Chapter 4
Network Construction Techniques

Abstract In many areas of machine learning, networks are used to model local relationships between data points and to build global structures from local information. Building networks is often a necessary step when dealing with problems arising from applications in machine learning or data mining. This fact becomes crucial when we want to apply network-based learning methods to vector-based data sets, in which a network must be constructed from the input data set using some convenient network formation criteria. In this chapter, we review the main ingredients that are needed to construct a graph from non-networked data. In special, we discuss transformation of vector-based and time series data. Several similarity functions are also discussed.

4.1 Introduction

Networks are essential for encoding information, and data in network format is increasingly abundant in fields ranging from computational biology to computer vision. The transformation from unstructured data to a network data representation can always be performed in a lossless manner. The inverse transformation, however, is often a lossy one. Let us give an example. Consider the WWW that is inherently represented by a network format. In such a network, pages are vertices and links exist between different pages if one page references another one. Now suppose we desire to extract a vector-based data out of this network. A very difficult task would be to model the recursiveness of cycles in the network topology when we now go to a vector-based format. Moreover, the local and global topologies of the pages relationships would probably be distorted by the transformation. In addition, considering that there are more than one network component, then some shortest paths between members of different components would be infinity in the network. To model this extreme dissimilarity in a vector-based format would be difficult, because the information of whether or not vertices are in the same component is structural and depends on the topology of the data relationships, which in turn is not easily modeled in a vector-based format.

From that example, it is clear that networks embed more information than vector-based data sets. This additional information is made up of several ingredients, among which the most important one is the structural or topological information

T.C. Silva, L. Zhao, *Machine Learning in Complex Networks*,
DOI 10.1007/978-3-319-17290-3_4

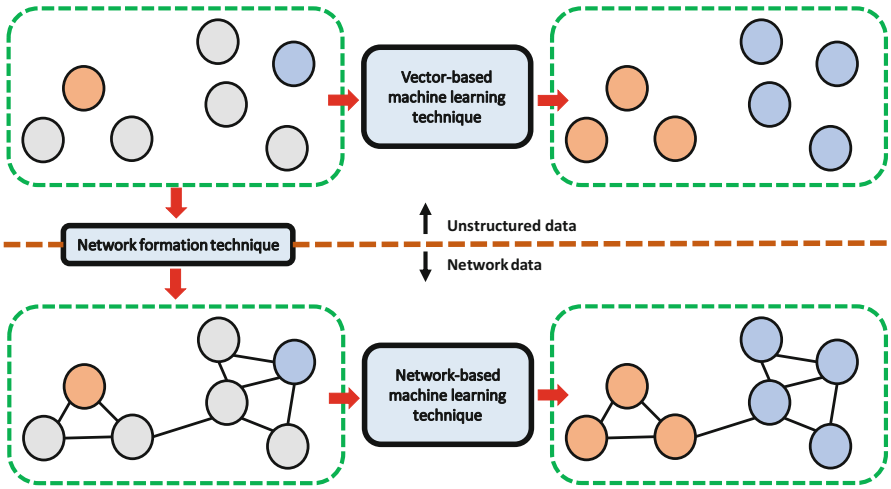

Fig. 4.1 Differences of vector- and network-based machine learning tasks. We illustrate using a semi-supervised learning classification. *Colored* vertices denote labeled data, while *gray* data symbolize unlabeled data. Network formation methods interface between unstructured and structured network data

of the data relationships. In this way, the network topology is able to encode in an elegant manner interactions of the data items in a systematic manner, going from local to global structural information. Thus, a natural question that arises is how can we build networks from unstructured or vector-based data, such that the resulting network encodes as much information as possible? The structure, in principle, must be estimated using the network formation technique based on some heuristics. In this chapter, we discuss the problem of network formation, a task that serves as interface between unstructured and structured network data.

Figure 4.1 illustrates where network formation techniques stand in an overall machine learning scheme. We illustrate using a semi-supervised learning task.[1] First, we see that there is a one-to-one correspondence of $\mathscr{X}$ and $\mathscr{V}$, i.e., each data item in the data set is a vertex in the resulting network. Edges are created using some heuristics that capture similarity among data items. Note these edges that naturally encode similarity are only explicitly modeled in a network environment. Thus, they are estimated by the network formation procedure that interfaces between unstructured and structured network data.

Following our previous notation on general machine learning tasks, given a set of N data points $\mathscr{X} = \{x_1, \ldots, x_N\}$ that is not in a networked format, we can transform it into a network $\mathscr{G}$ that consists of (1) the vertex set $\mathscr{V} = \{v_1, \ldots, v_V\}$ and (2) the

[1]Recall that, in a semi-supervised setting, the goal is to propagate the labels from the labeled to the unlabeled set.

edge set $\mathscr{E}$, which is a subset of $\mathscr{V} \times \mathscr{V}$. The transformation is performed a mapping procedure $g : \mathscr{X} \rightarrow \mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$. We now discuss how the sets $\mathscr{V}$ and $\mathscr{E}$ that compose the network are obtained.

In relation to the vertex set, for the majority of machine learning applications, $\mathscr{V} = \mathscr{X}$ holds, i.e., each data item exactly corresponds to a vertex in the resulting network. To illustrate, in a handwritten digits recognition, each digit in $\mathscr{X}$ would correspond to a vertex in $\mathscr{V}$. Some learning techniques, however, may use reduced or expanded sets of the data items. For example, we can compact several very similar date items in $\mathscr{X}$ into a super-vertex that essentially represents in a summarized manner all of those data items.

We have been using N to denote the cardinality or number of data items in $\mathscr{X}$. In a network setting, the number of vertices in $\mathscr{V}$ is symbolized as $V = |\mathscr{V}|$, which is not necessarily (but often is) equal to N, as we have previously discussed.

Now we discuss how to obtain the set of edges $\mathscr{E}$. We process the decision of establishing or not edges in $\mathscr{E}$ in accordance with two factors:

- *A proper similarity function s*: the similarity function $s : \mathscr{V} \times \mathscr{V} \mapsto \mathbb{R}$ enables us to quantify how different or similar two data items are with respect to their attributes. That is, the similarity function transforms two data items[2] into a scalar value. Applying the similarity function to all of the pairs of vertices, we are able to construct (1) the similarity matrix $\mathbf{S}$, in which $\mathbf{S}_{ij} = s(v_i, v_j)$, where $v_i, v_j \in \mathscr{V}$, or (2) equivalently the dissimilarity matrix $\mathbf{D}_{ij} = d(v_i, v_j)$.
- *A network formation technique*: we decide whether or not to add a link between v_i and v_j by using some rules applied on the similarity matrix $\mathbf{S}$ or on the dissimilarity matrix $\mathbf{D}$.

In this chapter, we discuss these two ingredients that are needed to build up a network from non-networked data. First, we review the theory behind the definition of similarity functions as well as some well-known examples. Following that we show network formation techniques that are applied to vector-based data, in which each data item is represented by a feature vector.[3] After, we deal with the issue of constructing networks from time series. In this case, there is an additional caveat of temporal data dependency that introduces some complexity in the network formation process.

[2]The term data item is used in a wide sense. It may denote feature vectors, time series, graphical objects, among many other types of objects.

[3]Essentially, the representation of data in the format of feature vector covers a wide spectrum of applications in the real world. In image-based applications, for instance, where each data item is symbolized by an image, we can always extract some features from that image and construct a feature vector. Normally, face recognition systems and image processing tools do this for robustness (due to relative uniqueness and invariance between different images) and computational efficiency reasons.

4.2 Similarity and Dissimilarity Functions

The concepts of similarity and dissimilarity are widely employed in the artificial intelligence domain. Among the several fields that they appear, we highlight applications in data mining, information retrieval, pattern matching, genetics, drug discovery, and fuzzy logic [1, 2, 21, 30, 33, 48]. In a general sense, similarity and dissimilarity express a comparison between two elements. Though intuitive, several different formalizations of similarity and dissimilarity exist in the literature. Another prominent characteristic is the duality linking the similarity and dissimilarity concepts, which are opposite terms yet somehow interrelated. This duality also extends to properties of the data items, which could be very useful if properly exploited. Thus, every property of a similarity should have a correspondence with one property of a dissimilarity and vice versa.

Several researches have tried to formalize these concepts but the main properties of similarity or dissimilarity are still under discussion [13, 40]. The lack of basic common theory underlying these two functions leads to incompatible definitions or results. Duality is often neglected and there are few studies about how transformations of similarity to dissimilarity functions can alter their properties [40]. If the similarity function s is well-behaved, one way to calculate its dual, the dissimilarity function d, is:

$$d(x_i, x_j) = \sqrt{s(x_i, x_i) + s(x_j, x_j) - 2s(x_i, x_j)}, \tag{4.1}$$

in which x_i and x_j are two arbitrary data items.

Similarity measures are peculiar kinds of indicators that are mainly descriptive coefficients and not estimators of some statistical parameter. We note that it is difficult to give reliable confidence intervals for most measures of similarity and probable errors can be estimated only by certain types of randomization procedures [28, 49].

4.2.1 Formal Definitions

Similarity and dissimilarity express the degree of coincidence or divergence between two elements of a given domain. Thus, it is reasonable to treat them as functions since the objective is to measure or calculate this value between any two elements of the domain. We first define a similarity function in the following [38].

Definition 4.1. Similarity function: Let $\mathscr{X}$ be a non-empty set where an equality relation is defined. If s is a similarity function, then s is upper bounded, exhaustive, and total, whose domain and range are as follows:

$$s : \mathscr{X} \times \mathscr{X} \longmapsto I_s \subset \mathbb{R}, \tag{4.2}$$

in which I_s is upper bounded by construction, since we assumed s is upper bounded, where $\max_{\mathbb{R}} I_s = s_{\max}$. Moreover, the similarity function s satisfies the following properties:

1. *Reflexivity*: $s(x, x) = s_{\max}$.
2. *Strong reflexivity*: $s(x, y) = s_{\max} \iff x = y$.
3. *Symmetry*: $s(x, y) = s(y, x)$.
4. *Boundedness*[4]: s is lower bounded when $\exists a \in \mathbb{R} : s(x, y) \geq a, \forall x, y \in \mathscr{X}$. This statement is equivalent to affirm that $s = s_{\min} = \min_{\mathbb{R}} I_s$ exists.
5. *Closedness*[5]: This property assures the existence of a lower bound. In special, the closedness property asks for the existence of $x, y \in \mathscr{X} : s(x, y) = s_{\min}$.
6. *Transitivity*: If τ_s is a transitivity operator, then the following must hold: $s(x, y) \geq \tau_s(s(x, z), s(z, y)), \forall x, y, z \in \mathscr{X}$.

We see that s takes as input two data items from $\mathscr{X}$ and outputs a bounded real-valued scalar value. The more similar two objects are, the greater is the similarity value between them. In the next paragraph, we define the concept of dissimilarity or distance function [38].

Definition 4.2. Dissimilarity function: Let $\mathscr{X}$ be a non-empty set where an equality relation is defined. If d is a dissimilarity function, then d is lower bounded, exhaustive, and total, whose domain and range are as follows:

$$d : \mathscr{X} \times \mathscr{X} \longmapsto I_d \subset \mathbb{R}, \tag{4.3}$$

in which I_d is lower bounded by construction, since we assumed d is lower bounded, where $\min_{\mathbb{R}} I_d = d_{\min}$. Moreover, the dissimilarity or distance function d satisfies the following properties:

1. *Reflexivity*: $d(x, x) = d_{\min}$.
2. *Strong reflexivity*: $d(x, y) = d_{\min} \iff x = y$.
3. *Symmetry*: $d(x, y) = d(y, x)$.
4. *Boundedness*: d is upper bounded when $\exists a \in \mathbb{R} : d(x, y) \leq a, \forall x, y \in \mathscr{X}$. This statement is equivalent to affirm that $d = d_{\max} = \max_{\mathbb{R}} I_d$ exists.
5. *Closedness*: This property assures the existence of an upper bound. The closedness property asks for the existence of $x, y \in \mathscr{X} : d(x, y) = d_{\max}$.
6. *Transitivity*: If τ_d is a transitivity operator, then the following must hold: $d(x, y) \leq \tau_d(d(x, z), d(z, y)), \forall x, y, z \in \mathscr{X}$.

[4]Recall that a set $\mathscr{S} \in \mathbb{R}^m$, $m > 0$, is *bounded* if there exists a number B such that $\|x\| \leq B, \forall x \in \mathscr{S}$, that is, if $\mathscr{S}$ is contained in some ball in $\mathbb{R}^m$.

[5]Recall that a set $\mathscr{S} \in \mathbb{R}^m$, $m > 0$, is *closed* if, whenever $\{x_n\}_{n=1}^{\infty}$ is convergent sequence completely contained in $\mathscr{S}$, its limit is also contained in $\mathscr{S}$. For example, the sets $\mathbb{R}^m$ and $\{(x, y) \in \mathbb{R}^2 : xy = 1\}$ are closed but not bounded.

We see that d takes as input two data items from $\mathscr{X}$ and outputs a bounded real-valued scalar value. The more dissimilar are two objects, the greater is the dissimilarity value between them.

