The GSOM–Algorithm for Growing Hypercubical Output Spaces in Self–Organizing Maps

Th. Villmann[†] * H.-U. Bauer[‡*]

[†]Universität Leipzig, Klinik für Psychotherapie und Psychosomatische Medizin
D–04107 Leipzig, Karl–Tauchnitz–Str.25, Germany
[‡]Max-Planck-Institut für Strömungsforschung
D–37073 Göttingen, Bunsenstr. 10, Germany

Abstract

We describe an extension of the SOM algorithm, called the Growing Self–Organizing Map (GSOM), which allows to adapt simultaneously the position of the map pointers into the input space, and the topology of the output space. In the GSOM the output space connectivity is constrained to that of a generalized hypercube, with the overall dimensionality, and the extensions along the individual dimensions being a subject of the adaptation. Results for GSOM–generated projections of synthetic and real–world data sets are discussed.

1 Introduction

Self-organizing maps (SOM) 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 which map data from an (possibly highdimensional) input space $V \subseteq \Re^{d_V}$ onto an output space A. Usually, A is a prespecified d-dimensional rectangular grid, yet other arrangements are also admissible. Associated to the neurons $\mathbf{r} \in A$ are weight vectors $\mathbf{w_r}$ which determine a particular position in V. The mapping $\Psi_{V \to A}$ is realized by a winner take all rule

$$\Psi_{V \to A} : \mathbf{v} \mapsto \mathbf{s} = \underset{\mathbf{r} \in A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{w_r}\| \ . \tag{1.1}$$

To achieve this projection, SOMs adapt the pointer positions $\mathbf{w_r}$ such that the input space is mapped onto the output space in a most faithful way. 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 $\mathbf{v}$,

$$\triangle \mathbf{w_r} = \epsilon h_{\mathbf{rs}} (\mathbf{v} - \mathbf{w_r}) \ . \tag{1.2}$$

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,

$$h_{\mathbf{rs}} = \exp\left(-\frac{\|\mathbf{r} - \mathbf{s}\|^2}{2\sigma^2}\right) \ . \tag{1.3}$$

Obviously, the neighborhood preservation feature depends on the choice of the output space topology and the dimensionality required by the input data which is usually unknown *a priori*.[1] Yet the

*This work has been supported by Deutsche Forschungsgemeinschaft, SFB 185 "Nichtlineare Dynamik", TP E3.

[1]consider, e.g. speech data, which are usually represented in a 10-20 dimensional input space, but fill only a lower-dimensional submanifold

output space A has to be specified prior to learning. Therefore, a problem emerges if neighborhood preservation have to be optimized by the network.[2] However, one can try to adapt not weights $\mathbf{w_r}$, but also the structure of A. In the present paper we describe an algorithm (called Growing SOM – GSOM) which combines the pointer adaptation of the usual SOM with a growth procedure for *hypercubical output spaces*. Depending on the application, these may be easier to handle in further processing than less constrained or unconstrained output space graphs [3, 4, 7]. We here require the output space A to consist of a (rectangular) hypercube but we allow a variable overall dimensionality and variable dimensions along the individual directions in the hypercube. This can be cast in a formal way by writing the output space positions as $\mathbf{r} = (i, j, k, ...), \quad 1 < i < n_1, 1 < j < n_2, ...$with $N = n_1 \times n_2 \times ...$ denotes the overall number of nodes.

2 The GSOM–Algorithm

The GSOM starts from an initial 2-neuron configuration, learns according the regular SOM-algorithm, adds neurons to the output space with respect to a certain criterion to be described below, learns again, adds again, etc., until a prespecified maximum number $N_{\max}$ of neurons is distributed. During this procedure, the output space topology remains to be of the form $n_1 \times n_2 \times ...$, with $n_j = 1$ for $j > d$, where d is the current dimensionality of A. Hence, 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. 1a). 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 among the data points $\mathbf{v}$ which are mapped onto a common neuron according to (1.1) (i.e. among the data points in the masked Voronoi cell $\Omega_\mathbf{r}$ of neuron $\mathbf{r}$). When reconstructing $\mathbf{v}$ from neuron $\mathbf{r}$, an error $\theta = \mathbf{v} - \mathbf{w_r}$ remains:

$$\theta = \sum_{i=1}^{d} a_i(\mathbf{v}) \frac{\mathbf{w_{r+e_i}} - \mathbf{w_{r-e_i}}}{\|\mathbf{w_{r+e_i}} - \mathbf{w_{r-e_i}}\|} + \mathbf{v}' \; , \tag{2.1}$$

decomposed along the different directions, which result from projecting back the output space grid into the input space A (Fig. 1b). Thereby, $\mathbf{e}_i$ denotes the unit vector in direction i of the A.[3]

