_0, x_1]$
- Argmin ○ changes when $x_0 > x_1$
- Minimum function: "edge" at $x_0 = 1.0$
- Argmin (derivative): hard decision for x_0 or x_1
- Gradient: discontinuity when argmin changes



Differentiable Minimum Functions

- Gradient: discontinuity when argmin changes
- Why is the discontinuity problematic?
  - "Winner takes it all"
  - Toy example: hard choice between x_0 and x_1
  - Alignment: hard choice for one path
  - Full gradient flow goes to a single path!
  - What if we are not sure about the best path?
- "Soft Choice" between x_0 and x_1?



Overview

Classical DTW ↑ Differentiability ↓ Convex Regularization

Differentiability - - → Soft DTW

Global formulation ↑ Soft DTW ↓ Recursive formulation

- Implementation
- Extensions
- Relation to CTC
- Practical Considerations

Smoothing Functions via Convex Regularization

M. Blondel and V. Roulet, "The elements of differentiable programming", arxiv preprint, 2025

- Represent an optimization problem as a „dual problem"
- Transform: „convex conjugate"

$f(x)$ → Conv. conjugate → $f^*(q)$ → Conv. conjugate → $f^{**}(x) = f(x)$

Primal Space | Dual Space | Primal Space

Theorems: f^* convex

Guarantees: / | f^* is convex | /

Smoothing Functions via Convex Regularization

M. Blondel and V. Roulet, "The elements of differentiable programming", arxiv preprint, 2025

- Calculate convex conjugate for minimum function
- Guarantee: $\min^*$ is convex
- No guarantee for $\min^{**}$

$\min(x)$ → Conv. conjugate → $\min^*(q)$ → Conv. conjugate → $\min^{**}(x) = \min(x)$

Primal Space | Dual Space | Primal Space

Theorems: f^* convex | f^* strongly convex ⇒ f^{**} smooth

Guarantees: / | $\min^*$ is convex | /

Smoothing Functions via Convex Regularization

M. Blondel and V. Roulet, "The elements of differentiable programming", arxiv preprint, 2025

- Can we enforce strong convexity in the dual space?
- Add a strongly convex regularizer Ω to $\min^*$
- $\min_\Omega$ is guaranteed to be smooth!

$\min(x)$ → Conv. conjugate → $\min^*(q)$ ⊕ $\Omega(q)$ → $\min^*_\Omega(q)$ → Conv. conjugate → $\min^{**}_\Omega(x) = \min_\Omega(x)$

Primal Space | Dual Space | Primal Space

Theorems: Sum of (weakly) convex and strongly convex function is strongly convex | f^* convex | f^* strongly convex ⇒ f^{**} smooth

Guarantees: / | $\min^*$ is convex | $\min^*_\Omega$ is strongly convex | $\min_\Omega$ is smooth

Softmin

- Popular choice for $\Omega(q)$: Entropy function

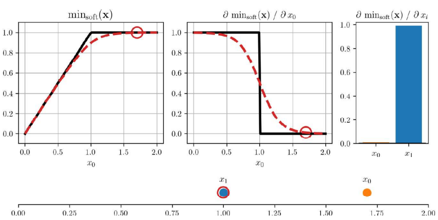
$$\Omega(q) = \sum_{q_i \in q} q_i \log q_i$$

- Solving for optimum yields closed-form "softmin"

$$\min^\gamma_{\text{soft}}(x) = -\gamma \log \sum_i \exp\left(-\frac{x_i}{\gamma}\right)$$

… with gradient $\left[\nabla \min^\gamma_{\text{soft}}\right]_i = \frac{\exp\left(-\frac{x_i}{\gamma}\right)}{\sum_j \exp\left(-\frac{x_j}{\gamma}\right)}$

- Temperature parameter γ controls smoothness



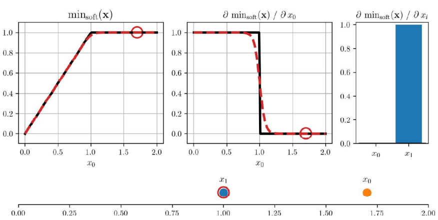
Softmin Temperature

- Softmin:

$$\min^\gamma_{\text{soft}}(x) = -\gamma \log \sum_i \exp\left(-\frac{x_i}{\gamma}\right)$$

… with gradient $\left[\nabla \min^\gamma_{\text{soft}}\right]_i = \frac{\exp\left(-\frac{x_i}{\gamma}\right)}{\sum_j \exp\left(-\frac{x_j}{\gamma}\right)}$

- Small temperature γ: approach hardmin

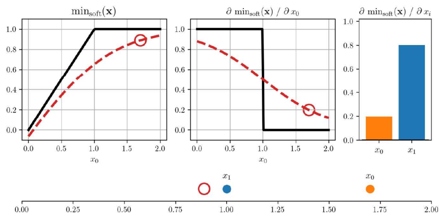


Softmin Temperature

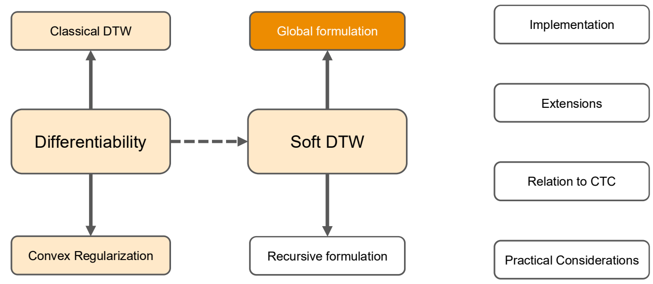
- Softmin:

  $\min^{\gamma}_{\text{soft}}(x) = -\gamma \log \sum_i \exp\left(-\frac{x_i}{\gamma}\right)$

  … with gradient $\left[\nabla \min^{\gamma}_{\text{soft}}\right]_i = \frac{\exp\left(-\frac{x_i}{\gamma}\right)}{\sum_j \exp\left(-\frac{x_j}{\gamma}\right)}$
- Small temperature γ: approach hardmin
- High temperature γ: approach averaging
- We always compute a lower bound for min!



