



A graph-based superframework for mixture model estimation using EM: an analysis of US wholesale electricity markets

Carlo Mari¹ · Cristiano Baldassari²

Received: 3 October 2022 / Accepted: 7 March 2023
© The Author(s) 2023

Abstract

A fully unsupervised graph-based superframework is proposed to handle the EM initialization problem for estimating mixture models on financial time series. Using a complex network approach that links time series and graphs, the graph-structured information derived from the observed data is exploited to produce a meaningful starting point for the EM algorithm. It is shown that structural information derived by complex graphs can definitely capture time series behavior and nonlinear relationships between different observations. The proposed methodology is employed to estimate Gaussian mixture models on US wholesale electricity market prices using two different configurations of the superframework. The obtained results show that the proposed methodology performs better than conventional initialization methods, such as *K*-means based techniques. The improvements are significant on the overall representation of the empirical distribution of log-returns and, in particular, on the first four moments. Moreover, this approach has a high degree of generalization and flexibility, exploiting graph manipulation and employing functional operating blocks, which can be adapted to very different empirical situations.

Keywords Markov transition fields · Quantile networks · Graph embedding · Topological data analysis

1 Introduction

Network science is a well-established discipline that explores a broad variety of natural and social phenomena by describing actual relationships via complex network, or graph, structures [1]. There is a long history of effective applications of complex networks in various fields [2–5].

Several time series analysis techniques based on network science have been proposed in the previous decade, utilizing the huge body of research on network analysis and

providing new insights and innovative perspectives on understanding the granular structure of time series [6, 7]. Indeed, transforming a time series into a network improves in-depth the analysis, leading to the discovery of time series non-trivial topological features [8, 9] and providing new viewpoints and ideas for penetrating into the probabilistic structure of time series [10, 11].

1.1 Problem statement

Finite mixture models provide universal approximations to any continuous probability density and have proven to be very useful in describing observed data in many fields of application [12, 13]. In particular, Gaussian mixture models (GMMs) are intriguing instances of mixture models that are frequently used as an effective tool for data analysis and modeling, especially in signal and information processing [14–17]. The methods for estimating the parameters in a mixture model are crucial to the model performance. Among the others, the expectation-maximization (EM) algorithm, frequently used for training mixture models [18], provides an iterative approach for parameter estimation based on likelihood maximization.

Carlo Mari and Cristiano Baldassari have contributed equally to this work.

✉ Cristiano Baldassari
cristiano.baldassari@unich.it

Carlo Mari
carlo.mari@unich.it

¹ Department of Economics, University of Chieti-Pescara, Viale Pindaro, 42, 65100 Pescara, PE, Italy

² Department of Neuroscience, Imaging and Clinical Sciences, University of Chieti-Pescara, Via Luigi Polacchi, 11, 66100 Chieti, CH, Italy

However, the initialization settings of the EM technique have a significant influence on the quality of the resulting solution [19]. Indeed, EM maximum likelihood estimation is subject to the local-optima problem and choosing the right initial values is essential for getting accurate results [20–22]. Due to this predisposition toward local solutions, it is implausible that any given solution, regardless of the initialization strategy, would be globally optimum [22].

1.2 Related works

Clustering algorithms are used in the literature to provide meaningful solutions to the EM initialization problem. In particular, *K*-means [23] and Random algorithms [24] received great attention in recent years. In both these algorithms, the number of mixture components must be set exogenously. Moreover, cluster centers (centroids) are chosen randomly and, due to the high occurrence of local solutions, the estimation procedure must be re-initialized several times before a solution can be detected. In addition, no existing technique can determine when the number of initialization is sufficient to ensure a full examination of the likelihood function [22]. Other clustering algorithms for EM initialization, such as hierarchical clustering [25] or rough-enhanced-Bayes mixture estimation (REBMIX) [26], have been proposed in the literature. However, no one of these experimental strategies can be considered as the best one [24]. Some authors investigated the feasibility of realizing a fully unsupervised framework, i.e., without any guesswork about the number of mixture components, by mapping time series to complex networks, thus determining the optimal number of the mixture model components and the vector of initial parameters [27]. The proposed framework has done an outstanding job in addressing the local-optima problem, thus providing accurate estimates of GMM parameters.

1.3 Purpose

The aim of this paper is develop a fully unsupervised graph-based superframework to handle the EM initialization problem for estimating GMMs on financial log-return time series. In fact, although price time series are typically non stationary, log-return time series, computed as the difference in log-prices between two subsequent observations, demonstrate better behavior [28]. We will show that the proposed approach has a high degree of generalization and flexibility, taking advantage of graph manipulation and employing functional operating blocks that can be adapted to any empirical situation. This superframework is described in the end-to-end flowchart shown in Fig. 1.