The criterion for the growth algorithm now is to add nodes in that direction which has on average the largest error amplitudes a_i. First, we have to evaluate the residual errors $\mathbf{v}'$ in order to obtain an error amplitude $a_{d+1}(\mathbf{v})$ for growth into a new dimension. Simply using $a_{d+1} = \|\mathbf{v}'\|$ 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 a_{d+1}, suggesting to add more and more dimensions, even though this might not be warranted by the data. Therefore, we instead collect the residual errors $\mathbf{v}'$ for all the stimuli in a particular Voronoi cell ($\mathbf{v} \in \Omega_\mathbf{r}$), and compute their first principal component $\mathbf{v}_{PCA}$. Projection of $\mathbf{v}'$ onto the direction of $\mathbf{v}_{PCA}$ then yields $a_{d+1}(\mathbf{v})$,

$$\mathbf{v}' = a_{d+1}(\mathbf{v}) \frac{\mathbf{v}_{PCA}}{||\mathbf{v}_{PCA}||} + \mathbf{v}''.$$

We now normalize the deviation amplitudes for each stimulus individually and then compute the averaged amplitudes

$$\tilde{a}_i = \sum_{\mathbf{v}} \frac{\mid a_i(\mathbf{v}) \mid}{\sqrt{\sum_{j=1}^{d+1} a_j^2(\mathbf{v})}}, \quad i = 1, ..., d+1 \tag{2.2}$$

[2] 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.

[3] At the border of the output space grid, where not two, but just one neighboring neuron is available, we use $\frac{\mathbf{w_r} - \mathbf{w_{r-e_i}}}{||\mathbf{w_r} - \mathbf{w_{r-e_i}}||}$ or $\frac{\mathbf{w_{r+e_i}} - \mathbf{w_r}}{||\mathbf{w_{r-e_i}} - \mathbf{w_r}||}$ to compute the backprojection of the output space direction $\mathbf{e}_i$ into the input space.

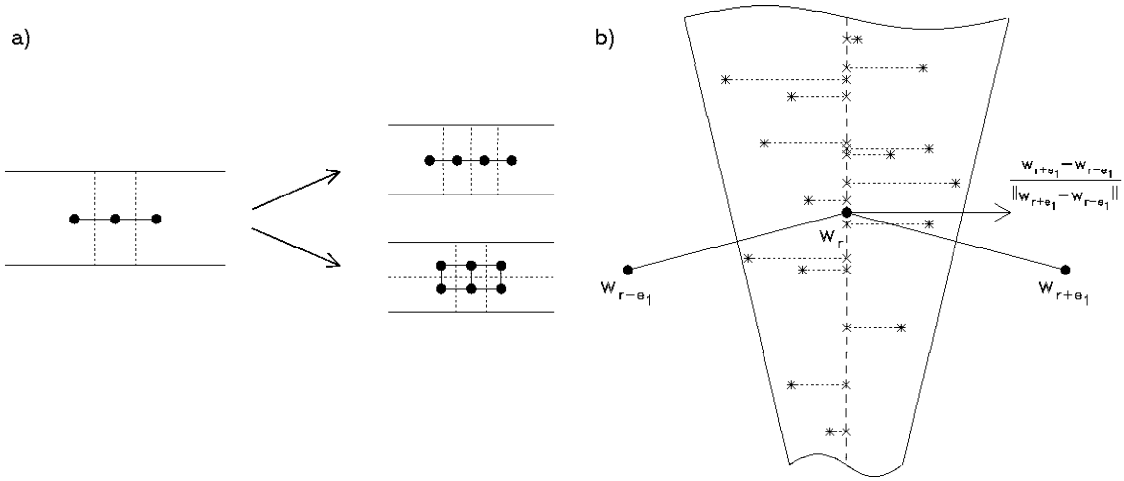

Figure 1: a) (left) Illustration of the basic decision to be made during the growth procedure: an output space lattice can either grow in an existing direction or it can extend into a new direction. b) (right) Illustration of the criterion for determining the correct growth direction. Consider the center 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 the local input space direction $\frac{\mathbf{w_{r+e_1}}-\mathbf{w_{r-e_1}}}{\|\mathbf{w_{r+e_1}}-\mathbf{w_{r-e_1}}\|}$ 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).

