Chapter 3

Adaptive Decomposition by Self-Organized Neural Networks

In this chapter we will describe two methods, based on self-organized neural networks [1], that can be used to compute a decomposition $\Theta := \{\Theta_1, \ldots, \Theta_{n_k}\}$ of a data set V with homogeneity function h. The decomposition is adaptive in the sense that the number n_k is chosen automatically — only an upper bound $\mathbb{k} \in \mathbf{N}$ has to be fixed a priori — so that Θ is fine enough to use it within our basic reduction algorithm (see section 2.3). Moreover, the second method that is an recently developed extension of the first one (see [29]), allows to compute non-uniform approximate box decompositions.

Since each decomposition of V is also a kind of clustering of V, the computation of a decomposition with small decomposition error (see Eq. (2.1)) has to be done heuristically in a shorter time than $\mathcal{O}(n^2)$. Otherwise there is no advantage of our basic reduction algorithm in comparison with a direct computation of an optimal k-cluster set of V.

We suppose that $\Omega \subset \mathbf{R}^q$ is a metric space, otherwise we will extend it sufficiently as described in the appendix. Further we assume that there exists a distance function $\mathrm{dist} : \Omega \times \Omega \longrightarrow \mathbf{R}$ so that for all $v, w \in V$ the following local maximum condition holds:

$$\mathrm{dist}(v, w) \approx 0 \implies h(v, w) \approx h_{max}(V) \ . \tag{3.1}$$

Usually, this condition is given for geometric cluster problems and also for many dynamic cluster problems (see the earlier discussion in section 1.2).

[1]For an introduction to neural networks see, e.g., [55]

3.1 Self-Organizing Maps (SOM)

Let V any data set in Ω with frequency function f. The following Lemma describes a way to compute an adaptive decomposition based on a given codebook:

Lemma 3.1.1 *Assume* $\Bbbk \in \mathbf{N}$ *and* $W := \{w_1, \ldots, w_\Bbbk\} \subset \Omega$.
Set $\Theta_W := \{\Theta_{w_1}, \ldots, \Theta_{w_\Bbbk}\}$ *with partitions* $\Theta_{w_p} \subset \Omega$ *so that for all* $v \in \Omega$*:*

$$v \in \Theta_{w_p} \iff p = \min\{s \,|\, \operatorname{dist}(v, w_s) = \min_{i=1,\ldots,\Bbbk} \operatorname{dist}(v, w_i)\}. \tag{3.2}$$

Further set $I := \{p \,|\, \Theta_{w_p}(V) \neq \emptyset, p = 1, \ldots, \Bbbk\}$ *with* $\Theta_{w_p}(V) := \Theta_{w_p} \cap V$. *Then* $\Theta_{W_I}(V) := \{\Theta_{w_p}(V) \,|\, p \in I\}$ *is a decomposition of* V *with* $n_k := |I|$ *partitions.*

Since we have $\operatorname{dist}(v, w) \leq \operatorname{dist}(v, w_s) + dist(w, w_s)$ for all $v, w \in \Theta_{w_s}(V)$, each method that tries to compute a W so that for $v \in \Theta_{w_s}(V)$ the distances $\operatorname{dist}(v, w_s)$ are minimized, can be used to generate a decomposition of V with small decomposition error.

At first, one might think of pure vector quantization (VQ) methods (see [35]). These methods often try to minimize the *distortion value* which is defined as:

$$\frac{1}{f(V)} \sum_{s=1}^{\Bbbk} \sum_{v \in \Theta_{w_s}(V)} \operatorname{dist}(v, w_s) f(v). \tag{3.3}$$

However, they have the tendency to produce codebook vectors that are maximally different, to achieve a more uniform decomposition of V. This might cause problems of so called *pseudo-clusters*, i.e. clusters C with nearly zero frequency value $f(C)$. Therefore it seems better to use a method that tends to gather codebook vectors in some more robust way. Here a powerful method are KOHONEN'S Self-Organizing Maps (SOM). The corresponding algorithm usually produces fast and good solutions even for high-dimensional Ω. It can be easily adapted to the case of cyclic data which will be essential for using it within biomolecular data (see chapter 5). Further it has the feature of topology approximation which avoids the appearance of pseudo-clusters and leads to decompositions that are rather robust under changes of the number $\Bbbk$.