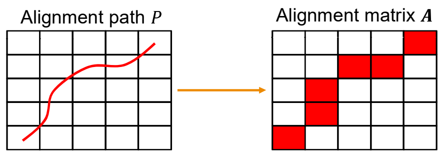
Overview



SoftDTW

Define alignment paths $P \in \mathcal{P}$ as equivalent alignment matrices $\boldsymbol{A} \in \mathcal{A}$ via a one-hot encoding $\boldsymbol{A} \in \mathbb{R}^{N \times M}$

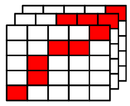
$$\boldsymbol{A}(n,m) = \begin{cases} 1, & \text{if } (n,m) \in P, \\ 0, & \text{else.} \end{cases}$$



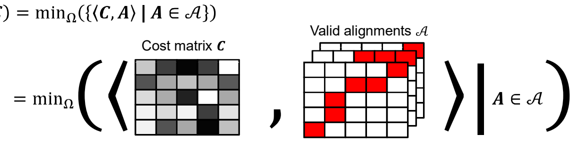
SoftDTW

M. Cuturi and M. Blondel, „Soft-DTW: a differentiable loss function for time series, ICML 2017

- Set of valid alignments $\mathcal{A} = \{\boldsymbol{A}_1, \dots, \boldsymbol{A}_I\}$ =

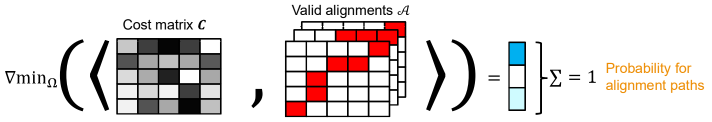


- $\text{SDTW}(\boldsymbol{C}) = \min_{\Omega}(\{\langle \boldsymbol{C}, \boldsymbol{A} \rangle \mid \boldsymbol{A} \in \mathcal{A}\})$

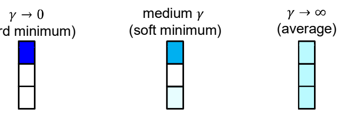


Gradient of SoftDTW

Gradient of minimum function $\nabla \min_{\Omega}$ denotes the influence of individual alignment paths on total cost



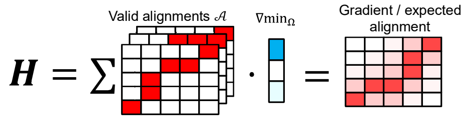
Behavior of $\nabla \min_{\Omega}$ depends on regularization strength:



Gradient of SoftDTW

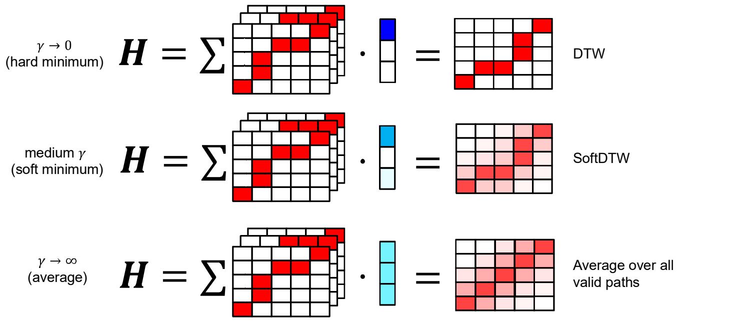
- Define gradient $\boldsymbol{H} \in \mathbb{R}^{N \times M}$ as influence of cost cell $\boldsymbol{C}(n,m)$ on total alignment cost $\text{SDTW}(\boldsymbol{C})$:

  $\boldsymbol{H}(n,m) := \frac{\partial\, \text{SDTW}(\boldsymbol{C})}{\partial\, \boldsymbol{C}(n,m)}$
- Gradient $\boldsymbol{H}$ is sum of alignment matrices $\boldsymbol{A}$, weighted with gradient $\nabla \min_{\Omega}$

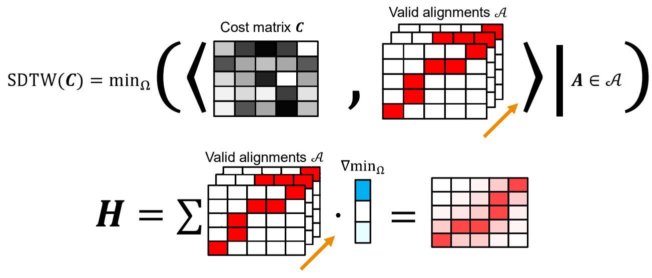


$$= \sum_{i=1}^{|\mathcal{A}|} \boldsymbol{A}^{(i)} \cdot \nabla \min_{\Omega}^{(i)}$$

Gradient for different regularization strengths

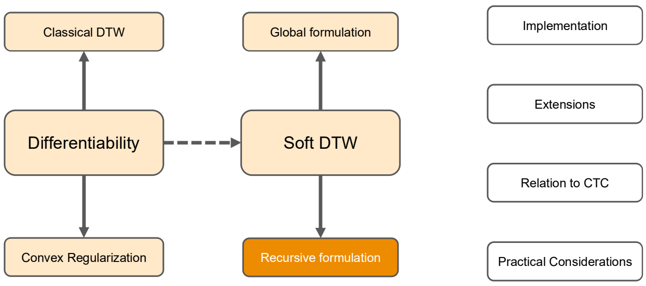


Summary: SDTW in Global Formulation



Problem: $|\mathcal{A}|$ grows exponentially!

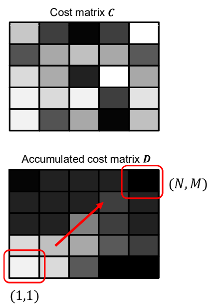
Overview



A Recursive Algorithm for SDTW: Forward

M. Cuturi and M. Blondel, „Soft-DTW: a differentiable loss function for time series, ICML 2017

- Compute SDTW recursively with dynamic programming
- Input: local cost matrix $\boldsymbol{C} \in \mathbb{R}^{N \times M}$
- Output: accumulated cost matrix $\boldsymbol{D} \in \mathbb{R}^{N \times M}$
- $\boldsymbol{D}(n, m)$: minimum cost over all paths leading to (n, m)
- $\boldsymbol{D}(N, M) = \mathrm{SDTW}(\boldsymbol{C})$
- Requirements:
  - Boundary conditions: start in $(1,1)$, end in (N, M)
  - Allowed step sizes $\mathcal{S} = \{(1,0), (0,1), (1,1)\}$



A Recursive Algorithm for SDTW: Forward

M. Cuturi and M. Blondel, „Soft-DTW: a differentiable loss function for time series, ICML 2017

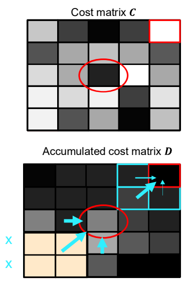
Recursion:

min over… | current local cost | + | acc. cost from incoming steps | evaluate for all steps

$$\boldsymbol{D}(n, m) = \min_\Omega(\{\boldsymbol{C}(n, m) + \boldsymbol{D}(n-i, m-j) \mid (i, j) \in \mathcal{S}\})$$

$$\boldsymbol{D}(N, M) = \mathrm{SDTW}(\boldsymbol{C})$$

Computational complexity: $\mathcal{O}(NM)$
(linear in sequence lengths)



Relation of Global and Recursive Formulation

A. Mensch and M. Blondel, "Differentiable dynamic programming for structured prediction and attention", ICML 2018

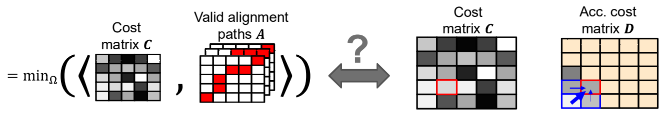
Global formulation

$$\mathrm{SDTW}^{\mathrm{glo}}(\boldsymbol{C}) = \min_\Omega(\{\langle \boldsymbol{C}, \boldsymbol{A} \rangle \mid \boldsymbol{A} \in \mathcal{A}\})$$

