T-61.181 Special course in Information Science

TIME IN SELF-ORGANIZING MAPS

N. Mozayyani and N. Alanou and J.F. Dreyfus and G. Vaucher
Spatio-temporal Data-coding Applied to Kohonen Maps
ICANN-1995

K. Torkkola
WarpNet: Self-Organizing Time Warping
WSOM-1997

Reviewed by Ramūnas Girdziušas
email: ramunasg@cc.hut.fi
28th October 2002

1 Introduction

The self organizing map (SOM) [2] is an array of the competing neurons that maps multidimensional spatial data into one or two-dimensional output structure. Such a map can be useful for data clustering and visualization purposes. Below we present short overview of the papers that propose several methods to use SOM with the data that has a spatio-temporal nature.

2 A Spatio-temporal Data-coding Applied to Kohonen Maps [3]

2.1 Encoding time dependent events as complex numbers

[3] have proposed the method to encode the time dependent data explicitly by extending the field of the SOM inputs from the real domain $\mathcal{R}$ into the complex plane $\mathcal{C}$.

Each event is represented as a complex number

$$x(\tau) = \rho(\tau)e^{i\phi(\tau)} = \rho(\tau)e^{i\arctan(\mu\tau)}, \tag{1}$$

where μ constant. In order to establish uniqueness between the time variable τ and the phase ϕ, the past events with the negative time are restricted to be in the interval $(-\frac{\pi}{2}, 0)$ while the positive time values (future events) are in $(0, +\frac{\pi}{2})$. The data samples are restricted to be always positive $\rho > 0$. The method is shown in Figure 1.

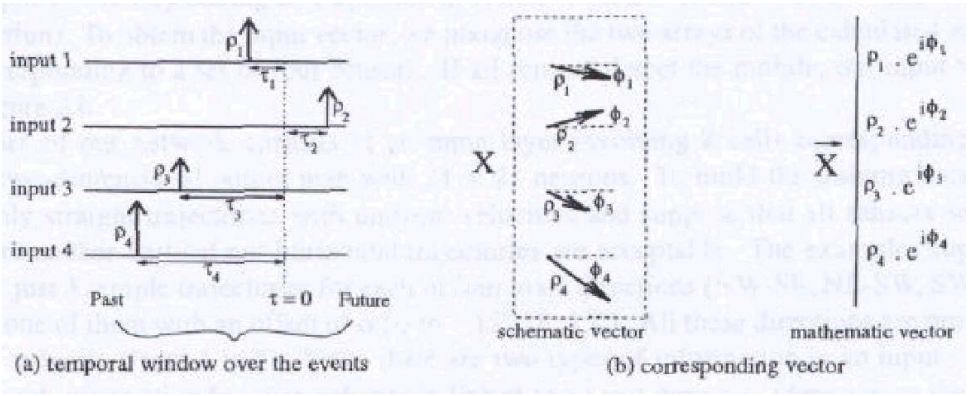

Figure 1: Vectorial representation of events.

Here one can see that the temporal sequence subject to clustering (detection, visualization) is passed through the tapped delay line thereby obtaining a sequence of the complex valued vectors that are further used as feature vectors with SOM.

2.2 Use of the SOM

The best matching unit is determined by using the Hermitian distance:

$$\delta(\mathbf{x}, \mathbf{w}_i)^2 = (\mathbf{x} - \mathbf{w}_i)^T (\mathbf{x} - \mathbf{w}_i)^*, \tag{2}$$

where $\mathbf{w}_i$ is an element of the SOM array, $\mathbf{x}$ is a complex feature vector and '$*$' denotes the complex conjugation.

2.3 Simulation results

Such an encoding method has been applied to detect velocity of the object moving in the plane. This problem is explained in Figure 2.

