A vector field visualization technique for Self-Organizing Maps

Georg Pölzlbauer[1], Andreas Rauber[1], and Michael Dittenbach[2]

[1] Department of Software Technology
Vienna University of Technology
Favoritenstr. 11-13, Vienna, Austria
{poelzlbauer,rauber}@ifs.tuwien.ac.at
[2] eCommerce Competence Center – ec3
Donau-City-Str. 1, Vienna, Austria
michael.dittenbach@ec3.at

Abstract. The Self-Organizing Map is one of most prominent tools for the analysis and visualization of high-dimensional data. We propose a novel visualization technique for Self-Organizing Maps which can be displayed either as a vector field where arrows point to cluster centers, or as a plot that stresses cluster borders. A parameter is provided that allows for visualization of the cluster structure at different levels of detail. Furthermore, we present a number of experimental results using standard data mining benchmark data.

1 Introduction

The Self-Organizing Map (SOM) [1] is a valuable tool in data analysis. It is a popular unsupervised neural network algorithm that has been used in a wide range of scientific and industrial applications [3], like Text Mining [6], natural language processing and monitoring of the condition of industrial plants and processes. It provides several beneficial properties, such as vector quantization and topology preserving mapping from a high-dimensional input space to a two-dimensional output space. This projection can be visualized in numerous ways in order to reveal the characteristics of the input data or to analyze the quality of the obtained mapping.

Our method is based on the SOM codebook and the neighborhood kernel, which induces a concept of proximity on the map. For each map unit, we compute a vector pointing to the direction of the most similar region in output space. We propose two methods of visualizing the results, a vector field plot, which can be seen analogous to flow visualization and gradient visualization, and a plot that emphasizes on the cluster structure of the map. The SOMs used for demonstration and experiments are trained on Fisher's well-known Iris data.

The rest of this paper is organized as follows. Section 2 describes several visualization techniques for SOMs and related work. Section 3 gives an overview of neighborhood kernel functions and their parametrization. In Section 4, our

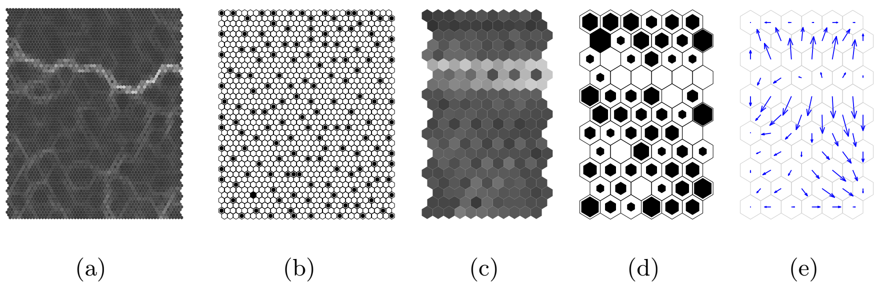

Fig. 1. 30×40 SOM: (a) U-Matrix, (b) Hit histogram, 6×11 SOM: (c) U-Matrix, (d) Hit histogram, (e) Vector Field with Gaussian kernel ($\sigma = 2$), see Section 5

visualization method is introduced, along with a description of its properties and interpretations. Section 5 presents experimental results, where the the influence of choices of neighborhood kernel, neighborhood radius and map size are investigated. Finally, Section 6 gives a short summary of the findings presented in this paper.

2 Related Work

In this section, we briefly describe visualization concepts for SOMs related to our novel method. The most common ones are component planes and the U-Matrix. Both take only the prototype vectors and not the data vectors into account. Component planes show projections of singled out dimensions of the prototype vectors. If performed for each individual component, they are the most precise and complete representation available. However, cluster borders cannot be easily perceived, and high input space dimensions result in lots of plots, a problem that many visualization methods in multivariate statistics, like scatterplots, suffer from. The U-Matrix technique is a single plot that shows cluster borders according to dissimilarities between neighboring units. The distance between each map unit and its neighbors is computed and visualized on the map lattice, usually through color coding. Recently, an extension to the U-Matrix has been proposed, the U*-Matrix [8], that relies on yet another visualization method, the P-Matrix [7]. Other than the original, it is computed by taking both the prototype vectors and the data vectors into account and is based on density of data around the model vectors. Interestingly, both the U*-Matrix and our novel method, among other goals, aim at smoothing the fine-structured clusters that make the U-Matrix visualization for these large SOMs less comprehensible, although the techniques are conceptually totally different. Other visualization techniques include hit histograms and Smoothed Data Histograms [4], which both take the distribution of data into account, and projections of the SOM codebook with concepts like PCA or Sammon's Mapping, and concepts that perform labeling of the SOM lattice [6]. For an in-depth discussion, see [9].

