Statistical Science
2023, Vol. 38, No. 3, 411–439
https://doi.org/10.1214/22-STS881

Statistical Embedding: Beyond Principal Components

Dag Tjøstheim, Martin Jullum and Anders Løland

Abstract. There has been an intense recent activity in embedding of very high-dimensional and nonlinear data structures, much of it in the data science and machine learning literature. We survey this activity in four parts. In the first part, we cover nonlinear methods such as principal curves, multidimensional scaling, local linear methods, ISOMAP, graph-based methods and diffusion mapping, kernel based methods and random projections. The second part is concerned with topological embedding methods, in particular mapping topological properties into persistence diagrams and the Mapper algorithm. Another type of data sets with a tremendous growth is very high-dimensional network data. The task considered in part three is how to embed such data in a vector space of moderate dimension to make the data amenable to traditional techniques such as cluster and classification techniques. Arguably, this is the part where the contrast between algorithmic machine learning methods and statistical modeling, represented by the so-called stochastic block model, is at its greatest. In the paper, we discuss the pros and cons for the two approaches. The final part of the survey deals with embedding in $\mathbb{R}^2$, that is, visualization. Three methods are presented: t-SNE, UMAP and LargeVis based on methods in parts one, two and three, respectively. The methods are illustrated and compared on two simulated data sets; one consisting of a triplet of noisy Ranunculoid curves, and one consisting of networks of increasing complexity generated with stochastic block models and with two types of nodes.

Key words and phrases: Statistical embedding, principal component, nonlinear principal component, multidimensional scaling, local linear method, ISOMAP, graph spectral theory, diffusion mapping, reproducing kernel Hilbert space, random projection, topological data analysis and embedding, persistent homology, persistence diagram, the Mapper, network embedding, spectral embedding, stochastic block modeling, Skip-Gram, neighborhood sampling strategies, visualization, t-SNE, LargeVis, UMAP.

1. INTRODUCTION

With the advent of the big data revolution, the availability of data has exploded. The dimension of the data can be in the thousands, if not in the millions, and the relationships between data vectors can be exceedingly complex. Also, data are arriving in new forms. One recent addition to data types is network data, sometimes with millions of nodes, and literally billions of edges (relationships between nodes). An example is the analysis of porous media, in oil exploration say, or of astronomical or physiological data. Such data contain cavities and complicated geometric structures. Another example is in natural languages with texts containing million of words. Is it possible to characterize language segments so as to discriminate one type of text from another?

These examples have to do with the characterization and simplification of highly complex and often unorganized data. From a mathematical and statistical point of view, these tasks are examples of embedding problems.

The goal of this survey could be said to be two-fold. First, to try to give a quite comprehensive survey of embedding methods and applications of these methods. The

Dag Tjøstheim is Professor Emeritus at the Department of Mathematics, University of Bergen, Bergen, Norway and Professor II at the Norwegian Computing Center, Oslo, Norway (e-mail: Dag.Tjostheim@uib.no). Martin Jullum is Senior Research Scientist at the Norwegian Computing Center, Oslo, Norway (e-mail: Martin.Jullum@nr.no). Anders Løland is Research Director at the Norwegian Computing Center, Oslo, Norway (e-mail: Anders.Loland@nr.no).

second objective of this article has been to make the statistical community more aware of current methods in this branch of data science bordering on machine learning.

Here is a brief overview of the contents of the paper. Section 2 gives a brief summary of principal components and points out some strengths and weaknesses. There are now a number of novel nonlinear methods, some of them in fact with roots going far back in time. In Section 3, we look at methods such as principal curves and surfaces, multidimensional scaling, local linear embedding, embedding via graphs (note that in this survey the terms "graph" and "network" will be used interchangeably), ISOMAP and Laplace eigenmaps, diffusion maps, kernel principal components using reproducing kernel Hilbert spaces and random projections. Section 4 has to do with the emerging field of topological data analysis and topological manifold embedding. Section 5 deals with embedding of network data, especially ultra high-dimensional networks. This is a topic of great practical interest, as can be understood for instance from the recent advances within social network analysis. Arguably, this is the theme where the contrast between algorithmic machine learning methods and statistical modeling is at its most pronounced. We discuss the pros and cons for the two approaches in Sections 5.2.4 and 5.7. Open problems in heterogeneous, directed and dynamic networks are also briefly covered in Section 5.

Finally, in Section 6, we go on to the extreme case of having an embedding of dimension 2, the plane. This has to do with visualization, of course, and we are presenting three visualization methods, t-SNE, LargeVis and UMAP, whose basis can be found in each of the preceding sections, namely nonlinear type embedding, network embedding and topological embedding. They are compared to principal component visualization.

To our knowledge, our survey paper is the first of such broad coverage. To avoid an overlong paper, some of the more technical and detailed aspects of the surveyed methods are relegated to the Supplementary Material (Tjøstheim, Jullum and Løland, 2023). There are many unsolved statistical problems, and we will try to point out some of these as we proceed.

We have chosen to illustrate our methods by two types of simulation experiments. First, a triple of noisy Ranunculoid (a concept originating in flower forms in botany) curves encapsulated in one another (cf. Figure 1(a)) (a situation in which principal components do not work), illustrates a number of the nonlinear methods of Section 3 and the topological embedding of Section 4. As a second example, we have included a network based simulation, generated by stochastic block models, with two types of nodes and varying degrees of complexity in their interaction. Among other things, these are used to illustrate and compare the three visualization methods of Section 6, for several choices of their input parameters. In the paper, we also refer to real data experiments that have been conducted especially in the network embedding literature.

2. PRINCIPAL COMPONENTS

Principal component analysis (PCA) was invented by Pearson (1901) as an analogue of the analysis of principal axes in mechanics. It was later independently developed by Harold Hotelling in the 1930s; see, for example, Hotelling (1933) and Hotelling (1936).

Given p-dimensional observations $X_1, \dots, X_n$, the Hotelling approach was along the lines that have since become standard: Let $X_i, i = 1, \dots, n$ have components $X_{ij}, j = 1, \dots, p$. The first principal component $V_1 = \{a_{j1}\}$ consists of the weights, which gives the linear combination $\sum_{j=1}^{p} a_{j1} X_{ij}$ maximum variance subject to the constraint that the Euclidean norm $\|V_1\| = 1$. The kth principal component $V_k = \{a_{jk}\}$ corresponds to the linear combination $\sum_{j=1}^{p} a_{jk} X_{ij}$ with the maximum variance subject to $\|V_k\| = 1$, and it being orthogonal to previously found V_j, $1 \le j \le k-1$. Or said in another way, the principal components constitute a sequence of projections in $\mathbb{R}^p$ of the data, mutually uncorrelated and ordered in variance.

Let $\mathbf{\Sigma}$ be the $p \times p$ population covariance matrix. Then it is well known (see, e.g., Jolliffe, 2002), that the principal components V_k are obtained by solving the eigenvalue problem

$$\mathbf{\Sigma} V_k = \lambda_k V_k, \tag{1}$$

where the largest eigenvalue λ_1 corresponds to the first principal component V_1, and where the variance explained by the kth principal component is given by $\lambda_k / \sum_{i=1}^{p} \lambda_i$.

The estimated principal components are obtained by considering an estimate of $\mathbf{\Sigma}$. Let $\mathbf{X}$ be the $n \times p$ centered data matrix $\mathbf{X} = \{(X_{ij} - \bar{X}_j)\}$ with $\bar{X}_j = n^{-1} \sum_i X_{ij}$, then an estimate of $\mathbf{\Sigma}$ is obtained from $n^{-1}[\mathbf{X}^T \mathbf{X}]$, and the estimated eigenvectors and eigenvalues are obtained from

$$\mathbf{X}^T \mathbf{X} \hat{V} = \hat{\lambda} \hat{V}. \tag{2}$$

The approach of Pearson (1901) is different, and the essence of his method is that he looks at a set of m principal components as spanning a hyperplane of rank m in $\mathbb{R}^p$ such that the sum of the distances from the data points to this hyperplane is minimized. The first principal component is then the line in $\mathbb{R}^p$ obtained by such a minimization. As will be seen, it is the Pearson approach, which is most amenable to generalizations to the nonlinear case.

Before we close this section, there is cause to ask why linear principal component analysis is so useful. It is clearly the most used statistical embedding method. Why? There are several reasons for this. One is its potential to

reduce the dimension of the original data. If a few principal components explain a large percentage of the variation, this in many cases means that the ensuing analysis can be concentrated to those components. These components can also be used henceforth in a factor analysis. And the number of needed components can often be decided by a clear cut percentage of variation explained, which, as was seen above, is straightforward to compute given the eigenvalues of the covariance matrix.

Principal components have been used with great success in a number of different fields, so diverse as, for example, quantitative finance, medicine, neuroscience, genetics, meteorology, chemistry, and recognition of handwritten characters. Many applications and the basis of the theory are given in the book by Jolliffe (2002). It is also quite robust and can work reasonably well for certain types of nonlinear systems, as seen in the comparative review by van der Maaten, Postma and van der Herik (2009).

However, there are also several shortcomings of linear principal components, which have inspired much recent research. The most obvious fault is the fact that it is a linear method, and data are often nonlinearly generated or located on or close to a submanifold of $\mathbb{R}^p$. This is sometimes aggravated by the fact that the PCA is based on the covariance matrix, and it is well known that a covariance between two stochastic variables is not always a good measure of statistical dependence. This has been particularly stressed in recent dependence literature, a survey of which is given in Tjøstheim, Otneim and Støve (2022a). Especially there exist statistical models and data where the covariance is zero although there may be a strong statistical dependence. An example is the so-called ARCH/GARCH time-series models for financial risk.

To do statistical inference in PCA often a Gaussian assumption is added as well. For Gaussian variables the covariance matrix describes the dependence relations completely, so that it would be impossible to improve on the PCA embedding by a nonlinear embedding. But increasingly, data sets are appearing where the Gaussian assumption is not even approximately true. Moreover, the dimension of data may be extremely large, not making it easily amenable to principal component analysis, which involves the solution of a p-dimensional eigenvalue problem. Note, however, that for practical purposes the first $k \ll p$ eigenvectors typically suffices even for complex high-dimensional settings. Such situations are handled in modern PCA software by utilizing extensions of the so-called Lanczos algorithm; see, for example, Baglama and Reichel (2005).

3. NONLINEAR EMBEDDINGS

There are a variety of possible nonlinear dependence structures, for each of which there are particular nonlinear algorithms adapted to the given structure.

For the so-called principal curve method (Hastie, 1984), the data are supposed to be concentrated roughly on a curve or more generally on a submanifold. Although in this case the data are not well represented by a linear model, they may still be well approximated by a local linear model giving rise to the LLE method (Roweis and Saul, 2000) or to ISOMAP (Tenenbaum, de Silva and Langford, 2000). Alternatively, the data may lie on a chained nonconvex structure; see, for instance, the example in Figure 1. For such and similar structures, one may try to map the dependence properties to a graph, leading to a Laplace eigenvalue problem (Belkin and Niyogi, 2002), and in its continuation to diffusion maps (Coifman and Lafon, 2006). In still other situations, it may be advantageous to use a nonlinear transformation of the data points, and then solve a resulting eigenvalue problem, as is done in kernel principal components (Schölkopf, Smola and Müller, 2005). One of the classical nonlinear methods is multidimensional scaling (MDS) (Torgerson, 1952), where an embedding is sought by preserving distances between individual data points. A combined linear and distance preserving method is represented by random projections, whose rationale is based on Johnson and Lindenstrauss (1984). All of these methods are presented in more detail in the following subsections, and most are illustrated in Figure 1.

3.1 Principal Curves and Surfaces

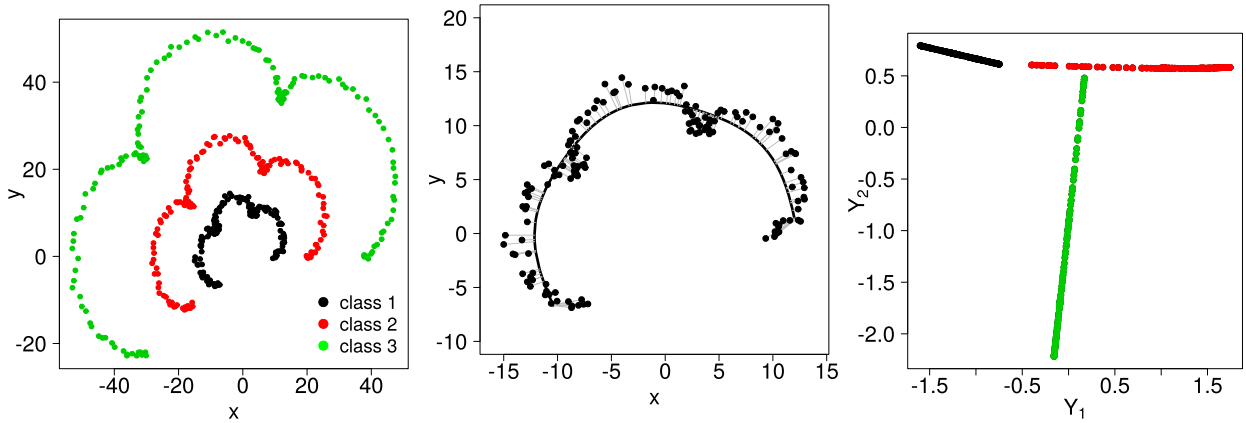
As mentioned in Section 2, it is the Pearson's hyperplane fitting that is perhaps the best point of departure for nonlinear PCA. Principal curves and surfaces were introduced in Hastie (1984) and Hastie and Stuetzle (1989). A brief summary is given in Hastie, Tibshirani and Friedman (2019, pages 541–544). Essentially, the idea is to replace the hyperplane by a hypersurface. It is simplest in the case of principal curves, generalizing the first principal component. Let $f(s)$ be a parameterized smooth curve in $\mathbb{R}^p$. The parameter s in this case is a scalar and can for instance be arc-length along the curve. For each p-dimensional data value X, one lets $s_f(X)$ be the point on the curve closest to X. Then $f(s)$ is called a principal curve for the distribution of the random vector X if

$$f(s) = E\big(X|s_f(X) = s\big).$$

This means that $f(s)$ is the average of all data points that project onto it. This is known as the self-consistency property. In practice, it turns out (Duchamp and Stuetzle, 1996) that there are infinitely many principal curves for a given multivariate distribution, but one is interested mainly in the smooth ones.

3.1.1 *Algorithm for finding one principal curve* $f(s)$.

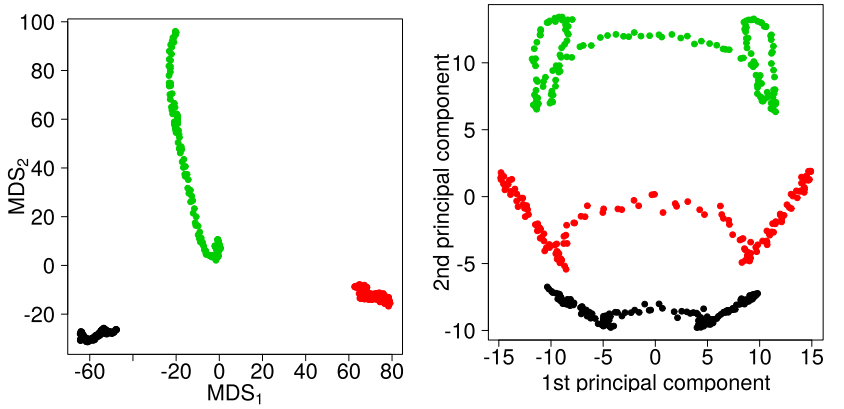
1. *Definitions of coordinate functions and* X. Consider the coordinate functions $f(s) = [f_1(s), \ldots, f_p(s)]$ and let X be the p-dimensional observational vector given by $X^T = (X_1, \ldots, X_p)$.



(a) The raw data.

(b) Principal curve on the innermost curve of Fig. 1a.

(c) Local linear embedding.



(d) ISOMAP.

(e) Kernel principal components (with a Bessel kernel).

FIG. 1. *Four different embedding methods applied to three parametric curves from the so-called Ranunculoid and perturbed by Gaussian noise with a standard deviation of* 1/2.

2. *The two alternating steps*:

$$E(X_j|\hat{s}_f(X) = s) \to \hat{f}_j(s); \quad j = 1, \ldots, p \tag{3}$$

and

$$\underset{s'}{\operatorname{argmin}} \|X - \hat{f}(s')\|^2 \to \hat{s}_f(X). \tag{4}$$

Here, the first step (3) fixes s and enforces the self-consistency requirement. The second step (4) fixes the curve and finds the closest point on the curve to each data point. The principal curve algorithm starts with the first linear principal component, and iterates the two steps in (3) and (4) until convergence is obtained using a given tolerated error.

Principal surfaces generalize principal curves to higher dimensional representations. The most commonly used is the two-dimensional principal surface with coordinate functions

$$f(y_1, y_2) = [f_1(y_1, y_2), \ldots, f_p(y_1, y_2)].$$

The estimates in step (3) and (4) above are obtained from two-dimensional surface smoothers. The scheme with a quantification of percentage reduction of variance seems to be lost in a principal curve and principal surface set-up. A different but related approach is taken by Ozertem and Erdogmus (2011), where principal curves and surfaces are studied in terms of density ridges. See also Section 4.1 for generalizations to non-Euclidean spaces and so-called manifold learning.

In Figure 1, we present a data set that will be used for illustration purposes throughout this section and also in Section 4 on topological data analysis. The raw data are presented in Figure 1(a). It consists of parts of three parametric curves, each being obtained from the so-called Ranunculoid, but with three different parameter sets. In addition, the curves have been perturbed by Gaussian noise. In Figure 1(b), we have illustrated the construction of a principal curve on the innermost curve of Figure 1(a). The main one-dimensional structure of the curve is well picked up, but it does not quite get all the indentions of the original curve. Compared to a linear principal regression curve it is a big improvement.

