Neural Processing Letters **13:** 237–251, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

Temporal Kohonen Map and the Recurrent Self-Organizing Map: Analytical and Experimental Comparison

MARKUS VARSTA[1], JUKKA HEIKKONEN[1], JOUKO LAMPINEN[1], and JOSÉ DEL R. MILLÁN[2]
[1]*Laboratory of Computational Engineering, Helsinki University of Technology, Miestentie 3, P.O. Box 9400, FIN-02015 HUT, Finland;*
e-mail: {*markus.varsta, jukka.heikkonen, jouko.lampinen*}*@hut.fi*
[2]*European Commission, Joint Research Centre, Institute for Systems, Informatics and Safety, I-21020 Ispra (VA), Italy; e-mail: jose.millan@jrc.it*

Abstract. This paper compares two Self-Organizing Map (SOM) based models for temporal sequence processing (TSP) both analytically and experimentally. These models, Temporal Kohonen Map (TKM) and Recurrent Self-Organizing Map (RSOM), incorporate leaky integrator memory to preserve the temporal context of the input signals. The learning and the convergence properties of the TKM and RSOM are studied and we show analytically that the RSOM is a significant improvement over the TKM, because the RSOM allows simple derivation of a consistent learning rule. The results of the analysis are demonstrated with experiments.

Key words: convergence analysis, self-organizing maps, temporal sequence processing.

1. Introduction

The Self-Organizing Map (SOM) [10] is probably the most popular unsupervised neural network model. The basic SOM is indifferent to the ordering of the input patterns. Real data, however, is often sequential in nature thus temporal context of a pattern may significantly influence its correct interpretation.

Since the SOM is quite popular in data mining applications, where it is primarily used for visualization and clustering, the idea of a SOM model that effectively accounts for temporal or other context of patterns is appealing and has been around for quite a while.

The chain of the best matching units (*bmu*s) for pattern sequences produce time varying trajectories of activity on the SOM. These trajectories were employed in [12] for visualization of speech signals and their variations. The activity trajectories have also found applications in process control and monitoring [15]. The hypermap idea [8] is to use several levels of information for the *bmu* selection. For example, in the case of sequential data, the *bmu* for the current pattern is searched from a subset of units restricted with the previous patterns. The self-organizing operator map analyzes the temporal sequence directly by associating each unit with an operator other than the usual Euclidean distance. These operators may for example be filters which are tuned to periodic phenomena in the data. Derivation of learning rules for generic operators is quite complicated. In [9] a genetic approach was proposed for parameter search but in special cases, such as Linear Prediction Coding [7], gradient descent can be used since the corresponding error function can be analytically defined. Operator maps have been applied to speech analysis [7].

The hierarchical model in [7] consists of two maps connected with a leaky integrator memory. The first of the maps transforms the input patterns. The transforms are stored in the leaky integrator memory which preserves a trace of the past transforms. The contents of the memory is the input of the second map. The idea is that the second map learns to distinguish different sequences by adapting to the different traces of transforms in the memory. Another hierarchical SOM based model [2] also has two maps but this model has a leaky integrator memory associated with both of the maps. The major difference between these hierarchical models is the computation of the contents of the leaky memory connecting the first and the second map. The model in [7] stores normalized inverted distances between the units and the input patterns into the memory. The model proposed in [2] uses distances in the map space of the units to a neighborhood of the best matching unit (*bmu*) of the first map.

One simple SOM based model that takes the temporal context of a pattern into account is the Temporal Kohonen Map (TKM) [3]. In the TKM the outputs of the units are replaced with leaky integrators, which effectively low pass filter the unit activities over the sequence of inputs. The TKM model was modified into the Recurrent Self-Organizing Map (RSOM) [16, 17] for better resolution, but it later turned out that the real improvement came from a consistent update rule for the network parameters. In this paper we analyze the properties of the TKM and the RSOM models. This analysis may also serve as an example of the risks in modifying a model without considering relevant aspects of the related algorithm. In the TKM the problem of the modification lays in the difficulty of updating the learning rule to accommodate for the modified activity rule. We show that the RSOM is a significant improvement over the TKM since it allows simple derivation of a consistent update rule.

2. The Self-Organizing Map

The Self-Organizing Map (SOM) [10] is a set of competitive units connected into a lattice with a topologic neighborhood function. The SOM can be regarded as a Linde–Buzo–Gray [10] vector quantizer units of which are arranged into a grid and locally attracted toward each other with strength determined by the neighborhood function. Later we will refer to the grid or the lattice of connected units as the map space. When properly trained the SOM forms a mapping of the input manifold, where the units close in the map space are close in the input space. However, the units close in the input space are not necessarily close in the map space since the dimensionality of the input manifold is often higher than the dimensionality map which consequently folds. The map space is usually one or two dimensional to facilitate visualization, for example.

The SOM, like vector quantizers in general, partitions the input space into convex regions of activity that are characterized with the following property: Every point in the space is closer to the centroid of its region than to the centroid of any other region. The centroids of the regions are defined by the weight vectors of the map. Partitioning of this kind is called Voronoi tessellation of the input space. The partitioning of an optimally trained map minimizes some error function but since the tessellation is discontinuous this error function cannot be an energy function. This is proven in, for example, [6].

