4 Methods of Projection

Dimensionality reduction techniques reduce the dimensions of the input space to facilitate the exploration of structures in high-dimensional data. Two general dimensionality reduction approaches exist: manifold learning and projection. Manifold-learning methods attempt to find a sub-space in which the high-dimensional distances can be preserved. These sub-spaces may have a dimensionality of greater than two. However, only two- or three-dimensional representations of high-dimensional data are easily graspable for to the human observer.
The goal of this chapter is the visualization of structures in high-dimensional data. Venna et al. argued that "manifold learning methods are not necessarily good for […] visualization […] since they have been designed to find a manifold, not compress it into a lower dimensionality" [Venna et al., 2010, p. 452], and it has been shown by van der Maaten et al that they do not outperform classical principal component analysis (PCA) for real-world tasks [L. J. van der Maaten et al., 2009].
Therefore, this chapter focuses on common projection methods. Many projection methods are characterized by an objective function that is optimized using gradient descent or a corresponding learning algorithm. The quality of the projection and, consequently, of the visualization will critically depend on the similarity concept chosen as the basis of the objective function, which may be based on either distance or local proximity; thus, the methods will be categorized on this basis. This chapter will attempt to relate the various projection approaches to the compact and connected structure types introduced in the previous chapter.

4.1 Common Approaches

Here, projection is used as a method for visualizing high-dimensional data in a two-dimensional space such that the discontinuities in the data are captured. Thus, the quality of a projection critically depends on the chosen similarity concept. This concept may be defined based on either distance or local proximity. The former type of similarity describes the arrangement of all given points in space and is sometimes called topography; the latter compares local neighborhoods and is sometimes called topology. Here, projections are called focusing if they are constructed using an iterative learning process that first adapts to the global intercluster distances and then focuses on more local intracluster distances.

4.1.1 Principal Component Analysis (PCA)

PCA assumes that the directions in the input space that show the highest variance contain the most information about the data set [Hotelling, 1933]. The coordinate system of the input space is replaced with a (principal) coordinate system in which the variance of the data is maximized. This is achieved by finding a set of weighted linear combinations of the original variables, where the weights are found through eigendecomposition (for a definition, see [Goodfellow et al., 2016, pp. 42-44]).
Pearson proposed an equivalent definition based on an objective function in which the average projection cost is minimized [Pearson, 1901]. The projection cost is defined in terms of the mean squared distances between the points $i \in I$ and the projected points $j \in O$:

M. C. Thrun, *Projection-Based Clustering through Self-Organization and Swarm Intelligence*, https://doi.org/10.1007/978-3-658-20540-9_4

$$E = \frac{1}{n}\sum_{l\in I}^{n} D(l,\hat{j}) \qquad (4.1)$$

where $\hat{j} = j + \sum_{i=m+1}^{n} b_i * \hat{u_i} = \sum_{i=1}^{m} b_i * u_i + \sum_{i=m+1}^{n} b_i * \hat{u_i}$ has the same dimension as $l \in I$. Here, *n* is the dimension of the input space *I*, m is the dimension of the output space *O*, the u_i are the basis vectors, and the b_i are constants. The minimization of *J* is achieved by choosing the basis vectors to be eigenvectors of the covariance matrix constrained by the orthonormality conditions [Duda et al., 2001, pp. 114-117]:

$$Cov(l,j) = \frac{1}{n}\sum_{l\in I}(l - mean_l)\,(j - mean_j) \qquad (4.2)$$

$$Cov(l,j) * u_i = \lambda_i u_i \qquad (4.3)$$

Now, the objective function E can be redefined in (4.4) in terms of the eigenvalues λ_i in (4.1) as

$$E = \sum_{i=m+1}^{n} \lambda_i \qquad (4.4)$$

where *n* is the dimension of the input space I and m is the dimension of the output space O. The largest eigenvalues correspond to the $1, \dots, m$ dimensions with the largest variance. Dimensions of the input space with small variances are discarded. Thus, PCA is an orthogonal projection of the data into a lower-dimensional space. It should be noted that "PCA remains a rather basic method and suffers from many shortcomings" [Lee/Verleysen, 2007, p. 226].