Recursive formulation

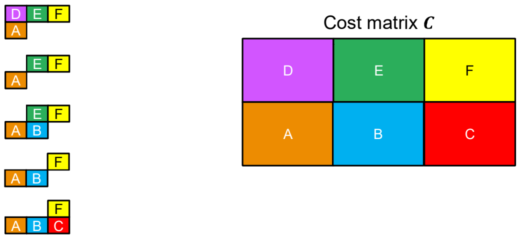
$$\boldsymbol{D}(n, m) = \min_\Omega(\{\boldsymbol{C}(n, m) + \boldsymbol{D}(n-i, m-j) \mid (i, j) \in \mathcal{S}\})$$

$$\mathrm{SDTW}^{\mathrm{rec}}(\boldsymbol{C}) = \boldsymbol{D}(N, M)$$

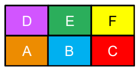


Dividing the Global Problem

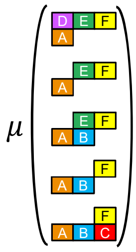
- 5 possible alignment paths



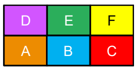
Dividing the Global Problem



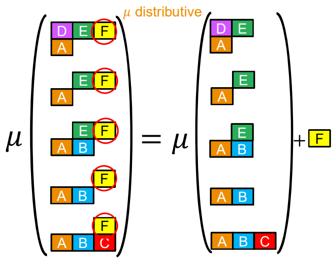
- 5 possible alignment paths
- Compute minimum cost over these 5 paths



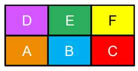
Dividing the Global Problem



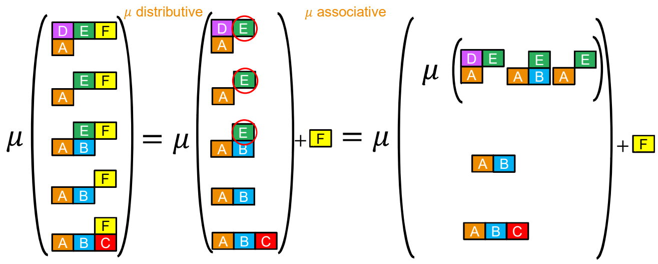
- Goal: facilitate the computation
- F is the end of every path
- Move it out of the min function!



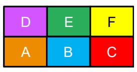
Dividing the Global Problem



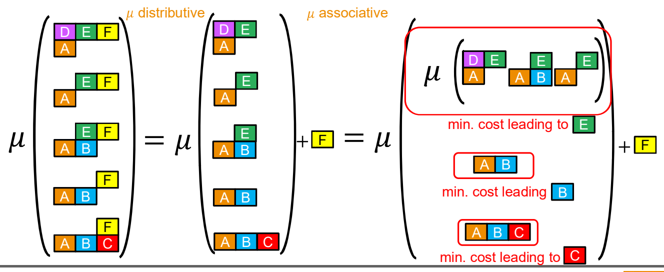
- Divide into sub-problems
- For example, all paths ending in E



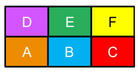
Dividing the Global Problem



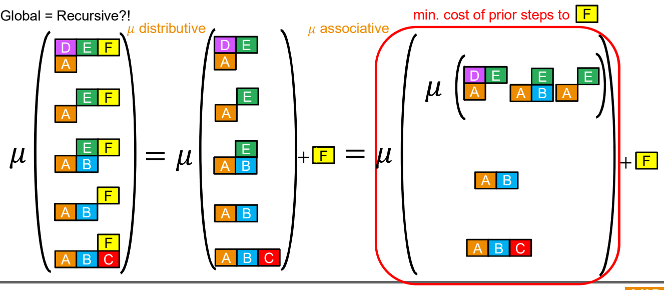
- We found solutions for sub-problems!



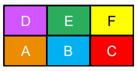
Dividing the Global Problem



- We found solutions for sub-problems!
- We have established the recursion!
- Global = Recursive?!



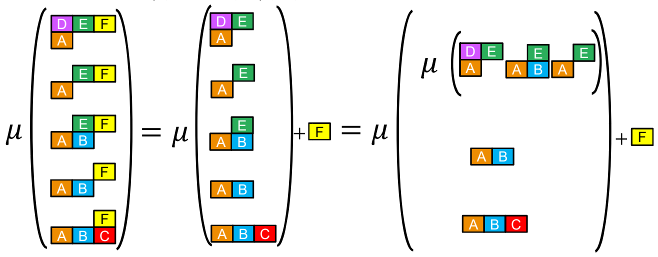
Dividing the Global Problem



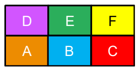
- Requirements

Distributivity:
$\mu(a+c, b+c) = \mu(a,b) + c$

Associativity:
$\mu(a,b,c) = \mu(\mu(a,b),c)$

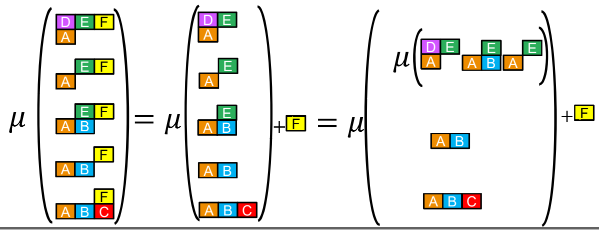


Dividing the Global Problem



Theoretical guarantees

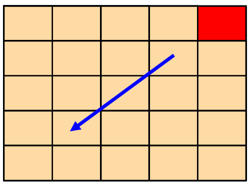
- Theorem: If μ is a regularized minimum function $\min_\Omega$, distributivity and associativity are fulfilled if and only if $\Omega(q) = \langle q, \log q \rangle$
- Global and recursive solutions are identical for $\mu = \mathrm{softmin}$!



A Recursive Algorithm for SDTW: Backward

- Follow the traditional DTW backtracking algorithm to calculate gradient matrix $\boldsymbol{H} \in \mathbb{R}^{N \times M}$
- Define gradient element $\boldsymbol{H}(n,m)$ as the probability of the minimum cost path going through cell (n,m)
- Initialize the recursion: $\boldsymbol{H}(N,M) = 1$ (all paths end in (N,M))
- Compute cells $\boldsymbol{H}(n,m)$ with a recursion in reverse order

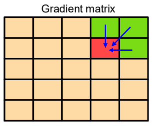
Gradient matrix $\boldsymbol{H}$



M. Cuturi and M. Blondel, „Soft-DTW: a differentiable loss function for time series, ICML 2017

A Recursive Algorithm for SDTW: Backward

Gradient matrix



Previously computed: $\boldsymbol{H}(n+i_s, m+i_j)$

Probability that following cell is part of minimum cost path

Obtain from gradient of forward step $\nabla \min_\Omega$

Probability that step of minimum cost to following cell comes from current cell

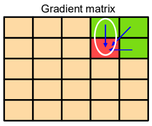
- Backward recursion: $\frac{\partial\, \mathrm{SDTW}(\boldsymbol{C})}{\partial\, \boldsymbol{D}(n,m)} = \sum_{s=1}^{S} \frac{\partial\, \mathrm{SDTW}(\boldsymbol{C})}{\partial\, \boldsymbol{D}(n+i_s, m+j_s)} \cdot \frac{\partial \boldsymbol{D}(n+i_s, m+j_s)}{\partial\, \boldsymbol{D}(n,m)}$