The target of the SOM approach is to minimize the sum of weighted errors between the input and the weight or the reference vectors. When Euclidean distance is the error metric we can formulate the error function $E(V, X)$ with

$$E(V, X) = \frac{1}{2}\sum_{i \in V}\sum_{j=1}^{J} h_{i,j}\|(\mathbf{x_j} - \mathbf{w_i})\|^2 \tag{1}$$

where V is the map, $\mathbf{w_i}$ are the weights of unit i and $X = \{\mathbf{x_1}, \ldots, \mathbf{x_J}\}$ is the set of J input vectors. The neighborhood function $h_{i,j}$, that weights terms of the sum, determines how much the distance of the input vector $\mathbf{x_j}$ to the weight vector $\mathbf{w_i}$ contributes to the total error. The neighborhood function $h_{i,j}$ is typically a monotonically decreasing positive function of the map distance from unit i to the best matching unit (*bmu*) b of the pattern $\mathbf{x_j}$. A common choice for the neighborhood function is a Gaussian

$$h_{i,b}(n) = \exp\{-\|\mathbf{r_i} - \mathbf{r_b}\|^2/\sigma(n)^2\},$$

where $\mathbf{r_i}$ are the map coordinates of the unit i and $\mathbf{r_b}$ are the map coordinates of the *bmu*. The width of the Gaussian bell is controlled by $\sigma(n)$ which is normally reduced as the learning progresses. The neighborhood function serves two purposes. While not apparent from Equation (1) the neighborhood function orders the map by pulling units close in the map space toward each other in the input space. On the other hand the neighborhood function acts as a regularization factor that smoothes the functional mapping on the map.

Deriving an exact learning algorithm to minimize Equation (1) is difficult because the neighborhood function and the tessellation are only piecewise continuous in the input space making the error function only piecewise differentiable with respect to the weights. However, when we ignore the discontinuities at the boundaries of the Voronoi cells, we can easily derive approximate learning rules. In the classical stochastic rule the gradient is approximated for each sample and the weights are updated toward the optimum for the sample. The stochastic approach is realized with a two step algorithm. In the first step the input vector $\mathbf{x}(n)$ at step n is assigned a *bmu* b with

$$\|\mathbf{x}(n) - \mathbf{w_b}(n)\| = \min_{i \in V} \|\mathbf{x}(n) - \mathbf{w_i}(n)\| \ , \tag{2}$$

where $\mathbf{w_b}(n)$ are the weights of the *bmu*.

In the second step the weights are updated toward the optimum for $\mathbf{x}(n)$ with the stochastic gradient descent rule

$$\mathbf{w_i}(n+1) = \mathbf{w_i}(n) + \gamma(n) h_{i,b}(n)(\mathbf{x}(n) - \mathbf{w_i}(n)) \ , \tag{3}$$

where $h_{i,b}(n)$ is the value of the neighborhood function and $0 < \gamma(n) \leqslant 1$ is a scalar adaptation gain. During the final quantization stage the σ of the neighborhood function is set to zero, which means that the function becomes Kronecker delta. For correct rule, the adaptation gain or the learning rate must satisfy the Robbins–Monroe conditions for stochastic parameter estimation [1, 4, 5, 10].

The alternative batch approach is available for static input sets. In the batch approach the approximate gradient is evaluated for the entire input set and the weights are updated to the global optimum given the current partitioning of the data. In contrast with the stochastic rule the map may reach an equilibrium state where all units are exactly at the centroids of the samples in their regions of activity [10]. The batch rule, like the stochastic rule, can be implemented with a two step algorithm. In the first step each sample is assigned a *bmu* with Equation (2). In the second step the weights can be updated with

$$\mathbf{w_i} = \frac{\sum_{j=1}^{J} h_{i,b_j} \mathbf{x_j}}{\sum_{n=1}^{J} h_{i,b_j}} \tag{4}$$

where b_j is the bmu for the pattern $\mathbf{x_j}$. However, since the neighborhood function is identical for samples with the same *bmu* we can rewrite Equation (4) in a computationally more efficient form

$$\mathbf{w_i} = \frac{\sum_{j \in V} h_{i,j} \Omega_j \mathbf{c_j}}{\sum_{j \in V} h_{i,j} \Omega_j} \ , \tag{5}$$

where $\mathbf{c_j}$ is the centroid of the samples in the Voronoi cell of j and Ω_j is the cardinality of the set.

After a sufficient number of input vector presentations a mapping will form, i.e. the weight vectors will specify the centroids of clusters covering the input space. The point density of these centroids is related to the actual density in [4, 5], where it is shown that unlike for a random quantizer there is no magnification factor related to a quantizer generated with the SOM algorithm.

3. Temporal Kohonen Map and Recurrent Self-Organizing Map

In the Temporal Kohonen Map (TKM) model leaky integrators, that gradually lose their activity, are added into the outputs of the otherwise normal competitive units.

These integrators and consequently the decay of activation is modeled with the difference equation

$$U_i(n, d) = dU_i(n-1, d) - \tfrac{1}{2}\|\mathbf{x}(n) - \mathbf{w_i}(n)\|^2 \ , \tag{6}$$

where $0 \leqslant d < 1$ is a time constant, $U_i(n, d)$ is the activation of the unit i at step n while $\mathbf{w_i}(n)$ is the weight vector of the unit i and $\mathbf{x}(n)$ is the input pattern. The formula in Equation (6) preserves a trace of the past activations as weighted sum. In fact it incorporates a linear low pass filter in the outputs of the otherwise normal competitive units. The unit with the maximum activity is the *bmu* in analogy with the normal SOM.

