LETTER Communicated by Teuvo Kohonen

Auto-SOM: Recursive Parameter Estimation for Guidance of Self-Organizing Feature Maps

Karin Haese
Data Warehouse/Data Mining, Mummert & Partners Management Consulting, Braunschweig, Germany, D-38104

Geoffrey J. Goodhill
Department of Neuroscience, and Georgetown Institute for Cognitive and Computational Sciences, Georgetown University Medical Center, Washington, D.C. 20007, U.S.A.

An important technique for exploratory data analysis is to form a mapping from the high-dimensional data space to a low-dimensional representation space such that neighborhoods are preserved. A popular method for achieving this is Kohonen's self-organizing map (SOM) algorithm. However, in its original form, this requires the user to choose the values of several parameters heuristically to achieve good performance. Here we present the Auto-SOM, an algorithm that estimates the learning parameters during the training of SOMs automatically. The application of Auto-SOM provides the facility to avoid neighborhood violations up to a user-defined degree in either mapping direction.

Auto-SOM consists of a Kalman filter implementation of the SOM coupled with a recursive parameter estimation method. The Kalman filter trains the neurons' weights with estimated learning coefficients so as to minimize the variance of the estimation error. The recursive parameter estimation method estimates the width of the neighborhood function by minimizing the prediction error variance of the Kalman filter. In addition, the "topographic function" is incorporated to measure neighborhood violations and prevent the map's converging to configurations with neighborhood violations. It is demonstrated that neighborhoods can be preserved in both mapping directions as desired for dimension-reducing applications. The development of neighborhood-preserving maps and their convergence behavior is demonstrated by three examples accounting for the basic applications of self-organizing feature maps.

1 Introduction

Kohonen's self-organizing map (SOM) (Kohonen, 1982, 1984, 1995) has found wide application in the fields of data exploration, data mining, data classification, data compression, and biological modeling, due to its proper-

ties of "neighborhood preservation" and local resolution of the input space proportional to the data distribution. Since it was introduced, it has been improved in several ways. For instance, the assumption of a fixed architecture has been relaxed (Martinetz & Schulten, 1994; Haese & vom Stein, 1996; Bauer & Villmann, 1997), and it has been shown how the magnification factor can be controlled (Bauer, Der, & Herrmann, 1996). However, an important limitation still remains: the values of the learning parameters that are most appropriate for fast convergence to the minima of the neurons' individual energy functions depend on the unique properties of the data in each application. Thus, the successful application of the algorithm depends on the user's knowledge of the input data and experience with the algorithm. It would be greatly preferable for these parameters to be determined automatically during training to achieve the best possible outcome, thus relieving the user of extensive parameter studies and reducing the number of learning steps required. Kohonen (1995) suggested a partial solution to this problem by using a batch learning method that dispenses with the learning rate. Unfortunately, however, the learning rate is used to control the magnification factor of the final map, which is rapidly gaining interest in all field of application of SOMs, including speech recognition (Bauer et al., 1996), medicine (Villmann, in press a), and robotics (Villmann, in press b). Herrmann (1995) proposed a way to determine automatically the width of the neighborhood function, but at the cost of further parameters to be specified (though the values of these parameters may be less critical). A rather different approach to avoiding the necessity for choosing these parameters is generative topographic mapping (GTM) (Bishop, Svensen, & Williams, 1996). In this article we present the Auto-SOM: a method for automatic parameter estimation in the SOM based on estimation by a linear Kalman filter extended by a recursive parameter estimation method. We demonstrate its effectiveness on examples including a real application problem, and compare its performance with alternative versions of the SOM and with the GTM.

Feature maps consist of nodes $\boldsymbol{r}$ arranged on an $n_{\mathcal{A}}$-dimensional lattice. All nodes are elements of the set $\mathcal{A}$. To each node a weight vector $\boldsymbol{w_r}$ is assigned, which is a point in the $n_{\mathcal{M}}$-dimensional input space. The feature map develops by adapting the weight vectors $\boldsymbol{w_r}$ to the presented input data $\boldsymbol{v}$ of a manifold $\mathcal{M} \subseteq \Re$. $\boldsymbol{v}$ is assigned to the output unit $\boldsymbol{r}'$, which has the minimum Euclidean distance between $\boldsymbol{v}$ and the weight vector $\boldsymbol{w_{r'}}$:

$$\|\boldsymbol{w_{r'}}(j) - \boldsymbol{v}(j)\| = \min_{\boldsymbol{r}\in\mathcal{A}} \|\boldsymbol{w_r}(j) - \boldsymbol{v}(j)\|. \tag{1.1}$$

All weights are then adapted toward the input vector $\boldsymbol{v}$:

$$\boldsymbol{w_r}(j) = \boldsymbol{w_r}(j-1) + \Delta\boldsymbol{w_r}(j) \tag{1.2}$$

$$\Delta\boldsymbol{w_r}(j) = \epsilon(j) \cdot h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j) \cdot \left[\boldsymbol{v}(j) - \boldsymbol{w_r}(j-1)\right]. \tag{1.3}$$

The amplification factor consists of a learning coefficient $\epsilon(j)$ and a neighborhood function $h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)$. The latter is a gaussian function of width σ of the Euclidean distance $\|\boldsymbol{r} - \boldsymbol{r}'\|$ between the winner neuron and the updated neuron:

$$h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j) = e^{-\left(\frac{\rho^{\mathcal{A}}(\boldsymbol{r},\boldsymbol{r}')}{\sqrt{2}\sigma(j)}\right)^2}, \tag{1.4}$$

where $\rho^{\mathcal{A}}(\boldsymbol{r}, \boldsymbol{r}') = \sqrt{(r_1 - r'_1)^2 + \cdots + (r_{n_{\mathcal{A}}} - r'_{n_{\mathcal{A}}})^2}$. Commonly, both learning parameters $\epsilon(j)$ and $\sigma(j)$ are chosen to decay exponentially during the training process. This choice fulfills the necessary and sufficient convergence conditions for $\epsilon(j)$:

$$0 < \epsilon(j) < 1, \quad \lim_{j\to\infty} \sum \epsilon(j) \to \infty, \quad \lim_{j\to\infty} \sum \epsilon^2(j) < \infty. \tag{1.5}$$

However, appropriate initial values and rates of decrease of ϵ cannot be deduced. Furthermore, conditions on the choice of σ are unknown. Although empirical studies show that a certain relation between both parameters is essential for the weights to converge to the centers of Voronoi cells while preserving neighborhood relations as well as possible, the initial values and rates of the exponential decrease remain to be determined in the general case. Consequently, the learning parameters are still an unidentified function of the network architecture and the input data. Nonuniformly distributed data require particularly carefully chosen parameter courses for convergence to neighborhood-preserving, dimension-reducing feature maps. An initial approach to the problem of automatic parameter estimation has been formulated by Haese (1998, 1999). This estimates the learning coefficient $\epsilon(j)$ optimally within a state-space model of the learning process, which is calculated, predicted, and filtered by a linear Kalman filter (KF) algorithm. We first review this method in section 2, with additional justifications for the assumptions made. In section 3 we then extend it to achieve an optimal estimate of the width of the neighborhood function as a function of time. Instead of modeling the progress of neighborhood preservation qualities, which has been achieved only under major simplifications (see Haese, 1998, 1999), here we estimate the width of the neighborhood function by a recursive parameter estimation method (RPEM). The advantage of the RPE algorithm to estimate the width of the neighborhood function is that it aims to minimize the same prediction error,

$$\boldsymbol{e}(\sigma, j) = h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)v(j) - h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)\boldsymbol{w}_{\boldsymbol{r}}(\sigma, j|j-1), \tag{1.6}$$

which is minimized by the KF implementation of the SOM. The prediction error expresses the deviation between the actual learning process and the prediction of the process based on the process model. Thus, the KF and

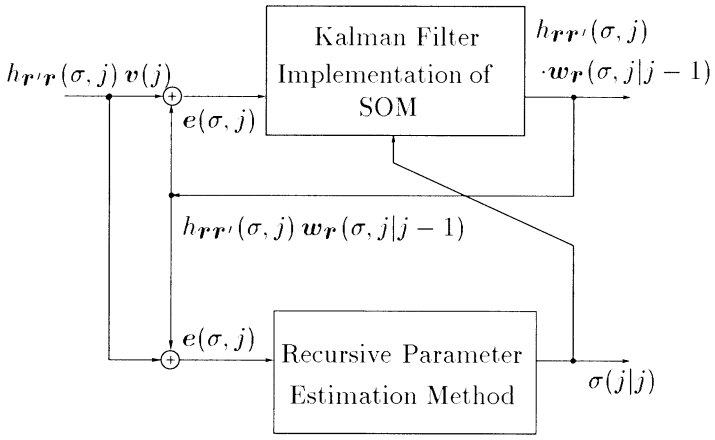

Figure 1: Block diagram of Auto-SOM. The autonomous learning parameter estimation method is based on a Kalman filter implementation of the original SOM and the recursive parameter estimation method for the width σ of the neighborhood function $h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)$. See Table 1 for notation.

the RPEM are coupled to yield the Auto-SOM, as depicted in Figure 1. In section 4 we demonstrate the performance of Auto-SOM on three examples and one real high-dimensional application problem. We discuss the results in comparison with the results of the original learning algorithm, the batch-learning version and the GTM. Table 1 summarizes the symbols used in this article.