Sum over steps

- Recap: Forward computation
$\boldsymbol{D}(n,m) = \min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i, m-j) \mid (i,j) \in \mathcal{S}\})$
- Gradient $\nabla \min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i, m-j) \mid (i,j) \in \mathcal{S}\})$ gives the probability that the path of minimum cost to (n,m) is coming from $(n-i, m-j)$

A Recursive Algorithm for SDTW: Backward

Gradient matrix



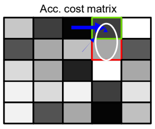
Previously computed: $\boldsymbol{H}(n+i_s, m+i_j)$

Probability that following cell is part of minimum cost path

Obtain from gradient of forward step $\nabla \min_\Omega$

Probability that step of minimum cost to following cell comes from current cell

Acc. cost matrix



Step: $(i,j) = (0,1)$

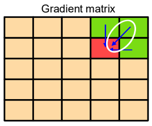
- Backward recursion: $\frac{\partial\, \mathrm{SDTW}(\boldsymbol{C})}{\partial\, \boldsymbol{D}(n,m)} = \sum_{s=1}^{S} \frac{\partial\, \mathrm{SDTW}(\boldsymbol{C})}{\partial\, \boldsymbol{D}(n+i_s, m+j_s)} \cdot \frac{\partial \boldsymbol{D}(n+i_s, m+j_s)}{\partial\, \boldsymbol{D}(n,m)}$

Sum over steps

- Recap: Forward computation
$\boldsymbol{D}(n,m) = \min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i, m-j) \mid (i,j) \in \mathcal{S}\})$
- Gradient $\nabla \min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i, m-j) \mid (i,j) \in \mathcal{S}\})$ gives the probability that the path of minimum cost to (n,m) is coming from $(n-i, m-j)$

A Recursive Algorithm for SDTW: Backward

Gradient matrix



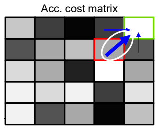
Previously computed: $\boldsymbol{H}(n+i_s, m+i_j)$

Probability that following cell is part of minimum cost path

Obtain from gradient of forward step $\nabla \min_\Omega$

Probability that step of minimum cost to following cell comes from current cell

Acc. cost matrix



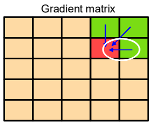
Step: $(i,j) = (1,1)$

- Backward recursion: $\frac{\partial\, \mathrm{SDTW}(\boldsymbol{C})}{\partial\, \boldsymbol{D}(n,m)} = \sum_{s=1}^{S} \frac{\partial\, \mathrm{SDTW}(\boldsymbol{C})}{\partial\, \boldsymbol{D}(n+i_s, m+j_s)} \cdot \frac{\partial \boldsymbol{D}(n+i_s, m+j_s)}{\partial\, \boldsymbol{D}(n,m)}$

Sum over steps

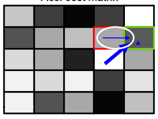
- Recap: Forward computation
$\boldsymbol{D}(n,m) = \min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i, m-j) \mid (i,j) \in \mathcal{S}\})$
- Gradient $\nabla \min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i, m-j) \mid (i,j) \in \mathcal{S}\})$ gives the probability that the path of minimum cost to (n,m) is coming from $(n-i, m-j)$

A Recursive Algorithm for SDTW: Backward



Previously computed: $\boldsymbol{H}(n+i_s, m+j_s)$ → Probability that following cell is part of minimum cost path

Obtain from gradient of forward step $\nabla\min_\Omega$ → Probability that step of minimum cost to following cell comes from current cell



Step: $(i,j) = (1,0)$

- Backward recursion: $\frac{\partial\,\mathrm{SDTW}(\boldsymbol{C})}{\partial\,\boldsymbol{D}(n,m)} = \sum_{s=1}^{S} \frac{\partial\,\mathrm{SDTW}(\boldsymbol{C})}{\partial\,\boldsymbol{D}(n+i_s,m+j_s)} \cdot \frac{\partial \boldsymbol{D}(n+i_s,m+j_s)}{\partial\,\boldsymbol{D}(n,m)}$

  Sum over steps

- Recap: Forward computation

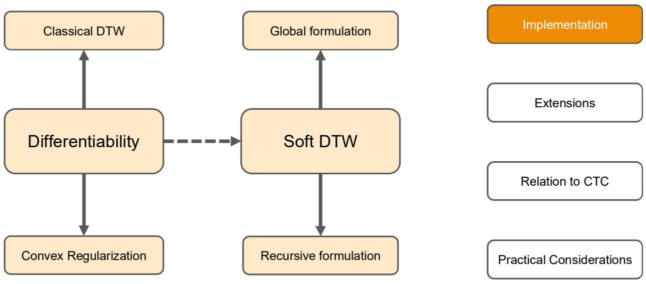
  $\boldsymbol{D}(n,m) = \min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i,m-j) \mid (i,j) \in \mathcal{S}\})$

- Gradient $\nabla\min_\Omega(\{\boldsymbol{C}(n,m) + \boldsymbol{D}(n-i,m-j) \mid (i,j) \in \mathcal{S}\})$ gives the probability that the path of minimum cost to (n,m) is coming from $(n-i,m-j)$

Summary: A Recursive Algorithm for SDTW

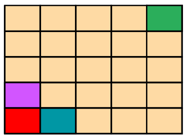
- Recursive forward pass of SDTW = „soft“ version of classical DTW forward pass (differentiable minimum instead of hard minimum)
- Recursive backward pass of SDTW = „soft“ version of classical DTW backtracking (probabilities for paths instead of hard decision)
- Recursion is identical to global formulation if $\min_\Omega = \mathrm{softmin}$
- Runtime linear in sequence lengths $\mathcal{O}(NM)$

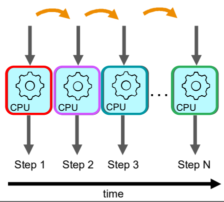
Overview

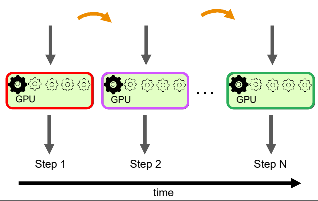


Efficient Computation

- SDTW recursion requires iterative processing
- Well-suited for CPUs
- Not efficient for GPUs

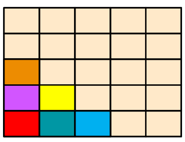


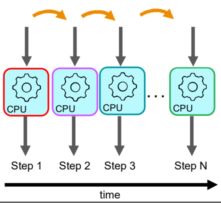


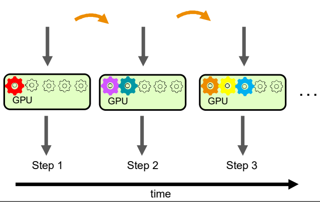


Efficient Computation

- Use parallel processing capabilities of GPU efficiently
- Group computations together
- Process along anti-diagonals