The update rule for the TKM is not specifically addressed in [3]. In the experiments, however, weights were updated toward the last sample of the input sequence using the normal stochastic SOM update rule in Equation (3), which, corresponds with stochastic gradient descent when the time constant d in the activity computation is zero.

In the *Recurrent Self-Organizing Map (RSOM)* the leaked quantity is the difference vector instead of its squared norm. These integrators are modeled with

$$\mathbf{y_i}(n, \alpha) = (1-\alpha)\mathbf{y_i}(n-1, \alpha) + \alpha(\mathbf{x}(n) - \mathbf{w_i}(n)) \ , \tag{7}$$

where $\mathbf{y_i}(n, \alpha)$ is the leaked difference vector for unit i at step n. The leaking coefficient α is analogous to the value of $1 - d$ in the TKM but in the RSOM formulation the sum of the factors is one to ensure stability when α is positive but less than one [14]. The RSOM formulation like the TKM formulation associates a linear low pass filter with each unit to preserve a trace of the past but in the RSOM the operator is moved from the unit outputs into the inputs.

After moving the leaky integrators into the difference vector computation we can treat the map much like the normal SOM when unit with

$$\|\mathbf{y_b}(n, \alpha)\| = \min_{i \in V} \|\mathbf{y_i}(n, \alpha)\|$$

is the *bmu*. To derive an update rule for the RSOM we first formulate an error function $E(n)$ for the current sample $\mathbf{x}(\mathbf{n})$

$$E(n) = \sum_{i \in V} h_{i,b}(n)\|\mathbf{y_i}(n, \alpha)\|^2,$$

where V is the map. The gradient direction of $E(n)$ with respect to $\mathbf{w_i}(n)$ is $\mathbf{y_i}(n, \alpha)$ and thus the stochastic weight update rule for $\mathbf{w_i}$ to minimize error $E(n)$ is

$$\mathbf{w_i}(n+1) = \mathbf{w_i}(n) + \gamma(n)h_{i,b}(n)\mathbf{y_i}(n, \alpha) \ .$$

In this derivation we ignored the discontinuities of the error function $E(n)$ at the boundaries of the Voronoi cells. This is the normal practise when deriving update rules for SOM models.

Table I. The properties of the TKM and the RSOM. .

Model	Bmu selection criterion	Weight update target
TKM	$\max U(\cdot, d)$	$\max U(\cdot, 0)$
RSOM	$\min \|\mathbf{y}(\cdot, \alpha)\|^2$	$\min \|\mathbf{y}(\cdot, \alpha)\|^2$

The second column is *bmu* selection criterion and the third column is the update rule target.

The key properties of the learning rules of the TKM and the RSOM models are summarized in Table I.

4. Comparison of TKM and RSOM

In this section we will discuss the learning properties of the TKM and the RSOM models. First in Section 4.1 we derive the optimal or the activity maximizing weights for a set of sequences and a single unit for both TKM and RSOM. The analysis directly extends to multiple units in the zero neighborhood case

$$h_{i,j}(n) = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases}$$

when the boundaries of the Voronoi cells are ignored. In 4.2 we look into the update rule of the TKM to see what the map actually learns and compare the results with the RSOM results.

4.1. OPTIMAL WEIGHTS

Brief mathematical analysis is sufficient to show how maximizing activity in the TKM should lead to similar weights as minimizing the norm of the leaked difference vector in the RSOM when the maps share the same topology and data. Let us first consider a single TKM unit and a set $\mathcal{S} = \{X_1, X_2, ..., X_N\}$ of sequences. The samples of the sequence $X_j \in \mathcal{S}$ are $\mathbf{x_j}(1), \mathbf{x_j}(2), ..., \mathbf{x_j}(n_j)$, where n_j is the length of the sequence X_j. In the TKM the goal is to distinguish different sequences by maximizing the activity of the corresponding *bmu*. For the set $\mathcal{S}$ of sequences and weights $\mathbf{w_T}$ the activity $U(\mathcal{S}, \mathbf{w_T})$ over $\mathcal{S}$ is the sum

$$U(\mathcal{S}, \mathbf{w_T}) = \frac{-1}{2} \sum_{X_j \in \mathcal{S}} \sum_{k=1}^{n_j} d^{(n_j - k)} \|\mathbf{x_j}(k) - \mathbf{w_T}\|^2 . \tag{8}$$

Since the activity $U(\mathcal{S}, \mathbf{w_T})$ is a parabola, it is everywhere continuous and differentiable with respect to $\mathbf{w_T}$. Consequently its maximum lies either at its extreme or at the single zero of $\partial U(\mathcal{S}, \mathbf{w_T})/\partial \mathbf{w_T}$. From

$$\frac{\partial U(\mathcal{S}, \mathbf{w_T})}{\partial \mathbf{w_T}} = 0$$

we obtain

$$\mathbf{w_T} = \frac{\sum_{X_j \in \mathcal{S}} \sum_{k=1}^{n_j} d^{(n_j-k)} \mathbf{x_j}(k)}{\sum_{X_j \in \mathcal{S}} \sum_{k=1}^{n_j} d^{(n_j-k)}} \; . \tag{9}$$

