Some recent trends in embeddings of time series and dynamic networks

Dag Tjøstheim[1,2] Martin Jullum[2] Anders Løland[2]

[1]University of Bergen
[2]Norwegian Computing Center

Abstract

We give a review of some recent developments in embeddings of time series and dynamic networks. We start out with traditional principal components and then look at extensions to dynamic factor models for time series. Unlike principal components for time series, the literature on time-varying nonlinear embedding is rather sparse. The most promising approaches in the literature is neural network based, and has recently performed well in forecasting competitions. We also touch upon different forms of dynamics in topological data analysis. The last part of the paper deals with embedding of dynamic networks, where we believe there is a gap between available theory and the behavior of most real world networks. We illustrate our review with two simulated examples. Throughout the review, we highlight differences between the static and dynamic case, and point to several open problems in the dynamic case.

1 Introduction

Traditional time series analysis handles equidistant observations in time, scalars or vectors. If it is a vector time series, the number of components is typically small or moderate, a possible exception being a panel of time series.

A characteristic feature of the big data revolution is that observations may be multivariate with literally millions of components. This has put restrictions on some of the traditional statistical methods and prompted the introduction of new ones, the latter often being treated in the machine learning literature. The large number of components have necessitated the search for new embeddings methods, where an embedding is assumed to keep original characteristic features of the data, but in a much lower-dimensional space, making further analysis easier and manageable. Principal component analysis is probably still the most used embedding method, but it cannot cope with data of ultra-high dimension due to the fact that then an ultra-dimensional eigenvalue problem is involved. Moreover, the data may be nonlinear in structure where the dependency relationships cannot be well modeled by variances and covariances as required by principal component analysis. Finally, ordinary principal components are not dynamic in nature, not taking care of auto dependencies in its construction, although that particular problem has been nearly fully addressed in the last two decades through the concepts of dynamic principal components and dynamic factor analysis.

A number of nonlinear alternative embedding methods exist, and are surveyed in Tjøstheim et al. (2022a), TJL in the following. In that paper, we also reviewed topological data analysis (TDA), such an analysis being able to handle data with voids and cavities.

Another feature in current data analysis is that data often come in the form of networks or graphs. The mathematical theory of graphs dates far back in time, but the statistical analysis of networks is of much more recent date. In fact there has been a tremendous growth in the literature on networks and their applications. Again we refer to TJL for a review of this development.

The survey in TJL is virtually completely limited to a static situation, and in fact an overwhelming part of the literature is restricted to this situation. But it is clear that in many cases the static assumption fails. Consider for example a network of bank customers. Such a network is changing in time: the strength of relationships between customers will in general change, and there may appear new customers joining the network and others leaving. This causes problems for instance in use of network methods in the detection of bank fraud such as money laundering (Jullum et al., 2020). Another example of a time changing network occurs in imaging of changes in a brain network (Kucyi et al., 2017). An obvious third example is online social networks where 'friends' and followers connect and interact with each other, enter and leave groups as times go by (Sarkar and Moore, 2005).

How does this mesh with available modeling tools for analyzing such cases? Does there exist a statistical theory for time-varying embeddings and time-varying networks, and can it be applied to tasks as those mentioned above? The answer is a partial yes. Relevant methods are in the process of being developed, but many unsolved problems remain. This means that this area of research may be a treasure trove for time series analysts looking for new and inspiring problems. Much of the research so far is very recent and mostly found in the machine learning literature, often in the form of preprints. A main goal of this paper is to try, in a time-varying context, to focus on possibilities for building a bridge between algorithmic machine learning and a more model-based statistical approach. This will be done by advancing from TJL to a survey of very recent and challenging problems in time series and dynamic networks embeddings.

In Section 2 we start with the principal component embedding method in time series and the classical work of David Brillinger. Inspired by his work in the last couple of decades there have been important developments in dynamic principal component analysis and dynamic factor analysis. We also explain why it does not always succeed. Alternative time series embeddings are discussed in Section 3 by starting out with nonlinear statistical embedding methods such as for instance multidimensional scaling and local linear methodology. Time-varying topological data analysis is the topic of Section 4, and in Section 5 we treat temporary variation and time series in networks. In the paper we extend two illustrative examples from TJL to a situation where we look at robustness of a number of methods to certain changes in time. We should also stress that in reading this paper, even though an attempt has been made to make it self-contained, it may be an advantage to have the general paper TJL at hand, where the fundamental concepts and definitions can be looked up in more detail.

2 Principal components in time series

In general the principal component method is probably the most used method for reducing the dimension of data but still trying to retain the essential information. For iid (independent identically distributed) vector data this makes the principal component method to a yardstick against which other embedding methods may be compared. For time series the situation is less clear.

As is well known, for p-dimensional observations the population principal components $V_j, j = 1, \dots, p$ are obtained by solving the eigenvalue problem

$$\mathbf{\Sigma} V_j = \lambda_j V_j, \tag{1}$$

where $\mathbf{\Sigma}$ is the $p \times p$ dimensional population covariance matrix of the data. Let the observations $X_i, i = 1, \dots, T$ have components X_{ji}, and let $\mathbf{X}$ be the matrix $\mathbf{X} = \{(X_{ji} - \bar{X}_j)\}$, with $\bar{X}_j = T^{-1} \sum_i X_{ji}$, then an estimate of $\mathbf{\Sigma}$ is obtained from $T^{-1}[\mathbf{X}\mathbf{X}^T]$, and the estimated principal components are obtained from

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

There are two main weaknesses of the PCA (principal component analysis). The eigenvalue equations are based on second moments only, and if the dependencies of the data are not well described by second moments, as is not infrequently the case for financial time series, the value of principal components could be strongly reduced, and one may profit by using some of the nonlinear methods laid out in the next section (see also TJL). The second weakness of PCA is that for very high dimensional data (large p) the eigenvalue problem may be cumbersome and time consuming to solve. Such solutions are not always needed, however, see Section 5.5 on scalability.

For time series there is another difficulty. The PCA method, as applied traditionally, neglects all dependence in time inherent in say auto and cross correlations in time. In fact there are many examples where auto dependencies are simply ignored, for example in portfolio analysis of stock data by PCA, neglecting that stock data do not consist of iid data. The consequences of using ordinary PCA on time series data and possible associated pitfalls have recently been analyzed by Zhang and Tong (2022). These authors derive asymptotic theory for eigenvalues and eigenvectors for the PCA eigenvalue equation (2), under several forms for time series dependence. This problem has been considered earlier, notably by the recipient of this birthday volume, Masanobu Taniguchi, in Tanigushi and Krishnaiah (1987).

A more serious problem, though, is the structural difficulties that can emerge (or are likely to emerge) if auto dependence is neglected. For example a principal component that is obtained as one of the least significant in terms of a low eigenvalue, may actually be of central importance because it may happen to have a strong time dependence in terms of autocorrelation.

Is there an alternative approach where this general problem is taken into account? The answer to this question is yes, and is constituted by the classical contribution of David Brillinger in Brillinger (1969) and in Chapter 9 of his book Brillinger (1975). The device used by Brillinger was to replace the covariance matrix $\mathbf{\Sigma}$ by the $p \times p$ dimensional spectral density matrix

$$\mathbf{f}(\omega) = (2\pi)^{-1} \sum_{u=-\infty}^{\infty} \mathbf{\Sigma}_u \exp\{-i\omega u\}, \tag{3}$$

where $\mathbf{\Sigma}_u$ is the auto and cross covariance matrix at lag u. This expression includes all the second order dependence in time, and it gives a PCA decomposition in the frequency domain by solving the eigenvalue problem

$$\mathbf{f}(\omega) V_j(\omega) = \lambda_j(\omega) V_j(\omega),$$

resulting in principal components $V_j(\omega)$ depending on frequency. By taking the inverse Fourier transform of the eigenvectors of the spectral density matrix, the co-called dynamic principal components are the result. This procedure assumes that the time series is stationary, guaranteeing the existence of the spectral distribution and in addition that it is absolutely continuous so that the spectral density matrix exists. In Pena and Yohai (2016) the stationarity assumption is dropped. The general scheme and idea of Brillinger is kept, but Pena and Yohai (2016) allows for dynamic principal components that may not be linear combinations of the observations, and they may be based on a variety of loss functions, including robust ones. On the other hand this more general framework restricts the possibility of undertaking a rigorous statistical inference procedure.

For ordinary principal components there is an intimate connection between principal components and factor analysis. This is somewhat less straightforward, using asymptotic arguments, in the dynamic case. It is explained in considerable detail in Section 3.1 of Hallin and Lippi (2013). That paper is one of a series of influential and much cited papers by Forni, Hallin, Lippi and Reichlein. The first paper, where fundamentals are explained, is Forni et al. (2000). In that paper, so-called generalized dynamic factor models are introduced. We refer to the introduction of that paper for a historic account of developments leading up to generalized dynamic factor models, including papers by Sargent and Sims (1977), Geweke (1977), Chamberlain (1983), Chamberlain and Rotschild (1983), Stock and Watson (2002). The model proposed in Forni et al. (2000), and elaborated on in a number of follow-up papers referenced in Hallin and Lippi (2013), can be stated as

$$X_{jt} = b_{j1}(L)u_{1t} + b_{j2}(L)u_{2t} + \cdots b_{jm}(L)u_{mt} + \xi_{jt}, \tag{4}$$

where $\{X_{jt}, j = 1, \ldots, p, t = 1, \ldots, T\}$ is the original collection of time series, $b_{j1}(L), \ldots, b_{jm}(L)$ are polynomials (or more general infinite MA components) in the lag operator L associated with each of m orthogonal white noise factor time series $\{u_{jt}\}$; the common components. The series $\{\xi_{jt}, j = 1, \ldots, p, t = 1, \ldots, T\}$ are series of so-called idiosyncratic components, which are zero-mean vector time series such that ξ_{jt} is orthogonal to $u_{i,t-k}$ for any i, j, t and k. The authors provide identification conditions, propose an estimator for the common components, and prove convergence results as both p and T tend to infinity. They use the model to construct a coincident index for GDP for countries in the European Union. Some very recent related papers are Chen et al. (2022) and Yu et al. (2022).

A theoretical challenge with the dynamic factor models stated in (4) is that one has to let p and/or T tend to infinity in some of the theoretical derivations of the model. For example, these factor models are only asymptotically identifiable. If one is willing to simplify the models, at least some of these difficulties can be avoided. A fine example of such a modification is the paper Lam and Yao (2012). They propose a model of form

$$X_t = \mathbf{A}Y_t + \varepsilon_t$$

where $\mathbf{A}$ is a $p \times m$ dimensional matrix, $\{Y_t\}$ is a m-dimensional latent factor process and $\{\varepsilon_t\}$ is a p-dimensional vector white noise process. They obtain asymptotic properties of estimators under two settings: (i), where $T \to \infty$ with p fixed and (ii) when both T and p tend to infinity.

In spite of the success of dynamic factor type models, these models remain linear models with dependence based on second order moments such as variances and covariances. (A possible untried way out of this difficulty is to replace covariances and cross covariances with corresponding local covariances and cross covariances as in Tjøstheim et al. (2022c).) In addition the dynamic factor models do depend on solving an eigenvalue problem which may turn out to be a prohibitive task if the dimension of $\{X_t\}$ is very large. Fortunately, there are alternative nonlinear embedding procedures that can be tried out as an alternative. These were surveyed in the static case in TJL. The dynamic case will be treated in the next section.

3 Nonlinear dynamic embeddings

3.1 The methods

We start by briefly listing the nonlinear embedding methods. Each method has been developed with a specific nonlinear situation in mind, and none of them work equally well in all situations. Presently we just mention the main principle underlying the construction of each method. Much more details are given in TJL. The dynamic extension for some of them are treated subsequently in Section 3.2.

For the principal curve method, Hastie (1984), the data are supposed to be concentrated roughly on a curve or more generally on a submanifold. Although the data in this case 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 of Roweis and Saul (2000) and to ISOMAP (Tenenbaum et al., 2000). Distance preservation is the dominating principle in the classical nonlinear method of multidimensional scaling – MDS (Torgerson, 1952), and in fact ideas from MDS have served as a basis for several of the more recent nonlinear embedding methods.

A case which is particularly difficult to handle for PCA is the case where data lie on chained non-convex structures as for instance in Figure 1a. In this figure we present a data set that will be used for illustration purposes throughout and also in Section 4 on topological data analysis. The raw data of Figure 1a consists of parts of three parametric curves, each being obtained from the so-called Ranunculoid[1], but with three different parameter sets. The arc lengths are 72 for class 1, 96 for class 2 and 120 for class 3. In addition, the curves have been perturbed by independent Gaussian noise with a standard deviation of 2 (and independence between the two dimensions). For such and similar structures one may try to map the dependence properties to a graph or network (we use graph and network interchangeably in this paper), see TJL.

In still other situations it may be advantageous to use a nonlinear transformation of the data points prior to embedding, and then solve a resulting eigenvalue problem for the embedded variables, as is done in kernel principal components in an associated reproducing kernel Hilbert space, see e.g. Schölkopf et al. (2005). All of these methods and some additional ones are explained in more detail in TJL, and most of them illustrated by applying them to simulated data as in Figure 1.