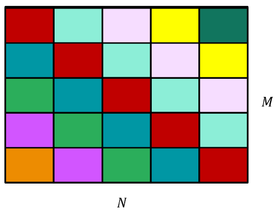






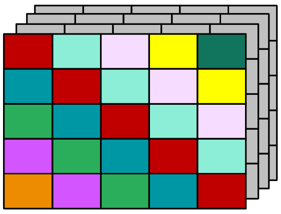
Efficiency & Implementation

- Elements along the anti-diagonals are independent of each other
- Number of “group” computations: $\#\mathrm{diag} = N + M - 1$
- Example: $N = M = 5$
  - Number of individual elements: $N \cdot M = 25$
  - Number of anti-diagonals (groups): $N + M - 1 = 9$
- The same holds for the backward pass



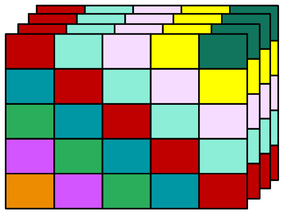
Batch Processing

- Independence along the batch dimension



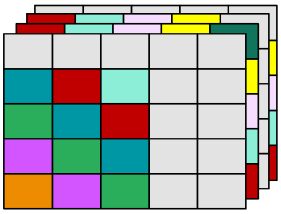
Batch Processing

- Independence along the batch dimension
- Group anti-diagonals together for all batch elements
- Number of groups doesn‘t change compared to single-matrix processing
- Batch processing over multiple cost matrices comes „free“

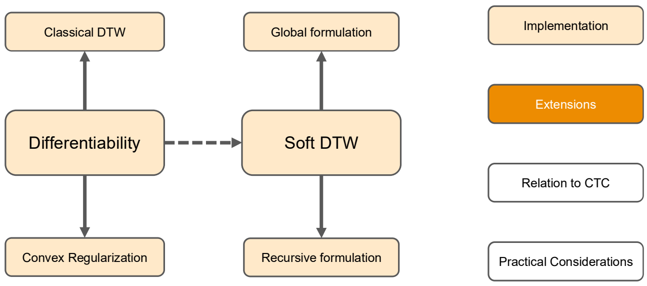


Batch Processing

- How to deal with difference sequence lengths in a batch?
- Pad all cost matrices to same size and concatenate
- Do group processing along anti-diagonals
- Skip computation if outside current sequence length



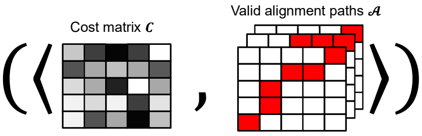
Overview



SDTW as Generalized Alignment Framework

- Objective: $\mathrm{SDTW}(\boldsymbol{C}) = \min_\Omega(\langle \boldsymbol{C}, \boldsymbol{A}\rangle \mid \boldsymbol{A} \in \mathcal{A})$

$\mathrm{SDTW}(\boldsymbol{C}) = \min_\Omega\Big(\Big\langle$

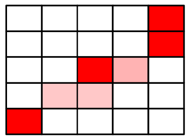


- SDTW provides an efficient framework for computing $\min_\Omega(\langle \boldsymbol{C}, \boldsymbol{A}\rangle \mid \boldsymbol{A} \in \mathcal{A})$
- We can relax constraints on the alignments $\mathcal{A}$ to make SDTW mor flexible

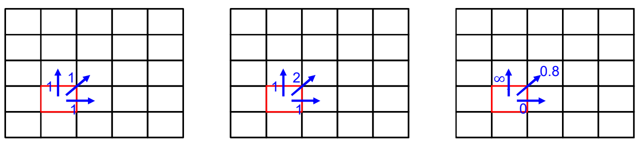
SDTW with Variable Step Weights

- Choose flexible weights for every step
- Avoid diagonatlization for equal sequence lengths
- Control influence of target repetition (horizontal step)
- Include prior knowledge on likelihood of certain steps
- Use step weight ∞ to „block“ certain steps

Example for soft alignment



J. Zeitler, M. Krause, and M. Müller, „Soft Dynamic Time Warping with Variable Step Weights“, ICASSP 2024

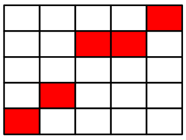


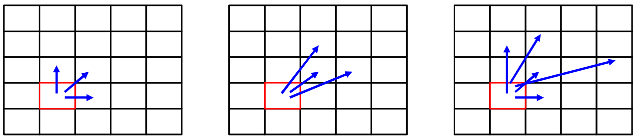
SDTW with Flexible Step Sizes

- Skip certain frames or targets
- 2-1-softDTW

J. Zeitler and M. Müller, „A Unified Perspective on CTC and SDTW using Differentiable DTW", submitted to IEEE Transactions of Audio, Speech, and Language Processing, 2025

Example for soft alignment



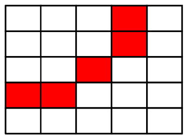


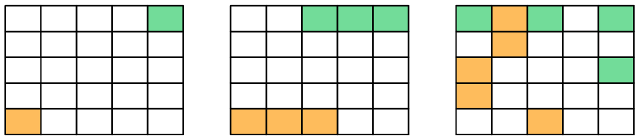
SDTW with Flexible Boundary Conditions

- Subsequence-softDTW
- Prediction and target sequences do not need to align at the boundaries

J. Zeitler and M. Müller, „Subsequence SDTW: A Framework for Differentiable Alignment with Flexible Boundary Conditions", submitted to ICASSP 2026

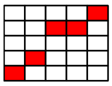
Example for soft alignment





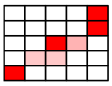
SDTW as Generalized Alignment Framework

- Flexible Step Sizes:
  - Skip certain frames or targets
  - 2-1-softDTW



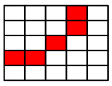
J. Zeitler and M. Müller, „A Unified Perspective on CTC and SDTW using Differentiable DTW", submitted to IEEE Transactions of Audio, Speech, and Language Processing, 2025

- Flexible Step Weights:
  - Choose flexible weights for every step
  - Avoid diagonatlization for equal sequence lengths
  - Control influence of target repetition (horizontal step)
  - Include prior knowledge on likelihood of certain steps
  - Use step weight ∞ to „block" certain steps



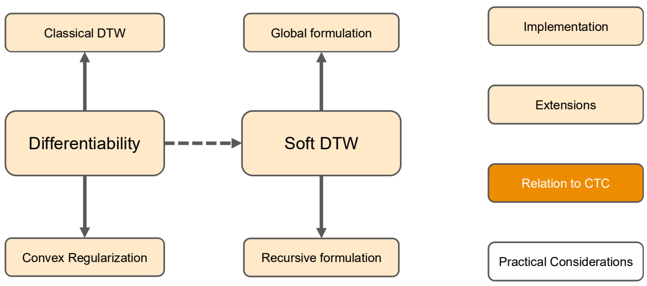
J. Zeitler, M. Krause, and M. Müller, „Soft Dynamic Time Warping with Variable Step Weights", ICASSP 2024

- Flexible Boundary Conditions
  - Subsequence-softDTW
  - Prediction and target sequences do not need to align at the boundaries



J. Zeitler and M. Müller, „Subsequence SDTW: A Framework for Differentiable Alignment with Flexible Boundary Conditions", submitted to ICASSP 2026

Overview

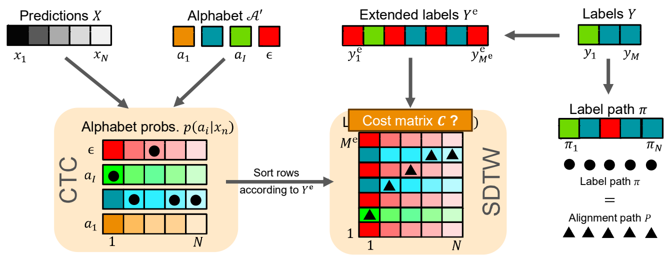


Relation to CTC

- CTC…
  - has a finite target alphabet
  - is widely used in speech processing
  - has an unintuitive formulation
- SDTW…
  - is based on an arbitrary cost matrix
  - is widely used in signal processing
  - has an intuitive formulation
- Both algorithms align sequences and are fully differentiable
- Can we establish a connection?