In the following we give a short general description of the SOM method. For an exhaustive presentation see [48].

To be in correspondence with the usual notation in the literature, we suppose that there exists a probability distribution P_ρ on Ω with a probability density function $\rho : \Omega \to \mathbf{R}_0^+$ so that $\rho(v) = \frac{f(v)}{f(V)}$ for $v \in V$. If this is not the case, one has to replace all integral signs by sums and has to use f directly.

Each SOM is formed by a q-dimensional input-layer that is fully connected with the two-dimensional Kohonen layer, which is a neural $m_x \times m_y$ grid G with rectangular or hexagonal topology and $\Bbbk = m_x m_y$ grid neurons. The coordinate tuple of each neuron s on the grid is denoted by $z_s \in G$ and each neuron s is uniquely related to a q-dimensional codebook vector w_s. After a suitable initialization of the codebook vectors, the SOM is trained in L time steps by a repeated presentation of vectors of the q-dimensional input space Ω according to the probability distribution P_ρ. For each presented input vector the SOM computes a so called winner neuron and its neighboring neurons on the grid and adapts the related codebook vectors so that the distance to the input vector is reduced. To achieve convergence, the learning rate of the distance reduction $\alpha : \{0, \ldots, L\} \to [0, 1]$ and the width of the neighborhood of the winner neuron, the so called neighborhood radius function $\gamma : \{0, \ldots, L\} \to \mathbf{R}_0^+$, shrink to zero with time. After a suitable number of training steps the codebook vectors that are related to neighboring neurons on the grid, are neighboring in the input space according to the chosen distance function. Therefore the codebook vectors not only determine via Eq. (3.2) a decomposition of Ω, but also approximate the topology of the input space via the neighborhood structure of the grid.

Algorithmic Realization In the following we describe the initialization of the codebook vectors, the definition of the winner neuron together with its grid neighborhood and the specification of the codebook adaptation rule.

Initialization. We suggest to choose the initial values $w_1(0), \ldots, w_\Bbbk(0)$ as approximately P_ρ-distributed random vectors with $w_s(0) \in \Omega$.

Winner neuron and grid neighborhood. Let $x = (x_1, \ldots, x_q)^T \in \Omega$ be an any input vector and $w_1, \ldots, w_\Bbbk \in \Omega$ the actual codebook vectors of the SOM. Then we call neuron $p \in \{1, \ldots, \Bbbk\}$ the *winner neuron* for input x, if

$$p = \min\{s \mid \operatorname{dist}(x, w_s) = \min_{i=1,\ldots,\Bbbk} \operatorname{dist}(x, w_i)\}. \tag{3.4}$$

Note that Eq. (3.4) is equivalent to $x \in \Theta_{w_p}$, if Θ_{w_p} is defined according to Eq. (3.2).

To determine the neighboring neurons of the winner neuron, one has to specify a grid distance function $\eta : G \times G \times \mathbf{R}^+ \to [0, 1]$. Usually one uses either the bubble grid distance

$$\eta_{\text{bubble}}(z_s, z_p, \gamma) := \begin{cases} 0 & \text{if } \|z_s - z_p\| \leq \gamma \\ \\ 1 & \text{else,} \end{cases}$$

or the Gaussian grid distance

$$\eta_{\text{gaussian}}(z_s, z_p, \gamma) := 1 - \exp\left(-\frac{\|z_s - z_p\|^2}{2\gamma^2}\right),$$

where γ denotes the actual neighborhood radius and $\|\cdot\|$ the two-dimensional Euclidean distance. A neuron s belongs to the neighborhood of winner neuron p if $\eta(z_s, z_p, \gamma) < 1$. If we choose $\eta_{gaussian}$, then the neighborhood of each neuron covers *all* grid neurons.