The workflow is composed by three sections, namely the input, the distribution estimation, and the output. The input

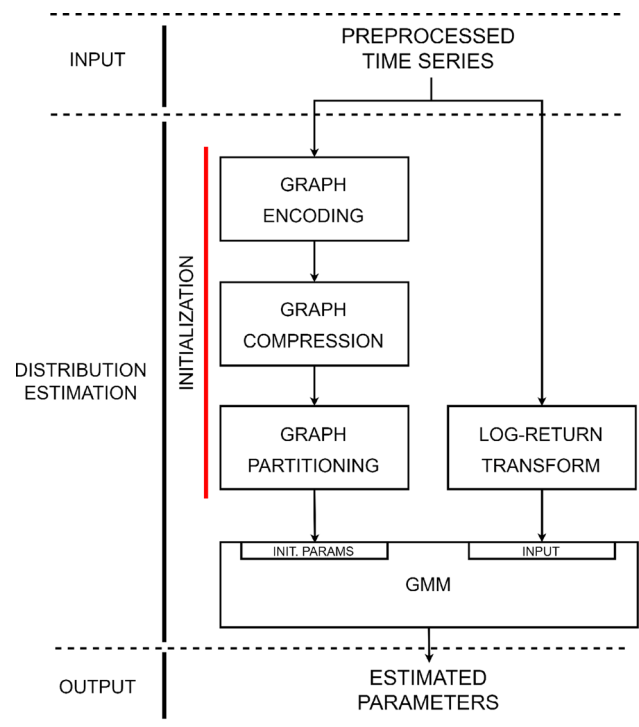


Fig. 1 End-to-end workflow diagram. The red line highlights the initialization blocks

is the possibly preprocessed log-price time series. Indeed, in some cases time series of market prices needs to be preprocessed to account for possibly missing data imputation and trends [29]. Then, the preprocessed log-price time series is splitted in two branches both entering the distribution estimation section, one going into the initialization blocks (highlighted in red in Fig. 1), and the other branch going directly into the GMM model block after a log-return transform. In the initialization blocks, the preprocessed time series undergoes three major initialization transformations, namely graph encoding, graph compression and graph partitioning, in order to determine the vector of initialization parameters. First, the graph associated to the preprocessed time series must be created (graph encoding); then, by lowering the complexity of the graph (graph compression), log-return communities (graph partitioning) can be identified. Next, the number of the GMM components is set equal to the number of the detected communities, and the remaining initializing parameters of the EM algorithm are determined by log-return community membership, as will be explained in the text. In this way, all the inputs of the GMM block, i.e., the initializing parameters and the log-return time series, are composed. Finally, the output section provides the estimated parameters through the EM algorithm.

The graph encoding step is realized by mapping the preprocessed log-price time series (hereinafter, log-price

time series), into a quantile network (according to the taxonomy proposed in the literature [11]), using a Markov transition field (MTF) as adjacency matrix [30–33]. The use of a MTF as adjacency matrix is particularly well suited to fully account for the transitions between the observed dynamic states [30].

To reduce the complexity of the created graph, we employed two different graph compression techniques, namely graph coarsening [34] and graph embedding [35]. Graph coarsening is a model-driven technique used to reduce the size of a graph maintaining essential properties. Despite the plethora of literature on graph coarsening, model-driven methods are the most popular, as data-driven techniques are still under-explored [34]. Graph embedding, on the other hand, is based on a data-driven approach, transforming the graph to a multi-dimensional Euclidean vector space. The essential idea of graph embedding methods is to learn effective feature representations of network nodes in an unsupervised manner preserving the graph structural properties [35, 36]. In this research, we will employ a variety of embedding techniques using the Karate Club open source Python framework [37, 38], which includes state-of-the-art and cutting-edge approaches for conducting unsupervised learning on graph-structured data. The classes of embedding, as listed on the Karate Club paper and suitable for our procedure, are: (i) neighborhood preserving embedding; (ii) structural embedding; (iii) attributed embedding; (iv) meta-embedding. In particular, we used the so-called diff2Vec embedding method as representative of the first class [39]; the GraphWave method for the second class [40]; the attributed social network embeddings (ASNE) for the third class [41], and network embedding update (NEU) for the fourth class [42].