4.1.2 Independent Component Analysis (ICA)

> *"Independent component analysis (ICA) is a method for finding underlying factors or components from multivariate (multi-dimensional) statistical data. What distinguishes ICA from other methods is that it looks for components that are both statistically independent, and nonGaussian" [Hyvärinen et al., 2004].*

Let $I = (l_1, \dots, l_n)$ be defined as the matrix of the data in the input space. ICA assumes that *I* is a linear combination of non-Gaussian independent components *S* as follows:

$$I = S * A \qquad (4.5)$$

where *A* is a linear mixing matrix and $S = (j_1, \dots, j_n), j \in O$. ICA unmixes *I* by estimating a matrix $W = A^{-1}$ such that

$$I * W = S \qquad (4.6)$$

With the goal of estimating *W*, the central limit theorem and matrix search can be used to maximize the non-Gaussianity. In the fastICA algorithm [Hyvärinen, 1997], the non-Gaussianity is defined as the negentropy *F*, and it is approximately maximized by maximizing the objective function in (4.7)

$$E(j) \approx \left[F\{G(j)\} - F\{G(N(m=0, s=1))\}\right]^2 \qquad (4.7)$$

where *N* is a Gaussian and G is a contrast function, e.g., $G(u) = -exp(-\frac{au^2}{2})$.

Constraints on the estimated contrast function *G* include pre-whitening and the centering of the data in the input space [Hyvärinen et al., 2004].

4.1.3 Non-linear metric multidimensional scaling (MDS) techniques

Multidimensional scaling (MDS) was originally proposed by [Torgerson, 1952]. MDS techniques attempt to preserve the pairwise distances *D(l, j)* of the input space in the output space to the greatest possible extent. Therefore, MDS techniques minimize an objective (error) function *E* that is, as given in [Kruskal, 1964b], defined as

$$E(D,d) = \sum_{j,l=1,j<l}^{n} \Big(f\big(D(l,j)\big) - d(l,j)\Big)^2 \qquad (4.8)$$

where *f(D(l, j))* is a *non-metric*, monotonic transformation of the distances in the input space [Kruskal, 1964a, p. 7]. *E* is often called the stress, and *E* is minimized in an attempt to reproduce the general rank ordering of the distances. This minimization is usually performed via gradient descent.

However, the objective function *E* depends on the scale on which the distances are measured. It is preferable to normalize the objective *E* to reduce it to the same units in which the distances are expressed (Eq.4.9). Sammon mapping [Sammon] is one type of MDS technique and uses the error function

$$E(D,d) = \frac{1}{\sum_{j,l=1,j<l}^{n} D(l,j)} \sum_{j,l=1,j<l}^{n} \frac{\big(D(l,j) - d(l,j)\big)^2}{D(l,j)} \qquad (4.9)$$

4.1.4 Curvilinear Component Analysis (CCA)

When a non-linear structure is being analyzed, MDS cannot reproduce all distances. Therefore, [Demartines/Hérault] proposed a projection method that favors local neighborhoods. Curvilinear component analysis (CCA) attempts to reproduce short distances before reproducing long distances [Demartines/Hérault, 1995]. The objective function is defined in (4.10) as

$$E(D,d) = \sum_{j,l=1,j<l}^{n} \big(D(l,j) - d(l,j)\big)^2 * h(D(l,j),r) \qquad (4.10)$$

where h: $\mathbb{R} \rightarrow [0,1]$ is a neighborhood function that depends on a radius R as follows:

$$h(D(l,j),R) = \begin{cases} 1, & if D(l,j) \le R \\ 0, & otherwise \end{cases} \qquad (11)$$

4.1.5 t-Distributed Stochastic Neighbor Embedding (t-SNE)

The t-distributed stochastic neighbor embedding (t-SNE) technique is an enhanced version of SNE [Hinton/Roweis, 2002] in which the Kullback-Leibler divergence (KLD) is symmetrized and the crowding problem solved. The latter is achieved by redefining the conditional probabilities in the output space O through the application of Student's t-distribution with

$$p(l|j) = \begin{cases} \dfrac{(1 + \mathrm{d(l,j)}^2)^{-1}}{\sum_{\mathrm{l,j}\in\mathrm{I}}(1 + \mathrm{d(l,j)}^2)^{-1}}, & l \neq j \\ 0, & l = j \end{cases} \qquad (4.12)$$