Besides those mathematical properties, there are two desirable attributes of all similarity measures. First, the measure should be independent of sample size and of the number of classes in the population [49]. Second, the measure should increase smoothly from some fixed minimum to a fixed maximum, as the samples become more similar. We refer to the researches in [11, 12, 14, 40, 49, 50] for an extensive analysis of the properties of similarity functions.

4.2.2 Examples of Vector-Based Similarity Functions

In this section, we show some traditional examples of similarity/dissimilarity functions. Assume we have a data set $\mathscr{X} = \{x_1, \ldots, x_N\}$ with $N > 1$ data items. Moreover, we characterize each data item with a feature vector $x_i = [x_{i1}, \ldots, x_{iP}]$ with $P > 0$ features or attributes. Data items x_i and x_j are both members of $\mathscr{X}$.

Before we start to explore examples of similarity functions, we give an overall intuition of the types of features or attributes that we can face.

A feature or attribute can be classified as one of the following types:

- *Categorical or nominal attribute*: a categorical feature is one that has two or more categories with no intrinsic ordering. For example, gender is a categorical variable having two categories (male and female) and there is no intrinsic ordering to the categories. Hair color is also a categorical variable having a number of categories (blonde, brown, brunette, red, etc.) and there is no agreed way to order these from highest to lowest. A purely categorical variable is one that simply allows you to assign categories but you cannot clearly order the variables.
- *Ordinal attribute*: An ordinal variable is similar to a categorical variable. The difference between the two is that there is a clear ordering scheme for ordinal variables. For example, suppose you have a variable, economic status, with three categories: low, medium, and high. In addition to being able to classify people into these three categories, you can order the categories as low, medium, and high. Now consider a variable like educational experience with values such as elementary school graduate, high school graduate, some college, and college graduate. These also can be ordered as elementary school, high school, some college, and college graduate.
- *Numerical or quantitative attribute*: The values of a quantitative variable can be ordered and measured. Height and weight are examples of numerical attributes.

Categorical and ordinal attributes are also termed as qualitative attributes, as we cannot numerically operate on them (multiplication and division, for instance,

are not defined). Numerical attributes, in contrast, are classified as quantitative attributes, since mathematical operations can be performed on these types of features.

In the next section, we provide some representative similarity and distance measures. For a comprehensive review, see [33].

4.2.2.1 Numerical Data

In this section, we suppose that the attributes in x_i and x_j are all numerical. They are called feature vectors and have an arbitrary dimension of $P > 0$. The notation $x_i(k)$, $k \in \{1, \ldots, P\}$, indexes the k-th component of the attribute vector x_i. There are a total of N data items.

Definition 4.3. Euclidean distance: The Euclidean distance between x_i and x_j is:

$$d_{\text{Euclidean}}(x_i, x_j) \triangleq \sqrt{\sum_{k=1}^{P} \left[x_i(k) - x_j(k)\right]^2}. \tag{4.4}$$

Definition 4.4. Weighted Euclidean distance: The weighted Euclidean distance between x_i and x_j is:

$$d_{\text{WEuclidean}}(x_i, x_j) \triangleq \sqrt{\sum_{k=1}^{P} W_k \left[x_i(k) - x_j(k)\right]^2}, \tag{4.5}$$

in which W_k denotes the weight given for the k-th attribute.

Remark 4.1. If we give unitary weight for all of the attributes in the feature vector, then the weighted Euclidean distance reduces to the traditional Euclidean distance.

Definition 4.5. Manhattan or city-block distance: The Manhattan or city-block distance between x_i and x_j is:

$$d_{\text{Manhattan}}(x_i, x_j) \triangleq \sum_{k=1}^{P} \left|x_i(k) - x_j(k)\right|. \tag{4.6}$$

Definition 4.6. Chebyshev or supremum distance: The Chebyshev or supremum distance between x_i and x_j is:

$$d_{\text{Supremum}}(x_i, x_j) \triangleq \max\left(\left|x_i(1) - x_j(1)\right|, \ldots, \left|x_i(P) - x_j(P)\right|\right). \tag{4.7}$$

Definition 4.7. Minkowski distance (L_λ metric): The Minkowski distance or L_λ metric, $\lambda \geq 1$, between x_i and x_j is:

$$d_{\text{Minkowski}}(x_i, x_j) \triangleq \left[\sum_{k=1}^{P} \left| x_i(k) - x_j(k) \right|^{\lambda} \right]^{\frac{1}{\lambda}}. \tag{4.8}$$

Remark 4.2. The family of Minkowski functions is obtained by varying λ over 1 to ∞. The Minkowski distance is a generalization of the previous discussed metrics. We list them in the following:

- L_1 metric: Manhattan or city-block distance as in Definition 4.5.
- L_2 metric: Euclidean distance as in Definition 4.3.
- L_∞ metric: Chebyshev or supremum distance as in Definition 4.6.

Definition 4.8. Mahalanobis distance: The Mahalanobis distance between x_i and x_j is:

$$d_{\text{Mahalanobis}}(x_i, x_j) \triangleq \sqrt{\sum_{k=1}^{P} \left(x_i - x_j \right)^T \boldsymbol{\Sigma}^{-1} \left(x_i - x_j \right)}, \tag{4.9}$$

in which $\boldsymbol{\Sigma}$ is the $P \times P$ sample covariance matrix, whose (i,j)-th entry, $\boldsymbol{\Sigma}_{ij}$, is given by:

$$\boldsymbol{\Sigma}_{ij} \triangleq \frac{1}{N-1} \sum_{k=1}^{N} \left(x_i(k) - \bar{x}_i \right) \left(x_j(k) - \bar{x}_j \right), \tag{4.10}$$

in which $\bar{x}_i$ and $\bar{x}_j$ are the sample means that in turn are expressed as:

$$\bar{x}_i \triangleq \frac{1}{N} \sum_{k=1}^{N} x_i(k). \tag{4.11}$$

In the next example, we provide the intuition behind the definition of the Mahalanobis distance.

Example 4.1. Consider the problem of estimating the probability that a test point in an Euclidean space belongs to a set of training data points. A natural first step would be to find the average or center of mass of these training data points. Intuitively, the closer the test point in question is to this center of mass, the more likely it is to belong to the set.

A simple refinement would be to quantify if the set is spread out over a large or a small range, so that we can decide whether a given distance from the center is noteworthy or not. The simplistic approach is to estimate the standard deviation of the distances of the sample points from the center of mass. If the distance between

the test point and the center of mass of the training set is less than one standard deviation, then we might conclude that it is highly probable that the test point belongs to the set of training data points. The further away it is, the more likely that the test point should not be classified as belonging to the set.

This intuitive approach can be made quantitative by defining the normalized distance between the test point and the set of training data points as $\frac{x-\mu}{\sigma}$, where μ is the sample average and σ, the sample standard deviation of the set. By plugging these values into the normal distribution, we can derive the probability of the test point belonging to the set.

The drawback of the above approach is that we assume that the points in the training data set are distributed around the center of mass in a spherical manner. If we were dealing with a non-spherical distribution, such as the ellipsoidal distribution, then we would expect the membership probability of that test point to depend not only on the distance from the center of mass, but also on the direction. In those directions in which the ellipsoid has a short axis, the test point is expected to be closer if it is really a member of that set, while in those directions where the axis has large amplitude, the test point can be further away from the center.

Putting this on a mathematical basis, the ellipsoid that best represents the probability distribution of the training data set can be estimated by building the covariance matrix of the samples. The Mahalanobis distance is simply the distance of the test point from the center of mass divided by the width of the ellipsoid in the direction of the test point. This behavior is mathematically represented by (4.9).

Definition 4.9. Gaussian kernel similarity (radial basis function or heat kernel): The Gaussian kernel similarity between x_i and x_j is:

$$s_{\text{Gaussian}}(x_i, x_j) \triangleq a \ \exp\left(-\frac{\| x_i - x_j \|^2}{2\sigma^2}\right), \tag{4.12}$$

in which $\sigma > 0$ is the variance of bandwidth of the Gaussian function and a is scaling constant. The term $\| x_i - x_j \|$ is the Euclidean norm between x_i and x_j.

Definition 4.10. Harmonic mean similarity: The harmonic mean similarity between x_i and x_j is:

$$s_{\text{Harmonic}}(x_i, x_j) \triangleq 2 \sum_{k=1}^{P} \frac{x_i(k) x_j(k)}{x_i(k) + x_j(k)}. \tag{4.13}$$

Definition 4.11. Cosine similarity: The cosine similarity between x_i and x_j is:

$$s_{\text{Cosine}}(x_i, x_j) \triangleq \frac{\sum_{k=1}^{P} x_i(k) x_j(k)}{\|x_i\| \|x_j\|} = \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|}, \tag{4.14}$$

in which $\langle ., . \rangle$ denotes the inner product operator and $\|.\|$ is the Euclidean norm.

Definition 4.12. Pearson correlation similarity: The Pearson correlation similarity between x_i and x_j is:

$$\begin{aligned} s_{\text{Pearson}}(x_i, x_j) &\triangleq \frac{\sum_{k=1}^{P}(x_i(k)-\bar{x}_i)(x_j(k)-\bar{x}_j)}{\|x_i-\bar{x}_i\|\|x_j-\bar{x}_j\|} \\ &= \frac{\langle x_i-\bar{x}_i, x_j-\bar{x}_j\rangle}{\|x_i-\bar{x}_i\|\|x_j-\bar{x}_j\|} \\ &= s_{\text{Cosine}}(x_i-\bar{x}_i, x_j-\bar{x}_j). \end{aligned} \tag{4.15}$$

Remark 4.3. Cosine similarity can be applied to deal with document similarity and image similarity. It should be noted that cosine similarity is affected by vector translation. The Pearson correlation similarity, however, addresses this problem by being translation-invariant due to the demeaning process. In addition, cosine and Pearson correlations are scale-invariant due to the normalization process.

Definition 4.13. Dice similarity [36]: The Dice similarity between x_i and x_j is:

$$s_{\text{Dice}}(x_i, x_j) \triangleq \frac{2\sum_{k=1}^{P} x_i(k)x_j(k)}{\sum_{k=1}^{P} x_i(k)^2 + x_j(k)^2}. \tag{4.16}$$

Definition 4.14. Kumar-Hassebrook similarity [29]: The Kumar-Hassebrook similarity between x_i and x_j is:

$$s_{\text{KH}}(x_i, x_j) \triangleq \frac{\sum_{k=1}^{P} x_i(k)x_j(k)}{\sum_{k=1}^{P} x_i(k)^2 + x_j(k)^2 - x_i(k)x_j(k)}. \tag{4.17}$$

4.2.2.2 Categorical Data

In this section, we deal with categorical data. We consider that each entry of the P-dimensional feature vectors x_i and x_j can assume either a present or an absent value (dichotomous feature). If there are multiple categories, we define the category of interest as present, while all of the others are considered to be in the absent class. Therefore, when comparing the vectors x_i and x_j, we can face four different scenarios that are delineated in Table 4.1. We see that:

- M_{11} denotes the number of occurrences of coincident present values in x_i and x_j.
- M_{01} is the number of occurrences in which there is an absence in x_i and a presence in x_j.
- M_{10} represents the number of occurrences in which there is a presence in x_i and an absence in x_j.
- M_{00} symbolizes the number of occurrences of coincident absent values in x_i and x_j.

Table 4.1 Possible outcomes when comparing two entries of categorical data

		x_j	
		Present	Absent
x_i	Present	M_{11}	M_{10}
	Absent	M_{01}	M_{00}

Definition 4.15. Hamming distance: The Hamming distance between x_i and x_j is

$$d_{\text{Hamming}}(x_i, x_j) \triangleq \sum_{k=1}^{P} \mathbb{1}_{[x_i(k) \neq x_j(k)]} = M_{01} + M_{10}, \tag{4.18}$$

i.e., the Hamming distance is defined as the minimum number of replacements that are needed to transform x_i into x_j.

Definition 4.16. Jaccard similarity [26]: The Jaccard similarity between x_i and x_j is

$$s_{\text{Jaccard}}(x_i, x_j) \triangleq \frac{M_{11}}{M_{11} + M_{01} + M_{10}}. \tag{4.19}$$

Remark 4.4. In the Hamming distance, each value is equally important. In some applications, however, it may be interesting to give more importance to some classes in detriment to others. In light of that, suppose we have a problem where there is a feature vector of P movies and we want to compute the similarity of movie taste of two persons. In this case, if there are several movies that they have not watched, we can not say two persons are similar simply because none of them watched any movies in the feature vector. In contrast, if these two persons have seen a significant quantity of common watched movies, we can say they are similar to some extent. That is, we give more weight to those entries in which both persons have mutually watched the film in detriment to other configurations. This is a kind of weighted Hamming distance that is known as Jaccard similarity.