3.2 Multidimensional Scaling

The idea of multidimensional scaling (MDS) goes far back, but similar ideas have recently got a revival in statistical embedding through algorithms such as LLE, ISOMAP (see the next subsections) and t-SNE (see Section 6). It can be roughly formulated as finding suitable coordinates for a set of points given their mutual distances. This problem was first considered by Young and Householder (1938). These methods were further developed and applied to scaling of psychometric distances between pairs of stimuli by Torgerson (1952). A fine review of the essentials of multidimensional scaling is given in Hastie, Tibshirani and Friedman (2019, pages 570–572). Their emphasis is on viewing multidimensional scaling as a general method for dimensionality reduction of data in $\mathbb{R}^p$. They therefore start with a set of observations $X_1, \ldots, X_n \in \mathbb{R}^p$ where d_{ij} is some form of distance measure (not necessarily Euclidean) between observation X_i and X_j. In fact, in the general theory of multidimensional scaling the d_{ij} may be considered as a dissimilarity measure between objects (e.g., psychological stimuli) i and j.

From a dimension reduction point of view, multidimensional scaling seeks values $Y_1, \ldots, Y_n \in \mathbb{R}^m$, often $m = 2$ for visualization purposes, by minimizing the so-called stress function

$$S(Y_1, \ldots, Y_n) = \sum_{i \neq j} \big(d_{ij} - \|Y_i - Y_j\|\big)^2,$$

which means choosing $\{Y_j, j = 1, \ldots, n\}$ such that one strives to preserve distances when going from $\mathbb{R}^p$ to $\mathbb{R}^m$. This is known as the least squares or Kruskal–Shephard scaling. A gradient descent algorithm can be used to minimize S. A variation on this is the so-called Sammon mapping, Sammon (1969), which minimizes

$$S_{Sm}(Y_1, \ldots, Y_n) = \sum_{i \neq j} \frac{(d_{ij} - \|Y_i - Y_j\|)^2}{d_{ij}}.$$

Note that multidimensional scaling creates an embedding between two Euclidean spaces, $\mathbb{R}^p$ and $\mathbb{R}^m$. This is different from principal surfaces (Section 3.1) and many of the other methods in this survey, which creates embeddings from $\mathbb{R}^p$ to a lower-dimensional *manifold*.

3.3 LLE—Local Linear Embedding

Principal curves and surfaces represent an early example of local modeling and manifold embedding. Manifold embedding will be taken up from a more general point of view in Section 4 with its connections to recent advances in TDA (Topological Data Analysis). However, it is convenient at this point to briefly mention the early work of Roweis and Saul (2000) that resembles the principal surface methodology in that it is a local method. In fact, it is a local *linear* model, and locally linear methods are well known and much used in nonparametric regression. But here the viewpoint is different since there is no clearly defined dependent variable. Actually in that respect, it is like the recent local Gaussian modeling of Tjøstheim, Otneim and Støve (2022b).

Suppose that the data $X_1, \ldots, X_n$ are p-dimensional vectors sampled from an inherent m-dimensional manifold. One assumes that each data point lies on or close to a locally linear patch of the manifold. The local geometry of these patches is characterized by linear coefficients that reconstruct each data point from its neighbors.

The LLE algorithm consists of three main steps:

1. *Find the nearest neighbors $N(i)$ of X_i*, for example, by a nearest neighborhood algorithm, such as kNN (k-nearest neighbors).

2. *Construct weights w_{ij} by minimizing the cost function* (5) subject to the constraint that $w_{ij} = 0$ if x_j does not belong to the set of neighbors of X_i, and such that $\sum_j w_{ij} = 1$. Weights for nonneighbors are 0.

$$M_1(w) = \sum_i \Big\| X_i - \sum_{X_j \in N(i)} w_{ij} X_j \Big\|^2, \tag{5}$$

3. *Map each high-dimensional observation X_i to a low-dimensional vector Y_i* representing global internal coordinates on the manifold. This is done by choosing m-dimensional coordinates to minimize the embedding cost function over Y,

$$M_2(Y) = \sum_i \Big\| Y_i - \sum_j w_{ij} Y_j \Big\|^2, \tag{6}$$

where the weights w_{ij} are fixed to the values obtained in step 2. The optimization in (6) can be done by solving a sparse $m \times m$ eigenvalue problem.

The assumption of Roweis and Saul (2000) is here that one can expect the w_{ij}-characterization of local geometry in the original data space to be equally valid for local patches of the manifold. In particular, the same weights w_{ij} that reconstruct the ith data point in p dimensions should also reconstruct its embedded manifold coordinates in m dimensions.

From Figure 1(c), it is seen that the three parts of the Ranunculoid in Figure 1(a) are clearly separated with LLE, especially in the Y_2-direction.

3.4 Embedding via Graphs and ISOMAP

Some of the primary purposes of statistical embedding is to use the embedded vectors or coordinates for feature extraction, clustering and classification. The most used clustering method is probably the K-means algorithm. (See, e.g., Hastie, Tibshirani and Friedman, 2019, Chapter 14.3.) This method does not work well if the clusters form nonconvex subsets of the data space. Examples of

this are the clusters consisting of 3 concentric noisy circles in $\mathbb{R}^2$, or of the more complicated structure of the three curves in Figure 1(a).

For a given point cloud in $\mathbb{R}^p$, a method of circumventing such problems is to embed the points in a similarity graph or network. Given a set of data points $X_1, \ldots, X_n$, a similarity measure $s_{ij} \geq 0$ between X_i and X_j can simply be the Euclidean distance between X_i and X_j. The intuitive goal of clustering is to divide the points into groups such that the similarity between two groups is weak, whereas the similarity between points within a group is typically strong. If we do have similarity information between the points, a convenient way to represent this is to form a similarity graph $G = (V, E)$. Each node $v_i \in V$ in the graph represents a data point X_i. Two nodes in the graph are connected if their similarity $s_{ij} \geq \tau$ for some threshold $\tau > 0$. The similarity weights s_{ij} are used as edge weights w_{ij}. The problem of clustering can now be reformulated using the similarity graph: one wants to find a partition of the graph such that the edges between different groups have low weight, and the edges within a group have high weights.

Given a point cloud in $\mathbb{R}^p$ there are several ways of constructing a corresponding similarity graph:

(i) The ε-neighborhood graph: Here, one connects all points, and gives them weight $w_{ij} = 1$, that have pairwise distances less than ε.

(ii) k-nearest neighbor graph: Here, one can connect node v_i with node v_j if v_j are among the k nearest neighbors of v_i. Symmetrization leads to an undirected graph and $w_{ij} = s_{ij}$.

(iii) The fully connected graph: All points with positive similarity are connected with each other, and we take $w_{ij} = s_{ij}$. As an example of a similarity measure, one can take $s_{ij} = \exp(-\|X_i - X_j\|^2/2\sigma^2)$, where σ is a parameter that controls the strength of the similarity.

An early concrete graph embedding algorithm is ISOMAP (Tenenbaum, de Silva and Langford, 2000, de Silva and Tenenbaum, 2002). Apart from clustering, ISOMAP has gained considerable use as a nonlinear dimension reduction method, by combining graph representation with multidimensional scaling seeking distance preservation; see op. cit. references for details. The input is the distances $d_X(i, j)$ between all pairs of X_i and X_j of the n data points. The output is m-dimensional vectors Y_i in $\mathbb{R}^m$. The algorithm consists of three main steps:

1. *Construct the neighborhood graph* G according to (i) or (ii) above. Set edge lengths equal to $d_X(i, j)$.
2. *Compute shortest paths* $d_G(i, j)$ between all pairs in the graph G, for example, by Dijkstra's algorithm or the Floyd–Warshall algorithm (Cormen et al., 2022).
3. *Construct* m*-dimensional embeddings* Y_i by applying multidimensional scaling from Section 3.2 to the matrix of graph distances $D_G = \{d_G(i, j)\}$.

The results of applying the ISOMAP algorithm on the curves in Figure 1(a) are given in Figure 1(d). The curves are well separated both in the MDS_1 and MDS_2 directions.

3.5 Graph Representation and Laplace Eigenmaps

In this subsection, we will just give a brief presentation of Laplace eigenmaps and graph spectral theory mainly based on Belkin and Niyogi (2002, 2003). As elsewhere in this section, we start with a point cloud in $\mathbb{R}^p$. We then aim at reducing the dimension by searching for a manifold embedding of lower dimension.

In Section 5, we will *start* with a network and use graph spectral theory to find an embedding of the network in Euclidean space or on a manifold such that it can subsequently be used for purposes of clustering and classification. A few more details of graph spectral theory will be given then.

To introduce Laplacian eigenmaps, we need some more graph notation: The weighted adjacency matrix of the graph is the matrix $\mathbf{A} = \{a_{ij}\}$, $i, j = 1, \ldots, n$, where $a_{ij} = w_{ij}$ is the weight on the edge between nodes v_i and v_j. If $a_{ij} = 0$, this means that the nodes v_i and v_j are not connected by an edge. We still assume that the graph is undirected so that $a_{ij} = a_{ji}$. The degree of a node $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} w_{ij}, \tag{7}$$

with $a_{ii} = 0$. The degree matrix $\mathbf{D}$ is defined as the diagonal matrix with the degrees $d_1, \ldots, d_n$ along the diagonal.

The Laplacian eigenmap algorithm consists of three main steps:

1. *A graph is constructed* using the strategy outlined in (i), (ii) or (iii) of Section 3.4. This is used to establish the edges of the graph.
2. *The weights of the edges are determined.* Belkin and Niyogi (2003) present two choices. The first choice, as in Section 3.4, is to choose the so-called heat kernel

$$w_{ij} = \exp^{-\|X_i - X_j\|/\sigma} \tag{8}$$

if the nodes are connected using the ε-strategy of Section 3.4, and putting $w_{ij} = 0$ if they are not connected. A second alternative is just to let $w_{ij} = 1$ if v_i and v_j are connected, and $w_{ij} = 0$ if not.
3. *Find the Laplacian eigenmaps.* Assume that the graph G as constructed above is connected. If not, use the algorithm given below for each connected component. Define the Laplacian matrix by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ and $\mathbf{A}$ are, respectively, the degree and adjacency matrix defined above. The Laplacian eigenmaps are then obtained by solving the eigenvalue problem

$$\mathbf{L} f_i = \lambda_i \mathbf{D} f_i, \quad i = 0, 1, \ldots, p - 1, \tag{9}$$

with

$$0 = \lambda_0 \le \lambda_1 \le \lambda_{p-1},$$

where it is easily verified that 0 is a trivial eigenvalue corresponding to the eigenvector $f_0 = [1, 1, \ldots, 1]$. This eigenvector is left out, and the next m eigenvectors are used for an embedding in m-dimensional Euclidean space

$$X_i \to \sum_{j=1}^{m} \langle X_i, f_j \rangle f_j,$$

where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathbb{R}^p$. The Laplacian eigenmaps preserve local information optimally in a certain sense (Belkin and Niyogi, 2003).

3.6 Diffusion Maps

The representation of the Laplace matrix and a corresponding Laplace–Beltrami diffusion operator is just one way of finding a meaningful geometric description of a data set. As will be seen in this subsection, it is possible to introduce an associated Markov chain that can be used to construct coordinates called diffusion maps.

Following Coifman and Lafon (2006), it is convenient to think of the data set $\mathbf{X}$ as a measure space $(\mathbf{X}, \mathcal{B}, \mu)$ with an associated kernel k satisfying $k(x, y) = k(y, x)$ and $k(x, y) \ge 0$. In terms of Section 3.5, k may be associated with the adjacency matrix $\mathbf{A}$, and $\mu(x)$ with the discrete measure with $\mu(x_i) = 1/n$, where n is the number of observations. Generally, we let $d(x) = \int_{\mathbf{X}} k(x, y)\, d\mu(y)$, which corresponds to the definition of degree in (7).

The next step is to introduce the probability transition distribution $p(x, y) = k(x, y)/d(x)$. Then clearly $\int_{\mathbf{X}} p(x, y)\, d\mu(y) = 1$, and p can be viewed as a transition kernel of a Markov chain on $\mathbf{X}$. The operator $Pf(x) = \int_{\mathbf{X}} p(x, y) f(y)\, d\mu(y)$ is the corresponding diffusion operator.

A main idea of the diffusion framework is that running the Markov chain forward in time, or equivalently, taking larger powers of P, will allow one to reveal relevant geometric structures of different scales. We denote by p_L the L-step transition kernel.

The Markov chain has a stationary distribution, it is reversible and if $\mathbf{X}$ is finite and the graph of the data is connected, then it is ergodic (cf. Coifman and Lafon, 2006). Further, P has a discrete sequence of eigenvalues $\{\lambda_i\}$ and eigenfunctions ψ_i such that $1 = \lambda_0 \ge \lambda_1 \ge \cdots$, and $P\psi_i = \lambda_i \psi_i$. This corresponds to the eigenvalue problem in (9).

Let $\pi(x)$ be the stationary distribution of the Markov chain. Coifman and Lafon (2006) show that the family of so-called diffusion distances $\{D_L\}$ can be written as

$$\begin{aligned} D_L(x, y)^2 &= \int_{\mathbf{X}} \big(p_L(x, u) - p_L(y, u)\big)^2 \frac{d\mu(u)}{\pi(u)} \\ &= \sum_{i \ge 1} \lambda_i^{2L} \big(\psi_i(x) - \psi_i(y)\big)^2. \end{aligned} \tag{10}$$

Since the eigenvalues in (10) are less than one, the expansion can be broken off after a finite number of terms $m(\delta, L)$, where $m(\delta, L) = \max\{i \in \mathbb{N}\}$, such that $|\lambda_i|^L > \delta |\lambda_1|^L$, where δ is a measure of the precision desired in this approximation. Each component $\lambda_i^L \psi_i(x)$, $i = 1, \ldots, m(\delta, L)$ is termed a diffusion coordinate, and the data are mapped into an Euclidean space of dimension $m(\delta, L)$.

By choosing the kernel k appropriately, various diffusion operators can be obtained. We refer to Coifman and Lafon (2006) for more details.

There are a number of applications of diffusion maps. For an application to gene expression data, see Haghverdi, Buettner and Theis (2015).

3.7 Kernel Principal Components

The standard linear Fisher discriminant seeks to discriminate between two or more populations by using the global Gaussian likelihood ratio method in an attempt to separate the populations linearly by separating hyperplanes. This is of course not possible for the data in Figure 1(a). An alternative is to use a local Gaussian Fisher discriminant, which leads to nonlinear hypersurfaces (Otneim, Jullum and Tjøstheim, 2020). Still another possibility is to use transformations of the original data into nonlinear features and then try to find linear hyperplanes in this feature space. To find the linear hyperplanes, scalar products between vectors are used; this being the case both in the linear Fisher discriminant and in case there is a nonlinear feature space. As a function of the original coordinates of observations, the inner product in the feature space is termed a kernel. The support vector machine (SVM) discrimination analysis is based on such an idea.

An analog procedure can be used in so-called kernel PCA (Schölkopf, Smola and Müller, 2005). Consider a set of data vectors $X_1, \ldots, X_n$ with $X_i \in \mathbb{R}^p$ that sums to the zero-vector. Recall that in ordinary principal components analysis the estimated principal components are found by solving the eigenvalue problem $\mathbf{C}f = \lambda f$, where, $\mathbf{C}$ is the empirical $p \times p$ covariance matrix given by

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} X_i X_i^T,$$

and corresponding to the matrix $\mathbf{X}^T\mathbf{X}$ in Section 2. In kernel PCA, the starting point is to map the data vector X_i into a nonlinear feature vector $\Phi(X_i)$, $\Phi : \mathbb{R}^p \to F$, where F is an inner product space in general different from $\mathbb{R}^p$, such that $\sum_{i=1}^n \Phi(X_i) = 0$.

Consider the $n \times n$ matrix $\mathbf{K}_\Phi = \{\langle \Phi(X_i), \Phi(X_j) \rangle\}$ and the eigenvalue problem

$$\mathbf{K}_\Phi \alpha = n\lambda\alpha, \tag{11}$$

where α is the column vector with entries $\alpha_1, \ldots, \alpha_n$. Let f^l be the lth eigenvector corresponding to nonzero eigenvalues. It can be shown that (Schölkopf, Smola and Müller, 2005) for principal components extraction, one can compute the projections of the image of a data point X onto the eigenvectors f^l according to

$$\langle f^l, \Phi(X)\rangle = \sum_{i=1}^{n} \alpha_i^l \langle \Phi(X_i), \Phi(X)\rangle. \tag{12}$$

Please observe that neither (11) nor (12) requires the $\Phi(X_i)$ in explicit form. All that is required is their inner product, termed the kernel. Replacing $\Phi(X), \Phi(Y)$ by the kernel is known as the *kernel trick* (Ajzerman, Brawerman and Rozonoer, 1956, Boser, Guyon and Vapnik, 1992). The point is that one can start with a suitable kernel instead of having to do the mapping $\Phi(X)$. It can be shown by methods of functional analysis that there exists for any positive definite kernel k, a map Φ into some inner product space F, such that k constitutes the inner product of this space. This space would in general be of infinite dimension (function space), so there it is the opposite of dimensionality reduction. To show that this works and to put this into a rigorous mathematical context, one uses the framework and the properties of a reproducing kernel Hilbert space (RKHS). A recent tutorial is given in Gretton (2019).

Substituting kernel functions for $\langle \Phi(X), \Phi(Y)\rangle$ one obtains the following algorithm for kernel PCA: One computes the dot product matrix

$$\mathbf{K}_\Phi = \big\langle \Phi(X_i), \Phi(X_j)\big\rangle = k(X_i, X_j),$$

solve the eigenvalue problem for $\mathbf{K}_\Phi$, normalize the eigenvector expansion coefficient α^k and extract principal components (corresponding to the kernel k, of which there are several choices) of an observational point X by computing projections on the eigenvectors as in equation (12). The general question of choosing an optimal kernel for a given problem is unsolved both for kernel PCA and SVM.

The results of using the kernel principal component method on the data in Figure 1(a) can be seen in Figure 1(e). The curves are clearly separated along the second kernel principal component. The two dents in the two innermost curves of Figure 1(a) are also reproduced.

It is of interest to look at the curves in Figure 1(a) and their nonlinear representations when the noise is increased. This is done in Figure 2. In Figure 2(a), it is seen that with the increased noise the two innermost curves are not separated any more, but rather form a quite complicated closed curve. The principal curve for the innermost curve (with the other two removed) is seen in Figure 2(b). The overlap of the two innermost curves is clearly seen for the local linear embedding, the ISOMAP and the kernel principal component in Figures 2(c)–2(e). It seems that only kernel principal component is close to separating the original three curves. For the two others, the two innermost curves coalesce. In fact, for local linear embedding, the innermost curve more or less degenerates to two points.