The weights $\mathbf{w_T}$ are optimal as they maximize the activity $U(\mathcal{S}, \mathbf{w_T})$ of the unit over the set $\mathcal{S}$. When all sequences have the same length n, the inner sum of the denominator of Equation (9) is constant allowing us to simplify the the equation to

$$\mathbf{w_T} = \frac{1}{\Omega_{\mathcal{S}}} \sum_{X_j \in \mathcal{S}} \mathbf{w_T^j}$$

where $\Omega_{\mathcal{S}}$ is the cardinality of $\mathcal{S}$ and $\mathbf{w_T^j}$ are the optimal weights for the sequence $X_j \in \mathcal{S}$ defined with

$$\mathbf{w_T^j} = \frac{\sum_{k=1}^{n} d^{(n-k)} \mathbf{x_j}(k)}{\sum_{k=1}^{n} d^{(n-k)}} \; .$$

These weights are the mean of the per sequence optimal weights, and they also are a good approximation when all sequences are sufficiently long for the chosen d.

For the RSOM unit the leaked difference vector $\mathbf{y}(X, \mathbf{w_R})$, where $X = \mathbf{x}(1), \ldots, \mathbf{x}(n)$ is the input sequence and $\mathbf{w_R}$ are the RSOM weights, is

$$\mathbf{y}(X, \mathbf{w_R}) = \alpha \sum_{k=1}^{n} (1-\alpha)^{(n-k)} (\mathbf{x}(k) - \mathbf{w_R}).$$

Since the goal is to minimize the norm of the leaked difference vector, for the set $\mathcal{S}$ we can write

$$E(\mathcal{S}, \mathbf{w_R}) = \sum_{X_j \in \mathcal{S}} \|\mathbf{y}(X_j, \mathbf{w_R})\|^2$$

for the error function $E(\mathcal{S}, \mathbf{w_R})$, which is minimized at the optimum weights. $E(\mathcal{S}, \mathbf{w_R})$ defines a parabola just like $U(\mathcal{S}, \mathbf{w_T})$ for the TKM and thus the optimal weights are either at an extreme or at the single zero of the derivative of the error function with respect to the weights $\mathbf{w_R}$. From

$$\frac{\partial E(\mathcal{S}, \mathbf{w_R})}{\partial \mathbf{w_R}} = 0$$

we obtain

$$\mathbf{w_R} = \frac{\sum_{X_j \in \mathcal{S}} \left(\sum_{k=1}^{n_j} (1-\alpha)^{(n_j-k)} \sum_{k=1}^{n_j} (1-\alpha)^{(n_j-k)} \mathbf{x_j}(k) \right)}{\sum_{X_j \in \mathcal{S}} \left(\sum_{k=1}^{n_j} (1-\alpha)^{(n_j-k)} \right)^2} \; . \tag{10}$$

The optimal RSOM weights in Equation (10) are quite close to the weights specified in Equation (9). The small difference comes from the location of the leaky integrators.

Much like with the TKM we can simplify Equation (10) if we assume that all sequences have the same length n. We get

$$\mathbf{w_R} = \frac{1}{\Omega_{\mathcal{S}}} \sum_{X_j \in \mathcal{S}} \mathbf{w^j_R} \ ,$$

where $\mathbf{w^j_R}$ are the optimal weights for the sequence $X_j \in \mathcal{S}$ defined with

$$\mathbf{w^j_R} = \frac{\sum_{k=1}^{n} (1-\alpha)^{(n-k)} \mathbf{x_j}(k)}{\sum_{k=1}^{n} (1-\alpha)^{(n-k)}} \ .$$

These weights are identical with the corresponding TKM weights when $d = 1 - \alpha$. From the analysis we observe that the optimal weights for both models are linear combinations of the samples in the sequences.

4.2. LEARNING ALGORITHMS

Since the update rule of the RSOM approximates gradient descent to minimize the sum of the squared norms of the leaked difference vectors regularized by the neighborhood, the map explicitly seeks to learn the weights defined in the previous section. With the TKM this is not the situation. We show that generally the steady state weights of the TKM do not maximize the activity and use simulations to show how this affects the behavior of the TKM. To simplify the analysis we only consider the zero neighborhood case.

By definition, in a steady state further training causes no changes in weights. In practice this means that the derivative of the objective function is zero with respect to the weights given a static set of input patterns. Though in the stochastic training scheme reaching a steady state is not possible in finite time, criteria for a steady state can be defined and their impact considered when we study the equivalent batch approach. For the batch approach we split the TKM algorithm in two. In the first stage the data is Voronoi partitioned among the units with the network activity function. In the second stage the new weights given the partitioning are computed. While proving convergence for any SOM model is very difficult [4, 6], if the TKM converges the weights have to satisfy the criteria we define here.