Definition 4.17. Sørensen similarity [45]: The Sørensen similarity between x_i and x_j is

$$s_{\text{Sørensen}}(x_i, x_j) \triangleq \frac{2M_{11}}{2M_{11} + M_{01} + M_{10}}. \tag{4.20}$$

Remark 4.5. As opposed to Jaccard similarity, the matches in the Sørensen similarity are given more importance than mismatches. Deciding between the two is a matter of the type of data we have at hand. If many entries are present in the population but are not present in the sample, it may be useful to use Sørensen coefficient rather than Jaccard.

Definition 4.18. Simple matching similarity: The simple matching similarity between x_i and x_j is

$$s_{\text{SM}}(x_i, x_j) \triangleq \frac{M_{11} + M_{00}}{M_{11} + M_{00} + M_{01} + M_{10}}. \tag{4.21}$$

Remark 4.6. Simple matching similarity is a good option to choose when absent and present values are equally valuable in the data.

Definition 4.19. Baroni-Urbani and Buser similarity [4]: The Baroni-Urbani and Buser similarity between x_i and x_j is

$$s_{\text{BUB}}(x_i, x_j) \triangleq \frac{\sqrt{M_{11}M_{00}} + M_{11}}{\sqrt{M_{11}M_{00}} + M_{11} + M_{01} + M_{10}}. \tag{4.22}$$

Remark 4.7. The square root term in (4.22) is introduced to help in removing the size bias common in other similarity measures, such as the Jaccard.

4.3 Transforming Vector-Based Data into Networks

Given the similarity matrix $\mathbf{S}$ or the dissimilarity matrix $\mathbf{D}$, one direct approach of building a network would be to establish links between pairs of vertices with weights according to $\mathbf{S}_{ij}$ or, equivalently, to some function on the reciprocal of $\mathbf{D}_{ij}$. This approach would frequently lead to the emergence of almost complete networks. Generally speaking, a good network satisfies the following criteria: (1) it should possess a giant component to mantain the vertices connected; (2) it should be as sparse as possible in order to better reveal the relationships between the vertices. The existence of links with very small weights, however, may lead to poor results if used by network-based algorithms. Sparsification, hence, is important because it leads to improved efficiency in the learning stage, better accuracy, and robustness to noise. We can think of small-valued links as noises that would just jeopardize the learning process, providing misleading information to the machine learning algorithm by connecting two distant vertices. Therefore, the resulting network topology would be largely distorted by these noisy links. The removal of these links stands as an important pre-processing step for enhancing the efficiency of network-based learning algorithms.

Following that reasoning, the two most traditional types of nearest neighbor networks that sparsify the similarity or dissimilarity matrices are [5]:

- *k-nearest neighbors network* (k-NN): this is, in general, a directed network. An edge from v_i to v_j exists if and only if v_j is among the k most similar elements to v_i. In computational terms, we have to sort, in an independent manner, all of the rows of $\mathbf{D}$ in ascending order. Once sorted, given a row i, links are established

among vertex i and the first k entries that are in the sorted list corresponding to the elements in the i-th row of $\mathbf{D}$.

- *ϵ-radius network*: this is an undirected network whose edge set consists of pairs (v_i, v_j) such that $\mathbf{D}_{ij} \leq \epsilon$, where $\epsilon \in \mathbb{R}_+$.

The k-nearest neighbors network is, in general, a directed network because v_j can be one of the k nearest neighbors of v_i, but the converse may not be true. In contrast, the ϵ-radius network is by construction an undirected network, because $\mathbf{D}_{ij} = \mathbf{D}_{ji}$, as we evaluate each entry of the distance matrix using a distance function. In this way, if $\mathbf{D}_{ij} < \epsilon$, it must be the case that $\mathbf{D}_{ji} < \epsilon$. Hence, the existence of links always occurs in both directions.

The k-nearest neighbors and the ϵ-radius techniques are considered as static network formation methods. This is because they treat in a uniform manner data items that are in dense and sparse regions. We now list a set of network formation techniques that employ adaptive or dynamical information:

- *Network formation using combinations of the k-nearest neighbors and ϵ-radius techniques* [43, 44]: we can devise a network formation technique that employs both heuristics based on one or more criteria. For instance, we can activate the k-nearest neighbors network when we are at sparse regions. Conversely, we can employ the ϵ-radius technique in dense regions.
- *b-matching network* [27]: As opposed to the k-NN network, the b-matching network ensures that each vertex in the graph has the same number of edges and therefore produces a balanced or regular graph. It relies on an optimization process.
- *Linear neighborhood network* [47]: the idea is to approximate the entire network by a series of overlapped linear neighborhood patches, and the edge weights in each patch are determined by a standard quadratic programming procedure. The initial neighborhoods of the vertices are set in a static way. Then, the linear embedding makes dynamical adjustments in the edge weights.
- *Relaxed linear neighborhood network* [10]: it approximates the entire network by a series of overlapped linear neighborhood patches, where the neighborhood of any vertex is captured dynamically based on the density/sparsity of its surrounding. Moreover, the relaxed linear neighborhood technique explores the degree of neighborhood during the reconstruction method rather than using fixed assignments. As a consequence, it does not get affected by outliers, producing networks that are more robust.
- *Network formation using clustering heuristics* [15]: this method uses data clustering heuristics to perform the network formation process. Specifically, it employs the single-link method, which is a clustering heuristic that is capable of constructing connected and sparse networks, while also maintaining the cluster structure of the original data set.
- *Network formation using overlapping histogram segments* [42]: this technique uses overlapping histogram segments to perform the network construction. The resulting network always produces a connected graph with vertices of the same community densely interconnected and with vertices of different commu-

nities sparsely interconnected. In essence, it is based on the k-NN technique, but with adaptive k values that are learned from the data distribution.

In the following section, we discuss in detail each of the mentioned techniques.

4.3.1 Analysis of k-Nearest Neighbors and ϵ-Radius Networks

We supply in Fig. 4.2a, b the geometrical intuition of the k-nearest neighbors and the ϵ-radius networks, respectively. In the k-NN, once set a reference vertex, we simply sort all of the remainder vertices in accordance with the selected distance function, thus ranking those vertices. With the selected parameter k, we only establish links with those vertices that are ranked below that threshold k. In Fig. 4.2a, note that $k = 2$. Now, in the ϵ-radius network formation technique, we only establish links to those vertices whose similarities to the reference vertex are within the threshold ϵ. A noticeable difference between these two techniques is in the number of links that emerges from a reference vertex. In the ϵ-radius technique, the number of links is not pre-determined: the network formation process keeps establishing links as long as there are vertices within the range ϵ. Both techniques have their advantages and shortcomings that we discuss further.

Parameters k and ϵ play an important role in transforming the raw data into a corresponding network, since these parameters are sensitive to the local structure of the data. Thus, depending on the choices of k and ϵ, the resulting network topology may not reliably maintain the properties of the underlying data distribution.

We first discuss that caveat for k-NN. When k is large enough, it forces the creation of links between pairs of vertices that are not similar at all. To give an example, consider the data in Fig. 4.3a, in which we build a network with $k = 3$. Even though it is apparent that two well-behaved clusters exist in the data, the reference vertex is forced to connect to distant members of the other cluster. Conversely, if we choose k too small, we may sparsify the link structure in such a way that clusters are not formed in regions where the underlying data distribution seems to have real clusters. As an example, consider Fig. 4.3b, where we want to

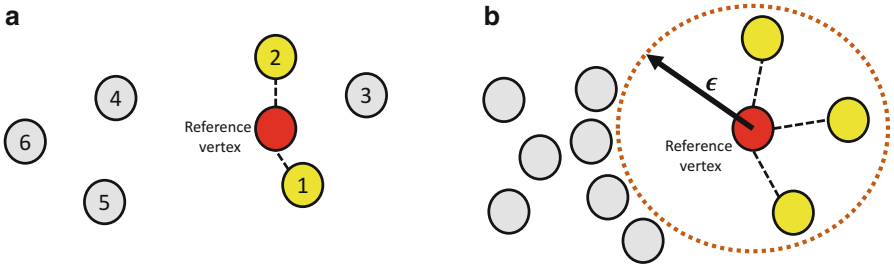

Fig. 4.2 Geometrical intuition of the k-nearest neighbor and ϵ-radius techniques, (**a**) 2-nearest neighbors, (**b**) ϵ-radius

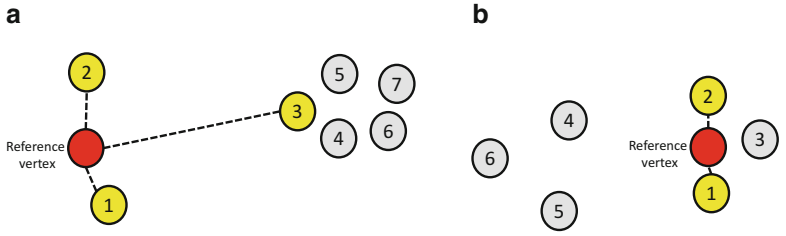

Fig. 4.3 Limitations of the k-nearest neighbors network formation technique. (**a**) Too large k ($k = 3$). (**b**) Too small k ($k = 2$)

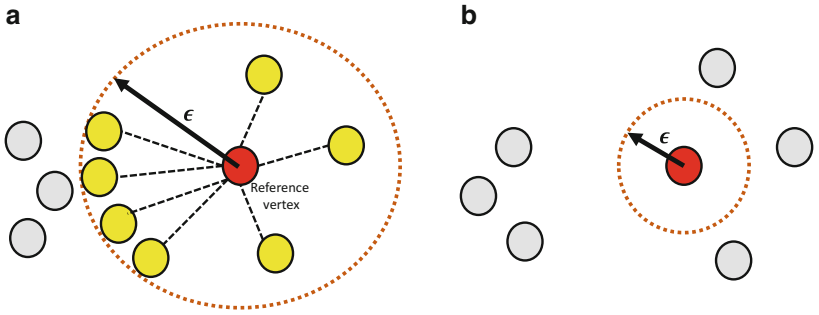

Fig. 4.4 Limitations of the ϵ-radius network formation technique. (**a**) Too large ϵ. (**b**) Too small ϵ

build a network with $k = 2$. Now, we have the opposite picture: the k-NN breaks into two apparent well-behaved clusters due to the small value of k. Note also that, provided that $k > 0$, no singletons[6] will arise in the network.

As we have pointed out, parameter ϵ also plays a major role in correctly translating the raw data into a proper network topology. Depending on the data distribution, a small increase on the value of ϵ may increase the network density to a large extent. Therefore, the network topology is very sensitive to the selected value of ϵ. Consider Fig. 4.4a, in which we employ the ϵ-radius to construct a network with a large ϵ. If we slightly reduce the value of ϵ, we get the network portrayed in Fig. 4.2b. We see here that a small increase on ϵ is responsible for an explosive increase in the number of connections established by the reference vertex. Conversely, if we choose a small value for ϵ, we may end up getting singletons in the network, as we can see in Fig. 4.4b. In both extremes, the resulting network may not represent well the true data distribution.

Some studies point that the network constructed using k-nearest neighbors and ϵ-radius techniques have dramatic influences on clustering techniques [34]. The k-NN network remains the more common approach since it is more adaptive to scale and data density.

[6]Singletons are those vertices that do not have connections to other vertices.

4.3.2 Combination of k-Nearest Neighbors and ϵ-Radius Network Formation Techniques

Recall that the ϵ-radius technique creates a link between two vertices if they are within a distance ϵ, while the k-NN sets up a link between vertices i and j if i is one of the k nearest neighbors of j. Both approaches have their limitations when sparsity or density is a concern. For sparse regions, the k-NN forces a vertex to connect to its k nearest vertices, even if they are far apart. In this scenario, one can say that the neighborhood of this vertex would contain dissimilar points. Equivalently, improper ϵ values could easily result in disconnected components, subgraphs, or isolated singleton vertices.

In order to combine the strengths of both approaches, a suitable combination (among many others) is given as follows. If $\mathcal{N}(v_i)$ denotes the neighborhood of v_i, then:

$$\mathcal{N}(v_i) = \begin{cases} \epsilon\text{-radius}(v_i), & \text{if } |\epsilon\text{-radius}(v_i)| > k \\ k\text{-NN}(v_i), & \text{otherwise} \end{cases} \tag{4.23}$$

in which $\epsilon{-}radius(v_i)$ returns the set $\{v_j, j \in \mathcal{V} : \mathbf{D}_{ij} <= \epsilon\}$, and k-NN(v_i) returns the set containing the k nearest of vertex v_i. Note that the ϵ-radius technique is used for dense regions ($|\epsilon$-radius$(v_i)| > k$), while the k-NN is employed for sparse regions.

