INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Growing a Hypercubical Output Space in a Self-Organizing Feature Map

H.-U. Bauer *, Th. Villmann †

TR-95-030

July 1995

*Permanent Adress: Institut für theor. Physik, Univ. Frankfurt/Main, Robert-Mayer-Str. 8-10, 60054 Frankfurt, Germany, email: bauer@chaos.uni-frankfurt.de

†Permanent Adress: Fachbereich Mathematik/Informatik, Universität Leipzig, Augustusplatz 10/11, 04109 Leipzig, Germany, email: villmann@informatik.uni-leipzig.d400.de

Abstract

Neural maps project data given in a (possibly high-dimensional) input space onto a neuron position in a (usually low-dimensional) output space grid. An important property of this projection is the preservation of neighborhoods; neighboring neurons in output space respond to neighboring data points in input space. To achieve this preservation in an optimal way during learning, the topology of the output space has to roughly match the effective structure of the data in the input space. We here present a growth algorithm, called the GSOM, which enhances a widespread map self-organization process, Kohonen's Self-Organizing Feature Map (SOFM), by an adaptation of the output space grid during learning. During the procedure the output space structure is restricted to a general hypercubical shape, with the overall dimensionality of the grid and its extensions along the different directions being subject of the adaptation. This constraint distinguishes the present algorithm from other, less or not constrained approaches to the problem of map topology adaptation. Depending on the embedding of neural maps in larger information processing systems, a regular neuronal grid can be essential for a successful operation of the overall system. We apply our GSOM-algorithm to three examples, two of which involve real world data. Using recently developed methods for measuring the degree of neighborhood preservation in neural maps, we find the GSOM-algorithm to produce maps which preserve neighborhoods in a nearly optimal fashion.

1 Introduction

Neural maps constitute an important neural network paradigm. In brains, neural maps occur in all sensory modalities as well as in motor areas. In technical contexts, neural maps are utilized in the fashion of neighborhood preserving vector quantizers. In both cases these networks project (map) data from some (possibly high-dimensional) input space onto a position in some output space, such that a continuous change of a parameter of the input data leads to a continuous change of the position of a localized excitation in the neural map.

To achieve this projection, neural maps are self-organized by unsupervised learning schemes. First such models were proposed by von der Malsburg [1] and Willshaw and von der Malsburg [2], later Kohonen's self-organizing feature map algorithm (SOFM) found a wide distribution ([3], for recent reviews see [4] or [5]). In the present article, we also base our ideas on this latter algorithm, which will be recapitulated in some detail in the second section. In the SOFM, as well as in most other map formation algorithms, neurons are arranged as an output space grid A. The grid can in principle be of any dimensionality, or can extend to any dimension along its individual directions. Associated to the neurons are weights which determine a particular position in the input space and which are adapted during the learning phase. In biological language these weights determine a receptive field. If the vector quantization property is to be stressed, one would rather call the neuron plus its weight vector a codebook vector.

As was mentioned above, the characteristic feature distinguishing neural maps from other neural network paradigms, or from regular vector quantizers, is the preservation of neighborhoods. Obviously, this feature depends on the choice of the output space topology. As a very simple example, consider the map from the unit square onto an output space shaped like a line, a square or a cube. Of these, only the square will be able to preserve neighborhoods (in a sense, which will be clarified in the fifth section). In contrast, the line and cube will induce neighborhood violations (see Fig. 1.1). The proper dimensionality required by the input data is usually not known a priori (consider, e.g. speech data, which are usually represented in a 10-20 dimensional input space, but fill only a lower-dimensional submanifold in this input space). Yet the output space grid has to be specified prior to learning. Therefore, a problem emerges if neighborhood preservation is to be optimized by the network. This problem is specific to technical applications of neural map algorithms since in biological modeling the output space topology is usually chosen following assumptions about the connectivity of the underlying tissue.

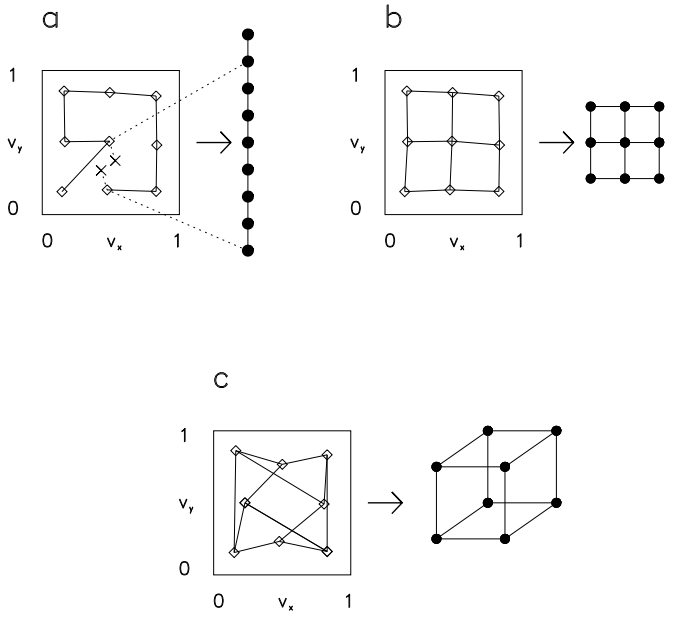

Figure 1.1: SOFMs of the unit square onto neuron grids of different topology: a: one-dimensional (line with $N = 9$ neurons), b: two-dimensional (square with $N = 3 \times 3 = 9$ neurons), c: three-dimensional (cube with $N = 2 \times 2 \times 2 = 8$ neurons). In the one-dimensional case, nearby stimuli (crosses) can be mapped onto well separated positions in the output space. In the three-dimensional case, neighboring neurons in the output space can have well separated receptive field centers in the input space, as is manifest by the two longest connection lines. Both cases constitute neighborhood violations, which can be avoided only by matching input and output spaces (case b).

Two types of strategies have been developed to solve this problem. First one can self-organize several maps of different output space topology, measure their degree of neighborhood preservation, and select the map with the optimal preservation value. To this purpose, measures for neighborhood preservation have been developed, the topographic product P [6] and the topographic function Φ [7, 8]. This post-training selection strategy has been demonstrated to work not only with regard to the overall dimensionality of the output space, but also with regard to the dimensions along the individual directions of the output grid [9].

The second approach involves advanced learning schemes which not only adapt the receptive field center positions of neurons, but also the topology of the output space itself. Examples for such algorithms include the "Topology Representing Network" TRN [10], the "Growing Cell Structure"-algorithm [11], and other schemes [12], [13]. All these algorithms yield an output space connectivity which is not easy to formalize as a simple structure, like e.g. as

a hypercube (with varying dimensions along the different directions). Instead, it has to be described as a symmetric graph of possibly very complicated structure.

Most often, a neural map projecting data of technical origin is embedded in a larger system which is to achieve some task. Then, subsequent operations have to be performed on the projected data. Depending on the task, a general graph structure of the output space can prohibit such further processing. Consider the example of data points lying on a low-dimensional submanifold of a high-dimensional data space which are to be visualized in a color or gray-value code, with the (up to) three data dimensions being spanned by the (up to) three dimensions of some color space. Here, it is essential to have not only neighborhood preservation per se, but neighborhood preservation in an orderly arrangement of neurons in the output space. Only then a transformation of output space position into color indices [14, 15] can be performed. A second example involves the application of the PSOM-algorithm [16, 17] to obtain a continuous, "parametrized" representation of the input data in the output space. In the PSOM-framework, the map manifold is expanded in a small set of "basis manifolds", with the expansion coefficients being determined at a small number of reference points. The mapping values at these reference points can be obtained using the regular SOFM-algorithm. For the expansion to work, the reference points have to span the input space in a neighborhood preserving way, and the same time have to be arranged on a (generalized) hypercubical lattice.