The community detection task is performed in the graph partitioning block. We used two different techniques, the Louvain method [43] and a clustering technique based on topological data analysis (TDA) [44]. The Louvain Method provides a technique of community discovery based on the optimization of the graph modularity. Instead, operating with a TDA-based clustering technique, the community discovery process is performed directly on the graph embedding using the Topological Mode Analysis Tool (ToMATo) [45]. This latter strategy, compared to other current clustering methods, demonstrates a superior advantage by providing an accurate unsupervised system for determining how many stable clusters the data contain through to the use of persistent homology [44, 46–48].

We applied the proposed superframework to US wholesale electricity markets, investigating the behavior of daily electricity prices at Palo Verde (Southwest area), PJM (Northeast region), SP15 (Southern California) and Nepoch (New England) in the time interval January 1, 2017 to

December 31, 2021. US electricity price time series show irregular sampling (lack of daily data points) as well as trend and seasonality. Typically, electricity prices may be higher in winter and in summer, and the seasonal component must account for this semiannual periodicity. A trend must be included to account for expected inflation and possibly for a real power price escalation rate (positive or negative). For these reasons, a preprocessing consisting of a gap filling in procedure and a seasonal-trend removal is first accomplished.

Within the superframework and focusing on the initialization blocks, we identify two different frameworks that combine the techniques described above: the first framework is composed by (i) graph encoding using MTF, (ii) graph compression with graph coarsening techniques, (iii) graph partitioning with the Louvain method; the second framework as (i) graph encoding using MTF, (ii) graph compression with graph embedding, (iii) graph partitioning with ToMATo clustering. We will call graph approximation framework (GA framework) the first framework and graph representation learning framework (GRL framework) the second. In a previous paper [27], we have shown that the GA framework performs better than conventional initialization methods, such as *K*-means and random-based techniques. In this paper, we will see that the GRL framework provides even more accurate results than the GA framework. The higher likelihood values reached imply a stronger learning performance than the GA framework in the distribution estimation task. This is due also to the fact that the GRL framework has a more conservative pipeline by design (less information loss in the graph compression phase), and the need for custom feature engineering as graph coarsening is reduced by superior data-driven and unsupervised graph machine learning techniques. In addition, embedding methods combined with TDA tools made feature extraction highly accurate and efficient.

To the best of our knowledge, this is the first study in which these embedding approaches are used with quantile graphs created with time series MTF encoding. In particular, we highlight that the combination of NEU meta-embedding applied to Diff2Vec, and the ASNE method stand out as especially effective embedding models, and their use with an MTF-encoded quantile graph is a completely new approach.

The rest of this study is structured as follows. The superframework is described in depth in Sect. 2. Section 3 focuses on the empirical analysis. Section 4 concludes. The code to reproduce our results is available in the Github repository at <https://bit.ly/3zFtbQY>.

2 The methodology

2.1 Overview

This section illustrates the proposed superframework approach to the estimation of mixture models, outlining in detail two distinct operational frameworks, namely the GA and the GRL frameworks, that will be used in the empirical analysis. Let us start, therefore, by briefly reviewing some basic facts about Mixture Models and the EM estimation algorithm.

Consider a possibly preprocessed log-price time series $\{x_t\}_{t=1}^N$ and its log-return tranform $\{r_t\}_{t=1}^{N-1}$, where

$$r_i = x_{i+1} - x_i, \quad i = 1, 2, \dots, N - 1. \tag{1}$$

Suppose that the values, r_i , $i = 1, 2, \dots, N - 1$, are extracted in an IID manner from an underlying random model described by a probability density $p(r_i)$. We assume that $p(r_i)$ is a finite mixture distribution with C components,

$$p(r_i | \Theta) = \sum_{c=1}^C \alpha_c p_c(r_i | z_{ic} = 1, \theta_c). \tag{2}$$

In Eq. (2), $z_i = \{z_{i1}, z_{i2}, \dots, z_{iC}\}$ with $i = 1, 2, \dots, N - 1$, is a vector of C unobservable latent binary random variables that are mutually exclusive and exhaustive, i.e., one and only one of the z_{ic} is equal to 1, while the others are equal to 0). The vector z_i plays the role of an indicator random variable representing the identity of the mixture component responsible for the generation of the outcome r_i ; $p_c(r_i | z_{ic} = 1, \theta_c)$ denotes the density distributions of the mixture components with parameters θ_c ; $\alpha_1, \alpha_2, \dots, \alpha_C$ are the mixture weights, i.e., positive numbers such that

$$\sum_{c=1}^C \alpha_c = 1, \tag{3}$$

representing the probability that the value r_i was generated by the component c ; $\Theta = \{\alpha_1, \dots, \alpha_C, \theta_1, \dots, \theta_C\}$ is the complete set of the mixture model parameters [49]. In the case of univariate Gaussian mixture models, the components of the model are described by univariate Gaussian densities,

$$p_c(r_i | \theta_c) = \frac{1}{\sigma_c \sqrt{2\pi}} \exp\left(\frac{-(r_i - \mu_c)^2}{2\sigma_c^2}\right), \tag{4}$$

with parameters $\theta_c = \{\mu_c, \sigma_c^2\}$ denoting, respectively, the mean and the variance of the single component distribution.