A very important asset of PCA is the relationship of PCA to factor analysis, where there exists a simple eigenvalue ratio-criterion for choosing the number of factors, or the number of principal components to keep in the general embedding. It should be noted that something similar to factor analysis is much harder to establish in nonlinear type embeddings. And for other nonlinear methods there seems to be no obvious way of determining the embedding dimension, meaning that it is often done by a trial and error routine.

[1] https://mathworld.wolfram.com/Ranunculoid.html

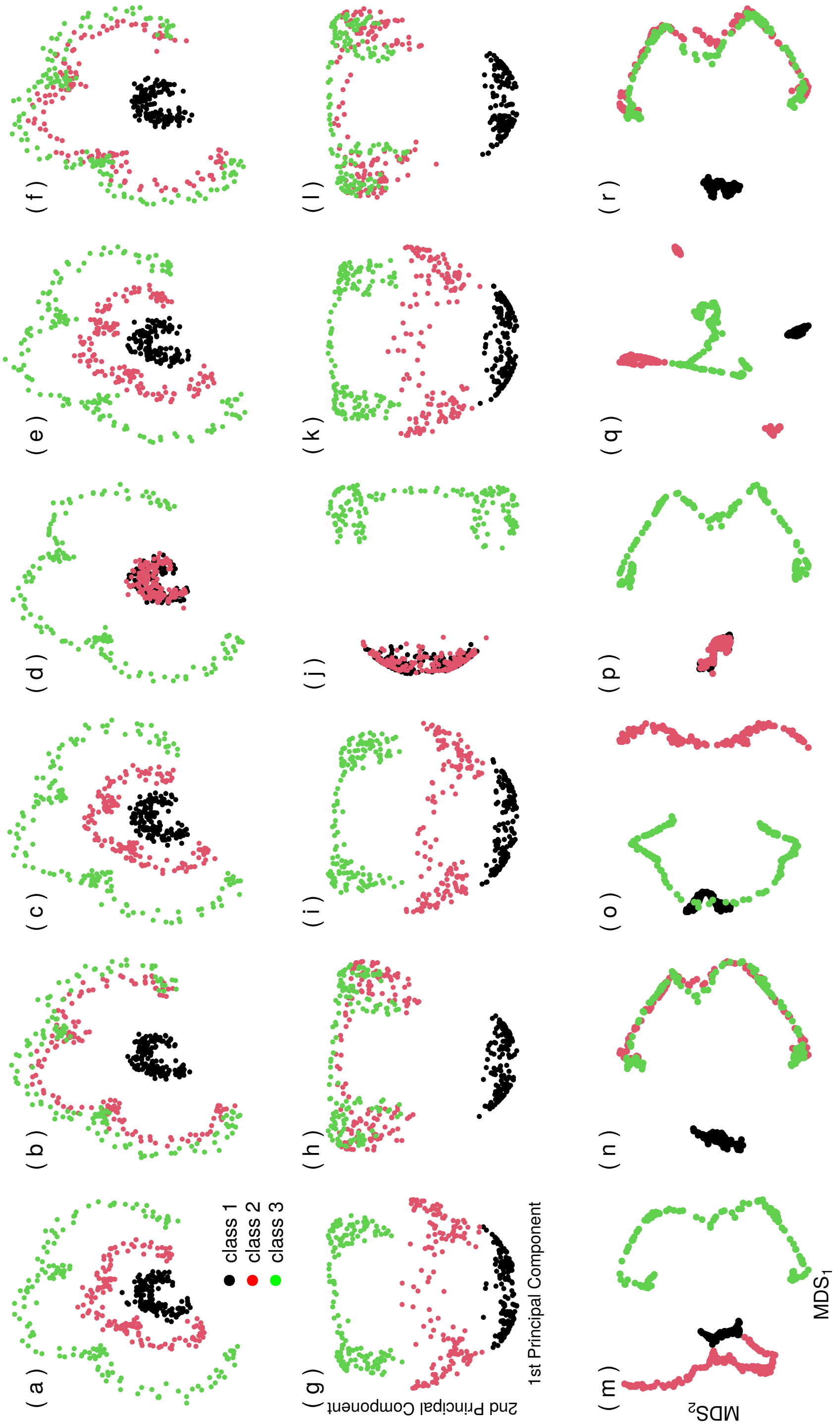

Figure 1: a: Three parametric curves from the so-called Ranunculoid, perturbed by Gaussian noise with a standard deviation of 2. b-f: The middle Ranunculoid curve oscillates between the innermost and outermost curves, five time points are shown here. g-l: Kernel principal components (with a Bessel kernel) of a-f. m-r: ISOMAP, with MDS_1 and MDS_2 directions, of a-f.

The most common application of embeddings of iid data is clustering and classification. The embedding often results in a lower dimensional space, and clustering is more easily done in this space than in the original high-dimensional case. This is in particular the case if the embedding is done to a low-dimensional Euclidean space $\mathbb{R}^m$, to which standard clustering methods like k-means can be applied.

For time series data as treated in the present paper, forecasting comes in as an additional and perhaps even more important problem. How does one forecast an ultra-high dimensional time series if one wants to include cross and auto dependence between the series. Lately artificial neural networks have played an increasingly important role in such a setting.

To make the paper more self-contained a brief review of recent developments in artificial neural networks is contained in the Supplement (Tjøstheim et al., 2022b).

The neural network models, as seen in the Supplement, can contain a rather large number of unknown parameters to be learned during training. The possibility of over-fitting is very real and quite often a regularization term is added to penalize too many parameters. This is in a way analogous to the use of a penalty term in the AIC criterion for time series. We give examples in Section 3.3, equations (5) and (6).

3.2 Time-varying nonlinear embedding

In the beginning of this section we listed a number of embeddings methods. How do these methods hold up in the presence of dynamics in form of time variations? We have to distinguish between two types of time variations: stationary or nonstationary.

The algorithms for the methods of MDS, ISOMAP, LLE, spectral graph methods, diffusion maps, random projections and kernel principal components all work if time variations are neglected in the sense that computations are done as if the data are iid. They may produce useful results but the (asymptotic) foundation that these results build on is uncertain. This has been illustrated by Zhang and Tong (2022) in the case of asymptotic analysis of eigenvalues and eigenvectors if ordinary principal components are applied in a time series situation. We are not aware that any corresponding analysis has been undertaken for the nonlinear embedding methods mentioned in this paragraph.

For principal components there exists a modification of the method in the so-called dynamic principal components and dynamic factors initially based on the work by Brillinger (1975) in the spectral domain. This was reviewed in Section 2, and has been used in many successful applications. Again, to the best of our knowledge, there is no work in this direction for the nonlinear embeddings mentioned in the previous paragraph. A very different alternative may be to take the Brillinger spectral approach in another direction, by using either a higher order spectrum or a local spectrum, cf. Jordanger and Tjøstheim (2022), as a point of departure for a nonlinear dynamic principal component analysis.

As far as embedding in a time-varying framework is concerned, it may be advantageous to use a neural network-based learning approach. The reason is that in the training process the time sequence of the observations are taken into account. (This dependence on time may even be nonstationary.) A prime example of embedding in using neural networks is the much cited paper by Hinton and Salakhutdinov (2006), where they use autoencoding as a tool.

Autoencoding in its simple basic form has two main parts: an encoder that maps the input into the code, and a decoder that maps the code to a reconstruction of the signal. Essentially it is a one hidden layer feedforward neural network, where the output y-variable in principle should be identical to the input x-variable, but where in practice autoencoders are typically forced to reconstruct the input approximately, preserving only the most relevant aspects of the data. The low dimensional embedding is constituted by the hidden layer representation. The autoencoders have been further developed from this simple one hidden layer structure. Some of the most powerful and recent artificial intelligence methods involve autoencoders stacked inside deep neural networks, see e.g. Domingos (2015).

Autoencoding is a data analytic or machine learning technique that has turned out to be instrumental in dimension reduction. But unlike dynamic principal components it does not rest on a traditional mathematical statistics foundation. It contains parameters, the embedding dimension being one of them, that must be chosen, and it may not be obvious how an optimal choice can be made. This potential lack of a statistical model-based routine to estimate parameters in an algorithm accentuates a difference between a machine learning approach and a mathematical statistics approach. We refer to TJL for a further discussion of this.

When it is clear that the time variations are nonstationary, one has to take this into account in the embedding. Then, unlike the stationary case, the embedding can be expected to change significantly with time. There are some papers on this, largely empirical in nature. Time-varying multidimensional scaling is examined in Lopes and Machado (2014), He et al. (2018). Time-varying principal curves are looked at by Li and Guedj (2021), whereas two types of ISOMAP streaming are considered in Mahapatra and Chandola (2020) and an extension of ISOMAP for data with both spatial and temporal relationships is developed in Jenkins and Matarić (2004). Lian et al. (2015) study diffusion maps using a Kullback-Leibler criterion. Another application of diffusion maps to periodic physiological data is in Lin et al. (2021).

To our knowledge there is no systematic comparison of the embedding methods under time-varying circumstances. Can one find classes of nonstationarity and corresponding classes of "optimal" embedding methods? Are some embedding methods more robust than others to say periodic disturbances or to outliers? This seems to be a wide open research field.

In TJL we illustrated some main embedding algorithms on a data set consisting of the chained Ranunculoid curves with added Gaussian noise, as depicted in Figure 1a. Here we develop this illustration further by examining how the various methods react to changes in the data structure. We let the Ranunculoid curves go through a sinusoidal movement. More precisely, we let the middle Ranunculoid curve oscillate between the curves 1 and 3 as depicted in Figures 1b-1f, where five time points are displayed. In addition, different seeds have been used in the generation of the point cloud for each time point. The present example could be considered as a very simple toy example motivated by physiological experiments that quite often has an oscillatory structure.

In the static case of TJL we concluded that PCA does not work for such a chain of non-convex curves, and it is therefore meaningless to try to use PCA to describe the temporal data in Figures 1b-1f. However, the nonlinear embedding methods mentioned in the current section have the potential to pick up this oscillatory movement. This is illustrated by the kernel principal component method[2] in Figures 1g-l and for the

[2] We have used the R function `kpca` from the `kernlab` package (Karatzoglou et al., 2022) using a Bessel kernel function with $\sigma = 0.1$ and an order of 2.

ISOMAP[3] in Figures 1m-r. It is seen that it gives a faithful embedding representation of the periodic pattern of the Ranunculoid structures. The embedding is rotated 90 degrees in Figure 1j, which might be because the sample eigenvectors are biased (Hellton and Thoresen, 2017). As can be expected the kernel principal component method is not able to distinguish between all three curves when the middle Ranunculoid curve is very close to the innermost or the outermost curve. When applying the ISOMAP algorithm on the curves in Figures 1a-f, the curves are reasonably well separated both in the MDS_1 and MDS_2 directions for Figures 1m and 1q, which are the two cases where the three classes are quite well separated (in Figures 1a and 1e). Where there is considerable overlap between two classes, as in Figures 1b, 1d and 1f, only the non-overlapping class is well separated from the other two classes by the ISOMAP algorithm, as seen in Figures 1n, 1p and 1r. As indicated earlier in this section, there is not much literature on nonlinear time-varying embeddings, and there seems to be room for more analytic and empirical work.

Sometimes it is recommended to take averages to recover the main structure of a sequence of embedding plots. This has been done in Figures 2a-2f, where Figures 2c and 2e represent embeddings of the average of the point clouds in Figures 1a-1c, and Figures 2d and 2f represent the average of Figures 1d-1f. In this very simple case the principal embedding patterns are recovered. This is especially the case for the kernel principal components in Figure 2d, chiefly due to the better separation of the innermost curves in Figure 1e as compared to Figure 1c. For the ISOMAP algorithm, the classes are quite well separated for both average point clouds, but the forms of the structures appear to be more unstable.

3.3 Forecasting and embedding

In TJL, the time dimension was neglected, and the embedding was primarily motivated by clustering and classification. In time-varying or dynamic problems an added motivation for embedding is forecasting.

In finance, but also in other areas, there is a need to forecast very high dimensional time series. For example, it may be necessary to obtain simultaneous forecasts of a very large collection of time series. Since such time series are often cross- and autocorrelated, one ideally wants to take advantage of the joint history to produce better forecasts. However, use of traditional and classical forecasting methods such as vector autoregressive, GARCH, exponential smoothing and state space modeling (Hyndman et al., 2008; Hyndman and Athanasopoulos, 2018) may not be expected to work well when the dimension radically increases.

One alternative to avoid this problem is simply to make individual univariate predictions for each time series. This means losing all cross dependence information in the joint history of the process.

A compromise is to use embedding of the time series to a manageable dimension, then make a forecast using for instance one of the standard methods for the embedded series and transform back again. This is straightforward if a principal component (or even principal dynamic factor) is used, and the dimension is not so high that the eigenvalue problem creates trouble. Indeed, an advantage of using principal components is that it is easy to transform back to the original time series, since everything is linear. For

[3] We have used the R `ISOMAP` function from the `RDRToolbox` package (Bartenhagen, 2020) with $k = 10$ neighbors and the modified ISOMAP algorithm, which respects nearest and farthest neighbours.