In Figure 1, the hit histogram and U-Matrix visualizations are depicted for SOMs trained on the Iris data set with 30×40 and 6×11 map units, respectively.

The feature dimensions have been normalized to unit variance. The U-Matrix reveals that the upper third of the map is clearly separated from the rest of the map. The hit histogram shows the projection of the data samples onto the map lattice. It can be seen that this SOM is very sparsely populated, because the number of map units is higher than the number of data samples. When the two methods are compared, it can be observed that the fine cluster structures in the U-Matrix occur exactly between the map units that are occupied by data points. It is one of the goals of this work to create a representation that allows a more global perspective on these kinds of maps and visualize it such that the intended level of detail can be configured.

To our best knowledge, the neighborhood kernel that is described in the next section has not been used for visualization purposes. Apart from the SOM training algorithm the neighborhood function is applied in the SOM Distortion Measure [2], which is the energy function of the SOM with fixed radius, where the neighborhood kernel is aggregated and serves as a weighting factor comparable to the one we use in this paper.

3 SOM Neighborhood Kernels

A particularly important component of the Self-Organizing Map is the concept of adjacency in output space, i.e. the topology of the map lattice, and its definition of neighborhood that affects the training process. Our visualization technique heavily depends on this neighborhood kernel as a weighting factor. The neighborhood kernel is a parameterized function that takes the distance between two map units on the lattice as input and returns a scaling factor that determines by which amount the map unit is updated for each iteration. The parameter the kernel depends on is the neighborhood radius $\sigma(t)$, which is itself a monotonically decreasing function over time t. σ controls the width of the kernel function, such that high values lead to kernels that are stretched out and low values result in sharply peaked kernels. In this work, we will not consider the radius as a function of time as the training process does, but rather as a parameter that has to be specified before the visualization can be applied.

The kernel function $h_\sigma(d_{input})$ has the property of decreasing monotonically with increasing distance d_{input}. This distance will be formally defined in the next section, but can be roughly envisioned as the number of units that lie between two map units. Examples of neighborhood kernels are the Gaussian kernel, the bubble function, and the inverse function. The Gaussian kernel is the most frequently used kernel for the SOM. It has the well-known form of the Gaussian Bell-Shaped Curve, formally

$$h_\sigma^{\mathrm{G}}(d_{input}) = \exp\left(-\frac{d_{input}^2}{2\sigma}\right) \tag{1}$$

Since the returned value is exponentially decreasing for higher values of d_{input}, the effects on the training process are neglegible. Thus, the kernel is frequently modified to cut off the function at input values greater than σ:

$$h_\sigma^{\mathrm{c/G}}(d_{input}) = \begin{cases} h_\sigma^{G}(d_{input}) & \text{if } d_{input} \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

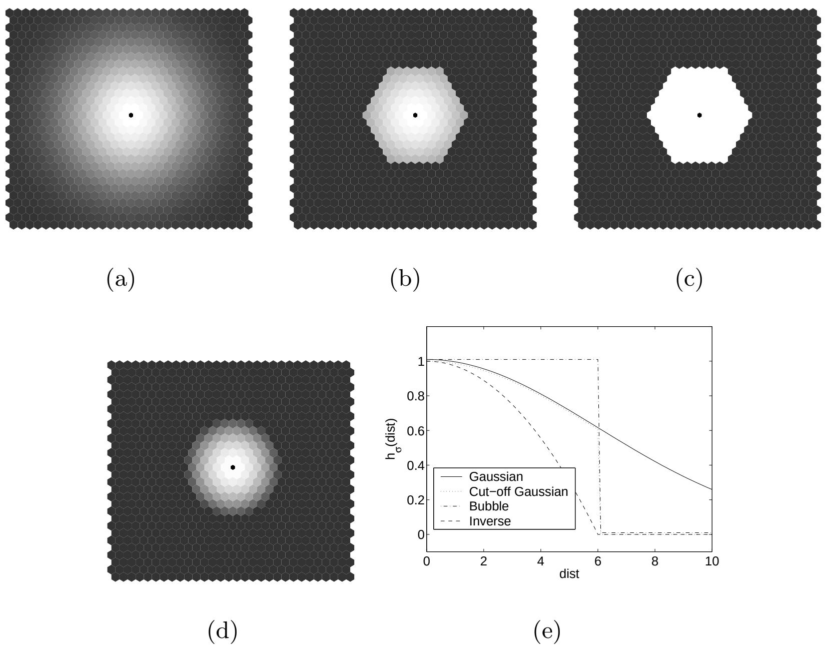

Fig. 2. Overview of different kernel functions: (a) Gaussian kernel, (b) cut-off Gaussian kernel, (c) bubble function, (d) inverse function, (e) comparison of neighborhood functions