Mixture models can be efficiently estimated by maximum likelihood using the EM algorithm [18]. EM is an iterative procedure that begins with an initial estimate of

parameters $\Theta = \Theta_0$ and then iteratively updates Θ until convergence is achieved. Each iteration consists of two steps, the E-step and the M-step.

In the E-Step, given a set of parameters Θ , the so-called membership weight of the data point r_i in component c is computed. It is defined as

$$w_{ic} = p(z_c = 1 | r_i, \Theta), \tag{5}$$

with

$$\sum_{c=1}^C w_{ic} = 1. \tag{6}$$

In this way, we obtain the $N \times C$ matrix of membership in which each row sum to 1. From the Bayes rule, we can cast Eq. (5) in the following equivalent form,

$$w_{ic} = \frac{\alpha_c p_c(r_i | z_{ic} = 1, \theta_c)}{\sum_{m=1}^C \alpha_m p_m(r_i | z_m = 1, \theta_m)}. \tag{7}$$

In the M-step, the algorithm computes the parameter values that maximizes the log-likelihood, starting from the values obtained by suitably aggregating the membership weights generated in the E-step. In the case of Gaussian mixture models such an aggregation can be performed in the following way,

$$\alpha_c = \frac{N_c}{N}, \tag{8}$$

$$\mu_c = \frac{1}{N_c} \sum_{i=1}^N w_{ic} r_i, \tag{9}$$

$$\sigma_c^2 = \frac{1}{N_c} \sum_{i=1}^N w_{ic} (r_i - \mu_c)^2, \tag{10}$$

where $N_c = \sum_{i=1}^N w_{ic}$ represents the effective number of data points assigned to component c . We notice that both the mean and the variance are computed in a way similar to how standard empirical average and variance are computed, but with a fractional weight w_{ic} .

Once the new values of the parameters are obtained via maximum likelihood in the M-step, they are used in the subsequent iterations (composed of both the E-step and the M-step). The iterative procedure continues until convergence is achieved.

The EM algorithm can be started by selecting a set of initial parameters and then performing the E-step, or by selecting a set of initial weights and then conducting a first M-step. The initial parameters or the initial weights can be chosen randomly or could be derived using some heuristic method [49]. Within the superframework, we identified two different operational frameworks to perform this initialization task, namely the GA and GRL frameworks. Figure 2 depicts the specific tasks performed in each

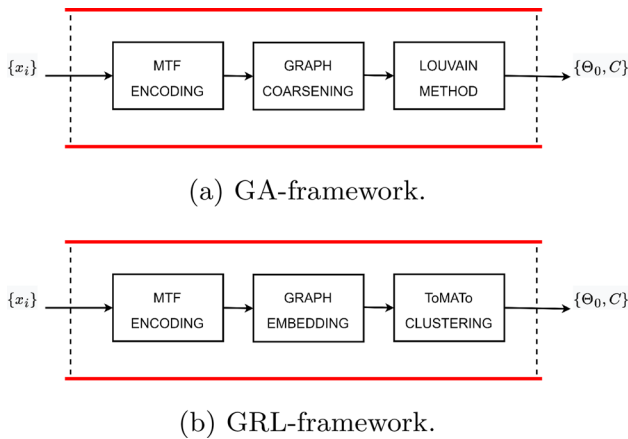


Fig. 2 The initialization blocks

framework, putting into evidence major distinctions between the GA and GRL frameworks.

In both frameworks, the preprocessed log-price time series is encoded into a graph network using a MTF-based encoding. Then, the graph will undergo a compression phase performed through graph coarsening techniques in the GA framework, and graph embedding techniques in the GRL framework. Finally, the community detection is accomplished by a partitioning procedure performed using the Louvain Method to optimize the modularity of the graph in the GA framework, and ToMaTo multidimensional clustering of the graph embedding in the GRL framework. Table 1 summarizes the specific task of each block of Fig. 2.