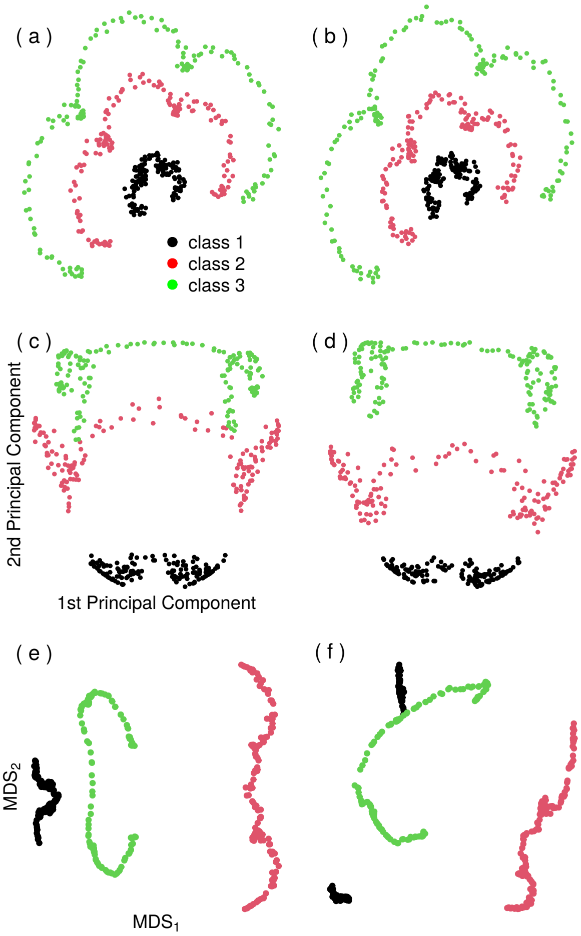

Figure 2: a and b: Average of the first three (a-c) and last three (d-f) time points from Figure 1. c and d: Kernel principal components (with a Bessel kernel) of a and b. e and f: ISOMAP, with MDS_1 and MDS_2 directions, of a and b.

the nonlinear case the back-transformation problem is far more serious. This has been examined by e.g. Papaioannou et al. (2021). They use locally linear embedding (LLE) and diffusion maps as their two embeddings routines, and with radial basis function interpolation and geometric harmonics to create back-transformations.

There are also a number of papers where neural networks methods are used. First, let us consider the linear temporal model by Yu et al. (2016). One reason for selecting this paper is its transparent handling of the regularization mechanism to avoid overfitting of the embedding model. We will use the notation in Nguyen and Quanz (2021) in describing this.

Consider a multivariate time series $X_t,\ t = 1, \ldots, T$ of dimension p, where p may be in the range of 10^3 to 10^6, or even higher. The collection of time series forms a $p \times T$ dimensional matrix $\mathbf{X}$. By using an embedding, this matrix is changed to a $m \times T$ dimensional matrix $\mathbf{Y}$, where m is the embedding dimension. To train a neural network like RNN (Recursive Neural Network, see the Supplement for a definition), data can be batched temporarily. Denote by $\mathbf{X}_B$ a batch of data containing a subset of b samples

$\mathbf{X}_B = \{X_t, \ldots, X_{t+b-1}\}$, where $B = \{t, \ldots, t+b-1\}$ are time indices. Yu et al. (2016), as described by Nguyen and Quanz (2021), perform a constrained linear embedding resulting in $\mathbf{Y}_B$ of dimension $m \times b$ or regularization by solving

$$\min_{\mathbf{Y},\mathbf{F},\mathbf{W}} \left(\mathcal{L}(\mathbf{Y}, \mathbf{F}, \mathbf{W}) = \frac{1}{|\mathcal{B}|} \sum_{B \in \mathcal{B}} \mathcal{L}_B(\mathbf{Y}_B, \mathbf{F}, \mathbf{W}) \right), \tag{5}$$

where $\mathcal{B}$ is the set of all B data batches and each batch loss is defined by

$$\mathcal{L}_B(\mathbf{Y}, \mathbf{F}, \mathbf{W}) = \frac{1}{pb} ||\mathbf{X}_B - \mathbf{F}\mathbf{Y}_B||_{l_2}^2 + \lambda \mathcal{R}(\mathbf{Y}_B, \mathbf{W}), \tag{6}$$

where $\mathbf{X}_B$ and $\mathbf{F}$ are of dimension $p \times b$ and $p \times m$, $\mathcal{R}(\mathbf{Y}_B, \mathbf{W})$ is a regularization of $\mathbf{Y}_B$ parameterized by $\mathbf{W}$ to enforce certain regular properties of the latent terms, and λ is a regularization parameter. The regularity that Yu et al. (2016) impose is to assume that $\{Y_t\}$ follows a vector autoregressive model so that $Y_t = \sum_{j=1}^{L} \mathbf{W}^{(j)} Y_{t-j}$ where L is a predefined lag parameter and $\mathbf{W^{(j)}}$ is a $m \times m$ matrix. Then the regularization becomes

$$\mathcal{R}(\mathbf{Y}_B, \mathbf{W}) \doteq \sum_{l=L+1}^{b} ||Y_l - \sum_{j=1}^{l} \mathbf{W}^{(j)} Y_{l-j}||_{l_2}^2.$$

The optimization is then solved via alternating minimization with respect to the variables $\mathbf{Y}, \mathbf{F}$, and $\mathbf{W}$.

This example illustrates the regularization for a linear model, but essentially the same method can be used for a nonlinear embedding and for a neural network type regularization. Nguyen and Quanz (2021) use autoencoding as an embedding device. The embedded time series process of lower dimension is subsequently forecasted using a neural network LSTM forecaster as outlined in the Supplement.

Again parameters such as the embedding dimension, the lag structure, the regularization parameter and the parameters entering in the LSTM algorithm have to be chosen or estimated. It does not seem to be clear how an optimality theory for this procedure can be established or how sensitive the embedding and the resulting forecast are to these choices.

Until recently, traditional forecasting methods such as ARMA-based and exponential smoothing (McKenzie, 1984), and state-based models have consistently outperformed machine learning methods such as RNNs in large scale forecasting competitions (Makridakis et al., 2020a,b; Crone et al., 2011). A key reason for recent successes of deep learning in forecasting is multitask univariate forecasting, sharing deep learning parameters across all series, possibly with some series-specific scaling factors or parametric model components (Salinas et al., 2019; Smyl, 2020; Bandara et al., 2020; Wen et al., 2017). Indeed, the winner of the M4 forecasting competition of Makridakis et al. (2020b), involving 100 000 time series and 61 forecasting methods, was a hybrid exponential smoothing-RNN model (Smyl, 2020). This is an example of a very successful interaction of machine learning methods with a traditional parametric forecasting modeling. Clearly, this is an inspiration for more of this interaction in other problems like clustering and, later, embedding of networks.

It should also be noted that perhaps too much work in forecasting has been concentrated on point forecasting. As argued by among others Nguyen and Quanz (2021), low-dimensional embedding is probably an important tool to obtain a probability distribution of forecasts.

4 Dynamic topological data analysis

The field of topological data analysis (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 et al. (2002) and Zomordian and Carlsson (2005). Chazal and Michel (2021) give a relatively nontechnical review which is also oriented towards statistics, and a short summary can be found in Section 4.2 of TJL. We include a few main points from that paper in the following.

For our purposes of statistical embedding, TDA brings in some new aspects in that topological properties are emphasized and can potentially be used as new characterizations of the data cloud. An important device is the so-called persistence diagram which depicts the persistence, or lack thereof, of certain topological features as the scale in describing data cloud changes.

Assume that n data points $X_1, \dots, X_n$ are at or close to a smooth compact submanifold S. One may estimate S by trying to cover the data cloud by a collection of balls of radius ε, such that

$$\hat{S} = \cup_{i=1}^{n} B(X_i, \varepsilon), \tag{7}$$

where $B(X_i, \varepsilon) = \{x : ||x - X_i|| \leq \varepsilon\}$. 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 $\cup_{i=1}^{n} B(X_i, \varepsilon)$ will consist of n distinct connected sets (of so-called homology zero). But as the radius of the balls increases, some of the balls will have non-zero intersection, and the number of connected sets will decrease. For ε big enough one can easily imagine that the set $\cup_{i=1}^{n} B(X_i, \varepsilon)$ 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 (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. For the circle example mentioned above, the birth and death of the hole will be well above the diagonal, and it has a time of death which may be considerably larger than its time of birth.

In TJL we have gone through this process for the data set of the three chained Ranunculoids displayed in Figure 1a of the present paper, and the reader is referred to Figures 3 and 4 of that paper for a description of the persistence diagram for two levels of noise.

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, $\beta_0, \beta_1, \dots$. 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 $\beta_0 = 1, \beta_1 = 2, \beta_2 = 1$.

4.1 Forms of dynamics in TDA

The simplest form of investigating the dynamics is just to analyze a collection of high dimensional time series, say, by taking snapshots at several time points and find the persistence diagram for the data cloud of each snapshot. In this series of persistence diagrams one may look for key features like type of periodicities that may be difficult to obtain by other means. As mentioned, periodicities is a main feature of several physiological processes, for instance in the brain, see e.g. Chung et al. (2022).

Time changes in the persistence diagram can be illustrated by the previous Ranunculoid example. This has been depicted in Figure 3 for the same periodic time variation of the Ranunculoid curves as in Figure 1. The Ranunculoid plots of Figures 3a-3f correspond to the plots in Figures 1a-1f. Here, Figure 1a, the static case, corresponds to Figure 4a in TJL (using different seeds though). The persistence diagrams in the present paper for the static case in Figure 3g for classes (1,2) and in Figure 3m for classes (2,3) and in Figure 3s for classes (1,2,3) correspond to Figures 4e, 4f and 4h of TJL.

The gray points represent sets of homology zero (isolated sets), and the black points represent sets of homology one; i.e. one dimensional holes. The gray columns at the left is just the time of death for all the sets around the individual points as the radius ε of expression (7) for the individual points increases. Naturally, for the static picture the gray column for classes (2,3) reaches higher than for classes (1,2) since the distance between classes 2 and 3 is larger. The black points at the right of the gray column represent small holes that form as ε increases. They are due to indents in the point spreads of the Ranunculoids. If the Ranunculoids were to be replaced by noiseless circles having the same center they would disappear. By comparing the diagrams in Figures 3g and Figure 3m, it is seen that the holes have larger lifetimes for the (2,3) class, which is natural in view of the larger indents in curve 3 in Figure 3a.

The temporal patterns of the persistence diagrams that appear as time progresses are quite intuitive. For example, corresponding to the proximity of curves 2 and 3, and the larger distance between curves 1 and 2 in Figure 3b and Figure 3f, there are few holes of any length of lifetime in diagrams 3n and 3r, whereas there are three holes of sizable lifetime in Figures 3h and 3l, corresponding to the three main indents of curve 2 for the (1,2) diagram. As the radii ε increase, the two curves coalesce, and we have a death at the gray point above the gray column in the (1,2) diagram in Figures 3h and 3l.

For the case of Figure 3d where the curves 1 and 2 are very close, the situation is reversed as can be seen from Figures 3j and 3p. The other intermediate situations in Figures 3c and 3e give intermediate persistence diagrams. Finally, the case of (1,2,3) taken simultaneously, because of the relative simple situation of Figure 3, the persistence diagrams are virtually superpositions of the diagrams for (1,2) and (2,3).

The point of all this is that the time variation of the persistence diagrams reveals temporary topological variation which is not apparent from the nonlinear plots associated with the methods of Section 3, as it is partially revealed in Figures 1 and 2.