Even though in both examples the embedding of the neural map in a larger system (the visualization scheme and the PSOM, resp.) puts constraints on the possible structure of the output space, the problem of output space adaptation to the a priori unknown structure underlying the input data remains. To get around this problem, a method is needed which can adapt output space topologies in an unsupervised way, but at the same time obeys constraints on the possible connectivity.

Here we present an algorithm which achieves such an adaptation by growing hypercubical output spaces up to a prespecified maximum number of nodes. The algorithm, called the GSOM for "Growing Self-Organizing Map" and described in detail in the third section, is based on an evaluation of the difference vectors between data points and corresponding codebook vectors in the Voronoi cells of individual neurons. In the fourth section, we apply the GSOM to a pedagogical toy problem example and to two more challenging data sets, one set of speech data, and one set of precompressed image data. In the fifth section the resulting GSOM-maps are evaluated with regard to their neighborhood preservation properties. This is an interesting consistency check, because the neighborhood preservation properties of the map were not evaluated and utilized during the growth procedure and,

hence, constitute an independent observable. To quantify the neighborhood preservation, we use both the topographic product P and the topographic function Φ, in this way also providing a comparison between these two competing measures. A discussion of our results concludes the paper.

2 Kohonen Algorithm for Self-Organizing Feature Maps

The growth algorithm we develop in this paper is based on Kohonen's Self-Organizing Feature Map algorithm (SOFM). The SOFM describes a map $\Psi_{V\to A}$ which projects data points $\mathbf{v}$ in some input space $V \subseteq \Re^{d_V}$ to a neuron position $\mathbf{r}$ in some output space A. Each neuron has associated to it a pointer $\mathbf{w_r} \in \Re^{d_V}$. The mapping prescription is a winner take all rule, i.e. $\mathbf{v}$ is mapped onto that neuron $\mathbf{s} \in A$ the pointer $\mathbf{w_s}$ of which is closest to $\mathbf{v}$,

$$\Psi_{V\to A} : \mathbf{v} \mapsto \mathbf{s} = \mathrm{argmin}_{\mathbf{r}} \|\mathbf{v} - \mathbf{w_r}\| . \tag{2.1}$$

This is reminiscent of a usual vector quantizer. However, in the SOFM the neurons are not just indexed but are arranged on a regular grid A. A common choice for A is a (two-dimensional) rectangle. Other ordered arrangements are also admissible. In the present paper we require the output space to consist of a "generalized rectangle", i.e. we require it to be orthogonally organized, but with variable overall dimensionality and variable dimensions along the individual directions. Lacking a better nomenclature, we also call this structure a hypercube, ignoring the possibly differing extensions along the different directions of A. The structure of A can be cast in a formal way by writing

$$\begin{aligned} &\mathbf{r} = (i, j, k, ...), \\ &1 \le i \le N_1, \quad 1 \le j \le N_2, ... \quad . \end{aligned} \tag{2.2}$$

$N = N_1 \times N_2 \times ...$ denotes the overall number of nodes in the map.

The main part of the SOFM algorithm consists in an adaptation procedure for the pointers $\mathbf{w_r}$ such that the input data is mapped to the output space in a most faithful way. To this purpose a sequence of data points $\mathbf{v}$ is presented to the map, the current most proximate neuron $\mathbf{s}$ is determined, and the pointer $\mathbf{w_s}$ as well as the pointers $\mathbf{w_r}$ of neurons in the neighborhood of $\mathbf{s}$ are shifted towards $\mathbf{v}$,

$$\Delta \mathbf{w_r} = \epsilon h_{\mathbf{rs}} (\mathbf{v} - \mathbf{w_r}). \tag{2.3}$$

The property of "being in the neighborhood of $\mathbf{s}$" is captured by the neighborhood function $h_{\mathbf{rs}}$, which is evaluated in the output space, and which is usually chosen to be of Gaussian shape,

$$h_{\mathbf{rs}} = \exp\left(-\frac{\|\mathbf{r}-\mathbf{s}\|^2}{\sigma^2}\right). \tag{2.4}$$

Now what does mapped in a "most faithful" way mean in this context? Considering that apart from the property of neighborhood preservation, the functioning of the map resembles that of a vector quantizer, one could consider the total reconstruction error as a quality measure for the map. In fact, the SOFM algorithm approaches the well-known LBG-Algorithm [18] in the limit $\sigma \to 0$ (no interaction between nodes) and, hence, reproduces its optimization properties in this limit. However, the distinguishing feature of the neural map paradigm is the interaction between neurons in the $\sigma \neq 0$ regime. The interaction is implemented here via the $h_{\mathbf{rs}}$-term in Eq. (2.3) which causes neighboring neurons in A to respond to neighboring data points in V. In biological applications of this algorithm this is the essence for modeling sensotopic brain maps, for example retinotopic maps [19]. In technical contexts, the virtue of neighborhood preservation depends on the particular application at hand; two examples were mentioned in the introduction and more will be given in the discussion.

Violation or preservation of neighborhoods in SOFMs are influenced by two features of the algorithm. First, the width of the neighborhood function σ determines in a loose way of speaking the rigidity of the map, with small values of σ allowing fine details of the data manifold in input space to be resolved. Large values of σ force the map to average out directions of small extension in the data manifold. Since the overall structure of the map can get trapped in twisted states if σ is chosen to be too small [20], one usually starts the algorithm with a large value of σ (a sizable fraction of the grid size) in order to ensure rough preordering. σ is then decreased to successively resolve finer details of the data manifold. A time-honored reduction scheme is exponential reduction, [1]

$$\sigma(t) = \sigma_0 \exp\left(\frac{t}{\tau_\sigma}\right). \tag{2.5}$$

The σ-reduction scheme ensures that the map actually approaches a neighborhood preserving final structure, provided that such a structure exists. If the topology of the output space does not match that of the data manifold, neighborhood violations are inevitable. This is

[1] To have the map approach a (rather) constant final state, one often also exponentially reduces the width of the learning step size ϵ, $\epsilon(t) = \epsilon_0 \exp\left(\frac{t}{\tau_\epsilon}\right)$. Convergence is guaranteed, however, for algebraic decay only [21].

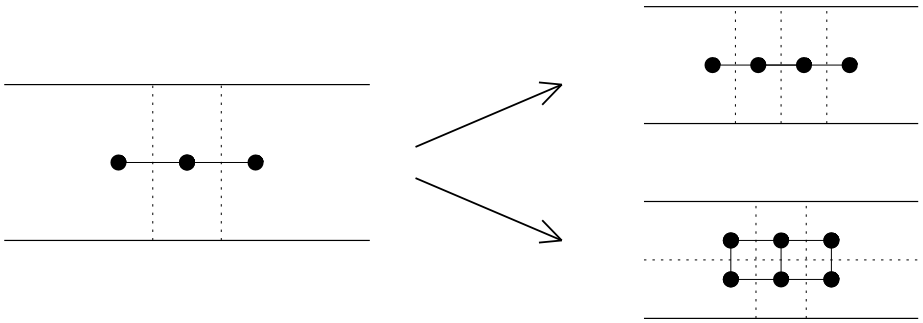

Figure 3.1: Illustration of the basic decision to be made during the growth procedure: an output space lattice can either grow in a direction, which had a nonvanishing extension previously (upper case) or it can extend into a new direction (lower case).

the case for too low-dimensional output spaces as well as for too high-dimensional spaces (see Fig. 1.1). This argument leads us to the second feature of the SOFM which has to be chosen correctly before learning, which is the topology of the output space. The impact of an incorrect output space topology on, e.g., SOFM-based speech recognition schemes has been discussed previously [6]. Using the topographic product P, or an improved variant, the topographic function Φ [8], as a means to quantify neighborhood preservation, one can try out different output space topologies, and chose the one which preserves neighborhoods best. This rather cumbersome method can be dispensed with using the following algorithm.