Codebook adaptation rules. Let neuron p be the winner neuron for input $x(t) = (x_1(t), \dots, x_q(t))^T \in \Omega$ at time t and $w_1(t), \dots, w_{\mathbb{k}}(t) \in \Omega$ the actual codebook vectors. Further let $\alpha(t)$ and $\gamma(t)$ be two time-dependent linear or log-linear functions that decrease to zero with $\alpha(0) \leq 1$ and $\gamma(0) \leq \frac{\min\{m_x, m_y\}}{2}$.
Then the new codebook vectors $w_1(t+1), \dots, w_{\mathbb{k}}(t+1)$ are computed as

$$w_s(t+1) := w_s(t) + \alpha(t)\, \mathrm{neigh}(z_s, z_p, t)\, (x(t) - w_s(t)) \tag{3.5}$$

with $\mathrm{neigh(z_s, z_p, t) := 1 - \eta(z_s, z_p, \gamma(t))}$.

In the case that we set $\gamma(0) = 0$, the SOM is a pure VQ algorithm and therefore optimizes the distortion value [48]. If we allow neighborhood learning, e.g., $\gamma(0) > 0$, the formulation of an energy function that is minimized by the SOM is not possible [47]. Recently slight modifications of the adaptation rules have been suggested that allows the formulation of an energy function without destroying the essential features of the SOM [38, 40]. For further theoretical investigations of the SOM algorithm, especially a comparison to pure VQ methods, see [54, 11, 12].

3.2 Self-Organizing Box Maps (SOBM)

The basic idea of the recently developed Self-Organizing Box Maps (SOBM) method [29] is to compute *codebook boxes* $\hat{W}_s := (\hat{W}_{s_1}, \dots, \hat{W}_{s_q}) \in \mathrm{BOX}(\Omega)$ with $\hat{W}_{s_i} = [l_{s_i}, r_{s_i}] \subset \mathbf{R}$ instead of codebook vectors $w_s \in \Omega$. This is done in such a way that each codebook box is a nearly optimal box approximation of its corresponding partition $\Theta_{\hat{W}_s} \subset \Omega$:

We will call any set $B = \bigotimes_{i=1}^{q} [l_i, r_i] \in \mathrm{BOX}(\Omega)$ with $l_i, r_i \in \mathbf{R}$ an optimal box approximation of a set $M \subset \Omega$ with respect to P_ρ, if

$$P_\rho(B \setminus M) + P_\rho(M \setminus B) \;\rightarrow\; \min .$$

Algorithmic Realization Obviously, this change of concept induces changes of the SOM algorithm, which we arrange here:

Initialization. Let $w_1(0), \dots, w_{\mathbb{k}}(0)$ be different initial values for the codebook vectors of the traditional SOM, e.g., approximately P_ρ-distributed random vectors with $w_s(0) \in \Omega$ for $s = 1, \dots \mathbb{k}$. 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 so that $\hat{W}_s \cap \hat{W}_p = \emptyset$ for all $s, p \in \{1, \ldots, \Bbbk\}$.

Winner neuron. We suppose that the problem specific q-dimensional distance function $\mathrm{dist}(x, y)$ with $x, y \in \Omega$ can be written as a function F of q one-dimensional distance measures $d_i(x_i, y_i)$, which means that $\mathrm{dist}(x, y) = F(d_1(x_1, y_1), \ldots, d_q(x_q, y_q))$. Note that many popular distance measures, as e.g., the Euclidean distance, just exhibit this feature. Obviously we need a distance measure DIST that permits to compute the distance between an input vector $x \in \Omega$ and codebook boxes $\hat{W}_s \in \mathrm{BOX}(\Omega)$. For that purpose, we suggest

$$\mathrm{DIST}(x, \hat{W}_s) := F(\hat{d}_1(x_1, \hat{W}_{s_1}), \ldots, \hat{d}_q(x_q, \hat{W}_{s_q}))$$

with

$$\hat{d}_i(x_i, \hat{W}_{s_i}) := \begin{cases} 0 & \text{if } x_i \in \hat{W}_{s_i} \\ \min\{d_i(x_i, l_{s_i}), d_i(x_i, r_{s_i})\} & \text{else.} \end{cases}$$

