T-61.181 Special Course in Information Science I
Models for temporal sequence processing

Juho Kannala
jkannala@cc.hut.fi
50106D
TFY IV

November 10, 2002

Contents

1 Introduction

In this report we first introduce two SOM-models for temporal sequence processing. These are the temporal Kohonen map (TKM) and the recurrent self-organizing map (RSOM). TKM was first proposed by Chappell and Taylor [4] and it inspired the RSOM-model [1]. In both models each neuron of an ordinary SOM is supplied with a leaky integrator that gives a "memory" to the map. However the details of the learning algorithms of the models are different. In Section 2.3 we compare the properties of TKM and RSOM following the treatment in [2].

The second part of the report discusses models proposed by Privitera and Morasso for generating and learning of temporal sequences [5, 6]. The models contain two dimensional maps of neurons with some kind of leaky integrators. In [7] the models have been applied in generating and learning of rapid movement sequences.

2 Two SOM-models for temporal sequence processing

An ordinary SOM consists of a two-dimensional grid of nodes. Each node contains a weigth vector $\mathbf{w}_i$ of the same dimension as the input space. In the basic SOM algorithm the weight vectors are modified on the basis of the input samples. At the end of the learning phase the distribution of the weight vectors approximates the input distribution and the topology of the input space is (partly) preserved on the two-dimensional map. [3].

In the iterative SOM algorithm the input vectors $\mathbf{x}(n)$ are presented to the map one at a time. For each input a best matching unit (*bmu*) $\mathbf{w}_b$ is chosen as

$$||\mathbf{x}(n) - \mathbf{w}_b|| = \min_i ||\mathbf{x}(n) - \mathbf{w}_i|| \tag{1}$$

and the weights are updated by

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \alpha(n) h_{ib}(n)(\mathbf{x}(n) - \mathbf{w}_i(n)), \tag{2}$$

where $\alpha(n)$ is the monotonically decreasing learning rate factor and h_{ib} is the neighbourhood function [3].

If all input samples are known in advance it is possible to use the batch version of the SOM algorithm to update the weights:

$$\mathbf{w}_i = \sum_{j \in N_i} h_{ij} \Omega_j \mathbf{c}_j / \sum_{j \in N_i} h_{ij} \Omega_j. \tag{3}$$

Here $\mathbf{c}_j$ is the centroid of the input samples for which j is the *bmu*, Ω_j is the cardinality of the set, N_i is the neighbourhood of the unit i and h_{ij} is the value of the neighbourhood function for $j \in N_i$ [2]. Often only a few iterations of (3) are enough.

If one wants to process *sequences* of input patterns $\{\mathbf{x}(n) | n = 1, \ldots, M\}$ then the basic SOM model has to be modified because it considers each input as separate. With the TKM- and RSOM-models also the previous inputs have an effect on the self-organization as presented in the following.

2.1 TKM

Chappell and Taylor proposed in [4] a modification of SOM known as temporal Kohonen map (TKM). In TKM each node i of an ordinary SOM is supplied with a potential

$$a_i(n) = da_i(n-1) - (1/2)||\mathbf{x}(n) - \mathbf{w}_i(n)||^2, \tag{4}$$

where $0 < d < 1$ and $\mathbf{x}(n)$ and $\mathbf{w}_i(n)$ are the input and weight vectors at time n. The form of a_i is loosely related to the basic SOM algorithm. The update of weights in (2) can be seen as a gradient method maximizing the last term of (4). The first term gives the map a "memory", which is essential in temporal sequence processing. The recursive equation (4) can be written also in the form where the dependence on past input values is better visible:

$$a_i(n) = -(1/2)\sum_{k=1}^{n} d^{n-k}||\mathbf{x}(k) - \mathbf{w}_i(k)||^2 + d^n a_i(0). \tag{5}$$

The potential (4) is used in the choice of the best matching unit during the training of the TKM-model. The *bmu* for input $\mathbf{x}(n)$ is the node for which (4) is maximized. If $d = 0$ the *bmu* would be the same as given by (1), but with nonzero d also the previous inputs play a role. The update of weights is done like in the basic SOM (2). Usually the input sequences are of fixed length and it might be sensible to reset the potential (4) to zero between different input sequences during the training. However Chappell and Taylor do not mention if they have done it or not.