Figure 4.5 portrays how the combination of k-NN and ϵ-radius can transform vector-based data into networked data. In the example, we use $k = 3$ and desire to evaluate the neighbors of the colored vertices. Recall that k is employed as a threshold when defining whether vertices are located in sparse or dense regions. Whenever the number of neighbors within a radius ϵ of a given reference vertex is

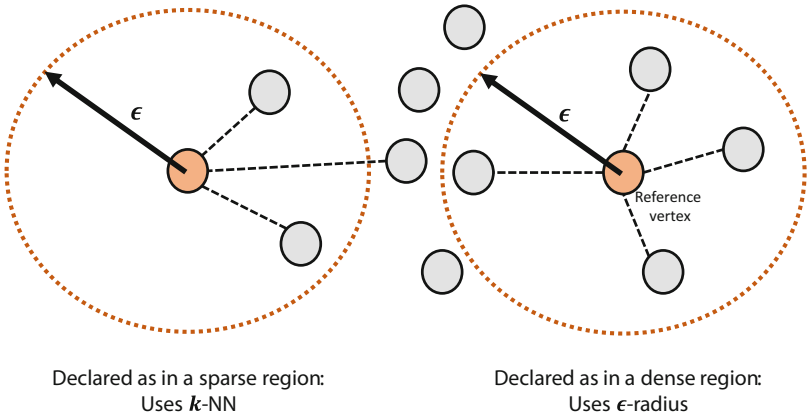

Fig. 4.5 Combination of the k-NN and ϵ-radius technique as a network formation technique. In the example, we use $k = 3$ and the ϵ is geometrically depicted within the figure

smaller than k, we use the k-NN, which effectively forces the reference vertex to find more distant neighbors. When the number of neighbors is larger than k, then we declare the reference vertex as in a dense region. In this case, it connects to all of the neighbors within the radius ϵ, as governed by the ϵ-radius technique. One nice property of this network formation technique is that it adapts the heuristics of network formation depending on local density of data items. As it uses the k-NN in sparse regions, the combination of k-NN and ϵ-radius methods as a network formation technique tend to prevent the emergence of many network components.[7]

4.3.3 *b-Matching Networks*

The b-matching method is introduced in [27]. As opposed to the k-NN network, the b-matching network ensures that each vertex in the network has the same number of edges and therefore produces a balanced or regular network.

For a non-weighted adjacency matrix $\mathbf{A}$, the b-matching network formation technique relies on an optimization framework as follows:

$$\begin{aligned} &\min_{\mathbf{A}} \sum_{i,j \in \mathscr{V}} \mathbf{A}_{ij}\mathbf{D}_{ij} \\ &s.t. \sum_{j \in \mathscr{V}} \mathbf{A}_{ij} = b, \\ &\mathbf{A}_{ii} = 0, \\ &\mathbf{A}_{ij} = \mathbf{A}_{ji}, \end{aligned} \tag{4.24}$$

$\forall i, j \in \mathscr{V}$. $\mathbf{D}$ denotes the dissimilarity matrix in which $\mathbf{D}_{ij}$ represents the dissimilarity between the i-th and j-th feature vectors. We can use any of the discussed metrics in Sect. 4.2 to construct $\mathbf{D}$. Note that the search space is on all binary matrices with the same dimension as of $\mathbf{A}$. The first restriction forces every vertex to link to exactly b other vertices. The second constraint prevents the emergence of self-loops. The third restriction guarantees that the resulting network is symmetric.

The k-NN network formation technique can also be stated as an optimization framework similar to that in (4.24), except for the third restriction. At the end of the network formation, the k-NN technique may not necessarily construct symmetric matrices. Though the out-degree of each of the vertices matches parameter k due to the first restriction, the in-degree varies (is at least k). This arises due to the asymmetrical behavior of the k-NN technique, in the sense that we can have vertex j as one of the k-nearest neighbors of vertex i and the converse may not hold. By introducing the third restriction, we are effectively forcing symmetry in the

[7] Note, however, that more than one network component may arise when k is small.

adjacency matrix. Consequently, the out- and in-degree are exactly equal to b (regular network).

The optimization framework described in (4.24) can be efficiently implemented using the algorithm of loopy belief propagation [25].

4.3.4 Linear Neighborhood Networks

The linear neighborhood network is introduced in [47]. The building block of the linear neighborhood network is based on the locally linear embedding technique, which we first explore before going to the referred network formation technique.

Locally linear embedding (LLE) is a dimensionality reduction technique [39]. The motivation behind the introduction of this class of methods is as follows. Our mental representations of the world are formed by processing large numbers of sensory inputs. Pixel intensities of images, power spectra of sounds, and the joint angles of articulated bodies are clear examples of these inputs. While complex stimuli can be represented by points in a high-dimensional vector space, they typically have a much more compact description. Coherent structure in the world leads to strong correlations between inputs (such as between neighboring pixels in images), generating observations that lie on or close to a smooth low-dimensional manifold. The dimensionality reduction problem involves mapping high-dimensional inputs into a low-dimensional summarizing space with as many coordinates as observed modes of variability. Dimensionality reduction techniques are widely employed in real-world applications, such as in the pre-processing of images of faces, spectrograms of speech, and other multidimensional signals in general. In essence, they are able to compress the signals in size and to discover compact representations of their variability.

The LLE algorithm is based on simple geometric intuitions. Suppose the data consist of V real-valued vectors x_i,[8] each with dimension P, sampled from some smooth underlying manifold. Provided there are sufficient data (such that the manifold is well-sampled), we expect that each data point and its neighbors will lie on or close to a locally linear patch of the manifold.

Instead of using pairwise relations between data items such as the k-NN and the ϵ-radius network formation techniques, the linear neighborhood network technique [47] uses locally linear embedding in the network formation step. Thus, the algorithm employs neighborhood information of the data items when establishing links. Therefore, each data point is optimally reconstructed using a linear combination of its neighbors [39]. With this simplification, we define the network formation process in terms of a constrained optimization process, whose goal is to minimize the following objective function:

[8] Each data item here exactly corresponds to a vertex, so that $N = V$.

$$C(\mathbf{A}) = \sum_{i \in \mathcal{V}} \left|\left| x_i - \sum_{j \in \mathcal{N}(x_i)} \mathbf{A}_{ij} x_j \right|\right|^2, \tag{4.25}$$

in which $\mathbf{A}_{ij}$ is the contribution or edge weight of x_j to x_i in the weighted adjacency matrix $\mathbf{A}$ of the network, $\mathcal{N}(x_i)$ represents the neighborhood of x_i, and $\|.\|$ is the Euclidean norm. The initial neighborhoods of all of the vertices are set in a static way. For instance, we can use k-nearest neighbors or ϵ-radius techniques.

Note that we can rewrite (4.25) in terms of the individual contributions of each vertex to the objective function, as follows:

$$C(\mathbf{A}) = \sum_{i \in \mathcal{V}} C_i(\mathbf{A}), \tag{4.26}$$

in which:

$$C_i(\mathbf{A}) = \left|\left| x_i - \sum_{j \in \mathcal{N}(x_i)} \mathbf{A}_{ij} x_j \right|\right|^2, \tag{4.27}$$

In the optimization process, we apply the following constraints:

$$\begin{aligned} \sum_{j \in \mathcal{N}(x_i)} \mathbf{A}_{ij} = 1, \forall i \in \mathcal{V} \\ \mathbf{A}_{ij} \geq 0, \quad i, j \in \mathcal{V} \end{aligned} . \tag{4.28}$$

The weight $\mathbf{A}_{ij}$ increases as x_j and x_i become more similar. In the extreme case, when $x_i = x_k \in \mathcal{N}(x_i)$, then $\mathbf{A}_{ik} = 1$ and $\mathbf{A}_{ij} = 0, j \neq k$, $x_j \in \mathcal{N}(x_i)$, is the optimal solution. Thus, we can employ $\mathbf{A}_{ij}$ to measure the similarity of x_j to x_i. As the neighborhoods of x_j and x_i may differ, then $\mathbf{A}_{ij} \neq \mathbf{A}_{ji}$ may hold in the general case. Applying some algebraic manipulations on (4.27), we see that:

$$\begin{aligned} C_i(\mathbf{A}) &= \left|\left| x_i - \sum_{j \in \mathcal{N}(x_i)} \mathbf{A}_{ij} x_j \right|\right|^2 \\ &= \left|\left| \sum_{j \in \mathcal{N}(x_i)} \mathbf{A}_{ij} (x_i - x_j) \right|\right|^2 \\ &= \sum_{j,k \in \mathcal{N}(x_i)} \mathbf{A}_{ij} \mathbf{A}_{ik} (x_i - x_j)^T (x_i - x_k) \\ &= \sum_{j,k \in \mathcal{N}(x_i)} \mathbf{A}_{ij} \mathbf{G}_{jk}^i \mathbf{A}_{ik}, \end{aligned} \tag{4.29}$$

in which $\mathbf{G}^i_{jk}$ represents the (j,k)-th entry of the local Gram matrix

$$\mathbf{G}^i_{jk} = (x_i - x_j)^T (x_i - x_k) \tag{4.30}$$

at point x_i. Thus, the reconstruction weights of each data item can be solved by the following standard quadratic programming problems:

$$\begin{aligned} &\min_{\mathbf{A}} \sum_{j,k \in \mathscr{N}(x_i)} \mathbf{A}_{ij} \mathbf{G}^i_{jk} \mathbf{A}_{ik} \\ &\text{s.t.} \sum_{j \in \mathscr{N}(x_i)} \mathbf{A}_{ij} = 1, \mathbf{A}_{ij} \geq 0, \forall i \in \mathscr{V} \end{aligned} \tag{4.31}$$

Intuitively, the way we construct the entire network $\mathbf{A}$ is to first shear the whole network into a series of overlapped linear patches, and then paste them together.

4.3.5 *Relaxed Linear Neighborhood Networks*

This method is introduced in [9]. It is an extension of the linear neighborhood network discussed in the previous section.

The noticeable advantage of this method is that it uses dynamic neighborhood information, as opposed to fixed k neighbors of [47]. In summary, the technique approximates the entire network by a series of overlapped linear neighborhood patches, where the neighborhood $\mathscr{N}(x_i)$, $\forall x_i \in \mathscr{V}$, is captured dynamically via the data density in the vicinities.

Instead of finding fixed k neighbors of each vertex x_i, the relaxed linear neighborhood method captures the boundary of each vertex $\mathscr{B}(x_i)$ based on neighborhood information and declares as neighbors vertices within this boundary. We can capture this dynamic feature by using a combination of the k-NN and ϵ-radius approaches to define the neighborhood between any x_i and x_j as:

$$\mathscr{N}_{x_i;k,\epsilon}(x_j) = \begin{cases} 1, & |\mathscr{N}_\epsilon(x_i)| > k \\ \mathscr{N}_{x_i;k}(x_j), & \text{otherwise} \end{cases}, \tag{4.32}$$

in which $|\mathscr{N}_\epsilon(x_i)|$ denotes the number of neighbors if we apply the ϵ-radius technique in x_i, and $\mathscr{N}_{x_i;k}(x_j) \in \{0,1\}$ returns 1 if x_j in is the k-nearest neighborhood of x_i, and 0 otherwise. Thus if there is a large enough number of vertices in the ϵ-vicinity ($> k$), then the boundary is identified. Otherwise, we employ k-NN. We define the boundary set of any x_i as:

$$\mathscr{B}(x_i) = \left\{ j \in \mathscr{V} : \mathscr{N}_{x_i;k,\epsilon}(x_j) = 1 \right\}. \tag{4.33}$$

It must be noted that we may run into problems if we only consider the established radius and density of the neighborhood. For instance, if we fix large values for k and ϵ, vertices located at dense regions would include more vertices than necessary. Conversely, for small values of k and ϵ, weak neighborhood bonds would be established in sparse regions. An adaptive algorithm that can handle a wide range of changing interval would be advantageous. It should also include information provided by neighboring vertices closest to the corresponding vertex, which can take neighborhood relations into consideration in a more intelligent way. One way to accomplish that is to extend the neighborhood definitions in (4.32) and (4.33) and account for the data sensitivity with varying distances to neighbor points based on the parameter $k > 0$:

$$N_{x_i}(x_j) = \max\left[1 - k\frac{d(x_i, x_j)}{d_{\max}}, 0\right], \tag{4.34}$$

in which $d_{\max}$ is the network diameter whose definition we recall:

$$d_{\max} = \max_{x_i, x_j \in \mathscr{V}} d(x_i, x_j), \tag{4.35}$$

and $d(x_i, x_j)$ is a suitable dissimilarity or distance function, such as those defined in Sect. 4.2. Parameter k plays the role in determining the neighborhood radius and is adjusted as follows:

$$1 - k\frac{\epsilon}{d_{\max}} = 0 \Rightarrow k = \frac{d_{\max}}{\epsilon}. \tag{4.36}$$

The new boundary set of any given x_i includes:

$$\mathscr{B}(x_i) = \left\{x_j \in \mathscr{V} : N_{x_i}(x_j) \in (0, 1]\right\}. \tag{4.37}$$

Instead of measuring pairwise relations, neighborhood information to represent the network is employed. Similarly to the studies in [39, 47], each vertex is reconstructed using a linear combination of its dynamic neighbors:

$$\begin{aligned} &\min_{\mathbf{A}} \sum_{i \in \mathscr{V}} \left\| x_i - \sum_{j\,:\,x_j \in \mathscr{N}(x_i)} N_{x_i}(x_j)\mathbf{A}_{ij}x_j \right\|^2 \\ &\text{s.t.} \sum_{j \in \mathscr{V}} \mathbf{A}_{ij} = 1, \mathbf{A}_{ij} \geq 0, \forall i \in \mathscr{V}, \end{aligned} \tag{4.38}$$

in which $N_{x_i}(x_j) \in [0, 1]$ is the degree of neighborhood to the boundary set $\mathscr{B}(x_i)$ and $\mathbf{A}_{ij}$ is the degree of contribution of x_j to x_i. When $N_{x_i}(x_j) = 0$, no links exist.