Then the winner neuron p has to match a condition analogous to Eq. (3.4):

$$p = \min\{s \,|\, \mathrm{DIST}(x, \hat{W}_s) = \min_{i=1,\ldots,\Bbbk} \mathrm{DIST}(x, \hat{W}_i)\}. \qquad (3.6)$$

Obviously we can use Eq. (3.6) to define for each codebook box $\hat{W}_s$ the corresponding partition $\hat{\Theta}_s := \Theta_{\hat{W}_s} \subset \Omega$ analogously to Eq. (3.2).

Codebook adaptation rules. In analogy to the SOM algorithm, the SOBM algorithm has to adapt the codebook *boxes*. This will be done by the following rules:

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

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

with a 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 adaptation of the nearest interval boundary, as when they are inside the interval. In the following we will motivate the suggested adaptation rule.

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 box 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 adaptation steps. If this difference becomes larger, it is not only possible that $P_\rho(\hat{W}_s \setminus \hat{\Theta}_s)$ increases so that $\hat{W}_s$ is no longer a good box approximation of $\hat{\Theta}_s$. Also the probability grows that one observes overlaps between the boxes after the algorithm stops (see Figure 3.1).

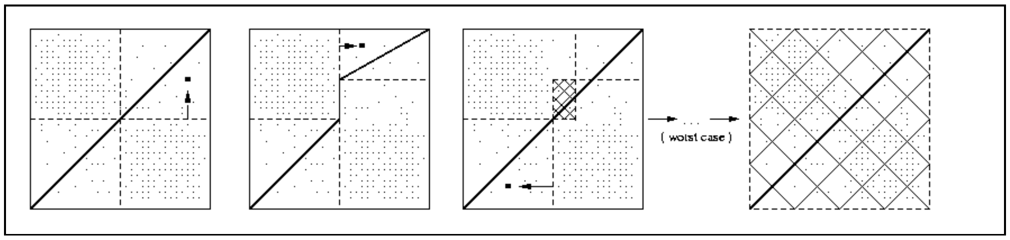

Figure 3.1: **Poor partitioning in the absence of interval shrinkage.**

If, however the overlap between the boxes is too large, $\hat{W}$ and its corresponding decomposition are no longer an approximate box decomposition. 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, the so called interior adaptation. It is obvious that the adaptation 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. 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 box approximation of $\hat{\Theta}_s$ 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 adaptation. 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, we suppose that there exist $a_i, b_i \in \mathbf{R}$ so that we have $\Omega_\rho := \{x \in \Omega |\, \rho(x) > 0\} \subset \bigotimes_{i=1}^q [a_i, b_i]$. Let $\hat{\Theta}_s(t)$ be the decomposition 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.

$$\text{boxvol}(\Delta_s(t)) := \prod_{i=1}^q (r^*_{s_i}(t) - l^*_{s_i}(t)) \;\rightarrow\; \min .$$

For our further expositions we define for $M \subset \Omega$ with $P_\rho(M) > 0$, the conditional probability density function ρ_M on M via

$$\rho_M(\omega) \;:=\; \begin{cases} \frac{\rho(\omega)}{P_\rho(M)} & \text{if } \omega \in M \\ 0 & \text{else.} \end{cases}$$