We have a set $\mathcal{S} = \{X_1, ..., X_N\}$ of discrete sequences and a map V. Last sample of each sequence $X_j \in \mathcal{S}$ is $\mathbf{x_j}(n_j)$ where n_j is the length of the sequence $X_j \in \mathcal{S}$. In a steady state the TKM weights have to be in the centroids of the last samples of the sequences in the Voronoi cells of the units. This observation is a direct consequence of the weight update toward the last samples of the sequences corresponding with maximizing the activity when the time delay coefficient d was set to zero. When

$d = 0$ TKM activity for unit i in Equation (8) reduces to

$$U_i(\mathcal{S}_i, \mathbf{w_i}) = \frac{-1}{2} \sum_{X_j \in \mathcal{S}_i} (\mathbf{w_i} - \mathbf{x_j}(n_j))^2$$

and the corresponding steady state weights at $\partial U_i(\mathcal{S}_i, \mathbf{w_i})/\partial \mathbf{w_i} = 0$ are

$$\mathbf{w_i} = \frac{1}{\Omega_{\mathcal{S}_i}} \sum_{X_j \in \mathcal{S}_i} \mathbf{x_j}(n_j) \;, \forall i \in V \;, \tag{11}$$

where $\mathcal{S}_i \subset \mathcal{S}$ is the set of sequences in the Voronoi cell of i and $\Omega_{\mathcal{S}_i}$ is the cardinality of $\mathcal{S}_i$. These weights are necessary for a steady state. The optimal TKM weights with respect to the activity rule were defined in the previous section. The weights

$$\mathbf{w_{T_i}} = \frac{\sum_{X_j \in \mathcal{S}_i} \sum_{k=1}^{n_j} d^{(n_j-k)} \mathbf{x_j}(k)}{\sum_{X_j \in \mathcal{S}_i} \sum_{k=1}^{n_j} d^{(n_j-k)}} \;, \quad \forall i \in V \tag{12}$$

maximize activity with our simplifying assumptions.

The problem with the TKM is the discrepancy between the optimal weights and the necessary steady state weights. Figure 1, which has a portion of a TKM during training, shows this graphically. The arrow 'Gradient direction to maximize activity' shows the steepest descent direction to maximize activity while the arrow 'TKM update direction' shows the actual update direction toward the last sample of the sequence.

We ran several simulations to show the impact of the discrepancy between the *bmu* selection and the weight update in the TKM. The first simulation involves a 1D map in a discrete 1D input manifold with seven input patterns. We initialized a 25 unit map with optimal weights (see axis 1 in Figure 3) to maximize the total activity

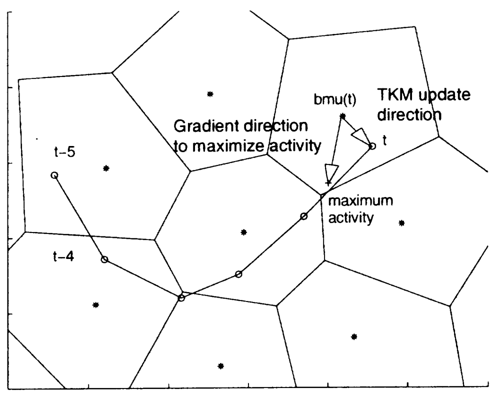

Figure 1. A piece of a TKM during training. The units, and their Voronoi cells, are marked with asterisks (∗) and the input sequence with little circles (○). The plus (+) is drawn at the activity maximizing weights. The arrows show the optimal and the actual TKM update directions.

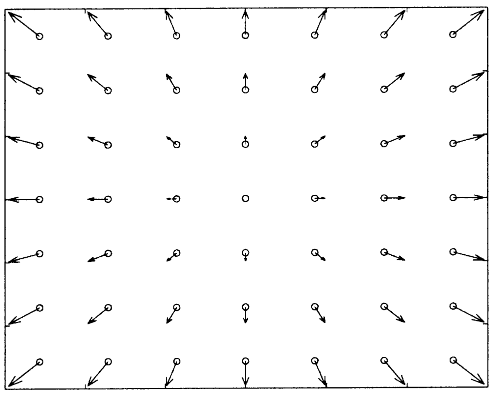

Figure 2. Approximation of the mean bias between the activity maximizing update directions and the TKM update directions for a regular 7×7 grid of input patterns in a 2D input manifold. We considered all sequences of length seven and computed the approximation for $d = 0.15$ for the time delay.

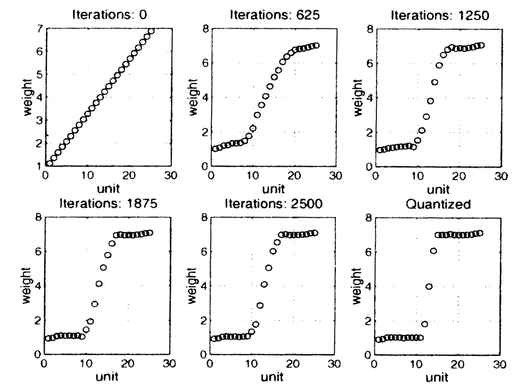

Figure 3. A map initialized with near optimal weights and trained with the TKM approach. Notice how most of the units are drawn into the edges.

when the 1D inputs were $1 \ldots 7$ and the leaking coefficient d was 0.1429. The selection of d leads to a nearly uniform optimal distribution of weights in the input manifold. The nearly optimally initialized map was further trained by randomly picking one of the inputs, thus creating long random sequences, and updating the weights using the stochastic training scheme. The samples of the random sequences were corrupted with additive Gaussian noise $\sim N(0, 0.125)$.