3 The GSOM-algorithm for maps with hypercubical output spaces

Our new growth algorithm, the GSOM, starts from an initial 2-neuron configuration, learns using the regular SOFM-algorithm, adds neurons to the output space according to a criterion to be described below, learns again, adds again, etc., until a prespecified maximum number of neurons is distributed. During this procedure, the output space topology remains to be of the form (2.2), with $N_{j>d} = 1$, where d is the current dimensionality of the output space grid A. So the initial configuration is $2 \times 1 \times 1 \times ...$, $d = 1$. From there it can grow either by adding nodes in one of the directions which are already spanned by the output space, i.e. by having $N_i \rightarrow N_i + 1$, $i \leq d$, or by adding a new dimension, i.e. $(N_{d+1}=1) \rightarrow (N_{d+1}=2)$, $d \rightarrow d + 1$ (see Fig. 3.1). Which of the existing dimensions is to grow, or whether a new dimension is to be introduced, is decided on the basis of the fluctuations within the masked Voronoi cells of the neurons. The masked Voronoi cell $\Omega_{\mathbf{r}} \subseteq V$ is the subset of data points

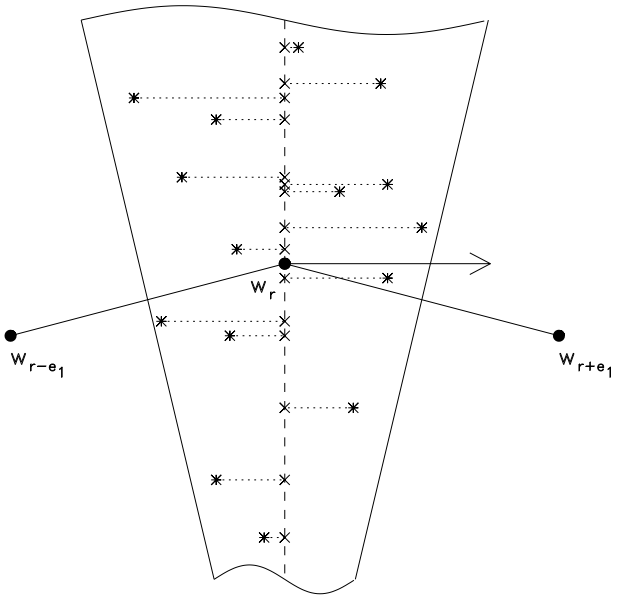

Figure 3.2: Illustration of the criterion for determining the correct growth direction. Consider the center neuron with receptive field center position $\mathbf{w_r}$ in a hypothetical one-dimensional chain of neurons. From the receptive-field-center positions $\mathbf{w_{r+e_1}}$ and $\mathbf{w_{r-e_1}}$ of its output space neighbors a local input space direction can be estimated (large arrow) which corresponds to output space direction $\mathbf{e_1}$. The stimuli (stars) within the Voronoi cell of neuron $\mathbf{r}$ can now be decomposed into a parallel and a perpendicular part relative to this local direction. The average of the relative size of the resp. decomposition amplitudes then determines whether the output space is extended along $\mathbf{e_1}$ or whether a new dimension is added. (Here, a rough visual estimate suggests to add a new dimension).

$\mathbf{v} \in V \subseteq \Re^{d_V}$, which are mapped onto node $\mathbf{r}$. When reconstructing $\mathbf{v}$ from neuron $\mathbf{r}$, an error $\mathbf{v} - \mathbf{w_r}$ remains. This reconstruction error can be decomposed along the different directions, which result from projecting back the output space grid into the input space (Fig. 3.2),

$$\begin{aligned}\mathbf{v} - \mathbf{w_r} &= a_1(\mathbf{v}) \frac{\mathbf{w_{r+e_1}} - \mathbf{w_{r-e_1}}}{\left\| \mathbf{w_{r+e_1}} - \mathbf{w_{r-e_1}} \right\|} + a_2(\mathbf{v}) \frac{\mathbf{w_{r+e_2}} - \mathbf{w_{r-e_2}}}{\left\| \mathbf{w_{r+e_2}} - \mathbf{w_{r-e_2}} \right\|} + \ldots + \mathbf{v}' \\ &= \sum_{i=1}^{d} a_i(\mathbf{v}) \frac{\mathbf{w_{r+e_i}} - \mathbf{w_{r-e_i}}}{\left\| \mathbf{w_{r+e_i}} - \mathbf{w_{r-e_i}} \right\|} + \mathbf{v}'. \end{aligned} \tag{3.1}$$

The criterion for the growth algorithm now is to add nodes in that direction which has on average the largest error amplitudes a_i. To cast this idea into mathematical form, a few

more minor considerations have to be made. First, we have to evaluate the residual errors $\mathbf{v}'$ in order to obtain an error amplitude $b(\mathbf{v})$ for growth into a new dimension. Simply using $b = \|\mathbf{v}'\|$ is not sufficient, because in a high-dimensional input space, with noise in all irrelevant directions, the curse of high dimensions would yield large values for b, suggesting to add more and more dimensions, even though this might not be warranted by the data. Therefore, we instead first collect all residual errors $\mathbf{v}'$ for all the stimuli in a particular Voronoi cell ($\mathbf{v} \in \Omega_{\mathbf{r}}$), and compute the direction of largest variability $\mathbf{e}_{PCA}$ for these (i.e. their first principal component). Projection of $\mathbf{v}'$ onto $\mathbf{e}_{PCA}$ then yields $b(\mathbf{v})$. So we further decompose $\mathbf{v}'$ according to

$$\mathbf{v}' = b(\mathbf{v})\mathbf{e}_{PCA} + \mathbf{v}''. \tag{3.2}$$

The last paragraph made clear, that the computation of $a_i(\mathbf{v})$ and $b(\mathbf{v})$ is performed following two completely independent tracks. Two different variables a and b were used for this reason. However, from now on, $a_i(\mathbf{v})$ and $b(\mathbf{v})$ will be treated on equal footing, and, therefore, we now simplify the notation by defining

$$a_{d+1}(\mathbf{v}) = b(\mathbf{v}). \tag{3.3}$$

Since the dimensionality d of the output grid can increase during the growth process, this definition is valid only during the calculations for one growth step. For each stimulus, we now normalize the deviation amplitudes, and then average over all the stimuli. This leads to averaged amplitudes

$$\tilde{a}_i = \sum_{\mathbf{v}} \frac{\mid a_i(\mathbf{v}) \mid}{\sqrt{\sum_{j=1}^{d+1} a_j^2(\mathbf{v})}}, \quad i = 1, ..., d+1 \tag{3.4}$$

Before we can determine the direction j in which to grow, a last argument has to be taken into account. One could grow into that direction, the amplitude $\tilde{a}_i$ of which is largest. On the other hand, if one more layer in the output space is used to span a particular direction i of the input data manifold, the deviations into this direction will decrease roughly by a factor of $n_i/(n_i+1)$. So one could also base the decision of the expected deviation amplitudes $\tilde{a}_i \cdot n_1/(n_i+1)$ after the growth. Here we propose to balance between the two options and to pick the growth direction j according to

$$j = \text{argmax}_i \left(\sqrt{\frac{n_i}{n_i+1}} \tilde{a}_i \right), \quad i = 1, ..., d, d+1 \tag{3.5}$$

Once a direction in which to grow has been determined, new nodes have to be initialized. Obviously, it is advantageous to use as much information about the data set as the current