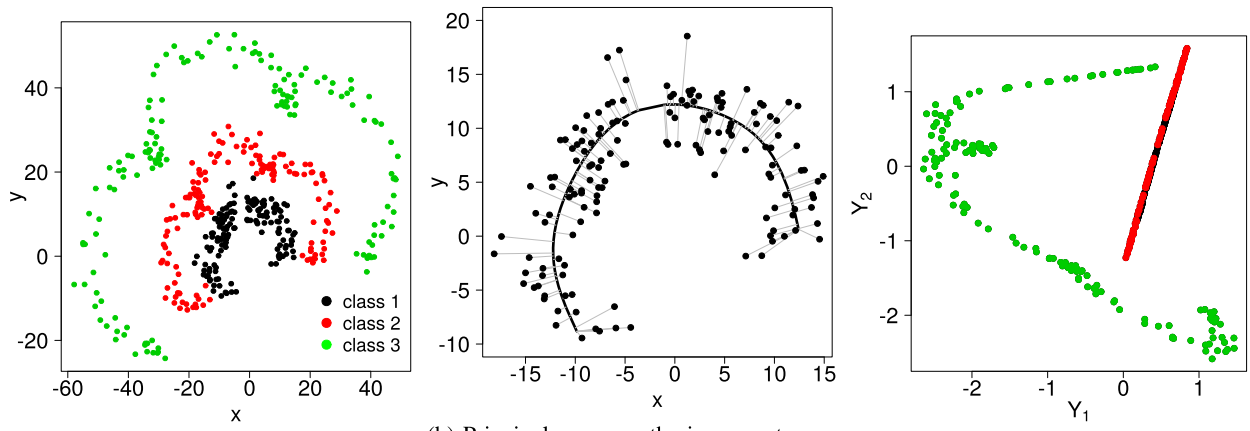
The ISOMAP picture is also interesting. The innermost curve is split into two opposite curves. This is consistent with the gap in the innermost curve in the middle of it. It is also worth noting that the loop formed on the left-hand side of the two innermost curves is reproduced at the bottom of the ISOMAP plot.

3.8 Random Projection

A number of embedding methods depends on a linear or nonlinear transformation of the data. This is for instance the case for principal components, where the transformation is found by solving an eigenvalue problem involving the data. To be more specific, let us return to the principal component method of Section 2. Here, there is a $n \times p$ data matrix $\mathbf{X}$. Estimated principal components $\hat{V}_1, \ldots, \hat{V}_m$ are then found by solving the eigenvalue problem (2). Let us denote by $\hat{\mathbf{V}}$ the $p \times m$ matrix $\hat{\mathbf{V}} = [V_1, \ldots, V_m]$ of the first m principal components. Then an embedding to the m-dimensional space is essentially done by the transformation $\tilde{\mathbf{X}} = \mathbf{X}\hat{\mathbf{V}}$. For a large p this is burdensome computationally. Similarly, the dimension of the eigenvalue problem may be in the millions for the eigenvalue problem (9) for graph representation. When cross-validation routines are added for training in a possible classification problem the amount of computations is prohibitive (Josse and Husson, 2012).

There is, however, another and very different way to avoid the high computational cost. This is via the so-called random projection method, whose rationale is based on the Johnson–Lindenstrauss lemma, Johnson and Lindenstrauss (1984). In a random projection algorithm, the transformation matrix $\hat{\mathbf{V}}$ based on the data is simply replaced by a matrix $\mathbf{U}$ such that $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{U}$, where each element of the matrix $\mathbf{U}$ is obtained by drawings from a random variable. In a normal random projection (cf. Li, Hastie and Church, 2007, Section 2.1), the elements U_{ij} are all sampled i.i.d. from a standard normal $U_{ij} \sim N(0, 1)$.This certainly implies an enormous saving of computational cost, but one may ask whether it makes sense. After all, the matrix $\mathbf{U}$ is drawn independently of the data $\mathbf{X}$.

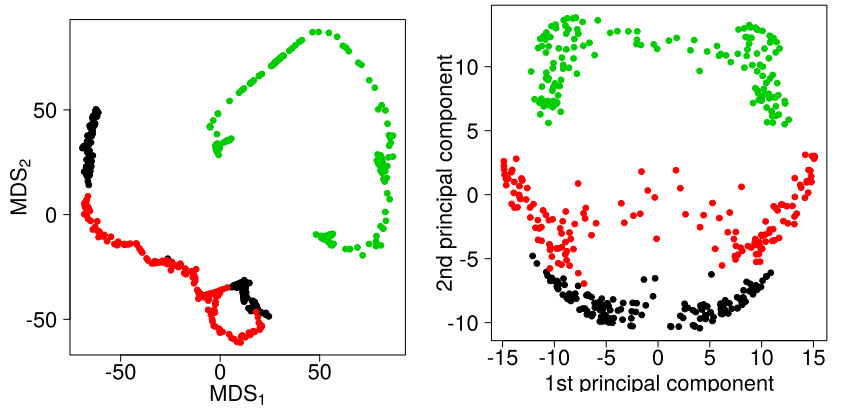
The Johnson–Lindenstrauss lemma is helpful here. This says that under relatively mild conditions distance relationships are kept approximately invariant under the random projection. There are many formulations of this lemma. We state the one used in Li, Hastie and Church (2007, Lemma 2): If $m > G(2\log n - \log\delta)/\varepsilon^2$, where $G = 4/(1 - 2\varepsilon/3)$, then with probability at least $1 - \delta$, and remarkably, independent of $\mathbf{X}$ and p, the squared l_2



(a) The raw data.

(b) Principal curve on the innermost curve of Fig. 2a.

(c) Local linear embedding.



(d) ISOMAP.

(e) Kernel principal components (with a Bessel kernel).

FIG. 2. *Four different embedding methods applied to three parametric curves from the so-called Ranunculoid and perturbed by more Gaussian noise than in Figure* 1(*a*) (*we have used a standard deviation of* 2 *instead of* 1/2 *for the noise here*).

distance between any pair of projected data points can be approximated within a factor of $(1 \pm \varepsilon)$, $(0 < \varepsilon < 1)$, of the squared l_2 distance of the original data after normal random projections. Alternative formulations and proofs can be found in, for example, Ghojogh et al. (2021).

Several attempts have been made to apply the random projections to clustering, classification and regression. Perhaps not unexpectedly, it has been found that random projections may fail exactly because the transformation $\mathbf{U}$ is constructed without taking the intrinsic structure of the original data into account. This issue has been sought avoided in various ways (Cannings and Samworth, 2017, Xie, Li and Xue, 2018).

3.9 A Few Other Techniques

There are several other alternative methods in nonlinear dimension reduction. Perhaps the most used one is Independent Components Analysis (ICA). The main concepts of the method are described in a much cited paper by Hyvärinen and Oja (2000).

In ICA, the aim is again to obtain latent factors, and in format the decomposition is the same as the PCA decomposition except that the components are now required to be independent. One might remark that ICA essentially starts from a factor analysis solution to dimension reduction and looks for rotations that lead to independent components. From this point of view, ICA is just another factor rotation along with the traditional varimax and quartimax.

Two other methods will be very briefly mentioned. These are both neural network based methods. One of them consists in so-called autoencoding in deep neural networks, and can be represented by Hinton and Salakhutdinov (2006). The other is the method of self-organizing maps, which can be said to have originated by another much cited paper, Kohonen (1982).

4. TOPOLOGICAL EMBEDDINGS AND TOPOLOGICAL DATA ANALYSIS (TDA)

The present section concerns topological embeddings and data analysis. We will divide our exposition in three parts, manifold learning, persistent homology, and finally the Mapper algorithm. It is the persistent homology part

that is usually identified with TDA. Our point of departure is in all cases a point cloud in $\mathbb{R}^p$. In part one, the objective is to examine whether there is a possibility of embedding the point cloud in a lower-dimensional manifold. In the two other parts, the aim is to try to find additional topological features that may characterize the point cloud and its embedding. In order to avoid an overlong paper, parts of the TDA survey have been moved to the Supplementary Material (Tjøstheim, Jullum and Løland, 2023). A main introductory reference to manifold learning and TDA is Chazal and Michel (2021).

4.1 Manifold Learning

Already in the Pearson (1901) treatment of principal components, the point cloud of data is embedded on a hyperplane in $\mathbb{R}^p$. The approach of ISOMAP and local linear embedding are early examples of representing the data in a lower-dimensional manifold.

A main aspect of manifold learning is that one looks for a non-Euclidean subspace to make an embedding that may not easily be achieved in an Euclidean space $\mathbb{R}^m$, but more efficiently on a manifold. One trivial example is the case where the point cloud in the plane is concentrated on a circle with only small additional perturbations. The data can then essentially be reduced from two-dimensional space (the plane), not to the line ($\mathbb{R}$), but to the circle, which is a one-dimensional manifold. For a more complex example, we refer to the Ranunculoid of Figure 1(a).

In the more general case, manifold learning consists in finding a smooth compact submanifold S of $\mathbb{R}^p$ on which the point cloud data may be reasonably located.

One may estimate S by trying to cover the data cloud by a collection of balls of radius ε, such that

$$\hat{S} = \bigcup_{i=1}^{n} B(X_i, \varepsilon), \tag{13}$$

where n is the number of observations and $B(X_i, \varepsilon) = \{x : \|x - X_i\| \le \varepsilon\}$, and where X_i is observation number i of the point cloud. This was suggested by Devroye and Wise (1980) in another context. If the observations X_i are all exactly on S and with ε depending on n, it is possible to prove convergence of $\hat{S}$ to S at the rate of $O_P(\log n/n)^{1/r}$, where r is the dimension of S, and the distance between S and $\hat{S}$ is the Hausdorff distance between sets.

It is not likely that a sample will fall precisely on S. A more realistic model is that one observes $Y_i = X_i + \delta_i$, where X_i comes from a distribution with support on S, and δ_i are samples from a noise distribution. In this case, the convergence rate of the estimation of S is very slow (Genovese et al., 2012). An interesting example of two-dimensional data, but where there is a set S of dimension 1 with a high concentration of data, is the data set of galaxies treated in Chen et al. (2015a, 2015b).

In a theoretical analysis, often the dimension r of the embedding manifold is assumed known. In practice, one may need to estimate r; see Levina and Bickel (2004), Little, Maggioni and Rosasco (2011) and Kim, Rinaldo and Wasserman (2019). It may be possible to estimate an r-dimensional and high density region R that is close to S. One way to make this more precise is through the idea of density ridges.

The ridge set can then be estimated by the ridge of the kernel density estimator. The properties of this estimator is studied in Genovese et al. (2014) and Chen, Genovese and Wasserman (2015). A popular algorithm for finding the ridge set estimator was given by Ozertem and Erdogmus (2011), the so-called SCMS algorithm. Recently, Qiao and Polonik (2021) proposed two novel algorithms for estimating ridge lines in ridge regression. They provide theoretical guaranties for their convergence in probability using the Hausdorff distance between the estimated and theoretical ridge. There are no analog results for the SCMS algorithm.

4.2 Persistent Homology and Persistence Diagrams

In our context, the concept of homology can be seen as coming from a desire to answer the question of whether two sets are topologically similar. For instance, is an estimate $\hat{S}$ of S topologically similar to S, or is it at all possible to find an estimate of S that is topologically similar to S? The answer to this question depends on what is meant by "similar".

Two sets S and T equipped with topologies are homeomorphic if there exists a bicontinuous map from S to T. Markov (1958) proved that, in general, the question of whether two spaces are homeomorphic is undecidable for dimension greater than 4.

However, it is possible to use the weaker notion of homology, and it is much easier to determine whether two spaces are homologically equivalent. Strictly speaking, homology is a way of defining topological features algebraically using group theory; see, for example, Carlsson (2009) for a precise definition. Intuitively, it means that one can compare connected components, holes and voids for two spaces. The zeroth-order homology of a set corresponds to its connected components. The first-order homology corresponds to one-dimensional holes (like a ring structure), whereas the second-order homology corresponds to two-dimensional holes (like a soccer ball) and so on for higher dimensions. If two sets are homeomorphic, then they are homologically equivalent, but not vice versa.

Homology is a main topic of TDA. To establish a link with the previous subsection, consider the estimate $\hat{S} = \bigcup_{i=1}^{n} B(X_i, \varepsilon)$ of equation (13). One of the first results about topology and statistics is due to Niyogi, Smale and

Weinberger (2008). They showed that under certain technical conditions the set $\hat{S}$ has the same homology as S with high probability.

In many ways, topological data analysis has been identified with the subject of persistent homology. This is concerned with the homological structure of data clouds at various scales of the data, and to see how the homology changes (how persistent it is) over these various scales; cf. also Section 3.6. A main introductory source is Chazal and Michel (2021).

The field of TDA is new. It has emerged from research in applied topology and computational geometry initiated in the first decade of this century. Pioneering works are Edelsbrunner, Letscher and Zomorodian (2002) and Zomorodian and Carlsson (2005). An early survey paper at a relatively advanced mathematical level but with a number of interesting and illustrative examples is Carlsson (2009). Wasserman (2018) and Chazal and Michel (2017) are somewhat less technical and more oriented toward statistics; see also Ghrist (2018).

For our purposes of statistical embedding, TDA brings in some new aspects in that topological properties are emphasized in the embedding. This is done to start with in so-called persistence diagrams, which depict the persistence, or lack thereof, of certain topological features as the scale in describing a data cloud changes. In complicated situations, persistence diagrams can be computed from simplical complexes. This is a particularly interesting concept since it generalizes the embedding of a point cloud in a graph. A one-dimensional simplical complex can be identified with a graph, whereas generalizations allow for describing cycles and voids of the data.

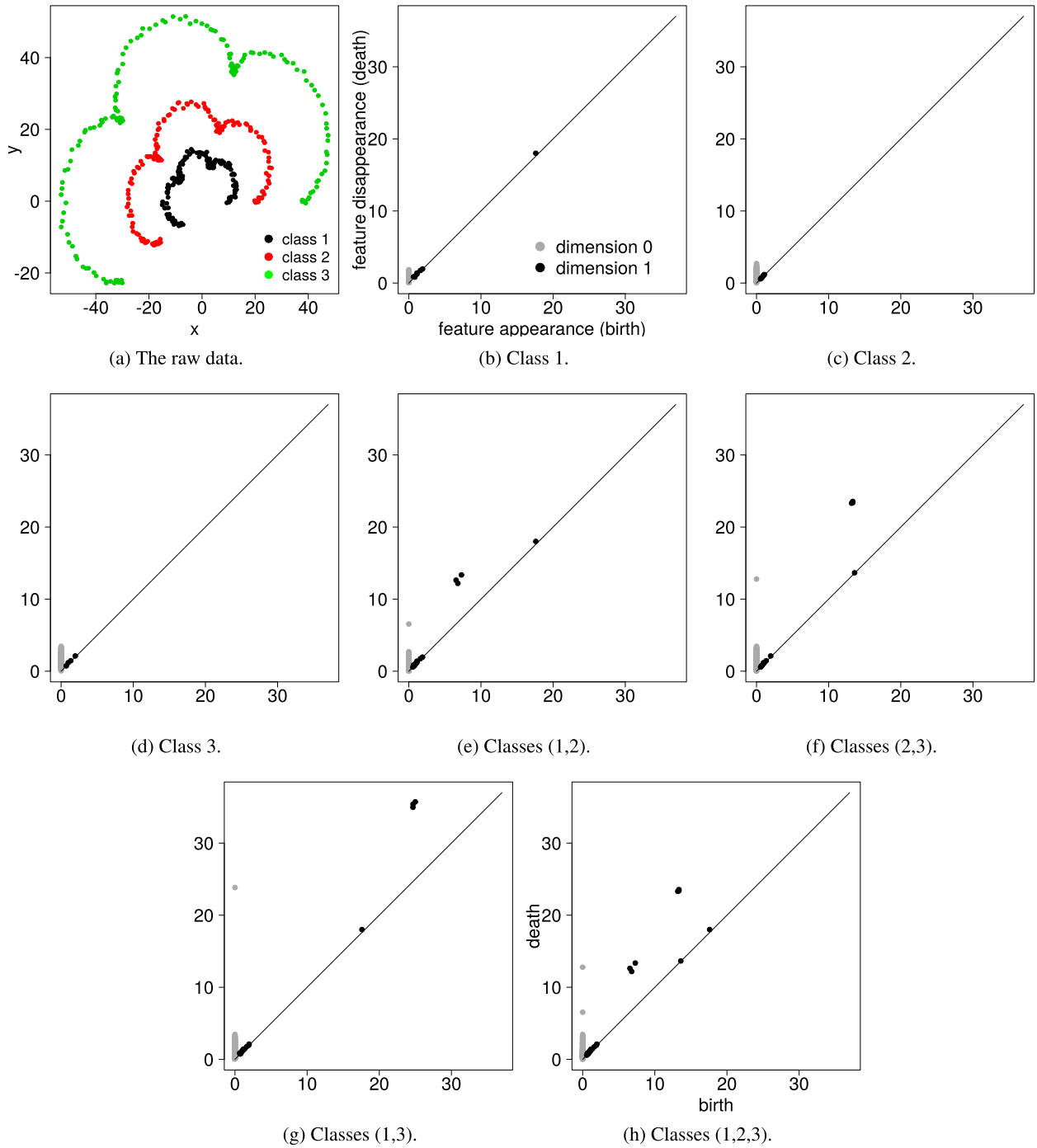
To introduce the persistence diagram, recall the estimator $\hat{S}$ in (13) as a union of balls $B(X_i, \varepsilon)$ of radius ε. One may question what happens to this set as the radius of the balls increases. Consider, for example, a data cloud that contains a number n of isolated points that resembles a circular structure. Let each point be surrounded by a neighborhood consisting of a ball centered at each data point and having radius ε. Then initially and for a small enough radius ε, the set $\bigcup_{i=1}^{n} B(X_i, \varepsilon)$ will consist of n distinct connected sets (homology zero). But as the radius of the points increases, some of the balls will have a nonzero intersection, and the number of connected sets will decrease. For ε big enough, one can easily imagine that the set $\bigcup_{i=1}^{n} B(X_i, \varepsilon)$ is large enough so that it covers the entire circular structure obtaining an annulus-like structure of homology 1, but such that there still may exist isolated connected sets (of homology 0) apart from the annulus. Continuing to increase the radius, one will eventually end up with one connected set of zero homology.

This process then involves a series of births (at ε-radius zero n sets are born) and deaths of sets as the isolated sets coalesce. A useful plot is the persistence diagram, which has the time (radius) of birth on the horizontal axis and the time (radius) of death on the vertical axis. The birth and death of each feature is represented by a point in the diagram. All points will be above or on the diagonal then.