Another kernel is the bubble function, which exclusively relies on this principle of cutting off at radius σ. It is a simple step function, formally

$$h_\sigma^{\mathrm{B}}(d_{input}) = \begin{cases} 1 \text{ if } d_{input} \leq \sigma \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

Another option is the inverse proportional function:

$$h_\sigma^{\mathrm{I}}(d_{input}) = \begin{cases} 1 - \frac{d_{input}^2}{\sigma^2} & \text{if } d_{input} \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

which shows a sharper decrease than the Gaussian kernel.

The different kernel functions are depicted in Figure 2, which shows the values of the kernel for the map unit located in the center, indicated by a black dot. Figure 2(e) shows a plot of the kernels as function of the distance between the units and fixed neighborhood radius. All the graphics use the same value of 6 for parameter σ.

4 A Vector Field Based Method for Visualization

In this section, we introduce our visualization technique for the SOM. Similar to the U-Matrix, only the prototype vectors and their pairwise similarities are investigated. In the U-Matrix, only the differences between direct neighbors are considered. We aim to extend this concept to include the region around the units according to the neighborhood kernel. Furthermore, we wish to obtain the direction for each unit where the most similar units are located. The resulting visualization is analogous to gradient vector fields where units are repelled from or attracted to each other.

First, we have to make some formal definitions. The type of SOM that we will consider has a two-dimensional lattice, consisting of a number M of map units p_i, where i is between 1 and M. Each of the map units is linked to a model vector m_i of input dimension N. Each of the m_i is linked to the output space by its position on the map. To distinguish between feature space and map lattice, we explicitly write p_i for the position vector of map unit that represents prototype vector m_i; the index i connects input and output space representation. We denote the horizontal and vertical coordinates of the map unit as p_i^u and p_i^v, respectively. Thus, the distance between two prototype vectors m_i and m_j, or p_i and p_j, can be determined both in input and output space:

$$d_{input}(m_i, m_j) = ||m_i - m_j||_{input} \tag{5}$$

where $||.||_{input}$ is a suitable distance metric and

$$d_{output}(p_i, p_j) = \sqrt{(p_i^u - p_j^u)^2 + (p_i^v - p_j^v)^2} \tag{6}$$

which is the Euclidean Distance.

The neighborhood kernel requires the distance between the model vectors' positions on the map lattice $d_{output}(p_i, p_j)$ as its input. This kernel function computes how much the prototype vectors influence each other during the training process. We will use it as a weighting function that allows us to compute the similarity (in terms of input space distance) of map units that are close to each other on the map.

Our technique plots arrows for each map unit like in gradient field visualizations. A unit's arrow points to the region where the most similar prototype vectors are located on the map. The length of this arrow shows the degree of how much the area it is pointing to is more similar to it than the opposite direction.

Each arrow is computed for unit p_i as a two-dimensional vector a_i. It can be decomposed in u and v coordinates, denoted as a_i^u and a_i^v. For each of the two axes, we compute the amount of dissimilarity along positive and negative directions. Our method determines these vectors in a two-step process: First, the computations for each map unit are performed separately for the positive and negative directions of axes u and v, and finally, these components are aggregated by a weighting scheme to calculate the coordinates of a_i.

The angle α that identifies the direction of p_j seen from p_i on the map lattice is defined in basic trigonometry as

$$\alpha(p_i, p_j) = \arctan(\frac{p_j^v - p_i^v}{p_j^u - p_i^u}) \tag{7}$$

The influence of the neighborhood kernel projected onto the u and v axes is computed as

$$w^u(p_i, p_j) = \cos(\alpha(p_i, p_j)) \cdot h_\sigma(d_{output}(p_i, p_j)) \tag{8}$$

$$w^v(p_i, p_j) = \sin(\alpha(p_i, p_j)) \cdot h_\sigma(d_{output}(p_i, p_j)) \tag{9}$$

Here, the influence of the neighborhood kernel is distributed among the two axes according to the position of p_i and p_j on the map and serves as a weighting factor in the following steps. The neighborhood kernel relies on the width parameter σ, which determines the influence of far-away map units.