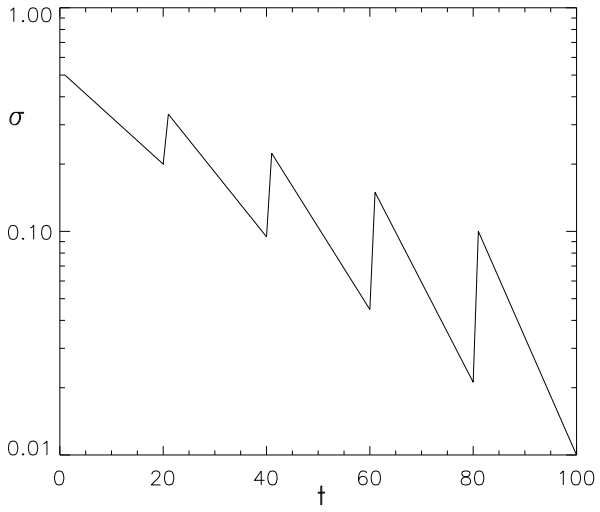

Figure 3.3: Time course of the reduction of the neighborhood width σ during learning. In this illustration, we have five learning phases, with 20 time steps each. σ is initialized at $\sigma_{ii} = 0.5$ and decreases during the first learning phase to $\sigma_{fi} = 0.2$. The σ-values at the beginning of the learning phases are successively reduced to $\sigma_{if} = 0.1$, the final values to $\sigma_{ff} = 0.01$.

map has already acquired. If the grid had an extension $n_j > 1$ into the growth direction, one can add nodes at the layer $n_j/2$ of the output space. The new nodes are then initialized according to

$$\mathbf{w}_{\nu_1,\nu_2,..,n_j/2+1,...\nu_d} = \frac{1}{2}(\mathbf{w}_{\nu_1,\nu_2,..,n_j/2,...\nu_d} + \mathbf{w}_{\nu_1,\nu_2,..,n_j/2+2,...\nu_d}) \tag{3.6}$$

In case of a new dimension, one can utilize the previous information by growing into the direction of the respective principal components.

After each growth step, a new learning phase has to take place, in order to readjust the map. Since the map structure has changed, a possibility for misordering exists. Therefore, the learning phase has to start with a large value of σ again. However, as in regular SOFM-learning, σ ought to have small values towards the end of a training phase in order to resolve details of the data distribution. These two somewhat opposing considerations require that σ be expanded and cooled in a saw-tooth like fashion, with large values at the beginning of each learning cycle, and small values at the end. In our simulations we used an exponential decrease during each learning phase, on which we superimposed further exponential decreases of the starting and end-values for σ from learning phase to learning phase. A typical such time course, parametrized with σ_{ii}, the initial value for σ in the first

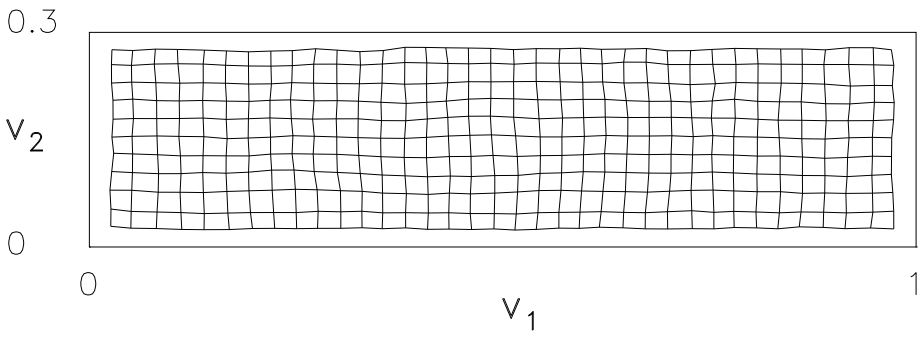

Figure 4.1: Exemplary map for example 1, the map from a two-dimensional input space ($0 \leq v_1 \leq 1$, $0 \leq v_2 \leq 0.3$) onto a map self-organized by the GSOM-algorithm. The final map has an extension of $36 \times 11 = 396$ nodes.

learning phase, σ_{fi}, the final value in the first learning phase, σ_{if}, the initial value of the last learning phase, and σ_{ff}, the final value in the last learning phase, is depicted in Fig. 3.3. We did not notice a strong impact of the reduction scheme on the structure of the resulting maps. The only mentionable effect is that the size of the σ_{ff} determines to what extent the map can follow all wrinkles of the data distribution. σ_{ff} therefore has impact on the neighborhood preservation properties of particular maps. This effect is not unique for our GSOM-algorithm, but occurs in regular SOFMs as well.

4 Examples

We now turn to some exemplary applications of the new algorithm. What outcomes can be expected to occur? Even though the rules for adding new dimensions, or for choosing not to do so, are heuristically plausible, it remains to be seen, whether the algorithm actually increases the dimensionality first, and then, later, stops to add new dimensions. This test is particularly interesting with high-dimensional real world data sets, where the curse of high dimension could fool the algorithm into adding evermore output space dimensions. Secondly, the algorithm should lead to an optimally adapted output space, and, hence, the resulting map should preserve neighborhoods well. However, the degree of neighborhood preservation was neither measured nor enforced during the adaptation. Therefore, this property of the resulting maps is an independent feature we will have to separately check for (see section 5).

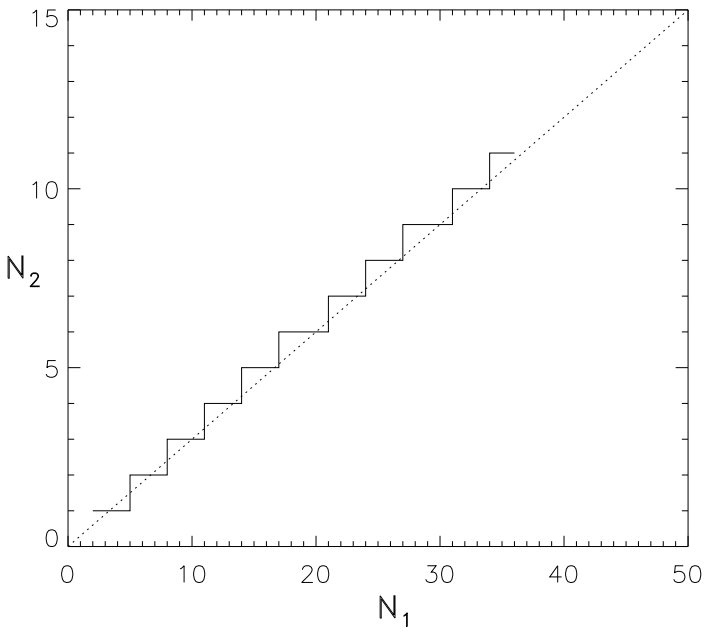

Figure 4.2: Sequence of map states during the growth process for example 1 (solid line), parametrized by their node number N_1, N_2. The process started at $N_1 = 2, N_2 = 1$, increased nodes in the $\mathbf{e_1}$-direction until $N_1 = 5, N_2 = 1$, added a new dimension ($N_1 = 5, N_2 = 2$), etc. During the whole process, the relation of node number did not deviate much from 0.3, the relation specified by the input space extensions.

4.1 Example 1: Map of a rectangular input space

As a pedagogical first example, we consider evenly distributed data points in a rectangular input space $V \subset \Re^2$, $V = [0, 1.0] \times [0, 0.3]$. These are to be projected by a map self-organized with the GSOM-algorithm. A typical result for one simulation (out of many) is shown in Fig 4.1. The maximum number of nodes $N_{max} = 400$ was fixed before learning, other parameters were $\sigma_{ii} = 0.5$, $\sigma_{fi} = 0.1$, $\sigma_{if} = 0.1$, $\sigma_{ff} = 0.02$, $\epsilon_i = 0.9$, $\epsilon_f = 0.01$. The GSOM-algorithm adjusted the output space at a dimension of $d = 2$, with 36×11 ($= 396$) nodes along the two directions, the optimal result possible. At intermediate steps during the growth phase, a ratio of $n_x : n_y \approx 10 : 3$ between the node numbers along both directions was approximately kept constant (Fig. 4.2). We observed analogous results for similar input space geometries (3d instead of 2d, other aspect ratios). In all cases, correct output space topologies were grown, with no dependence of the results on a fine-tuning of particular parameters (like, e.g., the σ-reduction parameters).