4.3.6 *Network Formation Using Clustering Heuristics*

This network formation technique is introduced in [15]. Recall that k-NN and ϵ-radius network formation methods generate either disconnected or densely connected networks. Neither of the two situations is desirable for the majority of data mining and machine learning tasks. For example, Fig. 4.6 shows a data set with 300 data samples and the resulting networks generated using the k-NN method with various values of k. Note that the generated networks become connected only when $k \geq 33$. This means that sometimes we need a very large k to generate a connected network. In these cases, the generated networks are dense, which in turn can cause: inefficiency in the processing procedure that leads to high computational time; the weak performance of the algorithm due to the homogeneity of the network or even both.

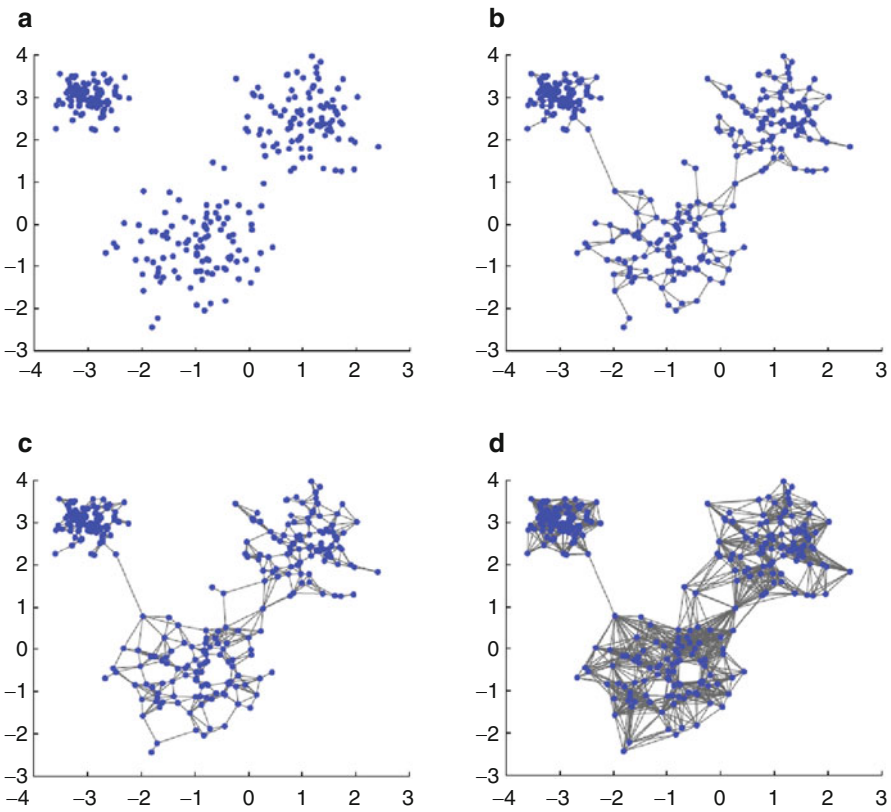

Fig. 4.6 Networks formed by the k-NN in a data set with three distinct groups. (**a**) Original network with 300 vertices. Results for k-NN when (**b**) $k = 5$, (**c**) $k = 20$, and (**d**) $k = 33$. Reproduced from [31] with permission from the author

In [15], the authors proposed a method that is based on data clustering heuristic to perform the network formation process. Specifically, they employ the single-link method to overcome the aforementioned problem that happens with traditional network formation techniques. The clustering heuristics for network formation is capable of constructing connected and sparse networks and, at the same time, tends to keep the cluster structure of the original data set. The method consists of the following steps:

1. Generate an initial totally disconnected network, where each vertex represents a data instance. In this way, we have V vertex groups (each vertex is in an isolated group).
2. Construct a dissimilarity matrix using any distance measure, for example, the Euclidean distance, to represent distances between all pairs of groups. According to the single-linkage criteria, the dissimilarity between two groups is computed as the dissimilarity between the two closest vertices.
3. Identify the two closest groups and denote them by G_1 and G_2, respectively.
4. Calculate the average dissimilarity among vertices (data instances) within each group G_1 and G_2, and denote them by d_1 and d_2, respectively.
5. Select the k-most similar pairs of vertices between G_1 and G_2, and generate an edge between each of the k selected pairs if their dissimilarity is smaller than the threshold: $d_c = \lambda \max(d_1, d_2)$, where $\lambda > 0$.
6. Update the dissimilarity matrix considering the merging result in Step 5.
7. If the number of groups is larger than one, return to Step 3;

The condition in Step 7 guarantees that the final network is connected.

In order to illustrate the effectiveness of the method, Fig. 4.7 shows the resulting networks for an artificial data set composed of three clusters of different sizes and densities. In this simulation, the following values for k are used: 3, 5 and 20. We see that the generated networks are connected and relatively sparse. At the same time, the original cluster features are well preserved. Figure 4.8 shows the network construction results by varying the threshold parameter λ. Again, we see the good performance of the method.

4.3.7 Network Formation Using Overlapping Histogram Segments

This network formation technique is introduced in [42]. The network formation using overlapping histogram segments always produces a connected graph with vertices of the same community densely interconnected and with vertices of different communities sparsely interconnected.

As we have discussed, the k-NN network formation technique is widely employed in the machine learning domain. However, it suffers from several problems, among which we can highlight: (1) the constructed network may not

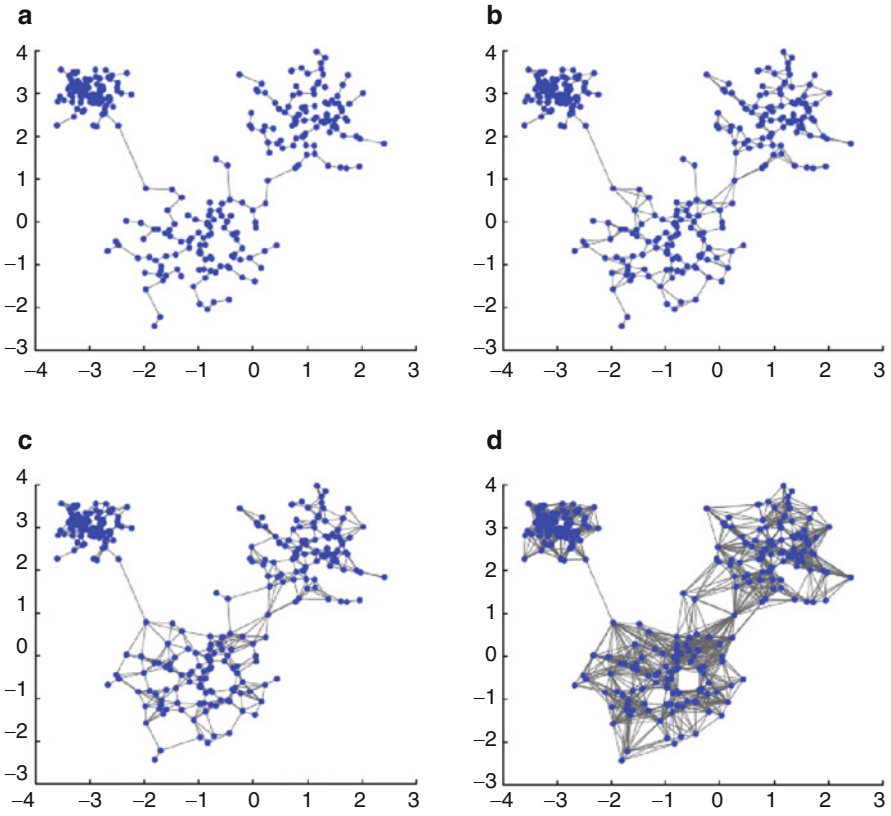

Fig. 4.7 Networks formed by the application of the network formation technique based on clustering heuristics on the data set in Fig. 4.6a with $\lambda = 3$. (**a**) $k = 1$, (**b**) $k = 3$, (**c**) $k = 5$, (**d**) $k = 20$. Reproduced from [15] with permission from Elsevier

necessarily be connected, and even worse (2) the constructed network may not reliably represent the data distribution. For instance, consider a data set with two clusters: a very large cluster and another very small. If we select a large k value, these two clusters would be heavily interconnected, as each of the vertices belonging to the small cluster would be compelled to connect to vertices from the large cluster. Conversely, if k is small, the large cluster may get fragmented into several small communities. Both situations are undesirable in data analysis. The network formation using overlapping histogram segments addresses these problems.

Define a mapping function $h : \mathscr{X} \mapsto [0, 1]$, which receives a data item with $P > 0$ attributes and converts it into a scalar value in the unitary interval.[9] The function h should be smooth in the sense that similar items receive approximately

[9]We use the unitary interval with no loss of generality.

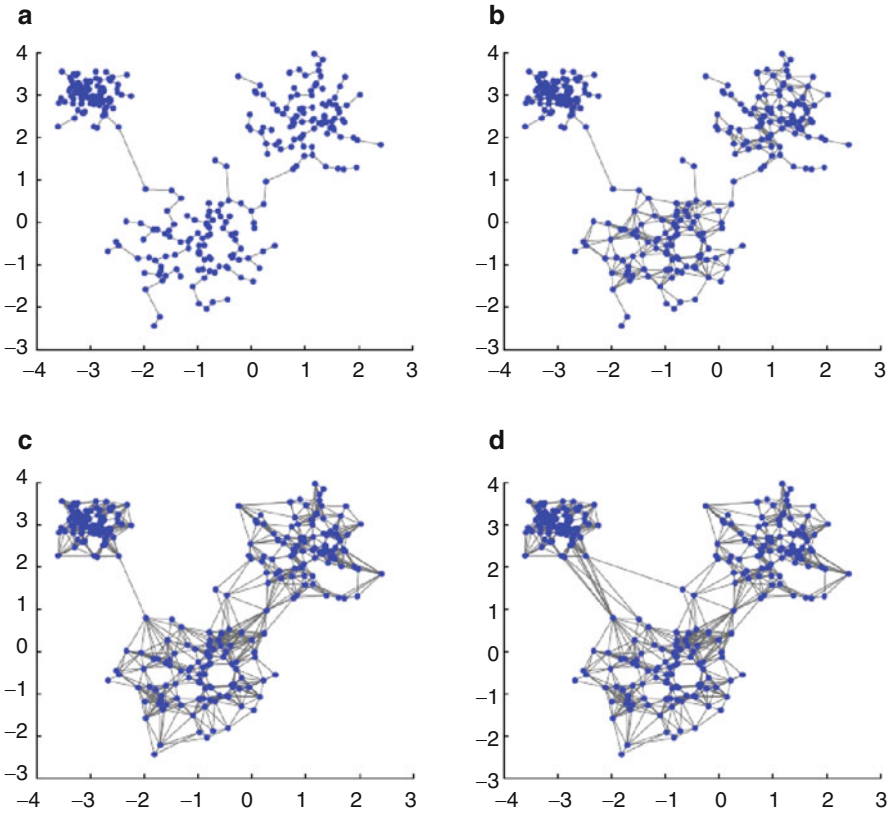

Fig. 4.8 Networks formed by the application of the network formation technique based on clustering heuristics on the data set in Fig. 4.6a with $k = 5$. (**a**) $\lambda = 1$, (**b**) $\lambda = 2$, (**c**) $\lambda = 4$, (**d**) $\lambda = 8$. Reproduced from [31] with permission from the author

similar scalar values. For instance, in a gray-level image in which each vertex is a pixel, $P = 1$ and h is simply the identity function. When $P > 1$, we can use for instance a linear weighted combination of the attributes in the feature vector.

Once transformed the vector-based data set, we construct the histogram of distribution h that resides inside the interval $[0, 1]$ by definition. We also define the set of overlapping intervals $\mathscr{I}$ that are constructed using overlapping histogram segments as follows:

$$\mathscr{I} = \{[0, d]; [d - \kappa, 2d]; \ldots; [(M - 1) \times d - \kappa, 1]\}, \tag{4.39}$$

in which M denotes the number of intervals, $d > 0$ is the non-overlapping window width over adjacent intervals, and $\kappa \geq 0$ is the overlapping factor. The overlapping factor κ is essential for the network formation. It serves the purpose of not letting the resulting network become disconnected.