Then, we decompose the amount of dissimilarity in its positive and negative direction for both axes for each pair of map units p_i, p_j:

$$con^u_+(p_i, p_j) = \begin{cases} d_{input}(m_i, m_j) \cdot w^u(p_i, p_j) & \text{if } w^u(p_i, p_j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$con^u_-(p_i, p_j) = \begin{cases} -d_{input}(m_i, m_j) \cdot w^u(p_i, p_j) & \text{if } w^u(p_i, p_j) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where con^u_+ denotes the contribution of map unit p_j's dissimilarity in positive direction along u, and con^u_- in negative direction. The definition of con^v_+ and con^v_- follows analogously. For example, a map unit p_j that lies to the lower right of p_i results in $con^u_-(p_i, p_j) = con^v_+(p_i, p_j) = 0$, and some positive values for $con^u_+(p_i, p_j)$ and $con^v_-(p_i, p_j)$ according to the distance in output space, which is weighted through the neighborhood kernel, and also its distance in input space, which is directly measured by the factor d_{input}.

Next, the sum of contributions in both directions is computed for each p_i

$$diss^u_+(p_i) = \sum_{j=1\ldots M, j\neq i} con^u_+(p_i, p_j) \tag{12}$$

$$diss^u_-(p_i) = \sum_{j=1\ldots M, j\neq i} con^u_-(p_i, p_j) \tag{13}$$

Again, $diss^v_+$ and $diss^v_-$ are defined analogously. The variable $diss^u_+(p_i)$ indicates how much m_i is dissimilar from its neighbors on the side in the positive u direction. In a gradient field analogy, this value shows how much it is repelled from the area on the right-hand side.

Next, we aggregate both negative and positive components into the resulting vector a_i. Normalization has to be performed, because units at the borders of the map lattice would have components pointing outside of the map equal to zero, which is not intended. The sums of the neighborhood kernel weights w_i pointing in positive and negative directions are

$$w^u_+(p_i) = \sum_{j=1\ldots M, j\neq i} \begin{cases} w^u(p_i, p_j) & \text{if } w^u(p_i, p_j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$w_-^u(p_i) = \sum_{j=1\dots M, j\neq i} \begin{cases} -w^u(p_i,p_j) & \text{if } w^u(p_i,p_j) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Finally, the u component of the gradient vector a is computed as

$$a_i^u = \frac{diss_-^u(p_i) \cdot w_+^u(p_i) - diss_+^u(p_i) \cdot w_-^u(p_i)}{diss_+^u(p_i) + diss_-^u(p_i)} \tag{16}$$

and likewise for the v direction. The weighting factor w_+^u is multiplied with the component in the other direction to negate the effects of units close to the border in which case the sum of the neighborhood kernel is greater on one side. If this normalization would be omitted, the vector a would be biased towards pointing to the side where units are missing. For map units in the center of the map's u-axis, where w_+^u and w_-^u are approximately equal, Equation (16) can be approximated by this simpler formula

$$a_i^u \approx \mu \cdot \frac{diss_-^u(p_i) - diss_+^u(p_i)}{diss_+^u(p_i) + diss_-^u(p_i)} \tag{17}$$

where μ is a constant factor equal to $\frac{w_+^u + w_-^u}{2}$ and is approximately the same for all units in the middle of an axis.

The results obtained for different ratios and proportions of $diss_+$ and $diss_-$ are briefly described:

– If negative and positive dissimilarities are roughly equal, the resulting component of a will be close to zero.
– If the positive direction is higher than the negative one, a will point into the negative direction, and vice versa. The reason for this is that the prototype vectors on the negative side of the axis are more similar to the current map unit than on the positive side.
– If one side dominates, but the second side still has a high absolute value, the normalization performed in the denominator of Equation (16) decreases the length of the vector.