In [Van der Maaten/Hinton], the distance between two data points is redefined as the conditional probability that j would pick l, where $l, j \in I$, as follows:

$$P(l|j) = \begin{cases} \dfrac{\exp\left(-\dfrac{D(l,j)^2}{2\sigma(l)^2}\right)}{\sum_{\mathrm{l,j}\in\mathrm{I}} \exp\left(-\dfrac{D(l,j)^2}{2\sigma(l)^2}\right)}, & l \neq j \\ 0, & l = j \end{cases} \qquad (4.13)$$

where $\sigma(l)$ is the variance of a Gaussian that is centered on data point *j*. If the projection is correct, then the conditional probabilities will be equal [Van der Maaten/Hinton]. Therefore, the objective function is defined using the symmetric KLD in (14) as

$$E = \sum_i \sum_j \frac{P(l|j) + P(j|jl)}{2n} * log\left(\frac{\dfrac{P(l|j) + P(j|jl)}{2n}}{p(l|j)}\right) \qquad (4.14)$$

4.1.6 Neighborhood Retrieval Visualizer (NeRV)

[Venna et al., 2010] reintroduced the idea of misses used by [Ultsch/Herrmann, 2005], where misses are similar data points $(l_I, j_I) \in i$ that are mapped onto far separated points $(l_O, j_O) \in O$ [Ultsch/Herrmann, 2005]. Conversely, if a pair of closely neighboring positions (l_O, j_O) represents a pair of distant data points, then this pair is called a false positive. From the information retrieval perspective, this approach allows one to define the precision F_P and the recall F_R for the case in which the neighborhoods are simply binary. However, [Venna et al., 2010] goes a step further by replacing such binary neighborhoods with probabilistic ones, which are loosely inspired by the SNE approach [Hinton/Roweis, 2002]. The neighborhood of the point l is defined in terms of the relevance of the $j \in I$ points around l:

$$p_l(j) = \frac{\exp\left(-\dfrac{D(l,j)^2}{\sigma_l^2}\right)}{\sum_{k\neq j} \exp(-\dfrac{D(l,k)^2}{\sigma_l^2})} \qquad (4.15)$$

where σ_l is set to the value for which the entropy of $p_l(j)$ is equal to log(knn) and knn is a rough upper limit on the number of relevant neighbors that is set by the user [Venna et al., 2010]. The authors propose a default value of 20 effective nearest neighbors. Similarly, the corresponding neighborhood in the output space is defined as

$$q_l(j) = \frac{\exp\left(-\dfrac{d(l,j)^2}{\sigma_l^2}\right)}{\sum_{k\neq j} \exp(-\dfrac{d(l,k)^2}{\sigma_l^2})} \qquad (4.16)$$

These neighborhoods are compared based on the mean of the KLD, which is used to define the precision F_P and recall F_R:

$$F_R = -\frac{1}{N}\sum_l^N \sum_{j\neq l} p_j(l) * \log(\frac{p_j(l)}{q_j(l)}) \qquad (4.17)$$

$$F_P = -\frac{1}{N}\sum_l^N \sum_{j\neq l} q_j(l) * \log(\frac{q_j(l)}{p_j(l)}) \qquad (4.18)$$

The objective function is then defined in (19) as

$$E = \lambda \sum_{ij} p_j(l) * \log\left(\frac{p_j(l)}{q_j(l)}\right) + (1-\lambda) \sum_{ij} q_j(l) * \log\left(\frac{q_j(l)}{p_j(l)}\right) \qquad (4.19)$$

The objective function E is non-linearly optimized via conjugate gradient descent. In the absence of prior knowledge, the neighborhoods p are defined as symmetric Gaussians or heavy-tailed distributions. The weighting between precision and recall must be set by the user using the parameter λ. Weighting precision over recall means that if points are similar to each other in the output space, then they will also be similar to each other in the input space, whereas weighting recall over precision means that if points are similar in the input space, then they will also be similar in the output space. Note that the KLD and the symmetric KLD do not follow the triangle inequality for metric spaces.
The projection approach used in the Neighborhood Retrieval Visualizer (NeRV) method is randomly initialized by default, resulting in stochastic projections (see Figure 4.1). However, there exists an option to use PCA projection for initialization.