Using $\rho_{\hat{\Theta}_s(t)}$, we can compute the conditional expectation value $E(\hat{W}_s(t+1))$ for each actual codebook vector $\hat{W}_s(t)$ 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

$$\rho_{\hat{\Theta}_s(t),i}(x_i) \quad := $$

$$\int_{a_1}^{b_1} \cdots \int_{a_{i-1}}^{b_{i-1}} \int_{a_{i+1}}^{b_{i+1}} \cdots \int_{a_q}^{b_q} \rho_{\hat{\Theta}_s(t)}((x_1, \ldots, x_q)^T)\, dx_1 \ldots dx_{i-1}\, dx_{i+1} \ldots dx_q \, .$$

Upon considering our above adaptation rule we obtain:

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

and

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

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 $[l_i^*(t), r_i^*(t)]$ so that

$$\rho_{\hat{\Theta}_s(t),i}(x_i) := \begin{cases} \frac{1}{r_i^*(t) - l_i^*(t)} & \text{if } X = (x_1, \ldots, x_q)^T \in \Delta_s(t) \\ 0 & \text{else}\, . \end{cases}$$

Hence, we arrive at:

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

For the left hand boundary, we analogously obtain:

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

By means of the intuitive choice

$$c(l_{s_i}(t), r_{s_i}(t)) := \frac{1}{6}\,(r_{s_i}(t) - l_{s_i}(t)) \tag{3.7}$$

we end up with

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

and

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

in terms of some model quantity

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

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 $t \in \{0, \ldots, L\}$.

Lemma 3.2.1 *For any $s \in \{1, \ldots, q\}$ and all $t \in \{0, \ldots, L\}$ we have*

$$l_{s_i}(t) < r_{s_i}(t) \implies l_{s_i}(t+1) < r_{s_i}(t+1).$$

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

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

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

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

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

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

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

$\square$

Note that Lemma 3.2.1 is usually not true if $\alpha(t) \geq 6$.

Hence if $l_{s_i}(0) < r_{s_i}(0)$, Lemma 3.2.1 guarantees that $c(l_{s_i}(t), r_{s_i}(t)) > 0$ and $\psi_{s_i}(t) > 0$ for all $t \in \{0, \ldots, L\}$.

Therefore we obtain

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

and

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

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

Using this correction term the presented SOBM algorithm is suitable to generate approximate box decompositions of V (see Definition 2.2.1):

Lemma 3.2.2 *Assume* $\hat{W} := \{\hat{W}_1, \ldots, \hat{W}_{\Bbbk}\} \subset \Omega$ *so that* $\hat{W}_p \in \mathrm{BOX}(\Omega)$ *is a nearly optimal box approximation of* $\Theta_{\hat{W}_p}$ *for* $p = 1 \ldots, \Bbbk$. *Set* $\Theta_{\hat{W}_p}(V) := \Theta_{\hat{W}_p} \cap V$. *Then* $\Theta_{\hat{W}_I}(V) := \{\Theta_{\hat{W}_{p(V)}} \,|\, p \in I\}$ *with* $I := \{p \,|\, \Theta_{\hat{W}_p}(V) \neq \emptyset\}$ *is a decomposition of* V *with* $n_k := |I| \leq \Bbbk$ *partitions and* $(\Theta_{\hat{W}_I}(V), \hat{W}_I)$ *with* $\hat{W}_I := \{\hat{W}_p \,|\, p \in I\}$ *is an approximate box decomposition.*

Proof: There exists a small $\delta > 0$ so that for any $p \in I$ we have

$$f(\hat{W}_p \setminus \Theta_{\hat{W}_p}) + f(\Theta_{\hat{W}_p} \setminus \hat{W}_p) < \delta f(V).$$