2 Kalman Filter Implementation of the Self-Organizing Algorithm

We now review the state-space model of the self-organizing learning process proposed by Haese (1998). This state-space model is the basis on which a linear KF acts to estimate and predict the inner states of the process. A state-space model in discrete time gives the relations between the input, noise, and output signals, $\boldsymbol{u}(j)$, $\boldsymbol{q}(j)$, and $\boldsymbol{y}(j)$, respectively, in the form of a first-order difference equation using an auxiliary state vector $\boldsymbol{x}(j)$. The outputs $\boldsymbol{y}(j)$ are some linear combination of the states $\boldsymbol{x}(j)$. Noise $\boldsymbol{q}_{\boldsymbol{x}}(j)$ and $\boldsymbol{q}_{\boldsymbol{y}}(j)$ is assumed to corrupt the states as well as the output of the process, leading to the general state-space model:

$$\boldsymbol{x}(j+1) = \boldsymbol{A}(j)\boldsymbol{x}(j) + \boldsymbol{B}(j)\boldsymbol{u}(j) + \boldsymbol{q}_{\boldsymbol{x}}(j) \tag{2.1}$$

$$\boldsymbol{y}(j) = \boldsymbol{C}(j)\boldsymbol{x}(j) + \boldsymbol{q}_{\boldsymbol{y}}(j). \tag{2.2}$$

The KF uses the state and measurement equations, 2.1 and 2.2, respectively, as well as assumptions about the noise density distributions, to predict the

Table 1: List of Symbols.

A	transition scalar of σ in RPEM
$\boldsymbol{A}$	state transition matrix of a general state-space model
$\mathcal{A}$	set of nodes $\boldsymbol{r}$ on the feature map
$\boldsymbol{B}$	input matrix of a general state-space model
$\boldsymbol{C}$	measurement matrix of a general state-space model
$\mathcal{D}^{\mathcal{M}}(\boldsymbol{r}, \boldsymbol{r}')$	graph of nodes' connections in $\mathcal{M}$
$\boldsymbol{e}$	prediction error
$E\{\cdot\}$	expected value
$E_{\boldsymbol{r}}$	dominant part of a neuron's individual energy function
$F(\cdot)$	topographic function
$h_{\boldsymbol{r}\boldsymbol{r}'}$	neighborhood function
j	learning step
$l(\cdot)$	criterion function
$\boldsymbol{K}_{\boldsymbol{r}}$	Kalman gain matrix of $\boldsymbol{w}_{\boldsymbol{r}}$
$\boldsymbol{K}_{\sigma}$	Kalman gain matrix of RPEM
$\mathcal{M}$	set of input pattern v
$n_{\mathcal{A}}$	dimension of feature map
$n_{\mathcal{M}}$	dimension of the input space
$\boldsymbol{o}_{\boldsymbol{r}}$	response at node $\boldsymbol{r}$ to a pattern v
$\hat{\boldsymbol{o}}_{\boldsymbol{r}}$	measured output vector at $\boldsymbol{r}$
$\boldsymbol{P}_{\boldsymbol{r}}$	covariance matrix of $\boldsymbol{w}_{\boldsymbol{r}}$
P_{σ}	covariance matrix of σ in RPEM
q_{err}	quantization error
$\boldsymbol{q}\boldsymbol{o}_{\boldsymbol{r}}$	measurement noise in $\hat{\boldsymbol{o}}_{\boldsymbol{r}}$
$\boldsymbol{q}\boldsymbol{w}_{\boldsymbol{r}}$	system noise in weight transition
$\boldsymbol{q}\boldsymbol{x}$	noise corrupting the states $\boldsymbol{x}$
$\boldsymbol{q}\boldsymbol{y}$	noise corrupting the output $\boldsymbol{y}$
$\boldsymbol{Q}\boldsymbol{o}_{\boldsymbol{r}}$	covariance matrix of $\boldsymbol{q}\boldsymbol{o}_{\boldsymbol{r}}$
$\boldsymbol{Q}\boldsymbol{w}_{\boldsymbol{r}}$	covariance matrix of $\boldsymbol{q}\boldsymbol{w}_{\boldsymbol{r}}$
$\boldsymbol{r}$	node position vector on feature map
$\boldsymbol{r}'$	node position of winner neuron
$\boldsymbol{u}$	vector of input signals to a general state-space model
v	input vector to feature map
$\boldsymbol{w}_{\boldsymbol{r}}$	weight vector at location $\boldsymbol{r}$
W	matrix of derivatives of $\frac{d\boldsymbol{w}_{\boldsymbol{r}}}{d\sigma}$
$\boldsymbol{x}$	state vector of a general process
$\boldsymbol{x}(j\|j-1)$	predicted states $\boldsymbol{x}$
$\boldsymbol{x}(j\|j)$	corrected states $\boldsymbol{x}$
$\boldsymbol{y}$	output vector of a general process
$\hat{\boldsymbol{y}}$	measured output vector $\boldsymbol{y}$
α	positive constant of Newton algorithm
ϵ	general learning coefficient
Λ	weighting matrix in RPEM
$\Omega_{\boldsymbol{r}}$	Voronoi cell of a neuron at $\boldsymbol{r}$
$\#\Omega_{\boldsymbol{r}}$	cardinality of a Voronoi cell $\Omega_{\boldsymbol{r}}$
$\Psi_{\mathcal{M}\to\mathcal{A}}$	mapping from input space $\mathcal{M}$ to output space $\mathcal{A}$
$\Psi_{\mathcal{A}\to\mathcal{M}}$	mapping from output space $\mathcal{A}$ to input space $\mathcal{M}$
Φ	matrix of derivatives of $\frac{d\boldsymbol{e}}{d\sigma}$
$\rho^{\mathcal{A}}(\boldsymbol{r}, \boldsymbol{r}')$	Euclidean distance between $\boldsymbol{r}$ and $\boldsymbol{r}'$ measured in the output space
σ	width of the neighborhood function

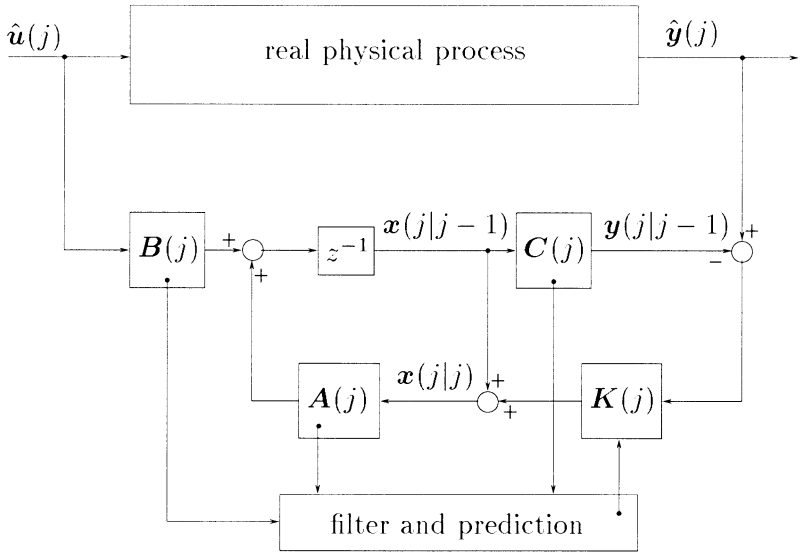

Figure 2: Basic concept of a Kalman filter application to a physical process. A state-space model of the real process is incorporated in the KF algorithm with the state vector $\boldsymbol{x}(j)$ and matrices $\boldsymbol{A}(j)$, $\boldsymbol{B}(j)$, $\boldsymbol{C}(j)$. The KF algorithm corrects its state predictions $\boldsymbol{x}(j|j-1)$ on the basis of the input and output measurements, $\hat{\boldsymbol{u}}(j)$ and $\hat{\boldsymbol{y}}(j)$, respectively.

states $\boldsymbol{x}$ and the outputs $\boldsymbol{y}$ one time step in advance. The predictions $\boldsymbol{y}(j|j-1)$ are then compared with the observed outputs $\hat{\boldsymbol{y}}(j)$ (see Figure 2). This comparison leads to a residual term, which is weighted with the Kalman gain $\boldsymbol{K}$, in order to correct the predictions $\boldsymbol{x}(j|j-1)$, so that the estimate of the actual states $\boldsymbol{x}(j|j)$ of the process at time step j is achieved (see Figure 2). For our application to the SOM, the only observations of the process (see Figure 3) are the inputs randomly selected from a set $\mathcal{M}$ of input patterns. We assume the measured output $\hat{\boldsymbol{o}}_{\boldsymbol{r}}(j)$ of each neuron in response to input pattern $\boldsymbol{v}(j)$ to be the input pattern weighted by the neighborhood function $h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)$:

$$\hat{\boldsymbol{o}}_{\boldsymbol{r}}(\sigma, j) = h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)\boldsymbol{v}(j), \tag{2.3}$$

associated with $\hat{\boldsymbol{y}}$ in our general state-space model. We model these outputs by $\boldsymbol{o}_{\boldsymbol{r}}(\sigma, j)$. This is done by first assigning the input $\boldsymbol{v}(j)$ to its nearest neighbor weight vector $\boldsymbol{w}_{\boldsymbol{r}'}(j)$ and then multiplying the weights with their values of the neighborhood function $h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)$ to yield

$$\boldsymbol{o}_{\boldsymbol{r}}(\sigma, j) = h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)\boldsymbol{w}_{\boldsymbol{r}}(\sigma, j) + \boldsymbol{q}_{\boldsymbol{o}_{\boldsymbol{r}}}(\sigma, j). \tag{2.4}$$

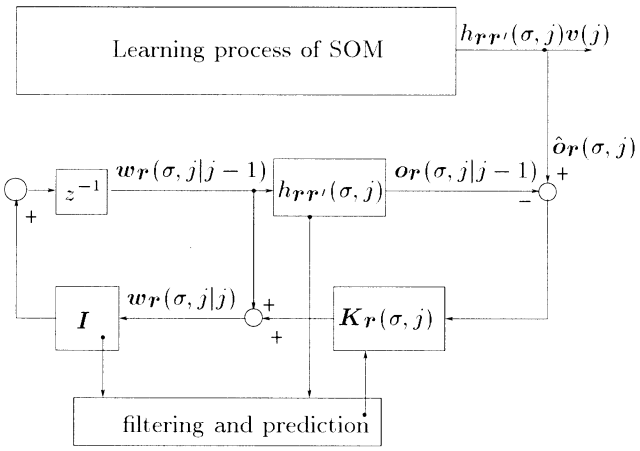

Figure 3: Kalman filtering concept of the self-organizing learning process model. The weight vectors $\boldsymbol{w_r}$ are the states of the process, transfered by the identity matrix $\boldsymbol{I}$ to the next learning step and mapped onto the predicted measurements $\boldsymbol{o_r}(\sigma, j|j-1)$ by the neighborhood function $h_{\boldsymbol{r}\boldsymbol{r}'}$.