Figure 3 shows the progress of a sample run for the TKM. The TKM quickly 'forgets' the initial weights because they do not satisfy the steady state criterion we derived earlier. Notice how the units are drawn toward the extremes of the input space leaving only a couple of units to cover bulk of the space. Similar 1D experiment with the RSOM in Figure 4 yields a practically unchanged result.

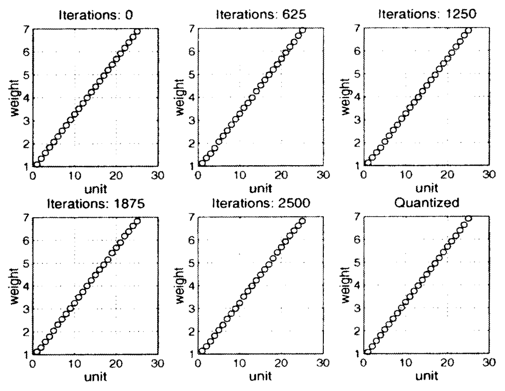

Figure 4. A map initialized with near optimal weights and trained with the RSOM approach.

We can intuitively explain the reason for the units being drawn toward the edges in the TKM with Figures 1 and 2. For sequences that end near the edges of the input manifold the activity maximizing TKM weights and consequently the *bmu*s are systematically closer to the center of the manifold than the last samples of the sequences which the units are updated toward. We can see this bias in Figure 1 in the difference between the activity maximizing update direction and the actual update direction. The bias causes units to be attracted toward the edge and especially corner samples. Once a unit is close enough it will no longer be the *bmu* for any non trivial sequence of moving value.

Figure 2 shows an approximation of mean bias between the activity maximizing update directions and the TKM update directions for a regular 7×7 grid of input patterns in a 2D input manifold. We considered all sequences of length seven and computed the approximation using $d = 0.15$ for the time delay. The bias is zero only at the center of the manifold and becomes larger the closer the input is to the edge. The lengths and the directions of the arrows show the relative magnitude and direction of the bias for the sequences ending at that particular input. Formally

$$\mathbf{u_j} \approx \sum_{X_k \in \mathcal{S}_j} \mathbf{x_j} - \mathbf{w_{X_k}}$$

where $\mathbf{u_j}$ is the arrow drawn at input $\mathbf{x_j}$, $\mathcal{S}_j$ is the set of sequences the last sample of which is $\mathbf{x_j}$, X_k is a sequence in $\mathcal{S}_j$ and $\mathbf{w_{X_k}}$ are the activity maximizing TKM weights for X_k. The arrows form what resembles a gradient field of smooth bump. The behavior of the TKM in the 2D simulations supports the intuitive result in the figure.

In the 2D simulations we trained hundred TKM and RSOM maps to estimate the weight distributions in the input manifold with Gaussian kernels. The maps were trained with Luttrell's incremental approach [13]. The maps were trained with random sequences by picking one of the possible input patterns and corrupting it with additive Gaussian noise $\sim N(0, 0.125)$. We used two 2D input manifolds

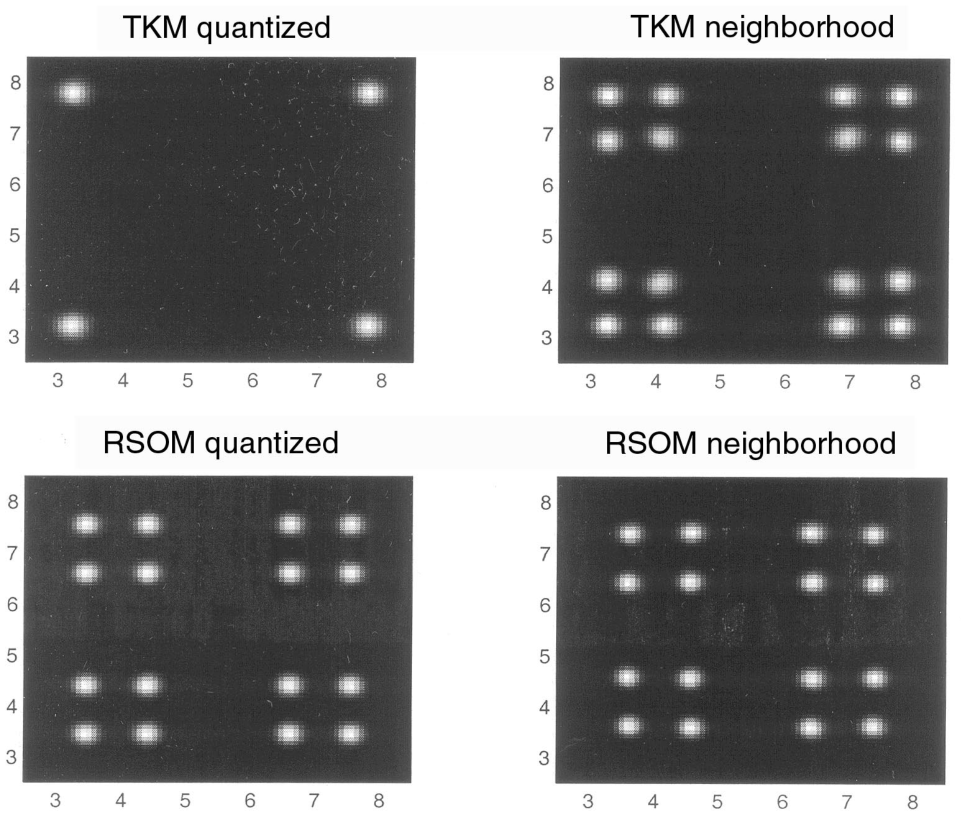

Figure 5. Kernel density estimates for the weight distributions of 4 $\times$ 4 TKM and RSOM maps in 2D manifolds with four input patterns one in each corner. The figure depicts how the weight vectors of the hundred independent runs were distributed into the input manifold. As the manifold was the same the optimal distribution is a four by four grid, where each of the 16 locations encodes one of the 16 possible combinations of two out of the four input patterns. Lighter shade signifies higher density in other words higher probability of a weight vector to appear at that spot.