In their article Chappell and Taylor tested their method with a 4×4 network receiving two-dimensional binary input. The length of the sequences was two. As one might expect the obtained result was that each of the 16 possible sequences had its own *bmu* and they were organized in the "right" order (not only on the basis of the last input pattern).

2.2 RSOM

The recurrent self-organizing map (RSOM) was inspired by the TKM-model. In RSOM each node is supplied with a temporally leaked difference vector y_i instead of a scalar valued potential,

$$\mathbf{y}_i(n) = (1-\lambda)\mathbf{y}_i(n-1) + \lambda(\mathbf{x}(n) - \mathbf{w}_i(n)). \tag{6}$$

Here $0 < \lambda < 1$ is the memory depth parameter. The *bmu* is the unit having the shortest $\mathbf{y}$,

$$\mathbf{y}_{bmu} = \min_i ||\mathbf{y}_i||^2. \tag{7}$$

And the training rule is of the form:

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \alpha(n)h_{ib}(n)\mathbf{y}_i(n). \tag{8}$$

If $\lambda = 1$ in (6) the RSOM looses its memory and reduces to the basic SOM. The equation (6) can be written as

$$\mathbf{y}_i(n) = \lambda\sum_{k=1}^{n}(1-\lambda)^{n-k}(\mathbf{x}(k) - \mathbf{w}_i(k)) \tag{9}$$

where it is assumed that $n > 0$ and $\mathbf{y}_i(0) = 0$. The update equation (8) resembles the gradient minimization of $||\mathbf{y}_i(n)||^2$ with respect to $\mathbf{w}_i$.

2.3 Comparison of TKM and RSOM

In [2] the properties of TKM and RSOM are compared and analysed. Here we represent a similar analysis where the steady state conditions for the SOM-models are considered.

In steady state the weight vectors $\mathbf{w}_i(k)$ in equations (5) and (9) can be written as constants $\mathbf{w}_i$ inpedendent of the time step. Thus $\mathbf{y}_i(n)$ and $a_i(n)$ can be seen as a functions of the variable $\mathbf{w}_i$:

$$a_i(n) = -(1/2) \sum_{k=1}^{n} d^{n-k} ||\mathbf{x}(k) - \mathbf{w}_i||^2 + d^n a_i(0). \tag{10}$$

$$\mathbf{y}_i(n) = \lambda \sum_{k=1}^{n} (1-\lambda)^{n-k} (\mathbf{x}(k) - \mathbf{w}_i) \tag{11}$$

We further assume a nonzero neighbourhood only for the *bmu* and take some specific map unit i into consideration. The corresponding set of length n input sequences having unit i as a *bmu* is marked with S_i. The number of input sequences in S_i is Ω_i.

As described in the previous section RSOM seeks to minimize $||\mathbf{y}_i(n)||^2$. Setting $\partial ||\mathbf{y}_i(n)||^2 / \partial \mathbf{w}_i = 0$ we obtain for the optimal weights

$$\mathbf{w}_i^l = \sum_{k=1}^{n} (1-\lambda)^{n-k} \mathbf{x}^l(k) / \sum_{k=1}^{n} (1-\lambda)^{n-k} \tag{12}$$

for each input $\mathbf{x}^l \in S_i$. Actually the optimal weigths must be computed by minimizing

$$\sum_{l=1}^{\Omega_i} ||\mathbf{y}_i^l(n)||^2, \tag{13}$$

where $\mathbf{y}_i^l(n)$ is the leaked difference vector for input $\mathbf{x}^l \in S_i$. By setting the derivative equal to zero we obtain

$$\mathbf{w}_i = (1/\Omega_i) \sum_{l=1}^{\Omega_i} \mathbf{w}_i^l, \tag{14}$$

where $\mathbf{w}_i^l$ is as in (12). This resembles the batch update formula (3) with neigbourhood $h_{ij} = \delta_{ij}$.