In Figure 3(b), our visualization technique is shown for the 30×40 SOM trained on the Iris data set with a Gaussian kernel with $\sigma = 5$. If compared to the U-Matrix in Figure 1(a), it can be seen that the longest arrows are observed near the cluster borders, pointing to the interior of their cluster and away from these borders. Adjacent units, for which the arrow points in different directions, are clearly along a cluster border. The length of the arrows indicates how sharp the border is. In the middle of these transitions, arrows are sometimes drawn with almost no distinguishable length or direction. The corresponding prototype vectors are likely to be very far away from either cluster, and are referred to as interpolating units, since they do not represent any data vectors in a vector quantization sense, but are only a link connecting two distant data clouds. Cluster centers also have small dot-like arrows pointing in no distinguishable direction, but the difference is that the surrounding arrows are pointing in their direction, and not away from them. Another property of this visualization is that the units on the edges of the map never point outside of it, which is desired and stems from the normalization performed in (16).

One interesting extension to our visualization method is that the results can also be depicted to show the cluster borders themselves with a slight modification in the representation by depicting not the direction of the gradient, but rather the hyperplane obtained by rotation of 90 degrees in either direction. In our case of a two-dimensional map lattice, the hyperplane is a one-dimensional line. We choose to depict this line with length proportional to the original arrow. The result is visualized in Figure 3(e). The emphasis of this dual representation is stressing cluster borders, while information on directions is omitted.

We have found that our method is most useful when applied in combination with other visualization techniques, such as hit histograms and component planes. What can be learned from comparing the positions of the different Iris species to our method is that the class membership of the data samples correlates with the cluster structure in case of the Setosa species, while Versicolor and Virginica do not show a distinguishable separation. This is of course a well-known fact about the Iris data set, and application of our technique to more complex data is subject to further research and is addressed in [5].

5 Experiments

In this section, we will investigate the empirical results of our method applied to SOMs of different sizes, as well as how the choice of parameter σ influences the visualization, and the effects of different kernel functions.

First, we examine the effect of the map size, i.e. the number of prototype vectors. The data vectors remain the same for both maps. The smaller version of the SOM consists of 6×11 units, and the larger one of 30×40 units. In the latter case, the number of data vectors (150) is much lower than the number of map units (1200). The visualization for the smaller version is depicted in Figure 1(e). U-Matrix and vector field plots for the larger map are shown in Figures 1(a) and 3, respectively. In the smaller SOM the gap between the upper third part representing the well-separated Setosa species and the lower two-thirds of the map can clearly be distinguished, as in the larger SOM. However, the larger version of the SOM gives more insight into the structure of the data. Transitions and gradual changes in directions and length can be distinguished more easily at this higher granularity.

In the next experiment, we investigate the influence of the width parameter σ. In Figure 3, the large Iris SOM is visualized with three different values of σ. Figures 3(a), (d) show the two methods for $\sigma = 1$. The visualization with this width is the one most closely related to the U-Matrix technique, since only distances between direct neighbors are regarded, while the influence of slightly more distant units is neglected. Of all the visualizations shown here, these two are chiseled the most and are least smooth. The frequent changes in direction of neighboring arrows is due to the very local nature of this kernel. In Figures 3(b), (e) the visualization is shown for $\sigma = 5$, where the increased neighborhood radius produces a smoothing effect over the vector field. Here, changes in direction between close arrows can be better distinguished and result in a visually more comprehensible