This model is assumed to be corrupted by a noise sequence $\boldsymbol{q_{o_r}}(\sigma, j)$ to be described. Finally, we find the general states $\boldsymbol{x}$ in case of the special learning process model to be the weights $\boldsymbol{w_r}(\sigma, j)$. At the end of the training, the weights will define a static mapping from the input space onto the feature map. This leads to the state equation of the self-organizing learning process:

$$\boldsymbol{w_r}(\sigma, j) = \boldsymbol{w_r}(\sigma, j-1) + \boldsymbol{q_{w_r}}(\sigma, j). \tag{2.5}$$

Equation 2.5 implies that especially in the first phase of learning, the assumption of a static transition of the weights is not at all correct. During this training phase, the system noise, $\boldsymbol{q_{w_r}}(\sigma, j)$, accounts for the dynamic transition. The noise covariances are high at this time, of the order of magnitude of the measurement noise covariances $\boldsymbol{q_{o_r}}(\sigma, j)$, which will result in high Kalman gain elements to correct the system model with the measurement $\hat{\boldsymbol{o}}_{\boldsymbol{r}}$. The noise sequences $\boldsymbol{q_{w_r}}(\sigma, j)$ are easily deduced to be gaussian distributed with zero mean and covariance,

$$E\{\boldsymbol{q_{w_r}}(\sigma, j) \cdot \boldsymbol{q}^T_{\boldsymbol{w_r}}(\sigma, j+n)\} = \begin{cases} \boldsymbol{Q_{w_r}}(\sigma, j) & n = 0 \\ 0 & n \neq 0, \end{cases} \tag{2.6}$$

starting from the proof that the neurons' weights during the learning follow approximately gaussian distributed stochastic processes (Yin & Allinson, 1995). Yin and Allinson showed that for any complex neighborhood function, the weight vectors converge in the mean square sense to the centers of

Voronoi cells $\Omega_{\boldsymbol{r}}$,

$$\lim_{j\to\infty} E\{\boldsymbol{w_r}(j)\} = \frac{1}{\#\Omega_{\boldsymbol{r}}} \sum_{\boldsymbol{v}\in\Omega_{\boldsymbol{r}}} v, \tag{2.7}$$

with variance

$$\lim_{j\to\infty} E\{(\boldsymbol{w_r}(j) - E\{\boldsymbol{w_r}(j)\})^2\} = 0. \tag{2.8}$$

$\#\Omega_{\boldsymbol{r}}$ denotes the cardinality of the data set $\Omega_{\boldsymbol{r}}$. As the whole process model is parameterized by the width σ of the neighborhood function, it is not surprising that the state and the measurement noise covariance are also at least implicit functions of σ. Because the weights $\boldsymbol{w_r}$ are modeled to be independent, the state covariance matrix is diagonal. The diagonal elements of the covariance matrix $\boldsymbol{Q_{w_r}}(\sigma, j)$ are the average quadratic distances between the weight vectors and their expected values:

$$diag(\boldsymbol{Q_{w_r}}(\sigma, j)) = \frac{1}{N_{\mathcal{A}}} \sum_{\boldsymbol{r}\in\mathcal{A}} \left(\boldsymbol{w_r}(\sigma, j) - E\{\boldsymbol{w_r}(\sigma, j)\}\right)^2. \tag{2.9}$$

The expected values $E\{\boldsymbol{w_r}(\sigma, j)\}$ are known to converge to the centers of the Voronoi cell $\Omega_{\boldsymbol{r}}$ in the limit $\sigma \to 0$. This final state is exactly modeled by the state equation of the learning process (see equation 2.5). Thus, the corresponding expected values are

$$E\{\boldsymbol{w_r}\} = \frac{1}{\#\Omega_{\boldsymbol{r}}} \sum_{\boldsymbol{v}\in\Omega_{\boldsymbol{r}}} v. \tag{2.10}$$

Hence, the distribution and convergence analysis by Yin and Allinson provides fundamental results for the justification of the KF implementation of the self-organizing learning process. In like manner, it can be used to describe the statistics of the measurement noise for the modeled state $\sigma \to 0$. Considering that the measurements are independent, it follows from equations 2.3 and 2.4 that the noise sequences $\boldsymbol{q_{o_r}}$ are white, fulfilling

$$E\{\boldsymbol{q_{o_r}}(\sigma, j) \cdot \boldsymbol{q}^T_{\boldsymbol{o_r}}(\sigma, j+n)\} = \begin{cases} \boldsymbol{Q_{o_r}}(\sigma, j) & n = 0 \\ 0 & n \neq 0 \end{cases} \tag{2.11}$$

and with zero mean. In general its distribution is not gaussian but specified by the data distribution in the Voronoi cells.

Again modeling the states $\boldsymbol{w_r}$ as independent of each other gives the diagonal shape of the covariance matrix $\boldsymbol{Q_{o_r}}(\sigma, j)$. In order to quantify the covariance matrix elements, we consider the nearest neighbor to the input

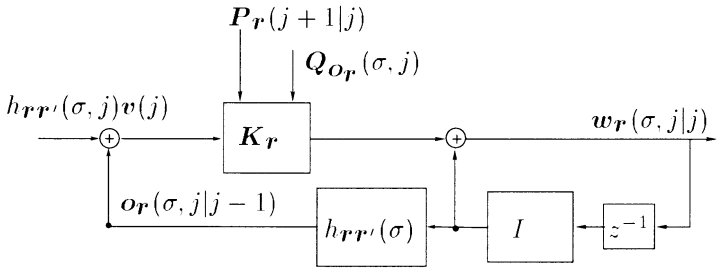

Figure 4: Linear Kalman filtering of the self-organizing learning process model. The weight vector $\boldsymbol{w_r}$ is transferred by the identity matrix $\boldsymbol{I}$ to the next learning step and mapped onto the predicted measurements $\boldsymbol{o_r}(\sigma, j|j-1)$ by the neighborhood function $h_{\boldsymbol{rr'}}$.

vector $v(j)$ to be at location $\boldsymbol{r''}$. With the nearest neighbor to the input vector $v(j-1)$ at location $\boldsymbol{r'}$, the deviation between the observed output and the prediction according to our model is

$$\boldsymbol{q_{o_r(j)}} = h_{\boldsymbol{rr''}}(\sigma, j)v(j) - h_{\boldsymbol{rr'}}(\sigma, j)\boldsymbol{w_r}(\sigma, j|j-1). \tag{2.12}$$

Evaluating the corresponding covariance matrix is extremely computationally demanding. Therefore, we consider for our calculation the experimental outcome $v(j)$ in learning step j to be approximately in the center of the Voronoi cell $\Omega_{\boldsymbol{r''}}$ and compensate for this simplification by averaging the variance over all neurons. Therefore, we expect the average variance for each neuron to be

$$diag(\boldsymbol{Q_{o_r}}(\sigma, j)) = \frac{1}{N_{\mathcal{A}}} \sum_{\boldsymbol{r}\in\mathcal{A}} \sum_{\boldsymbol{r'},\boldsymbol{r''}\in\mathcal{A}}$$
$$\left(h_{\boldsymbol{rr''}}(\sigma, j) \frac{1}{\#\Omega_{\boldsymbol{r''}}} \sum_{\boldsymbol{v}\in\Omega_{\boldsymbol{r''}}} v(j) - h_{\boldsymbol{rr'}}(\sigma, j)\boldsymbol{w_r}(\sigma, j|j-1) \right)^2. \tag{2.13}$$

Thus, the linear KF applies to the parameterized state-space description of the learning process (see equations 2.4 and 2.5) as a minimal variance estimator of the states $\boldsymbol{w_r}$ and the measurements $\boldsymbol{o_r}$. This estimation is performed as depicted in the block diagram, Figure 4.

The output $\boldsymbol{o_r}(\sigma, j|j-1)$ is predicted on the basis of $(j-1)$ measurements $\hat{\boldsymbol{o}}_{\boldsymbol{r}}$ and the model of the learning process (see equations 2.4 and 2.5). $\boldsymbol{o_r}(\sigma, j|j-1)$ is compared with the actual measurement $h_{\boldsymbol{rr'}}(\sigma, j)v(j)$. The residual is attenuated by the Kalman gain matrix $\boldsymbol{K_r}$ and then used to correct the predicted weights $\boldsymbol{w_r}(\sigma, j|j-1)$, resulting in the equations

$$\boldsymbol{w_r}(\sigma, j|j) = \boldsymbol{w_r}(\sigma, j|j-1) + \boldsymbol{K_r}(\sigma, j)\,[\hat{\boldsymbol{o}}_{\boldsymbol{r}}(j) - \boldsymbol{o_r}(\sigma, j|j-1)], \tag{2.14}$$

$$\boldsymbol{o_r}(\sigma, j|j) = h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)\boldsymbol{w_r}(\sigma, j|j), \tag{2.15}$$
$$\boldsymbol{K_r}(\sigma, j) = \boldsymbol{P_r}(\sigma, j|j-1)\, h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)$$

$$[\, h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)\, \boldsymbol{P_r}(\sigma, j|j-1)\, h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j) + \boldsymbol{Q_{o_r}}(\sigma, j-1)]^{(-1)} \tag{2.16}$$

$$\boldsymbol{P_r}(\sigma, j|j) = \boldsymbol{P_r}(\sigma, j|j-1) - \boldsymbol{K_r}(\sigma, j)\, h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)\, \boldsymbol{P_r}(\sigma, j|j-1). \tag{2.17}$$

These filter equations follow from Bayes' rule applied to the a posteriori probability density of the states $\boldsymbol{w_r}$, assuming a gaussian probability density function with covariance matrix $\boldsymbol{P_r}$.