4.2 Example 2: Map of image data

As next example, we used feature vectors which were derived from an image data compression application. To obtain the data, the well-known image of a girl with a hat, given as a 512×512 pixel image with 8 bits per pixel was divided into non-overlapping sections of 8×8 pixels. The resulting 64-dimensional data points were mapped by a $64 \times 16 \times 64$ multilayer-perceptron onto themselves. After training, the multilayer-perceptron was used a data compression device, with the actual data point being replaced by the activity pattern of the hidden units (8 bit resolution) (For previous investigations of this example, see [22, 23]). This compressed data, i.e. 4096 feature vectors of dimension $d_V = 16$, were then used as input data for simulations of the GSOM-algorithm ($N_{max} = 256$, $\sigma_{ii} = 0.5$, $\sigma_{fi} = 0.1$, $\sigma_{if} = 0.1$, $\sigma_{ff} = 0.02$, $\epsilon_i = 0.9$, $\epsilon_f = 0.01$ as in the previous example) This led to three maps with output space topologies of $12 \times 5 \times 4$, $13 \times 9 \times 2$ and $12 \times 9 \times 2$ nodes, respectively. These results demonstrate that the GSOM indeed recruits new dimensions, yet stops to do so at an dimensionality way below d_V ($d = 3$ as opposed to $d_V = 16$ in this case). Repeated application of the algorithm does not lead to exactly identical map structures due to the stochasticity of the initialization and the stimulus sequence. Yet, the overall dimensionality of the three maps coincides and the length ratios along the different directions are approximately equal, so the results are reproducible. The issue of neighborhood preservation in the resulting maps is adressed in the next section.

4.3 Example 3: Map of DPI speech data

Our last data set is based on speech data. It is the same data set we used in previous publications [6, 8], and has been acquired in the III. Physikalisches Insitut, Universität Göttingen. Preprocessing of the data has been described elsewhere [24]. Here it suffice to say, that the data come as 2013 feature vectors ($d_V = 19$), representing the ten (German) digits, spoken ten times by one speaker, with the excess 0-vectors at the beginning and end of words taken out of the data (this constitutes the difference to the number of 4500 data vectors, used in [6]). Application of the GSOM-algorithm lead to maps with $2 \times 12 \times 3 \times 3$, $3 \times 9 \times 3 \times 3$ and $9 \times 3 \times 3 \times 3$ nodes, for a limit of $N_{max} = 256$ nodes. Again, the resulting output space dimension d is adjusted to a prima facie plausible value. The resulting output space topologies are quite similar, underlining that the GSOM-algorithm delivers reproducible results.

5 Neighborhood preservation in self-organizing maps

Two measures have been developed to quantify the degree of neighborhood preservation in neural maps, the topographic product P and the topographic function Φ. We now use both these measures to assess the quality of the GSOM-generated maps and of some fixed output space SOFMs, which we generated for comparison.

5.1 Topographic product

In both approaches, the central idea is to not compare absolute distances, but instead to evaluate the ordering of neighbors in input and output space. For the topographic product, all neighborhood orders are considered. For each node (which is indexed by its position $\mathbf{r}$ in output space) the sequences $\mathbf{n}_k^A(\mathbf{r})$ and $\mathbf{n}_k^V(\mathbf{r})$ have to be determined, where $\mathbf{n}_k^A(\mathbf{r})$ denotes the k-th neighbor of $\mathbf{r}$, with distance measured in the output space, and $\mathbf{n}_k^V(\mathbf{r})$ denotes the k-th neighbor of $\mathbf{r}$, with distances evaluated in the input space between $\mathbf{w}_\mathbf{r}$ and $\mathbf{w}_{\mathbf{n}_k^V(\mathbf{r})}$. In case of degeneracies, as occur in hypercubical output spaces, a random ordering among the degenerate nodes can be used. From these sequences, an intermediate quantity

$$P_3(\mathbf{r},k) = \left(\Pi_{l=1}^{k} \frac{d^V\left(\mathbf{w}_\mathbf{r}, \mathbf{w}_{\mathbf{n}_l^A(\mathbf{r})}\right)}{d^V\left(\mathbf{w}_\mathbf{r}, \mathbf{w}_{\mathbf{n}_l^V(\mathbf{r})}\right)} \cdot \frac{d^A\left(\mathbf{r}, \mathbf{n}_l^A(\mathbf{r})\right)}{d^A\left(\mathbf{r}, \mathbf{n}_l^V(\mathbf{r})\right)} \right)^{\frac{1}{2k}} \tag{5.1}$$

is defined. Further averaging over neighborhood orders k and nodes $\mathbf{r}$ finally leads to the topographic product

$$P = \frac{1}{N(N-1)} \sum_{\mathbf{r}} \sum_{k=1}^{N-1} \log\left(P_3(\mathbf{r},k)\right). \tag{5.2}$$

P can take on positive or negative values, which have to be interpreted as follows:

$P < 0$: Output space too low-dimensional, neighborhood relations are violated.

$P \approx 0$: Output space approximately matches topology of input data, neighborhood relations preserved.

$P > 0$: Output space too high-dimensional, neighborhood relations are violated.

In conclusion, the topographic product P is a comparatively simple to implement diagnostic method, it is derived from heuristic considerations, and it can be evaluated on the basis of the map parameters only, with no knowledge of the underlying data distribution necessary.

5.2 Topographic function

Recently, a second scheme for the quantification of neighborhoods has been proposed, the topographic function Φ, which is grounded in mathematical topology. Apart from its more stringent mathematical foundation, the topographic function evaluates input space neighborhood relations on the basis of the actual data, instead of using the map pointer positions $\mathbf{w}_{\mathbf{r}}$. In this way it avoids classifying strangely curved maps as violating neighborhoods, if these folds are induced by strangely curved data manifolds. In order to keep this paper self-contained, we provide the reader with a brief derivation of Φ in this subsection. A more detailed account can be found elsewhere [7, 8].

For the evaluation of Φ, the distance measure in the input space between $\mathbf{w}_{\mathbf{r}}$, $\mathbf{w}_{\mathbf{r}'}$ is the minimal path length $d^{\mathcal{D}_V}(\mathbf{w}_{\mathbf{r}}, \mathbf{w}_{\mathbf{r}'})$ between $\mathbf{w}_{\mathbf{r}}$, $\mathbf{w}_{\mathbf{r}'}$ in the induced Delaunay-graph $\mathcal{D}_V$ of the $\mathbf{w}_{\mathbf{r}}$. $\mathcal{D}_V$ in turn corresponds in an one-to-one way to the Voronoi tessellation of V by the masked Voronoi cells $\Omega_{\mathbf{r}}$. The vectors $\mathbf{w}_{\mathbf{r}}$ and $\mathbf{w}_{\mathbf{r}'}$ are connected if and only if there masked Voronoi cells are adjacent, i.e. $\Omega_{\mathbf{r}} \cap \Omega_{\mathbf{r}} \neq \emptyset$. In this way the structure of the input space V is explicitly taken into account. Neighborhood relations between nodes $\mathbf{r}, \mathbf{r}'$ in the output space are given by the lattice structure of A, determined by the Euclidean distance and denoted as the strong neighborhood in A. Furthermore, to compensate for degeneracy effects in a rectangular lattice, it is necessary to define a second neighborhood between the nodes $\mathbf{r}, \mathbf{r}' \in A$ in addition to the above strong neighborhood. This so-called weak neighborhood is based on the maximum norm, $\left\|\mathbf{r} - \mathbf{r}'\right\|_{\max} = \max_{k=1}^{d} \left|\mathbf{r}_k - \mathbf{r}'_k\right|$. Now we say that the map $\Psi_{V \to A}$ is neighborhood preserving if $\mathbf{w}_{\mathbf{r}}$, $\mathbf{w}_{\mathbf{r}'}$ which are connected in the induced Delaunay-graph $\mathcal{D}_V$ are projected onto nodes $\mathbf{r}, \mathbf{r}'$ which are weakly neighbored in A. On the other hand, the map $\Psi_{A \to V}$ is neighborhood preserving if nodes $\mathbf{r}$, $\mathbf{r}'$ which are strongly neighbored in A have receptive field centers $\mathbf{w}_{\mathbf{r}}$, $\mathbf{w}_{\mathbf{r}'}$ which are connected in $\mathcal{D}_V$. Using these concepts we define for each node $\mathbf{r} \in A$