The partitioning step serves as a feature extraction procedure for the distribution estimation to generate automatically the number of components of the mixture model and the initialization parameters. In fact, since each log-return value, $r_i = x_{i+1} - x_i$, can be uniquely associated to the value x_i , the detected communities can be viewed also as log-return communities. In both frameworks the number of components is set equal to the number of the detected communities, and the other initialization parameters, contained in the initializing vector Θ_0 , are computed through community membership (i.e., $w_{ic} = 1$ if the observation r_i

belongs to the cluster c , 0 otherwise) using Eqs. (8–10). Then, a first M-step is performed. Once the new values of the parameters are obtained via maximum likelihood in the M-step, they are used in the subsequent iterations (composed of both the E-step and the M-step) until convergence is obtained.

2.2 Graph encoding with Markov transition fields

This methodology is based on mapping the log-price time series into a graph. The purpose is to identify typical patterns or similar contexts with the aim to form a network and detect communities of nodes that may be associated with the components of a mixture model. In order to map a time series onto a complex network, the adjacency matrix must be defined. If the underlying dynamics is determined by a Markov process, the Markov transition matrix can be used as an adjacency matrix [11]. In fact, it allows us to properly account for the transitions among different states of the observed dynamics. In such a case, the time series must be first quantized in order to define the dynamical states. This task can be accomplished in the following way. Let us consider on the real axis a certain number, say Q , of adjacent intervals by positioning $Q + 1$ cut points, namely $\{q_0, q_1, \dots, q_Q\}$, to divide log-price observation data into continuous intervals, hereinafter bins, with equal probabilities,

$$P(q_k) - P(q_{k-1}) = 1/Q. \tag{11}$$

In this way, we can group observation data into Q bins, B_1, B_2, \dots, B_Q , that identify the Q states of the dynamics. In our analysis, both the empirical distribution, hereinafter Quantile binning, and the normal distribution fitted to time series data, hereinafter normal binning, are used. In the latter case, the whole real axis is divided into Q interval with equal probability. In both cases, each observation is mapped to the corresponding bin,

$$\{x_1, x_2, \dots, x_N\} \rightarrow \{d_1, d_2, \dots, d_N\}, \tag{12}$$

where d_i denotes the bin containing the value x_i . After

Table 1 Initialization tasks. GA and GRL task columns are merged when the GA and GRL tasks are the same

GA block	GRL block	GA task	GRL task
Graph Encoding	Graph Encoding	Transforming the log-price time series into a quantile network using the MTF as adjacency matrix	
Graph Coarsening	Graph Embedding	Reducing graph dimension by graph coarsening	Representation learning of the MTF matrix by a graph node embedding
Louvain Method	ToMaTo Clustering	Community detection by optimizing the modularity of the graph	Community detection by multidimensional clustering of the graph embedding

assigning each x_i to its corresponding bin d_i , we construct a $Q \times Q$ weighted matrix Ω by counting transitions among bins in the manner of a first order Markov chain. By transition we mean a transition between two consecutive observations, namely from x_i to x_{i+1} , that identify a transition between the bin (the state) d_i to the bin (the state) d_{i+1} . The Markov transition matrix can be, then, obtained by identifying each element of Ω , namely $\Omega_{h,k}$, with the relative frequency of transitions between bins B_h and B_k ($h, k = 1, \dots, Q$). However, the information contained in the Markov matrix is too coarse. In order to refine the analysis we introduced the so-called Markov transition field (MTF) [30, 50], a $N \times N$ matrix whose elements are defined as follows,

$$M_{i,j} = \Omega_{d_i,d_j}, \quad (13)$$

with $i, j = 1, \dots, N$. In a Markov transition field, the information contained in the Markov transition matrix is spread out along the whole time series. Indeed, $M_{i,j}$ represents the probability associated to a transition from the bin d_i which contains x_i to the bin d_j containing x_j , as given in the Markov transition matrix Ω . The M matrix results as a spread of the Ω matrix along the time axis and is intended to enhance analytical capabilities, enabling a comprehensive investigation of the associated network. The Markov transition field M allows us to map a time series into a complex network. This can be done by using the matrix M as the adjacency matrix of the graph G , mapping the vertices (nodes) V to the row-column indices (i, j) and the edge weights to $M_{i,j}$.

2.3 Graph compression

This section discusses some graph compression techniques, namely graph coarsening in the GA framework and unsupervised embedding of graph in the GRL framework.

2.3.1 Graph coarsening

The goal of graph coarsening is to find a smaller graph \bar{G} , which is a good approximation of G [51]. In the GA framework, to make the whole estimation procedure computationally more feasible and efficient, the size of the matrix M is reduced by averaging its elements in each non-overlapping $g \times g$ sub-matrix with a kernel $1/g^2$, thus obtaining a reduced $S \times S$ square matrix, \bar{M} , to be used as adjacency matrix,¹ for the generation of the coarsened graph \bar{G} .