Equation 2.14 provides the self-organizing learning algorithm with an individual learning coefficient $K_{\boldsymbol{r}}$ for each weight $w_{\boldsymbol{r}}$, which is optimal with respect to minimizing the expected covariance of the innovation term $E\{(\hat{\boldsymbol{o}}_{\boldsymbol{r}}(j) - \boldsymbol{o_r}(\sigma, j|j))(\hat{\boldsymbol{o}}_{\boldsymbol{r}}(j) - \boldsymbol{o_r}(\sigma, j|j))^T\}$. In contrast, the original learning rule executes the adaptation control for all weights in general with the learning coefficient $\epsilon(j)$. Consequently, equations 2.14 through 2.17 execute the self-organizing learning rule, providing an individual learning coefficient for each weight during the training.

Finally, the prediction of the states, $\boldsymbol{w_r}(\sigma, j|j-1)$, the outputs, $\boldsymbol{o_r}(\sigma, j|j-1)$, and the state covariance matrix, $\boldsymbol{P_r}(\sigma, j|j-1)$, is performed by the equations

$$\boldsymbol{w_r}(\sigma, j|j-1) = \boldsymbol{w_r}(\sigma, j-1|j-1), \tag{2.18}$$
$$\boldsymbol{o_r}(\sigma, j|j-1) = h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma, j)\boldsymbol{w_r}(\sigma, j|j-1), \tag{2.19}$$
$$\boldsymbol{P_r}(\sigma, j|j-1) = \boldsymbol{P_r}(\sigma, j-1|j-1) + \boldsymbol{Q_{w_r}}(\sigma, j-1). \tag{2.20}$$

We now have a set of system models for the learning process parameterized by the width σ of the neighborhood function. Each of these models describes the law for a linear KF to predict the learning process and adapt the neurons' weights by minimizing the expected covariance matrix of the innovation term. The problem we are now faced with is to select at each learning step an appropriate value of $\sigma(j)$. We do this by making use of the information provided by the prediction error sequence,

$$\{e\} = \{j|\boldsymbol{e}(\sigma, j) = \hat{\boldsymbol{o}}_{\boldsymbol{r}}(j) - \boldsymbol{o_r}(\sigma, j|j-1)\}, \tag{2.21}$$

which will lead to the RPEM.

3 Recursive Parameter Estimation of the Width of the Neighborhood Function

In general, a good system model is one that is good at predicting, that is, producing small prediction errors $\boldsymbol{e}(\sigma, j)$. Therefore, we will determine the parameter σ by minimizing a function $l(\cdot)$ of the prediction error sequence $\{e\}$. A recursive estimation method (Ljung & Söderström, 1983; Ljung, 1999)

is suitable to tackle the problem. In accordance with our system model, we expect the error to be a zero-mean white noise sequence. Then a quadratic criterion function of the prediction error is known to be a good choice (Ljung, 1999) to measure the validity of the model:

$$l(\boldsymbol{e}, \sigma, j) = \frac{1}{2}\boldsymbol{e}^T(\sigma, j)\Lambda^{(-1)}\boldsymbol{e}(\sigma, j). \tag{3.1}$$

The best choice for the weighting matrix $\Lambda^{(-1)}$ is the covariance matrix of the prediction error $\boldsymbol{e}$, so that the equation $E\{\boldsymbol{e}\boldsymbol{e}^T - \Lambda^{(-1)}\} = 0$ is solved numerically (Ljung & Söderström, 1983) by

$$\Lambda(j) = \Lambda(j-1) + \alpha(j)[\boldsymbol{e}(j)\boldsymbol{e}^T(j) - \Lambda(j-1)]. \tag{3.2}$$

Minimization of the expected value $V(\sigma) = E\{l(\boldsymbol{e}, \sigma, j)\}$ of the criterion function (see equation 3.1) is performed by the Newton algorithm,

$$\sigma(j+1) = \sigma(j) - \left[\frac{d^2}{d\sigma^2}V(\sigma)\right]^{(-1)}\left[\frac{d}{d\sigma}V(\sigma)\right]^T. \tag{3.3}$$

The numerical estimation of the derivatives in this equation leads to a recursive scheme of the parameter estimation proposed by Ljung and Söderström (1983) and Ljung (1999). This parameter estimation technique can be reformulated in terms of KF equations 3.4 through 3.6 and its prediction equations 3.13 and 3.14 (Haese, 1990). The filter equations,

$$\sigma(j|j) = \sigma(j|j-1) + \boldsymbol{K}_\sigma(j)\boldsymbol{e}(j) \tag{3.4}$$

$$\boldsymbol{K}_\sigma(j) = P_\sigma(j|j-1)\Phi(j)\,[\Phi^T(j)P_\sigma(j|j-1)\Phi(j) + \Lambda(j)]^{(-1)} \tag{3.5}$$

$$P_\sigma(j|j) = P_\sigma(j|j-1) - \boldsymbol{K}_\sigma(j)\Phi^T(j)P_\sigma(j|j-1), \tag{3.6}$$

correct the predicted width σ of the neighborhood function by adding the attenuated prediction error of the neuron's output $\boldsymbol{o_r}$. The attenuation is performed by the Kalman gain matrix $\boldsymbol{K}_\sigma$. $\boldsymbol{K}_\sigma$ is calculated on the basis of the estimated prediction error covariance matrix $\Lambda(j)$, the derivative $\Phi^T(j)$ of the prediction error, and the derivative $W(j+1)$ of the weights with regard to the state σ:

$$\begin{aligned}\Phi^T(j) &= \frac{d\boldsymbol{e}}{d\sigma} \\ &= h_{\boldsymbol{r}\boldsymbol{r}'}(\sigma(j|j), j)\left(\frac{d\boldsymbol{w_r}(j+1|j)}{d\sigma}\right)_{|\sigma(j|j)} \\ &\quad + \left(\frac{dh_{\boldsymbol{r}\boldsymbol{r}'}}{d\sigma}\right)_{|\sigma(j|j)} \boldsymbol{w_r}(j+1|j, \sigma(j|j))\end{aligned} \tag{3.7}$$

$$W(j+1) = \left(\frac{d\boldsymbol{w_r}(j+1|j)}{d\sigma}\right)_{|\sigma(j|j)}. \tag{3.8}$$

Finally, the prediction of the width $\sigma(j|j-1)$ and its covariance $P_\sigma(j|j-1)$ is essential for the self-adaptive estimation of σ. Therefore, we are looking for a transition model describing the transition from $\sigma(j-1|j-1)$ to $\sigma(j|j-1)$ as precisely as possible. We could use the decrease on the gradient of some cost function considered to qualify final map configurations, such as the quantization error or Luttrell's training vector quantizer (TVQ) cost function. However, the quantization error,

$$q_{err} = \frac{1}{\#\Omega} \sum_{\boldsymbol{r}'} \sum_{\boldsymbol{v} \in \Omega_{\boldsymbol{r}'}} (v - w_{\boldsymbol{r}'})^2, \tag{3.9}$$

does not depend on the current states of both learning parameters $\epsilon(j)$ and $\sigma(j)$. This deficiency is remains in Luttrell's TVQ cost function,

$$E_{TVQ} = \frac{1}{2} \sum_{\boldsymbol{r}'} \sum_{\boldsymbol{r}} h_{\boldsymbol{r}\boldsymbol{r}'} \sum_{\boldsymbol{v} \in \Omega_{\boldsymbol{r}}} (v - w_{\boldsymbol{r}'})^2, \tag{3.10}$$

although here dependence on $\sigma(j)$ is given. In the same way, the individual energy functions of the neurons

$$\tilde{E}_{\boldsymbol{r}} = \sum_{\boldsymbol{r}'} h_{\boldsymbol{r}\boldsymbol{r}'} \sum_{\boldsymbol{v} \in \Omega_{\boldsymbol{r}}} (v - w_{\boldsymbol{r}'})^2, \tag{3.11}$$

first introduced by Tolat (1990), do not fulfill our demands. But Erwin, Obermayer, and Schulten (1992) have corrected equation 3.11 for this deficiency and, moreover, revealed that the individual energy function consists of two terms. The part

$$E_{\boldsymbol{r}} = \epsilon \frac{1}{N_{\mathcal{A}}} \sum_{\boldsymbol{r}' \in \mathcal{A}} h_{\boldsymbol{r}\boldsymbol{r}'} \frac{1}{\#\Omega_{\boldsymbol{r}'}} \sum_{\boldsymbol{v} \in \Omega_{\boldsymbol{r}'}} (\boldsymbol{v} - \boldsymbol{w}_{\boldsymbol{r}})^2 \tag{3.12}$$

dominates for any map configuration except for highly disordered feature maps. During our simulations with Auto-SOM, we found that disordered maps occur only during the very first training cycles, so that we can restrict our model to the gradient of this term. We finally implement the average over the individual energy functions, which leads to a good transition model for σ. Thus, the prediction equations follow:

$$\sigma(j+1|j) = A(\sigma, j)\sigma(j|j) \tag{3.13}$$

$$P_\sigma(j+1|j) = \left(\frac{dA(\sigma, j)\sigma}{d\sigma} \right)^2 P_\sigma(j|j) \tag{3.14}$$

with

$$A(\sigma, j) = 1 - \frac{1}{N_{\mathcal{A}}} \sum_{\boldsymbol{r} \in \mathcal{A}} \frac{1}{\sigma} \frac{dE_{\boldsymbol{r}}}{d\sigma}. \tag{3.15}$$