This guarantees $f(\Theta_{\hat{W}_p} \cap \hat{W}_p) > f(\Theta_{\hat{W}_p}) - \delta f(V)$ for any $p \in I$. Since $\Theta_{\hat{W}_I}(V)$ is a decomposition of V by construction and $f(M \cap V) = f(M)$ for any subset M of Ω, this yields:

$$\mathrm{overlay}_f(\Theta_{\hat{W}_I}(V), \hat{W}_I) > 1 - \delta n_k.$$

One easily verifies that for any $p \in I$, we have

$$f(\hat{W}_p \cap \bigcup_{\widetilde{p} \neq p} \hat{W}_{\widetilde{p}}) \leq \sum_{s \in I} f(\hat{W}_s \setminus \Theta_{\hat{W}_s}) = \sum_{s \in I} f(\hat{W}_s) - \sum_{s \in I} f(\hat{W}_s \cap \Theta_{\hat{W}_s})$$

Since $\sum_{s \in I} f(\hat{W}_s) \leq f(V)$, this yields:

$$\begin{aligned} \mathrm{overlap}_f(\hat{W}_I) &\leq \sum_{s \in I} \left(1 - \frac{\sum_{s \in I} f(\hat{W}_s \cap \Theta_{\hat{W}_s})}{\sum_{s \in I} f(\hat{W}_s)} \right) \\ &\leq \sum_{s \in I} \Big(1 - \mathrm{overlay}_f(\Theta_{\hat{W}_I}(V), \hat{W}_I) \Big) \;<\; \delta n_k^2. \end{aligned}$$

□

3.3 Comparison SOM - SOBM

Upon comparing codebooks W and $\hat{W}$, computed by the original SOM and the SOBM algorithm with the same parameters and initialization, one will observe clear similarities. In most cases the orientation of the maps and the visually identifiable clusters are equal (see subsection 5.2.2 for an example).

For each codebook vector $w_p \in W$ one can usually find a codebook box $\hat{W}_s \in \hat{W}$ with $w_p \in \hat{W}_s$. Therefore the SOBM algorithm will be at least as powerful as the classical SOM algorithm. In the following, however, we will show that the SOBM algorithm has important advantages.

For simplicity we suppose 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 $\mathrm{neigh}(z_s, z_{\overline{s}}, t) = 0$ for $t \in \{0, \ldots, L\}$.

For the purpose of illustration, we define two probability density functions ρ_1 and ρ_2 (see Figure 3.2):

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

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

We have used the original SOM algorithm and our extended algorithm with $c = 0$ and c as defined in Eq. (3.7) to compute the codebooks for ρ_1 and ρ_2.

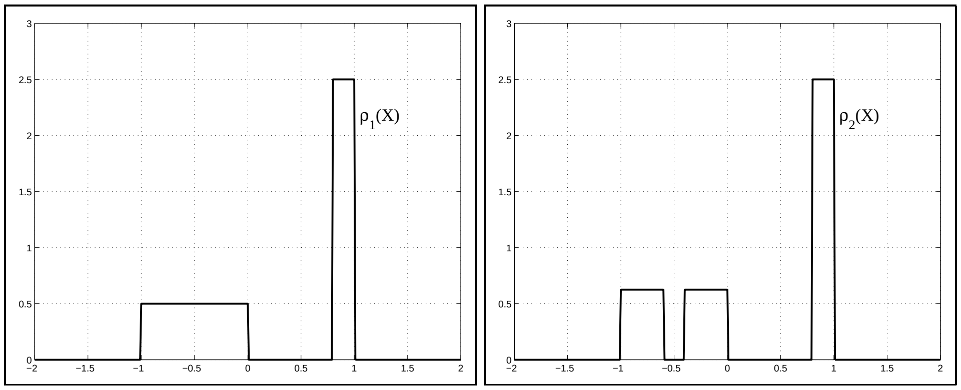

Figure 3.2: **Probability density functions ρ_1 and ρ_2**

Table 3.1 shows the results (random codebook initialization, $\alpha(0) = 0.9$ and $L = 10000$).