To compare features originated from specific persistence diagrams, a distance measure between such diagrams is needed. Several such distance measures exist; perhaps

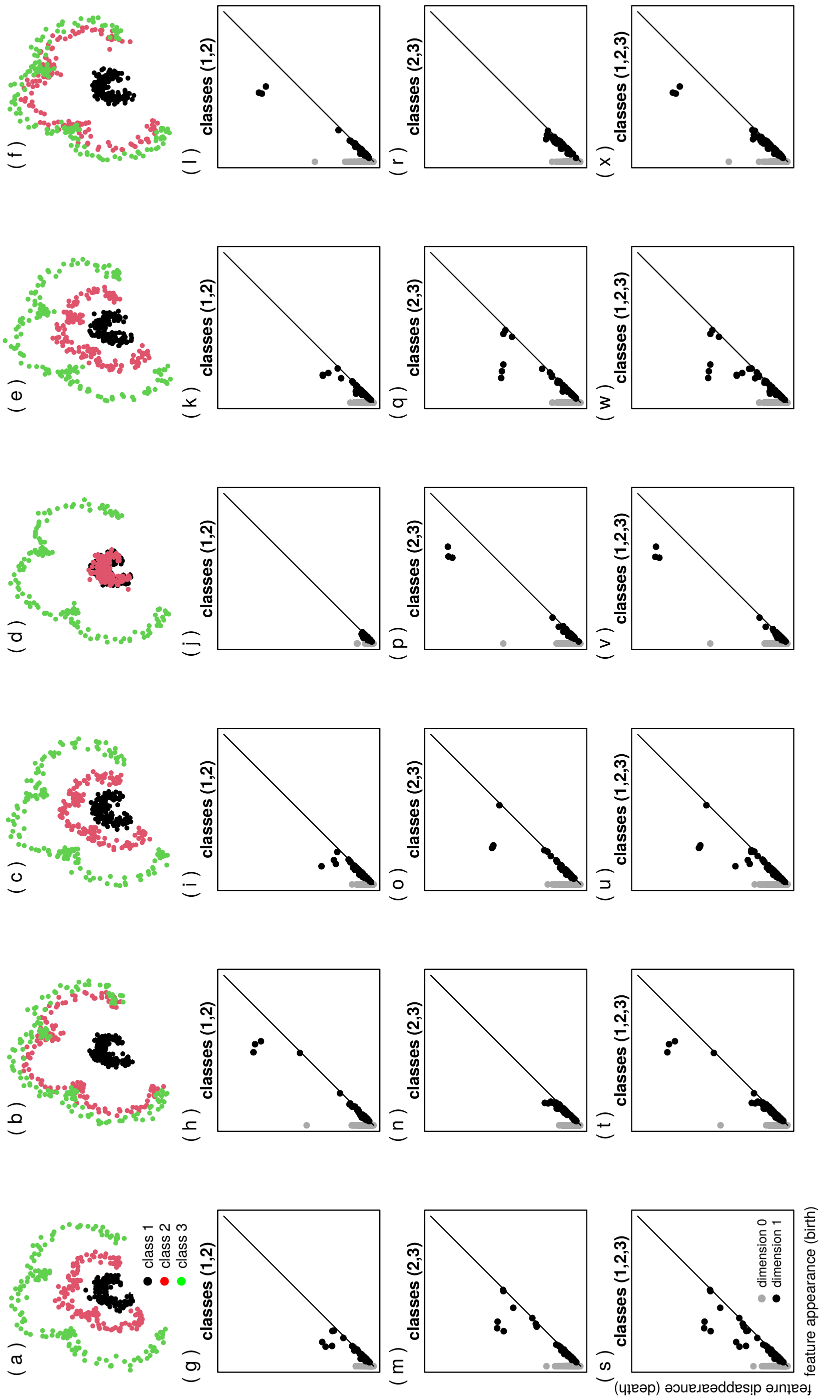

Figure 3: a-f: These correspond to the plots in Figure 1. g-x: Persistence diagrams for classes (1,2), (2,3) and (1,2,3) for a-f.

the most well-known is the bottleneck distance. Given two diagrams C_1 and C_2, the bottleneck distance is defined by

$$\delta_\infty(C_1, C_2) = \inf_\gamma \sup_{z \in C_1} ||z - \gamma(z)||_\infty,$$

where γ ranges over all bijections between C_1 and C_2. Intuitively, this is like overlaying the two diagrams and asking how much one has to shift the diagrams to make them the same (Wasserman, 2018). The practical computation of the bottleneck distance amounts to the computation of perfect matching in a bipartite graph for which classical algorithms can be used (Chazal and Michel, 2017).

An alternative to the bottleneck distance is the Wasserstein distance given by

$$\delta_p(C_1, C_2) = \inf_\gamma \sum_{z \in C_1} ||z - \gamma(z)||_p.$$

A further development of the persistence diagram is to use the so-called persistent landscapes (Bubenik, 2015), which has the advantage that it is a function space. The bottleneck distance is also a natural tool in statistical inference on persistent landscapes, cf. Chazal et al. (2015). Gidea and Katz (2018) have used persistent landscapes in an examination of financial time series.

The snapshot procedure is a primitive procedure, that although it can illustrate non-stationary variations of a point cloud, it does not really attempt to model the individual time series whose simultaneous observations create the point cloud. A quite different approach that tries to take individual time series structure into account, is using the Takens' embedding procedure of a time series.

Consider a (one-dimensional) time series $\{X_t, t = 1, \ldots, T\}$. Takens' embeddings procedure can be used to convert the time series into a point cloud with points $v_i = \{X_i, X_{i+\tau}, \ldots, X_{i+(d-1)\tau}\}$, where d specifies the dimension of the embedding frame of the points and τ a delay parameter. Taking $d = 2$ and $\tau = 1$ yields a point cloud in the plane with $v_i = \{X_i, X_{i+1}\}$. This scatterdiagram in the plane is often used to illustrate the dynamic properties of a time series, and has in particular been used to illustrate limit cycles for nonlinear time series (Tong, 1990). In general both d and τ will have to be determined in practice. Different practices have been followed here. The delay parameter τ may be selected as the smallest time lag where the sample autocorrelation becomes insignificant, using an e.g. critical bound $2/\sqrt{T}$ corresponding approximately to a 95% confidence interval. The most used algorithm for determining d is probably the so-called false neighborhood method. The embedding dimension is then determined as the integer such that the nearest neighbor of each point in dimension d remains the nearest neighbor in dimension $d + 1$, and the distances between them remain approximately the same. Perea and Harer (2015) have suggested the use of $d = 15$ on time series after a cubic spline interpolation, which allows handling of unevenly spaced time series.

It is not straightforward to generalize this approach to multivariate time series, but see Gidea and Katz (2018), and Bourakna et al. (2022) which has its main focus on brain wave data collected over different channels. An article aimed directly to a statistical audience is Wang et al. (2018).

In topological analysis of brain data, the data of the various channels are usually modeled as a brain network. For dynamically changing brain networks it is assumed that the node sets (channels) are fixed, while weights on links may be changing in time.

If one builds persistent homology at each fixed time, the resulting persistence diagrams are also time dependent.

Topological analysis with persistence diagrams has also been based on functions, which may be thought of as signals in continuous time. When data is in the form of a continuous function $f : \mathbb{R}^d \to \mathbb{R}$ or more generally for a real-valued function on a manifold, one may then use sublevel set filtrations carried out by discretizations defined by the sublevel data sets

$$L_\lambda(f) = \{z : f(z) \leq \lambda, z \in \mathbb{R}^d\} \tag{8}$$

for a lattice $z = (k_1\delta, k_2\delta, \ldots, k_d\delta)$ for a given δ and sets of constants $(k_1, k_2, \ldots, k_d)$, and where $0 \leq \max_z f(z)$. This define a simplex as a set of components in $L_\lambda(f)$, which are neighbors, i.e. $z_1, z_2 \in L_\lambda(f)$, and $|z_{1,j} - z_{2,j}| \leq \delta, j = 1, \ldots, d$, and as a next step so-called simplical complexes.

Actually, here, and in more complicated situations, the persistence diagram is not computed directly from scale shifts as in the expression (7) but from simplical complexes. This approach is particularly interesting since it generalizes the embedding of a point cloud in a graph as will be treated in the dynamic case in the next section. We give a brief description in the Supplement. Much more details can be found in Chazal and Michel (2021). In case $\mathbb{R}$ is replaced by a manifold the sublevel construction can still be undertaken, and can be seen as a part of Morse theory applied to topological analysis, see e.g. Adams et al. (2011).

The sublevel construction in (8) and the subsequent persistence diagram construction can also be used to look for periodic topological structures. The function f in (8) may be thought of as a Fourier transform for continuous-valued time series or e.g. as a Walsh Fourier transform for a categorical time series, see Ravisshanker and Chen (2019) for examples. Conceivably one could also use nonlinear spectral curves as in Jordanger and Tjøstheim (2022) to look for topological features not revealed by the ordinary power spectrum.

There are a number of problems of interest for statisticians in general and for time series analysts in particular in TDA. Chazal and Michel (2021) mention four topics in the static case, but with self-evident interest in the dynamic TDA case:

1. Proving consistency and studying the convergence rates of TDA methods.
2. Providing confidence regions for topological features and discussing the significance of estimated topological quantities.
3. Selecting relevant scales (i.e. selecting ε in the examples discussed above and in the Supplement) at which topological phenomena should be considered as functions of observed data.
4. Dealing with outliers and providing robust methods for TDA.

Concerning the last point there may be an unexplored connection between the graph representation of topological features and graph anomaly detection as detailed in e.g. Ma et al. (2021). Chazal and Michel (2021) have used the bootstrap in a static TDA situation. One may want to introduce the block bootstrap to take better care of dependence structures. There are also recent contributions to hypothesis testing, Moon and Lazar (2020), sufficient statistics, Curry et al. (2018), and Bayesian statistics for topological data analysis, Maroulas et al. (2020).

It has been seen that construction of neighborhood graphs and generalization of these are important tools in TDA and elsewhere. In the next section we look at the situation where the data are given in terms of a graph and where time variations are included.

5 Dynamic graphs

In the preceding sections we have seen how graphs can be useful tools in nonlinear embeddings of a point cloud, and in TDA in handling of a point cloud when using simplical complexes (cf. the Supplement for more details). In the present section it is assumed that the data themselves are given in the form of a network. With the increasing use of the internet and big data, analysis of large networks is becoming more and more important There is a wide field of applications ranging over such diverse areas as e.g. finance, medicine and sociology, including criminal networks. A broad overview can be found in the recent book by Newman (2020). More foundational problems are covered in Crane (2018).

5.1 The static case

Both research and applications have been overwhelmingly concentrated on static networks. But a change is presently taking place, since the static assumption in many cases is an untenable one. In many types of networks, as time goes on, new nodes are added to the network, others are vanishing, and the strength of the connection between nodes are changing, or may even be severed. This realization has led to a rapid recent increase in the interest for dynamic networks. In this paper we will only be able to give a glimpse of this development, but it harbors several open and exciting problems for time series analysts. To put this into context, however, we first need to give a very brief overview of methods for static networks. This is from TJL and Kazemi et al. (2020).

By embedding a network in Euclidean space $\mathbb{R}^m$ or on a submanifold in $\mathbb{R}^m$, the nodes of the network are represented by vectors on which further analysis like clustering and classification can be undertaken. From our point of view, two methods stand out as particularly interesting: Spectral embedding and embedding via the so-called Skip-Gram neural network method.

Spectral embedding is motivated by the clustering problem where clusters form communities in the network. The problem is to find these communities. This is done by minimizing a cut functional or maximizing a modularity functional. In both cases the minimizing/maximizing leads to an eigenvalue problem analogous to the Laplace eigenvalue problem mentioned in Section 3.1. A faster solution of the modularity problem is obtained by the so-called Louvain method for community detection.

It is computationally costly to solve an eigenvalue problem for high dimensions, and networks often have ultra-high dimensions. These problems are to a large degree alleviated in the neural network-based Skip-Gram procedure. Here the eigenvalue problem is removed altogether, and the neural net training is speeded up by so-called negative sampling. The neural net in Skip-Gram has only one hidden layer. The idea is for each node to associate neighboring fragments of the network by performing random walks governed by the weights on the links of the neighboring nodes to the given node. These fragments are fed through the hidden units and the optimal linear combinations are learned by requiring essential conservation of the fragments as they are passed through the hidden units. The dimension of the hidden layer in the network is much lower (perhaps in the range from 500-600 or so). The output linear combinations are taken as representations (or embeddings) of the nodes.

It has been found that the method works rather well in practice. The basic publications (Mikolov et al., 2013a,b) have well over 20 000 citations. There are several versions of the method, and as will be seen, there is also a dynamic one, to be mentioned shortly. For more details and references the reader is referred to TJL.

The neural network approach has a natural extension to networks with several layers. We refer to the Supplement for a brief description of these, including autoencoding, convolution networks and recurrent networks, mainly from a prediction point of view, but they can equally be used in a static framework for embedding, clustering and classification.

The methods presented in this section can all be characterized as machine learning methods. There is really no statistical model involved. Nevertheless these procedures contain parameters or hyper parameters that have to be determined. The performance of the methods may depend quite critically on the choice of these parameters (Peixito, 2021). They can be determined by optimizing an object type function in some cases, but in other situations one has resorted to more trial and error procedures.

In an attempt to counter some of these problems, a more traditional statistical model has been introduced for community finding in networks. This is the stochastic block model (SBM). In the simplest undirected stochastic block model each of the 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. This results in k^2 probability parameters for the model, and there are several ways of estimating them for a given real data network. There are a number of papers in which asymptotic distributional properties and consistency are developed (see e.g. Zhang and Chen (2020) and references therein). Unfortunately, however, the simple SBM does not work well for many real world networks. One relatively simple generalization is the degree corrected stochastic block model (dcSBM).

5.2 The dynamic case

If we have a sequence of networks in time $\{(G_t, V_t), t = 1, \ldots, T\}$, where G_t denotes the nodes at time t, and V_t the links at this time, the simplest procedure to analyze dynamics is to obtain a snapshot embedding for each time point t. If we just need one embedding for the entire time period, this can be simply obtained by taking averages of the adjacency matrices and finding the embedding corresponding to this average. Alternatively, but much more time consuming, one may find an embedding separately for each time point t, and then take the average of the embedding vectors (cf. Figure 2). If one wishes to give more weight to the most recent observations, this can be obtained by taking a weighted average.

