Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

T. GALLIAT[†] AND P. DEUFLHARD[†] [‡]

Adaptive hierarchical cluster analysis by Self-Organizing Box Maps

[†]Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany.
Internet: http://www.zib.de/DataMining
[‡]Freie Universität Berlin, Fachbereich Mathematik und Informatik, Germany

ZIB-Report 00-13 (April 2000)

Adaptive hierarchical cluster analysis by Self-Organizing Box Maps

T. Galliat[1] P. Deuflhard[1,2]

[1] Konrad-Zuse-Zentrum Berlin, Takustr. 7, 14195 Berlin, Germany
[2] Freie Universität Berlin, Fachbereich Mathematik und Informatik, Arnimallee 2–6, 14195 Berlin, Germany

Abstract

The present paper aims at an extension of KOHONEN's Self-Organizing Map (SOM) algorithm to be called Self-Organizing Box Map (SOBM) algorithm; it generates box codebooks in lieu of point codebooks. Box codebooks just like point codebooks indirectly define a Voronoi tessellation of the input space, so that each codebook vector represents a unique set of points. Each box codebook vector comprises a multi-dimensional interval that approximates the related partition of the Voronoi tessellation. Upon using the automated cluster identification method that has recently been developed by the authors, the codebook vectors can be grouped in such a way that each group represents a point cluster in the input space. Since the clustering usually depends on the size of the SOM, one cannot be sure, whether the clustering comes out to be optimal. Refinement of part of the identified clusters would often improve the results. This paper presents the concept of an adaptive multilevel cluster algorithm that performs such refinements automatically. Moreover the paper introduces a concept of essential dimensions and suggests a method for their identification based on our herein suggested box codebooks. Applications of the algorithm to molecular dynamics will be described in a forthcoming paper.

Keywords. Self-Organizing Maps, cluster analysis, Voronoi tessellation, feature extraction, essential dimensions, multilevel methods.

1 Introduction

Traditional methods for cluster analysis of high-dimensional data (see [2] for a survey) imply the problem of assessment of clustering quality. In particular, if one wants to automate the cluster analysis process, one needs a measure to compare different clusterings. Although there are several suggestions for such measures, they all share the disadvantage that they depend on the a priori unknown number of clusters, i.e., they cannot be employed to compare clusterings with different numbers of clusters. An alternative methodical approach that tries to avoid this—obviously—systematic problem, is cluster analysis by Self-Organizing Maps (SOMs). We have recently shown in [6], how SOMs can be combined with eigenmode analysis for an automated cluster identification. Although this approach is very powerful and generally applicable, there are still

two mutually connected problems. First, the clustering still depends slightly on the chosen map size. Second, we cannot be sure that the cluster level is optimal, i.e., it may happen that a sub-clustering of one or more clusters would lead to a better total clustering. Obviously, it is not promising to solve these problems by searching for a suitable quality measure for clusterings, because then one would be just in the same situation as within the traditional methods for cluster analysis. Rather the present paper aims at the derivation of a *hierarchical multilevel* method together with a stopping criterion, which monitors whether a cluster refinement is necessary.

In Section 2 we develop an extended SOM algorithm that computes box codebook vectors. Box codebooks represent more information about the data distribution than point codebook vectors, as they are generated by the original SOM algorithm. Therefore they possibly open the door for further applications of SOMs within Data Mining. In Section 3 we introduce the concept of essential dimensions and suggest an identification method that uses box codebook vectors. Finally in Section 4 we describes our multilevel cluster algorithm together with the stopping criterion that depends on the concept of essential dimensions.

2 Self-Organizing Box Map Algorithm

In this section we describe the extensions of KOHONEN'S Self-Organizing Map algorithm, so that it generates box codebooks instead of point codebooks. We compare the original SOM and the so called Self-Organizing Box Map (SOBM) algorithm and suggest a promising combination of both. The reader is assumed to be familiar with the basics of the SOM algorithm, otherwise see [7, 4] for an introduction.

In the following, we consider a q–dimensional input space $\Omega \subset \mathbf{R}^q$ and a probability distribution P_ρ on Ω with a probability density function $\rho : \Omega \to \mathbf{R}_0^+$, so that $P_\rho(\Omega) = \int_\Omega \rho(\omega) d\omega = 1$. Note that in the usual case, where we have only a finite number of sample vectors, such a function ρ can always be constructed if we use a discrete input space. In this case all integral signs have to be replaced by sums.

Based on P_ρ we want to compute during T time-steps a two-dimensional SOM formed by an $n \times m$ grid with rectangular or hexagonal topology and $k = nm$ codebook vectors $W_1, \ldots, W_k$. For each codebook vector, $z_s \in G$ denotes the (x, y) position of the related neuron s on the grid $G := \{1, \ldots, n\} \times \{1, \ldots, m\}$. For the computation of the map, we use a problem specific distance function $\mathrm{dist} : \Omega \times \Omega \to \mathbf{R}$, a time-dependent learning rate $\alpha : \{0, \ldots, T\} \to [0, 1]$ and a time-dependent neighborhood function $\mathrm{neigh} : G \times G \times \{0, \ldots, T\} \to [0, 1]$ with $\mathrm{neigh}(z(s), z(s), t) = 1$ for all $s \in \{1, \ldots, k\}$ and $t \in \{0, \ldots, T\}$.

Using the original SOM algorithm we generate point codebook vectors, i.e., $W_s = (W_{s_1}, \ldots, W_{s_q}) \in \Omega$. If we want to compute box codebook vectors, i.e., $\hat{W}_s := ([l_{s_1}, r_{s_1}], \ldots, [l_{s_q}, r_{s_q}]) \subset \Omega$, we have to extend the algorithm suitably.

To compute box codebook vectors, we have to make alterations to the initialization step, the winner function and the adaption rule of the original SOM algorithm. To avoid confusion, we use W_s for point codebook vectors and $\hat{W}_s$ for box codebook vectors. Note that $\hat{W}_s$ is a box in Ω, if we use the following definition:

Definition 2.1 *Let* $[l_i, r_i] \subset \mathbf{R}$ *denote a non-void interval for all* $i = 1, \ldots, q$ *and let* $\Delta \subset \Omega \subset \mathbf{R}^q$*, such that* $\Delta := \bigotimes_{i=1}^{q} [l_i, r_i] := ([l_1, r_1], \ldots, [l_q, r_q])$*. Then* Δ *is called a box in* Ω*. Set* $\mathrm{BOX}(\Omega) := \{\Delta \mid \Delta \text{ arbitrary box in } \Omega\}$.

Initialization Let $W_1(0), \ldots, W_k(0)$ be the different initial values for the codebook vectors, e.g., selected as arbitrary, but different random vectors in Ω. For our extended algorithm, we choose $\hat{W}_s(0) := \bigotimes_{i=1}^{q} [l_{s_i}(0), r_{s_i}(0)]$ with $l_{s_i}(0) = W_{s_i}(0)$ and $r_{s_i}(0) = W_{s_i}(0) + \epsilon$ in terms of a small positive value ϵ, the initial width of the interval, such that $\hat{W}_s \cap \hat{W}_p = \emptyset$ for all $s, p \in \{1, \ldots, k\}$.