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

Table 3.1: **Codebooks 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 these intervals are of different width, we get no hint about this fact, if we look at the codebook vectors w_s and $w_{\overline{s}}$.
- The codebook boxes are box approximations of the partitions, which they implicitly define. These approximations are perfect if we use the correction term c as defined in Eq. (3.7).
- The point codebooks are equal for both probability density functions, i.e. although ρ_1 and ρ_2 are different, we cannot distinguish them by looking at

the codebook vectors. The situation is quite different if we use the correction term c and look at the codebook boxes. 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 cannot 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 the SOBM 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 3.3 shows ρ_3 with $\mu_1 = -0.5$, $\mu_2 = 0.5$ and $\sigma = 0.27$ and Table 3.2 gives the corresponding computational results.

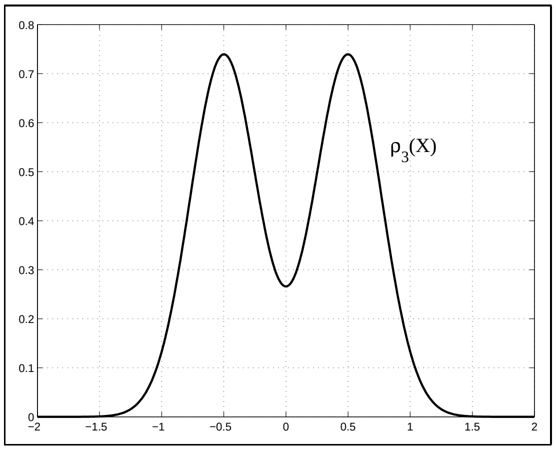

Figure 3.3: **Probability density functions** ρ_3

	ρ_3
SOM	$w_s = -0.5,\ w_{\overline{s}} = 0.5$
SOBM$(c=0)$	$\hat{W}_s = [-0.67, -0.33], \hat{W}_{\overline{s}} = [0.31, 0.67]$
SOBM	$\hat{W}_s = [-0.85, -0.15],\ \hat{W}_{\overline{s}} = [0.13, 0.86]$

Table 3.2: **Codebook vectors for** ρ_3

Although the concept of codebook boxes develops its full power still within the computation of approximate box decompositions the advantages in comparison with point codebooks are already obvious.

A disadvantage of the SOBM algorithm is that it requires more computing time than the classical SOM algorithm. Although the difference depends on the chosen implementation, one easily checks that the number of variables that have to be adapted and to be evaluated are doubled. Therefore in the worst case the SOBM algorithm doubles the computing time of the original SOM algorithm.

3.4 Computational complexity

To speed up the computing time, one may think about a combination of the SOM and the SOBM algorithm. In the following we suggest such a combination, which has turned out to be quiet powerful in our first applications (see chapter 5).

As usual in the original SOM algorithm, we first compute in $L_1 := u \cdot \Bbbk$ steps the codebook vectors $w_1, \ldots, w_{\Bbbk}$ with a suitable average number of codebook updates u, e.g., $u = 100$, a large learning rate at the beginning, e.g., $\alpha(0) = 1$, and with neighborhood adaptation, i.e. $\mathrm{neigh}(z_s, z_p, t) > 0$ for $t < L_1$. This is often called the *ordering phase* of the SOM algorithm.

After this ordering phase one usually passes on to another adaptation cycle with $L_2 \geq L_1$ adaptation steps, a low learning rate α and no neighborhood adaptation, i.e. $\mathrm{neigh}(z_s, z_p, t) = 0$ for $s \neq p$ and $t \in [L_1, L_1 + L_2]$. 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.

To achieve convergence, in the classical SOM algorithm L_2 is usually much larger than L_1, e.g., a factor 3 or more. In our combined approach, we set $L_2 \approx L_1$ and use the SOBM algorithm for an additional convergence phase: We first initialize the codebook boxes $\hat{W}_s(0)$ by using the earlier computed representatives $w_s(L_2)$ within the described initialization routine. Then we adapt the codebook boxes in $L_3 \approx L_2$ time steps with a low learning rate and no neighborhood adaptation.