In TKM the choice of *bmu* tries to maximize $a_i(n)$. The maximization of $\sum_l a_i^l(n)$ leads to similar results (equations (12) and (13)) as the minimization of (14). The only difference is that in (12) $(1-\lambda)$ is replaced with d. However the problem with the TKM is that the update rule for the weights does not have a memory. Instead the update of the weights is done towards the last input pattern of a sequence. We again consider the steady state and the set S_i of input sequences whose *bmu* is the unit i. The optimal weights in the sense of the TKM update rule are thus found by minimizing

$$\sum_{l=1}^{\Omega_i} ||\mathbf{w} - \mathbf{x}^l(n)||^2. \tag{15}$$

The solution is

$$\mathbf{w}_i = (1/\Omega_i) \sum_{l=1}^{\Omega_i} \mathbf{x}^l(n), \tag{16}$$

which is different than (14). This indicates that the update rule and the choice of the *bmu* conflict in the TKM-model.

Figures 1 and 2 show the convergence of the weights of TKM and RSOM in a test carried out by Varsta et al. [2]. The figures confirm the results of previous analysis. In the test there was a 20×20 map receiving two dimensional input: $\{\mathbf{x}(n) = (a, b) \mid a, b = \{4, 8, 12, 16, 20, 24, 28\} \; \forall n\}$. The length of the input sequences was not told. The convergence of weights for TKM and RSOM is shown in Figures 1 and 2. It can be seen that the weigths of the RSOM cover the input space more uniformly. In Figure 1 is also shown the mean difference between (16) and (14) for sequences ending with different inputs. These differences seem to explain the behaviour of weights in the TKM case.

3 Model by Privitera and Morasso

The articles [5], [6] and [7] describe some approaches to temporal sequence processing. The basic component of the models is a two dimensional map of computational units ("neurons"). Each neuron has an activation function whose current value depends also on its activation history, besides the current input. The authors use the term: *a map of leaky integrators*. The idea is to generate and learn input sequences through sequential activation of the map units.

Here we will describe the model presented in [6] more closely. We follow the notation of Privitera and Morasso and thus the symbols may have a different meaning than in the previous section. The map is trained by feeding it with the input sequences $x_k = (x_k(1), \ldots, x_k(t))$. Each neuron i of the map has a weight vector $\tilde{x}_i^h$ for each time step h. At the instant t the weights are modified by the rule:

$$\Delta \tilde{x}_i^h = \begin{cases} \eta(t)(x(h) - \tilde{x}_i^h)(S_i(t) - S_i(t-1)) & \text{if } h \leq t \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $S_i(0) = 0$ and for $t > 0$

$$S_i(t) = \frac{\sum_{h=1}^{t} G(||x(h) - \tilde{x}_i^h||)}{\sum_{h=1}^{t} \sum_j G(||x(h) - \tilde{x}_j^h||)}, \tag{18}$$

and where $G(\cdot)$ is a Gaussian of fixed variance. The aim is that after the training each neuron has learned some pattern sequence stored in its weight vectors. However for me this seems not so obvious from the previous equations.

After the training the map is used to recognize the learned sequences from a continuous input. For each neuron there is a variable $\tau_i(t)$ representing the index of the *active* weight of neuron i at time t. At each step the current input vector is compared with the *active* weigth vectors of the neurons. If the pattern and weight are similar enough the activation of that particular neuron is increased and $\tau_i(t+1) = \tau_i(t) + 1$. So every neuron tracks some particular sequence from the input. It does not matter if the patterns of the sequence are not strictly subsequent in the input but they must be in the right order. When all patterns of some stored sequence have been observed the variable τ for that particular