where the sparse manifold had patterns only in its four corners. This simulation essentially repeats the experiment in [3]. The other manifold was denser and had 49 input patterns arranged in a regular 7 $\times$ 7 grid.

For the sparse manifold we trained maps with sixteen units arranged in four by four grid using $d = 0.3$ for the time delay in the TKM and $\alpha = 0.7$ for the RSOM leaking parameter. The resulting estimates of weight distributions in the input manifold are in Figure 5 where lighter shade means that the likelihood of a weight appearing at that point is higher. The distributions are meaningful because the optimal weights as derived in the previous section are linear combinations of the patterns in the input sequences. As a consequence the way that the maps partition the input manifold directly reflect the way they partition the sequence space as well. When quantized with the zero neighborhood the, TKM concentrates all its units

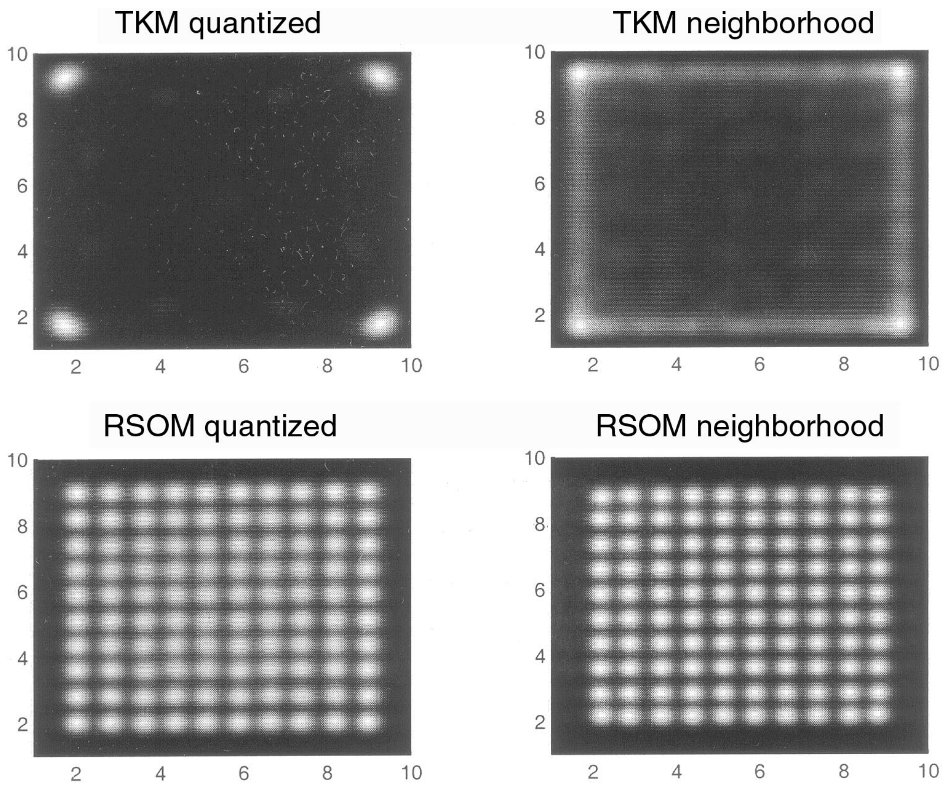

Figure 6. Kernel density estimates for the weight distributions of 10 × 10 TKM and RSOM maps in 2D manifolds with 49 input patterns in regular 7 × 7 grid. Like with the sparse manifold the linear combinations of the 49 possible input patterns somewhat uniformly span the manifold and hence the optimal distribution of the weights is a regular 10 × 10 grid of units. The RSOM roughly satisfies this condition with or without the neighborhood at the end but the TKM fails in both cases because of the learning rule discrepancy. In the case of the TKM the weights of all hundred maps are scattered mainly in the corners while some are left along the edges when the neighborhood was left on. Lighter shade signifies higher density. This experiment demonstrates how TKM systematically loses much of its expressive power by concentrating all of its units near the edges of the manifold.

at the four input patterns as expected from the update rule. When the neighborhood is not turned off the map forms a four by four grid of units where each unit is sensitive to one of the 16 possible combinations of two input patterns. With the RSOM the result is not very dependent on the treatment of the neighborhood. The map creates a four by four grid of units which in the case when the neighborhood was retained was slightly denser.