4.2 Emergent Self-Organizing Map (ESOM)

Self-organizing (feature) map (SOM) was invented by [Kohonen, 1982a, 1982b] and is a type of unsupervised neural learning algorithm. In contrast to other neural network models[20] a SOM consists of an ordered two-dimensional layer of neurons called units. Neurons are interconnected nerve cells in the human neocortex [H. Ritter et al., 1992, p. 22], and the SOM approach was inspired by somatosensory maps (e.g. see [Hennig et al., 2015, p. 421] cites [Haykin, 1994], see also [Kandel, 2012, p. 335]). There are two types of SOM algorithms: online and batch [Fort et al., 2001]. The first is stochastic, whereas the second is deterministic, which means that it yields reproducible results for a given parameter setting. However, Fort et al. have argued "that randomness could lead to better performances" [Fort et al., 2001, p. 12].
The main differences between batch-SOM [Kohonen/Somervuo, 2002] and online-SOM [Kohonen, 1995] lie in the updating and averaging of the input data. In batch-SOM, prototypes (see Eq. 4.20 below) are assigned to the data points and the influences of all associated data points are calculated simultaneously, in contrast to online-SOM, in which sequential training of the neurons is applied (as described in detail below). The batch-SOM method has been shown to produce topographic mappings of varying quality depending on the pre-defined parametrization [Fort et al., 2001], and "the representation of clusters in the data space on maps trained with batch learning is poor compared to sequential training" [Nöcker et al., 2006]. An important comparison between the batch-SOM approach and ant-based clustering was presented by [Herrmann/Ultsch, 2008c] and will be elaborated upon in chapter 7. No objective function is used in online-SOM [Lee/Verleysen, 2007, p. 241], and SOM remains a reference tool for two-dimensional visualization [Lee/Verleysen, 2007, p. 244].
In one common approach to applying the SOM concept, the algorithm acts as an extension of the k-means algorithm [Cottrell et al., 2016] or is a partitioning method of the k-means type [Murtagh/Hernández-Pajares, 1995]. In such a case, only a few units are used in the SOM algorithm to represent the data [Reutterer, 1998], which results in direct clustering of the data. Here, each neuron can be considered to represent a cluster. For example, Cottrell and de Bodt

[20] For an overview, see [H. Ritter et al., 1992], for deep learning see [Goodfellow et al., 2016].

used 4x4 units to represent the 150 data points in the Iris data set ([Ultsch et al., 2016a] cites [Cottrell, 1996]). Therefore, the conventional SOM algorithm is called k-means-SOM here. This SOM algorithm also has two common extensions called Heskes-SOM [Heskes, 1999] and Cheng-SOM; these two extensions include objective functions [Cheng, 1997] and are not discussed further in this thesis**.** The optimization of objective functions in general will be discussed in chapter 6, where it will be argued that it is not useful for the goal of this thesis. Chapter 7 will show that objective functions are incompatible with self-organization.

The other approach to applying SOM is to exploit its emergent phenomena through self-organization, in which case it is necessary to use a large number of neurons (>4000) [Ultsch, 1999]. This enhancement of the online-SOM approach is called emergent SOM (ESOM). In such a case, the neurons serve as a projection of the high-dimensional input space instead of a clustering, as is the case in k-means-SOM.