We will go through the steps of this procedure in the case of the noisy Ranunculoid structure of Figure 1(a). We will start by considering each of the three curves, then pair of curves and finally all three curves. The corresponding persistence diagrams are displayed in Figure 3, and these diagrams furnish the topological embedding signature of the data, which is rather different from, and presents additional information compared to, the embeddings in Figures 1 and 2.

Consider first the individual curves in Figures 3(b)–3(d) (Figure 3(a) is identical to Figure 1(a)). Here, class 1, 2 and 3 in Figures 3(b)–3(d) represent the persistence diagram of the innermost to the outermost curves, respectively. The gray points represent sets of homology zero (isolated sets) and black points represent sets of homology one, that is, one-dimensional holes. The gray column at the left is just the time of death for all the sets around the individual points as the radius for the individual neighborhoods increase. Naturally, the column is highest for the outermost curve in Figure 3(d), where the distances between points are largest. The black points at the right-hand side of the columns mark small holes that temporarily arise in this process due to indents in the point spreads. For the innermost curve, there is a black point at the far right with a short lifetime. This is due to the opening in this curve, which is just great enough for there to form an annulus as the radii increase.

Next, to the diagram of the pairwise curves: The pair $(1, 2)$ consists of the two innermost curves, and the persistence diagram is displayed in Figure 3(e). The points of curve 1 can again be found. In addition, at birth time, there is a gray point above the gray column. This is just due to the fact that there are two curves at the starting point. As time (and radii) increase the two curves coalesce and we have a death at the gray point above the gray column. The three black points being born at approximate time 6 and living for about time 6 to time 12 come from holes that are created as curve 1 and 2 are approximating each other. The explanation for the pair $(2, 3)$ is much the same. In this case, it takes more time before the curves 2 an 3 coalesce, so the gray point at time zero are farther up. Here, too, 3 holes are formed as the curves 2 and 3 approach each other. One hole has very short lifetime, it is almost on the diagonal, where as the two others almost coincide and have far longer lifetime. This has to do with the different levels of indention on the two curves. Finally, for the pair $(1, 3)$, the gray point at zero is even farther up, reflecting the increased distance between the curves 1 and 3. Again the pattern of curve 1 is dominating as for the pair $(1, 2)$. The indents of curve 1 are small in



(a) The raw data. (b) Class 1. (c) Class 2.

(d) Class 3. (e) Classes (1,2). (f) Classes (2,3).

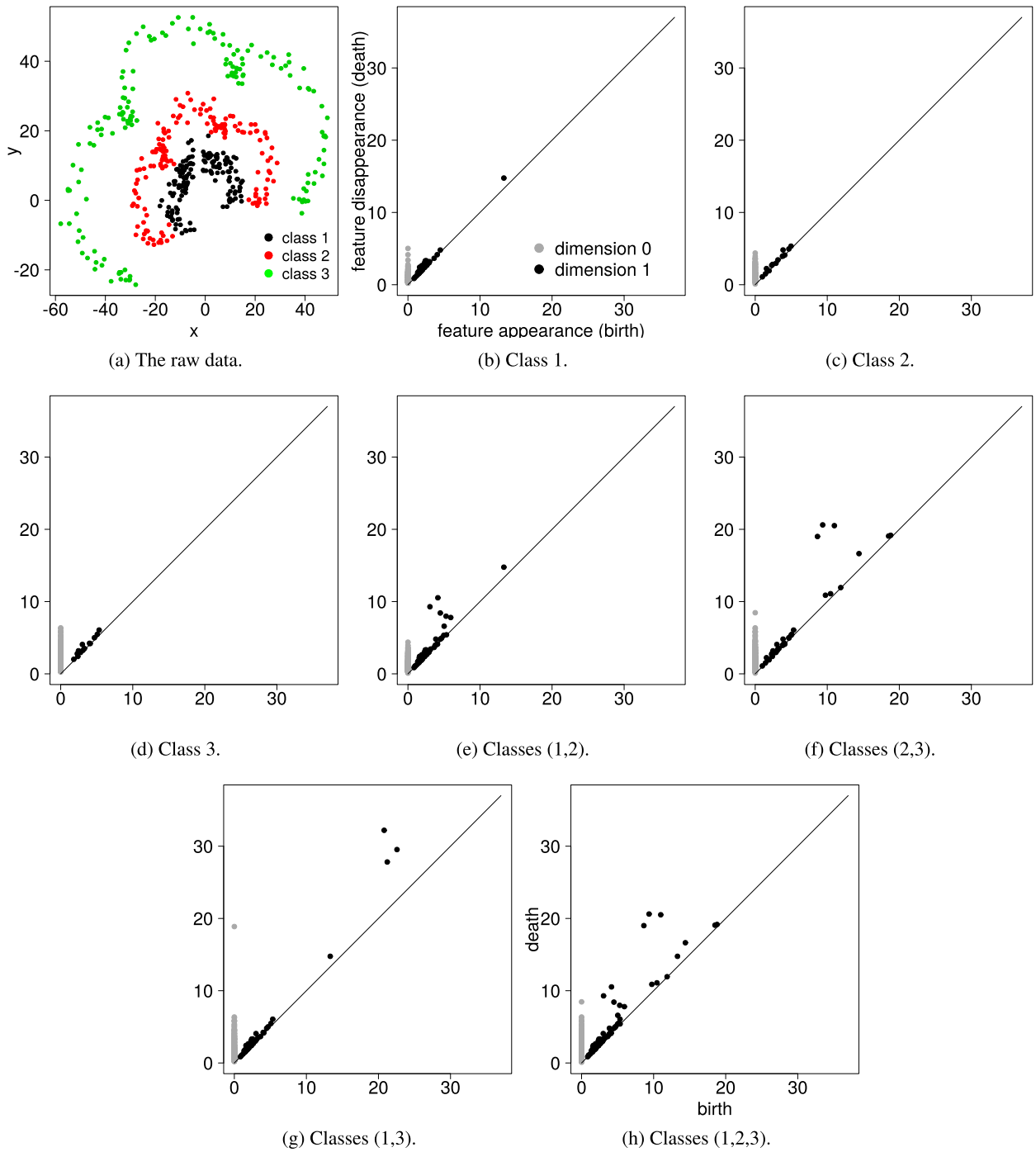
(g) Classes (1,3). (h) Classes (1,2,3).

FIG. 3. *Persistence diagrams for combinations of classes* 1, 2 *and* 3.

comparison with the indents of curve 3, and this explains that it takes longer time for holes to appear as these two curves are approaching each other.

The diagram for the triple of curves $(1, 2, 3)$ in Figure 3(h) is roughly obtained by superposition of the pattern for the pairwise curves. There is a difference at birth time zero, though. The uppermost point for the pair $(1, 3)$ has disappeared. The explanation is obvious. The curves 1 and 2 coalesce first due to least distance between them. Curve 3 is then coalescing with the set combined by curve 1 and 2, which has a distance from curve 3 equal to the distance between 2 and 3, such that the second gray point at zero correspond to the gray point at zero for the pair $(2, 3)$.

One can also construct persistence diagrams for the more noisy curves of Figure 2. This is shown in Figure 4. The pattern is a bit more complex as is expected, but the individual points can be interpreted as before. In particular, due to the more irregular patterns of the noisy curves, the gray columns to the left extend farther up, and the birth



(a) The raw data. (b) Class 1. (c) Class 2.

(d) Class 3. (e) Classes (1,2). (f) Classes (2,3).

(g) Classes (1,3). (h) Classes (1,2,3).

FIG. 4. *Persistence diagrams for combinations of classes* 1, 2 *and* 3.

of holes of dimension 1 has an earlier birth, there are more of them and they exhibit a somewhat more complex pattern.

The idea is that this description of a point cloud in the plane, as indicated above, may be generalized to higher dimensions and much more complicated structures with multiple holes and voids of increasing homology. The number of sets of different homologies are described by the so-called Betti numbers, $\beta_0, \beta_1, \ldots$. In a non-technical jargon, β_0 is the number of connected components ($\beta_0 = n$, n being the number of isolated points in the start of our example), β_1 is the number of one-dimensional holes, so $\beta_1 = 1$ if there is only one connected ring structure, and $\beta_0 = 1$, $\beta_1 = 0$ when the radius is so great that there is only one connected set altogether. The hole is one-dimensional since it suffices with a one-dimensional curve to enclose it, whereas the inside of a soccer ball is two-dimensional, it can be surrounded by a two-dimensional surface, and has $\beta_0 = 1$, $\beta_1 = 0$ and $\beta_2 = 1$. A torus has $\beta_0 = 1$, $\beta_1 = 2$, $\beta_2 = 1$. In Figures 3

and 4, it is a trivial exercise to find the Betti numbers (0 or 1) for any chosen interval of time (radius) of these figures.

The extension of the persistence diagrams to more general structures requires relatively advanced use of mathematical tools. We only indicate some main concepts in Section 1 of the Supplementary Material (Tjøstheim, Jullum and Løland, 2023). Section 1 of that Supplementary Material is concluded by formulating some explicit and open statistical problems in TDA.

There are many applications of TDA in general and of persistence diagrams in particular. Two recent applications to cancer research are Bukkuri, Andor and Darcy (2021) and Crawford et al. (2020), where the latter introduces a variation of a persistent homology transformation to facilitate the difficulties in integration with traditional statistical models. In this type of cancer study, time series are important. The use of TDA to analyze time-series data is discussed in Ravisshanker and Chen (2019).

4.3 The Mapper

In Section 3, we have outlined a number of methods for projecting high-dimensional data to lower dimensions, thus making the projected data more amenable for characterization such as, for example, clustering and classification. Some of these methods strive to make the distance between points invariant, others not. But in all cases there is a risk of missing important topological information during the projection operation. The Mapper algorithm suggested in a seminal paper by Singh, Memoli and Carlsson (2007) tries to handle this issue by back-projecting the characterization in the lower-dimensional space to the original space by considering preimages of the clustering, say, in the low-dimensional space. More precisely, the Mapper algorithm consists of the following steps:

Consider a point cloud of data $\mathbf{X}$, and let f be the mapping of $\mathbf{X}$ to a lower-dimensional space, obtained by principal components or one of the other dimensionality reduction methods of Section 3. Let $\mathbf{Y} = f(\mathbf{X})$ be the set of data points in the lower-dimensional space, often assumed to be $\mathbb{R}^m$ or even $\mathbb{R}^1$. Then:

1. Cover the range of values $\mathbf{Y} = f(\mathbf{X})$ by a collection $\mathcal{U} = \{U_1, \ldots, U_S\}$ of intervals, or possibly more general sets, which overlap.
2. Apply a clustering algorithm to each of the preimages $f^{-1}(U_s)$, $s = 1, \ldots, S$. Even though U_s may be connected, $f^{-1}(U_s)$ of course may not be connected due to the potential complicated topological relationships in the original space. This defines a pullback cover $\mathcal{C} = \{C_{1,1}, \ldots, C_{1,k_1}, \ldots, C_{S,1}, \ldots, C_{S,k_S}\}$ of the point cloud $\mathbf{X}$, where $C_{s,k}$ denotes the kth cluster of $f^{-1}(U_s)$.
3. Each node $v_{s,k}$ of the Mapper corresponds to one element $C_{s,k}$, and two nodes $v_{s,k}$ and $v_{s',k'}$ are connected if and only if $C_{s,k} \cap C_{s',k'}$ is not empty.

The algorithm results in a graph (or more generally a simplical complex). The essential design problems consist in the choice of the transformation f and the covering $U_1, \ldots, U_S$ in the lower-dimensional space. Unfortunately, according to Chazal and Michel (2021), Mapper is quite sensitive to the choice of covering, the number of covering sets and the overlap between them, making the method potentially unstable. A classical strategy consists in exploring some range of design parameters, and selecting the ones that turn out to provide the most informative output from the user's perspective.

There is a statistical analysis including parameter selection in Carrière, Michel and Oudot (2018). They demonstrate aspects of statistical convergence and ensuing optimality problems. They also derive confidence regions of topological features such as loops and flares.

The Mapper algorithm has found many applications, especially for its capability of detecting loops and flares in the mapping of the original data space. A recent example of applications to cell description is given in Carrière and Rabadán (2020).

5. EMBEDDING OF NETWORKS

In Sections 3.4 and 3.5, graphs (or networks) were used as a tool in embedding a point cloud in $\mathbb{R}^p$, making it possible among other things to do cluster analysis involving nonconvex clusters. In the present section, the *starting point* is a network or collection of networks, and the task is to embed the network in an Euclidean space $\mathbb{R}^m$ or to map it to a manifold. This is used to obtain a vector representation of each node of the network.

Why is it important to be able to embed a network in such a way? The main reason is simply that for many purposes it is easier to work with a set of n vectors than with a network consisting of n nodes. One has standard methods for dealing with vectors. For example, one can do clustering of vectors, which in a social network could correspond to finding and grouping communities in the network. And one can also compare and classify networks by looking at their embedded sets of n-dimensional vectors.

With the increasing use of the internet and big data, the analysis of large networks is becoming more and more important. There is a very wide field of applications ranging over such diverse areas as, for example, finance, medicine and sociology, including criminal networks. A broad overview can be found in the recent book by Newman (2020). A fine detailed survey is Cui et al. (2019).

With ultra high dimension and very large data sets, there is a need for fast methods. With the recent technique of Skip-Gram, described in some detail in Section 5.3 and in Section 2 in the Supplementary Material (Tjøstheim, Jullum and Løland, 2023), one is able to handle networks with millions of nodes and billions of edges such that each

node is represented by a vector of dimension 500–600, say. On such vectors, one can use standard discrimination and clustering. One may also do further embedding to lower-dimensional vectors, as described in Section 6, to visualize data of very high dimension.

In our survey of network embedding methods, we will start with spectral graph methods in Section 5.2 after a brief introduction on characterization of graphs in Section 5.1. The spectral method requires solving an eigenvalue problem, and this puts a limitation on the number of nodes and edges. This restriction is to a large degree bypassed in neural network based methods, in particular in the Skip-Gram algorithm. This algorithm was originally introduced in natural language analysis, which has independent interest in that the words in a language text can be embedded in a vector in $\mathbb{R}^m$ reflecting not only the word count in a text but also the syntax of the text. A language text is not a network and, therefore, some details of the embedding analysis of a language text are covered in Section 2 of the Supplementary Material. Ideas and methods developed in such a framework have proved vitally important, however, for fast and efficient embedding of networks as is demonstrated in Section 5.3. That section is chiefly concerned with symmetric undirected networks, but briefly mentioning directed networks, heterogeneous networks and dynamic networks, where there are many open statistical and data processing problems, in the ensuing sections.

There are several issues of statistical interest related to embedding of networks. One may therefore think that there is a potential synergy effect that both the statistics and machine learning community could benefit from. We will try to make this more clear in the sequel. One issue is the lack of statistical modeling and inference in the algorithmic machine learning industry. It is important to realize that there now exists a growing statistical literature that is in process of being integrated in algorithms on finding communities in networks. We refer to Sections 5.2.4 and 5.7.1. See also the three keypoints formulated in the concluding remarks in Section 7.

5.1 A Few Elementary Concepts of Graph Theory and Matrix Representations

We have already introduced some elementary graph concepts in Sections 3.4 and 3.5. In this brief introductory section, we supplement these to more fully explain the spectral based clustering algorithms for networks.

We consider a graph $G = (V, E)$, where V and E are the sets of nodes and edges, respectively. The graph is supposed to be undirected, which means that an edge goes in both directions between two neighboring nodes. Let $n = |V|$ be the number of nodes in (V, E). Then the graph can be represented by a $n \times n$ matrix $\mathbf{M}$, such that an element M_{ij} of this matrix represents some property of the pair of nodes v_i and v_j. When V is large, this matrix may be huge. Later, representation matrices of dimension $n \times m$ will be introduced where $m \ll n$. Diagonal elements M_{ii} encode information of the node v_i only, such as the degree of v_i (number of edges emanating from v_i or more generally as in equation (7) for a weighted graph). A simple example of such a matrix is the adjacency matrix A, which was mentioned in Section 3.5.

An adjacency matrix A for an undirected graph is symmetric with real eigenvalues, both negative and positive. In many applications, it is useful to have a nonnegative definite matrix. One example of such a matrix is the Laplace matrix, a version of which was introduced in Section 3.5 for a general weighted undirected graph. It is given by

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{14}$$

where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D} = \operatorname{diag}(d_i)$ is the diagonal matrix having the degree of the nodes along the diagonal.

The normalized Laplacian L_N is defined by

$$L_{N,ij} = \begin{cases} 1 & \text{if } i = j, \\ -1/\sqrt{d_i d_j} & \text{if } i \text{ and } j \text{ are adjacent}, \\ 0 & \text{otherwise.} \end{cases}$$

This matrix can also be written $\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. It is nonnegative definite and it has all its eigenvalues $0 \leq \lambda \leq 2$.

5.2 Spectral Embedding and Graph Clustering

A basic task in network clustering is community structure detection. It is perhaps best thought of as a data technique used to throw light on the structure of large-scale network data sets, such as social networks, web data networks or biochemical networks. It is normally assumed that the network of interest divides naturally into subgroups, and the task is to find those groups.

For the purpose of community grouping and division, a criterion is required that can measure both the internal structure within each group, where the goal is to maximize the dependence between members of a group, but also such that the dependence between each group is minimized. There are two main methods for doing this, either by minimizing the so-called cut between the groups, the mincut problem or by maximizing the modularity. Both are discussed below using network spectral embedding.

5.2.1 *Minimizing the cut functional.* A useful tutorial on spectral clustering is given by von Luxburg (2007). A more recent alternative account is given in Zheng (2016).

Given a graph $G = (V, E)$ with adjacency matrix $\mathbf{A}$, we would like to find a partition of V in groups $V_1, \ldots, V_k$

such that the number of edges between each group is minimized. This leads to the mincut problem.