The situation changed when we used the more densely sampled input manifold and 100 unit maps units arranged in 10 × 10 grid. For this simulation we set the time delay factor at $d = 0.15$ and $\alpha = 0.85$ accordingly. The resulting estimates of the weight distributions are in Figure 6. Again without the neighborhood the TKM concentrated all its units near the corner inputs reflecting the intuitive result in

Figure 2. With the neighborhood, more units were left to cover the core of the manifold as the neighborhood stiffens the map up but the improvement is not as significant as it was with the sparse manifold. Increasing the radius of the neighborhood made the phenomenon more pronounced. The properties of the model are such that the conflicting activity and update rules force units toward the corners of the manifold but stiffening the map up with strong neighborhood partially counters this effect.

Now recall the optimal weights we derived for TKM and RSOM in Equations (9) and (10) respectively. TKM concentrated most of its units in the edges and the corners of the manifold leaving only a few units to cover the bulk when all input patterns were not in the corners of the manifold. In the sparse manifold the conflicting activity and update rules were countered with the neighborhood but the same neighborhood radius did not help with the dense manifold. Increasing the neighborhood radius would help in the simple manifold but this approach could not be used in more complicated manifolds since the large neighborhood radius would not allow the map to follow the manifold. In our opinion using the neighborhood to correct the inherent problem in the model design is not the correct approach. In these simulations the TKM model wasted a considerable part of it expressive power. The RSOM on the contrary systematically learned weights that nearly optimally spanned the input manifold. The problem with the TKM could be resolved if the TKM was trained with a rule that did not require gradient information. The chances are, however, that such a rule would be computationally very demanding because it would require repeated evaluations of the target function.

5. Conclusions

In this paper the RSOM is compared against the TKM both analytically and with experiments. The analysis shows that the RSOM is a significant improvement over the TKM, because only the RSOM allows simple derivation of an update rule that is consistent with the activity function of the model. In a sense the RSOM provides a simple answer to the question regarding the optimal weights aroused in [3] but possibly at the cost of biological plausibility, which motivated the original TKM. The RSOM approach has been applied in an experiment with EEG data [17] containing epileptiform activity and in time series prediction using local linear models [11].

References

1. Bishop, C. M.: *Neural Networks for Pattern Recognition*, Oxford University Press, (1995).
2. Carpinteiro, O. A. S.: A hierarchical self-organizing map model for sequence recognition, In: *Proceedings of ICANN '98*, September (1998), pp. 816–820.
3. Chappel, G. J. and Taylor, J. G.: The temporal Kohenen map, *Neural Networks*, **6** (1993), 441–445.

4. Cottrell, M.: Theoretical aspects of the som algorithm, In: *Proceedings of WSOM '97, Workshop on Self-Organizing Maps*, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, June (1997), pp. 246–267.
5. Cottrell, M.: Theoretical aspects of the som algorithm, *Neurocomputing*, **21**(1998), 119–138.
6. Erwin, E., Obermeyer, K. and Schulten, K.: Self-organizing maps: ordering, convergence properties and energy functions, *Biological Cybenetics*, **67** (1992), 47–55.
7. Kangas, J.: *On the Analysis of Pattern Sequences by Self-Organizing Maps*, PhD thesis, Helsinki University of Technology, Espoo, Finland, (1994).
8. Kohenen, T.: The hypermap architecture, In: T. Kohenen, K. Mäkisara, O. Simula and J. Kangas, (eds), *Artificial Neural Networks*, Vol. II, Amsterdam, Netherlands, North-Holland, (1991), pp. 1357–1360.
9. Kohenen, T.: Things you have'nt heard about the Self-Organizing Map, In: *Proceedings of the ICNN '93, International Conference on Neural Networks*, Piscataway, NJ, IEEE, IEEE Service Center, (1993), pp. 1147–1156.
10. Kohenen, T.: *Self-Organizing Maps*, Vol. 30 of *Lecture Notes in Information Sciences*, Springer, 2nd Edn, (1997).
11. Koskela, T., Varsta, M., Heikkonen, J. and Kaski, K.: Prediction using rsom with local linear models, *Int Journal of Knowledge-Based Intelligent Engineering Systems*, **2**(1) (1998), 60–68.
12. Leinonen, L., Kangas, J., Torkkola, K. and Juvas, A.: Dysphonia detected by pattern recognition of spectral composition, *J. Speech and Hearing Res.*, **35** April (1992), 287–295.
13. Luttrell, S. P.: Image compression using a multilayer neural network, *Pattern Recognition Letters*, **10** (1989), 1–7.
14. Proakis, G. and Manolakis, D. G.: *Digital Signal Processing: Principles, Algorithms, and Applications*, Macmillan Publishing Company, (1992).
15. Tryba, V. and Goser, K.: Self-Organizing Feature Maps for process control in chemistry. In: T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, (eds), *Artificial Neural Networks*, Amsterdam, Netherlands, North-Holland, (1991), pp. 847–852.
16. Varsta, M., del Ruiz Millán, J. and Heikkonen, J.: A recurrent self-organizing map for temporal sequence processing, In: *Proceedings of the ICANN '97*, Springer-Verlag, Berlin, Heidelberg, New York, October 1997. ISBN 3-540-63631-5.
17. Varsta, M., Heikkonen, J. and del Ruiz Millán, J.: Context learning with the self organizing map, In: *Proceedings of WSOM '97, Workshop on Self-Organizing Maps*, Helsinki University of Technology, Neural Networks Research Centre, June 1997.