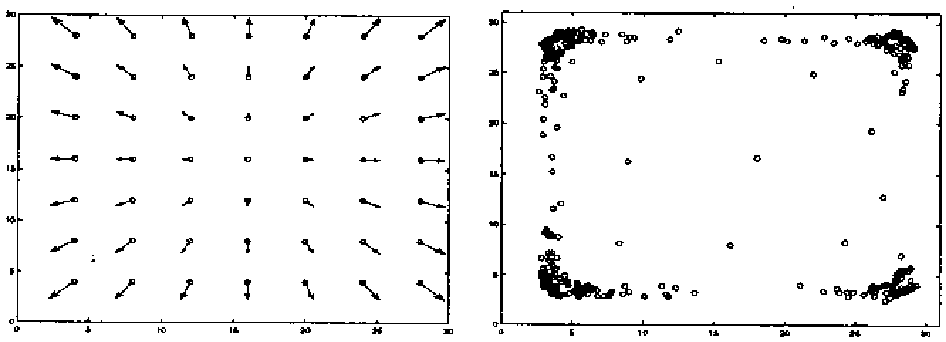

Figure 1:
Left: The mean difference between (16) and (14) for sequences having the last input at points marked with $\circ$.
Right: The convergence of weights in TKM-model for 20×20 map with 2D input.
Figure taken from [2].

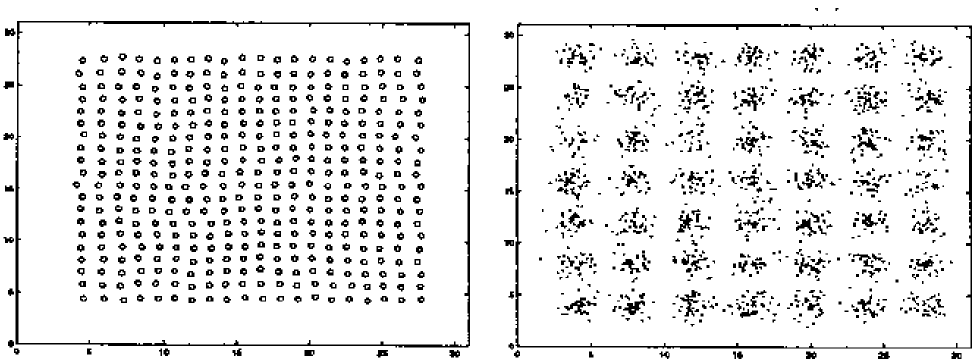

Figure 2:
Left: The convergence of weights in RSOM-model for 20×20 map with 2D input.
Right: The input vectors which build up the sequences.
Figure taken from [2]

neuron is reset to 1. So the neuron restarts to track the first input pattern of the sequence stored to it.

Privitera and Morasso presented an experiment of their model where a 4×4-map is trained with sequences of vectors in $\Re^5$. Each input vector defines a peak shape pattern. A sequence of this kind of patterns is shown in Figure 3.c. Figure 3.b shows the location of some stored sequences after the training and in Figure 3.a is shown the activation of the neurons as a function of time with the input of Figure 3.c.

4 Conclusions

The two SOM-models for temporal sequence processing, TKM and RSOM, were presented in Section 2. The analysis of the models revealed that the RSOM is a better justified solution for organizing sequences of input patterns. The tests carried out by Varsta et al. also confirm this conclusion. However some test examples with real world problems would be needed to evaluate the actual significance of these methods.

The model of Privitera and Morasso has the advantage that it can recognize a learned input sequence also when it is mixed with other sequences. However the description of the model in [6] was very brief and some things did not cleared up. It seems for example that there is no such topological ordering in the model like in the RSOM.

References

[1] M. Varsta, J. Millan and J. Heikkonen, "A recurrent self-organizing map for temporal sequence processing",*Proc. Int. Conf. on Artificial Neural Networks* (ICANN'97), Lausanne, Switzerland, 1997, pp. 421-426.

[2] M. Varsta, J. Millan, J. Heikkonen and J. Lampinen, "On the convergence properties of the temporal Kohonen map and the recurrent self-organizing map", *Proc. Int. Conf. on Artificial Neural Networks* (ICANN'98), Vol. 2, pp. 687-692.

[3] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, 1997.

[4] G. J. Chappell and J. G. Taylor, "The temporal Kohonen map",*Neural Networks*,vol. 6, 1993, pp. 441-445.

[5] C. Privitera and P. Morasso, "A new approach to storing temporal sequences", *Proc. Int. Joint Conf. Neural Networks* (IJCNN'93), Nagoya, Japan, 1993, pp. 2745-2748.

[6] C. Privitera and P. Morasso, "The analysis of continuous temporal sequences by a map of sequential leaky integrators", *Proc. IEEE Int. Conf. on Neural Networks*, Orlando, FL, 1994, pp. 3127-3130.