Let $W(V_i, V_j) \doteq \frac{1}{2}\sum_{m\in V_i, l\in V_j} w_{ml}$, where w_{ml} is the weight for the edge between the nodes v_m and v_l. In the unweighted situation, w_{ml} is 1 if there is an edge between v_m and v_l and 0 if not. Let $\bar{V}_i$ be the complement of V_i. The mincut approach to clustering is simply defined for a given k by choosing the partition $V_1, \ldots, V_k$, which minimizes the normalized cutsize

$$\mathrm{NCut}(V_1, \ldots, V_k) \doteq \frac{1}{2}\sum_{i=1}^{k} \frac{W(V_i, \bar{V}_i)}{\mathrm{vol}(V_i)} = \sum_{i=1}^{k} \frac{\mathrm{cut}(V_i, \bar{V}_i)}{\mathrm{vol}(V_i)},$$

where $\mathrm{vol}(V_i) = \sum_{v_l\in V_i} d_l$, d_l being the weighted degree of v_l. A similar criterion is the RatioCut criterion (Wei and Cheng, 1989).

The normalized Laplace matrix can be written as $\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. Let $\mathbf{H}$ be the $n \times m$ matrix whose columns are the m eigenvectors corresponding to the m smallest (nonzero) eigenvalues of $\mathbf{L}_N$. The m-dimensional row vectors of $\mathbf{H}$ then constitute an embedding of the nodes of the graph minimizing the normalized cut-functional of the graph. These embedding vectors are next used as a point of departure for clustering and classification of nodes.

5.2.2 *Maximizing the modularity.* Modularity is an alternative concept in the use of spectral methods in clustering. Modularity was introduced by the highly cited papers of Girvan and Newman (2002) and Newman and Girvan (2004), and after that has been further developed as in Newman (2006). See also Bickel and Chen (2009) for an alternative using a nonparametric point of view.

It was seen in the previous subsection that the principle underlying the cut-size algorithms is that a good division of a network is one in which there are few edges between communities. Newman (2006) states that this is not necessarily what one should look for. He argues that a good division is one in which there are fewer than *expected* edges between communities.

This idea then is quantified using the measure of modularity. Assume first that there are two potential classes. Again, we suppose that the network contains $n = |V|$ nodes, and we introduce the vector s, whose ith component is given by $s_i = 1$ if node v_i belongs to group 1 and $s_i = -1$ if it belongs to group 2. The edge between nodes v_i and v_j is characterized by the adjacency matrix $\mathbf{A}$. The element A_{ij} then represents the "number of edges" between v_i and v_j. The expected number of edges between v_i and v_j if edges are placed at random is $d_i d_j/2d$, where d_i and d_j are the degrees of the nodes and $d = \frac{1}{2}\sum_i d_i$ (undirected network). The modularity is then defined by

$$Q = \frac{1}{4d}\sum_{ij}\left(A_{ij} - \frac{d_i d_j}{2d}\right)s_i s_j = \frac{1}{4d}s^T\mathbf{B}s, \tag{15}$$

where the matrix $\mathbf{B}$ is defined by

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2d}.$$

This is easily generalized to the case of k classes, and the modularity is maximized by computing the eigenvectors of the $\mathbf{B}$ matrix. Corresponding to $\mathbf{H}$, let $\mathbf{S}$ be the $n \times m$ matrix whose columns are the eigenvectors corresponding to the top m eigenvalues of $\mathbf{B}$. The n m-dimensional row vectors of $\mathbf{S}$ then constitute an embedding of the n nodes of the network maximizing the modularity.

5.2.3 *The Louvain method for community detection.* The so-called Louvain method for community detection based on modularity was introduced in a paper by Blondel et al. (2008). They start with a network with n nodes, and where each node defines a community. Then one goes successively through the nodes of the net and for each node v_i, with neighbors v_j one investigates the gain in modularity if v_i is removed from its community and placed in the community of v_j. The node v_i is then placed in the community for which this gain is maximum (in case of a tie, a breaking rule is used). An updating formula for the change in the modularity Q is given in Blondel et al. (2008). This is continued until the whole graph has been covered. In the next round, the procedure in the first round is repeated, but this time with the communities formed in the first step as entities. This is continued until there is no increase in Q.

There is no eigenvalue problem that needs to be solved in this algorithm. This makes it possible to apply the Louvain algorithm for substantially larger networks. One example that the authors refer to is a mobile phone company with a network composed of 2.6 million users.

5.2.4 *Statistical modeling, SBMs and finding communities.* The methods in Sections 5.2.1–5.2.3 all belong to the algorithmic approach. An intuitively reasonable object function is maximized or minimized to find communities in a network. This is in line with the most popular approach to statistical embedding, where as such no statistical model is involved. There are no parameters that should be estimated, and in terms of which the fit of the model can be assessed.

These two different approaches, the algorithmic versus the statistical modeling one, have recently been discussed in several papers. The most recent one seems to be Peixito (2021), who is staunchly critical to the algorithmic approach in general and to the methods of finding communities in Sections 5.2.1–5.2.3 in particular. The author demonstrates that maximizing the modularity Q of Section 5.2.2 could lead to falsely finding communities in a completely random environment. On the other hand, he gives examples where in given situations use of Q leads to underestimation of the number of communities. This

may be part of a general problem of some machine learning algorithms, at least it is something that deserves closer attention, as indicated in the third keypoint of Section 7.

Peixoto argues for parametric statistical models from which networks can be *generated*, and where the structure of the net depends on the type of statistical models used and on the values of the parameters of these models. The generated model structures can be compared to real life networks, and parameters may be estimated by seeking to fit a generated model structure to the real life data. The most used statistical model is probably the so-called stochastic block model, SBM, where a block may be thought of as a community. The history of these models goes back at least to Holland, Laskey and Leinhardt (1983). Another early publication for a slightly more general model is Hoff, Raftery and Handcock (2002). There are several papers on the theoretical aspects of the SBM that will be briefly mentioned in Section 5.7.1. A review paper is Lee and Wilkinson (2019). Here, we will base ourselves on Karrer and Newman (2011) and Newman and Reinert (2016), since they are directly and explicitly related to maximizing modularity Q, Newman being the main originator of the modularity principle.

In the simplest undirected stochastic block model, each of the n nodes is assigned to one of k blocks (communities), and undirected edges are placed independently between node pairs with probabilities that are a function only of the block membership of the nodes. If we denote by b_i the block to which node i belongs, then one can define a $k \times k$ matrix of probabilities such that the matrix element p_{v_i,v_j} is the probability of an edge between nodes i and j. These probabilities are the k^2 parameters of the model, and there are several ways of estimating them for a given real data network.

Unfortunately, however, this simple block model does not work well for many real world networks, and tends to give bad results in obtaining plausible communities. There are generalizations of the simple SBM model, but they may lead to models that are far more difficult to estimate. One relatively simple generalization is the degree corrected stochastic block model (dcSBM) that seems to work much better on real life networks. The dcSBM was suggested by Karrer and Newman (2011). It allows for heterogeneity in the number of degrees for the nodes, which is a phenomenon that is often observed in practice, whereas the simple SBM results in a model where each node has the same expected degree, which in many cases is clearly unrealistic. Karrer and Newman also demonstrate that in a certain approximative sense the dcSBM can be related to the modularity function Q from equation (15).

5.3 Embedding a Network Using Skip-Gram

For large networks, the cut-size spectral clustering method and the modular method (possibly with the exception of the Louvain method) run into problems because it is costly to solve eigenvalue problems for the high dimensions that may occur in network embedding. These problems are to a large degree alleviated in a neural net based Skip-Gram procedure. This procedure was first developed in word embedding in a language text (from this the nomenclature "Skip-Gram"). Here, the eigenvalue problem is eliminated altogether, and the neural net training is speeded up using so-called negative sampling or hierarchical processing. See also Section 2 in the Supplementary Material (Tjøstheim, Jullum and Løland, 2023), which contains a brief account of natural language embedding, and may be of some independent interest.

A concept of neighborhood is needed to extend word processing to networks where words are replaced by nodes and the vocabulary with the network itself. In natural language processing, defining a neighborhood of a word in a text is not difficult: simply taking n_1 and n_2, $n_i \geq 0$ context words in front and after the word, respectively. Before embarking on the neighborhood problem, partly to define notation, let us formally write up the analog of the Skip-Gram model, presented in some detail in the language analysis in Section 2 of the Supplementary Material, for a network. The notation $N(v)$ is used for the neighborhood of a node $v \in V$ in a network $G = (E, V)$. Neighborhoods are more precisely defined in Section 5.3.2. The analysis to be presented next applies mainly to the static undirected case. Extensions to directed, heterogeneous and dynamic networks are briefly discussed in separate subsections.

We let f be the mapping from V to the embedding feature space $\mathbb{R}^m$. The goal is to associate each node v in V with a feature vector $f(v)$ in $\mathbb{R}^m$. When representing the whole network in this way, we obtain an $n \times m$ dimensional matrix with $n = |V|$.

5.3.1 *The Skip-Gram*. We proceed to formulate the Skip-Gram architecture for an undirected symmetric network. One seeks to optimize an objective function in finding a representation $f(v)$ such that the conditional probability for obtaining individually the elements in $N(v)$, given an input node v, is maximized; that is, find f such that

$$\sum_{v \in V} \log P\big(N(v)|f(v)\big) \tag{16}$$

is maximized.

The maximization is done by training a one-layer hidden neural network, which has as possible inputs n vectors, one for each node in the network. A fixed input vector has as desired output a probability distribution on the nodes. It should be concentrated as well as possible to the neighbors (suitably defined) of the input node. The idea is to train the neural net through its hidden layer so that this is achieved to the highest possible degree. Only linear

transformations are used from the input layer to the hidden layer and essentially also from the hidden layer to the output, although a logistic-type transformation is used to transform the outputs to probabilities. A few basic facts of neural networks are given in Section 2.1 of the Supplementary Material (Tjøstheim, Jullum and Løland, 2023).

The training is done successively by going through this process for each input node several times and is stopped when the deviation from the obtained probability distribution on the outputs is close enough to the ideal desired one, which is completely concentrated on the sought neighboring nodes. At each step of this procedure, each node has an input vector representation and an output vector representation. It is the output vector representation that is of interest since it describes the relation between a node and its neighbors. This training process strives to maximize the function in (16).

To make this optimization problem tractable, the following two assumptions are made (not always made explicitly in the language processing papers):

(1) Conditional independence: The conditional likelihood is factorized as

$$p\big(N(v)|f(v)\big) = \prod_{n_i \in N(v)} P\big(n_i|f(v)\big). \tag{17}$$

(2) Symmetry in feature space and softmax: A source node and a neighborhood node have a symmetric effect on each other in the embedding feature space. Accordingly, the conditional likelihood for every source-neighborhood pair is modeled as a softmax unit, parameterized by a dot product of their features

$$P\big(n_i|f(v)\big) = \frac{\exp(f(n_i)\cdot f(v))}{\sum_{u\in V}\exp(f(u)\cdot f(v))}. \tag{18}$$

This is nothing but a suitable parametrization of the multinomial logistic regression model, but in the data science literature "softmax unit" is preferred. Formula (18) may be compared to the development in Section 2.4 in the Supplementary Material (Tjøstheim, Jullum and Løland, 2023).

With the above assumptions and taking logarithms in (18), the objective function in equation (16) simplifies to

$$\max_f \sum_{v\in V}\Bigg[-\log\Bigg(\sum_{u\in V}\exp\big(f(v)\cdot f(u)\big)\Bigg) + \sum_{n_i\in N(v)} f(n_i)\cdot f(v)\Bigg]. \tag{19}$$

In the training of the neural net, one avoids solving a high-dimensional eigenvalue problem, but there is an obvious computational issue involved. As the size of the network increases with n, the neural net with the associated input and output vectors representations becomes heavy to update. For each step of the training, in principle, all of these representations have to be updated. The updating of the node input vectors is cheap, but learning the output vectors, which are the vectors of interest, is expensive. For each training instance, one has to iterate through every node of the network (cf. the summation over u in (18) and (19)), compute the output and the prediction error and finally use the prediction error in a gradient descent algorithm to find the new output vector representation.

The idea of negative sampling, first introduced in Mikolov et al. (2013) in text analysis, makes the training process amenable by not sampling over the entire network for each update of a node, but rather a small sample of nodes. Obviously, the output nodes in the neighborhood of a given node should be included in the update sample, that is, the last sum of (19). They represent the ground truth and are termed positive samples. In addition, a small number k of nodes (noise or negative samples) should be updated. Mikolov et al. (2013) suggest that $k = 5-20$ are useful for small training sets, whereas for large training sets $k = 2-5$ may be sufficient; see Section 2.6 of the Supplementary Material for more details (Tjøstheim, Jullum and Løland, 2023). The sampling is done via a probability mechanism where each word (node) is sampled according to its frequency in the text. It will be seen below how this can be done in the network case. In addition, Mikolov et al. (2013) recommends, from empirical experience, that in the further analysis each frequency should be raised to the power of 3/4 (cf. again Section 2.6 of the Supplementary Material). This seems also to have been adopted in the network version of negative sampling. Clearly, a more thorough statistical analysis, also including the choice of k, would be of interest. We refer again to the first of the three keypoints of Section 7.

We will return to the question of negative sampling in the next subsection, where a sampling strategy S is introduced for creating neighborhoods of a node v.

5.3.2 *Neighborhood sampling strategies.* Various authors have suggested different sampling strategies of the nodes of a network. We will go through three main strategies, which seem to be representative of this field as of the last 5 years. All of these contain parameters to be chosen for which, to our knowledge, an optimality theory is lacking.

Perozzi, Al-Rfou and Skiena (2014) device a sampling strategy they call "DeepWalk". Consider a node v, and denote by w_{vu} the weight of its (undirected) edge with another node u. Let the degree variable be $d_v = \sum_u w_{vu}$. Then start a random walk from v by letting it choose the one-step neighbor u with probability $P(u|v) = w_{uv}/d_v$. Next, repeat this for the node u, and so on until L steps, say, have been obtained. The walk may return to v for one or more of its steps. This procedure is now repeated γ times obtaining γ random walks starting in v. These may

be compared to text segments in natural language processing. Analog to a moving window in a language text we now let a window of size $2K+1$, where $2K+1 \leq L$, glide along the random walk paths. For each window, there is a center node numbered u', $K \leq u' \leq L-K$ and we define a neighborhood $N_S(u')$ and K nodes prior to u' and K nodes after u' in the considered random walk path. For each such configuration, we apply the Skip-Gram procedure (16)–(19). In this way, for each node v we generate $\gamma \times (L-2K)$ segments of nodes. Note that this creation of segments in paths of random walks can be carried out before the optimization process takes place. When applied to all of the nodes of the network it results in a collection of $n \times \gamma \times (L-2K)$ segments of nodes that correspond to windows of words in a language text. This sets up a frequency distribution over the nodes corresponding to the frequency distribution of words in the vocabulary in a text. Negative sampling of nodes can then be applied to this frequency distribution of nodes.

The LINE (Large-scale Information Network Embedding) was introduced by Tang et al. (2015). They use a slightly different optimization criterion than (17). Somewhat similar to Grover and Leskovec (2016), LINE introduces the concepts of first- and second-order proximities.

Qiu et al. (2018) and Qiu et al. (2019) obtain a unifying view of the DeepWalk and LINE among other algorithms. Recent activity in deep learning and recursive neural networks should also be mentioned (Young et al., 2018). Software packages are available for all of the algorithms mentioned in this section, and a number of real data examples are given in the publications cited.

5.4 Directed Network

In many applications of networks, one deals with a directed network, for example, in causality networks. This is a network where the weight on edges between nodes v_i and v_j may be different, so that $w_{ij} \neq w_{ji}$, and one may even have $w_{ij} > 0$ but $w_{ji} = 0$. Rohe, Qin and Yu (2016) have looked at this from a spectral graph point of view. Directed graphs have also been attempted incorporated in the Skip-Gram procedure; see, for example, Zhou et al. (2017, page 2944). The undirected sampling strategy described in Section 5.3.2 can again essentially be used. To illustrate, let w_{ij} be the weight of the edge in a transition from v_i to v_j. In a money laundering investigation, for example, where the nodes may be bank accounts, w_{ij} may be proportional to the number of transactions from account i to account j. Similarly, one may define w_{ji}. The probability of going from node v_i to v_j can then be given as $p_{ij} = w_{ij}/d_i$, where $d_i = \sum_{j \in N_S(i)} w_{ij}$ and $N_S(i)$ is the first-order neighborhood of v_i.

5.5 Heterogeneous Network Representation

Heterogeneous here refers to a situation where there are different types of nodes in a network, and there may be different types of edges. If these are treated with homogeneous techniques neglecting the heterogeneity, inferior results may result.

Two papers will be briefly mentioned, one is an extension of the LINE approach, the other is an extension of the DeepWalk methodology. In these two papers, the Skip-Gram algorithm is applied on so-called metapaths, paths consisting of a sequence of relations defined between *different* node types. The introduction of metapaths to heterogeneous graphs came before the Skip-Gram procedure; see Sun et al. (2012).

It is natural also to mention the extension of LINE found in the PTE (Predictive Text Embedding) of Tang, Qu and Mei (2015). PTE deals with a text network embedding, but the method is applicable to a general network.

Dong, Chawla and Swami (2017) introduce a form of random walk sampling for heterogeneous networks, which is analogous to or extends the sampling procedures in Perozzi, Al-Rfou and Skiena (2014) and Grover and Leskovec (2016). Skip-Gram is combined with the metapath sampling as discussed by Sun et al. (2012).

5.6 Embedding of Dynamic Networks

Most of the work on embedding of networks has been done on static networks. There is no time dimension involved to trace the dynamic evolution of the network. In many situations, this is of course not very realistic. Consider, for example, a bank network. New accounts are opened, other accounts are closed. New types of transactions between accounts are appearing, others are becoming old and less relevant. Or in more general network language, new nodes are coming into the network, others are removed. New edges are created, others are discarded. Weights between edges may easily change in time. In a heterogeneous network, new types of nodes may enter the system, others may leave. An early empirical investigation of changes in social networks is contained in Kossinets and Watts (2006); see also Greene and Cunningham (2011).

An obvious brute force solution is to use a moving window and then do an embedding, and possible clustering in each window. But clearly such a procedure is time consuming and nonefficient if there are many (overlapping) windows. One would like to have an updating algorithm that can keep information in the previous window and combine it with new information in the new window. To our knowledge, the literature here is quite limited.