$$\begin{aligned} f_{\mathbf{r}}(k) &\stackrel{def}{=} \#\left\{\mathbf{r}' \mid \left\|\mathbf{r} - \mathbf{r}'\right\|_{\max} > k \;;\; d^{\mathcal{D}_V}\left(\mathbf{r}, \mathbf{r}'\right) = 1\right\}, \\ f_{\mathbf{r}}(-k) &\stackrel{def}{=} \#\left\{\mathbf{r}' \mid \left\|\mathbf{r} - \mathbf{r}'\right\| = 1 \;;\; d^{\mathcal{D}_V}\left(\mathbf{r}, \mathbf{r}'\right) > k\right\}, \end{aligned} \tag{5.3}$$

with $k = 1, \ldots, N-1$. Here $\#\{\cdot\}$ denotes the cardinality of a set. The topographic function Φ_A^V then is defined by

$$\Phi_A^V(k) \stackrel{def}{=} \begin{cases} \frac{1}{N} \sum_{\mathbf{r}' \in A} f_{\mathbf{r}'}(k) & k > 0 \\ \Phi_A^V(1) + \Phi_A^V(-1) & k = 0 \\ \frac{1}{N} \sum_{\mathbf{r}' \in A} f_{\mathbf{r}'}(k) & k < 0 \end{cases} . \tag{5.4}$$

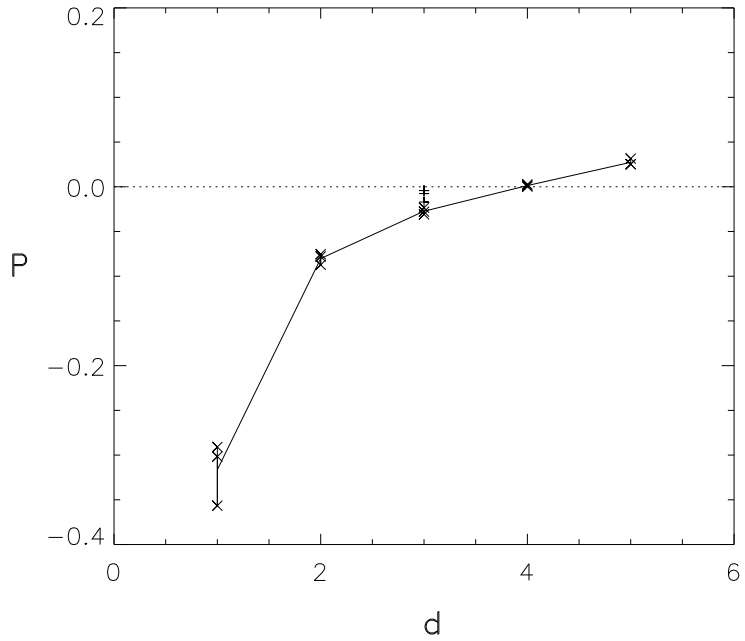

Figure 5.1: Values of the topographic product P for various maps with fixed output spaces (crosses), dimensions $d = 1 - 5$, trained with the image data of example 2. The solid line connects the mean values, computed from three maps for each value of d. The course of $P = P(d)$ for the fixed output space maps suggests a data dimension of between 3 and 5, with $P \approx 0$ at $d = 4$, The GSOM-algorithm leads to maps, which roughly coincide with this range of overall dimensionalities, and which yield neighborhood preservation values P (+-signs) comparable with those of the best fixed output space maps.

This yields $\Phi_A^V \equiv 0$ and, in particular, $\Phi_A^V(0) = 0$, if both $\Psi_{V\to A}$ and $\Psi_{A\to V}$ are neighborhood preserving. The largest $k^+ > 0$ with $\Phi_A^V(k^+) \neq 0$ yields the range of the largest fold if the effective dimension of the data manifold V is larger than the lattice dimension d. The smallest $k^- < 0$ with $\Phi_A^V(k) \neq 0$ holds yields the range of the largest fold if the effective dimension of the data manifold V is smaller then d. Small values of k^+ and k^- indicate that there are only local conflicts, whereas large values indicate a global dimensional conflict. To compare the topographic functions of different lattices it is useful to normalize the k-values, such that $k \in [-1, 1]$.

Calculating Φ_A^V requires to determine the induced Delaunay-graph $\mathcal{D}_V$. A way to determine $\mathcal{D}_V$ has been proposed in [10]. Let $\mathbf{C}$ be a connectivity matrix which determines connections between units $\mathbf{r}, \mathbf{r}' \in A$ (in addition to the connectivity matrix defined by the fixed lattice structure). Initially, the elements $\mathbf{C}_{\mathbf{r}\mathbf{r}'} \in \{0, 1\}$ of $\mathbf{C}$ are set to zero. It can be shown that simply by sequentially presenting input vectors $\mathbf{v} \in V$ and each time connecting those two

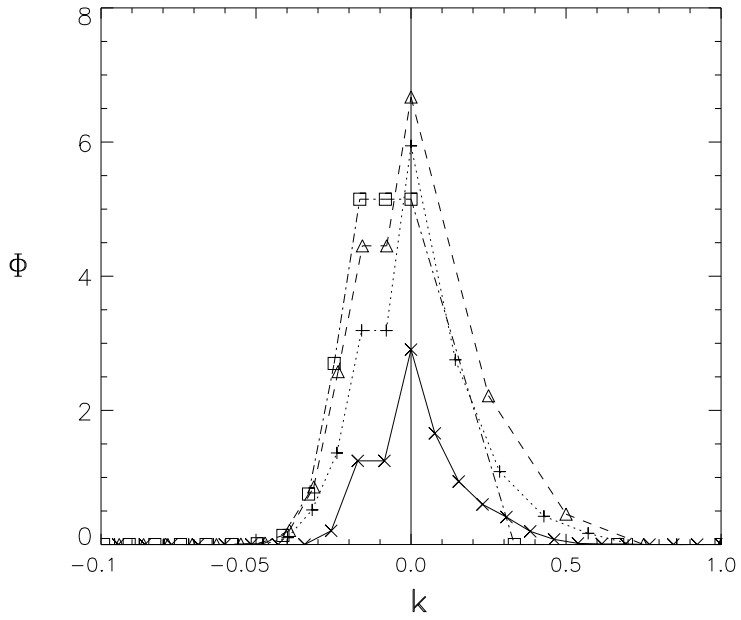

Figure 5.2: Topographic function $\Phi(k)$ for three fixed output space maps and one GSOM-map of example 2 (+-signs, dotted line: $d = 3$, triangles, dashed line: $d = 4$, squares, dash-dot line: $d = 5$, crosses, solid line: $13 \times 9 \times 2$-GSOM map). Observe the stretched axis for $k < 0$ to obtain better resolution in this regime. The GSOM-map achieves by and large lower values for $\Phi(k)$ than the fixed output space maps, indicating fewer violations of neighborhoods.

units $\mathbf{r}^*$, $\mathbf{r}'^*$ (setting $\mathbf{C}_{\mathbf{r}^*\mathbf{r}'^*} = 1$) the pointers $\mathbf{w}_{\mathbf{r}^*}$, $\mathbf{w}_{\mathbf{r}'^*}$ of which are closest and second closest to $\mathbf{v}$, the connectivity matrix $\mathbf{C}$ converges to

$$\lim_{t\to\infty} \mathbf{C}_{\mathbf{r}\mathbf{r}'} = 1 \quad \Leftrightarrow \quad \Omega_{\mathbf{r}} \cap \Omega_{\mathbf{r}'} \neq \emptyset \tag{5.5}$$

[10]. After a sufficient number of input vectors $\mathbf{v}$ have been presented, the connectivity matrix $\mathbf{C}$ connects nodes and only nodes $\mathbf{r}$, $\mathbf{r}'$ the receptive fields $\Omega_{\mathbf{r}}$, $\Omega_{\mathbf{r}'}$ of which are adjacent and, hence, defines the induced Delaunay-graph $\mathcal{D}_V$.