If one selects the left hand boundaries $l_{s_i}(0)$ randomly, one could think about selecting the right hand boundaries $r_{s_i}(0)$ randomly, too. However, as we will see, the interval width influences the chance of the related neuron to be the winner neuron, which makes this procedure doubtful. Therefore we define the interval width to be equal for all codebook vectors after the initialization step. Furthermore, we will show that for the suggested way of initialization, the resulting maps can be compared with maps that are computed with the original algorithm. If we selected both interval boundaries randomly, a comparison would turn out to be rather difficult.

Winner function Suppose that the problem specific q-dimensional distance function $dist(X, Y)$ with $X = (X_1, \ldots, X_q)$, $Y = (Y_1, \ldots, Y_q) \in \Omega$ can be written as a function of q one-dimensional distance measures $d_i(X_i, Y_i)$, which means

$$\mathrm{dist}(X, Y) := f(d_1(X_1, Y_1), \ldots, d_q(X_q, Y_q)).$$

In fact, many popular distance measures just exhibit this feature. As an example, for the Euclidean distance we have:

$$f_{\mathrm{euclid}}(d_1, \ldots, d_q) := (\sum_{i=1}^{q} d_i)^{1/2} \quad \text{with} \quad d_i(X_i, Y_i) := (X_i - Y_i)^2.$$

Let $X = (X_1, \ldots, X_q) \in \Omega$ be an arbitrary input and $W_1, \ldots, W_k$ respectively $\hat{W}_1, \ldots, \hat{W}_k$ the actual codebook vectors of the SOM.

In the original algorithm, the *winner neuron* $p \in \{1, \ldots, k\}$ matches the following necessary condition:

$$\mathrm{dist}(X, W_p) = \min_{s \in \{1, \ldots, k\}} \mathrm{dist}(X, W_s)\,. \tag{1}$$

In the case that there is more than one neuron, which matches Eq. (1), various strategies are used. Sometimes the winner is chosen randomly, but usually

the one with the lowest index is taken as the actual winner.

In our extended algorithm, we have to check both interval boundaries. Therefore we compute for each $\hat{W}_s = (\hat{W}_{s_1}, \ldots, \hat{W}_{s_q})$ the distance from the input X with the distance function $\mathrm{DIST} : \Omega \times \mathrm{BOX}(\Omega) \to \mathbf{R}$:

$$\mathrm{DIST}(X, \hat{W}_s) := f(\hat{d}_1(X_1, \hat{W}_{s_1}), \ldots, \hat{d}_q(X_q, \hat{W}_{s_q}))$$

with

$$\hat{d}_i(X_i, \hat{W}_{s_i}) := \begin{cases} d_i(X_i, l_{s_i}) + \kappa\, d_i(X_i, r_{s_i}) & \text{if } X_i < l_{s_i} \\ \kappa\, d_i(X_i, l_{s_i}) + d_i(X_i, r_{s_i}) & \text{if } X_i > r_{s_i} \\ \kappa\, d_i(X_i, l_{s_i}) + \kappa\, d_i(X_i, r_{s_i}) & \text{else} \end{cases}$$

for some $\kappa \in [0, 1]$ and $\hat{W}_{s_i} = [l_{s_i}, r_{s_i}]$.

Then the winner neuron has to match a condition analogous to Eq. (1):

$$\mathrm{DIST}(X, \hat{W}_p) = \min_{s \in \{1, \ldots, k\}} \hat{\mathrm{dist}}(X, \hat{W}_s)\,. \tag{2}$$

Suppose first, we choose $\kappa = 0$. Then the distance is zero, if the input X_i is inside the interval $\hat{W}_{s_i}$ respectively depends only on the nearest interval boundary, if X_i is outside the interval. Obviously, in this case the algorithm tends to favor big intervals. If the dimension q of the input space is "low" this may cause problems, because it could prevent a sufficient expansion of the map. Also the generated box codebook vectors could be unsuitable for the identification of essential dimensions (see Section 3). Therefore in such situations it is better to use a larger κ, so that not only both interval boundaries are considered for the computation of the distance, when the input is outside the interval, but also if X_i is inside the interval. If one uses the Euclidean distance, $\kappa = 0.25$ is a good choice for low-dimensional Ω. In this case we have

$$X_i \approx r_{s_i} \implies \kappa\, d_i(X_i, l_{s_i}) \approx \left(\frac{r_{s_i} - l_{s_i}}{2}\right)^2$$

and

$$X_i \approx l_{s_i} \implies \kappa\, d_i(X_i, r_{s_i}) \approx \left(\frac{r_{s_i} - l_{s_i}}{2}\right)^2.$$

y If the dimension of the input space is high enough, $\kappa = 0$ seems to cause no problems. In this case usually each codebook vector has some big intervals and some small intervals. This balances the preferring of the big intervals.

If there is more than one neuron, that matches Eq. 2, one can use the same strategies as for point codebook vectors. But in the extended algorithm, one

may also think about additional strategies. One possibility is to choose the winner neuron p whose related box has minimal volume:

$$\text{boxvol}(\hat{W}_p) = \min_{\hat{W}_s \in S(X)} \text{boxvol}(\hat{W}_s)$$

with

$$S(X) := \{\hat{W}_p \,|\, \hat{\text{dist}}(X, \hat{W}_p) = \min_{s \in \{1,\ldots,k\}} \hat{\text{dist}}(X, \hat{W}_s)\}$$

and

$$\text{boxvol}(\hat{W}_s) := \prod_{i=1}^{q} (r_{s_i} - l_{s_i})\,.$$

This strategy favors smaller boxes, what makes sense, if one prefers boxes that differ not too much in volume.

Adaption rule Let neuron p be the winner neuron for input $X(t) = (X_1(t), \ldots, X_q(t)) \in \Omega$ at time $t \in \{0, \ldots, T-1\}$ and $W_1(t), \ldots, W_k(t)$ respectively $\hat{W}_1(t), \ldots, \hat{W}_k(t)$ the actual codebook vectors.

In the classical SOM method, the new point codebook is computed in the following way (remember that z_s denotes the neuron position on the grid):

$$W_s(t+1) := W_s(t) + \alpha(t)\,\text{neigh}(z_s, z_p, t)\,(X(t) - W_s(t)), \quad s = 1, \ldots, k.$$

In the extended SOBM method, the algorithm has to adapt the interval boundaries:

$$\begin{aligned} l_{s_i}(t+1) \;:=\; & l_{s_i}(t) \\ & + g(l_{s_i}(t), r_{s_i}(t), X_i(t))\;\alpha(t)\,\text{neigh}(z_s, z_p, t)\,(X_i(t) - l_{s_i}(t)) \\ & - \alpha(t)\;c(l_{s_i}(t), r_{s_i}(t)) \end{aligned}$$

$$\begin{aligned} r_{s_i}(t+1) \;:=\; & r_{s_i}(t) \\ & + g(-r_{s_i}(t), -l_{s_i}(t), -X_i(t))\;\alpha(t)\,\text{neigh}(z_s, z_p, t)\,(X_i(t) - r_{s_i}(t)) \\ & + \alpha(t)\;c(l_{s_i}(t), r_{s_i}(t)) \end{aligned}$$

with same linear function $g : \mathbf{R}^3 \to [0,1]$, $g(a,b,x) := \begin{cases} 1 & \text{if } x < a \\ 0 & \text{if } x > b \\ \frac{b-x}{b-a} & \text{else.} \end{cases}$

and a function $c : \mathbf{R}^2 \to \mathbf{R}_0^+$ that is independent of the input $X(t)$ and will be defined later.