However, there are a number of problems with this approach in addition to the fact that it may be close to impossible to carry out because of the time needed to produce each snapshot for large networks. It turns out that the embeddings we have mentioned so far, perhaps especially the Skip-Gram procedure, may be rather unstable in the sense that a relatively small change in the network from one time point to another may produce large changes in the embedded vectors, even for vector representation for nodes lying at a considerable distance from where the main changes are taking place, see the illustrations in Section 5.3, in particular Figure 5. Regularization analogously to the formulas (5) and (6) has been suggested, but where the added penalty term penalizes big changes in time for the embedding vectors for the nodes.

For the spectral embedding approach it has been proposed that small perturbations could be handled by Taylor expansion. If the Laplacian matrix **L** and the degree matrix **D** are changed by small amounts, updated eigenvalues and eigenvectors can be obtained. The Davis-Kahan theorem (cf. Yu et al. (2015)) gives an approximation error for the top m eigenpairs. An alternative approach in the spectral representation is to stack the adjacency matrices $\mathbf{A}_1, \dots, A_T$ into an order 3 tensor (Dunlavy et al., 2011). Such embeddings can be used to make predictions of links.

Networks in continuous time driven by differential equations have been used to explore spectral theory of brain oscillations in Raj et al. (2020).

The relationship between dynamic graphs, spectral theory and TDA is examined further in Bronski and DeVille (2014) and in Xu et al. (2020). Relationships with time series networks generated by topological data analysis is treated in Varley and Sporns (2022).

For the case of Skip-Gram, more precisely for the so-called LINE-version, Du et al. (2018) propose a decomposition procedure, where new nodes can be taken into the objective function in a separate part. The objective function for learning the embedding of nodes is given by

$$max_{u,c} \sum_{(i,j)} w_{ij} (\log \sigma(c_j \cdot u_i) + k E_{v_n \sim P_n(v)} [\log \sigma(-c_n \cdot u_i)]).$$

Here w_{ij} is the weight function between node i and j, u and c are vector embedding representations of a node v, when it is treated as a central node and specific successor node, respectively. The parameter $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = 1/(1+e^{-x})$, and P_n is the noise distribution over n nodes for the so-called negative sampling procedure with k negative samples. For details, see Du et al. (2018) in the dynamic case and TJL (Section 5.3) in the static case. It is shown in Du et al. (2018) that the objective function can be decomposed so that a group of nodes can successively be kept fixed, and that nodes can be added in three different ways. This allows for successive time-varying embedding of nodes in such a way that only the neighbors of new nodes needs to be updated. A criterion for which nodes with accompanying weights should be updated is given. In this way the processing time is considerably reduced as compared to do a full training update of the entire graph at selected time intervals. It is claimed in the paper that taking nodes away can be treated in an analogous manner. The method is illustrated on some social networks.

The Skip-Gram analysis is mentioned as one possible method in a relatively large survey article on representation (embedding) learning for dynamic graphs by Kazemi et al. (2020). Both the static and the dynamic case are covered in this paper. In particular existing models are examined from an encoder-decoder perspective. There are a number of useful references.

All of these approaches are mainly algorithmic in nature, essentially extensions of machine learning algorithms used in the static case. How about the SBM model that was heralded as a contrast to the algorithmic methods in the static case? Is there a dynamic theory for it? Compared to the static case it seems to be quite limited. Xu and Hero (2014) transform the entries θ_{ij} in the $k \times k$ matrix of governing probabilities of the transitions between the k blocks, to continuous variables on a different scale by the logit transformation $\psi_{ij} = \log(\theta_{ij}) - \log(1-\theta_{ij})$. Next, these are made time dependent, a Gaussian noise is introduced and the matrix Ψ or rather the vectorized ψ_t is assumed to follow a Kalman state space system with state equation

$$\psi_t = \mathbf{F}_t \psi_{t-1} + v_t$$

where $\mathbf{F}_t$ is a matrix function of time-varying coefficients (often assumed known), and $\{v_t\}$ is a Gaussian noise process.

Ludkin et al. (2018) have an interesting approach where edges are switched on or off according to a hidden Markov chain. The starting point is the SBM model described at the end of Section 5.1. The k static probabilities for transition for one block to another is allowed to be time-varying, and it is allowed that a node may shift to another block. More precisely, let $C_i(\cdot)$ denote the community membership process for node i. It is assumed that regardless of the block that node i belongs to, the node spends an exponential time in this block before moving to another block randomly chosen among the $k-1$ other blocks. This makes $C_i(t)$ into a continuous time Markov chain. Moreover, the authors introduce a simple edge dynamics. It should be noted that only simple SBM models are made dynamic, not the more useful dcSBM model.

5.3 Dynamic SBM illustration

In Section 6.5 of TJL we illustrated the static SBM model on a simulated system containing two communities using various configurations, and studied to what extent different visualization methods that were put on top of a simple graph embedding managed to discriminate between the two communities. It is of interest, we think, to extend this illustration to a dynamic case.

In the present illustration, we start out with a graph where each node is labeled either red or green. This graph then evolves dynamically as nodes are removed one by one. At each stage of the graph reduction, we embed the current version of the graph in a (64-dimensional) Euclidean space, using the Skip-Gram algorithm node2vec, and then apply three different visualization methods (PCA, t-SNE and UMAP). Once this is done for all stages, we study and discuss how the visualizations evolves over time, and how well the visualizations preserve the discrimination between the node classes (red and green), of which it is completely unaware. The purpose of this is to get a feeling of the stability of the different visualization methods both visually and in terms of their ability to discriminate between the two classes.

The graph we start out with is a combination of three subgraphs sampled from SBM models, and is displayed in the left panel of Figure 4. The three subgraphs each have 20 nodes and stem from the following following SBM combinations:

Graph a: Average node degree $d = 3$, ratio of between-block edges over within-block edges $\beta = 0.2$

Graph b: Average node degree $d = 2$, ratio of between-block edges over within-block edges $\beta = 0.4$

Graph c: Average node degree $d = 2$, ratio of between-block edges over within-block edges $\beta = 0.2$, and an unbalanced community proportion; a probability of $3/4$ for label 1 (red) and a probability of $1/4$ for label 2 (green).

To connect these three subgraphs into a single starting graph, we add edges between 6 random pairs of nodes from the different subgraphs that have the same label. Finally,

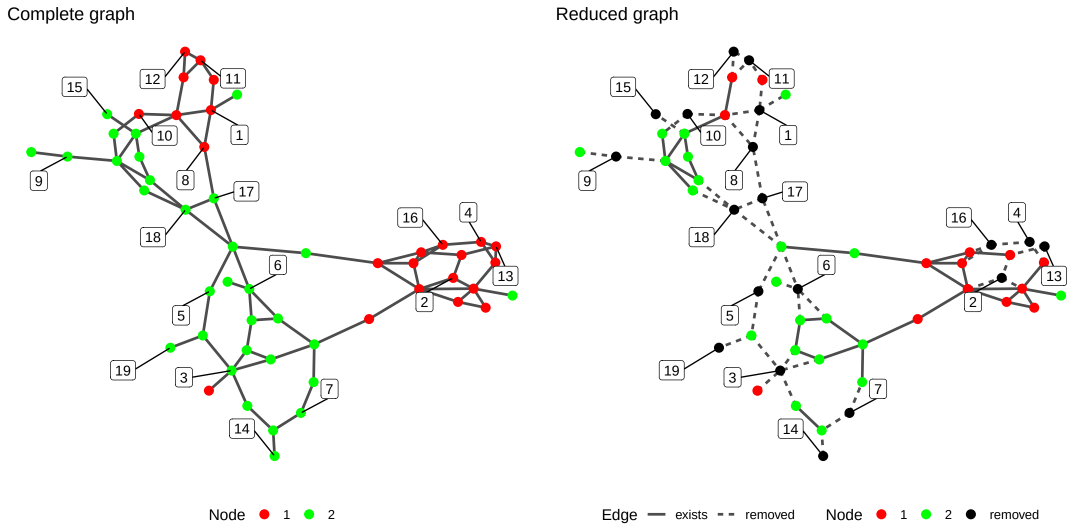

Figure 4: Illustration of graph used in the dynamic embedding illustration. The left panel shows the complete graph before reduction. The right panel shows the graph in the 20th stage, after removal of 19 nodes (and their associated edges). The colors indicate each node's community. Black nodes and dashed edges indicates, respectively, removed nodes and edges. The numbers indicate order in which the nodes are removed.

any nodes without edges are removed from the graph. This graph construction process gave us the starting stage of the dynamic graph which is displayed in the left panel of Figure 4, with 54 nodes (23 red and 31 green). As mentioned, in each stage of the dynamic graph, one random node is removed. The right panel of Figure 4 shows the last stage (20th stage) where 19 nodes have been randomly removed. In that figure, the removed nodes are marked by black dots, and numbered according to the iteration in which they are removed. The associated removed edges are marked as dashed lines. The removal of nodes eventually resulted in a non-connected graph.

In the node2vec-embedding process which is carried out at each stage, we have used an embedding dimension of $m = 64$, with $L = 30$ nodes in each random walk, $\gamma = 200$ walks per node, and a word2vec window length of $K = 10$ where all nodes are included (cf. Section 5.3.2 and 6.5 of TJL).

In a succeeding visualization step, the 64-dimensional embedding is reduced to a 2-dimensional vector with three different methods: PCA, t-SNE and UMAP. For t-SNE, we show results with a perplexity parameter of both $p = 5, 10$, while for UMAP, we show results with three different combinations of the number of neighboring sample points (n), and the effective minimum distance between embedded points (m): $(n = 25, m = 0.01); (n = 5, m = 0.01); (n = 5, m = 0.75)$. PCA has been computed with the R-function `stats::prcomp`, t-SNE has been computed with `Rtsne::Rtsne` and UMAP using `uwot::umap`. All of these have been used with default parameters unless otherwise mentioned. More details of these embedding/visualization routines are given in Sections 6.1 and 6.3 of TJL (the LargeVis algorithm is omitted here to obtain a simpler presentation).

Figure 5 gives the embedding structure for every second stage in the dynamic graph (reduction process), where the x- and y-axis of the visualization approaches have been standardized (mean zero and standard deviation 1) to ease comparison. The uppermost row of this figure displays the embeddings for the complete graph in the left panel of Figure 4. It is seen that the group structure of Figure 4 is well taken care for most methods, but the shape of the embedding (as expected) vary considerably from one method to another. The problem mentioned in the second paragraph of Section 5.2 is obvious in Figure 5. See for example the change in embeddings for PCA where the locations of red and green dots move around quite a bit as nodes are removed. Despite this behavior, the internal community structure does not change that much. Such a lack of directional invariance appears to a smaller or lesser degree for all of the methods. Separate experiments also demonstrates that such directional changes can occur when different seeds are used for each iteration. Clearly one has to be aware of this fact to avoid misinterpretation of visualization plots.

If the relative structure of groups of red and green dots in the sequence of embeddings is more or less invariant, then the hope is that the classification scores can still be meaningful and relatively stable from one iteration to another. This is confirmed by Figure 6, showing the classification scores as a function of time for two basic classification algorithms based on k-nearest neighbors: The upper panel of Figure 6 shows the classification scores when the class of a node is determined using the average of the 5 nearest neighbors; in the lower panel of the figure, the class is determined by the majority vote among the 5 nearest neighbors. For reference, the curve identified by "original embedding" gives the classification results for the 64-dimensional Skip-Gram node2vec-embedding (i.e. without any 2-dimensional reduction).

Both panels in Figure 6 indicate that PCA are clearly inferior to the others in the initial stages, but does well for higher iterations. The scores are quite stable as a function of time except for a possible jump at the removal of the 11th node after which the classification scores steadily decrease. The 11th node can be identified in Figure 4 and it is presumably strategically located in the upper group of the red nodes. The main pattern is intuitively reasonable and, noteworthy, it seems to be largely independent of any changes in orientation of the embeddings. The low score of the PCA in the initial stages is apparent also from the embeddings plot in Figure 5. In particular, see the embedding with 6 nodes removed (row 4), which mirrors the low classification scores for PCA for iteration 6. In both Figure 6a and 6b, PCA has a very low score at iteration 5. Inspecting the corresponding embedding plot (iteration 5 not shown in Figure 5) it is seen that this is not due to directional instability but simply that PCA at this iteration produces an embedding with much overlap, which in turn makes it harder to classify. To some degree this is present also for iteration 6.

Clearly one cannot draw general conclusions based on this special example, but the example does illustrate some of the problems that one can be expected to encounter in a more general dynamic situation. There is a need both for more experiments and for more theory.