One might 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 $\tilde{a}_i \cdot N_i/(N_i+1)$ after the growth. Here we propose to balance between the two options and to pick the growth direction j according to

$$j = \text{argmax}_i \left(\sqrt{\frac{n_i}{n_i+1}} \tilde{a}_i \right), \quad i = 1, ..., d, d+1 \tag{2.3}$$

Once a direction in which to grow has been determined, new nodes have to be initialized. 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

$$\mathbf{w}_{\nu_1,\nu_2,..,N_j/2+1,...\nu_d} = \frac{1}{2}(\mathbf{w}_{\nu_1,\nu_2,..,N_j/2,...\nu_d} + \mathbf{w}_{\nu_1,\nu_2,..,N_j/2+2,...\nu_d}) \tag{2.4}$$

In case of a new dimension, one can utilize the previous information by growing into the direction of the respective principal components.[4]

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

[4]We should note at this point that the dimensionality d of the output space grid can increase during the growth process; consequently the decomposition of $\mathbf{v}$ into the different directions can involve an increasing number of terms.

phase has to start with a large value of σ again. However, as in regular SOM-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. 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 determines to what extent the map can follow all wrinkles of the data distribution. This effect is not unique for our GSOM-algorithm, but occurs in regular SOMs as well.

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 is an independent feature we will have to separately check for. Several measures have been established for this purpose (for a detailed study see [6]), here we used the topographic product P [1]. During the computation of P for each node $\mathbf{r}$ 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 $\mathbf{w}_{\mathbf{n}_k^V(\mathbf{r})}$. These sequences and further averaging over neighborhood orders k and nodes $\mathbf{r}$ finally leads to

$$P = \frac{1}{N(N-1)} \sum_{\mathbf{r}} \sum_{k=1}^{N-1} \frac{1}{2k} \log \left(\Pi_{l=1}^{k} \frac{d^V\left(\mathbf{w}_\mathbf{r}, \mathbf{w}_{\mathbf{n}_l^A(\mathbf{r})}\right)}{d^V\left(\mathbf{w}_\mathbf{r}, \mathbf{w}_{\mathbf{n}_l^V(\mathbf{r})}\right)} \cdot \frac{d^A\left(\mathbf{r}, \mathbf{n}_l^A(\mathbf{r})\right)}{d^A\left(\mathbf{r}, \mathbf{n}_l^V(\mathbf{r})\right)} \right) \quad . \tag{2.5}$$

P can take on positive or negative values, which have to be interpreted as follows: If $P < 0$ holds the output space is too low-dimensional, in contrast if we have $P > 0$ the output space too high-dimensional. In both cases neighborhood relations are violated. Only if $P \approx 0$ is valid the output space approximately matches topology of input data.

3 Results of Simulations

In a series of exploratory simulations we established that the GSOM-algorithm, mapping data points in input spaces of simple geometry (lines, rectangles, squares, cubes) reliably lead to matching output space geometries, with no sensitive dependence on particular parameters. As a more interesting example let us now consider maps for data points which do not cover the whole input space, but lie on a lower-dimensional, nonlinear submanifold. In order to be still able to plot the results we chose a three-dimensional input space, where the data points $\mathbf{v} = (v_1, v_2, v_3)$ were parameterized by r, s according to

$$v_1 = r + 0.5, \quad v_2 = s + 0.3, \quad v_3 = 4rs(1-r), \quad 0 < r < 1, \quad 0 < s < 0.3. \tag{3.1}$$

Running the GSOM-algorithm on this data set, with $N_{max} = 256$, $\sigma_{ii} = 0.5$, $\sigma_{fi} = 0.1$, $\sigma_{if} = 0.1$, $\sigma_{ff} = 0.02$, $\epsilon_i = 0.9$, and $\epsilon_f = 0.01$. At the end, the GSOM-algorithm adjusted the output space at a dimension of $d = 2$, with 28×9 $(= 252)$ nodes along the two directions. This length ratio reflects the length ratio of the parameter space underlying the data; it was approximately kept constant during the learning procedure.