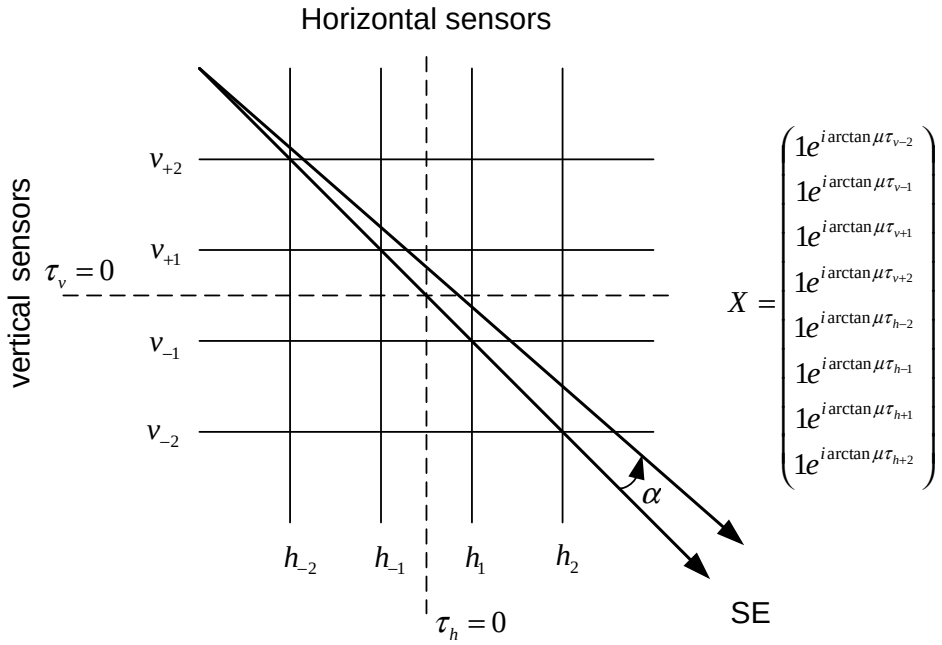

Figure 2: Motion detection problem: (a) sensors and (b) encoded SOM input vector.

The authors chose SOM with 21×21 neurons. 8-dimensional input vector consisted of unit magnitude complex numbers whose phases were estimated by substituting sensor times into Eq. (2). Training data was generated by moving object in NW-SE, NE-SW, SW-NE and SE-NW directions with a slight variations by the angles $\alpha = -15^o, 0^o, 15^o$. Each trajectory was repeated for the five different velocity values. SOM was trained by using 1000 iterations, the resulting best-matching units are shown in Fig. 3.

The authors also experimented by adding random 20% velocity perturbations and setting complex feature vector magnitudes to zero. It was claimed that even under such circumstances SOM clusters the directions reliably.

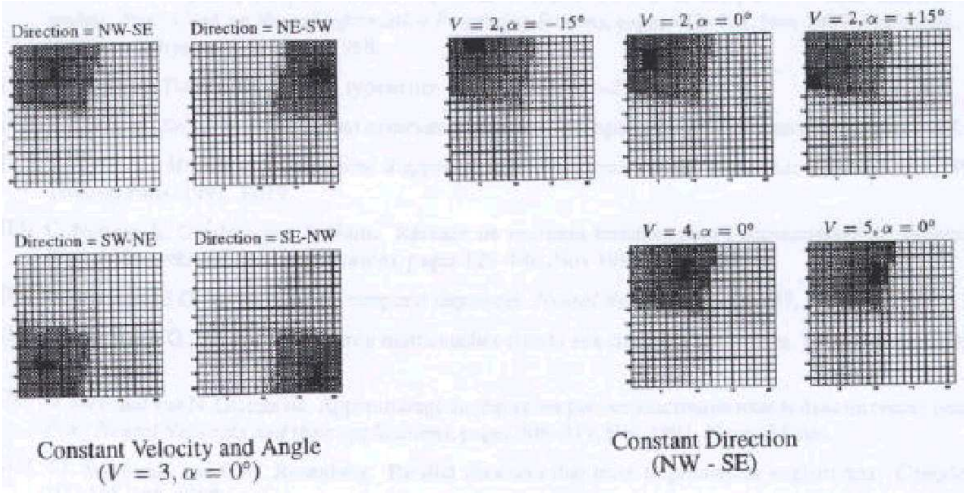

Figure 3: Simulation results.

2.4 Critical appraisal

The idea of the paper is that instead of simple time delay lattices, it is possible to consider the complex phasors, or two harmonic signals, whose magnitudes are modulated by data. The proposed method to encode time is simple and easily programmable because standard SOM training can be used if one replaces the complex vectors by stacking their real and imaginary parts into one single feature vector. The problem is that the authors do not present any comparative analysis with the other approaches, such as SOM used with the time delay lines. Another difficulty with the method is that the time encoding is data independent and otherwise very arbitrary. One can view the complex feature vector as two real feature vectors, weighted by harmonic signals whose phase differs by $\frac{\pi}{2}$. There are infinitely many possible weighting functions, and one particular choice needs more justification.