5.4 Statistical modeling of dynamic networks

What is perhaps the largest difference between the dynamic systems we have been treating in this paper and an ordinary time series dynamic situation is the absence in dynamic networks of an explicit recursive system like the one that is present in the autoregres-

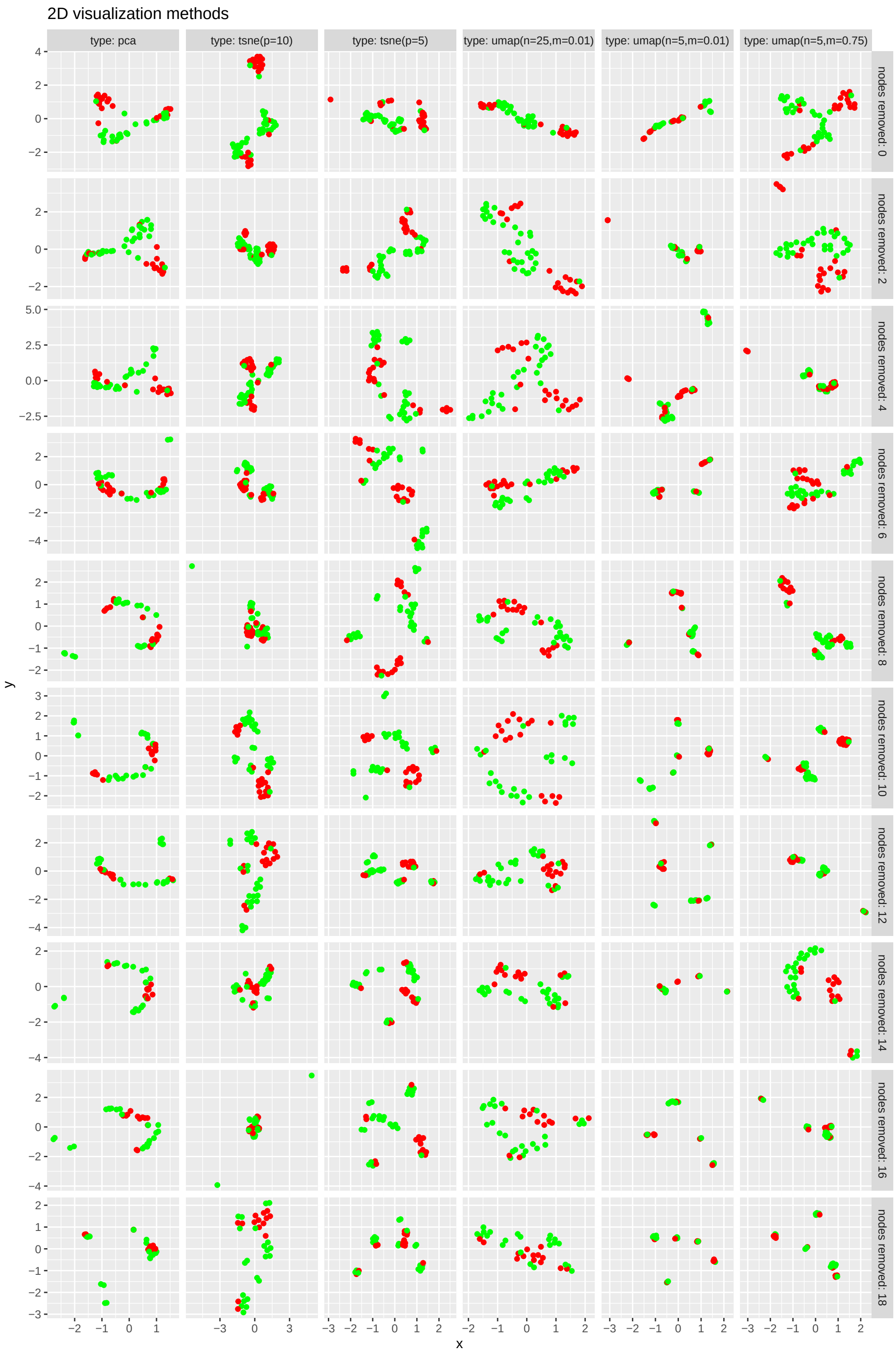

Figure 5: Different visualizations of every second stage of the node2vec embeddings in the dynamic embedding illustration.

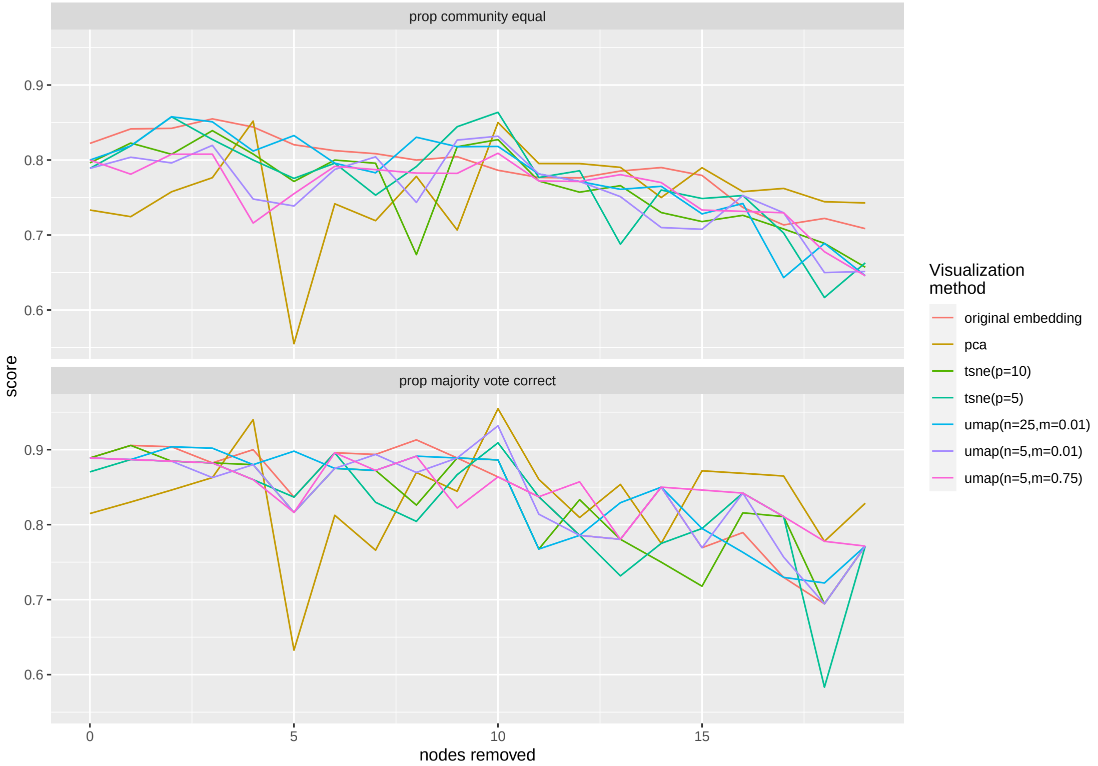

Figure 6: Classification scores plotted as a function of stage/node removal in the dynamic embedding illustration. The classification scores are based on a k-nearest neighbors algorithm with $k = 5$.

sive model (possibly nonlinear and multivariate). Lately there have been some models tending in this direction (see also TJL).

A recent example of rigorous statistical modeling of a dynamic network is Zhu et al. (2017). They model the network structure by a network vector autoregressive model. This model assumes that the response of each node at a given time point is a linear combination of (a) its previous value, (b) the average of connected neighbors, (c) a set of node-specific covariates and (d) independent noise. More precisely, if n is the number of nodes, let Y_{it} be the response collected from the ith subject (node) at time t. Further, assume that a q dimensional node-specific random vector $Z_i = (Z_{i1}, \ldots, Z_{iq})^T \in \mathbb{R}^q$ can be observed. Then the model for Y_{it} is given by

$$Y_{it} = \beta_0 + Z_i^T \gamma + \beta_1 n_i^{-1} \sum_{j=1}^{n} a_{ij} Y_{j,t-1} + \beta_2 Y_{i,t-1} + \varepsilon_{it}. \tag{9}$$

Here, $n_i = \sum_{j \neq i} a_{ij}$, $a_{ii} = 0$, is the total number of neighbors of the node v_i associated with Y_i, so it is the degree of v_i. The term $\beta_0 + Z_i^T \gamma$ is the impact of covariates on node v_i, whereas $n_i^{-1} \sum_{j=1}^{n} a_{ij} Y_{j,t-1}$ is the average impact from the neighbors of v_i. The term $\beta_2 Y_{i,t-1}$ is the standard autoregressive impact. Finally the error terms $\{\varepsilon_{it}\}$ are assumed to be independent of the covariates and iid normally distributed.

Given this framework, conditions for stationarity are obtained, and least squares estimates of parameters are derived and their asymptotic distribution found.

The authors give an example analyzing a Sina Weibo data set, which is the largest twitter-like social medium in China. The data set contains weekly observations of $n =$ 2,982 active followers of an official Weibo account. An extension of the model (9) is contained in Zhu and Pan (2020).

There are a number of differences between the network vector autoregression modeled by equation (9) and the dynamic network embeddings treated earlier in this section. First of all, (9) treats the dynamics of the nodes themselves and not of an embedding. Even if the autoregressive model do introduce some (stationary) dynamics in time, the parameters are static; i.e. no new nodes are allowed, and the relationship between them is also static as modeled by the matrix $\mathbf{A} = \{a_{ij}\}$. From this point of view, as the authors are fully aware of, the model (9) 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 similar to Ludkin et al. (2018), but in the context of the dcSBM model). The hope is that it will be amenable to statistical inference. Krampe (2019) treats dynamic networks with a fixed number of nodes, but where the dynamic structure is modeled by a doubly stochastic matrix.

For some very recent contributions to network autoregression, see Armillotta et al. (2022) and references therein. Armillotta and Fokianos (2022a) consider integer valued network variables and analyze linear and log linear Poisson autoregressive networks. This is motivated by the fact that many net variables take discrete values. They consider fixed and increasing network dimension and show that quasi-likelihood inference provides consistent and asymptotically normally distributed estimators. In Armillotta and Fokianos (2022b) nonlinear models and tests for linearity are introduced. Moreover, consistency and asymptotic normality of the quasi-likelihood estimator are established for continuous and count nonlinear network autoregressions under quite standard smoothness conditions.

5.5 Scalability

When going from the static to the dynamic setting, scalability of algorithms can be of paramount importance. We will therefore in the following briefly comment on how two important classes of methods can deal successfully with larger problems.

Traditionally, PCA has been seen as a method that does not scale well, since the eigenvalue problem may be cumbersome and time consuming to solve. The complexity of calculating all eigenvalues of a $p \times p$ matrix is $O(p^3)$. However, one may typically only be interested in relatively few of the first principal components, and advances in matrix algorithms (Lehoucq and Sorensen, 1996) has led to software such as `RSpectra` (Qiu and Mei, 2022) that scales well for large problems.

Since the emergence of the neural network-based Skip-Gram approach, many efficient algorithms have been implemented, such as DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), node2vec (Grover and Leskovec, 2016), NetMF (Qiu et al., 2018) and ProNE (Zhang et al., 2019). For example, Zhang et al. report that it takes ProNE only about 29 hours to embed a network of 0.1 billion nodes and 0.5 billion edges by using one thread, while it takes LINE over one week and may take DeepWalk/node2vec several months by using 20 threads. The efficiency of ProNE is linearly correlated with network density.

5.6 Concluding remarks

We have given a survey of embeddings of time series and dynamic networks. We have covered dynamic factors for time series, and dynamic versions of nonlinear embeddings, topological data analysis embeddings, and network embeddings. The embeddings have been illustrated by two groups of simulated examples. The differences between a more or less purely algorithmic approach and an approach based on more statistical modeling have been pointed out, and we have seen that an algorithmic approach is clearly dominating. The literature on dynamic embeddings is much more sparse than the static case. This holds even though the dynamic embeddings could be far more realistic for many practical cases. Throughout the review, we have pointed out a number of open modeling problems. We encourage time series analysts in particular, but also more general mathematical statisticians, to try to deal with these.

Acknowledgments

The authors would like to thank two anonymous referees for very helpful comments that improved the quality of this paper. This work was supported by the Norwegian Research Council grant 237718 (BigInsight).

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Adams, H., Atanasov, A., and Carlsson, G. (2011). Morse theory in topological data analysis. arXiv preprint arXiv:1112.1993.

Armillotta, M. and Fokianos, K. (2022a). Poisson network autoregression. arXiv:2104.06296v3.

Armillotta, M. and Fokianos, K. (2022b). Testing linearity for network autoregressive models. arXiv:2202.03852v1.

Armillotta, M., Fokianos, K., and Krikidis, I. (2022). Generalized Linear Models Network Autoregression. In *International Conference on Network Science*, pages 112–125. Springer.

Bandara, K., Bermeir, C., and Hewamalage, H. (2020). LSTM-MSNet: Leveraging forecasts on sets of related time series with multiple seasonal patterns. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14.

Bartenhagen, C. (2020). *RDRToolbox: A package for nonlinear dimension reduction with Isomap and LLE.* R package version 1.40.0.

Bourakna, A., Chung, M., and Ombao, H. (2022). Topological data analysis for multivariate time series data. arXiv:2204.13799v1.