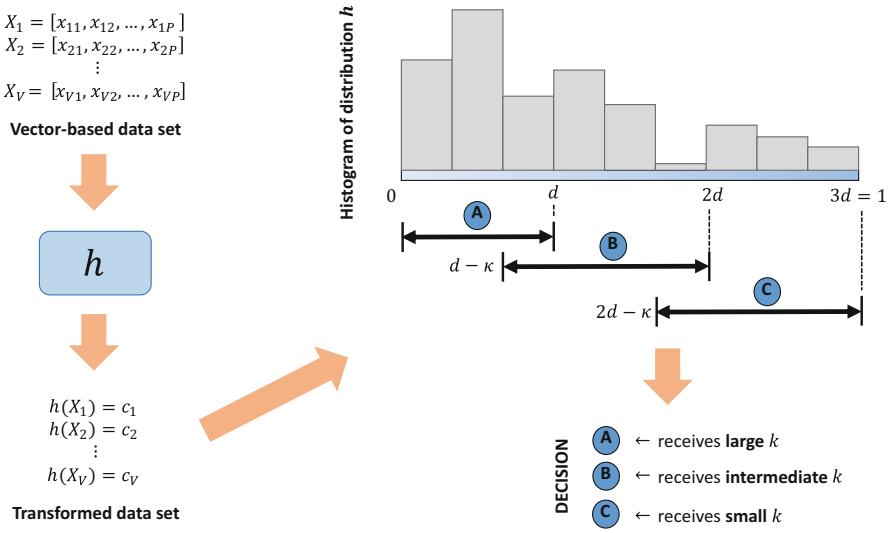

Fig. 4.9 Illustration of the network formation using overlapping histogram segments. In the histogram, we have $d = 3$ and $\kappa = 1$

Let S_i represent the quantity of vertices that is inside interval $i \in \mathscr{I}$. The network is formed by connecting each vertex in i to its k_i most similar neighbors. That is, k_i is adaptive as it varies from interval to interval. Mathematically, k_i is given by:

$$k_i = S_{\max}^2 - (S_{\max} - S_i)^2, \tag{4.40}$$

in which $S_{\max} = \max(S_1, S_2, \ldots, S_M)$. Note that vertices in regions with several vertices are given a large k.[10] Vertices inside regions that have few other vertices are given a small k. In our previous example, the vertices in the large clusters would have large k and those in the small cluster, small k. This behavior gives two nice properties for the resulting network: (1) the number of intercommunity links is expected to be small and (2) the large cluster is not fragmented.

We illustrate the entire network formation process in Fig. 4.9.

We follow the methodology in [42] and apply the network formation process in pixel clustering. In this case, each pixel in the image corresponds to a vertex. We use grayscale images, so h is the identity function. For the clustering technique, we use the greedy modularity function.[11]

[10] Read k in the sense of the k-nearest neighbors in which k is global and static.

[11] In brief, the modularity measures how good a particular network division is in terms of communities. It ranges from 0 to 1. The larger the modularity, the better defined are the communities. See Chap. 6 for more information.

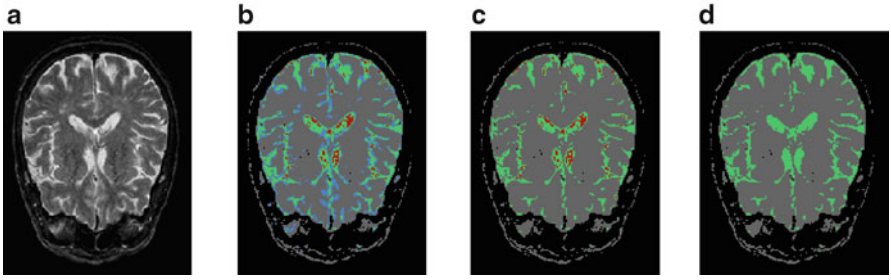

Fig. 4.10 Pixel clustering of a human brain. (**a**) Brain image. (**b**) Results for five clusters. (**c**) Results for four clusters. (**d**) Results for three clusters. The *colors* in (**b**)–(**d**) represent the clusters. We use $d = 0.008$ and $\kappa = 0.5d$

We illustrate the potentialities of the network formation technique based on overlapping histogram segments with pixel clustering tasks. Figure 4.10 shows the results for pixel clustering of a human brain. As the greedy modularity technique is a hierarchical technique, we can see the communities at different levels of granularity. Figures 4.10b–d portray the clustering results for five, four, and three communities, respectively. The network formation process attains a maximum modularity of 0.81 when there are five communities, suggesting that the network communities are well-defined. The histogram of the human brain is supplied in Fig. 4.11. To note the difference of the discussed technique with the k-nearest neighbors technique, we plot the number of vertices as a function of k values in Fig. 4.12. In the original k-NN, the k value is static and global. In contrast, the network formation technique based on overlapping histogram segments has adaptive k.

As another illustrative pixel clustering example, consider the image in Fig. 4.13, in which there are two dogs in the grass. We use a grayscale version of this image in our simulations. The pixel clustering results are given in Fig. 4.14a–d. The maximum modularity is 0.74 and is achieved when four communities are found. The histogram of Fig. 4.13 in grayscale is displayed in Fig. 4.15. To inspect how the underlying network is formed, we also plot the value of k assumed by the vertices in Fig. 4.16. Note that there are two peaks corresponding to the dogs' color and the background composed of grass. Note that the great advantage of this technique is that it can adapt k in terms of community sizes such as to construct well-defined communities (large modularity).

4.3.8 More Advanced Network Formation Techniques

In the previous section, we have presented the k-nearest neighbor, ϵ-radius and other dynamical network formation techniques. The way that they have been introduced assumes that we have no information about the data relationships except the raw data set itself. This hypothesis is consistent with unsupervised

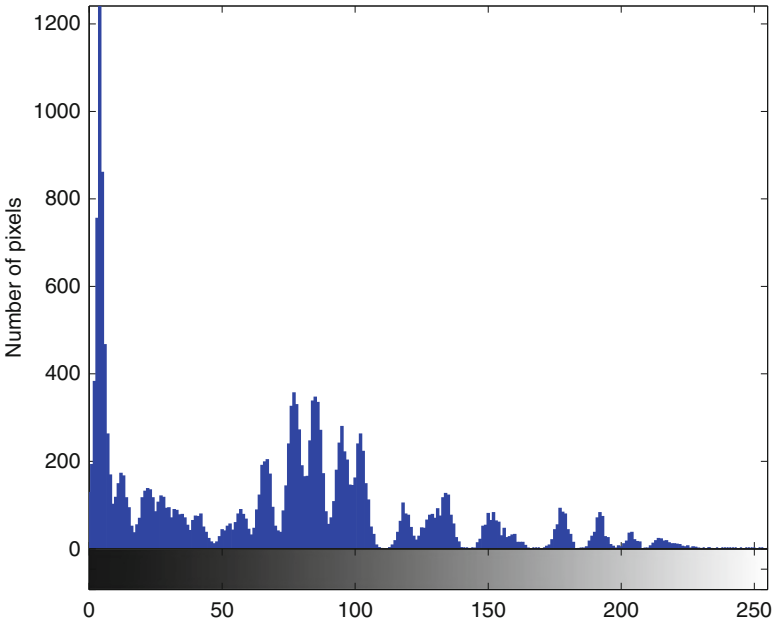

Fig. 4.11 Histogram of the human brain image in Fig. 4.10a. We rescale the h function to the usual pixel interval $[0, 255]$ to facilitate the understanding of the grayscale tones. For instance, the *black color* in a pixel is represented by the value 0 and the *white color* is denoted by 255

learning techniques. Methods that lie in the semi-supervised and supervised learning paradigms, however, are provided with additional information, other than the data set itself. As we have seen, we term that additional (external) information as labels or targets. Each data item has a corresponding label indicating the specific class to which it belongs in the analyzed domain. When these labels are discrete, we have a semi-supervised or supervised classification task. When the labels are continuous, we term the learning process as semi-supervised or supervised regression.

A natural refinement in these network formation techniques is to also consider the labels of the data items when creating links. For instance, we may be interested in creating links between pairs of vertices that only belong to the same class. This constraint would lead to the emergence of more than one network component, each of which with vertices of the same class. In other applications, it would be desirable to have connections between members of different classes. In this case, each network component would possibly have a mixture of members of several different classes.

The way we employ different constraints in our network formation procedure defines the topological properties of the resulting network. In addition, the constraints that we are able to bring are not limited to the labels in case of semi-supervised or supervised learning techniques. For instance, if we have a technique that potentially can create several network components, we can create links if

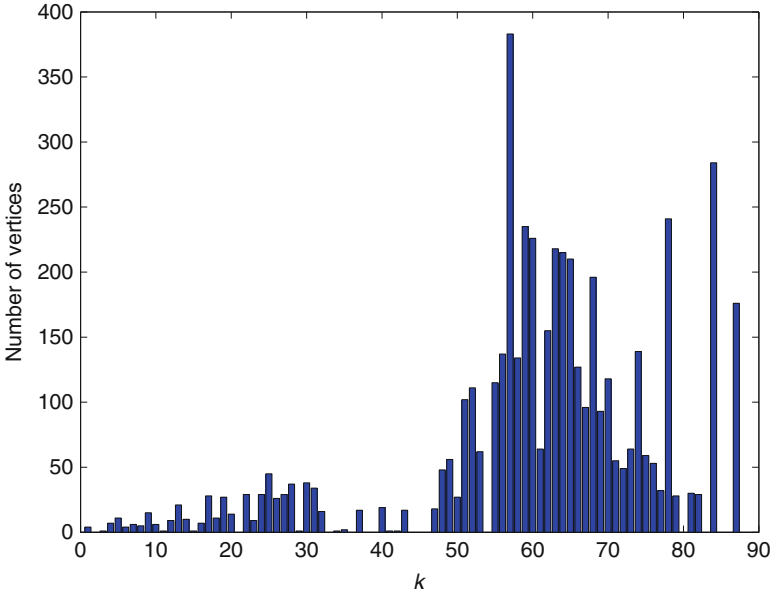

Fig. 4.12 Number of vertices as a function of their assumed k value (network formation parameter) for the human brain image depicted in Fig. 4.10a

the new member that is being tested meets some topological requirements of all of the current members of that component. The topological requirements can be fashioned using local, mixed, and global information. Examples of local information are similar in- and out-degree or in- and out-strength. Network measurements that carry mixed information include similar clustering coefficient, closeness, and betweenness values. Among examples of global information, we can highlight assortativity, network diameter, and the rich club effect.[12] In fact, in one of our case studies in Chap. 8, we discuss more advanced network formation techniques that use both network topological aspects and labels of the data items to construct and to evolve the network in a self-learning process.

4.4 Transforming Time Series Data into Networks

Various time series network construction techniques have been studied in [17]. A time series is a sequence of data points, normally consisting of successive measurements made over a time interval [22]. Examples of time series are ocean tides, counts of sunspots, and the daily closing values of stock markets. Time series

[12] C.f. Sect. 2.3.5 for a more comprehensive classification of network measurements.

Fig. 4.13 Picture of two dogs in the grass. Photo by Liang Zhao

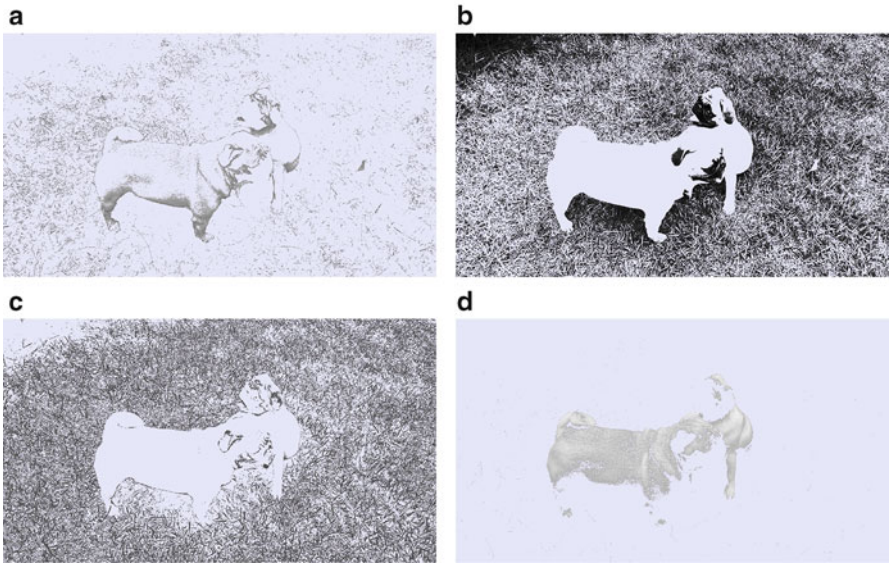

Fig. 4.14 Pixel clustering results of the image portrayed in Fig. 4.13. We use $d = 0.008$ and $\kappa = 0.5d$. (**a**) Community 1. (**b**) Community 2. (**c**) Community 3. (**d**) Community 4

are largely employed in any domain of applied science and engineering that involves temporal measurements, such as in: statistics, signal processing, pattern recognition, econometrics, mathematical finance, weather forecasting, intelligent transport and

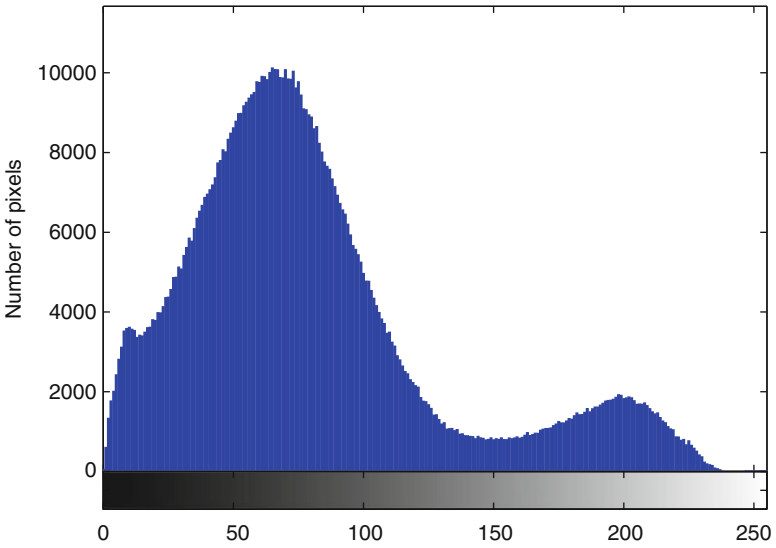

Fig. 4.15 Histogram of the image in Fig. 4.13. We rescale the h function to the usual pixel interval $[0, 255]$ to facilitate the understanding of the grayscale tones. For instance, the *black color* in a pixel is represented by the value 0 and the *white color* is denoted by 255

trajectory forecasting, earthquake prediction, electroencephalography, control engineering, astronomy, and communications engineering [6, 7, 19, 22, 32, 46].