In subsequent runs we investigated whether the data submanifold could also be detected in a much higher-dimensional space, with noise along all the irrelevant dimensions. To this purpose, we increased the input space dimension to $d_V = 10$, with $v_{1,2,3}$ as in (3.1), and $v_{4 \le i \le 10} = \delta \eta_i$, where η_i is a random variable with uniform distribution and $-0.5 < \eta_i < 0.5$. Simulating GSOMs for this modified data set we find no difference to the noiseless case at low noise levels ($\delta = 0.01$ and $\delta = 0.05$). At larger noise levels ($\delta = 0.1, 0.15$ and 0.2), additional output space dimensions

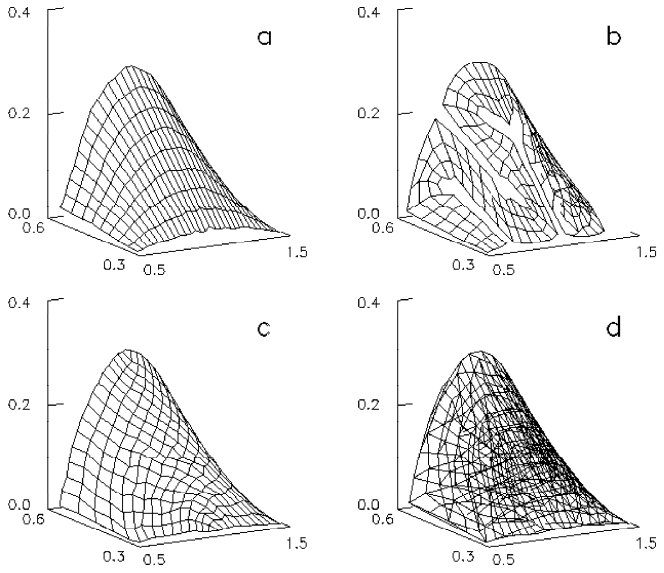

Figure 2: Visualization of the 28×9 GSOM-map (part a), and the 64×4 (b), 16×16 (c) and $7 \times 6 \times 6$ (d) fixed output space maps obtained for example 1. The weights vectors $\mathbf{w_r}$ lie on the two-dimensional submanifold of $\Re^3$ which is covered by the data points.

are recruited by the algorithm (up to a $10 \times 3 \times 2 \times 2 \times 2$ output space with $d_A = 5$, at $\delta = 0.2$). This increase of d_A does not indicate a break-down of our algorithm; instead it has to be regarded as a consequence of the deviation of the data points from a strictly two-dimensional submanifold. Depending on the number of map nodes available, the map tries to also represent the additional dimensions $4-10$. The issue of how to distinguish between relevant data dimensions and irrelevant noise dimensions, and how to control the stiffness parameter σ to avoid extensions into the irrelevant dimensions, are not specific to the present topic of output space growth and have been addressed elsewhere [5].

We also performed several tests of the GSOM algorithm on real world data sets. Our first example is based on speech data, which we have already used in previous publications [1]. The present data set has been acquired in the III. Physikalisches Institut, Universität Göttingen. It contains the ten (German) digits, spoken ten times each by one speaker. The preprocessing of the data resulted in 2013 feature vectors, in a $19d$ input space. For this data set the growth algorithm delivered maps with a) $2 \times 12 \times 3 \times 3$, b) $3 \times 9 \times 3 \times 3$ and c) $9 \times 3 \times 3 \times 3$ nodes, for a limit of $N_{max} = 256$ nodes. Using the topographic product P we now analyze these maps with regard to their degree of neighborhood preservation, which yields $P_a = -0.021$, $P_b = 0.0000325$, $P_c = -0.00396$, respectively. In order to be able to relate these values to those for regular maps, we also trained fixed output space maps of the dimensionalities $d = 1 - 5$. In these cases we obtained the values $P_1 = -0.23$, $P_2 = -0.055$, $P_3 = -0.003$, $P_4 = 0.020$, $P_5 = 0.049$. So the maps with grown output spaces yield the neighborhood preservation values about as good as those for the best fixed output space maps.