Note that instead of the above function g, also a smoother "sigmoid" function like $\overline{g}(a,b,x) := 1 - \frac{1}{1+\exp(-x+\frac{a+b}{2})}$ can be chosen in principle.

Suppose for the time being that $c = 0$, then one easily verifies that the left interval boundary is only adapted, if the input is left of the right interval boundary and vice versa. Further one observes that inputs outside the interval have a greater influence on the adaption of the nearest interval boundary, as when they are inside the interval. In the following we will motivate the suggested adaption rule.

Definition 2.2 *We call* $\Theta := \{\Theta_1, \ldots, \Theta_k\}$ *a Voronoi tessellation of* Ω *if*

$$\bigcup_{s=1}^{k} \Theta_s = \Omega \quad \text{and} \quad \Theta_p \cap \Theta_s = \emptyset \quad \text{for all } p, s \in \{1, \ldots, k\}$$

The Θ_s *are called partitions of the Voronoi tessellation.*

It is obvious, that the box codebook vectors $\hat{W}_1, \ldots, \hat{W}_k$ implicitly define a Voronoi tessellation $\hat{\Theta} := \{\hat{\Theta}_1, \ldots, \hat{\Theta}_k\}$, if we set

$$\hat{\Theta}_s := \{X \in \Omega | \ s \text{ is winner neuron for input } X\}.$$

Our goal is, to compute $\hat{W}_s$ that are good box approximations of the corresponding $\hat{\Theta}_s$ with respect to ρ.

Definition 2.3 *Let* Δ *be a box in* Ω *and* Υ *an arbitrary non-void subset of* $\mathbf{R}^q$. *Then* Δ *is called a box approximation of* Υ *with respect to* ρ, *if* $P_\rho(\Delta \setminus \Upsilon) = \int_{\omega \in \Delta \setminus \Upsilon} \rho(\omega)\, d\omega = 0$.
The value $P_\rho(\Delta \cap \Upsilon) = P_\rho(\Delta) - P_\rho(\Delta \setminus \Upsilon) = P_\rho(\Delta) = \int_{\omega \in \Delta} \rho(\omega)\, d\omega$ *indicates the quality of the approximation.*

One easily verifies, that after the initialization we have $\hat{W}_s \subset \hat{\Theta}_s$ for all $s = 1, \ldots, k$. Suppose now an input X that belongs to $\hat{\Theta}_s$. If $X_i \notin \hat{W}_{s_i}$, we have to widen the interval. Therefore the nearest interval boundary is "pulled" towards X_i. This is just the same method as in the original SOM algorithm. If $X_i \in \hat{W}_{s_i}$ the first strategy is to "do nothing", because in this case the interval seems to be all right. This however, turns out to be not a good idea, because the $\hat{\Theta}_s$ change over time, so that we can observe $\hat{W}_s \setminus \hat{\Theta}_s \neq \emptyset$ after several adaption steps. If this difference becomes larger, it is not only possible that $P_\rho(\hat{W}_s \setminus \hat{\Theta}_s) > 0$, so that $\hat{W}_s$ is no longer a box approximation of $\hat{\Theta}_s$. Also the probability grows, that one observes overlaps between the boxes after the algorithm stops, so that after few steps the boxes will represent a poor partitioning of Ω, too (see Figure 1).

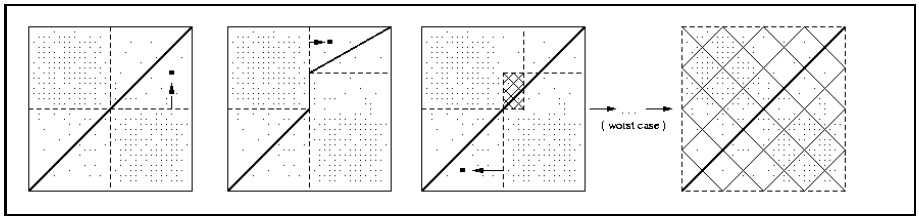

Figure 1: **Poor partitioning in the absence of interval shrinkage.**

If, however the boxes represent a poor partitioning of Ω, then the identification of the essential dimensions (see Section 3) becomes impossible or at least much more complicated. Therefore it is necessary to shrink the intervals. This could be done by adapting the interval boundaries when even the input X_i is inside the interval (so called interior adaption). It is obvious that the adaption of the nearest boundary should be greater than that of the opposite side. By doing this a new problem arises: Usually after some time there are more inputs X_i inside the interval than outside (remember that $X \in \hat{\Theta}_s$). As a consequence, the interval shrinks faster than it grows, which implies that the value $P_\rho(\hat{W}_s)$ shrinks, too. But then the approximation of the tessellation $\hat{\Theta}_s$ is not as good as it could be. Therefore one has to introduce something like a damping coefficient or a correction term, which reduces the inter-interval adaption. Such a parameter will depend on the ratio of the inputs inside and outside the interval. A direct computation would be impracticable, because it is very time consuming. So one has to think about certain heuristics, which only consider the interval width. Our approaches with a damping coefficient, appeared to supply unsatisfactory results. Excellent results were obtained by another approach, which uses an analytically derived correction term. This approach will be presented subsequently.

Correction term Without loss of generality, suppose that there are $a_i, b_i \in \mathbf{R}$, such that $\Omega_\rho := \{X \in \Omega |\, \rho(X) > 0\} \subset \bigotimes_{i=1}^{q} [a_i, b_i]$. Let $\hat{\Theta}_s(t)$ be the Voronoi partition that is defined via $\hat{W}_s(t)$ and let $\Delta_s(t) := \bigotimes_{i=1}^{q} [l^*_{s_i}(t), r^*_{s_i}(t)]$ be an optimal box approximation of $\hat{\Theta}_s(t)$ with minimal volume, i.e., $\mathrm{boxvol}(\Delta_s(t)) = \min\{\mathrm{boxvol}(\overline{\Delta}) \,|\, \overline{\Delta} \text{ optimal box approximation of } \hat{\Theta}_s(t)\}$.

Definition 2.4 *We call a box approximation Δ of Υ with respect to ρ optimal, if $P_\rho(\Delta) = \max\{P_\rho(\overline{\Delta}) |\, \overline{\Delta}$ arbitrary box approximation of Υ with respect to $\rho\}$.*

For our further expositions we need the following definition:

Definition 2.5 *For $\Upsilon \subset \Omega$ with $P_\rho(\Upsilon) > 0$, the conditional probability density function ρ_Υ on Υ is defined as*

$$\rho_\Upsilon(\omega) := \begin{cases} \frac{\rho(\omega)}{P_\rho(\Upsilon)} & \text{if } \omega \in \Upsilon \\ 0 & \text{else} \end{cases}$$

Using $\rho_{\hat{\Theta}_s(t)}$, we can compute the conditional expectation value $E(\hat{W}_s(t+1))$ for each actual codebook vector $\hat{W}_s(t)$, that gives the expectation value of $\hat{W}_s(t+1)$ under the condition that s is the winner neuron (note that this implicitly ensures $P_\rho(\hat{\Theta}_s(t)) > 0$).