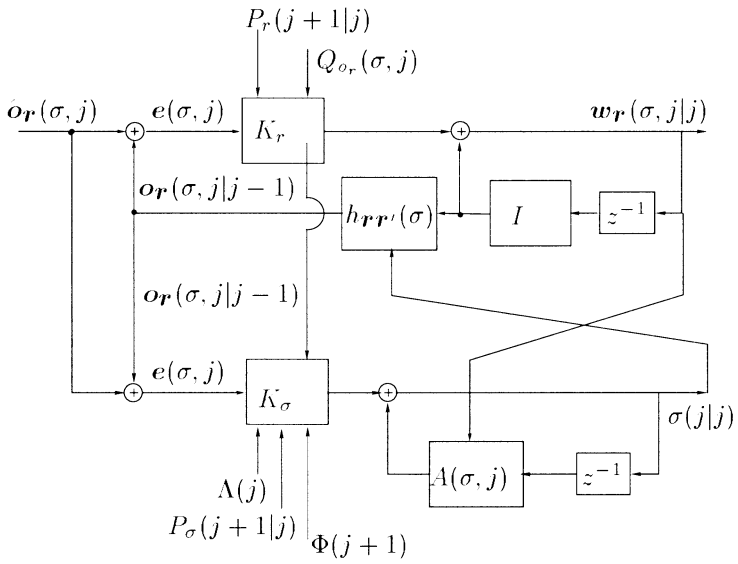

Figure 5: Detailed block diagram of the Auto-SOM. The combination of the original SOM's KF implementation and the KF formulation of the recursive parameter estimation of σ exploit the prediction error $\boldsymbol{e}(\sigma, j)$

Equation 3.13 shows that the state transition depends on the state itself. Therefore, this KF becomes the extended KF in its state covariance prediction equation, 3.14.

Finally, the RPEM gives the estimate of $\sigma(j|j)$ (see equation 3.4) by minimizing the quadratic criterion function in equation 3.1 with respect to σ. The combined algorithms presented in sections 2 and 3 establish the Auto-SOM, a learning parameter estimation method that minimizes the expected value of the prediction error for each neuron. The block diagram in Figure 5 illustrates how both algorithms depend on each other. In the upper part of the diagram, the linear KF and the model of the self-organizing learning process are shown. The output measurement $\hat{\boldsymbol{o}}_{\boldsymbol{r}}(j)$ is compared to its KF prediction $\boldsymbol{o_r}(\sigma, j|j-1)$ and amplified by the Kalman gain, which now controls the following increment of the state model. The state model describes the state transition of the filtered weights $\boldsymbol{w_r}(\sigma, j|j)$ by a time-delayed (expressed by $z^{(-1)}$) identity mapping. The predicted weights are attenuated with their corresponding value of the neighborhood function, yielding the prediction $\boldsymbol{o_r}(\sigma, j|j-1)$.

The filter estimating the width, σ, of the neighborhood function is depicted in the lower part of the diagram. Its structure is similar to the one of the SOM learning process in the upper half. Thus, the filter also exploits the prediction error $\boldsymbol{e}(\sigma, j)$ of the KF of the SOM learning process, but also uses

the SOM Kalman gain matrix, $\boldsymbol{K}_{\boldsymbol{r}}$, as well as derivatives of the prediction error $\Phi(j+1)$. On this basis, the gain $\boldsymbol{K}_\sigma$ is calculated to control the transition model of the width, σ, which was chosen to be the gradient descent of the sum of individual energy functions of the neurons.

3.1 Refinement to Guarantee Neighborhood Preservation. In the following, we are going to refine the transition of the width σ modeled by $A(\sigma, j)$ (see equation 3.15) in order to guarantee neighborhood preservation. Neighborhood preservation can be measured for the mapping $\Psi_{\mathcal{M}\to\mathcal{A}}$ from the input space $\mathcal{M}$ to the output space $\mathcal{A}$, and vice versa. The self-organizing algorithm in general is capable of preserving neighborhoods in the mapping $\Psi_{\mathcal{A}\to\mathcal{M}}$ only from the output to the input space. However, one can attempt to preserve neighborhoods in both directions by imposing constraints on the width of the neighborhood function. Typical problems for which the unconstrained SOM would finally converge to maps that violate the neighborhood preservation of the mapping $\Psi_{\mathcal{M}\to\mathcal{A}}$, are dimension-reducing applications. During the training phase, the algorithm stabilizes on bidirectional neighborhood-preserving map configurations, but when the width of the neighborhood function is further decreased, it loses these configurations. A phase transition occurs (Ritter & Schulten, 1989). From a different point of view, this means that the algorithm turns at the phase transition from a principal curve mapping to a mapping overfitting the input data. Several heuristic approaches to stabilize the training algorithm in a map configuration preserving the neighborhood, together with detailed analysis of phase transitions, have been proposed (Herrmann, 1995; Der, Balzuweit, & Herrmann, 1996). These suggest that overfitting can be avoided using the RPE method by measuring the neighborhood violations of the mapping $\Psi_{\mathcal{M}\to\mathcal{A}}$ and constraining the transition of the width σ while continuing to decrease $\epsilon(j)$. Hence, $A(\sigma, j)$ is constrained to be

$$A(\sigma, j) = \begin{cases} -\left(1 - \sum\limits_{\boldsymbol{r}\in\mathcal{A}} \frac{1}{\sigma}\frac{dE_{\boldsymbol{r}}}{d\sigma}\right) & F(int(\sigma)) > 0 \\ 1 - \sum\limits_{\boldsymbol{r}\in\mathcal{A}} \frac{1}{\sigma}\frac{dE_{\boldsymbol{r}}}{d\sigma} & F(int(\sigma)) = 0, \end{cases} \tag{3.16}$$

with F denoting the "topographic function" (Villmann, Der, Herrmann, & Martinetz, 1997), a well-known measure of neighborhood preservation (e.g., Bruske & Sommer, 1995; Bauer, Herrmann, & Villmann, 1999; Herrmann, Bauer, & Villmann, 1997). It is defined on the basis of the graph $\mathcal{D}^{\mathcal{M}}(\boldsymbol{r}, \boldsymbol{r}')$ of connections according to the induced Delaunay triangulation (Martinetz & Schulten, 1994) of the input space:

$$f_{\boldsymbol{r}}(k) \stackrel{def}{=} \#\left\{\boldsymbol{r}' | \quad \|\boldsymbol{r} - \boldsymbol{r}'\|_{max} > k\,; \quad d^{\mathcal{D}^{\mathcal{M}}}(\boldsymbol{r}, \boldsymbol{r}') = 1\right\}, \tag{3.17}$$

$$f_{\boldsymbol{r}}(-k) \stackrel{def}{=} \#\left\{\boldsymbol{r}' | \quad \|\boldsymbol{r} - \boldsymbol{r}'\| = 1\,; \qquad d^{\mathcal{D}^{\mathcal{M}}}(\boldsymbol{r}, \boldsymbol{r}') > k\right\}, \tag{3.18}$$

$$F(k) \stackrel{def}{=} \begin{cases} \frac{1}{N}\sum_{\boldsymbol{r}'\in\mathcal{A}} f_{\boldsymbol{r}'}(k) & k > 0 \\ F(1) + F(-1) & k = 0 \\ \frac{1}{N}\sum_{\boldsymbol{r}'\in\mathcal{A}} f_{\boldsymbol{r}'}(k) & k < 0. \end{cases} \tag{3.19}$$

This measure calculates the number and the order k of neighborhood violations and is the only measure known to account for the violation order. This information is exploited in the formulation of the transition of the width (see equation 3.16). The transition model will increase σ if violations are measured that are of higher order than the actual width $\sigma(j)$. In case these orders of violation occur, σ will not be decreased further until the algorithm has restored a neighborhood-preserving mapping $\Psi_{\mathcal{M}\to\mathcal{A}}$. The results are demonstrated in the third example (see Figures 8 and 9).

Finally, we guarantee the convergence of the learning algorithm to a neighborhood-preserving mapping $\Psi_{\mathcal{A}\to\mathcal{M}}$ from the output to the input space by applying the constrained transition of σ in equation 3.16 with the topographic function $F(-int(\sigma))$. In the following we demonstrate this by presenting results for particular applications.

4 Applications

The results obtained with Auto-SOM are demonstrated on three examples covering the main application problems: (1) approximation of input data distribution (two-dimensional uniformly distributed trained by a two-dimensional feature map), (2) dimension reduction with neighborhood preservation (three-dimensional nonlinear manifold trained by a two-dimensional feature map), and (3) dimension reduction. Finally, we demonstrate the performance of Auto-SOM using a real high-dimensional application problem by comparing it to the results with generative topographic mapping (Bishop et al., 1996).

For the first of these, we consider the proposed training method to reflect the uniform distribution of the input data $\Omega = \{\boldsymbol{v}(j)|0 \leq (\boldsymbol{v})_1 \leq 1 \wedge 0 \leq (\boldsymbol{v})_2 \leq 1\}$ by the distribution of the neurons. The training result using a 10×10 feature map is shown at the top of Figure 6. We find the nodes of the feature map uniformly distributed over the input data set, just moving within their Voronoi cell. The quantization error q_{err} (see equation 3.9) reaches 0.0021 after 70,000 learning steps. $\sigma(70{,}000)$ is equal to 0.96. These results can be reached by experts on SOM too, as can be deduced from our results with the original SOM. We took the estimated learning parameters as a guideline for our choice of the initial values and rates of decrease (see Figure 6). The final SOM trained with the original learning algorithm shows nearly the same quantization error after 82,000 learning steps and $\sigma(82{,}000) = 0.96$. In contrast to the exponentially decreasing learning parameters usually used with the original SOM, Figure 6 shows that the Auto-SOM estimates of the learning parameters decrease faster. The estimate of $\sigma(j)$ is at first decreasing much faster; later, the decreasing rate reduces

until it becomes very slow for $\sigma(j) < 1$. This is due to the individual energy function $E_{\boldsymbol{r}}$, equation 3.12, which depends on the learning coefficients. The individual learning coefficients, depicted only for the winner neuron (see Figure 6), decrease from values 0.3 to 0.06 during the training process. While $\sigma(j)$ is smoothly estimated, the estimates of the learning coefficients are noisier during the whole learning process, as demonstrated by the learning coefficient $\boldsymbol{K}_{\boldsymbol{r}'}(j)$ in Figure 6. This is due to the highly inaccurate model of the learning process at the beginning of the training. The weights are far from being static, which is modeled by equation 2.5. Additionally, the system and measurement noise covariances, $\boldsymbol{Q}_{\boldsymbol{w}_r}$ and $\boldsymbol{Q}_{\boldsymbol{o}_{\boldsymbol{r}'}}$, respectively, are in the same order of magnitude, which results in considerable fluctuations between the confidence in either the measurements or the process model. These fluctuations are reflected in the Kalman gain elements. High values of the Kalman gain express high confidence in the measurements $\hat{\boldsymbol{o}}_{\boldsymbol{r}}$; in contrast, low values express high confidence in the process model.