[7] R. Plamondon and C. Privitera, "A neural model for generating and learning a rapid movement sequence", *Biological Cybernetics*, Vol. 74, No. 2, 1995, pp. 117-130.

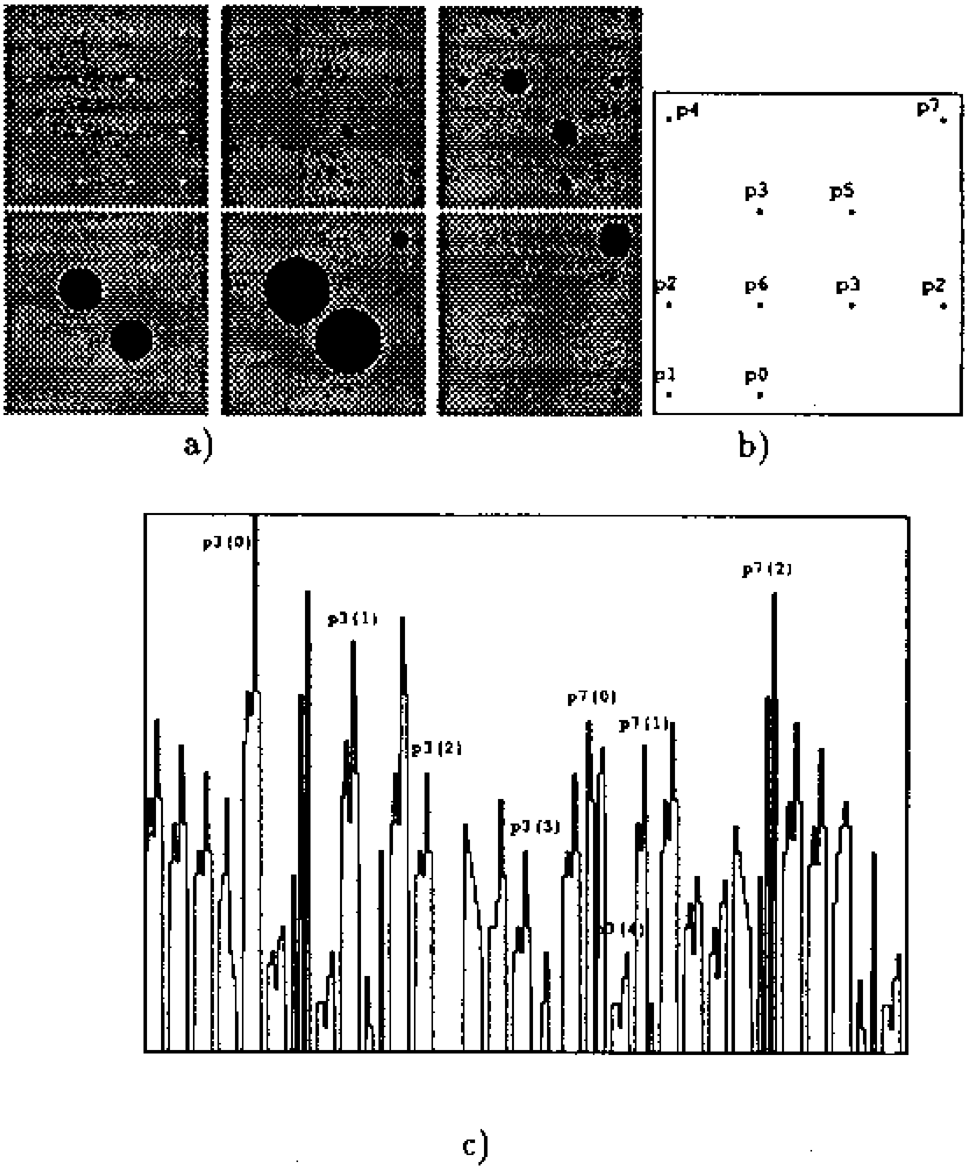

Figure 3: b) The 4×4-map storing sequences p0-p7. a) The activation of the neurons as a function of time with the input c).
Figure taken from [6].