We have $E(\hat{W}_s(t+1)) = \bigotimes_{i=1}^{q} [E(l_{s_i}(t+1)), E(r_{s_i}(t+1))]$ with

$$E(l_{s_i}(t+1)) := \int_{\Omega_\rho} l_{s_i}(t+1) \rho_{\hat{\Theta}_s(t)}(X)\, dX = \int_{a_i}^{b_i} l_{s_i}(t+1) \rho_{\hat{\Theta}_s(t),i}(X_i)\, dX_i\,,$$

$$E(r_{s_i}(t+1)) := \int_{\Omega_\rho} r_{s_i}(t+1) \rho_{\hat{\Theta}_s(t)}(X)\, dX = \int_{a_i}^{b_i} r_{s_i}(t+1) \rho_{\hat{\Theta}_s(t),i}(X_i)\, dX_i$$

and

$$\rho_{\hat{\Theta}_s(t),i}(X_i) := \int_{a_1}^{b_1} \dots \int_{a_{i-1}}^{b_{i-1}} \int_{a_{i+1}}^{b_{i+1}} \dots \int_{a_k}^{b_k} \rho_{\hat{\Theta}_s(t)}(X_1, \dots, X_k)\, dX_1 \dots dX_{i-1}\, dX_{i+1} \dots dX_k\,.$$

Upon considering our above adaption rule we obtain:

$$\begin{aligned} E(l_{s_i}(t+1)) &= l_{s_i}(t) \\ &\quad + \int_{a_i}^{l_{s_i}(t)} \alpha(t)(X_i - l_{s_i}(t))\rho_{\hat{\Theta}_s(t),i}(X_i)\, dX_i \\ &\quad + \int_{l_{s_i}(t)}^{r_{s_i}(t)} \frac{(r_{s_i}(t) - X_i)}{(r_{s_i}(t) - l_{s_i}(t))}\alpha(t)(X_i - l_{s_i}(t))\rho_{\hat{\Theta}_s(t),i}(X_i)\, dX_i \\ &\quad - \alpha(t)\, c(l_{s_i}(t), r_{s_i}(t)) \end{aligned}$$

and

$$\begin{aligned} E(r_{s_i}(t+1)) &= r_{s_i}(t) \\ &\quad + \int_{r_{s_i}(t)}^{b_i} \alpha(t)(X_i - r_{s_i}(t))\rho_{\hat{\Theta}_s(t),i}(X_i)\, dX_i \\ &\quad + \int_{l_{s_i}(t)}^{r_{s_i}(t)} \frac{(X_i - l_{s_i}(t))}{(r_{s_i}(t) - l_{s_i}(t))}\alpha(t)(X_i - r_{s_i}(t))\rho_{\hat{\Theta}_s(t),i}(X_i)\, dX_i \\ &\quad + \alpha(t)\, c(l_{s_i}(t), r_{s_i}(t))\,. \end{aligned}$$

Since $\Delta_s(t)$ is an optimal box approximation of $\hat{\Theta}_s(t)$, we may assume that

$$P_{\rho_{\hat{\Theta}_s(t)}}(\Delta_s(t)) = \int_{\omega \in \Delta_s(t)} \rho_{\hat{\Theta}_s(t)}(\omega)\, d\omega \approx 1.$$

Therefore, for simplicity, we suppose that the i-th components X_i of the inputs $X \in \hat{\Theta}_s(t)$ are uniformly distributed over $[l_i^*(t), r_i^*(t)]$, such that

$$\rho_{\hat{\Theta}_s(t),i}(X_i) := \begin{cases} \frac{1}{r_i^*(t) - l_i^*(t)} & \text{if } X = (X_1, \dots, X_k) \in \Delta_s \\ 0 & \text{else}\,. \end{cases}$$

Hence, we arrive at

$$\begin{aligned} E(r_{s_i}(t+1)) &= r_{s_i}(t) \\ &\quad + \int_{r_{s_i}(t)}^{r_i^*(t)} \frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))}(X_i - r_{s_i}(t))\, dX_i \\ &\quad + \int_{l_{s_i}(t)}^{r_{s_i}(t)} \frac{(X_i - l_{s_i}(t))}{(r_{s_i}(t) - l_{s_i}(t))}\frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))}(X_i - r_{s_i}(t))\, dX_i \\ &\quad + \alpha(t)\, c(l_{s_i}(t), r_{s_i}(t)) \end{aligned}$$

$$
\begin{aligned}
= \quad & r_{s_i}(t) \\
& + \frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))} \frac{(r_i^*(t) - r_{s_i}(t))^2}{2} \\
& + \frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))} \int_{l_{s_i}(t)}^{r_{s_i}(t)} \frac{(X_i - l_{s_i}(t))(X_i - r_{s_i}(t))}{(r_{s_i}(t) - l_{s_i}(t))} \, dX_i \\
& + \alpha(t)\, c(l_{s_i}(t), r_{s_i}(t)) \\
\\
= \quad & r_{s_i}(t) \; + \; \frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))} \frac{(r_i^*(t) - r_{s_i}(t))^2}{2} \\
& \qquad - \frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))} \frac{(r_{s_i}(t) - l_{s_i}(t))^2}{6} \; + \; \alpha(t)\, c(l_{s_i}(t), r_{s_i}(t)) \, .
\end{aligned}
$$

For the left hand boundary, we analogously obtain:

$$
\begin{aligned}
E(l_{s_i}(t+1)) \quad = \quad & l_{s_i} \; - \; \frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))} \frac{(l_{s_i}(t) - l_i^*(t))^2}{2} \\
& + \frac{\alpha(t)}{(r_i^*(t) - l_i^*(t))} \frac{(r_{s_i}(t) - l_{s_i}(t))^2}{6} \; - \; \alpha(t)\, c(l_{s_i}(t), r_{s_i}(t)) \, .
\end{aligned}
$$

By means of the intuitive choice

$$
c(l_{s_i}(t), r_{s_i}(t)) \; := \; \frac{1}{6} (r_{s_i}(t) - l_{s_i}(t)) \tag{3}
$$

we end up with

$$
\begin{aligned}
E(l_{s_i}(t+1)) \quad &= \quad l_{s_i} \; - \; \frac{1}{2}\alpha(t) \frac{(l_{s_i}(t) - l_i^*(t))^2}{(r_i^*(t) - l_i^*(t))} \; - \; \alpha(t)\, (1 - \vartheta_{s_i}(t))\, c(l_{s_i}(t), r_{s_i}(t)) \\
E(r_{s_i}(t+1)) \quad &= \quad r_{s_i} \; + \; \frac{1}{2}\alpha(t) \frac{(r_i^*(t) - (r_{s_i}(t))^2}{(r_i^*(t) - l_i^*(t))} \; + \; \alpha(t)\, (1 - \vartheta_{s_i}(t))\, c(l_{s_i}(t), r_{s_i}(t))
\end{aligned}
$$

in terms of some model quantity

$$
\vartheta_{s_i}(t) := \frac{(r_{s_i}(t) - l_{s_i}(t))}{(r_i^*(t) - l_i^*(t))} \, .
$$

This quantity measures the deviation of the actual interval width from the optimal one.