5.3 Results on neighborhood preservation for the examples

Using P and Φ, we now analyze the GSOM-maps of examples 2 and 3 (the map in the first example is obviously perfectly neighborhood preserving). In order to be able to gauge the values for P and Φ obtained for the GSOM-maps, we also trained fixed output space maps of different dimensionalities for both examples. The resulting values for P and Φ are depicted

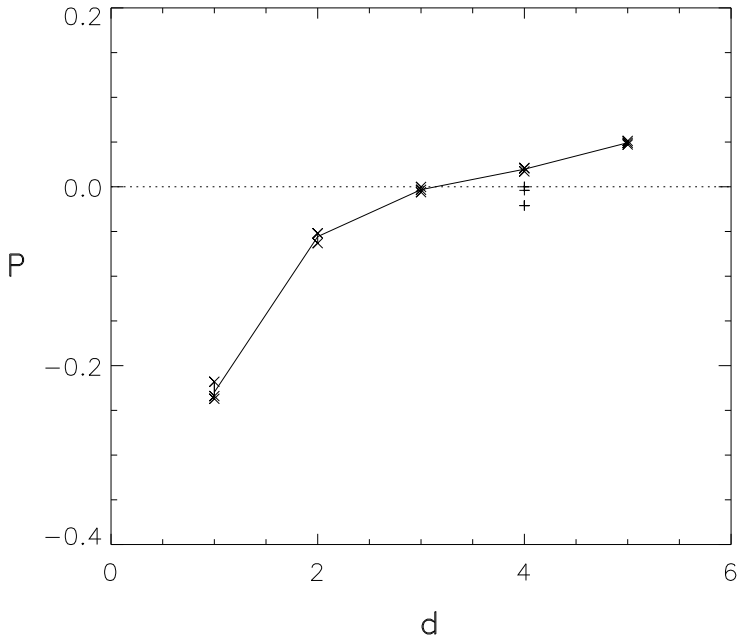

Figure 5.3: Analog. to Fig. 5.1, but for maps trained with the speech data of example 3.

in Figs. 5.1,5.2 and 5.3,5.4, respectively. The GSOM-maps turn out to have neighborhood preservation values which are about as good as or even better than the best values obtained by trial and error for fixed output space maps. For the speech data (example 3), the GSOM-maps yield the best value of P obtained so far for any map ($P = 3 \times 10^{-5}$). However, this result gives rise to a word of caution about the precision of neighborhood quantification. Since the maps are generated by self-organization processes which try to optimize the map within the limits of any output space given, many output spaces of similar shape can be appropriate to accommodate a perfectly neighborhood preserving map. For each of these, P and Φ will have rather low, but nonvanishing and slightly different (absolute) numerical values. Therefore, with real world data one should neither overinterprete low values like 3×10^{-5} nor expect the P and Φ measures to be able to identify an unique optimal output space topology. Instead, these measures, as well as the GSOM-algorithm, help identify a range of approximately correct output space topologies, and help rule out severely mismatching spaces which would result in strongly folded maps.

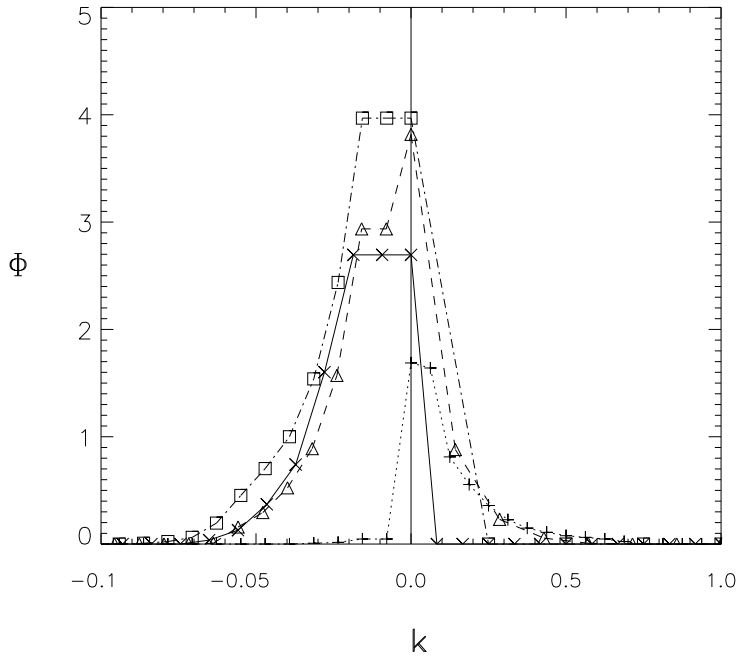

Figure 5.4: Analogous to Fig. 5.2, but for the speech data maps of example 3 (+-signs, dotted line: $d = 2$, triangles, dashed line: $d = 3$, squares, dash-dot line: $d = 4$, crosses, solid line: $2 \times 12 \times 3 \times 3$-GSOM map). Even though the GSOM-map does not obtain the lowest value for $\Phi(0)$, the vanishing of $\Phi(k > 0)$ indicates that no large scale folds of the output space into the input space occur in this map.

6 Discussion

The distinguishing feature of neural map-algorithms like Kohonen's SOFM-algorithm as compared to other vector quantizers, or to competitive learning schemes, is the property of neighborhood preservation. In technical applications of the SOFM, this property of neighborhood preservation is valuable only, if it either improves the convergence properties of the learning algorithm (see, e.g., [25]), as compared to non-neighborhood preserving algorithms, or if it is actually utilized by the overall system, of which the neural map is a part. Examples for this latter case include data representation schemes, where high-dimensional input data is either represented by a position in a two-dimensional region (which is easy to visually monitor), or where the neuron index of the winning node is transformed into a gray-value or color index in a systematic way. The advantage of neighborhood preservation in false-gray-value representation of color quantized images was recently highlighted by Dekker

[26]. The role of neighborhood preservation in the reconstruction of principal manifolds in the data by interpolation between the best-matching node and its neighbors has already been mentioned in the introduction. This argument does not only apply to sophisticated interpolation schemes like the PSOM [16, 17], but also to a simple linear interpolation using the best-matching node plus the second-best-matching among its nearest neighbors. A third argument for the advantages of neighborhood preservation in vector quantization was pointed out by Luttrell, who investigated the impact of noise on the reconstruction error during the transmission of compressed data [27]. As a final example for the role of neighborhood preservation in neural maps, we mention speech recognition systems. Here the neural map provides a dimension reducing preprocessing stage, which is followed by a trajectory recognizing algorithm, like a dynamic time warping algorithm [28, 29, 30]. Different version of a word follow similar trajectories in the high-dimensional feature vector space, which in turn are mapped onto similar trajectories in the low-dimensional map output space, where they can be recognized.