CTC Reformulation

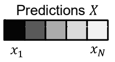
A. Graves et al., „Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks", ICML 2006

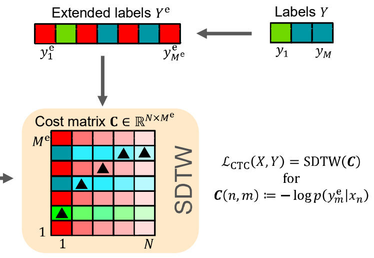


$$p(Y|X) = \sum_{\pi\in\kappa^{-1}(Y)} \prod_{n=1}^{N} p(\pi_n|x_n)$$

$$p(Y|X) = \sum_{P\in\mathcal{P}} \prod_{(n,m)\in P} p(y_m^\epsilon|x_n)$$

CTC Reformulation



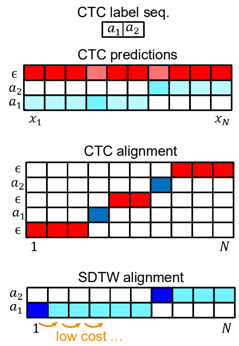


- Bring CTC and SDTW on a unified basis
- Adapt SDTW rules for alignment P: jumping of blanks is possible if adjacent label symbols are different
- Apply SDTW "tricks" to CTC
- Use CTC-like alignment for arbitrary features (e.g., real-valued labels)

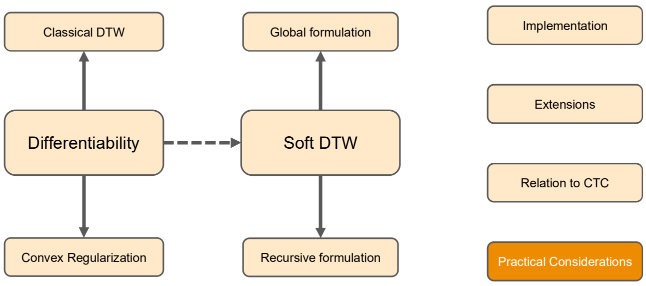
J. Zeitler and M. Müller, „A Unified Perspective on CTC and SDTW using Differentiable DTW", submitted to IEEE Transactions of Audio, Speech, and Language Processing, 2025

Dominance of Blank Symbol in CTC

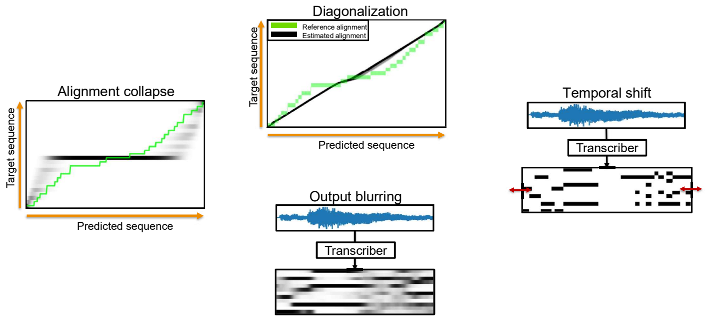
- CTC predictions dominated by blank
- Blank alignment is always "cheap" and leads to stabilization
- Spiky alignment of labels
- Predictions get even more blank-dominated
- Stabilization in SDTW: low cost for horizontal step (label repetition)
- Eliminate need for blank symbol



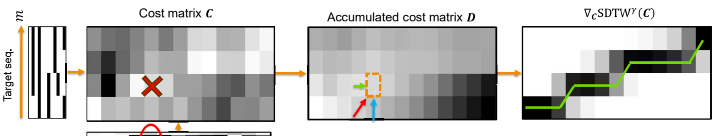
Overview



Common SDTW Problems & Pitfalls



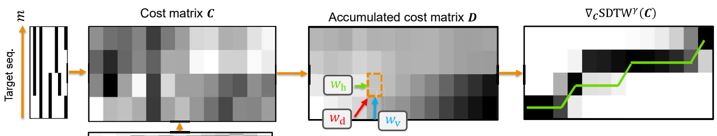
Problem 1: Alignment Collapse



$$\boldsymbol{D}(m,n) = \min{}^{\gamma}\left(\left\{\begin{array}{l}\boldsymbol{C}(m,n) + \boldsymbol{D}(m-1,n), \\ \boldsymbol{C}(m,n) + \boldsymbol{D}(m,n-1), \\ \boldsymbol{C}(m,n) + \boldsymbol{D}(m-1,n-1)\end{array}\right\}\right)$$

- Single corrupted predictions cause high values in cost matrix
- Alignment collapses to few target frames
- Training diverges

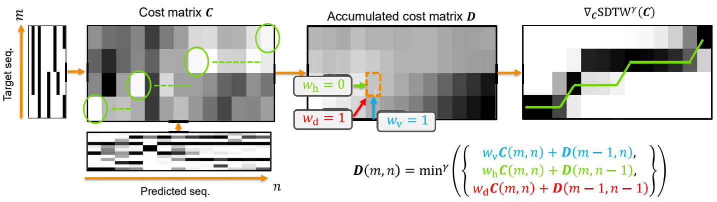
Problem 1: Alignment Collapse



$$\boldsymbol{D}(m,n) = \min{}^{\gamma}\left(\left\{\begin{array}{l}\boldsymbol{C}(m,n) + \boldsymbol{D}(m-1,n), \\ \boldsymbol{C}(m,n) + \boldsymbol{D}(m,n-1), \\ \boldsymbol{C}(m,n) + \boldsymbol{D}(m-1,n-1)\end{array}\right\}\right)$$

- Target frames are often repeated
- Reduce the influence of outliers of repeated targets
- Assign individual weight to alignment step directions

Problem 1: Alignment Collapse



$$D(m,n) = \min^\gamma \left(\left\{ \begin{array}{l} w_v C(m,n) + D(m-1,n), \\ w_h C(m,n) + D(m,n-1), \\ w_d C(m,n) + D(m-1,n-1) \end{array} \right\} \right)$$

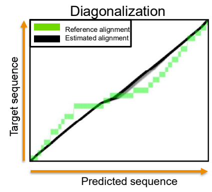
- Target frames are often repeated
- Reduce the influence of outliers of repeated targets
- Assign individual weight to alignment step directions
- Here: reduce horizontal step weight (low cost for repetition of same target)