In the following, we have to assure that the intervals are always well defined, i.e., we always have $l_{s_i}(t) < r_{s_i}(t)$ for all $t \in \{0, \ldots, T\}$.

Lemma 2.6

$$
l_{s_i}(t) < r_{s_i}(t) \quad \Longrightarrow \quad l_{s_i}(t+1) < r_{s_i}(t+1) \quad (\forall\, t \in \{0, \ldots, T\})
$$

Proof: Let p be the winner neuron for input X(t). Then one easily verifies:

$$
\begin{aligned}
(1) \qquad & X_i(t) < l_{s_i}(t) \qquad \Longrightarrow \\
& r_{s_i}(t+1) - l_{s_i}(t+1) \;=\; \left(1 + \frac{\alpha(t)}{3}\right)(r_{s_i}(t) - l_{s_i}(t)) \\
& \qquad - \underbrace{\underbrace{\alpha(t)}_{\geq 0}\,\underbrace{\mathrm{neigh}(z_s, z_p, t)}_{\geq 0}\,\underbrace{(X_i(t) - l_{s_i}(t))}_{<0}}_{\leq 0} \\
& \qquad \geq \; r_{s_i}(t) - l_{s_i}(t)
\end{aligned}
$$

$$
\begin{aligned}
(2) \qquad & X_i(t) > r_{s_i}(t) \qquad \Longrightarrow \\
& r_{s_i}(t+1) - l_{s_i}(t+1) \;=\; \left(1 + \frac{\alpha(t)}{3}\right)(r_{s_i}(t) - l_{s_i}(t)) \\
& \qquad + \underbrace{\alpha(t)\,\mathrm{neigh}(z_s, z_p, t)\,(X_i(t) - r_{s_i}(t))}_{\geq 0} \\
& \qquad \geq \; r_{s_i}(t) - l_{s_i}(t)
\end{aligned}
$$

$$
\begin{aligned}
(3) \quad & X_i(t) \in [l_{s_i}(t), r_{s_i}(t)] \quad \Longrightarrow \\
& r_{s_i}(t+1) - l_{s_i}(t+1) \;=\; \left(1 + \frac{\alpha(t)}{3}\right)(r_{s_i}(t) - l_{s_i}(t)) \\
& \qquad + \alpha(t)\,\mathrm{neigh}(z_s, z_p, t)\,\frac{(X_i(t) - l_{s_i}(t))}{(r_{s_i}(t) - l_{s_i}(t))}(X_i(t) - r_{s_i}(t)) \\
& \qquad - \alpha(t)\,\mathrm{neigh}(z_s, z_p, t)\,\frac{(r_{s_i}(t) - X_i(t))}{(r_{s_i}(t) - l_{s_i}(t))}(X_i(t) - l_{s_i}(t)) \\
& \qquad = \; \left(1 + \frac{\alpha(t)}{3}\right)(r_{s_i}(t) - l_{s_i}(t)) \\
& \qquad - 2\,\alpha(t)\,\underbrace{\mathrm{neigh}(z_s, z_p, t)}_{\leq 1}\,\underbrace{\frac{(r_{s_i}(t) - X_i(t))(X_i(t) - l_{s_i}(t))}{(r_{s_i}(t) - l_{s_i}(t))}}_{\leq \frac{1}{4}(r_{s_i}(t) - l_{s_i}(t))\ (*)} \\
& \qquad \geq \; \left(1 + \frac{\alpha(t)}{3} - \frac{\alpha(t)}{2}\right)(r_{s_i}(t) - l_{s_i}(t)) \\
& \qquad = \; \left(1 - \frac{\alpha(t)}{6}\right)(r_{s_i}(t) - l_{s_i}(t))
\end{aligned}
$$

$$
(*) \qquad \max_{l \leq x \leq r} (r - x)(x - l) \;=\; \frac{1}{4}(r - l)^2 \quad \text{for all } l, r \in \mathbf{R}
$$

Because $\alpha(t) \leq 1$ for all $t \in \{0, \ldots, T\}$, we have in all three cases:

$$
(r_{s_i}(t) - l_{s_i}(t)) > 0 \quad \Longrightarrow \quad (r_{s_i}(t+1) - l_{s_i}(t+1)) > 0.
$$

□

Note that Lemma (2.6) is not true, if $\alpha(t) \geq 6$.

Hence if $l_{s_i}(0) < r_{s_i}(0)$, Lemma (2.6) guarantees that $c(l_{s_i}(t), r_{s_i}(t)) > 0$ and $\vartheta_{s_i}(t) > 0$ for all $t \in \{0, \ldots, T\}$.

Therefore we obtain

$$\begin{aligned} \hat{W}_{s_i}(t) \subset [l_i^*(t), r_i^*(t)] &\implies \vartheta_{s_i}(t) \in]0, 1] \\ &\implies E(l_{s_i}(t+1)) < l_{s_i}(t) \text{ and } E(r_{s_i}(t+1)) > r_{s_i}(t) \end{aligned}$$

and

$$\hat{W}_{s_i}(t) = [l_i^*(t), r_i^*(t)] \implies E(l_{s_i}(t+1)) = l_{s_i}(t) \text{ and } E(r_{s_i}(t+1)) = r_{s_i}(t).$$

Now if we choose $\hat{W}_s(0) \in \Delta_s(0)$ we can be confident, that $\vartheta_{s_i}(T) \approx 1$ and therefore $\hat{W}_{s_i}(T) \approx [l_i^*(T), r_i^*(T)]$, whenever we use our extended algorithm with T time steps and T large enough. This means that $\hat{W}_s(T) \approx \Delta_s(T)$ and therefore $\hat{W}_s(T)$ is nearly an optimal box approximation of $\hat{\Theta}_s(T)$ with respect to ρ. Obviously the chosen function c is a suitable correction term for the interval shrinkage.

Comparison SOM - SOBM Upon comparing maps M and $\hat{M}$, computed by the original SOM and the extended SOBM algorithm with the same parameters and initialization, one will observe clear similarities. In most cases the orientation of the maps and the identifiable clusters are equal (for details about cluster analysis by SOMs see Section 4 or [6], respectively). Also for each point codebook vector W_p of M one can usually find an box codebook vector $\hat{W}_s$ of $\hat{M}$ with $W_p \in \hat{W}_s$. Therefore the extended algorithm will be at least as powerful as the classical algorithm, which is of utmost significance for applications. In the following, however, we will show that the new SOBM method has important advantages:

To see this, we suppose, for simplicity, that we have only an one-dimensional input space $\Omega = \mathbf{R}$. We want to compute a 2×1 map with neurons s and $\overline{s}$, the Euclidean distance function and $\text{neigh}(z_s, z_{\overline{s}}, t) = 0$ for $t \in \{0, \ldots, T\}$.