Let $M = \{m_1, \dots, m_n\}$ be the positions of neurons on a two dimensional lattice[21] (feature map) and $W = \{w(m_i) = w_i \mid i = 1, \dots n\}$ the corresponding set of weights or prototypes of neurons, then, the SOM training algorithm constructs a non-linear and topology-preserving mapping of the input space by finding the best matching unit (BMU) for each $l \in I$:

$$bmu(l) = \underset{m_i \in M}{\operatorname{argmin}}\{D(l, w_i)\}, \quad i \in \{1, \dots, n\} \qquad (4.20),$$

if in Eq. 4.20 a distance in the input space I between the point *l* and the prototype w_i is denoted. In each step, SOM learning is achieved by modifying the prototypes (weights) in a neighborhood as follows:

$$\Delta w(R) = \eta(R) * h(bmu(l), m_i, R) * (l - w(m_i)) \qquad (4.21)$$

The cooling scheme is defined by the neighborhood function $h: M \times M \times \mathbb{R}^+ \to [-1,1]$ and the learning rate $\eta: \mathbb{R}^+ \to [0,1]$, where the radius R decreases until $R = 1$ in accordance with the definition of the maximum number of epochs. In contrast to all previously introduced projection methods, no objective function is used in the ESOM algorithm. Instead, ESOM uses the concept of self-organization (see chapter 6 for further details) to find the underlying structures in data.

The structure of a (feature) map is **toroidal**; i.e., the borders of the map are cyclically connected [Ultsch, 1999], which allows the problem of neurons on borders and, consequently, boundary effects to be avoided. The positions $m \in M$ of the BMUs exhibit no structure in the input space [Ultsch, 1999]. The structure of the input data emerges only when a SOM visualization technique called U-matrix is exploited [Ultsch/Siemon, 1990].

Let $N(j)$ be the eight immediate neighbors of $m_j \in M$, let $w_j \in W$ be the corresponding prototype to m_j, then the average of all distances between prototypes w_i

$$u(j) = \frac{1}{n} \sum_{i \in N(j)} D(w(m_i), w(m_j)), n = |N(j)| \qquad (4.22)$$

A display of all U-heights in Eq. 4.22 is called a U-matrix [Ultsch/Siemon, 1990].

[21] In general this work uses the term grid if the resulting tiling is hexagonal and lattice if the resulting tiling is rectangular (see connected graph). In the context here the distinction is not important, therefore we use the term (feature) map.

"By formalizing the displayed structures, [Lötsch/Ultsch, 2014] showed that the U-matrix is an approximation of the Voronoi borders of the high-dimensional points in the output space:

Let bmu(l) and bmu(j) be the BMUs of data points l and j, where bmu(j) and bmu(l) have bordering Voronoi cells. On the borderline, there is a vertical plane (AU-height), which is the distance D(l, j) > 0 between the data points in the input space. In sum, the abstract U-matrix (AU-matrix) is the Delaunay graph of the BMUs weighted by the corresponding Euclidean distances in the input space" [Thrun et al., 2016a, p. 9].

4.2.1 Visualizations of SOMs

This section is reproduced in its entirety from [Thrun et al., 2016a]. The result of every Kohonen SOM algorithm is a set of neurons located on a map where a set W of prototypes corresponds to a set M of positions. In general, the positions on M are restricted to a grid/lattice, but a few approaches exist that change the positions in M, like Adaptive Coordinates [Merkl/Rauber, 1997]. Because these approaches are not grid/lattice based, they are not considered any further. BMUs define the locations of input points on the map. However, they exhibit no structure of the input space for a SOM [Ultsch, 1999]. However, the goal is to grasp the high-dimensional data structure and possibly even visualize cluster boundaries. Therefore, post-processing of the neurons is required for an informative representation of high-dimensional data. Three standard approaches are found in the literature:

The first approach projects the set W of prototypes with MDS [Torgerson, 1952] or some of its variants to a two-dimensional space [Kaski et al., 2000; Sarlin/Rönnqvist, 2013]. The result is mapped into the CIELab color space [Colorimetry, 2004]. In this uniform color space, perceptual differences in colors correspond to Euclidean distances in the map space as precisely as possible [Kaski et al., 2000]. The next two approaches visualize either the distances or density of the prototypes.