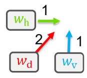
Weighted SDTW algorithm

- Efficient DP recursions for forward & backward passes
- Runtime is linear in the length of the predicted sequence (N) and the target sequence (M): $\mathcal{O}(NM)$

Problem 2: Diagonalization

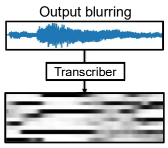
- Problem: computed SDTW alignment focuses only on the main diagonal
- Cause:
  - Equal lengths of prediction and target sequences
  - Sequences of equal length can be aligned using only diagonal steps
  - Taking one diagonal step is cheaper than taking a vertical and horizontal step („around the corner")
- Solution:
  - Choose SDTW with step weights
  - Set a higher step weight to diagonal step (e.g., 1-1-2)
  - A diagonal step gets the same weight as a horizontal + vertical step

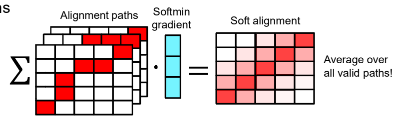




Problem 3: Output Blurring

- Problem: transcriber learns only blurry features
- Cause:
  - Softmin temperature $\gamma \rightarrow \infty$
  - Softmin becomes averaging
  - SDTW gradient is average over all paths
  - Blurry gradient leads to blurry features
- Solution:
  - Reduce softmin temperature $\gamma \approx 1$
  - If high softmin temperature is necessary in initial training, do gradual reduction

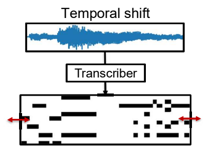




J. Zeitler and M. Müller, „Reformulating Soft Dynamic Timewarping: Insights into Target Artifacts and Prediction Quality", ISMIR 2025

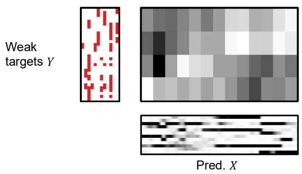
Problem 4: Temporal Shift

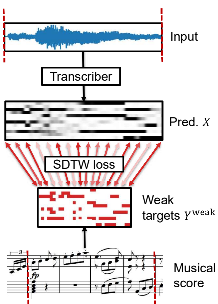
- Problem: small temporal shift between input and predictions
- Cause:
  - SDTW computes flexible alignment between predictions and weak targets
  - Alignment cost is invariant of (small) temporal shift
- Solutions:
  - Identify temporal shift of trained model and compensate during inference
  - Use a DNN with small temporal receptive field (1-1 mapping of input to output frames)
  - Use an auxiliary loss to evaluate smiliarity between the predictions and the input



Multi-Pitch & Pitch Class Estimation

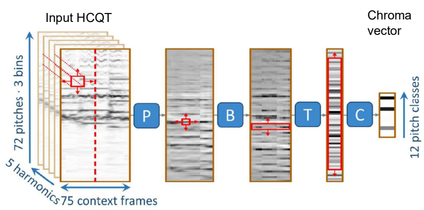
- Annotate corresponding segments in the input audio and the musical score (typically 10s – 30s)
- Retrieve weak targets from the musical score
- Weak targets represent sequence of simultaneously active notes, but no information about duration
- Cost function: Binary Cross-Entropy (BCE)
- $C(n,m) = BCE(x_n, y_m)$

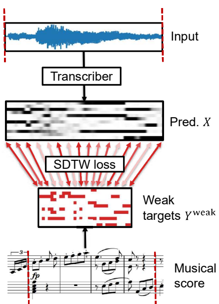




Multi-Pitch & Pitch Class Estimation

Parameter-efficient choice for deep learning of pitch (class) activations: musically motivated CNN [Weiss2021]

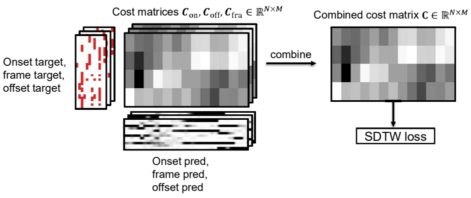


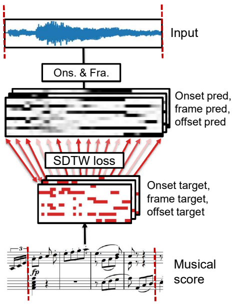


C. Weiß, J. Zeitler, T. Zunner, L. Brütting, and M. Müller: "Learning Pitch-Class Representations from Score-Audio Pairs of Classical Music", ISMIR 2021

Transcription with Multiple Features

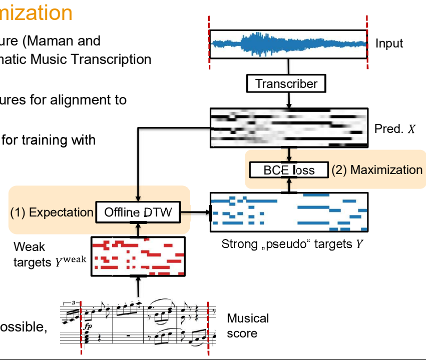
- Can use full transcription model like Onsets & Frames
- Use separate cost functions for onsets, frames, offsets
- Combine (add) all cost matrices into a single cost matrix and perform standard SDTW





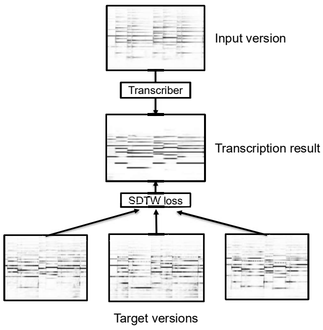
Relation to Expectation-Maximization

- Train on unaligned data with an EM-procedure (Maman and Bermano, „Unaligned Supervision for Automatic Music Transcription In-the-Wild“, ICML 2022)
- Expectation: use current predictions as features for alignment to weak targets using offline DTW
- Maximization: use aligned “pseudo” targets for training with element-wise loss function
- Interpretation:
  - a “hard” alignment is computed between predictions and labels
  - This hard alignment is used for training
- SDTW analogy:
  - Use SDTW with hardmin as diff. minimum function
- Limitations: no alignment post-processing possible, e.g., note snapping



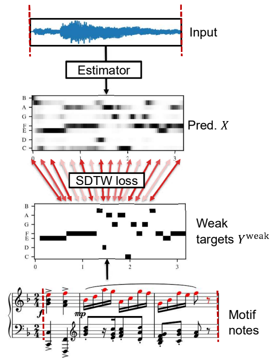
Cross-Version Training

- Semi-supervised training using cross-version information
- M. Krause et al., „Weakly supervised multi-pitch estimation using cross-versiong alignment“, ISMIR 2023
- Train without pitch annotations
- All versions are based on the same musical score
- Transcriber learns musical score implicitely



