Self-Organizing Maps & Learning Vector Quantization for Feature Sequences

Panu Somervuo and Teuvo Kohonen

Presented By: Aditya Sharma
Different approaches to feature sequences and the SOM

• Approach 1: Use each short time feature vector in the input sequence as the input to the SOM.
  – Doesn’t reflect sequential information at all.

• Approach 2: Use context vectors derived from a segment of the input sequence.
  – Fails to remove the temporal dependencies.

• Approach 3: Use feature space trajectories derived from the whole sequence.
Different approaches to feature sequences and the SOM

Figure 1. SOMs can be constructed for different abstraction levels of the input. This diagram illustrates what kind of model can be associated with each SOM node.
Dynamic time warping

Time series are expected to vary not only in terms of expression amplitudes, but also in terms of time progression since real-time processes may unfold with different rates in response to different experimental conditions.
Why Dynamic 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.
To find the best alignment between $\mathcal{A}$ and $\mathcal{B}$ one needs to find the path through the grid $P = p_1, \ldots, p_s, \ldots, p_k$ where $p_s = (i_s, j_s)$ which minimizes the total distance between them. $P$ is called a warping function.
**Time-Normalized Distance Measure**

**Time-normalized distance** between \( \mathcal{A} \) and \( \mathcal{B} \):

\[
D(\mathcal{A}, \mathcal{B}) = \left[ \frac{\sum_{s=1}^{k} d(p_s) \cdot w_s}{\sum_{s=1}^{k} w_s} \right]
\]

\( d(p_s) \): distance between \( i_s \) and \( j_s \)

\( w_s > 0 \): weighting coefficient.

**Best alignment path** between \( \mathcal{A} \) and \( \mathcal{B} \):

\[
P_0 = \arg\min_{p} (D(\mathcal{A}, \mathcal{B})).
\]
Optimisations to the DTW Algorithm

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

Restrictions on the warping function:
- monotonicity
- continuity
- boundary conditions
- warping window
- slope constraint.

reduction of the search space
The Choice of the Weighting Coefficient

Time-normalized distance between $\mathcal{A}$ and $\mathcal{B}$:

$$D(\mathcal{A}, \mathcal{B}) = \min_p \frac{\sum_{s=1}^{k} d(p_s) \cdot w_s}{\sum_{s=1}^{k} w_s}.$$ 

Seeking a weighting coefficient function which guarantees that:

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

is independent of the warping function. Thus

$$D(\mathcal{A}, \mathcal{B}) = \frac{1}{C} \min_p \left[ \sum_{s=1}^{k} d(p_s) \cdot w_s \right]$$

can be solved by use of dynamic programming.
Symmetric DTW Algorithm
(warping window, no slope constraint)

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

DP-equation:

\[
\begin{align*}
g(i, j) &= \min \left\{ g(i-1, j-1) + 2d(i, j), \right. \\
g(i-1, j) + d(i, j), g(i, j-1) + d(i, j) \left. \right\}.
\end{align*}
\]

Warping window: \( j - r \leq i \leq j + r \).

Time-normalized distance:

\[
D(\mathcal{A}, \mathcal{B}) = \frac{g(n, m)}{C}
\]

\[
C = n + m.
\]
Asymmetric DTW Algorithm
(warping window, no slope constraint)

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

DP-equation:

$$g(i,j) = \min \left\{ g(i-1,j-1) + d(i,j), \ g(i-1,j) + d(i,j) \right\}$$

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

Time-normalized distance:

$$D(\mathcal{A}, \mathcal{B}) = g(n, m) / C$$

$C = n$. 
Quazi-symmetric DTW Algorithm
(warping window, no slope constraint)

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

DP-equation:

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

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

Time-normalized distance:

$$D(\mathcal{A}, \mathcal{B}) = \frac{g(n,m)}{C}$$

$$C = n + m.$$
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.
LEARNING VECTOR QUANTIZATION AND EXPERIMENTS
The Learning Algorithm

• Use one sequence from each class as a prototype for that class.
• Use this prototype to “warp” all other members of that class and create their respective average path functions (which now resemble the prototype).
• The prototype can be updated sequentially or in batch (just like in other clustering algorithms).
• Use the newly created average path functions as inputs to the network.
• The BMU now will resemble a sequence of feature vectors rather than a single feature vector.
Figure 3. Prototype sequences of a DTW-SOM for 10-dimensional input feature vector sequences. The map size is 4×5 units. The initial map is shown on the top and the organized map after training is shown on the bottom. Dark shades of gray indicate low values and light shades of gray indicate high values of feature vector components, respectively. In phone /s/, the first cepstrum coefficient is a large negative number. This is shown as a black spot in the first feature vector component in the prototype sequences of the organized map.
Conclusions

• SOM (and consequently LVQ) can be used to detect and classify temporal sequences of feature vectors.

• Due to the DTW, both durational differences in the input sequences as well as spatial variances in the feature vectors can be tolerated.

• Application areas may include speech processing, natural handwriting processing, and process monitoring.
Restrictions on the Warping Function

**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.
Restrictions on the Warping Function

**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:** \(|i_s - j_s| \leq r\), 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.
**Restrictions on the Warping Function**

**Slope Constraint:** \( \frac{j_s^p - j_{s0}}{i_s^p - i_{s0}} \leq p \) and \( \frac{i_s^q - i_{s0}}{j_s^q - j_{s0}} \leq q \), where \( q \geq 0 \) is the number of steps in the \( x \)-direction and \( p \geq 0 \) is the number of steps in the \( y \)-direction.

After \( q \) steps in \( x \) one must step in \( y \) and vice versa: \( S = \frac{p}{q} \in [0, \infty] \).

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.
DTW Algorithm: Example

**Time Series A**

| 0.51 | 0.51 | 0.49 | 0.49 | 0.35 | 0.17 | 0.21 | 0.33 | 0.41 | **0.49** |
| 0.27 | 0.27 | 0.26 | 0.25 | 0.16 | **0.18** | **0.23** | **0.25** | **0.31** | 0.68 |
| 0.13 | 0.13 | 0.13 | 0.12 | **0.08** | 0.26 | 0.40 | 0.47 | 0.49 | 0.49 |
| 0.08 | 0.08 | 0.08 | **0.08** | 0.10 | 0.31 | 0.47 | 0.57 | 0.62 | 0.65 |
| 0.06 | 0.06 | **0.06** | 0.07 | 0.11 | 0.32 | 0.50 | 0.60 | 0.65 | 0.68 |
| 0.04 | **0.04** | 0.06 | 0.08 | 0.11 | 0.32 | 0.49 | 0.59 | 0.64 | 0.66 |
| **0.02** | 0.05 | 0.08 | 0.11 | 0.13 | 0.34 | 0.49 | 0.58 | 0.63 | 0.66 |

**Time Series B**

Euclidean distance between vectors