The second approach defines receptive fields around each position in M. The unified distance matrix (U-matrix), [Ultsch/Siemon, 1990] or one of its variants [Häkkinen/Koikkalainen, 1997; Hamel/Brown, 2011; Kraaijveld et al., 1995] , represents distances of prototypes (see equations above) by using proportional intensities of gray shades, color hues, shape or size. In [Kraaijveld et al., 1995], every neuron corresponds to a pixel. The gray value of each pixel is determined by the maximum unit distance from the neuron to its four neighbors (up, down, left, right). The larger the distance is, the lighter the gray value is. In [Häkkinen/Koikkalainen, 1997], additional unit distance visualization approaches are explained. The shapes and sizes of the receptive fields describe the dissimilarity of corresponding neurons. Apart from the U-matrix, visualizations of receptive fields in three dimensions or specific components of prototypes with receptive fields in two dimensions have been attempted [Vesanto, 1999]. It is also possible to add SOM quality measures to the receptive fields in a third dimension, e.g., [Vesanto et al., 1998].

The third approach connects the positions M by way of a specific scheme. In [Hamel/Brown, 2011], in addition to a U-matrix approach, neurons are connected with lines along the maximum gradient. The authors claim that clusters are the always-connected components of the graph defined by the U-matrix. [Merkl/Rauber, 1997] omitted the receptive fields approach, merely connecting map positions with lines, where the connection intensities reflect the similarity of the underlying prototypes. [K. Tasdemir/Merenyi, 2009] proposed the CONNvis technique, which visualizes the feature map by connecting neurons whose corresponding prototypes are adjacent in an input space with a dimensionality equal to that of the high-dimensional data. The

width of each connection line is proportional to the strength of the connection [K. Tasdemir/Merenyi, 2009].
In sum, all above described visualizations of large SOMs require an expert in the field for interpretation. To the best of the present author's knowledge, there are no 3D visualizations of ESOMs based on a 2D feature map currently in use[22].

4.2.2 Clustering with ESOM

Combining ESOM with the U*-matrix approach enables an application of [Ultsch et al., 2016a]:

> *"A single wall of AU-matrix represents the true distance information between two points in the data space. Valid density information at the midpoints between a BMU and a second BMU is calculated for [the] P-matrix, since the same volumes, i.e. spheres of a predefined radius, are used. The AU*matrix therefore represents the true distance information between two points weighted by the true density at the midpoint. The representation is such that high densities shorten the distance and low densities stretch this distance. Using transitive closure for these weighted distances allows classical clustering algorithms (AU*clustering) to actually perform distance- and density-based clustering, taking into account the complex structure of partially entwined clusters within the data."*

In contrast to the Databionic swarm approach, in which the shortest paths between AU-distances are calculated[23], this clustering approach uses only the direct neighborhood of the projected points. A computation of the abstract P-matrix is necessary because ESOM itself does not consider density. Overlaying a political map on the U*-matrix map reveals errors made by the ESOM algorithm during the annealing process. The political map shows the Voronoi areas of each cluster, where the color of each cluster area corresponds to the cluster label. The clustering is solid if every cluster consists of only one connected area, of which the borders are mountain ranges. The clustering process is sensitive to the parcel window parameter that is required for estimating the density of the high-dimensional data, and the clustering process is mostly conducted through an interactive approach requiring human intervention[24].

4.3 Types of Projection Methods

In the previous section, it was shown that projection methods such as CCA, MDS and NeRV are characterized by an objective function that is optimized using gradient descent or a corresponding learning algorithm, whereas others, such as ESOM, are not. However, the first obvious difference between types of projection methods is that between linear projection methods such as PCA or ICA and non-linear projection methods. Linear projection methods are only able to rotate the high-dimensional data space and choose the most interesting dimensions, such as the dimensions with the highest variance, as is the case for PCA.
In contrast to this approach, non-linear projection methods are able to disentangle structures, e.g., represent the Chainlink data set[25] in such a way that the two clusters are separated in the output space. The next major distinction between projection methods is the deterministic versus the stochastic approach. Some projection methods will always produce the same projection in the output space if all parameters remain unchanged. However, for many projection methods, such as t-SNE, their projections in the output space will drastically change with different trials

[22] Standard ESOM visualizations using the U-matrix are shown in supplementary D.
[23] See chapter 7 for details.
[24] For this reason, the ESOM/U-matrix clustering approach cannot be compared with other approaches in chapter 10.
[25] See the next chapter for details.

even when all settings of the projection method remain unchanged (see also examples in chapter 5, Figure 5.2). Hence, the results of deterministic methods are always reproducible, whereas stochastic methods may yield irreproducible results and require a statistical approach to assess their quality. Similarly to MDS techniques, deterministic projection methods are often based on Lyapunov functions (for further details, see [Lyapunov, 1992]). Here, it is assumed that linear and MDS techniques should only be able to visualize compact structures, which are based on the intra- versus intercluster distances of natural clusters (see the previous chapter for details).