Brillinger, D. (1969). The canonical analysis of stationary time series. In *Multivariate Analysis II*, pages 331–350. New York: Academic. Editor P.R. Krishnaiah.

Brillinger, D. (1975). *Time Series. Data Analysis and Theory.* Holt, Rinehart and Winston.

Bronski, J. C. and DeVille, L. (2014). Spectral theory for dynamics on graphs containing attractive and repulsive interactions. *SIAM Journal on Applied Mathematics*, 74(1):83–105.

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16:77–102.

Chamberlain, G. (1983). Funds, factors, and diversification in arbirage pricing models. *Ecometrica*, 51:1281–1304.

Chamberlain, G. and Rotschild, M. (1983). Arbitrage, factor structure and mean-variance analysis in large asset markets. *Econometrica*, 51:1305–1324.

Chazal, F., Fasy, B., Lecci, F., Rinaldo, A., and Wasserman, L. (2015). Stochastic convergence of persistence landscapes and silhouettes. *Journal of Computational Geometry*, 6:140–161.

Chazal, F. and Michel, B. (2017). An introduction to topological data analysis: fundamental and practical aspects for data scientists. arXiv: 1710.04019v1.

Chazal, F. and Michel, B. (2021). An introduction to topological data analysis: fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence: Machine Learning and Artificial Intelligence*, 4:1–28.

Chen, R., Yang, D., and Zhang, C.-H. (2022). Factor models for high-dimensional tensor time series. *Journal of the American Statistical Association*, 117:94–116.

Chung, M., Huang, S.-G., Caroll, I., Calhaun, V., and Goldsmith, H. (2022). Dynamic topological data analysis for brain networks vaia wasserstein graph clustering. arXiv:2201.00087v2.

Crane, H. (2018). *Probabilistic Foundations of Statistical Network Analysis.* Chapman and Hall.

Crone, S., Hibon, M., and Nikolopoulos, K. (2011). Advances in forecasting with neural networks: Empirical evidence from the nn3 competition on time series prediction. *International Journal of Forecasting*, 27:635–660.

Curry, J., Mukherjee, S., and Turner, K. (2018). How many directions determine a shape and other sufficiency results for two topological transforms. arXiv:1805.09782.

Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. In *Deeper into the brain.* Basic Books.

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 *IJCAI*, volume 2018, pages 2086–2092.

Dunlavy, D. M., Kolda, T. G., and Acar, E. (2011). Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):1–27.

Edelsbrunner, H., Letcher, D., and Zomorodian, A. (2002). Topological persistence and simplification. *Discrete Computational Geometry*, 28:511–533.

Forni, M., Hallin, M., Lippi, M. M., and Reic, L. (2000). The generalized dynamic factor model: identification and estiamtion. *The Review of Economics and Statistics*, 82:540–554.

Geweke, J. (1977). The dynamic factor analysis of economic time series. In *Latent variables in Socio-Economic models*, pages 365–383. North Holland Amsterdam. D.J. Aigner and A.J. Goldberger editors.

Gidea, M. and Katz, Y. (2018). Topological data analysis of finacial time series:landscapes of chrashes. *Physica A. Statistical Mechanics and its Applications*, 491.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. Kdd' 16, August 13-17, San Francisco, CA, USA.

Hallin, M. and Lippi, M. (2013). Factor models in high-dimensional time series – a time domain approach. *Stochastic Processes and their Applications*, 123:2678–2695.

Hastie, T. (1984). Principal curves and surfaces. Laboratory for Computational Statistics Technical Report 11, Stanford University, Department of Statistics.

He, J., Shang, P., and Xiong, H. (2018). Multidimensional scaling analysis of financial time series based on modified cross-sample entropy methods. *Physica A: Statistical Mechanics and its Applications*, 500:210–221.

Hellton, K. H. and Thoresen, M. (2017). When and Why are Principal Component Scores a Good Tool for Visualizing High-dimensional Data? *Scandinavian Journal of Statistics*, 44(3):581–597.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313:504–507.

Hyndman, R. and Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts.

Hyndman, R., Koehler, A., Ord, J., and Snyder, R. (2008). *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Science and Business Media.

Jenkins, O. C. and Matarić, M. J. (2004). A Spatio-Temporal Extension to ISOMAP Nonlinear Dimension Reduction. In *Proceedings of the 21st International Conference on Machine Learning*, page 56.

Jordanger, L. and Tjøstheim, D. (2022). Nonlinear spectral analysis: A local gaussian approach. *Journal of the American Statistical Association*, 117:1010–1027.

Jullum, M., Løland, A., Huseby, R. B., Ånonsen, G., and Lorentzen, J. (2020). Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control*, 23(1):173–186.

Karatzoglou, A., Smola, A., and Hornik, K. (2022). *kernlab: Kernel-Based Machine Learning Lab*. R package version 0.9-31.

Kazemi, S., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., and Poupart, P. (2020). Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21:1–73.

Krampe, J. (2019). Time series modeling on dynamic networks. *Electronic Journal of Statistics*, 13:4945–4976.

Kucyi, A., Hove, M. J., Esterman, M., Hutchison, R. M., and Valera, E. M. (2017). Dynamic brain network correlates of spontaneous fluctuations in attention. *Cerebral cortex*, 27(3):1831–1840.

Lam, C. and Yao, Q. (2012). Factor modeling for high-dimensional time series: Inference for the number of factors. *Annals of Statistics*, 40:694–726.

Lehoucq, R. B. and Sorensen, D. C. (1996). Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17(4):789–821.

Li, L. and Guedj, B. (2021). Sequential learning of principal curves: summarizing data streams on the fly. *Entropy*, 23.

Lian, W., Talmon, R., Zaveri, H., Cari, L., and Coifman, R. (2015). Multivariate time-series analysis and diffusion maps. *Signal Processing*, 1116:13–28.

Lin, Y.-T., Malik, J., and Wu, H.-T. (2021). Wave-shape oscillatory model for nonstationary periodic time series analysis. *Foundations of Data Science*, 3(2):99–131.

Lopes, A. and Machado, A. (2014). Analysis of temperature time-series: Embedding dynamics into the mds method. *Communications in Nonlinear Science and Numerical Simulation*, 19:851–871.

Ludkin, M., Eckley, I., and Neal, P. (2018). Dynamic stochastic block models. parameter estimation and detection in community structure. *Statistics and Computing*, 28:1201–1213.

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., Xiong, H., and Akoglu, L. (2021). A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*.

Mahapatra, S. and Chandola, V. (2020). Learning manifolds from non-stationary streaming data. arXiv:1804.08833v3.

Makridakis, S., Hyndman, R., and Petropoulos, F. (2020a). Forecasting in social settings: The state of the art. *International Journal of Forecasting*, 36:15–28.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020b). The m4 competition: 100 000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36:54–74.

Maroulas, V., Nasrin, F., and Obello, C. (2020). A bayesian framework for persistent homology. *SIAM Journal of Mathematical Sciences*, 2.

McKenzie, E. (1984). Genaral exponetial smoothing and the equivalent arma process. *Journal of Forecasting*, 3:333–344.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. CoRR, abs/1301,3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26.

Moon, C. and Lazar, N. A. (2020). Hypothesis testing for shapes using vectorized persistence diagrams. arXiv preprint arXiv:2006.05466.

Newman, M. (2020). *Networks.* Oxford University Press. 2nd revised edition.

Nguyen, N. and Quanz, B. (2021). Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9117–9125.

Papaioannou, O., Kevrekidis, I., Talmun, R., and Siettos, C. (2021). Time series forecasting using manifold learning. arXiv:2110.03625v4.

Peixito, T. (2021). Descriptive vs. inferential community detection: pitfalls, myths and half-truths. arXiv:2112.00183v1.

Pena, D. and Yohai, V. (2016). Generalized dynamic principal components. *Journal of the American Statistical Association*, 111:1121–1131.

Perea, J. A. and Harer, J. (2015). Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. KDD' 14, , http://dx.doi.org/10.1145/2623330.2623732.

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. Proceedings WSDM, ACM, New Tork, NY, USA.

Qiu, Y. and Mei, J. (2022). *RSpectra: Solvers for Large-Scale Eigenvalue and SVD Problems.* R package version 0.16-1.

Raj, A., Cai, C., Xie, X., Palacios, E., Owen, J., Mukherjee, P., and Nagarajan, S. (2020). Spectral graph theory of brain oscillations. *Human Brain Mapping*, 41:2980–2998.

Ravisshanker, N. and Chen, R. (2019). Topological data analysis (TDA) for time series. arXiv: 1909.10604v1.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326.

Salinas, D., Flunkert, V., and Gasthaus, J. (2019). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 8:136–153.

Sargent, T. and Sims, C. (1977). Business cycle modelling without pretending to have too much a priori economic theory. In *New Methods in Business Cycle Research*, pages 45–109. Federal Reserve Bank of Minneapolis. C.A. Sims editor.

Sarkar, P. and Moore, A. (2005). Dynamic social network analysis using latent space models. *Advances in Neural Information Processing Systems*, 18.

Schölkopf, B., Smola, A., and Müller, K.-L. (2005). Kernel principal components. *Lecture Notes in Computer Science*, 1327:583–588.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36:75–85.

Stock, J. and Watson, M. (2002). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, 97:1167–1179.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077.

Tanigushi, M. and Krishnaiah, P. (1987). Asymptotic distributions of functions of the eigenvalues of sample covariance matrix and canonical correlation matrix in multivariate time series. *Journal of Multivariate Analysis*, 22:156–176.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323.

Tjøstheim, D., Jullum, M., and Løland, A. (2022a). Statistical embedding: Beyond principal components. Statistical Science, to appear.

Tjøstheim, D., Jullum, M., and Løland, A. (2022b). Supplement to "Some recent trends in time series and dynamic networks embeddings".

Tjøstheim, D., Otneim, H., and Støve, B. (2022c). *Statistical Modeling Using Local Gaussian Approximation.* Academic Press.

Tong, H. (1990). *Nonlinear Time Series. A Dynamical System Approach.* Oxford University Press.

Torgerson, W. (1952). Multidimensional scaling: 1 theory and method. *Psychometrica*, 29:1–27.

Varley, T. and Sporns, D. (2022). Network analysis of time series: Novel approaches to network neuroscience. *Frontiers in Neuroscience*, pages 1–20.

Wang, Y., Ombao, H., and Chung, M. K. (2018). Topological data analysis of single-trial electroencephalographic signals. *The Annals of Applied Statistics*, 12(3):1506.

Wasserman, L. (2018). Topological data analysis. *Annual Review of Statistics and its Applications*, 5:501–532.

Wen, R., Torkkola, K., Narayanaswamy, B. M., and Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. In *Advances in Neural Information Processing Systems – Time Series Workshop*, volume 31.

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. (2020). Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*.

Xu, K. S. and Hero, A. O. (2014). Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):552–562.

Yu, H.-F., Rao, N., and Dhillon, I. (2016). Temporal regularized matrix factorization for high dimensional time series prediction. In *Advances in Neural Information Processing Systems*, volume 29, pages 847–855.

Yu, L., He, Y., Kong, X., and Zhang, X. (2022). Projected estimation for large-dimensional matrix factor models. *Journal of Econometrics*, 229:201–217.

Yu, Y., Wang, T., and Samworth, R. J. (2015). A useful variant of the davis–kahan theorem for statisticians. *Biometrika*, 102(2):315–323.

Zhang, J. and Chen, Y. (2020). Modularity based community detection in heterogeneous networks. *Statistica Sinica*, 30:601–629.

Zhang, J., Dong, Y., Wang, Y., Tang, J., and Ding, M. (2019). ProNE: Fast and Scalable Network Representation Learning. In *IJCAI*, volume 19, pages 4278–4284.

Zhang, X. and Tong, H. (2022). Asymptotic theory of principal component analysis for time series data with cautionary comments. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 185(2):543–565.

Zhu, X. and Pan, R. (2020). Grouped network vector autoregression. *Statistica Sinica*, 30(3):1437–1462.

Zhu, X., Pan, R., Li, G., Liu, Y., and Wang, H. (2017). Network vector autoregression. *Annals of Statistics*, 45:1096–1123.

Zomordian, A. and Carlsson, G. (2005). Computing persistent homology. *Discrete Computational Geometry*, 33:249–274.

Supplement to "Some recent trends in embeddings of time series and dynamic networks"

Dag Tjøstheim[1,2] Martin Jullum[2] Anders Løland[2]

[1]University of Bergen
[2]Norwegian Computing Center

1 Deep learning networks

Neural networks are used for a number of problems in prediction, classification and clustering. Currently, there is an intense activity involving among other things deep learning, where some remarkable results have been obtained. See Schmidhuber (2015) for an overview that is still very useful (even though it was written seven years ago).

In this section we explain the principle for a single layer artificial network and then go on to multiple layer networks and deep learning. Assume that we are given an n-vector x as input which may be thought of as a sliding window of data. In a neural network approach one is interested in transforming x via linear combinations of its components and possibly a nonlinear transformation of these linear combinations. Those transformations constitute what is called a hidden layer.