Satellites of LANDSAT-TM type produce pictures of the earth in 7 different spectral bands. The ground resolution in meter is 30×30 for the bands 1-5 and band 7. Band 6 has a resolution of 60×60 only. The spectral bands represent useful domains of the whole spectrum in order to detect and discriminate vegetation, water, rock formations and cultural features [9]. The spectral information, i.e., the intensity of the bands associated with each pixel of a LANDSAT scene is represented by a vector $\mathbf{v} \in V \subseteq \Re^{d_V}$ with $d_V = 7$, the number of spectral bands. Because of the

rougher resolution of band 6 (thermal band) this channel is often dropped. Hence, the LANDSAT data may be represented as clouds of data points in a 6-dimensional space. The aim of any classification algorithm is to subdivide this data space into subsets of data points which belong to a certain category corresponding to a specific feature like wood, industrial region, etc., each feature being specified by a certain prototype data vector. One way to get good results for visualization is to use a SOM dimension $d_A = 3$ [10]. Then we are able to interpret the positions of the neurons in the lattice A as a vector $c = (r, g, b)$ in the color space $\mathcal{C}$, where r, g, b are the intensity of the colors red, green and blue. This assigns colors to winner neurons so that we end up immediately with the pseudo color version of the original picture for visual interpretation. However, since we are mapping the data clouds from a 6-dimensional input space onto a *three*-dimensional color space there may arise dimensional conflicts and the visual interpretation may fail. We investigated a picture of the north–east region of Leipzig[5]. Here the GSOM algorithm ($N_{\max} = 300$) led us to a map with 18×16 neurons, i.e. a mapping which is uses only two dimensions of the three–dimensional color space. Is this result consistent with the results of other methods? In order to provide a comparison, we also trained standard SOMs of dimensions $d_A = 1, 2, 3$ (256, 16×16 and $7 \times 6 \times 6$ nodes, respectively). An evaluation of the degree of neighborhood preservation for these maps using the topographic product yield the values $P_1 = -0.0569$, $P_2 = 0.0066$, $P_3 = 0.0449$for the one–, two– and three–dimensional maps, respectively. So the map delivered by the GSOM coincides very closely with the map which would be picked by the more cumbersome trial and error method.

These two examples show that the growth algorithm works well and produces properly structures of the underlying lattice of neurons. For further investigations of the algorithm see [2].

References

[1] Bauer, H.-U.; Pawelzik, K.: Quantifying the Neighborhood Preservation of Self-Organizing Feature Maps. *IEEE Trans. on NN* **3(4)**, 570-579, (1992).

[2] Bauer, H.-U.; Villmann, Th.: Growing a Hypercubical Output Space in a Self-Organizing Feature Map. *IEEE Trans. on NN* **7(6)**, (1996).

[3] B. Fritzke: Growing Cell Structures—A Self-Organizing Network for Unsupervised and Supervised Learning, Neural Networks **7**, 1441-1460 (1994).

[4] A. Hämäläinen: Genetic Algorithm in Self-Organizing Map Design, ICANNGA 95, Alès, France (1995).

[5] Herrmann, M.: M. Herrmann, Self-Organizing Feature Maps with Self-Organizing Neighborhood Widths, in: *Proc. of ICNN 95 Perth* (1995).

[6] Herrmann, M.;Bauer, H.-U.; Villmann, Th.: Topology Preservation in Neural Maps. to appear in Proc. ESANN'97, Brüssel, (1997).

[7] Jockusch, S.; Ritter, H.: Self-Organizing Maps: Local Competition and Evolutionary Optimization, Neural Networks **7**, 1229-1240 (1994).

[8] Kohonen, T.: Self-Organization and Associative Memory. Springer, Berlin, Heidelberg (1989).

[9] Bähr, H.-P.; Vögtle, Th.: Digitale Bildverarbeitung: Anwendung in Photogrammetrie, Kartographie und Fernerkundung. Wichmann-Verlag, (1991).

[10] Groß, M.; Seibert, F.: Neural Network Image Analysis for Environmental Protection. Zentrum für graphische Datenverarbeitung, Darmstadt, (1993).

[5] obtained from UMWELT-FORSCHUNGSZENTRUM Halle-Leipzig, Germany, Arbeitsgruppe "Angewandte Landschaftsökologie"