Stochastic methods are mainly characterized by either a focusing approach or a self-organizing approach. Let k be the neighborhood extent, and let Γ be a graph; then, a projection method is of the focusing type if the result is constructed through an iterative learning process that adapts first to global neighborhoods $H(k_1 > 1, \Gamma, I)$ and later to local neighborhoods $H(k_2, \Gamma, I)$, where $k_1 > k_2$. Therefore, such methods should be capable of visualizing connected structures (see the previous chapter for details) if the annealing process is correctly chosen.

Self-organization is defined as spontaneous pattern formation by a system itself, without any central control[26] [Kelso, 1997, p. 8 ff.]. By means of self-organization, some projection methods, such as ESOM or Pswarm, are able to project data without requiring an objective function. Thus, self-organizing methods do not implicitly predefine the structures that are sought in the data of interest. The Pswarm projection method will be introduced in chapter 8 as part of the Databionic swarm clustering approach. An overview of the various types of projection methods is shown in Figure 4.1.

Assumptions regarding the types of structures that the projection methods in Figure 4.1 are able to visualize will be either disproven or verified in chapter 10 based on 100 trials per projection method (with the exception of ICA due to technical difficulties) of five artificial three-dimensional data sets.

[26] Further explained in chapter 7, p.79 ff.

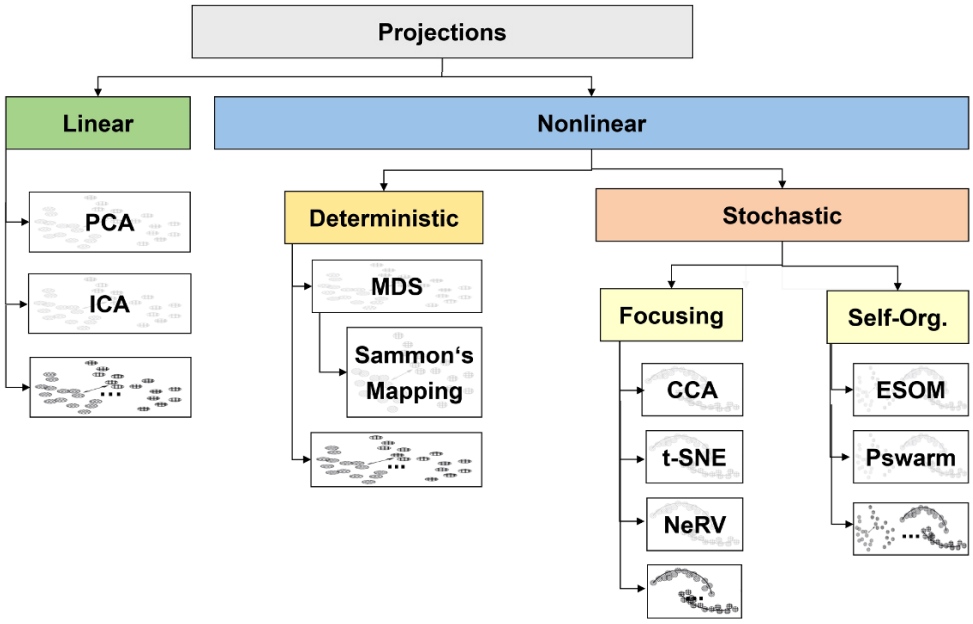

Figure 4.1: Overview of different types of projection methods. Here, it is argued that linear methods and MDS techniques are only able to visualize compact structures (shaded with the first pattern), whereas focusing projection methods should be able to visualize connected structures (shaded with the second pattern) if the annealing scheme is correctly chosen. For self-organizing methods, the structures that are sought in the data are not implicitly predefined. The ellipses indicate that this overview includes only common projection methods. Pswarm will be introduced in chapter 8 as a new approach based on swarm intelligence.