One interesting phenomenon arising in time series analysis is of state recurrence. Formally, in dynamical systems that model time series, there is recurrence of state x_i whenever that system reaches state x_j at another time j that is sufficiently similar to that initial state ($x_i \approx x_j$). Normally, time series data is a sequence in the phase space $\mathscr{T} = \{x_i \in \mathbb{R}^P\}_{i=0}^{\infty}$ that is periodically sampled at intervals Δi, where x_i denotes the time series state in an arbitrary P-dimensional phase space. The set $\mathscr{T}$ is termed as the trajectory of the phase space representing the time series.

Suppose we have that trajectory of the dynamical system in its phase space. The corresponding recurrence plot (RP) is represented by the following recurrence matrix:

$$\mathbf{R}_{ij} = \begin{cases} 1, & \text{if } \|x_i - x_j\| <= \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{4.41}$$

in which $\epsilon > 0$ is a constant that enables state equality up to a small error. Essentially, the recurrence matrix compares the system states at times i and j. If they are similar enough, then $\mathbf{R}_{ij} = 1$. Conversely, when states i and j are rather different, the corresponding entry in the recurrence matrix is $\mathbf{R}_{ij} = 0$. So the recurrence matrix tells us when similar states of the underlying system occur.

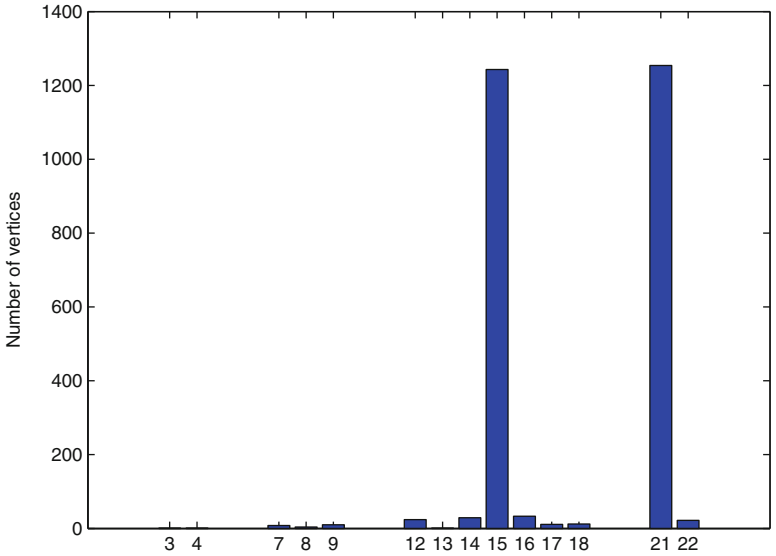

Fig. 4.16 Number of vertices as a function of their assumed k value (network formation parameter) for the image depicted in Fig. 4.13

Several approaches for transforming time series data into complex network have been proposed in the literature. The main approaches are the proximity networks and the transition networks [17]. The former is constructed using mutual proximity of different segments of a time series, while the latter considers transition probabilities between discrete states. The network connectivity of proximity networks is defined in a data-adaptive local fashion, where distinct regions centered at an arbitrary reference vertex in the phase space are considered. The idea can be understood as an adaptive ϵ-radius technique, which we have seen in Sect. 4.3. For transition networks, in contrast, the corresponding regions are rigid, i.e., they are determined by a fixed coarse-graining parameter evaluated in the phase space. In turn, proximity networks are characterized using mutual closeness or similarity of the time series trajectory.

In the following, we briefly discuss some network formation techniques for time series data.

4.4.1 Cycle Networks

The research by Zhang and Small [53] was pioneering in studying topological features of pseudo-periodic time series. In their technique, individual cycles in a time series are identified by the vertices of an undirected network. Links are established between pairs of vertices if cycles behave similarly. Zhang et al. [54]

introduce a generalization of the correlation coefficient applicable to cycles of possibly different lengths to quantify the proximity of cycles in the phase space. The correlation index is defined as the maximum of the cross-correlation between two signals when the shortest of both is slid relative to the longest one. Suppose we compare two cycles $C_1 = \{x_1, x_2, \ldots, x_\alpha\}$ and $C_2 = \{y_1, y_2, \ldots, y_\beta\}$, where $\alpha \leq \beta$.[13] Then, we compute:

$$\rho(C_1, C_2) = \max_{i=0,\ldots,\beta-\alpha} \langle (x_1, x_2, \ldots, x_\alpha), (y_{1+i}, y_{2+i}, \ldots, y_{\alpha+i}) \rangle, \tag{4.42}$$

in which $\langle ., . \rangle$ represents the standard correlation coefficient of two α-dimensional vectors. In this case, we evaluate the (i, j)-th entry of the adjacency matrix $\mathbf{A}$ of the network as follows:

$$\mathbf{A}_{ij} = \Theta\left(\rho(C_i, C_j) - \rho_{\max}\right) - \mathbb{1}_{[i=j]}, \tag{4.43}$$

in which $\Theta(.)$ is the Heaviside function that outputs 1 if the argument is positive and 0, otherwise. $\mathbb{1}_{[.]}$ is the Kronecker delta function that yields 1 if the argument is true and 0, otherwise. $\rho_{\max}$ is the maximum attainable correlation between any two given cycles (vertices). The Kronecker delta function is introduced to prevent self-loops in the resulting network. Another measure to quantify the proximity of cycles is based on phase space distance [53], which is expressed as:

$$D(C_1, C_2) = \min_{i=0,\ldots,\beta-\alpha} \frac{1}{\alpha} \sum_{j=1}^{\alpha} \parallel x_j - y_{j+i} \parallel. \tag{4.44}$$

Using the phase space distance, we compute the adjacency matrix as:

$$\mathbf{A}_{ij} = \Theta\left(D_{\max} - D(C_i, C_j)\right) - \mathbb{1}_{[i=j]}. \tag{4.45}$$

Cycle networks are robust against additive noise and have the advantage that explicit time delay embedding is avoided.

4.4.2 Correlation Networks

Consider each time series represented by a state vector x_i. That is, we have a set of time series for which we want to construct a representative network. The Pearson correlation coefficient can be calculated:

[13] We set the length assumption without loss of generality. If it is not the case, we can simply re-define C_1 and C_2 such as that the hypothesis holds true.

$$r_{ij} = \frac{\text{cov}(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}}, \tag{4.46}$$

in which $\text{cov}(x_i, x_j)$ denotes the covariance between the time series vectors x_i and x_j, and σ_{x_i}, the standard deviation of vector x_i.

The factor $1 - r_{ij}$ therefore is a proximity or similarity measure in the context of time series data. In order to construct the network, we establish a link between states or vertices i and j whenever $1 - r_{ij}$ is larger than a given threshold r [20, 51]:

$$\mathbf{A}_{ij} = \Theta\left(r - r_{ij}\right) - \mathbb{1}_{[i=j]}. \tag{4.47}$$

4.4.3 Recurrence Networks

Recurrence networks are complex networks whose adjacency matrices are given by the recurrence matrices of time series, as computed in (4.41). We define the adjacency matrix of a recurrence network by:

$$\mathbf{A}_{ij} = \mathbf{R}_{ij} - \mathbb{1}_{[i=j]}. \tag{4.48}$$

Because the recurrence matrix can be defined in different ways, there are distinct subtypes of recurrence networks that are characterized by somewhat different structural properties, such as the k-nearest neighbors networks and ϵ-recurrence networks [18, 20, 35]. In k-nearest neighbors networks, we consider every observation vector as a vertex i, which is then connected to its k nearest neighbors in the phase space. In ϵ-recurrence networks, the neighborhood of a vertex (time series) is all times series within a predefined phase space distance ϵ.

4.4.4 Transition Networks

In this method, we build a network from a single time series. The first step in order to build transition networks from time series is to find the amplitude of the analyzed time-varying signal. With that interval at hand, we then discretize it into a suitable set of K classes $\mathscr{S} = \{S_1, \ldots, S_K\}$. The transition probabilities measure the likelihood of the signal jumping from one class (region) to another [37, 41]. Mathematically, $\pi_{\alpha\beta} = P(x_{i+1} = S_\beta | x_i = S_\alpha)$ indicates the probability of the signal to reach region S_β given that it is currently at region S_α. This approach is equivalent to applying a symbolic discretization with static grouping [16] to the phase space of the studied system.

Unlike proximity networks, the resulting transition networks explicitly make use of the temporal order of observations, i.e., their connectivity represents causality relationships contained in the dynamics of the observed system. By introducing a

cutoff $p < 1$ to the transition probability $\pi_{\alpha\beta}$ between pairs of discrete states S_α and S_β, we obtain a non-weighted network representation, which is, however, still directed. Note that for a trajectory that does not leave a finite volume in phase space, there is only a finite number of discrete states S_i with a given minimum size in phase space. This implies the presence of absorbing or recurrent states in the resulting transition network.

The transition probability approach is well suited for identifying the states that have a special importance for the causal evolution of the studied system in terms of betweenness centrality and related measures. However, its main disadvantage is a significant loss of information on small amplitude variations [16].

4.5 Classification of Network Formation Techniques

In this section, we classify the network formation techniques with respect to the type of information they use in their construction processes. The types of network formation techniques are classified as using:

- *Quasi-local information*: these techniques are often restricted to geometrical issues of the data, such as shortest distances between pairs of vertices. In this way, they only employ information from a small set of vertices to construct the links.
- *Long-range information*: these techniques use not only local geometrical information, but long-range information such as the trajectory of shortest paths. That is, we not only account for distances between the endpoints of pairwise shortest paths, but we also use information from the trajectory itself from those shortest paths. In contrast, note that network formation techniques classified as using quasi-local information would only use the distance of the shortest path.
- *Global information*: these techniques use information from all of the data items at once to construct the network. For instance, it may rely on the data distribution itself to adjust parameters of the network formation process.

Tables 4.2 and 4.3 report the classification of the vector-based and time series network formation techniques, respectively. Techniques that are classified as using quasi-local information often share properties such as simplicity and generality, as they can be applied to any data set and for any machine learning task. In this way, they ignore global characteristics that are embedded within the data relationships and are not specialized to a domain-specific task. Techniques that are classified as using long-range and global information are more sophisticated in the sense that they are able to capture local and global characteristics of the data relationships. However, they are often employed to specific purpose tasks. For instance, the network formation technique using clustering heuristics tends to construct a network with the goal of data clustering. In other words, these techniques anticipate the desired result in the network construction phase (before the inference or clustering phases). Techniques that use global information are a tendency in this research topic.

Table 4.2 Classification of vector-based network formation techniques

Definition	Description	Classification
Section 4.3.1	k-NN	Quasi-local information
Section 4.3.1	ϵ-radius	Quasi-local information
Section 4.3.2	Combination of k-NN and ϵ-radius	Quasi-local information
Section 4.3.3	b-matching networks	Long-range information
Section 4.3.4	Linear neighborhood networks	Long-range information
Section 4.3.5	Relaxed linear neighborhood networks	Long-range information
Section 4.3.6	Network formation using clustering heuristics	Long-range information
Section 4.3.7	Network formation using overlapping histogram segments	Global information

The classes are designed using the type of information these techniques employ in the construction process

Table 4.3 Classification of time series network formation techniques

Definition	Description	Classification
Section 4.4.1	Cycle networks	Quasi-local information
Section 4.4.2	Correlation networks	Quasi-local information
Section 4.4.3	Recurrence networks	Quasi-local information
Section 4.4.4	Transition networks	Quasi-local information

The classes are designed using the type of information these techniques employ in the construction process

4.6 Challenges in Transforming Unstructured Data to Networked Data

In this chapter, we have discussed several network formation techniques. When applying network-based machine learning methods to given sets of data points, there are several choices to be made: the type of the network to be constructed and the network formation parameters. However, the question how these choices should be made has received little attention in the literature. This is not so severe in the domain of supervised learning, where parameters can be set using cross-validation. However, it poses a serious problem in unsupervised and semi-supervised learning. While different researchers use different heuristics to set these parameters, few systematic empirical studies have been conducted. For instance, it is important to know how sensitive the results are to the parameters that define the network formation technique. The problem becomes even more severe when we try to find theoretical models that justify the use of one parameter value in detriment to others.