With regards to the use of the method to detect velocity, the authors only show that under 'academic' noise conditions SOM clusters motions of 4 different directions very well. Determination of the precise arbitrary direction by using such a map is not obvious at all.

Finally, I would like to add that one could improve the method by using the hyper-complex numbers such as quaternions, where an input sequence could be enriched by adding a local frequency and some other signal characteristics.

3 WarpNet: Self-Organizing Time Warping [4]

The author of this paper develops one of many existing sequence warping algorithms based on the elastic matching concept. His algorithm is meant to oppose the so called dynamic time warping principle (DTW). To explain the authors idea, let us first briefly

review the DTW principle.

4 Dynamic time warping

The material in this section is thoroughly based on my thesis [1]. Consider two time signals $U = \{u_1, u_2, \ldots, u_n\}$ and $V = \{v_1, v_2, \ldots, v_n\}$. The sequence matching problem can be stated as to find the the optimal path $P_H = \{(i(h), j(h)), h = 1, \ldots, H\}$, along which the distance

$$D = \sum_{h=1}^{H} (u_{i(h)} - v_{j(h)})^2, \tag{3}$$

is minimal. The path relates each time sample of one sequence to the 'optimal' sample in another time sequence. **If one considers all possible warps, the complexity of such a combinatorial solution method is** $O(m^n)$.

The main idea in finding the optimal path P_H by using DTW is the utilization of dynamic programming, which states that the distance corresponding to the best sub-path ending at the point pair (u_i, v_j) can be expressed recursively:

$$D(u_i, v_j) = \min_k \{D(u_{i-1}, v_k) + d(u_{i-1}, v_k, u_i, v_j)\}, \tag{4}$$

where $d(u_{i-1}, v_k, u_i, v_j)$ is a cost associated with the path transition from pair (u_{i-1}, v_k) to (u_i, v_j). We consider only discrete DTW, so the u and v variables can be dropped from the notation. **Dynamic programming algorithm solves the problem in** $O(mn^2)$ **computational complexity.**

One of the fast forms of DTW is obtained by introducing a continuity constraint of the form $k \in \{j, j-1\}$. It is assumed that the cost of transition is constant for each k and simply equal to the distance between the two points in the pair $d(i-1, k, i, j) = d(i, j)$. The DTW recursion is then

$$D(i, j) = \min \left\{ \begin{array}{l} D(i-1, j) \\ D(i-1, j-1) \\ D(i, j-1) \end{array} \right\} + d(i, j), \tag{5}$$

The recursion (5) together with the boundary conditions $(i(1), j(1)) = (1, 1)$ and $(i(H), j(H)) = (n, m)$ constitutes the basis of the DTW algorithm. In words, the application of DTW proceeds as follows: construct a table by labeling the columns as the indices of the first stroke points and rows as the indices of the second stroke points. Then obtain the costs for the first row and the first column assuming that $D(0, 0) = D(0, 1) = D(1, 0) = 0$ and then apply the recursion (5). An example of DTW matching is shown in Figure 4.

Figure 4a shows the point correspondence when the beginning and the end of the sequences is guessed to be right and Figure 4b is the case when the sequence start and end points are determined wrongly. The optimal paths for both cases are shown in Figure 5.

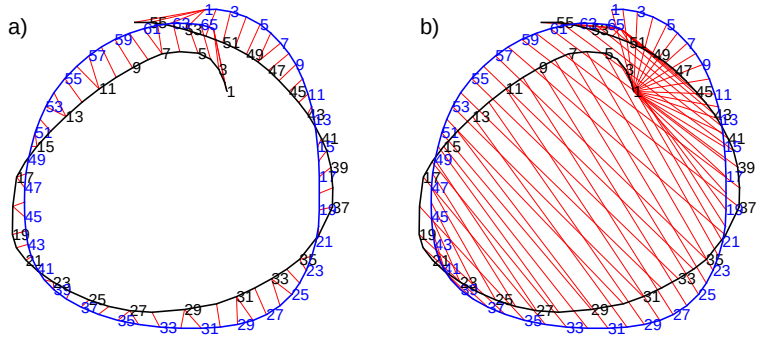

Figure 4: DTW sequence matching: point correspondences. a) is the case when the beginning and the end of the sequences are correct, b) is the case when the point order in the sequences is guessed wrongly.