For the purpose of illustration, we define two probability density functions ρ_1 and ρ_2 (see Figure 2):

$$\rho_1(X) := \begin{cases} 2.5 & \text{if } X \in [0.8,\ 1] \\ 0.5 & \text{if } X \in [-1,\ 0] \\ 0 & \text{else} \end{cases}$$

$$\rho_2(X) := \begin{cases} 2.5 & \text{if } X \in [0.8,\ 1] \\ 0.625 & \text{if } X \in [-1,\ -0.6] \\ 0.625 & \text{if } X \in [-0.4,\ 0] \\ 0 & \text{else.} \end{cases}$$

We have used the original SOM algorithm and our extended algorithm with $c = 0$ and c as defined in Eq. (3) to compute the codebooks for ρ_1 and ρ_2. Table 1 shows the results (random codebook initialization, $\alpha(0) = 0.9$, $T = 10000$).

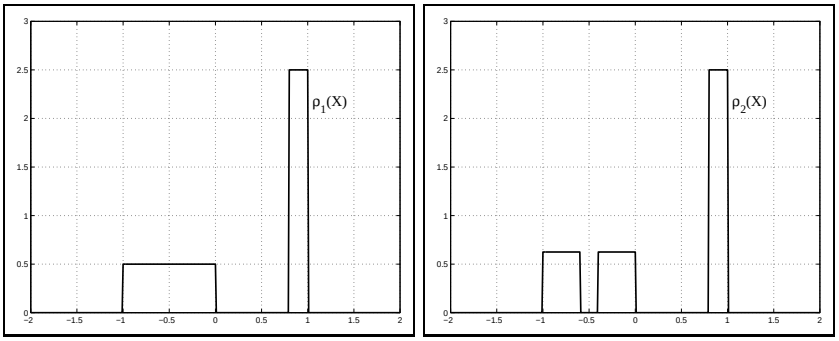

Figure 2: **Probability density functions ρ_1 and ρ_2**

	ρ_1
orig. SOM	$W_s = -0.5,\ W_{\overline{s}} = 0.9$
ext. SOM($c = 0$)	$\hat{W}_s = [-0.75, -0.25],\ \hat{W}_{\overline{s}} = [0.85, 0.95]$
ext. SOM	$\hat{W}_s = [-1.00, 0.00],\ \hat{W}_{\overline{s}} = [0.80, 1.00]$
	ρ_2
orig. SOM	$W_s = -0.5, W_{\overline{s}} = 0.9$
ext. SOM($c = 0$)	$\hat{W}_s = [-0.78, -0.22],\ \hat{W}_{\overline{s}} = [0.85, 0.95]$
ext. SOM	$\hat{W}_s = [-1.05, 0.07], \hat{W}_{\overline{s}} = [0.80, 1.01]$

Table 1: **Codebook vectors for ρ_1 and ρ_2**

Obviously, the following three observations are of interest:

- The probability density function ρ_1 is positive on $[-1, 0]$ and $[0.8, 1]$. Although this intervals are of different width, we get no hint about this fact, if we look at the point codebook vectors W_s and $W_{\overline{s}}$.
- The box codebook vectors are box approximations of the Voronoi partitions, which they implicitly define. The quality of this approximations is optimal if we use the correction term c as defined in Eq. 3.
- The point codebooks are equal for both probability density functions, i.e., although ρ_1 and ρ_2 are different, we can not distinguish them by looking at the point codebook vectors. The situation is quite different if we use the correction term c and look at the box codebook vectors. Here we see that the interval width of $\hat{W}_s$ in the case of ρ_2 is larger then in the case of ρ_1. If we look deeper, we see that the difference is approximately the width of the hole between -0.4 and -0.6 of ρ_2. This is not surprising, because the correction terms for $\hat{W}_s$ are equal in both cases, but the power of the interval shrinkage for $\hat{W}_s$ is lower in the case of ρ_2. Therefore the interval $\hat{W}_s$ can grow stronger in this case. Although we can not derive the differences between ρ_1 and ρ_2 from looking at the different $\hat{W}_s$, we at least get a hint that there are differences.

We have made similar observations for higher-dimensional input spaces and larger maps.

Additionally we want to show an intriguing feature of our extended algorithm. Look at he following probability density functions ρ_3:

$$\rho_3(X) := \frac{1}{2\sigma\sqrt{2\pi}} \left(\exp(-\frac{1}{2} \left(\frac{(X-\mu_1)}{\sigma} \right)^2 + \exp(-\frac{1}{2} \left(\frac{(X-\mu_2)}{\sigma} \right)^2 \right).$$

One observes that $\hat{W}_s \approx [\mu_1 - \sigma, \mu_1 + \sigma]$ and $\hat{W}_{\overline{s}} \approx [\mu_2 - \sigma, \mu_2 + \sigma]$. The approximation is the better, the larger the difference is between μ_1 and μ_2. Figure 3 shows ρ_3 with $\mu_1 = -0.5$, $\mu_2 = 0.5$ and $\sigma = 0.27$ and Table 2 gives the corresponding computational results.

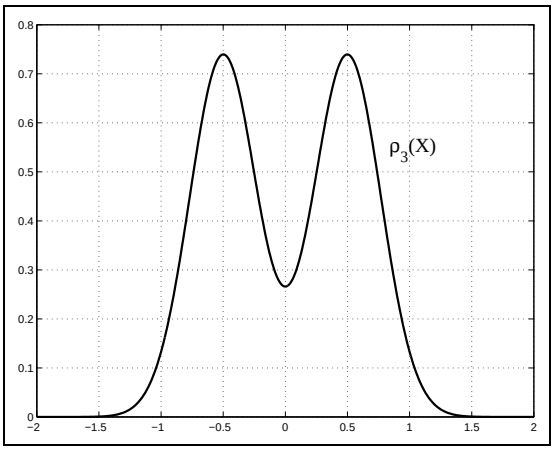

Figure 3: **Probability density functions ρ_3**

	ρ_3
orig. SOM	$W_s = -0.5$, $W_{\overline{s}} = 0.5$
ext. SOM($c = 0$)	$\hat{W}_s = [-0.67, -0.33]$, $\hat{W}_{\overline{s}} = [0.31, 0.67]$
ext. SOM	$\hat{W}_s = [-0.85, -0.15]$, $\hat{W}_{\overline{s}} = [0.13, 0.86]$

Table 2: **Codebook vectors for ρ_3**

Although the concept of box codebooks develops its full power still within the identification of essential dimensions and the multilevel cluster analysis, to be described in the subsequent sections the advantages in comparison with point codebooks are already obvious.

A disadvantage of our extended SOM algorithm is that it requires more computing time than the original algorithm. The difference depends strongly on the chosen implementation, but because the number of variables that have to be adapted and to be evaluated are doubled, in the worst case our extended algorithm doubles the computing time of the original algorithm.

Combination of SOM and SOBM To speed up the computing time, one may think about a combination of the original SOM and the extended SOBM algorithm. In the following we suggest such a combination, which has turned out to be quiet powerful in our first applications.

As usual in the original SOM algorithm, we first compute in T_1 steps the point codebook vectors $W_1, \ldots, W_k$ with a large learning rate $\alpha(0) \approx 0.9$ at the beginning and with neighborhood adaption, i.e., $\text{neigh}(z_s, z_p, t) > 0$ for $t < T_1$. Since we are mainly interested in the ordering of our map and not in the convergence of the W_s, we choose T_1 rather low. This is often called the *ordering phase* of the SOM algorithm.