After having seen why neighborhood preservation in neural maps can be advantageous, we now turn to a discussion of the different ways to ensure it. We pointed out in the introduction, that two components of the self-organization process have to match: the output space topology has to roughly match that of the data distribution, and the map self-organization process itself has to be correctly parametrized to avoid traps. In the present paper we are concerned with the former problem, and presented the GSOM-algorithm, which adapts the output space topology to the input data, but in a constrained way. The output space is forced to maintain the shape of a generalized hypercube. This is in contrast to other approaches to this problem [10, 11, 12, 13], which allowed less constrained or even completely unconstrained output space connectivities. Obviously, with less or no constraints these can possibly achieve an even better match than a GSOM. However, this comes at the expense of a more complicated connectivity structure. Depending on the task being performed on the output space data, such complicated structure can be prohibitive. A transformation of the data in color code, for example, may no more be possible. With regard to the speech recognition application, a hypercubical output space allows to use Euclidean distances for a subsequent classification of word trajectories, whereas a general symmetric graph would require some length distance on the graph. Which one will deliver better results, is an empirical question to be investigated in a different line of research. Similarly, it remains to be seen (and will depend on the data set), whether a reconstruction of data using interpolation schemes in neural maps performs better with rather good neighborhood preservation of GSOM-map plus an advanced interpolation technique like a PSOM, or with, say, a TRN-map [10], which delivers unconstrained and, hence, optimal neighborhood

preservation but allows for linear interpolation only. In summary the present algorithm delivers maps with output spaces which are adapted to deliver a rather good neighborhood preservation, but which are at the same time constrained to meet the demands of subsequent processing steps.

7 Acknowledgments

This work has been supported by the Deutsche Forschungsgemeinschaft (Sonderforschungsbereich 185 "Nichtlineare Dynamik", TP E3).

References

[1] C. v. d. Malsburg, Self-organization of orientation sensitive cells in the striate cortex, *Kybernetik* **14**, 85-100 (1973).

[2] D.J. Willshaw, C. v. d. Malsburg, How patterned neural connections can be set up by self-organization, Proc. R. Soc. Lond. **B 194**, 431-445 (1976).

[3] T. Kohonen, Self-organized formation of topologically correct feature maps, *Biological Cybernetics* **44**, 59-69 (1982).

[4] T. Kohonen, The Self-Organizing Map, Springer (1995).

[5] H. Ritter, T. Martinetz and K. Schulten, Neural Computation and Self-Organizing Maps, Addison Wesley, Reading, Mass. (1992).

[6] H.-U. Bauer and K. Pawelzik, Quantifying the Neighborhood Preservation of Self-Organizing Feature Maps, *IEEE Transactions on Neural Networks* **3(4)**, 570-579, (1992).

[7] Th. Villmann, R. Der, M. Herrmann and Th. Martinetz, Topology Preservation in SOFMs : General Definition and Efficient Measurement, *Informatik aktuell - Fuzzy-Logik*. Ed.: B. Reusch, Springer, 159-166 , (1994).

[8] Th. Villmann, R. Der, M. Herrmann and Th. Martinetz, Topology Preservation in SOFMs : Exact Definition and Measurement. sub. to *IEEE Transactions on Neural Networks* (1994).

[9] H.-U. Bauer, K. Pawelzik and T. Geisel, A Topographic Product for the Optimization of Self-Organizing Feature Maps, in: *Neural Information Processing Systems* (Proc. of NIPS 4), ed. J. Moody et al., Morgan Kauffman, 1141-1147 (1992).

[10] T. Martinetz and K. Schulten, Topology Representing Networks, *Neural Networks* **7**, 507-522 (1994).

[11] B. Fritzke, Growing Cell Structures—a Self-Organizing Network for Supervised and Unsupervised Learning, *Neural Networks* **7**, 114-1460 (1994).

[12] A. Hämäläinen, Using Genetic Algorithm in Self-Organizing Map Design, in Proc. ICANNGA 95, Alès, France, eds. D.W. Pearson, N.C. Steele, R.F. Albrecht, 364-367 (1995).

[13] S. Jockusch and H. Ritter, Self-Organizing Maps: Local Competition and Evolutionary Optimization, *Neural Networks* **7**, 1229-1239 (1994).

[14] M. Groß and F. Seibert, Neural Network Image Analysis for Environmental Protection. Zentrum für graphische Datenverarbeitung, Darmstadt, Germany, preprint (1993).

[15] P. Weierich and M. von Rosenberg, Unsupervised Detection of Driving States with Hierarchical Self-Organizing Maps, in Proc. of the ICANN93, eds. S. Gielen, B. Kappen, Springer (London), 246-249 (1993).

[16] H. Ritter, Parametrized Self-Organizing Maps, in: Proc. of the ICANN'93, eds. S. Gielen, B. Kappen, Springer (London), 568-575 (1993).

[17] H. Ritter, Parametrized Self-Organizing Maps for Vision Learning Tasks, in: Proc. of ICANN 94, eds. M. Marinaro, P.G. Morasso, Springer (London), 803-810 (1994).

[18] Y. Linde, A. Buzo and R. Gray, An Algorithm for Vector Quantizer Design, *IEEE Transactions on Communications* **COM-28**, 84-95, (1980).

[19] F. Wolf, H.-U. Bauer and T. Geisel, Formation of Field Discontinuities and Islands in Visual Cortical Maps, Biol. Cyb. **70**, 525-531 (1994).

[20] E. Erwin, K. Obermayer, K. Schulten, Self-Organizing Maps: Ordering, Convergence Properties and Energy Functions, *Biological Cybernetics* **67**, 47-55 (1992).

[21] H. Ritter and K. Schulten, Convergence Properties of Kohonen's Topology Conserving Maps: Fluctuations, Stability and Dimension Selection, *Biological Cybernetics* **60**, 59-71 (1988).

[22] N. Sonehara, M. Kawato, S. Miyake, K. Nakane, Proc. IJCNN 89 Washington, IEEE Press, II-35-41 (1989).

[23] B. Fritzke, Vector Quantizing with a Growing and Splitting Elastic Net , in: Proc. of the ICANN93, eds. S. Gielen, B. Kappen, Springer (London), 580-585 (1993).

[24] T. Gramss, H.W. Strube, Recognition of isolated words based on psychoacoustics and neurobiology, *Speech Comm.* **9**, 35-40 (1990).

[25] H. Speckmann, G. Raddatz and W. Rosenstiel, Consideration of Geometrical and Fractal Dimension of Self-Organizing Feature Maps to Get Better Learning Results, in: Proc. of ICANN 94, eds. M. Marinaro, P.G. Morasso, Springer (London), 342-345 (1994).

[26] A. H. Dekker, Kohonen neural networks for optimal color quantization, *Network* **5**, 351-367 (1994).

[27] S.P. Luttrell, Self-Organization: a derivation from first principles of a class of learning algorithms, Proc. IJCNN 89 Washingtonm, IEEE Press, II-495-498 (1989).

[28] J. Kangas, On the Analysis of Pattern Sequences by Self-Organizing Maps, PhD-Thesis, Helsinki University of Technology, (1994).

[29] W.D. Brandt, H. Behme, H.W. Strube, Bildung von Merkmalen zur Spracherkennung mittels phonotopischer Karten, in: *Fortschritte der Akustik-DAGA 91*, DPG GmbH, Bad Honnef, Germany, 1057-60 (1991).

[30] F. Mehler and P. Wilcox, Self-Organizing Maps in Speech Recognition Systems. Proc. of the First Int. Conf. on Appl. Synergetics and Synergetic Engineering (ICASSE'94), Eds. F. G. Böbel and T. Wagner, Erlangen, Germany, 20-26 (1994).