Summarizing, as a result of this combination — original SOM algorithm plus additional convergence phase with SOBM algorithm — we obtain a shorter computing time, as if we only use the SOBM algorithm, while getting comparable results. Additionally we avoid possible negative effects of the neighborhood adaptation on the generation of the codebook boxes.

Up to now, we have not answered the question, if the SOM/SOBM algorithm needs less than $\mathcal{O}(n^2)$ operations to compute a decomposition of any data set $V \subset \Omega$ with n data objects and dimension q.

Let u denote the average number of codebook updates that is sufficient to guarantee convergence of the SOM/SOBM algorithm, i.e. we need $\mathcal{O}(u \cdot \mathbb{k})$ adaptation steps. Since we have to compute the winner neuron and to adapt the codebook within each adaptation step, each of these steps costs $\mathcal{O}(q \cdot \mathbb{k})$ operations. Therefore we need $\mathcal{O}(u \cdot q \cdot \mathbb{k}^2)$ operations to generate a suitable codebook. In addition, the computation of a decomposition based on this codebook according to Eq. (3.2) can be done with $\mathcal{O}(q \cdot \mathbb{k} \cdot n)$ operations.

Since for large cluster problems we usually have

$$u \cdot \mathbb{k} \leq n \quad \text{and} \quad q \ll n,$$

we totally need

$$\mathcal{O}(u \cdot q \cdot \mathbb{k}^2 + q \cdot \mathbb{k} \cdot n) = \mathcal{O}(\mathbb{k} \cdot n)$$

operations to compute a decomposition of the data set V via the SOM/SOBM algorithm.

If we choose $\mathbb{k}$ significantly smaller than n, e.g., $\mathbb{k} = \mathcal{O}(\log n)$, this guarantees that we can compute a decomposition much faster than $\mathcal{O}(n^2)$.

Therefore the SOM/SOBM algorithm is a suitable heuristic for the computation of decompositions.

3.5 Practical extensions

In the following we shortly describe two practical extensions of the SOM and the SOBM algorithm, whenever they are used for computing decompositions of a data set V with frequency function f and homogeneity function h.

3.5.1 Pruning

Neuron pruning is a classical technique in the field of neural networks, to simplify the network architecture and therefore also the corresponding model. In our setting each neuron of the Kohonen layer corresponds with one codebook vector w_p. If now n_k is too large after the convergence phase of the SOM, we eliminate those neurons, whose associated codebook vector w_s only represents a small number of input objects, i.e. w_s with $f(\Theta_{w_p}) < \delta_1$ for sufficiently large δ_1, e.g., $\delta_1 := \frac{f(V)}{n_k}$.

Note that after such a neuron pruning, the corresponding decomposition Θ has changed, especially n_k is smaller than before. Pruning has the additional advantage that it prevents the appearing of pseudo clusters (see the earlier discussion in section 3.1).

3.5.2 Early stopping

A main problem of the SOM algorithm is the fact that the number of training steps of the convergence phase has to be fixed a priori and therefore must be set to a large value, because otherwise we cannot be sure that we will reach convergence. If we use our combined SOM/SOBM algorithm, the choice of the length of the SOM convergence phase is rather uncritical, because we have an additional convergence phase of the SOBM algorithm. At a first view, the determination of the right number of convergence steps for the SOBM algorithm seems to be as problematic as for the SOM algorithm. But if we look closer, we detect a nice early stopping criterion for the SOBM algorithm:

To guarantee that $(\Theta_{\hat{W}}(V), \hat{W})$ is an approximate box decomposition, we have to ensure that $\mathrm{overlap}(\hat{W})$ is small. Therefore we have to stop the adaptation of the codebook boxes, if $\mathrm{overlap}(\hat{W}) > \delta_2$ with small δ_2, e.g., $\delta_2 = 0.001$.