During the organization process, the system noise covariance $\boldsymbol{Q}_{\boldsymbol{w}_r}$ and the measurement noise covariances $Q_{\boldsymbol{o}_r}$ separate from each other by orders of magnitude. $\boldsymbol{Q}_{\boldsymbol{w}_r}$ tends to zero, expressing high confidence in the static process model, while $Q_{\boldsymbol{o}_r}$ tends to a limit, which depends on the input data set and the number of neurons $N_{\mathcal{A}}$ (see equation 2.13).

A comparison of Auto-SOM with the batch learning of SOMs is of high interest, because batch learning is known to converge within an order of magnitude fewer learning steps than the original SOM and additionally does not involve the learning rate. Therefore, we also trained the SOM with the batch-learning method. We started with the training result gained by Auto-SOM after 50,000 learning steps, which is already ordered. Within 5000 steps, batch learning was able to achieve a well-ordered map with equal quality ($q_{err} = 0.0022$) as those of Auto-SOM and the original SOM. Therefore, batch SOM here needs fewer learning steps than Auto-SOM. However, batch-learning techniques get stuck more often in local minima, because these techniques smooth the noisy optimization process. Therefore, they are faster, but the noise is known to be a remedy to escape from local minima. For instance, our final example of high-dimensional learning patterns trained with batch SOM did not lead to any map equal in quality to the maps from the original or the Auto-SOM.

For the second example, the Auto-SOM is trained with data from nonlinear manifolds. The data are chosen from the surface of a hemisphere (see Figure 7). Appropriate learning parameters are hard to guess, because the self-organizing algorithm has to find the projection of three-dimensional inputs onto the two-dimensional lattice of neurons. However, the Auto-SOM estimates the learning parameters σ and $\boldsymbol{K}_{\boldsymbol{r}'}$ by itself, as shown in Figure 7. Here they decrease at first linearly, but become sublinear when σ is between 2 and 1. With $\sigma = 0.9$ the map plotted in Figure 7 is achieved. Here the quantization error is 0.0003, a substantial improvement over the method proposed by Haese (1999), which was trained until σ was equal

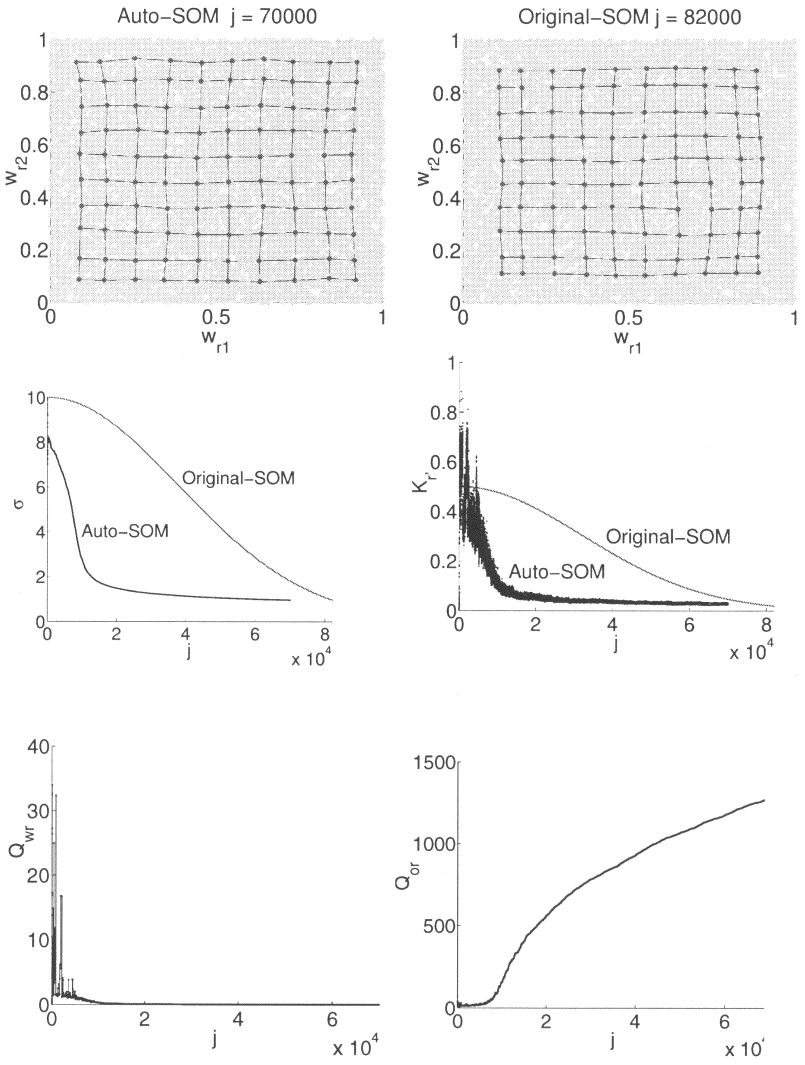

Figure 6: Feature map trained on uniformly distributed two-dimensional data using Auto-SOM with the estimated learning parameters σ, $\boldsymbol{K_{r'}}$ as well as the quantization error q_{err}, and the covariance matrices of the neuron's outputs and the weights, $\boldsymbol{Q_{o_{r'}}}$ and $\boldsymbol{Q_{w_{r'}}}$, respectively.

to 0.7 but remained with a higher quantization error $q_{err} = 0.0128$. Further training of the map in Figure 7 will continue in reducing the quantization error, but because the energy minimum is very flat, this occurs only very

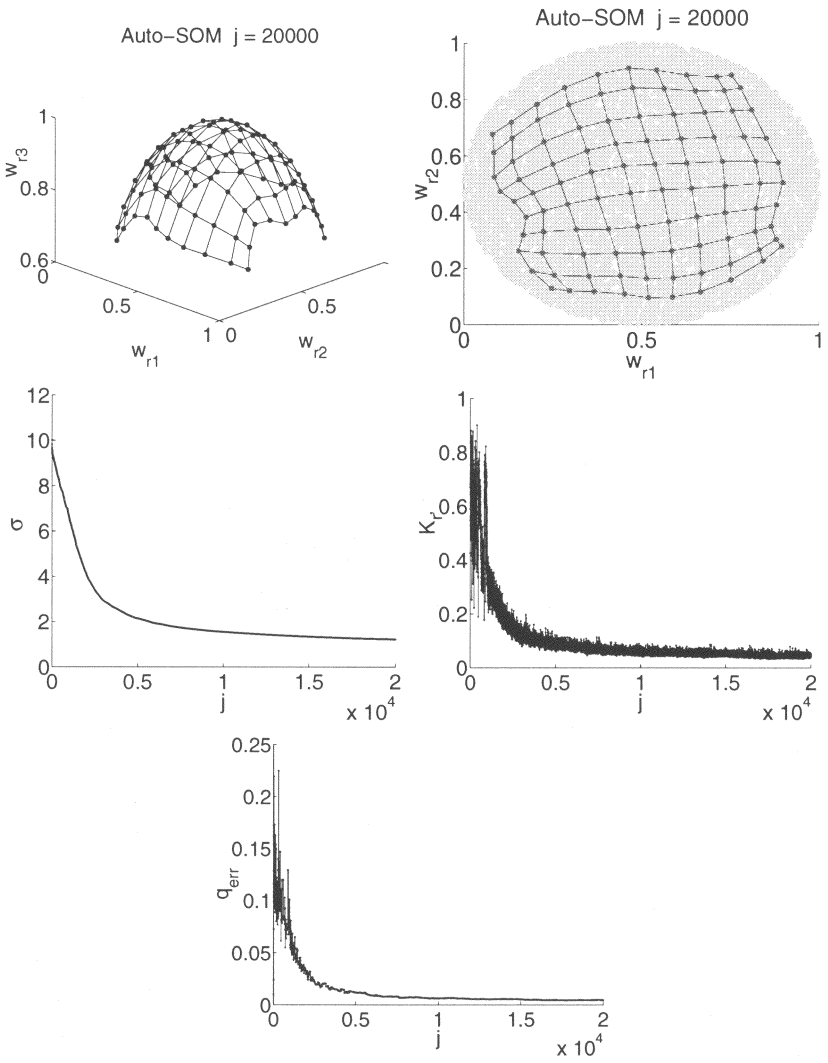

Figure 7: Auto-SOM learning results on training data taken from a nonlinear manifold. Auto-SOM estimates of σ and $\boldsymbol{K_{r'}}$, as well as the quantization error q_{err}, are shown during the course of training.