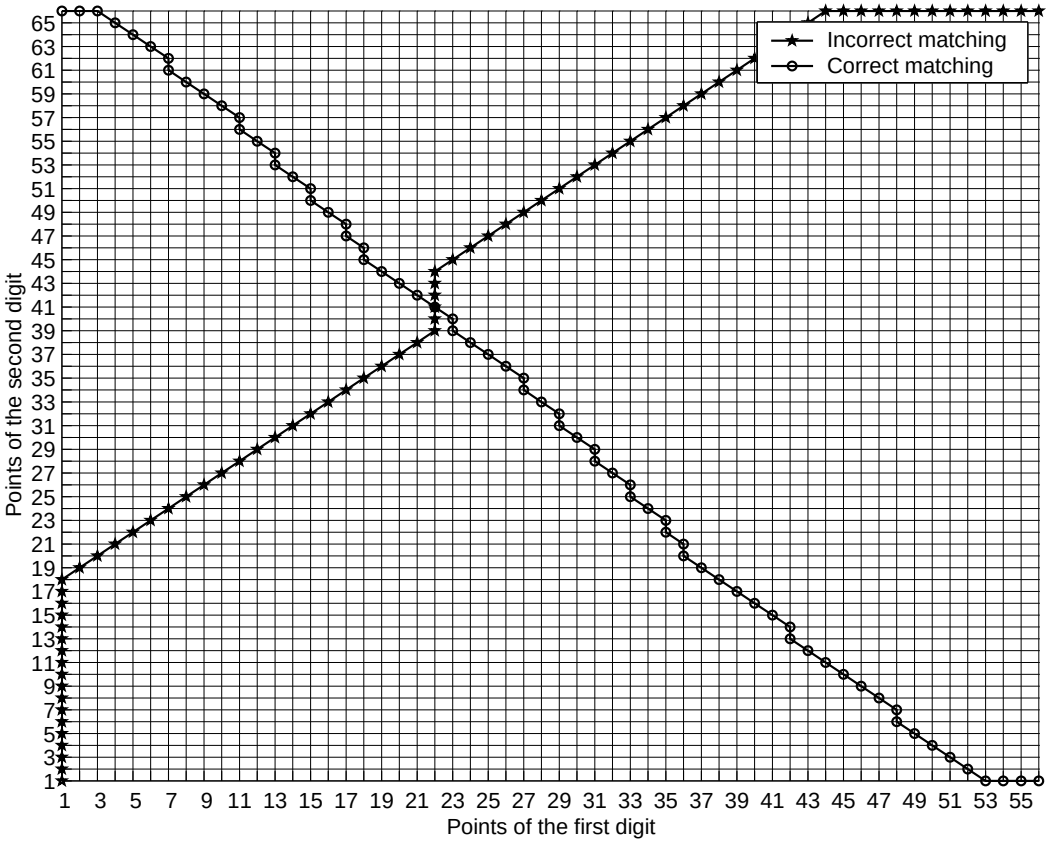

Figure 5: DTW sequence matching: optimal paths in opposite directions corresponding to the matchings shown in Figure 4. One can see that the optimal path lies not far from the diagonal line joining the first and the last matching pair of points.

The complexity of sequence matching by using DTW embedded in recursion (5) is only $O(nmc)$, where $c \ll n$ is the number of movements allowed by the continuity constraints, such as in (5).

4.1 WarpNet

The author of [4] considers the following approach to warp two given sequences:

1. Establish the initial warp as a diagonal path shown in Fig. 5.
2. For each warping sequence point estimate the best-matching sample in the template sequence. Best candidates are considered only from a pre-defined neighborhood. Estimate the time displacement to such a best-matching sample.
3. Smooth the displacement sequence by using, for example, the Gaussian filter.
4. Multiply the smoothed displacement by some fixed constant, called 'plasticity', which imposes global constraint on warping magnitude.
5. Repeat the previous iterations until a smoothed direction array does not change. Sample neighborhood, plasticity and smoothing filter window should be decreased after each iteration.

Figure 6 shows the matching curves and several example directions for relatively small neighborhood. This corresponds to one of the final iterations.