After this phase one usually passes on to another adaption cycle with a larger T_2, a low learning rate α and no neighborhood adaption, i.e., $\text{neigh}(z_s, z_p, t) = 0$ for $s \neq p$ and $t \in [T_1, T_2]$. After this so called *convergence phase* of the SOM algorithm, the codebook vectors are rather stable and good representatives of the input space and the used probability distribution.

In our combined approach, we use the extended SOM algorithm within the convergence phase: We first initialize the box codebook vectors $\hat{W}_i(0)$ by using the earlier computed point codebook vectors $W_1(T_1)$ within the described initialization routine. Then we adapt the box codebook vectors in T_2 time steps with a low learning rate and no neighborhood adaption.

Summarizing, as a result of this combination — original SOM algorithm within the ordering phase, SOBM algorithm within the convergence phase — we obtain not only a shorter computing time, but also avoid possible effects of the neighborhood adaption on the generation of the box codebook vectors.

3 Identification of essential dimensions

Suppose we have an input space Ω with probability density function ρ. Let P_{uni} denote the uniform probability distribution over Ω and define $|\Upsilon| := P_{uni}(\Upsilon)$ for $\Upsilon \subset \Omega$. If Ω is high-dimensional, one often observes that

$$\frac{|\Omega_\rho|}{|\Omega|} \ll 1\,,$$

i.e, the input space is sparse with respect to the probability distribution P_ρ.

Definition 3.1 *Let $\Omega \subset \mathbf{R}^q$ be an input space with probability density function ρ, $\Theta^* := \{\Theta_1^*, \ldots, \Theta_k^*\}$ a Voronoi tessellation of Ω and $\text{dist} : \Omega \times \Omega \to \mathbf{R}$ a distance function on Ω. Then we call Θ^* an optimal k-s-partitioning of Ω_ρ with respect to dist, if*

$$\sum_{X,Y \in \Theta_s^*} \rho(X)\rho(Y)\text{dist}(X,Y) + \sum_{X,Y \in \Omega \setminus \Theta_s^*} \rho(X)\rho(Y)\text{dist}(X,Y) =$$

$$\min_{\Theta}\Big(\sum_{X,Y \in \Theta_s} \rho(X)\rho(Y)\text{dist}(X,Y) + \sum_{X,Y \in \Omega \setminus \Theta_s} \rho(X)\rho(Y)\text{dist}(X,Y)\Big),$$

$$(\Theta := \{\Theta_1, \ldots, \Theta_k\} \text{ Voronoi tessellation of } \Omega)$$

If Θ^* is an optimal k-s-partitioning of Ω_ρ for all $s \in \{1,\ldots,k\}$, we call it an optimal k-partitioning of Ω_ρ with respect to dist.

By construction of the SOM algorithm, we can assume that the Voronoi tessellation, that is indirectly defined by the generated k codebook vectors, is a nearly optimal k-partitioning of Ω_ρ.

Suppose now that we have an optimal or nearly optimal k-partitioning of Ω_ρ. Then one observes, that often only a small number of dimensions — the so called essential dimensions — of Ω_ρ are necessary to discriminate the k partitions, if we only consider the points $X \in \Omega_\rho$.

Definition 3.2 *Let $\Omega \subset \mathbf{R}^q$ be an input space with probability density function ρ, $\mathrm{dist} : \Omega \times \Omega \to \mathbf{R}$ a distance function on Ω and $\Theta^* := \{\Theta_1^*,\ldots,\Theta_k^*\}$ an optimal k-partitioning of Ω_ρ with respect to dist. Dimension $i \in \{1,\ldots,q\}$ is called a non essential dimension for Θ_s^* in Ω_ρ , if Θ^* is an optimal k-s-partitioning of Ω_ρ with respect to $\mathrm{dist}_{*i} : \Omega \times \Omega \to \mathbf{R}$. The function dist_{*i} is defined as*

$$dist_{*i} := f(d_1,\ldots,d_{i-1},0,d_{i+1},\ldots,d_q).$$

Otherwise dimension i is called an essential dimension for Θ_s^* in Ω_ρ. If the dimension i is an essential dimension for Θ_s^* in Ω_ρ for all $s \in \{1,\ldots,k\}$, it is called an essential dimension for Θ^* in Ω_ρ.

Definition 3.3 *Let $\Omega \subset \mathbf{R}^q$ be an input space with probability density function ρ, $\mathrm{dist} : \Omega \times \Omega \to \mathbf{R}$ a distance function on Ω and $k \in \mathbf{N}, k \geq 2$. Then dimension $i \in \{1,\ldots,q\}$ is called an essential dimension in Ω_ρ of degree k, if there exists an optimal k-partitioning Θ^* of Ω_ρ with respect to dist, such that i is an essential dimension for Θ^* in Ω_ρ.*

In the following we describe a method for the identification of the non essential dimensions—and therefore also the essential dimensions—in Ω_ρ of degree k by using box codebooks.

Suppose, we computed a 1×1 SOBM for a probability density function ρ on Ω, by our extended algorithm. Let $\hat{W} = \bigotimes_{i=1}^q [l_i, r_i]$ be the generated box codebook vector. By construction of the SOBM algorithm, we have $P_\rho(\hat{W}) > 0$, so that the conditional probability density function $\rho_{\hat{W}}$ on $\hat{W}$ is well-defined. Based on $\rho_{\hat{W}}$ we use the extended or the combined algorithm, to generate an $m \times n$ SOBM. Let $\hat{W}_1,\ldots,\hat{W}_k$ be the resulting $k = mn$ box codebook vectors. Now we compute for each codebook vector $\hat{W}_s = \bigotimes_{i=1}^q [l_{s_i}, r_{s_i}]$ and for each dimension i an overlap measure η_i between the intervals $[l_{s_i}, r_{s_i}]$ and $[l_i, r_i]$:

$$\eta_i(\hat{W},\hat{W}_s) := \frac{1}{(r_i - l_i)} \int_{l_i}^{r_i} \chi_{[l_{s_i},r_{s_i}]}(\omega_i) d\omega_i$$

with characteristic function

$$\chi_M(x) := \begin{cases} 1 & \text{if } x \in M \\ 0 & \text{else} \end{cases}$$

If we choose a small $\nu > 0$ we have

$$\begin{aligned} \eta_i(\hat{W}, \hat{W}_s) > 1 - \nu \quad &\Longrightarrow \quad [l_{s_i}, r_{s_i}] \approx [l_i, r_i] \\ &\Longrightarrow \quad (\forall X \in \hat{W})\ \mathrm{DIST}(X, \hat{W}_s) \approx \mathrm{DIST}_{*i}(X, \hat{W}_s) \end{aligned}$$

with

$$\mathrm{DIST}_{*i} := f(\hat{d}_1, \ldots, \hat{d}_{i-1}, 0, \hat{d}_{i+1}, \ldots, \hat{d}_q).$$

This means that, if we have $\eta_i(\hat{W}, \hat{W}_s) > 1 - \nu$ for a small $\nu > 0$, the dimension i is not relevant for the discrimination between $\hat{W}_s$ and $\hat{W} \setminus \hat{W}_s$. In this case the dimension i is said to be *not essential* for box $\hat{W}_s$. If a dimension i is not essential for all boxes $\hat{W}_s$, we say it is not essential for $\hat{W}$.