¹ If g does not divide the observed time series length N , we use the Python function `numpy.linspace` that creates slightly different intervals to keep S an integer [52, 53].

2.3.2 Graph embedding

In the GRL framework we will make use of graph embedding, as a data-driven approach, instead of graph coarsening. Network embedding techniques have made significant contributions to the use of Machine Learning (ML) in network science [54]. Unsupervised feature extraction techniques from graph data are gaining popularity in the ML domain [35, 55, 56]. These methods automatically extract features which can be used as inputs for link prediction, node and graph classification, community detection, and a variety of other tasks in a wide range of real-world research and application contexts [35, 55, 57–59]. These graph mining technologies contributed in a significant way to the advancement and development of ML [60, 61]. In particular, node embeddings transform graph vertices to a Euclidean space, where nodes that are related according to a specific definition of closeness are close to each other. The Euclidean format, instead of the native graph, facilitates the use of conventional ML tools [37]. In this paper, we adopt graph embedding procedures belonging to the four classes of the Karate Club framework [37]: (i) neighborhood preserving embedding; (ii) structural embedding; (iii) attributed embedding; (iv) meta-embedding. While neighborhood preserving embeddings preserve the closeness of graph nodes, structural embeddings preserve the structural roles of nodes in the embedding space, and attributed embeddings maintain the neighborhood, the structure and the generic attribute similarity of nodes. Meta-embeddings are designed to produce embeddings with a higher representation quality. To ensure generality and completeness, in the empirical analysis we will use one method chosen from each class. The main characteristics of these methods are described below.

Diff2Vec is a neighborhood preserving embedding that uses diffusion processes on graphs to create node sequences with the aim of training a neural network. The weights of the trained neural network determine the embedding of the nodes. Experiments on community discovering revealed that this method detect high-quality communities [39].

GraphWave is a structural embedding that preserves the structural roles of nodes in the embedding space [40]. Different portions of a graph may contain nodes with comparable structural roles within their local network architecture. The discovery of such roles provides crucial information into the structure of networks. GraphWave computational complexity is proportional to the number of edges, enabling it to scale to huge networks.

Attributed social network embeddings (ASNE) enhance all the previous approaches focusing on neighborhood