There is an attempt to generalize the entire Skip-Gram methodology to a dynamic framework. This can be seen in Du et al. (2018). They utilize that a network may not change much during a short time in dynamic situations, thus the embedding spaces should not change too much either. A related paper venturing into heterogeneous networks meta paths is Bian et al. (2019). Zhu et al. (2017)

takes a more statistical modeling point of view on dynamic networks. The paper is briefly reviewed in the next subsection. Clearly, the theme of dynamic networks is an open and challenging field for data scientists and statisticians. Much late work is summed up, mostly from a machine learning point of view in Kazemi et al. (2020). Some recent trends in embedding of time series and dynamic networks are reviewed in Tjøstheim, Jullum and Løland (2023), with a number of additional recent references.

5.7 Network Embedding: Data Science and Machine Learning Versus Statistical Modeling

An overwhelming part of the literature on network embedding can be found in the machine learning journals and in proceedings on data and computational science. The emphasis has been on deriving methods that "work", that is, can be used in practical applications. Certain parts of some of the methods used are quite ad hoc such as the argument in Mikolov et al. (2013) where from empirical evidence the word count is raised to 3/4 power in the distribution forming the basis of the negative sampling. This has been followed up in later literature and does seem to work well. But it is not clear why. Moreover, there are few quantitative expressions of uncertainty or on statistical properties of the obtained results.

Many of the algorithms and methods discussed in this paper contain input parameters or hyperparameters, including the choice of the dimension of the embedding space. An important issue in both theory and practice is the setting of these parameters. The problem has to be treated with care to avoid instability in the embedded structure. The problem is briefly mentioned in Section 4.3, but the problem is relevant also in a more general context.

Broadly speaking, statistical methods use theoretically derived methods to choose hyperparameters necessary to fully specify a method, while the typical machine learning approach is to rely on hyperparameter optimization or so-called tuning. The former may require assumptions that are too strong or cannot be checked in practice. The latter typically requires additional data or retraining of models based on randomly dividing the data into subsets (cross-validation), which is computationally costly and comes with an uncertainty component due to the randomness in the data splitting. Many machine learning practitioners may enforce a rather basic and ad hoc trial and error optimization approach. Still, methods like Bayesian optimization (Shahriari et al., 2015) have gained significant momentum in the recent years. Bayesian optimization aims at solving the optimization problem using as few evaluations as possible. While the method uses statistical theory through its reliance on Gaussian processes, the hyperparameter selection problem is still based on optimization and possesses the aforementioned drawbacks. We think the machine learning methods could benefit from theoretically derived hyperparameter choices. There have been some attempts at choosing parameters for machine learning methods through the statistical information criterion approach (Claeskens, Croux and Van Kerckhoven, 2008, Lunde, Kleppe and Skaug, 2020), but it does not yet seem to have found its place in machine learning. The theoretical difficulty of deriving such criteria due to the lack of proper likelihoods in the training of the machine learning methods is an obvious obstacle. To avoid this, it might be possible to go in the direction of the generalized information criterion (GIC) (Konishi and Kitagawa, 2008), which does not require a likelihood, but rather relies on functionals of the data generating distribution and their associated influence functions. In any case, going forward, we believe it is worth looking in the direction of theoretically derived selection procedures for the machine learning community, and have as such identified this in our list of keypoints in Section 7.

5.7.1 *Stochastic block modeling*. The issues mentioned above appear to lead to a gap between data/computational science using algorithmic approaches and more traditional (and modern) statistical thinking. There is a clear need for results bridging this gap, as argued in the second keypoint in Section 7. In this subsection, we focus on stochastic block modeling, but it should be realized that other types of statistical models have been proposed; see Crane and Dempsey (2015).

In particular, Peter Bickel and his collaborators have taken up various problems of asymptotic theory for stochastic block models and related models. This includes hypothesis testing in Bickel and Sarkar (2016), asymptotic normality in Bickel et al. (2013), nonparametrics in Bickel and Chen (2009). Works more specifically directed towards asymptotics of spectral clustering can be found in Rohe, Chatterjee and Yu (2011) and in Lei and Rinaldo (2015). Most of these works require a delicate asymptotic balancing between the number of nodes, the degree of the nodes and the number of communities. An example of a heterogeneous model, which is analyzed rigorously from a statistical point of view, is Zhang and Chen (2020). For instance, the proposed modularity function is shown to be consistent in a heterogeneous stochastic block model framework. It is related to the Bickel and Chen (2009) paper. See also Decelle et al. (2011) who bring in algorithmic applications of block models using cavity methods to describe phase transitions in inference and learning.

A very important problem both in practice and in theory is the problem of determining the number of communities in community detection. In earlier literature, this number was actually taken to be known. In statistical likelihood-based models, one has attempted to find this number by letting it be an unknown parameter in the likelihood and then do likelihood integration. Wang and Bickel (2017) look at the problem from an underestimation and overestimation point of view. Newman and Reinert (2016) propose replacing the original Bernoulli-type likelihood by

an approximated Poisson likelihood, which is easier to handle computationally. Peixito (2021) discusses AIC and BIC-type approaches to this problem.

There has been made progress in the numerical estimation of the parameters in stochastic block-type models. Typically, a Bayesian approach has been used with extensive use of Markov Chain Monte Carlo; see, for example, Peixoto (2019). But we think it is fair to say that the dimension of the networks attacked by stochastic block modeling has been considerably less than the most general used algorithmic Skip-Gram models of Section 5.3.1.

5.7.2 *Dynamic graphs and time-series modeling in networks*. The discussion of an algorithmic approach versus stochastic modeling is also taken up in Tjøstheim, Jullum and Løland (2023), treating recent trends in embedding of time series and dynamic networks. Examples of an algorithmic, but with some statistical modeling aspects, are given in Lim and Zohren (2021) and Salinas et al. (2020), both involving deep learning networks and time series. There are also several papers with rigorous asymptotic analysis of networks and autoregressive models (Zhu et al., 2017, Zhu and Pan, 2020). Given such a framework, conditions for stationarity are obtained, and least squares estimates of parameters are derived and their asymptotic distribution found.

There are a number of differences between the network vector autoregression modeled in these publications and the dynamic network embeddings mentioned in Section 5.6. First of all, Zhu et al. (2017) treat the dynamics of the nodes themselves and not of an embedding. Even if the autoregressive model does introduce some (stationary) dynamics in time, the parameters are static; that is, no new nodes are allowed, and the relationship between them is also static as modeled by a matrix $\mathbf{A} = \{a_{ij}\}$. From this point of view, as the authors are fully aware of, the model is not realistic for the dynamics that takes place in practice for many networks. On the other hand, the introduction of a stochastic model that can be analyzed by traditional methods of inference is to be lauded. A worthwhile next step is to try to combine more realistic models with a stochastic structure, possibly regime-type models for the parameters, that is amenable to statistical inference. An attempt in this direction is made in Ludkin, Eckley and Neal (2018) in the context of stochastic block models.

For some very recent contributions to network autoregression, see Armillotta, Fokianos and Krikidis (2022) and references therein.

6. EMBEDDING IN 2 OR 3 DIMENSIONS AND VISUALIZATION

Visualization is an important part of data analysis. The problem can be stated as finding a good 2- or 3-dimensional representation of high-dimensional data and often with a large number of samples. Principal component analysis offers one possibility where the data are projected on the 2 or 3 first principal components. Although very useful, since it is linear and projects on a hyperplane, it generally fails to give a good characterization in cases where the data are concentrated on a nonlinear manifold which is a subset of $\mathbb{R}^p$.

It is appropriate to conclude this survey on embedding by the topic of visualization, where in principle any of the treated methods in this survey can be used by choosing the embedding dimension m to be 2. However, we have chosen to concentrate on three methods that are powerful and much used, and which are based on the main ideas in Sections 3, 4 and 5, respectively. The t-SNE algorithm was developed by van der Maaten and Hinton (2008) and van der Maaten (2014). It is based on ideas handling the connection between a high-dimensional x-scale and a low-dimensional y-scale, which are inherent already in multidimensional scaling. But unlike most earlier attempts, t-SNE is based on comparisons of *probability distributions* on the x and y-scale, which seems much more sensible in a nonlinear problem than applying moments and covariances.

Tang et al. (2016) introduced LargeVis, which is based on techniques reviewed in Section 5, especially the Skip-Gram procedure treated in Section 5.3. Finally, McInnes, Healy and Melville (2018) use methods from topological data analysis akin to ideas in Section 4 to derive their algorithm UMAP. Illustrations of the use of the three methods are given in Section 6.5.

6.1 t-SNE

SNE is an acronym for Stochastic Neighbor Embedding. That embedding and visualization technique was introduced by Hinton and Roweis (2002). The t in t-SNE refers to further developments in van der Maaten and Hinton (2008) using a t-distribution approximation on the y-scale.

Starting with SNE, the similarities between the points on the x-scale and y-scale are sought expressed in terms of pairwise Gaussian approximations. On the x-scale, high- dimensional Euclidean distances are expressed in conditional probabilities. The similarity of a data point X_i to a data point X_j is expressed as a Gaussian conditional probability $p_{j|i}$ such that for pairs of nearby data points, $p_{j|i}$ would be relatively high, whereas for widely separated points, $p_{j|i}$ could be infinitesimally small. The essential idea is to preserve the internal structure of the high-dimensional data by keeping similar data points close and dissimilar data points far apart, in the low-dimensional space. Mathematically, $p_{j|i}$ is given by

$$p_{j|i} = p_{j|i}(x_j|x_i) = \frac{\exp(-\|x_j - x_i\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_k - x_i\|^2/2\sigma_i^2)}, \tag{20}$$

where σ_i^2 is the variance of the Gaussian that is centered on the data point x_i. The parameter σ_i is chosen so that the probability distribution P_i, induced by $p_{j|i}$ for all j-s different from i, has a perplexity specified by the user. Here, the perplexity of P_i is given by

$$\text{Perp}_i = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}.$$

See Hinton and Roweis (2002) for more details.

The similarities on the x-scale is sought mapped into corresponding similarities in the low-dimensional y-scale by modeling the conditional probabilities by

$$q_{j|i} = \frac{\exp(-\|y_j - y_i\|^2)}{\sum_{k \neq i} \exp(-\|y_k - y_i\|^2)}.$$

The coordinates Y_i of a data point $X_i, i = 1, \ldots, n$ are then sought determined by minimizing the Kullback–Leibler distance (or cross entropy) between the $p_{j|i}$ and $q_{j|i}$, that is, by minimizing the cost function

$$C = \sum_i \text{KL}(P_i \parallel Q_i) = \sum_{i,j} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

The minimization of the cost function with respect to the y-coordinates can be done by using a gradient descent method, and the y-s are initialized by random, Gaussian values.

The SNE algorithm is hampered by a cost function, which is quite difficult to optimize in practice, and there is a so-called "crowding" problem in the sense that far apart points on the x-scale may be mapped in such a way that the joint probability q_{ij} may be even smaller than p_{ij}. These problems are attacked in t-SNE by symmetrization, modeling *joint* probabilities p_{ij} and q_{ij} and by using a t-distribution as an approximation at the y-scale having points in the tails mapped such that q_{ij} is larger than p_{ij} to avoid the crowding effect. This trick is also present for other local techniques for multidimensional scaling.

To avoid problems that may be caused by outliers on the x-scale, the "joint probabilities" on the x-scale are in fact computed as $p_{ij} = (p_{i|j} + p_{j|i})/2n$, which ensures $\sum_j p_{ij} > 1/2n$ for all data points X_i, such that each data point makes a significant contribution to the cost function. Further, on the y-scale a t-distribution structure of one degree of freedom is used,

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq \ell}(1 + \|y_k - y_\ell\|^2)^{-1}},$$

where it should be noted that a double sum is now used in the denominator. The cost function is given by

$$C = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

The details of the optimization can again be found in van der Maaten and Hinton (2008). In that paper, there is also a series of experiments comparing t-SNE with the Sammon mapping of MDS and the ISOMAP and LLE, where the t-SNE does extremely well.

The t-SNE algorithm is speeded up in the paper by van der Maaten (2014) by not going over all possible pairs (x_i, x_j) but only essentially over nearest neighbors.

6.2 LargeVis

Tang et al. (2016) propose a new algorithm for visualization, LargeVis. It starts with a speeded up *approximate* nearest neighbor algorithm that has complexity $O(n)$ as compared to $O(n \log n)$ for the speeded up nearest neighbor algorithms of van der Maaten (2014). The Tang et al. (2016) algorithm is built upon random projection trees but significantly improved by using neighbor exploring. The basic idea of this, similar to the LINE construct in Tang et al. (2015) and referenced in Section 5.3.2, is that "the neighbor of my neighbor is also likely to be my neighbor". Specifically, a few random projection trees are built to construct an approximate k-nearest neighbor graph, the accuracy of which may not be so high. Then for each node of the graph, the neighbors of its neighbor are searched, which are also likely to be candidates of its nearest neighbor. The accuracy may then be improved by multiple iterations. The claim is that the accuracy of this k-nearest neighbor graph quickly improves to almost 100% without investing in many trees. For the weights of the nearest neighbor graph, essentially the same procedure as in t-SNE is used. The graph is symmetrized by setting the weights between x_i and x_j to $w_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$, where $p_{i|j}$ and $p_{j|i}$ are defined via (20). Before using the LargeVis algorithm itself a preprocessing step can be used where the dimension is reduced to say 100 by using the Skip-Gram network embedding technique explained in Section 5.3. The negative sampling technique of Mikolov et al. (2013) is used in the Skip-Gram step.

For the time complexity of the optimization, done with asynchronous stochastic gradient descent, each stochastic gradient step takes $O(sM)$, where M is the number of negative samples, say M is from $5 - 10$, and s is the number of dimensions of the low-dimensional space, $s = 2, 3$. Therefore, the overall complexity is $O(sMn)$, which is linear in the number of nodes.

6.3 UMAP

Sections 4.1 and 4.2 were concerned with topological methods in manifold learning and persistence homology. In particular, filters of simplicial complexes were used in Section 1.2 of the Supplementary Material (Tjøstheim, Jullum and Løland, 2023). In the first part of McInnes, Healy and Melville (2018), these filters are generalized to simplicial sets. In addition, components of fuzzy set theory, category theory and functor theory are used to compute fuzzy topological representations.

Letting $\{Y_1, \ldots, Y_n\} \subseteq \mathbb{R}^m$ and $\{X_1, \ldots, X_n\} \subseteq \mathbb{R}^p$ with $m \ll p$, in visualization we have a situation where m is 2 or 3.

To compare two fuzzy sets generated by $\{X_1, \ldots, X_n\}$ and $\{Y_1, \ldots, Y_n\}$, respectively, fuzzy set cross entropy is used in UMAP. The use of advanced concepts of algebraic topology makes the first part of this paper hard to read. In the computational part of the paper, however, inspired by motivations and ideas of the first part, the authors specialize to a k-neighborhood graph situation where the analogy with t-SNE and LargeVis is easier to appreciate.

As with other k-neighbor graph based algorithms, UMAP, can be described in two phases. In the first phase, a particular weighted k-neighbor graph is constructed. In the second phase, a low-dimensional layout of this graph is made. The theoretical basis for UMAP in the first part of McInnes, Healy and Melville (2018) provides novel approaches to both of these phases.

Let $\{X_1, \ldots, X_n\}$ be the input data set with a jointly given matrix $\mathbf{D}$ that can be thought of as consisting of Euclidean distances between the data vectors. For each X_i, one can compute the set of k nearest neighbors $\{X_{i_1}, \ldots, X_{i_k}\}$. There are many choices of a nearest neighbor algorithm. McInnes, Healy and Melville (2018) use the algorithm of Dong, Moses and Li (2018).

This can be used to define a weighted directed graph $G' = (V, E, w)$. The nodes of G' are the set $\{X_1, \ldots, X_n\}$ the directed edges are $\{(X_i, X_{i_j}) | 1 \leq j \leq k, 1 \leq i \leq n\}$ and a weight function defined in McInnes, Healy and Melville (2018). Let $\mathbf{A}$ be the weighted adjacency matrix of G'. An undirected graph G is obtained by introducing the symmetric adjacency matrix

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^T - \mathbf{A} \circ \mathbf{A}^T,$$

where $\circ$ denotes the Hadamard (pointwise) product.

The $\{X_1, \ldots, X_n\}$ data set is next connected to a low-dimensional data set $\{Y_1, \ldots, Y_n\}$, where the dimension is 2 or 3 if visualization is considered. The transition from $\{X_1, \ldots, X_n\}$ to $\{Y_1, \ldots, Y_n\}$ is accomplished by a force directed graph layout algorithm. The history of this kind of graph layout goes far back, Tutte (1963). A more recent account can be found in Kobourov (2012). The details of the algorithm as used in UMAP with an iterative application of attractive and repulsive forces are given in McInnes, Healy and Melville (2018, page 14). It should be noted that the terminology of attractive and repulsive forces is used in van der Maaten and Hinton (2008) as well, but unlike their paper where there is a random set-like initialization, in UMAP a spectral layout (cf. Sections 3.5 and 5.2) is used to initialize the embedding. This is claimed to provide faster convergence and greater stability within the algorithm. Note that negative sampling, as treated in Section 5.3, is also important to reduce the computational burden.

6.4 A Brief Comparison of t-SNE, LargeVis and UMAP

A number of experiments were performed in McInnes, Healy and Melville (2018) with a comparison to t-SNE and LargeVis. The UMAP works on par with or better than these algorithms for those examples.

All of the embedding algorithms have been demonstrated to work well in a number of quite complicated situations. Nevertheless, as pointed out by McInnes, Healy and Melville, it is important to be aware of some weaknesses of these algorithms that could create fruitful challenges for further research.

t-SNE, LargeVis and UMAP all lack the strong interpretability of PCA and it is difficult to see that something like a factor analysis can be performed.

One of the core assumptions is that it is assumed that there exists a lower-dimensional manifold structure in the data. If this is not so, there is always the danger that a spurious noise driven embedding can be the result. This danger is reduced as the sample size increases. Developing an asymptotic analysis and finding more robust algorithms is clearly a challenge.

For all three algorithms a number of approximations are made, such as the use of approximate nearest neighbor algorithms and negative sampling used in optimization. Particularly for small sample sets the effect of these approximations may be nonnegligible.