slowly. For the third and last example, we demonstrate that neighborhoods can be preserved for both mappings, $\Psi_{\mathcal{A}\to\mathcal{M}}$ and $\Psi_{\mathcal{M}\to\mathcal{A}}$, if we impose the constraints in equation 3.16 on the width of the neighborhood function. The training data are selected randomly from a stripe-shaped area—here a rect-

angle with $\Omega = \{v | 0 \leq (v)_1 \leq 3 \wedge 0 \leq (v)_2 \leq 6\}$. If the smaller extension of the input data set, in direction $(v)_1$, is considered excessively as a dimension due to noise, a one-dimensional feature map is chosen to quantize the data. Without any constraints on the width σ, the feature map is guided to the configuration in Figure 8. The point of phase transition can be observed in the courses of $\boldsymbol{K_{r'}}$ and $\boldsymbol{Q_{w_r}}$. After about 24,000 learning steps, the elements of the covariance matrix $\boldsymbol{Q_{w_r}}$ increase, indicating that the stable map configuration just reached is lost. The distance between the weights and the centers of their Voronoi cells first tends to increase, before converging to a new map configuration with a smaller quantization error. This configuration preserves neighborhoods only from the output into the input space. In order to preserve neighborhoods for the mapping in the other direction, we need to remain in the first stable map configuration. This is achieved by refining the transition model of the width σ as proposed in equation 3.16. Thereby, the width of the neighborhood function remains above the theoretical value $\sigma_{crit} \approx 2.02s$ (Ritter & Schulten, 1989) with $s = 3$ being the width of the stripe. The width σ as well as the estimated learning coefficient $\boldsymbol{K_{r'}}$ and elements of the covariance matrices $\boldsymbol{Q_{w_r}}$ and $\boldsymbol{Q_{o_r}}$ are shown in Figure 9. They converge in the form of an attenuated oscillation. Finally, a mapping is reached that does preserve neighborhood relations for the mappings $\Psi_{\mathcal{A}\to\mathcal{M}}$ and $\Psi_{\mathcal{M}\to\mathcal{A}}$, but it has a higher quantization error than the map in Figure 8.

Finally, we show that Auto-SOM can find good mappings for real high-dimensional input data, which have to be inspected and therefore visualized. We compare the results of Auto-SOM with those of GTM proposed by Bishop et al. (1996) and further developed in Bishop, Svensen, and Williams (1997). Both algorithms can be regarded as kernel smoothers (Bishop, Svensen, & Williams, 1998). As the comparison of GTM and the batch SOM version in Bishop et al. (1998) has revealed once more the lack of guidelines for how to shrink the width of the neighborhood function with batch SOMs, now with Auto-SOM we provide a competing algorithm to GTM, whose learning parameters are determined automatically. Therefore, we compare both algorithms, Auto-SOM and GTM, using the problem of determining the fraction of oil in a multiphase pipeline (Bishop & James, 1993) . The pipeline carries a mixture of oil, water, and gas. Twelve measurements of the attenuation of gamma beams are taken to decide whether one of the phases (oil, water, or gas) belongs to laminar, homogeneous, or annular flow. The training and testing data set consist of 1000 data points, each drawn with equal probability from one of the three classes.

Figure 10 shows the different visualizations of the input data space using Auto-SOM without the refinements in section 3.1 and GTM, respectively. The number of patterns that neurons at location $\boldsymbol{r} = [r_1, r_2]^T$ respond to is plotted, assigning the class of flow by different gray levels. Auto-SOM shows high resolution of the nonempty parts of the input space with a small quantization error, 0.1294. Additionally, only seven neurons on the map are

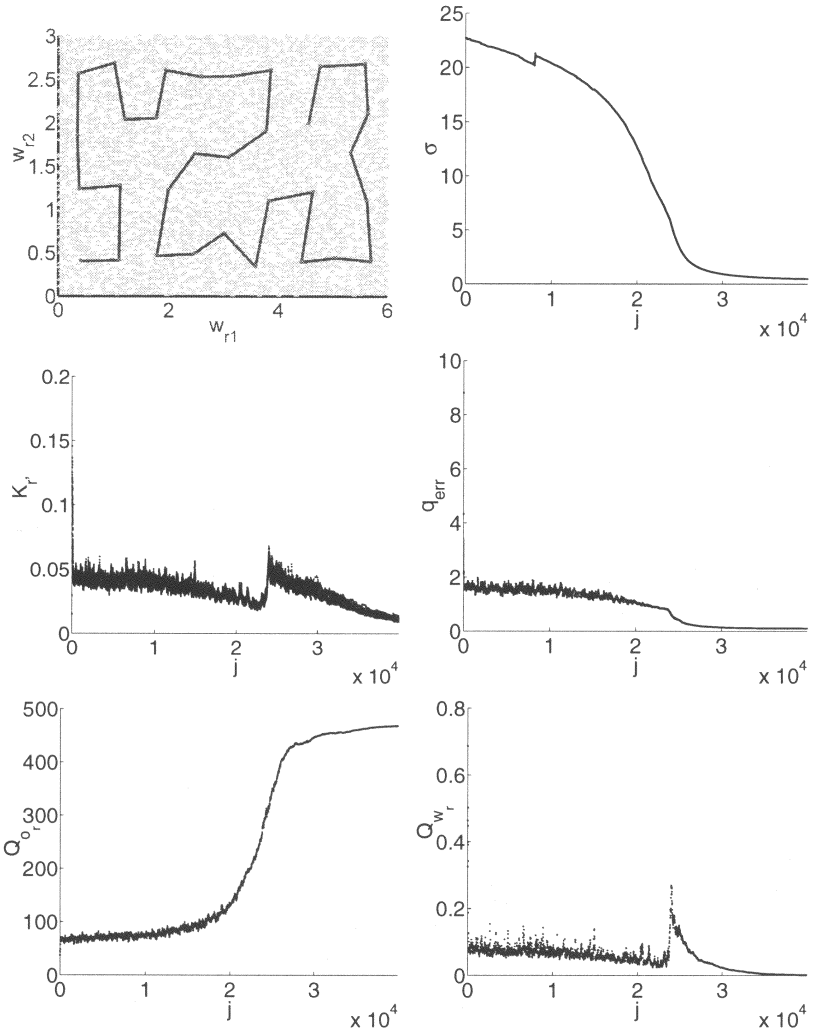

Figure 8: One-dimensional feature map trained on uniformly distributed two-dimensional data using Auto-SOM with the estimated learning parameters σ, $\boldsymbol{K_{r'}}$ as well as the quantization error q_{err} and the covariance matrices of the neuron's outputs and the weights, $\boldsymbol{Qo_{r'}}$ and $\boldsymbol{Qw_{r'}}$, respectively.

stimulated by patterns of different classes. The topology is preserved best in mapping direction $\Psi_{\mathcal{M}\to\mathcal{A}}$ with $F(k > 18) = 0$ and in mapping direction $\Psi_{\mathcal{A}\to\mathcal{M}}$ $F(-k > 9) = 0$. In comparison, the GTM with a 12×12 grid of nonlinear basis functions, with common width twice the distance between

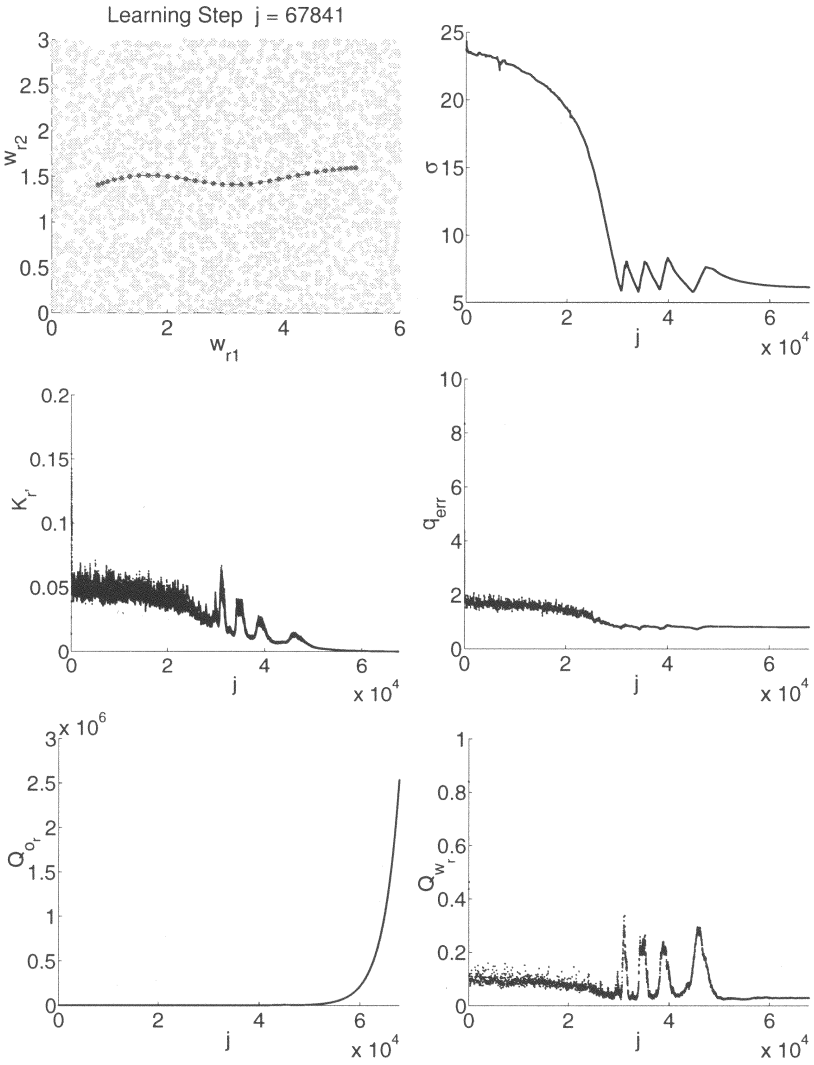

Figure 9: One-dimensional feature map trained to stabilize in a bidirectional neighborhood-preserving map configuration. The estimated learning parameters σ, $\boldsymbol{K_{r'}}$, the quantization error q_{err}, and the elements of the covariance matrices, $\boldsymbol{Qo_r}$ and $\boldsymbol{Qw_r}$, respectively, are depicted during the course of training.