In the domain of unsupervised learning, the study in [34] analyzes clustering measures, such as the normalized cut, in nearest neighbors networks (k-nearest neighbors and ϵ-radius). The investigation shows that, depending on the selected criteria to construct the network, the normalized cut criterion converges to different limit results. The fact that all of these nearest neighbors networks lead to different clustering criteria shows that we cannot study these criteria isolated from the network that they are applied to.

The picture is not different in the semi-supervised learning domain. According to [55], there is no reliable approach for model selection if only a few labeled instances are available. Unfortunately, this is often the case as labeling is expensive. We find a plethora of studies that deal with regularization frameworks that essentially solve constrained optimization processes. These frameworks are roughly composed of two abstract terms: the loss and the regularization functions. While the loss function penalizes decisions that flip labels of labeled vertices, the regularization function models the costs of propagating labels to unlabeled instances. These algorithms mainly adjust or propose new ways of modeling these two functions [56, 57]. The label propagation process, which strongly depends on the network topology, has been extensively studied in the literature. Up to now, little has been done for the network topology analysis *per se*, which is a product of a network formation process. Therefore, a relevant question that arises is whether the resulting network really represents the transformed (vector-based) data. If it is not the case, then the label diffusion process will probably be flawed, as the network topology will be incorrect. Even though most of them lack theoretical framework, we find some few empirical investigations in the literature about network formation, as follows [56]:

- *Construction of networks using domain knowledge*. For instance, the research in [3] builds networks for video surveillance using strong domain knowledge, where the network of web-camera images consists of time edges, color edges and face edges. Such networks reflect a deep understanding of the problem structure and how unlabeled data is expected to help. We note that constructing domain knowledge requires an active work of human experts. Recall that the labeling process of human experts is expensive and time-consuming. Moreover, edge construction process is even more slower as it is a problem with a mapping function of the type $\mathscr{V} \times \mathscr{V} \mapsto \mathbb{R}$, i.e., we consider the weight of every potential edge between arbitrary pairwise vertices.[14] In this way, even though domain knowledge may improve the performance of machine learning algorithms, it definitely turns into an unfeasible procedure as the number of the data items grows.
- *Construction of nearest neighbors networks*. Empirically, weighted k-NN network with small k tends to perform better. We can also build near complete networks, using, for instance, correlation or kernel Gaussian functions as similarity functions. In the sparsification process, we can apply several tricks. The investigation in [8] builds robust networks from multiple minimum spanning trees by perturbation and edge removal. When using a Gaussian function as edge weights, the bandwidth of the Gaussian needs to be carefully chosen. In turn, the study in [52] derives a cross-validation approach to tune the bandwidth for

[14] In contrast, the vertex labeling is a mapping task $\mathscr{V} \mapsto \mathscr{Y}$, which is much quicker than edge labeling.

each feature dimension, by minimizing the leave-one-out mean squared error of predictions and given labels on labeled points.

- *Construction of networks using local fit procedures*. The investigation in [23, 24] proposes an algorithm to de-noise points sampled from a manifold. That is, data points are assumed to be noisy samples of some unknown underlying manifold. They used the de-noising algorithm as a preprocessing step for network-based semi-supervised learning, so that the network can be constructed from better separated data points. Such preprocessing results in better semi-supervised classification accuracy.

4.7 Chapter Remarks

In this section, we have reviewed the main ingredients involved in constructing networks from vector-based and time series data. In special, we have seen that we need a proper similarity function and a suitable strategy for creating links in the network. Several similarity functions have been discussed. The shortcomings and advantages of the most well-known network formation techniques have been explored. We have also visited the inherent challenges that we face when building networks that reliably maintain the data distribution.

References

1. Aggarwal, C.C.: Mining text data. In: Data Mining, pp. 429–455. Springer International Publishing, New York (2015)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, New York (1999)
3. Balcan, M.F., Blum, A., Choi, P.P., Lafferty, J., Pantano, B., Rwebangira, M.R., Zhu, X.: Person identification in webcam images: an application of semi-supervised learning. In: ICML 2005 Workshop on Learning with Partially Classified Training Data, vol. 2. ACM Press (2005)
4. Baroni-Urbani, C., Buser, M.W.: Similarity of binary data. Syst. Zool. **25**, 251–259 (1976)
5. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
6. Bloomfield, P.: Fourier Analysis of Time Series: An Introduction. Wiley, New York (1976)
7. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting. Springer, New York (1996)
8. Carreira-Perpiñán, M.A., Zemel, R.S.: Proximity graphs for clustering and manifold learning. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, vol. 17, pp. 225–232 (2004)
9. Celikyilmaz, A., Hakkani-Tur, D.: A graph-based semi-supervised learning for question semantic labeling. In: Proceedings of the NAACL HLT 2010 Workshop on Semantic Search, pp. 27–35. Association for Computational Linguistics, Los Angeles, CA (2010)
10. Celikyilmaz, A., Thint, M., Huang, Z.: A graph-based semi-supervised learning for question-answering. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2, ACL '09, pp. 719–727. Association for Computational Linguistics (2009)

11. Cha, S.H.: Comprehensive survey on distance/similarity measures between probability density functions. Int. J. Math. Models Methods Appl. Sci. **1**, 300–307 (2007)
12. Chao, A., Chazdon, R.L., Colwell, R.K., Shen, T.J.: Abundance-based similarity indices and their estimation when there are unseen species in samples. Biometrics **62**(2), 361–371 (2006)
13. Cock, M.D., Kerre, E.: On (un)suitable relations to model approximate equality. Fuzzy Sets Syst. **133**, 137–153 (2003)
14. Colwell, R., Coddington, J.: Estimating terrestrial biodiversity through extrapolation. Philos. Trans. R. Soc. B Biol. Sci. **345**, 101–118 (1994)
15. Cupertino, T.H., Huertas, J., Zhao, L.: Data clustering using controlled consensus in complex networks. Neurocomputing **118**, 132–140 (2013)
16. Donner, R., Hinrichs, U., Scholz-Reiter, B.: Symbolic recurrence plots: A new quantitative framework for performance analysis of manufacturing networks. Eur. Phys. J. Spec. Top. **164**(1), 85–104 (2008)
17. Donner, R.V., Small, M., Donges, J.F., Marwan, N., Zou, Y., Xiang, R., Kurths, J.: Recurrence-based time series analysis by means of complex network methods. Int. J. Bifurcation Chaos **21**(4), 1019–1046 (2010)
18. Donner, R.V., Zou, Y., Donges, J.F., Marwan, N., Kurths, J.: Recurrence networks – a novel paradigm for nonlinear time series analysis. New J. Phys. **12**, 033025 (2010)
19. Durbin, J., Koopman, S.J.: Time Series Analysis by State Space Methods. Oxford University Press, Oxford (2001)
20. Gao, Z., Jin, N.: Flow-pattern identification and nonlinear dynamics of gas-liquid two-phase flow in complex networks. Phys. Rev. E **79**, 066303 (2009)
21. Giguère, S., Laviolette, F., Marchand, M., Tremblay, D., Moineau, S., Liang, X., Biron, A., Corbeil, J.: Machine learning assisted design of highly active peptides for drug discovery. Public Libr. Sci. Comput. Biol. **11**(4), e1004074 (2015)
22. Hamilton, J.D.: Time Series Analysis. Princeton University Press, Princeton, NJ (1994)
23. Hein, M., Maier, M.: Manifold denoising. In: Advances in Neural Information Processing Systems, vol. 19, pp. 561–568. MIT Press, Cambridge (2007)
24. Hein, M., Maier, M.: Manifold denoising as preprocessing for finding natural representations of data. In: Association for the Advancement of Artificial Intelligence, pp. 1646–1649. AAAI Press, San Jose (2007)
25. Huang, B.C., Jebara, T.: Loopy belief propagation for bipartite maximum weight b-matching. In: International Conference on Artificial Intelligence and Statistics, pp. 195–202 (2007)
26. Jaccard, P.: Etude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull. Soc. Vaud. Sci. Nat. **37**, 547 (1901)
27. Jebara, T., Wang, J., Chang, S.F.: Graph construction and b-matching for semi-supervised learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, pp. 441–448. ACM, New York, NY (2009)
28. Koleff, P., Gaston, K.J., Lennon, J.J.: Measuring beta diversity for presence-absence data. J. Anim. Ecol. **72**(3), 367–382 (2003)
29. Kumar, B.V.K.V., Hassebrook, L.G.: Performance measures for correlation filters. Appl. Opt. **29**, 2997–3006 (1990)
30. Libbrecht, M.W., Noble, W.S.: Machine learning applications in genetics and genomics. Nat. Rev. Genet. **16**, 321–332 (2015)
31. Lopez, J.P.H.: Análise de dados utilizando a medida de tempo de consenso em redes complexas (2011). Master Thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo (USP)
32. Luetkepohl, H.: Introduction to Multiple Time Series Analysis. Springer, New York (1991)
33. MacCuish, J.D., MacCuish, N.E.: Clustering in Bioinformatics and Drug Discovery. CRC Press, Boca Raton (2010)
34. Maier, M., von Luxburg, U., Hein, M.: Influence of graph construction on graph-based clustering measures. Neural Inf. Process. Syst. **22**, 1025–1032 (2009)
35. Marwan, N., Donges, J.F., Zou, Y., Donner, R.V., Kurths, J.: Complex network approach for recurrence analysis of time series. Phys. Lett. A **373**, 4246–4254 (2009)

36. Morisita, M.: Measuring of interspecific association and similarity between communities. Mem. Fac. Sci. Kyushu Univ. Ser E (Biology) **3**, 65–80 (1959)
37. Nicolis, G., Cantú, A.G., Nicolis, C.: Dynamical aspects of interaction networks. Int. J. Bifurcation Chaos **15**(11), 3467–3480 (2005)
38. Orozco, J., Belanche, L.: Towards a mathematical framework for similarity and dissimilarity. Technical Report, University of Sevilla (2005)
39. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**, 2323–2326 (2000)
40. Santini, S., Jain, R.: Similarity measures. IEEE Trans. Pattern Anal. Mach. Intell. **21**(9), 871–883 (1999)
41. Shirazi, A.H., Reza Jafari, G., Davoudi, J., Peinke, J., Tabar, M.R.R., Sahimi, M.: Mapping stochastic processes onto complex networks. J. Stat. Mech: Theory Exp. **2009**, P07046 (2009)
42. Silva, T.C., Zhao, L.: Pixel clustering by using complex network community detection technique. In: Proceedings of 7th International Conference on Intelligent Systems Design and Applications, pp. 925–932. IEEE Computer Society (2007)
43. Silva, T.C., Zhao, L.: Network-based high level data classification. IEEE Trans. Neural Netw. Learn. Syst. **23**(6), 954–970 (2012)
44. Silva, T.C., Zhao, L.: High-level pattern-based classification via tourist walks in networks. Inf. Sci. **294**(0), 109–126 (2015). Innovative Applications of Artificial Neural Networks in Engineering
45. Sørensen, T.: A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. Biol. Skr. **5**, 1–34 (1948)
46. Tsay, R.S.: Analysis of Financial Time Series. Wiley Series in Probability and Statistics. Wiley-Interscience, Hoboken, NJ (2005)
47. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. IEEE Trans. Knowl. Data Eng. **20**(1), 55–67 (2008)
48. Williams, J., Steele, N.: Difference, distance and similarity as a basis for fuzzy decision support based on prototypical decision classes. Fuzzy Sets Syst. **131**, 35–46 (2002)
49. Wolda, H.: Similarity indices, sample size and diversity. Oecologia **50**(3), 296–302 (1981)
50. Xu, Z., Xia, M.: Distance and similarity measures for hesitant fuzzy sets. Inf. Sci. **181**(11), 2128–2138 (2011)
51. Yang, Y., Yang, H.: Complex network-based time series analysis. Physica A **387**, 1381–1386 (2008)
52. Zhang, X., Lee, W.S.: Hyperparameter learning for graph based semi-supervised learning algorithms. In: The Conference on Neural Information Processing Systems (NIPS) (2006)
53. Zhang, J., Small, M.: Complex network from pseudoperiodic time series: topology versus dynamics. Phys. Rev. Lett. **96**, 238701 (2006)
54. Zhang, J., Luo, X., Small, M.: Detecting chaos in pseudoperiodic time series without embedding. Phys. Rev. E **73**, 016216 (2006)
55. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems, vol. 16, pp. 321–328. MIT Press, Cambridge (2004)
56. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
57. Zhu, X., Goldberg, A.B.: Introduction to Semi-Supervised Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, San Francisco (2009)