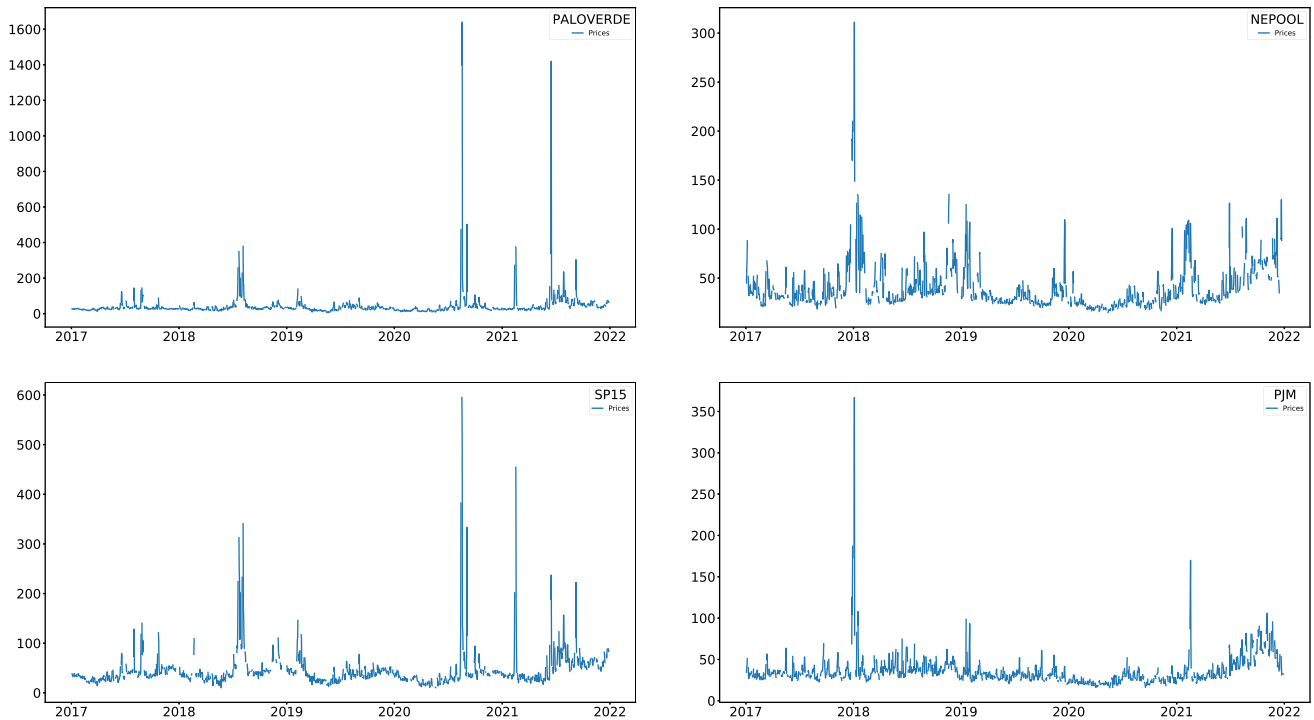


Fig. 3 Observed time series in the time interval January 1, 2017, to December 31, 2021. Data are expressed in nominal dollars per megawatt-hour

closeness and structural information [41]. For networks usually exist supplementary details that are referred to as features or attributes, which comprise not only the node information (adjacency matrix) but also node features (attributes), related to the node context. In essence, node attributes have enormous effects on the organization of networks. By studying attribute homophily and network structure together, it is possible to learn informative node representations.

Network embedding update (NEU) belong to the meta-embedding class [42]. It is an algorithm designed to improve the performance of any given network representation learning. The running time of NEU is almost negligible. We will apply the NEU meta-embedding method to any of the previously listed methods.

2.4 Graph partitioning

This section is devoted to describe the different strategies to perform a graph mining task, namely the community detection, in both the GA and GRL frameworks. In general, a network is regarded to have a community structure if its nodes can be divided into groups of highly connected internal nodes compared to the connections with the external nodes. Communities can be classified as overlapping or nonoverlapping, depending on whether a node can belong to more than one community or only one. In our scenario, we adopted the latter option. In the GA

framework, the Louvain method was applied on the coarsened graph; the GRL framework uses a clustering technique on the embedded multi-dimensional Euclidean space.

2.4.1 The louvain method

To identify communities, we used the Louvain method [43] in the GA framework. The Louvain method provides an unsupervised technique of community discovery based on the optimization of the graph modularity that does not need the number of communities or their sizes as inputs. Modularity is the ratio of the density of connections inside clusters to the density of connections between clusters. The choice of the Louvain algorithm is justified by the fact that it is well documented in the literature that this strategy performs very well on a variety of community detection benchmarks [62]. By partitioning the whole time series into communities, the Louvain method finds the number of mixture model components C by matching it to the number of observed communities.

2.4.2 ToMATo clustering

Since the graph embedding generates a high-dimensional Euclidean representation of the graph, we have chosen TDA, an emerging field of research with the aim of providing mathematical and algorithmic tools to understand