Given the input layer, the first step in forming the hidden layer is to form linear combinations

$$h_i = \sum_{j=1}^{n} w_{ij} x_j + w_{i0}, \tag{1}$$

where $i = 1, \ldots, m$, and where $b_i = w_{i0}$ is a so-called bias term.

In the case of one hidden layer, the output layer is given in terms of the hidden units as

$$y_j = \sum_{i=1}^{m} w'_{ij} h_i, \tag{2}$$

for $j = 1, \ldots, q$, where q in general may be different from n.

In a classification problem, y_j may be associated with an unnormalized probability for a class j, which is related to an appropriate neighborhood of a node v_j in a network. In such cases the output layer in (2) is also transformed. A common transformation is the so-called softmax function given by

$$\text{softmax}(y_j) = \frac{\exp(y_j)}{\sum_{i=1}^{q} \exp(y_i)}. \tag{3}$$

This is recognized (if there is no hidden layer) as the multinomial logistic regression model which is a standard tool in classification.

In a s-step time series forecasting problem, using the notation $Y_t = y_j$, Y_t may successively be X_{t+s} for an input training sets $\{X_1, \ldots, X_t\}$.

Using a training set, the coefficients (or weights) w_{ij} and w'_{ij} are determined by a penalty function measuring the distance between the output y and the target vector t, for example measured by the loss function $E = ||y-t||^2$. In a classification and clustering problem the training set consists of input vectors x belonging to known classes i. The target vector is a so-called "one hot" vector having 1 at the component j for the given target and zeros elsewhere. The weights are adjusted such that the output vector is as close as possible to this vector. The same, but a simpler procedure due to a simpler output, is followed for forecasting.

The error function is evaluated for each of the samples coming in as inputs, and the gradient of the error function with respect to y is evaluated with the weights being re-computed and updated in the direction of the gradient by stochastic gradient descent.

The weights w'_{ij} for the output layer are computed first. Then, w_{ij} are adjusted via the chain differentiation rule using so-called back propagation. Details are given in e.g. the appendix of Rong (2016). Schematically this may be represented by

$$w_{ij}^{(\text{new})} = w_{ij}^{(\text{old})} - \varepsilon \frac{\partial E}{\partial w_{ij}},$$

where ε is a sensitivity parameter. Initial values for the weights can be chosen by drawing from a set of uniform variables.

This simple one-layer forward scheme has been used with considerable success (Mikolov et al., 2013a,b) in graph embedding as described in several sections of TJL. Dynamic embedding of graphs is described specifically in Section 5 of TJL.

However, it has turned out that in many problems a much improved performance can be obtained by deep multiple layer networks, whose learning is said to be done by deep learning.

For iid data mainly treated in Section 3 of TJL, the relevant problem is the clustering and discrimination problem, the forecasting problem being meaningless. For the time series case treated in this paper a main emphasis is on prediction, and the following will be phrased with time series notation $\{X_t\}$. A prime advantage with deep learning is then that hidden units can be connected to each other recursively in time. Using the framework and notation of Lim and Zohren (2021), the goal is to predict future values of a target $Y_{i,t}$, for a given entity i at time t. Each entity can represent a logical grouping of temporal information – such as measurements from individual weather stations in climatology, or vital signs from different patients in medicine, a selection of economic time series – and can be observed at the same time, the simplest one-step-ahead forecasting models take the form

$$\hat{Y}_{i,t+1} = f(Y_{i,t-k:t}, X_{i,t-k:t,s_i}),$$

where $\hat{Y}_{i,t+1}$ is the model forecast, $Y_{i,t-k:t} = \{Y_{i,t-k}, \ldots, Y_{i,t}\}$, $X_{i,t-k:t} = \{X_{i,t-k}, \ldots, X_{i,t}\}$ are observations of the target and exogenous inputs respectively over a look-back window of length k, and s_i are static meta-data associated with the entity (e.g. station location in a meteorology network). The function f is the prediction function learned by the model. It contains the weight functions, the hidden units of the hidden layers, which can be chosen in various ways as shown in the following. The model is phrased here in terms of univariate forecasting, but can be extended to multivariate forecasting without loss of generality (Sen et al., 2019; Wen et al., 2017; Li et al., 2018; Salinas et al., 2019a).

1.1 Convolution neural networks (CNN)

There are several ways of choosing the function f. Convolution neural networks (CNN), Gu et al. (2018), were originally designed for spatial data but have been adapted to time series where researchers use multiple layers of causal convolutions. For an intermediate layer l, each causal convolution filter, in terms of hidden units, takes the form

$$h_t^{l+1} = A\Big((\mathbf{w} * h)(l,t)\Big) \tag{4}$$

with the convolution between weights and hidden units given by

$$(\mathbf{w} * h)(l,t) = \sum_{\tau=0}^{k} \mathbf{w}(l,\tau) h_{t-\tau},$$

In (4), h_t^l is an intermediate state at layer l, and $\mathbf{w}$ is a fixed matrix of convolution weights at layer l. The function A is an activation function such as a sigmoidal function.

The fixed filter weights, not depending on time t, is a feature inherited from the spatial case. In addition, CNNs are only able to use inputs within its fixed look-back window to make forecasts. It is necessary with a carefully tuned choice of k to take advantage of all relevant historical information. It is worth noting that a single causal CNN layer with a linear activation model is equivalent to an autoregressive model.

1.2 Recurrent neural networks (RNN)

Recurrent neural networks (RNN) do not require an explicit fixed specification of a look-back window as is the case for CNN. Given the natural interpretation of time series prediction as sequences of inputs and targets, many forecasting applications have been developed for temporal forecasting applications (Salinas et al., 2019b; Rangupuram et al., 2018; Lim et al., 2020). It has also been used in natural language processing (Young et al., 2018). At its core, RNN cell units contain an internal memory state which acts as a compact memory of past information. The memory state is recursively updated from new observations at each time step.

Recurrent neural networks are originally based on work by Rumelhart et al. (1986). Since then, a number of different RNN architectures have been developed. Just to give a flavor of these networks, we present the model equations for the three layer Elman network. This network is essentially a three layer network, with an input layer $\{X_t\}$, a hidden layer $\{h_t\}$ and an output layer $\{Y_t\}$. At each time step the input is fed forward and a learning rule is applied using a training set. The fixed back-connections save a copy of the previous values of the hidden units in the context unit (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state variable allowing it to perform such tasks as sequence-prediction that are beyond the power of a standard multilayer perceptron. The model equations for the Elman network are

$$h_t = \sigma_h(\mathbf{W}_h X_t + \mathbf{U}_h h_{t-1} + b_h),$$

$$Y_t = \sigma_y(\mathbf{W}_y h_t + b_y),$$

where X_t is an input vector, h_t a hidden layer vector, Y_t an output vector, $\mathbf{W}, \mathbf{U}, b$ are parameter matrices and a parameter vector b, σ_h and σ_y are activation functions, e.g.

of sigmoidal form. These should be compared to the simple feedforward system (1) and (2).

Jordan networks are similar to Elman networks. The context units are fed from the output layer instead of from the hidden layer, so that the model equations are

$$h_t = \sigma_h(\mathbf{W}_h X_t + \mathbf{U}_h Y_{t-1} + b_h),$$

$$Y_t = \sigma_y(\mathbf{W}_y h_t + b_y).$$

Elman and Jordan networks are also known as simple recurrent networks (SRN). There exist considerably more complicated recurrent networks with a number of layers as in the CNN network, but the idea of recursion in time is kept.

1.3 Long-short term memory (LSTM) recurrent networks

The long term gradients which are back-propagated can tend to zero or explode, i.e. tend to infinity, in classic RNNs. The problem is computational (or practical) when training the network using back propagation. The so-called long-short term memory (LSTM) recurrent network is designed to counter this problem. LSTM units partially solve the vanishing gradient problem because LSTM units allow gradient also to flow unchanged. However, LSTM networks can still suffer from the exploding gradient problem.

The vanishing gradient was first analyzed in a diploma thesis by Sepp Hochreiter at the Technological University of Munich under the guidance of Jürgen Schmidhuber. After having been published in a technical report as long short term memory and as a conference proceedings, the full LSTM paper appeared in 1997 in Neural Computation (Hochreiter and Schmidhuber, 1997). Since then there have been a substantial theoretical and not the least applied advances of the method. Now the original LSTM paper stands with a hefty more than 67 000 citations. The paper has become the most cited neural network article of the 20th century, and has been applied with considerable success to topics such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games and health care. LSTM is particularly well suited to classifying, and to processing and making predictions of time series data, since there can be lags of unknown duration between important events in a time series.

A common LSTM unit is composed of a cell unit, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell.

2 Simplical complexes

First, recall the definition of a simplex: Given a set $\mathbb{X} = \{X_0, \ldots, X_k\} \subset \mathbb{R}^p$ of $k+1$ "affinely independent vectors" (i.e., the vectors $(X_0, X_1, \ldots X_k)$ are linearly independent), the k-dimensional simplex $\sigma = [X_0, \ldots, X_k]$ spanned by $\mathbb{X}$ is the convex hull of $\mathbb{X}$. For instance, for $k = 1$ the simplex is simply given by the line from X_0 to X_1. The points of $\mathbb{X}$ are called the nodes of σ and the simplices spanned by the subsets of $\mathbb{X}$ are called the faces of σ. A geometric simplical complex K in $\mathbb{R}^p$ is a collection of simplices such that (i) any face of a simplex of K is a simplex of K, (ii) the intersection of any two simplices of K is either empty or a common face of both.

For a point cloud a graph can be constructed by connecting points that satisfy a neighborhood criterion, e.g. the distance is under a certain threshold. See for instance

Sections 3.4 and 5.1 in TJL. This leads to the standard notion of a neighboring graph from which the connectivity of the data can be analyzed and clustering can be obtained, including non-convex situations, as described by the Ranunculoid example. Using simplical complexes, where simplical complexes of dimension 1 are graphs, one can go beyond this simple form of connectivity. In fact a central idea in TDA (topological data analysis) is to build higher dimensional equivalents of neighboring graphs by not only connecting pairs but also $(k+1)$-tuples of nearby data points. This enables one to identify new topological features such as cycles and voids and their higher dimensional counterparts. Regarding embedding of networks, as treated in Section 5, such a technique could possibly be used to discover cycles in networks such as criminal rings in fraud detection, say.

Simplical complexes are mathematical objects that have both topological and algebraic properties. This makes them especially useful for TDA. There are two main examples of complexes in use. They are the Vietoris-Rips complex and the Čech complex. The Vietoris-Rips complex $V_\varepsilon(\mathbb{X})$ can be introduced in a metric space (M, d). It is the set of simplices $\mathbb{X} = [X_0, \dots, X_k]$ such that $d_{\mathbb{X}}(X_i, X_j) \leq \varepsilon$ for all (i, j).

These definitions should be compared to the use of ball-coverings in equation (7) of the main paper. The homology of V_ε can be computed using basic matrix operations. All relevant computations can be reduced to linear algebra. This gives a method of computing homology and persistent homology relating the complexes as ε varies as briefly mentioned in our simple introductory example of chain of circles, or the more involved example involving Ranunculoids.

Given a subset $\mathbb{X}$ of a compact metric space (M, d), the families of Vietoris-Rips complexes, $\{V_\varepsilon(\mathbb{X})\}_{\varepsilon \in \mathbb{R}}$ are filtrations, that is, nested families of complexes. As indicated earlier, the parameter ε can be considered as a data resolution level at which one considers the data set $\mathbb{X}$.

As in the introductory example in Section 4 of the main paper, the homology of a filtration $\{F_\varepsilon\}$ changes as ε increases: new connected components can appear, existing components can merge, and loops and cavities may appear or be filled. Persistence homology tracks these changes, identifies the appearing features, and attaches a lifetime to them in the persistence diagram.

References

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., and Chen, T. (2018). Recent advances in convolution neural networks. *Pattern Recognition*, 77:354–377.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018). Diffusion convolutional recurrent neural network: Datadriven traffic forecasting. In Proceedings of the International Conference on Learning Representations.

Lim, B. and Zohren, S. (2021). Time series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, pages 1–13.

Lim, B., Zohren, S., and Roberts, S. (2020). Learning independent bayesian filtering steps for time series prediction. In International Joint Conference on Neural networks.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. CoRR, abs/1301,3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26.

Rangupuram, S., Seeger, M., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, volume 32.

Rong, X. (2016). word2vec parameter learning explained. arXiv:1411.2738v4.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. (2019a). High-dimensional multivariate forecasting with low rank gaussian copula processes. In *Advances in Neural Information Processing Systems*, volume 33, pages 7796–7805.

Salinas, D., Flunkert, V., and Gasthaus, J. (2019b). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 8:136–153.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Sen, R., Yu, H., and Dhillon, I. (2019). Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances of Neural Information Processing Systems*, volume 33, pages 1–10.

Wen, R., Torkkola, K., Narayanaswamy, B. M., and Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. In *Advances in Neural Information Processing Systems – Time Series Workshop*, volume 31.

Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13:55–75.