Suppose now that dimension i is not essential for $\hat{W}$. If $\hat{W}$ is considered to be a nearly optimal box approximation of Ω_ρ and the $\hat{W}_s$ are nearly optimal box approximations of the indirectly defined $\hat{\Theta}_s$ with respect to ρ, then we may write:

$$(\forall X, Y \in \hat{\Theta}_s)\ \rho(X)\rho(Y)\mathrm{dist}(X, Y) \approx \rho(X)\rho(Y)\mathrm{dist}_{*i}(X, Y)\ (\forall s \in \{1, \ldots, k\}).$$

Since now $\hat{\Theta} := \{\hat{\Theta}_1, \ldots, \hat{\Theta}_k\}$ is a nearly optimal k-partitioning with respect to dist, it is also a nearly optimal k-partitioning with respect to dist_{*i}. Therefore the dimension i is not an essential dimension for $\hat{\Theta}$ in Ω_ρ.

4 Adaptive hierarchical cluster analysis

In this section we suggest a new adaptive multilevel approach for cluster analysis, that uses Self-Organizing Maps with box codebooks and the concept of essential dimensions.

Usually one distinguishes between partitioning and hierarchical cluster methods [2]. The partitioning methods, as the well known *k-means-algorithm*, aim at finding a good clustering by optimizing special partition functions. To do so, they need the — a priori unknown — number of clusters as an input. Trying different numbers is no real way out of this dilemma, because there exists no automatic criterion to decide, which is the correct number of clusters. Therefore the resulting clusterings have to be assessed by expert examination. In contrast to that, the known hierarchical methods do not need the number of clusters as input. Rather, they produce a hierarchical tree of possible clusterings, called dendrogram, and leave the problem of deciding on which tree level the clustering is best to a human expert. Although it is possible to think about combinations of both approaches, last but not least, one still remains with a non automatic system.

Self-Organizing Maps can also be used for cluster analysis: After computing the map one tries to find a clustering of the codebook vectors. Thereby one gets indirectly a clustering of the whole input space, too. Although it is possible to use a partitioning or hierarchical cluster algorithm for this task, this is not

the usual way, because in this case it is better to use them directly. Instead one traditionally uses methods, like the u-matrix method [1], that visualize the differences of the codebook vectors on the map. Together with additional visualization tools [5] they allow the human expert to identify clusters on the map and therefore in the whole input space. Although these cluster methods do neither need the number of clusters as an input, nor require direct comparisons between different clusterings, there is a big disadvantage: These methods do not allow an automated cluster identification and have an undesirable flavor of personal bias. To avoid this disadvantage, we recently suggested an approach [6], that automatically identifies clusters on the map by using Perron eigenmode analysis [3]. Although this method works already quite well, one problem still remains unsolved: the clustering depends on the chosen size of the map. This connection is not as strong as between the clustering and the chosen number of clusters in the case of partitioning methods, but also affects the cluster quality.

One possible remedy for this problem is to refine clusters, i.e. to select the objects of a cluster and then to cluster them again. By doing this we arrive at a *multilevel clustering* procedure. As already mentioned above, we are then still left with the question at which cluster level to stop. Of course, stopping at the optimal cluster level is aimed at, i.e., when no further "improvement" of the clustering is expected.

For an automatic stopping criterion, we suggest to exploit the concept of essential dimensions together with the following definition:

Definition 4.1 *Let* $\Theta^* := \{\Theta_1^*, \ldots, \Theta_k^*\}$ *be an optimal* k*-partitioning of* Ω_ρ *and* $I \subset \{1, \ldots, k\}$ *a neuron set that defines a cluster* $\Lambda := \bigcup_{i \in I} \Theta_i^*$ *in* Ω *with* $P_\rho(\Lambda_I) > 0$*. The cluster level of* Λ *is called* ι*-optimal, if there exist less than* ι *essential dimensions in* Ω_{ρ_Λ} *of degree* k*.*

At this stage, we are now ready to formulate our *multilevel cluster algorithm*:

Suppose an input space Ω with a probability density function ρ is given.

1. Set $\Lambda := \Omega$.
2. Compute a 1×1 SOM with box codebook vector $\hat{W}$ based on ρ_Λ.
3. Compute an $n \times m$ SOM with k box codebook vectors $\hat{W}_1, \ldots, \hat{W}_k$ based on $\rho_{\Lambda \cap \hat{W}}$.
4. Identify the essential dimensions in Ω_{ρ_Λ} of degree k as described in Section 3. *If* the cluster level of Λ is ι-optimal *then* stop, *else* go to the next step.
5. Identify the cluster $\Lambda_1, \Lambda_2, \ldots$ in Ω_{ρ_Λ} that are based on $\hat{W}_1, \ldots, \hat{W}_k$, e.g., by using our automatic cluster identification algorithm (for details see [6]).
6. Choose a cluster Λ_j. *If* $P_{\rho_\Lambda}(\Lambda_j) = 0$, *then* stop, *else* set $\Lambda := \Lambda_j$ and go to step 2. *Repeat* this for all identified clusters.

In our experiments done so far, the choice of the map size k is turned out to be not very critical. For small k the algorithm needs more refinement steps, but the computation of the maps was faster and vice versa. Therefore we recommend to choose k between 50 and 100.

For the dimension threshold level ι, one usually should choose $\iota = 1$. But to speed up the algorithm, one also can choose a higher ι. If Ω is high-dimensional, then this should not cause any deterioration of the clustering quality.

5 Conclusion

The paper presents an extension of the original SOM algorithm, the SOBM algorithm. It can be used to identify *essential dimensions* of a high-dimensional input space. Based on this concept a first multilevel cluster algorithm is described, that can be used for high quality and fully automated cluster analysis. Applications of this algorithm to molecular dynamics will be described in detail in a forthcoming paper.

Acknowledgments. The first author (T.G.) was partially supported by RISK–CONSULTING, PROF. DR. WEYER, KÖLN.

References

[1] A.Ultsch and D.Korus. Integration of neural networks with knowledge-based systems. In *Proc. IEEE Int.Conf.Neural Networks, Perth*, 1995.

[2] B.D.Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[3] P. Deuflhard, W. Huisinga, A. Fischer, and Ch. Schütte. Identification of almost invariant aggregates in nearly uncoupled Markov chains. Accepted in Lin. Alg. Appl., Available via http://www.zib.de/bib/pub/pw, 1998.

[4] G.Deboeck and T.Kohonen (Eds.). *Visual Explorations in Finance using Self-Organizing-Maps*. Springer, London, 1998.

[5] J.Vesanto. Som-based data visualization methods. *Intelligent Data Analysis*, (3):111–126, 1999.

[6] T.Galliat, W.Huisinga, and P.Deuflhard. Self-organizing maps combined with eigenmode analysis for automated cluster identification. To appear in: Proceedings of the 2nd International ICSC Symposium on Neural Computation, 2000, Berlin. Available via http://www.zib.de/bib/pub/pw, 1999.

[7] T.Kohonen. *Self-Organizing Maps*. Springer, Berlin, 2nd edition, 1997.