two neighboring basis function centers and constant degree of weight regularization, equal 1000.0 for 1000 iterations, resolves the input space as accurately as Auto-SOM with a quantization error of 0.1272, whereas the number

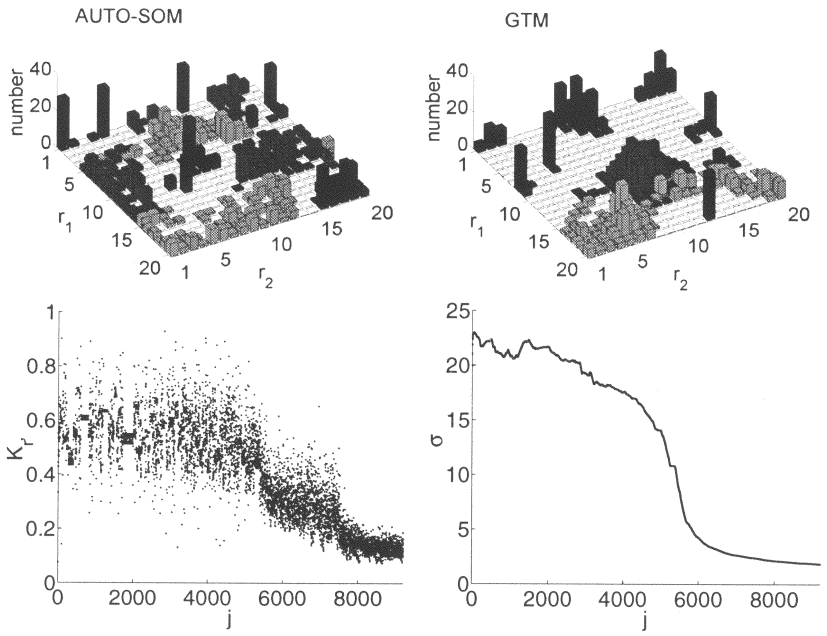

Figure 10: (Top) Auto-SOM and GTM learning results on training data taken from flow measurements in a three-phase pipeline. (Bottom) Auto-SOM estimates of σ and $\boldsymbol{K_{r'}}$ shown during the course of training.

of neurons always responding to different flow configurations nearly doubles to 13. The topology preservation achieved by the GTM is best in mapping direction $\Psi_{\mathcal{A}\to\mathcal{M}}$ with $F(-k > 7) = 0$ and in mapping direction $\Psi_{\mathcal{M}\to\mathcal{A}}$ $F(k > 16) = 0$. Thus, the Auto-SOM performs as well as the GTM does. Further results on this training set were studied applying the original SOM as well as the batch SOM. The original SOM converged after 60,000 training cycles to the best mapping with quantization error 0.098 and a minimum of 9 neurons always responding to different flow configurations. The topology preservation achieved is worse with $F(-k > 16) = 0$ in mapping direction $\Psi_{\mathcal{A}\to\mathcal{M}}$ and $F(k > 8) = 0$ in mapping direction $\Psi_{\mathcal{M}\to\mathcal{A}}$. Batch SOM did not show better results. From this we conclude that the algorithms converge to maps with very similar qualities in topology preservation and quantization of the test patterns. However, the Auto-SOM and the GTM algorithm render parameter studies unnecessary at the cost of greater computational expense. The greatest additional part to be performed with Auto-SOM in comparison to the batch SOM version is described by the linear KF equations, 2.16, 2.17, and 2.20, which increases with the number of neurons $N_{\mathcal{A}}$. However, these additional costs are only a fraction of those resulting from

even one additional parameter study, because the greatest part of CPU time is needed for searching the winner neurons.

Therefore, these results show that Auto-SOM is a feasible as well as a tractable method for estimating learning coefficients and the width of the neighborhood function, which leads to good results in all application domains of the self-organizing feature map. Consequently, the Auto-SOM is capable of relieving the user from laborious parameter studies. Furthermore, it has also been demonstrated how the underlying model of the RPEM can be modified to yield bidirectional neighborhood preserving mappings in dimension-reducing applications.

5 Conclusions

We have presented the Auto-SOM, an optimal learning parameter estimation method for the training of feature maps. A KF implementation of the original SOM calculates the learning coefficient for each neuron to estimate the weights' adaptation process with minimal variance. From its estimation error, the proper width of the neighborhood function is calculated by the RPEM, which acts on the basis of the neurons' individual energy functions. Both algorithms establish the Auto-SOM, which converges automatically to neighborhood-preserving feature mappings from the output into the input space. This is guaranteed by the topographic function measuring violations of neighborhood preservation. The performance and the features of Auto-SOM were discussed on example applications. Auto-SOM is demonstrated to yield best results, thus relieving the user from laborious parameter studies.

In addition, Auto-SOM can be modified to yield bidirectional neighborhood-preserving mappings. This feature is highly appreciated for vector quantization tasks of noisy data or any applications where it is desirable for the SOM to stabilize in neighborhood-preserving map configurations.

Furthermore, Auto-SOM can provide insight into the data under examination by analyzing the estimated learning parameters as well as the noise covariances. This has the potential to reveal further aspects of SOM features and its convergence properties.

Acknowledgments

K. H. was supported by the German Aerospace Center and German Research Foundation. G. J. G. was supported by grants from the NIH and DoD.

References

Bauer, H.-U., Der, R., & Herrmann, M. (1996). Controling the magnification factor of self-organizing feature maps. *Neural Computation, 8*, 757–765.

Bauer, H.-U., Herrmann, M., & Villmann, T. (1999). Neural maps and topographic vector quantization. *Neural Networks, 12*, 659–676.

Bauer, H.-U., & Villmann, T. (1997). Growing a hypercubical output space in a self-organizing feature map. *IEEE Transaction on Neural Networks, 8*, 218–226.

Bishop, C. M., & James, G. (1993). Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research, A327*, 580–593.

Bishop, C. M., Svensen, M., & Williams, C. K. I. (1996). *GTM: The generative topographic mapping* (Tech. Rep.). Birmingham, UK: Aston University.

Bishop, C. M., Svensen, M., & Williams, C. K. I. (1997). Magnification factors of the SOM and GTM Algorithm. In *Proceedings of the Workshop on Self-Organizing Maps (WSOM '97), Espoo, Finland* (pp. 333–338). Helsinki: Helsinki University of Technology.

Bishop, C. M., Svensen, M., & Williams, C. K. I. (1998). GTM: The generative topographic mapping. *Neural Computation, 10*(1), 215–234.

Bruske, J., & Sommer, G. (1995). Dynamic cell structure learns perfectly topology. *Neural Computation, 7*, 845–865.

Der, R., Balzuweit, G., & Herrmann, M. (1996). Constructing principal manifolds in sparse data sets by self-organizing maps using self-regulating neighbourhood width. In *Proc. ICNN'95, IEEE Int. Conf. on Neural Networks*.

Erwin, E., Obermayer, K., & Schulten, K. (1992). Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics, 67*, 47–55.

Haese, K. (1990). *Optimierung des Schätzverhaltens eines Kalman-Filters*. Unpublished master's thesis, RWTH Aachen, Germany.

Haese, K. (1998). Self-organizing feature map with self-adjusting learning parameters. *IEEE Transactions on Neural Network, 9*(6), 1270–1278.

Haese, K. (1999). Kalman filter implementation of self-organizing feature maps. *Neural Computation, 11*(5), 1211–1233.

Haese, K., & vom Stein, H.-D. (1996). Fast self-organizing of n-dimensional topology maps. In *VIII European Signal Processing Conference, Trieste, Italy* (pp. 835–838). EURASIP Edizioni LINT Trieste.

Herrmann, M. (1995). Self-organizing feature maps with self-organizing neighborhood widths. In *Proc. ICNN'95, IEEE Int. Conf. on Neural Networks, 6* (pp. 2998–3003). Piscataway, NJ: IEEE Service Center.

Herrmann, M., Bauer, H.-U., & Villmann, T. (1997). A comparison of topography measures for neural maps. In *Proceedings of the Workshop on Self-Organizing Maps (WSOM'97), Espoo, Finland* (pp. 274–279). Helsinki: Helsinki University of Technology.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*, 59–69.

Kohonen, T. (1984). *Self-organization and associative memory*. Berlin: Springer-Verlag.

Kohonen, T. (1985). *Self-organizing maps*. Berlin: Springer-Verlag.

Ljung, L. (1999). *System identification: Theory for the user*. Englewood Cliffs, NJ: Prentice Hall.

Ljung, L., & Söderström, T. (1983). *Theory and practice of recursive identification*. Cambridge, MA: MIT Press.

Martinetz, T., & Schulten, K. (1994). Topology representing network. *Neural Networks, 7*(3), 507–522.

Ritter, H., & Schulten, K. (1989). Convergence properties of Kohonen's topology conserving maps: Fluctuation, stability and dimension selection. *Biological Cybernetics, 60*, 59–71.

Tolat, V. (1990). An analysis of Kohonen's self-organizing maps using a system of energy functions. *Biological Cybernetics, 64*, 155–164.

Villmann, T. (in press a). Analysis of psychotherapy process data using neural maps. *IEEE Transactions on Neural Networks*.

Villmann, T. (in press b). Application of magnification control for the neural gas network in a sensorimotor architectur for robot navigation. In *Fortschrittsberichte des VDI, Workshop SOAVE 2000, Ilmenau*. VDI-Verlag.

Villmann, T., Der, R., Herrmann, M., & Martinetz, T. M. (1997). Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Transaction on Neural Networks, 8*(2), 256–266.

Yin, H., & Allinson, N. M. (1995). On the distribution and convergence of feature space in self-organizing maps. *Neural Computation, 7*(7), 1178–1187.

Received May 17, 1999; accepted June 14, 2000.