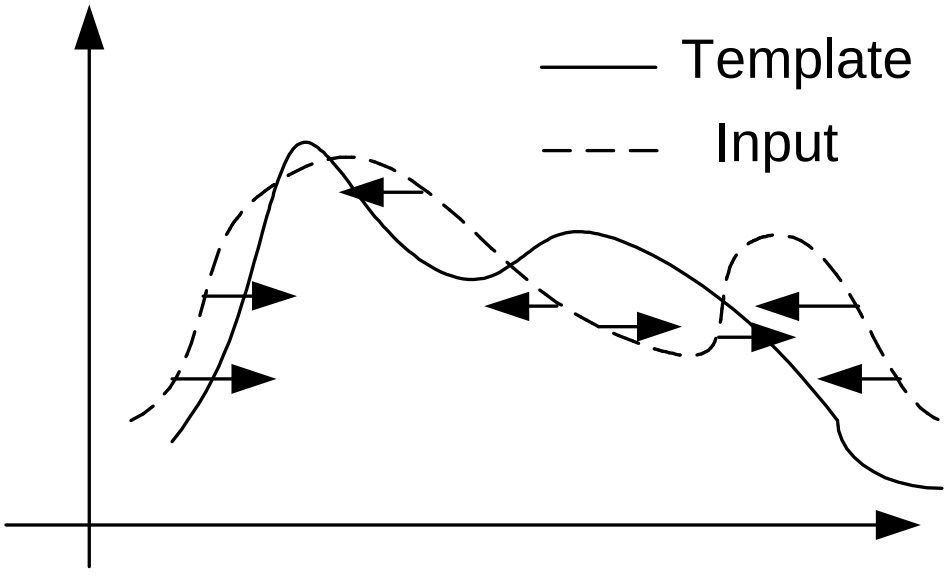

Figure 6: The principle of WarpNet.

The author compares the algorithm with the DTW method in speech recognition task and concludes that the proposed method has a comparable speed and performance and could be considered as an alternative to DTW approach.

5 Critics

The introduction of neighborhoods, in principle, corresponds to the DTW principle with a global path constraint:

$$\frac{m}{n}i - b \cdot m \leq j \leq \frac{m}{n}i + b \cdot m, \tag{6}$$

where b is a so-called pruning parameter. The special cases considered above are obtained when $b = 0$ (LTW) and $b = 1$ (DTW).

The difference is that the WarpNet algorithm does not employ any inherent to DTW recursions, and at first glance this should save computational resources significantly. The author states the complexity $O(mn)$ vs. $O(n)$ but to be more precise, it is $O(\eta mn)$ for DTW and $T \times O(\mu(m)n)$ for the WarpNet algorithm. It is important that η denotes the number of constraints in DTW and usually it is a very small number. However, μ is not fixed. In the beginning, when large plasticity and neighborhood size values are used, the complexity of a newly proposed algorithm (neglecting T) is the same as in DTW: $O(\mu(m)n) = O(mn) \approx O(\eta mn)$. Gradually it decreases down to $O(n)$, but the warping is performed many times, whereas DTW warping is performed only once! Torkkola's algorithm might have lower complexity only in case neighborhood sizes initially are already small, but then the algorithm would be essentially limited to only small warps.

The author's statement about the DTW optimality is misleading. An optimal method for the experiments would be a complete dynamic programming algorithm whose complexity is $O(mn^2)$. DTW, however, employs further approximations by constraining movements in the path table as shown in Eq. 5 and it is in no way optimal. An example given in Fig. 4 elaborates the idea that DTW is unable to find a proper reverse transformation in time. Many other situations when it is sub-optimal exists.

The only advantage of the proposed method is that it is not recursive and the only situation where it might be useful is the case of very long time sequences, warping of images or 3D data.

References

[1] R. Girdziušas. Discriminative on-line recognition of isolated handwritten characters. Master's thesis, Helsinki University of Technology, 2001.

[2] T. Kohonen. *Self-Organizing Maps*. Springer, 3rd edition, 2001.

[3] N. Mozayyani, N. Alanou, J.F. Dreyfus, and G. Vaucher. A spatio-temporal data-coding applied to kohonen maps. *ICANN*, 2:75–79, 1995.

[4] K. Torkkola. Warpnet: self-organizing time warping. *WSOM*, 1:169–174, 1997.