6.5 An Illustrating Example

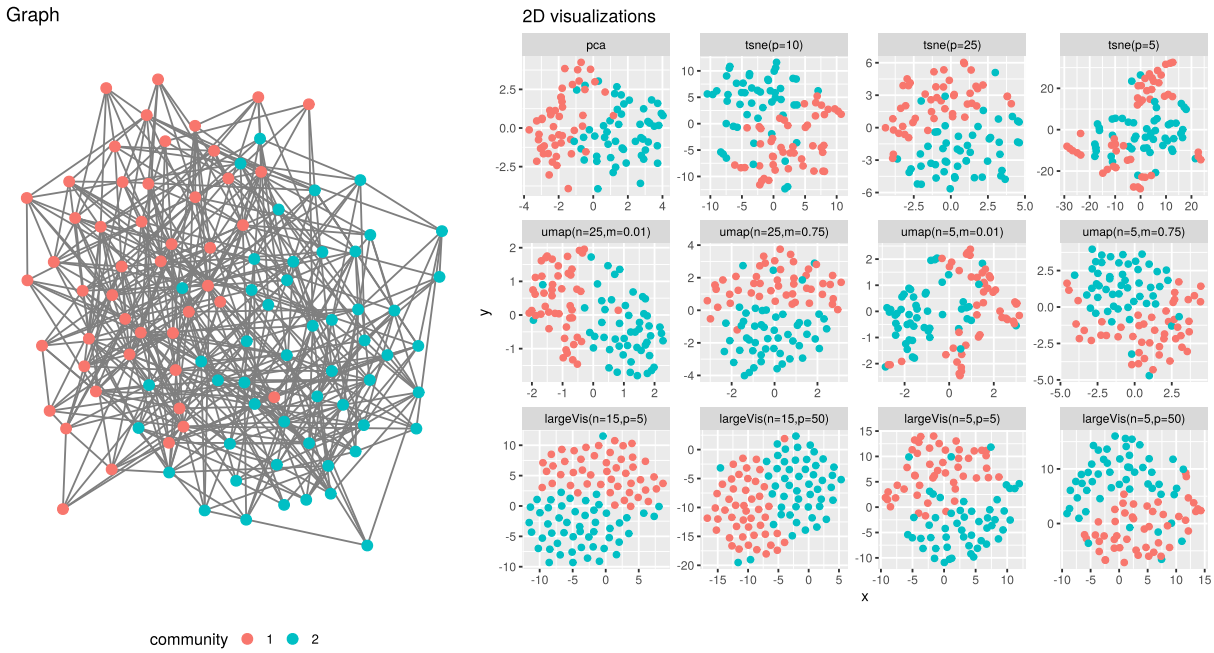
The illustrating example consists of two networks, each having two different types of nodes (colored red and blue, respectively) corresponding to two different communities. The first one, the homogeneous graph in Figure 5(a), is very simple and is simulated from a stochastic block model (Karrer and Newman, 2011), mentioned in Section 5.2.4, with 2 communities, 100 nodes, average node degree $d = 10$, and ratio of between-community edges over within-community edges $\beta = 0.4$. In this setup, the number of edges per node is Poisson distributed with expected number of edges of 10. This simple network has very little overlap between the two types of nodes.

The second one is somewhat more complex, the heterogeneous graph in Figure 5(b), and is simulated from three subgraphs **a**, **b** and **c**, that has 2 communities each:

Graph a: 30 nodes, average node degree $d = 7$, ratio of between-block edges over within-block edges $\beta = 0.2$

Graph b: 30 nodes, average node degree $d = 15$, ratio of between-block edges over within-block edges $\beta = 0.4$

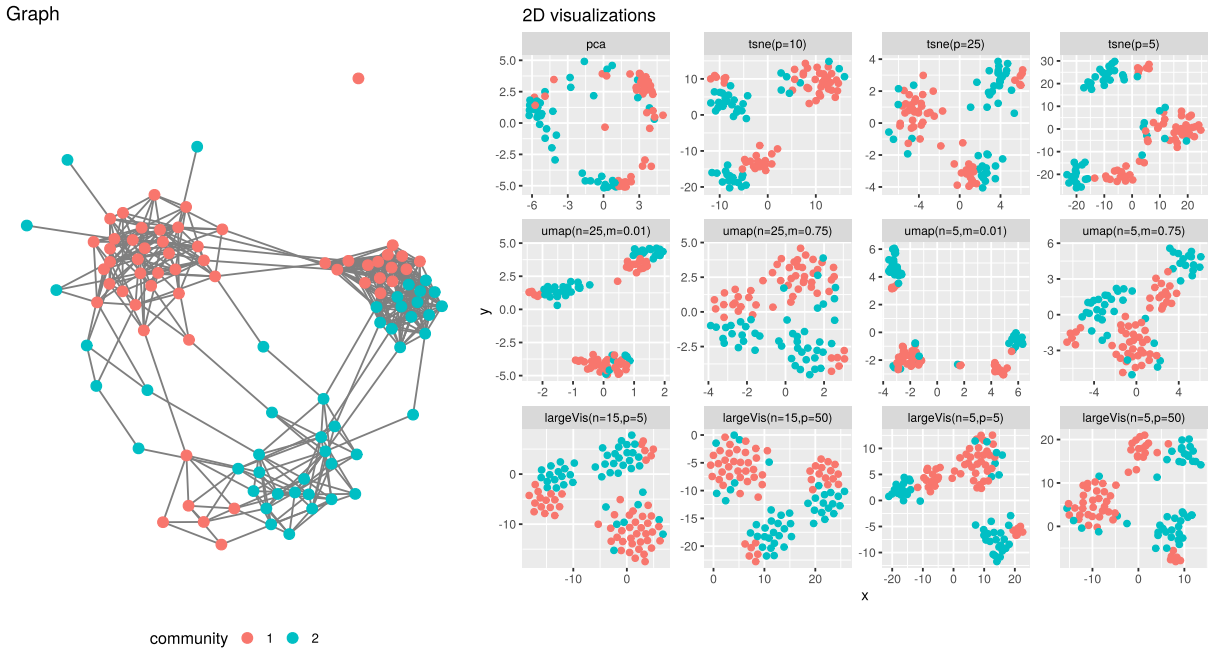
Graph c: 40 nodes, average node degree $d = 7$, ratio of between-block edges over within-block edges $\beta = 0.2$, and an unbalanced community proportion; a probability of 3/4 for community 1 and a probability of 1/4 for community 2



Classification scores

	org_embedding	pca	tsne(p=10)	tsne(p=25)	tsne(p=5)	umap(n=25,m=0.01)	umap(n=25,m=0.75)	umap(n=5,m=0.01)	umap(n=5,m=0.75)	largeVis(n=15,p=5)	largeVis(n=15,p=50)	largeVis(n=5,p=5)	largeVis(n=5,p=50)
prop community correct	0.814	0.864	0.776	0.852	0.772	0.88	0.844	0.808	0.82	0.902	0.872	0.846	0.818
prop majority vote correct	0.94	0.91	0.82	0.91	0.84	0.94	0.93	0.85	0.89	0.94	0.91	0.88	0.9

(a) Homogeneous graph from the stochastic block model.



Classification scores

	org_embedding	pca	tsne(p=10)	tsne(p=25)	tsne(p=5)	umap(n=25,m=0.01)	umap(n=25,m=0.75)	umap(n=5,m=0.01)	umap(n=5,m=0.75)	largeVis(n=15,p=5)	largeVis(n=15,p=50)	largeVis(n=5,p=5)	largeVis(n=5,p=50)
prop community correct	0.878	0.772	0.892	0.89	0.868	0.872	0.852	0.864	0.87	0.826	0.81	0.882	0.866
prop majority vote correct	0.92	0.8	0.91	0.91	0.88	0.92	0.94	0.88	0.93	0.9	0.85	0.9	0.9

(b) Heterogeneous graph from a combination of three stochastic block models.

FIG. 5. *Graphs, visualizations and classification results with a k-nearest neighbors algorithm with* $k = 5$.

To link graphs a, b and c, some random edges are added between nodes from the same community.[1]

The purpose of the illustrating example is to examine how well these network structures are managed by t-SNE, LargeVis and UMAP, how robust they are to parameter choices inherent in the three methods, and how they compare with traditional principal component analysis (PCA) visualization.

The visualization is done in two steps. First the networks are embedded in $\mathbb{R}^m$ with $m = 64$ using the Skip-Gram routine node2vec with (cf. Section 5.3.2) $L = 30$ nodes in each random walk and $\gamma = 200$ walks per node, and a word2vec window length of $K = 5$ where all nodes are included. The second step is to reduce the point cloud in $\mathbb{R}^{64}$ to $\mathbb{R}^2$, that is, the visualization step using PCA and the three visualization algorithms with a selection of different tuning parameters. (In t-SNE, p is the perplexity parameter; in LargeVis, n is the number of negative samples, p the total weight of positive interactions; in UMAP n is the number of nearest-neighbors, m is a distance parameter, where low m gives clumpier embeddings.) The results are given in Figures 5(a) and 5(b).

Underneath the figures are given classification scores for the two types of nodes (communities) in the study. These are classified on a neighborhood basis. In the first line of each subtable the class of a node is determined using the average of the 5 nearest neighbors; in the second by the majority vote among these 5 nearest neighbors. The first column "org_embedding" gives the classification results for the 64-dimensional embedding in step 1.

For the simple network, PCA does well, on par with the three other visualization algorithms, both visually and in the classification. The tuning parameters does not seem to make much of a difference with the exception of t-SNE with $p = 5$. For the more complicated network, PCA is in trouble both visually and with respect to classification. In this case, the dependence on tuning parameters seems to be greater, but most of the visualizations manage to pick out the three subgraphs **a**, **b** and **c**. For all values of the tuning parameters t-SNE, LargeVis and UMAP, all do clearly better than PCA. Somewhat surprisingly, perhaps, the embedding in 64 dimensions gives result not very different from those of the three visualizations routines. We also did experiments with other embedding dimensions ranging from 2 to 256. Again the classification results were not much different. This could be due to the fact that the number of nodes and links in these experiments are very modest compared to the real data experiments in the Skip-Gram references given in Sections 5.3.1 and 5.3.2, which has number of nodes and links of an entirely different order. A more involved illustrating example (but still with a moderate number of nodes) is given in Section 3 of the Supplementary Material (Tjøstheim, Jullum and Løland, 2023).

[1]For each pair of nodes between a pair of graphs, say Graph **a** and **c**, a new link is randomly sampled with a probability of 0.01, and links connecting two nodes from the same community are kept.

7. SOME CONCLUDING REMARKS

Principal components work well for linearly generated Gaussian data. It may also work well for other types of data and is probably still the most important statistical embedding method. But, on the other hand, it is not difficult to find examples where it does not work. The search for nonlinear extensions started long ago with the MDS method. In fact, multidimensional scaling methods contain ideas that have been found relevant in several recent nonlinear algorithms.

There is no universally superior method that works better than any of the others in all situations. For Gaussian or approximately Gaussian data ordinary principal components should be preferred. If the distribution can be approximated locally by a Gaussian, the potential of locally Gaussian methods as outlined in Tjøstheim, Otneim and Støve (2022b) could be investigated. Other nonlinear methods depend on local linear structures in the data. For data sets with holes or cavities, topological data analysis is a natural option. Data that form a network has artificial neural network methods as an obvious candidate. The Skip-Gram method of Section 5.3.1 is based on a single layer artificial network. Deep learning algorithms are based on multiple layer neural networks and is an attractive alternative for more complicated dependencies. The neural network approaches have an advantage in their speed, making it possible to treat ultra high-dimensional data sets with complex relationships.

In this paper, we have covered selected methods of nonlinear embedding generalizing PCA, topological embeddings in persistence diagrams, network embedding and embedding to dimension 2 (i.e., visualization). In addition, in the course of the review, we have pointed to some cases of an apparent and arguably widening gap between developments in data science, including computer and algorithmic-based methods, and more traditional statistical modeling methods. We have also sought to point out specific issues that could benefit from more input from statisticians. These may be conveniently summed up in the following keypoints:

1. In quite a few algorithms, there are parameters to be chosen, and the performance of the algorithm may depend quite strongly on these choices. Examples can be found in Skip-Gram, spectral community detection, the Mapper, and there are others. There is a need for well-founded methods for making in some sense optimal or near optimal choices of such parameters—in some cases

as an alternative to the computational expensive empirical optimization routines, which typically also have a randomness component. As mentioned in Section 5.7, information criterion based solution is one option, in particular likelihood-free methods like GIC might be one way to go about this.

2. It is highly desirable to reduce the gap between machine learning algorithmic techniques and statistical modeling. A good example of a bridging attempt is the stochastic block models for which one can do statistical inference and which has also resulted in decent network algorithms. One needs more of this!

3. More critical statistical work is needed to test the sanity and robustness of algorithms. One example is the close investigation of the modularity algorithm reported on in Section 5.2.4. It is useful to put algorithms to stress tests, but it is important to find a balancing point between such criticism and perceived usefulness of an algorithm.

It is crucial, however, to point out that this is a two-way relationship. We are hopeful that interaction between machine learning and statistical modeling could bring about synergy effects for both disciplines.

ACKNOWLEDGMENTS

The authors would like to thank two anonymous referees, an Associate Editor and in particular the editor for their constructive and very helpful comments that improved the quality of this paper.

FUNDING

This work was supported by the Norwegian Research Council Grant 237718 (BigInsight).

SUPPLEMENTARY MATERIAL

Supplement to "Statistical Embedding: Beyond Principal Components" (DOI: 10.1214/22-STS881SUPP; .pdf). The Supplement (Tjøstheim, Jullum and Løland, 2023) contains more details on persistence diagrams, simplical complexes and word embedding, as well as a more involved variant of the network example in Section 6.5.

REFERENCES

AIZERMAN, M. A., BRAVERMAN, E. M. and ROZONOER, L. I. (1956). Theoretical foundations of the potential function method in pattern recognition learning. *Autom. Remote Control* **25** 821–137.

ARMILLOTTA, M., FOKIANOS, K. and KRIKIDIS, I. (2022). Generalized linear models network autoregression. In *Network Science* 112–125. International Conference on Network Science.

BAGLAMA, J. and REICHEL, L. (2005). Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM J. Sci. Comput.* **27** 19–42. MR2201173 https://doi.org/10.1137/04060593X

BELKIN, M. and NIYOGI, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Information Processing Systems* (T. K. Leen, T. G. Dietterich and V. Treps, eds.). MIT Press, Cambridge, MA.

BELKIN, M. and NIYOGI, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **15** 1373–1396.

BIAN, R., KOH, Y. S., DOBBIE, G. and DIVOLI, A. (2019). Network embedding and change modeling in dynamic heterogeneous networks. In *Proceedings of the* 42*nd International ACM SIGIR Conference on Research and Development in Information Retrieval* 861–864.

BICKEL, P. and CHEN, A. (2009). A nonparametric view of network models and Newman–Girvan and other modularities. *Proc. Natl. Acad. Sci.* **106** 21068–21073.

BICKEL, P. J. and SARKAR, P. (2016). Hypothesis testing for automated community detection in networks. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **78** 253–273. MR3453655 https://doi.org/10.1111/rssb.12117

BICKEL, P., CHOI, D., CHANG, X. and ZHANG, H. (2013). Asymptotic normality of maximum likelihood and its variational approximation for stochastic blockmodels. *Ann. Statist.* **41** 1922–1943. MR3127853 https://doi.org/10.1214/13-AOS1124

BLONDEL, V. D., GUILLAUME, J.-L., LAMBIOTTE, R. and LEFEBVRE, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008** P10008.

BOSER, B. E., GUYON, I. M. and VAPNIK, V. N. (1992). A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on COLT*, ACM, Pittsburgh, PA.

BUKKURI, A., ANDOR, N. and DARCY, I. K. (2021). Applications of topological data analysis on oncology. *Front. Artif. Intell. Mach. Learn. Artif. Intell.* **4** 1–14.

CANNINGS, T. I. and SAMWORTH, R. J. (2017). Random-projection ensemble classification. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **79** 959–1035. MR3689307 https://doi.org/10.1111/rssb.12228

CARLSSON, G. (2009). Topology and data. *Bull. Amer. Math. Soc. (N.S.)* **46** 255–308. MR2476414 https://doi.org/10.1090/S0273-0979-09-01249-X

CARRIÈRE, M., MICHEL, B. and OUDOT, S. (2018). Statistical analysis and parameter selection for Mapper. *J. Mach. Learn. Res.* **19** Paper No. 12, 39 pp. MR3862419

CARRIÈRE, M. and RABADÁN, R. (2020). Topological data analysis of single-cell Hi-C contact maps. In *Topological Data Analysis—The Abel Symposium* 2018. *Abel Symp.* **15** 147–162. Springer, Cham. MR4338672 https://doi.org/10.1007/978-3-030-43408-3_6

CHAZAL, F. and MICHEL, B. (2017). An introduction to topological data analysis: Fundamental and practical aspects for data scientists. Preprint. Available at arXiv:1710.04019v1.

CHAZAL, F. and MICHEL, B. (2021). An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Front. Artif. Intell. Mach. Learn. Artif. Intell.* **4** 1–28.

CHEN, Y.-C., GENOVESE, C. R. and WASSERMAN, L. (2015). Asymptotic theory for density ridges. *Ann. Statist.* **43** 1896–1928. MR3375871 https://doi.org/10.1214/15-AOS1329

CHEN, Y. C., HO, S., FREEMEN, P. E., GENOVESE, C. R. and WASSERMAN, L. (2015a). Cosmic web reconstruction through density ridges: Methods and algorithm. *Mon. Not. R. Astron. Soc.* **454** 1140–1156.

CHEN, Y. C., HO, S., TENNETI, A., MANDELBAUM, R., CROFT, R., DIMATTEO, T., FREEMAN, P. E., GENOVESE, C. R. and WASSERMAN, L. (2015b). Investigating galaxy-filament alignments in hydrodynamic simulations using density ridges. *Mon. Not. R. Astron. Soc.* **454** 3341–3350.

CLAESKENS, G., CROUX, C. and VAN KERCKHOVEN, J. (2008). An information criterion for variable selection in support vector machines. *J. Mach. Learn. Res.* **9** 541–558. MR2417246 https://doi.org/10.2139/ssrn.1094652

COIFMAN, R. R. and LAFON, S. (2006). Diffusion maps. *Appl. Comput. Harmon. Anal.* **21** 5–30. MR2238665 https://doi.org/10.1016/j.acha.2006.04.006

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. and STEIN, C. (2022). *Introduction to Algorithms*, 3rd ed. MIT Press, Cambridge, MA. MR2572804

CRANE, H. and DEMPSEY, W. (2015). A framework for statistical network modeling. Preprint. Available at arXiv:1509.08185.