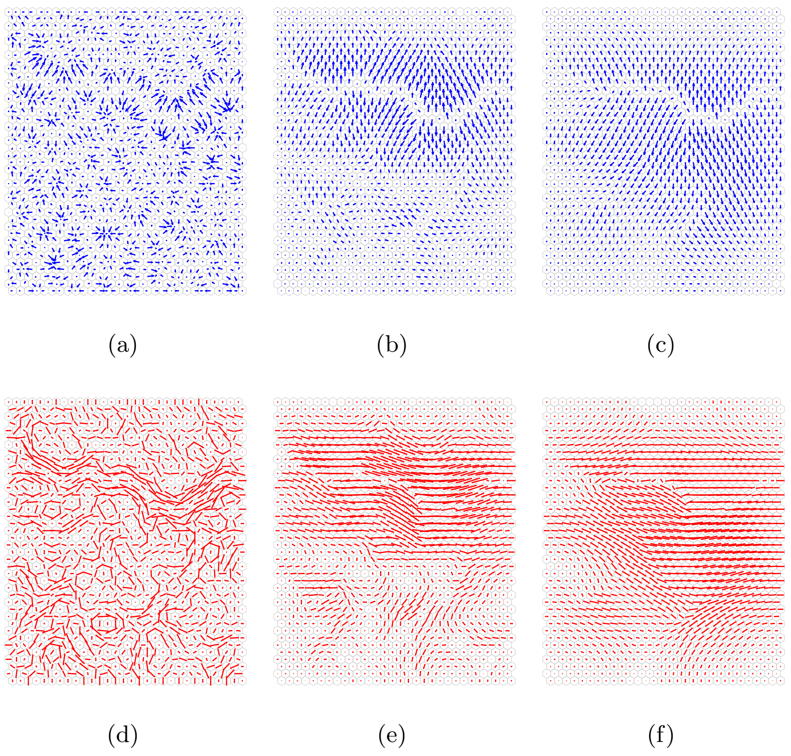

Fig. 3. 30×40 SOM trained on Iris data (a)-(c) Vector field representation with $\sigma = 1, 5, 15$, (d)-(f) Border representation with $\sigma = 1, 5, 15$

picture. The set of arrows is perceived as a whole and as less chaotic. It gives the impression of visualizing a somewhat more global structure. Finally, the visualization for $\sigma = 15$ is depicted in Figures 3(c), (f), where only big clusters can be perceived. The effect of σ can be summarized as follows: For a value of 1, the cluster representation is very similar to the U-Matrix, which is the method relying mostly on local differences. With higher values of σ, the kinds of perceived cluster structures gradually shift from local to global. The choice of σ has a deep impact on this visualization method and is dependant on the map size. Further experiments have shown that good choices are close to one tenth of the number of map units in the axis of the map lattice with fewer map units, but it also depends on the desired level of granularity.

Finally, we investigate the influence of the type of neighborhood function on the visualization. The examples in this paper so far are all performed with Gaussian kernels. Surprisingly, the differences to the inverse function and cut-off Gaussian kernel are so minimal that they are hardly distinguishable. The only exception is the bubble function, which is actually a very unusual choice for a neighborhood kernel during training. Since all the map units are treated equally within the sphere of this radius, and nodes on the borders of this circle are not weighted less than near the center, the visualization is harder to interpret than the other kernels. During training, cluster structures are introduced that are not present in the data set. We find that the bubble function is not appropriate for this kind of visualization, and conclude that the neighborhood kernel should be a continuous function.

6 Conclusion

In this paper, we have introduced a novel method of displaying the cluster structure of Self-Organizing Maps. Our method is distantly related to the U-Matrix. It is based on the neighborhood kernel function and on aggregation of distances in the proximity of each codebook vector. It requires a parameter σ that determines the smoothness and the level of detail of the visualization. It can be displayed either as a vector field as used in flow visualizations or as a plot that highlights the cluster borders of the map. In the former case, the direction of the most similar region is pointed to by an arrow. Our experiments have shown that this method is especially useful for maps with high numbers of units and that the choice of the neighborhood kernel is not important (as long as it is continuous), while the neighborhood radius σ has a major impact on the outcome.

Acknowledgements

Part of this work was supported by the European Union in the IST 6. Framework Program, MUSCLE NoE on Multimedia Understanding through Semantics, Computation and Learning, contract 507752.

References

1. T. Kohonen. *Self-Organizing Maps, 3rd edition.* Springer, 2001.
2. J. Lampinen and E. Oja. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2(2–3):261–272, 1992.
3. M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum. *Neural Computing Surveys*, 3:1–156, 2001.
4. E. Pampalk, A. Rauber, and D. Merkl. Using smoothed data histograms for cluster visualization in self-organizing maps. In *Proc. Intl. Conf. on Artifical Neural Networks (ICANN'02)*, Madrid, Spain, 2002. Springer.
5. G. Pölzlbauer, A. Rauber, and M. Dittenbach. A visualization technique for self-organizing maps with vector fields to obtain the cluster structure at desired levels of detail. In *International Joint Conference on Neural Networks (IJCNN2005)*, Montral, Canada, 2005.
6. A. Rauber and D. Merkl. Automatic labeling of self-organizing maps: Making a treasure-map reveal its secrets. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'99)*, Bejing, China, 1999. Springer.
7. A. Ultsch. Maps for the visualization of high-dimensional data spaces. In *Proc. Workshop on Self organizing Maps*, Kyushu, Japan, 2003.
8. A. Ultsch. U*-matrix: a tool to visualize clusters in high dimensional data. Technical report, Departement of Mathematics and Computer Science, Philipps-University Marburg, 2003.
9. J. Vesanto. *Data Exploration Process Based on the Self-Organizing Map.* PhD thesis, Helsinki University of Technology, 2002.