Enhancement of Motifs

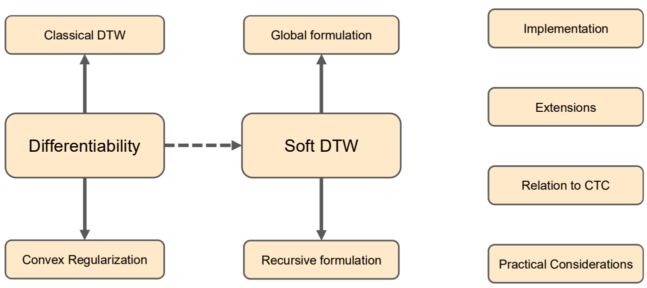
- Goal: enhance salience of certain musical structures, like melody or motifs
- Annotation: separately annotate motif notes in the musical score (see, e.g., BPS-motif)
- Represent motif notes as weak targets
- Train DNN to predict features that minimize SDTW distance to the weak targets



References

- S. Boyd and L. Vandenberghe, "Convex Optimization", Cambridge University Press, 2004
- A. Graves et al., „Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks“, ICML 2006
- M. Cuturi and M. Blondel, „Soft-DTW: a differentiable loss function for time series“, ICML 2017
- A. Mensch and M. Blondel, "Differentiable dynamic programming for structured prediction and attention", ICML 2018
- D. Stoller et al., „End-to-end lyrics alignment for polyphonic music an audio-to-character recognition model“, ICASSP 2019
- C. Wigington et al., „Multi-label connectionist temporal classification, ICDAR 2019
- F. Zalkow and M. Müller, „CTC-based learning of chroma features for score-audio music retrieval, IEEE TASLP 2021
- C. Weiß et al., "Learning Pitch-Class Representations from Score-Audio Pairs of Classical Music", ISMIR 2021
- C. Weiß and G. Peeters, „Learning multi-pitch estimation from weakly aligned score-audio pairs using a multi-label CTC loss“, WASPAA 2021
- M. Blondel et al., "Differentiable divergences between time series", AISTATS 2021
- J. Zeitler et al., "Stabilizing training with soft dynamic time warping: A case study for pitch class estimation with weakly aligned targets", ISMIR 2023
- J. Zeitler et al., "Soft dynamic time warping with variable step weights", ICASSP 2024
- M. Blondel and V. Roulet, "The elements of differentiable programming", arxiv preprint, 2025
- J. Zeitler and M. Müller, „A Unified Perspective on CTC and SDTW using Differentiable DTW“, submitted to IEEE Transactions of Audio, Speech, and Language Processing, 2025
- J. Zeitler and M. Müller, "Reformulating soft dynamic time warping: insights into target artifacts and prediction quality", ISMIR 2025
- J. Zeitler and M. Müller, „Subsequence SDTW: A Framework for Differentiable Alignment with Flexible Boundary Conditions“, submitted to ICASSP 2026

Overview



APPENDIX

Differentiable via Convex Regularization
Convex Optimization

- Let $f: \mathbb{R}^D \to \mathbb{R} \cup \{\infty\}$ denote a function with domain $\mathrm{dom}(f) := \{x | f(x) < \infty\}$
- Definition of convex conjugate: $f^*(y) := \sup_x \big(\langle x, y\rangle - f(x)\big)$; $\mathrm{dom}(f^*) := \{y | f(y) < \infty\}$
- Define Indicator function $I_{\mathcal{C}}(x) := \begin{cases} 0, & x \in \mathcal{C} \\ \infty, & x \notin \mathcal{C} \end{cases}$
- Choose $f(x) = \max(x)$
- $f^*(y) = \sup_x \underbrace{(\langle x, y\rangle - \max(x))}_{=\begin{cases} 0, & \text{if } y \in \Delta^D \\ \infty & \text{else} \end{cases}} = I_{\Delta^D}(y)$

Differentiable via Convex Regularization
Convex Optimization

A. Mensch and M. Blondel, "Differentiable dynamic programming for structured prediction and attention", ICML 2018

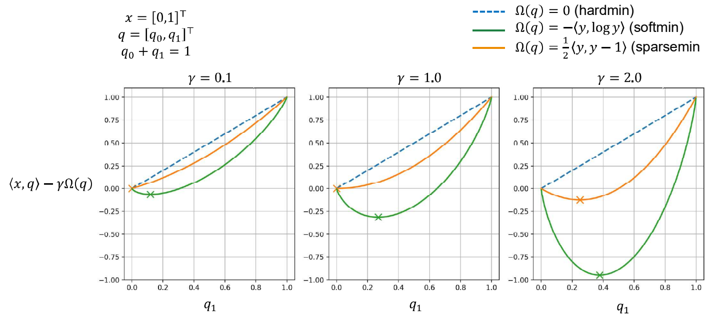
- Theorem: f^* is convex, even if f is non-convex
- Theorem: If f is strongly convex over $\mathrm{dom}(f)$, then f^* is smooth over $\mathrm{dom}(f^*)$
- Add a strongly convex regularizer $\Omega(q)$:
- $f_\Omega^*(q) = I_{\Delta^D} + \Omega(q)$
- Transform to primal space:
- $f_\Omega^{**}(x) = \sup_q \underbrace{\big(\langle x, q\rangle - I_{\Delta^D} - \Omega(q)\big)}_{=\begin{cases} -\infty & \text{if } q \notin \Delta^D \\ \langle x,q\rangle - \Omega(q) & \text{if } q \in \Delta^D \end{cases}} = \max_{q \in \Delta^D}\big(\langle x, q\rangle - \Omega(q)\big) = \max_\Omega(x)$
- $\max_\Omega(x)$ is now smooth, i.e., has a continuous derivative
- As $\max(x) = \max_{q \in \Delta^D}\langle x, q\rangle$, the function $\max_\Omega(x)$ can be seen as the max function plus an additional regularizer
- For minimum functions, we analogously have $\min_\Omega(x) = -\max_\Omega(-x)$
- Add a temperature parameter γ: $\max_\Omega^\gamma := \max_{q \in \Delta^D}\big(\langle x, q\rangle - \gamma\Omega(q)\big)$

Differentiable via Convex Regularization
Common convex regularizers Ω:

- Shannon entropy: $\Omega(y) = -\langle y, \log y\rangle$
  - Solving for optimum yields closed-form "softmin" $\min_{\text{soft}}^\gamma(x) = -\gamma \log \sum_i \exp\left(-\frac{x_i}{\gamma}\right)$
  - … with gradient $\left[\nabla \min_{\text{soft}}^\gamma\right]_i = \frac{\exp\left(-\frac{x_i}{\gamma}\right)}{\sum_j \exp\left(-\frac{x_j}{\gamma}\right)}$
- Gini entropy: $\Omega(y) = \frac{1}{2}\langle y, y - 1\rangle$
  - Solving for optimum yields "sparsemin": $\min_{\text{sparse}}^\gamma(x) = \langle y^*, x\rangle + \frac{\gamma}{2}\|y^*\|_2^2 - \frac{\gamma}{2}$
  - … with gradient $\nabla \min_{\text{sparse}}^\gamma(x) = \arg\min_{y \in \Delta^D} \left\|y + \frac{x}{\gamma}\right\|_2^2 = y^*$

Minimum functions with convex regularization

$x = [0,1]^\top$
$q = [q_0, q_1]^\top$
$q_0 + q_1 = 1$



Minimum functions with convex regularization