CRAWFORD, L., MONOD, A., CHEN, A. X., MUKHERJEE, S. and RABADÁN, R. (2020). Predicting clinical outcomes in glioblastoma: An application of topological and functional data analysis. *J. Amer. Statist. Assoc.* **115** 1139–1150. MR4143455 https://doi.org/10.1080/01621459.2019.1671198

CUI, P., WANG, X., PEI, J. and ZHU, W. (2019). A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* **31** 833–852.

DE SILVA, V. and TENENBAUM, J. (2002). Global versus local methods in nonlinear dimensionality reduction. *Adv. Neural Inf. Process. Syst.* **15**.

DECELLE, A., KRZAKALA, F., MOORE, C. and ZDEBOROVÁ, L. (2011). Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E* **84** 066106.

DEVROYE, L. and WISE, G. L. (1980). Detection of abnormal behavior via nonparametric estimation of the support. *SIAM J. Appl. Math.* **38** 480–488. MR0579432 https://doi.org/10.1137/0138038

DONG, Y., CHAWLA, N. V. and SWAMI, A. (2017). Metapath2vec: Scalable representation learning for heterogeneous networks. Kid 17, 2017, Halifax, NS, Canada.

DONG, W., MOSES, C. and LI, K. (2018). Efficient *k*-nearest neighbour graph construction for generic similarity measures. In *Proceedings of the* 20*th International Conference of the World Wide Web* 577–586, New York.

DU, L., WANG, Y., SONG, G., LU, Z. and WANG, J. (2018). Dynamic network embedding: An extended approach for Skip-Gram based network embedding. In *Proceedings of the* 27*th International Joint Conference on Artificial Intelligence*, *IJ*(*AI*-18).

DUCHAMP, T. and STUETZLE, W. (1996). Extremal properties of principal curves in the plane. *Ann. Statist.* **24** 1511–1520. MR1416645 https://doi.org/10.1214/aos/1032298280

EDELSBRUNNER, H., LETCHER, D. and ZOMORODIAN, A. (2002). Topological persistence and simplification. *Discrete Comput. Geom.* **28** 511–533. MR1949898 https://doi.org/10.1007/s00454-002-2885-2

GENOVESE, C. R., PERONE-PACIFICO, M., VERDINELLI, I. and WASSERMAN, L. (2012). Manifold estimation and singular deconvolution under Hausdorff loss. *Ann. Statist.* **40** 941–963. MR2985939 https://doi.org/10.1214/12-AOS994

GENOVESE, C. R., PERONE-PACIFICO, M., VERDINELLI, I. and WASSERMAN, L. (2014). Nonparametric ridge estimation. *Ann. Statist.* **42** 1511–1545. MR3262459 https://doi.org/10.1214/14-AOS1218

GHOJOGH, B., GHODSI, A., KARRAY, F. and CROWLEY, M. (2021). Johnson–Lindenstrauss lemma, linear and nonlinear random projections, random Fourier features and random kitchen sinks: Tutorial and survey. Preprint. Available at arXiv:2108.04172v1.

GHRIST, R. (2018). Homological algebra and data. In *The Mathematics of Data. IAS/Park City Math. Ser.* **25** 273–325. Amer. Math. Soc., Providence, RI. MR3839171

GIRVAN, M. and NEWMAN, M. E. J. (2002). Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **99** 7821–7826. MR1908073 https://doi.org/10.1073/pnas.122653799

GREENE, D. and CUNNINGHAM, P. (2011). Tracking the evolution of communities in dynamic social networks. Report Idiro Technologies, Dublin, Ireland.

GRETTON, A. (2019). Introduction to RKHS, and some simple kernel algorithms. Lecture notes.

GROVER, A. and LESKOVEC, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the* 22*nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 855–864.

HAGHVERDI, L., BUETTNER, F. and THEIS, F. J. (2015). Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics* **31** 2989–2998. https://doi.org/10.1093/bioinformatics/btv325

HASTIE, T. (1984). Principal curves and surfaces. Laboratory for Computational Statistics Technical Report 11, Stanford Univ., Dept. Statistics. MR2634007

HASTIE, T. and STUETZLE, W. (1989). Principal curves. *J. Amer. Statist. Assoc.* **84** 502–516. MR1010339

HASTIE, T., TIBSHIRANI, R. and FRIEDMAN, J. (2019). *The Elements of Statistical Learning*: *Data Mining*, *Inference*, *and Prediction*, 2nd ed. *Springer Series in Statistics*. Springer, New York. MR2722294 https://doi.org/10.1007/978-0-387-84858-7

HINTON, G. E. and ROWEIS, S. T. (2002). Stochastic neighbour embedding. *Adv. Neural Inf. Process. Syst.* **15** 833–840.

HINTON, G. E. and SALAKHUTDINOV, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science* **313** 504–507. MR2242509 https://doi.org/10.1126/science.1127647

HOFF, P. D., RAFTERY, A. E. and HANDCOCK, M. S. (2002). Latent space approaches to social network analysis. *J. Amer. Statist. Assoc.* **97** 1090–1098. MR1951262 https://doi.org/10.1198/016214502388618906

HOLLAND, P. W., LASKEY, K. B. and LEINHARDT, S. (1983). Stochastic blockmodels: First steps. *Soc. Netw.* **5** 109–137. MR0718088 https://doi.org/10.1016/0378-8733(83)90021-7

HOTELLING, H. (1933). Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.* **24** 417–441.

HOTELLING, H. (1936). Relations between two sets of variates. *Biometrika* **28** 321–377.

HYVÄRINEN, A. and OJA, E. (2000). Independent component analysis: Algorithms and applications. *Neural Netw.* **13** 411–430.

JOHNSON, W. B. and LINDENSTRAUSS, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. In *Conference in Modern Analysis and Probability* (*New Haven*, *Conn.*, 1982). *Contemp. Math.* **26** 189–206. Amer. Math. Soc., Providence, RI. MR0737400 https://doi.org/10.1090/conm/026/737400

JOLLIFFE, I. T. (2002). *Principal Component Analysis*, 2nd ed. *Springer Series in Statistics*. Springer, New York. MR2036084

JOSSE, J. and HUSSON, F. (2012). Selecting the number of components in principal component analysis using cross-validation approximations. *Comput. Statist. Data Anal.* **56** 1869–1879. MR2892383 https://doi.org/10.1016/j.csda.2011.11.012

KARRER, B. and NEWMAN, M. E. J. (2011). Stochastic blockmodels and community structure in networks. *Phys. Rev. E* (3) **83** 016107, 10 pp. MR2788206 https://doi.org/10.1103/PhysRevE.83.016107

KAZEMI, S. M., GOEL, R., JAIN, K., KOBYZEV, I., SETHI, A., FORSYTH, P. and POUPART, P. (2020). Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.* **21** Paper No. 70, 73 pp. MR4095349

KIM, J., RINALDO, A. and WASSERMAN, L. (2019). Minimax rates for estimating the dimension of a manifold. *J. Comput. Geom.* **10** 42–95. MR3918925 https://doi.org/10.20382/jocg.v10i1a3

KOBOUROV, S. (2012). Spring embedders and forced directed graph drawing algorithms. Preprint. Available at arXiv:1201.3011.

KOHONEN, T. (1982). Self-organized formation of topologically correct feature map. *Biol. Cybernet.* **43** 59–69.

KONISHI, S. and KITAGAWA, G. (2008). *Information Criteria and Statistical Modeling*. *Springer Series in Statistics*. Springer, New York. MR2367855 https://doi.org/10.1007/978-0-387-71887-3

KOSSINETS, G. and WATTS, D. J. (2006). Empirical analysis of an evolving social network. *Science* **311** 88–90. MR2192483 https://doi.org/10.1126/science.1116869

LEE, C. and WILKINSON, D. J. (2019). A review of stochastic block models and extensions for graph clustering. *Appl. Netw. Sci.* **4** 122.

LEI, J. and RINALDO, A. (2015). Consistency of spectral clustering in stochastic block models. *Ann. Statist.* **43** 215–237. MR3285605 https://doi.org/10.1214/14-AOS1274

LEVINA, E. and BICKEL, P. (2004). Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems* (L. Saul, Y. Weiss and L. Bottou, eds.) **17**. MIT Press, Cambridge, MA.

LI, P., HASTIE, T. J. and CHURCH, K. W. (2007). Nonlinear estimators and tail bounds for dimension reduction in l_1 using Cauchy random projections. *J. Mach. Learn. Res.* **8** 2497–2532. MR2353840 https://doi.org/10.1007/978-3-540-72927-3_37

LIM, B. and ZOHREN, S. (2021). Time-series forecasting with deep learning: A survey. *Philos. Trans. R. Soc. Lond. A* **379** Paper No. 20200209, 14 pp. MR4236146 https://doi.org/10.1098/rsta.2020.0209

LITTLE, A. V., MAGGIONI, M. and ROSASCO, L. (2011). Multiscale geometric methods for estimating intrinsic dimension. In *Proc. SampTA* 4:2.

LUDKIN, M., ECKLEY, I. and NEAL, P. (2018). Dynamic stochastic block models: Parameter estimation and detection of changes in community structure. *Stat. Comput.* **28** 1201–1213. MR3850391 https://doi.org/10.1007/s11222-017-9788-9

LUNDE, B. Å. S., KLEPPE, T. S. and SKAUG, H. J. (2020). An information criterion for automatic gradient tree boosting. Preprint. Available at arXiv:2008.05926.

MARKOV, A. (1958). The insolubility of the problem of homeomorphy. *Dokl. Akad. Nauk SSSR* **121** 218–220. MR0097793

MCINNES, L., HEALY, J. and MELVILLE, J. (2018). UMAP: Uniform manifold approximation for dimension reduction. Preprint. Available at arXiv:1802.03426v2.

MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. and DEAN, J. (2013). Distributed representation of words and phrases and their composability. In *Advances in Neural Information Processing Systems* 26: *Proceedings Annual* 27*th Conference on Neural Information Processing Systems*, *Lake Tahoe*, *Nevada*, *USA*.

NEWMAN, M. E. J. (2006). Modularity and community structure in networks. *Proc. Natl. Acad. Sci.* **103** 8577–8582.

NEWMAN, M. (2020). *Networks*, 2nd ed. Oxford Univ. Press, Oxford. MR3838417 https://doi.org/10.1093/oso/9780198805090.001.0001

NEWMAN, M. E. J. and GIRVAN, M. (2004). Finding and evaluating community networks. *Phys. Rev. E* **69** 026113.

NEWMAN, M. E. J. and REINERT, G. (2016). Estimating the number of communities in a network. *Phys. Rev. Lett.* **137** 078301.

NIYOGI, P., SMALE, S. and WEINBERGER, S. (2008). Finding the homology of submanifolds with high confidence from random samples. *Discrete Comput. Geom.* **39** 419–441. MR2383768 https://doi.org/10.1007/s00454-008-9053-2

OTNEIM, H., JULLUM, M. and TJØSTHEIM, D. (2020). Pairwise local Fisher and naive Bayes: Improving two standard discriminants. *J. Econometrics* **216** 284–304. MR4077395 https://doi.org/10.1016/j.jeconom.2020.01.019

OZERTEM, U. and ERDOGMUS, D. (2011). Locally defined principal curves and surfaces. *J. Mach. Learn. Res.* **12** 1249–1286. MR2804600

PEARSON, K. (1901). On lines and planes of closest fit to systems of points in space. *Philos. Mag.* **2** 559–572.

PEIXITO, T. P. (2021). Descriptive vs. inferential community detection: Pitfalls, myths and half-truths. Preprint. Available at arXiv:2112.00183v1.

PEIXOTO, T. P. (2019). Bayesian stochastic blockmodeling. In *Advances in Network Clustering and Blockmodeling* 289–332.

PEROZZI, B., AL-RFOU, R. and SKIENA, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the* 20*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 701–710.

QIAO, W. and POLONIK, W. (2021). Algorithms for ridge estimation with convergence guarantees. Preprint. Available at arXiv:2014.12314v1.

QIU, J., DONG, Y., MA, H., LI, J., WANG, K. and TANG, J. (2018). Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings WSDM*. ACM, New Tork.

QIU, J., DONG, Y., MA, H., LI, J., WANG, K. and TANG, J. (2019). NetSMF: Large-scale network embedding as sparse matrix factorization. In *Proceedings of the* 2019 *World Wide Web Conference*, *May* 13–17, *San Francisco*, *CA*, *USA*.

RAVISSHANKER, N. and CHEN, R. (2019). Topological data analysis (TDA) for time series. Preprint. Available at arXiv:1909.10604v1.

ROHE, K., CHATTERJEE, S. and YU, B. (2011). Spectral clustering and the high-dimensional stochastic blockmodel. *Ann. Statist.* **39** 1878–1915. MR2893856 https://doi.org/10.1214/11-AOS887

ROHE, K., QIN, T. and YU, B. (2016). Co-clustering directed graphs to discover asymmetries and directional communities. *Proc. Natl. Acad. Sci. USA* **113** 12679–12684. MR3576189 https://doi.org/10.1073/pnas.1525793113

ROWEIS, S. T. and SAUL, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science* **290** 2323–2326. https://doi.org/10.1126/science.290.5500.2323

SALINAS, D., FLUNKERT, V., GASTHAUS, J. and JANUSCHOWSKI, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* **36** 1181–1191.

SAMMON, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.* **18** 403–409.

SCHÖLKOPF, B., SMOLA, A. and MÜLLER, K.-L. (2005). Kernel principal components. *Lecture Notes in Comput. Sci.* **1327** 583–588.

SHAHRIARI, B., SWERSKY, K., WANG, Z., ADAMS, R. P. and DE FREITAS, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* **104** 148–175.

SINGH, G., MEMOLI, F. and CARLSSON, G. (2007). Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Eurographics Symposium on Point Based Graphics* (M. Botsch and R. Pajarola, eds.). The Eurographics Association.

SUN, Y., NORICK, B., HAN, J., YAN, X., YU, P. and YU, X. (2012). Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *KDD* '12: *Proceedings of the* 18*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1348–1356.

TANG, J., QU, M. and MEI, Q. (2015). PTE: Predictive text embedding through large-scale heterogeneous text networks. Preprint. Available at arXiv:1508.00200v1.

TANG, J., QU, M., WANG, M., ZHANG, M., YAN, J. and MEI, Q. (2015). LINE: Large-scale information network embedding. In *Proceedings of the* 24*th International Conference on World Wide Web* 1067–1077.

TANG, J., LIU, J., ZHANG, M. and MEI, Q. (2016). Visualizing large-scale and high-dimensional data. In *Proceedings of the* 25*th International Conference on World Wide Web* 287–297.

TENENBAUM, J. B., DE SILVA, V. and LANGFORD, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science* **290** 2319–2323. https://doi.org/10.1126/science.290.5500.2319

TJØSTHEIM, D., JULLUM, M. and LØLAND, A. (2023). Some recent trends in embedding of time series and dynamic networks. *J. Time Ser. Anal.* To appear. https://doi.org/10.1111/jtsa.12677

TJØSTHEIM, D., JULLUM, M. and LØLAND, A. (2023). Supplement to "Statistical embedding: Beyond principal components". https://doi.org/10.1214/22-STS881SUPP

TJØSTHEIM, D., OTNEIM, H. and STØVE, B. (2022a). Statistical dependence: Beyond Pearson's ρ. *Statist. Sci.* **37** 90–109. MR4371097 https://doi.org/10.1214/21-sts823

TJØSTHEIM, D., OTNEIM, H. and STØVE, B. (2022b). *Statistical Modeling Using Local Gaussian Approximation*. Elsevier/Academic Press, London. MR4382419

TORGERSON, W. S. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika* **17** 401–419. MR0054219 https://doi.org/10.1007/BF02288916

TUTTE, W. T. (1963). How to draw a graph. *Proc. Lond. Math. Soc.* (3) **13** 743–767. MR0158387 https://doi.org/10.1112/plms/s3-13.1.743

VAN DER MAATEN, L. (2014). Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.* **15** 3221–3245. MR3277169

VAN DER MAATEN, L. and HINTON, G. (2008). Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9** 2579–2605.

VAN DER MAATEN, L., POSTMA, E. and VAN DER HERIK, J. (2009). Dimensionality reduction: A comparative review. Tilburg Centre for Creative Computing, TiCC TR 2009.005.

VON LUXBURG, U. (2007). A tutorial on spectral clustering. *Stat. Comput.* **17** 395–416. MR2409803 https://doi.org/10.1007/s11222-007-9033-z

WANG, Y. X. R. and BICKEL, P. J. (2017). Likelihood-based model selection for stochastic block models. *Ann. Statist.* **45** 500–528. MR3650391 https://doi.org/10.1214/16-AOS1457

WASSERMAN, L. (2018). Topological data analysis. *Annu. Rev. Stat. Appl.* **5** 501–535. MR3774757 https://doi.org/10.1146/annurev-statistics-031017-100045

WEI, Y.-C. and CHENG, C.-K. (1989). Towards efficient hierarchical designs by ratio cut partitioning. In 1989 *IEEE International Conference on Computer-Aided Design. Digest of Technical Papers* 298–301. IEEE.

XIE, H., LI, J. and XUE, H. (2018). A survey of dimensionality reduction techniques based on random projection. Preprint. Available at arXiv:1706.04371v4.

YOUNG, G. and HOUSEHOLDER, A. S. (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika* **3** 19–22.

YOUNG, T., HAZARIKA, D., PORIA, S. and CAMBRIA, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **13** 55–75.

ZHANG, J. and CHEN, Y. (2020). Modularity based community detection in heterogeneous networks. *Statist. Sinica* **30** 601–629. MR4213981

ZHENG, Q. (2016). Spectral techniques for heterogeneous social networks. Ph.D. thesis, Queen's Univ., Ontario, Canada.

ZHOU, C., LIU, Y., LIU, X. and GAO, J. (2017). Scalable graph embedding for asymmetric proximity. In *Proceedings of the* 31*st AAAI Conference on Artificial Intelligence*.

ZHU, X. and PAN, R. (2020). Grouped network vector autoregression. *Statist. Sinica* **30** 1437–1462. MR4257540 https://doi.org/10.5705/ss.202017.0533

ZHU, X., PAN, R., LI, G., LIU, Y. and WANG, H. (2017). Network vector autoregression. *Ann. Statist.* **45** 1096–1123. MR3662449 https://doi.org/10.1214/16-AOS1476

ZOMORODIAN, A. and CARLSSON, G. (2005). Computing persistent homology. *Discrete Comput. Geom.* **33** 249–274. MR2121296 https://doi.org/10.1007/s00454-004-1146-y