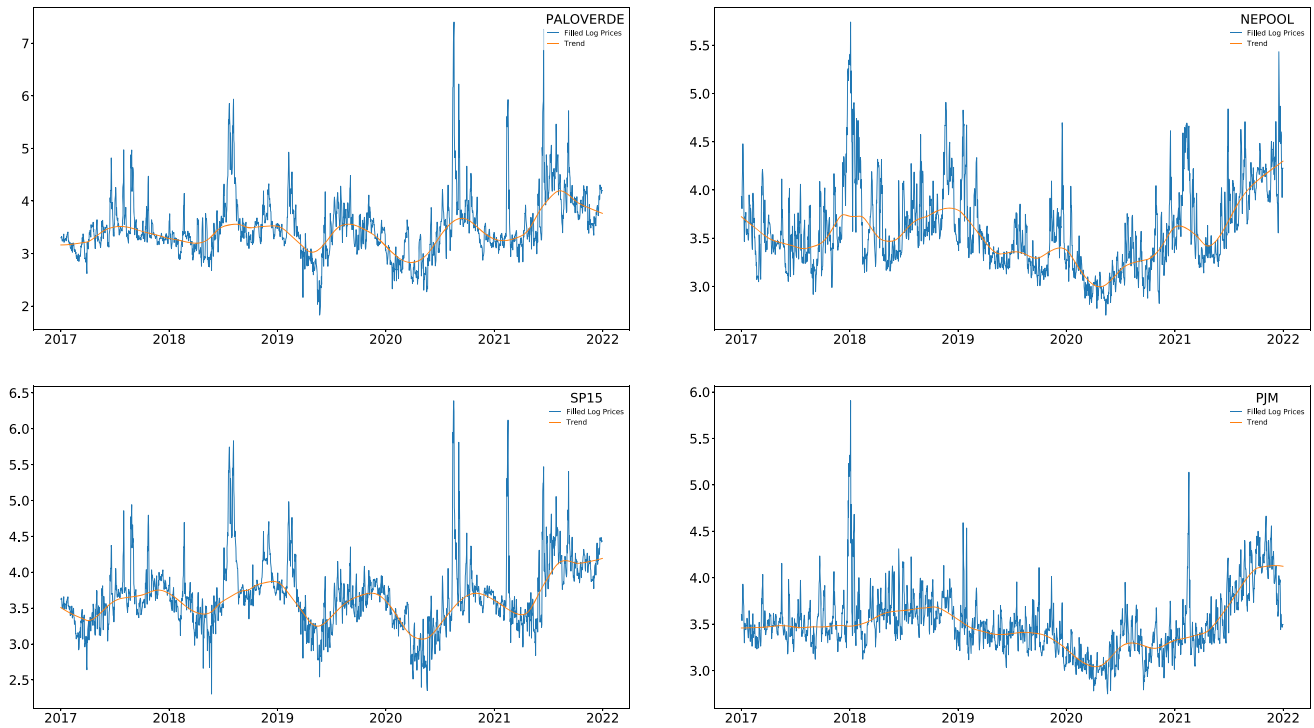


Fig. 4 Filled time series $\{x_t^f\}$ (in blue) and their trends $\{x_t^r\}$ (in red) (Color figure online)

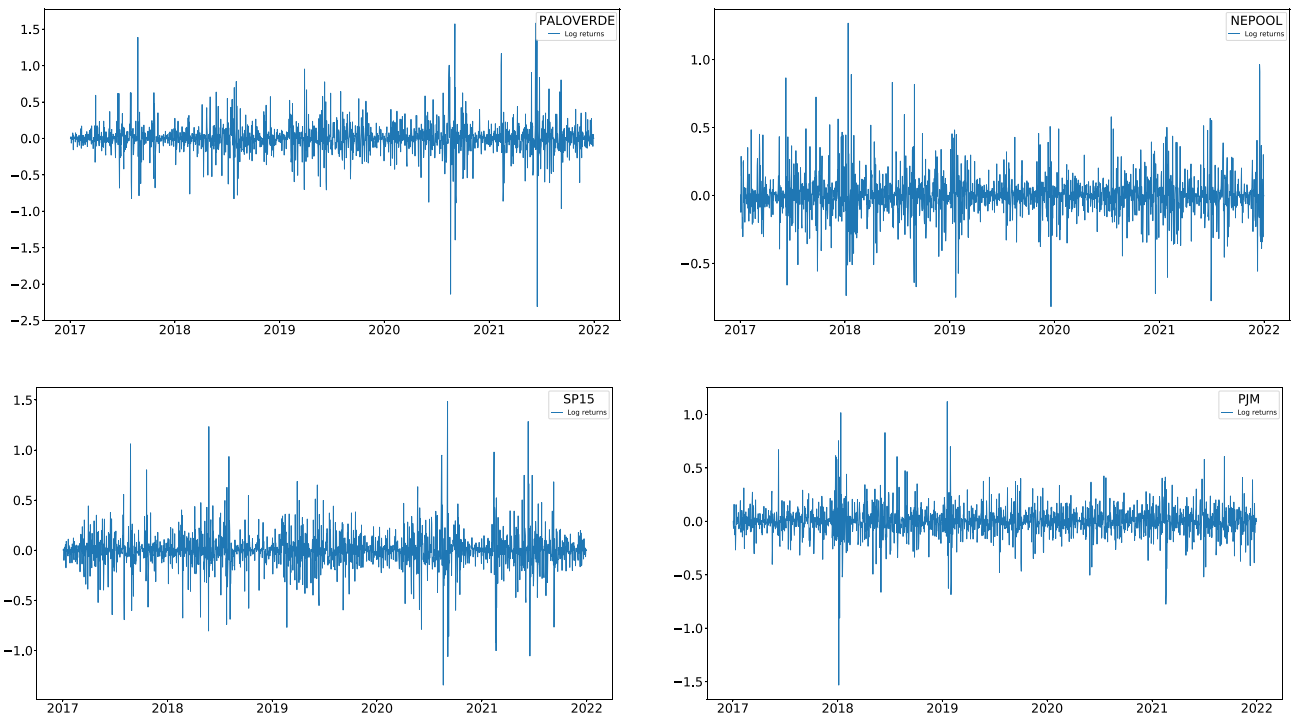


Fig. 5 Log-return time series

the topological and geometric structure of data, particularly suited for high-dimensional data [63, 64]. The ToMATo clustering technique is an unsupervised TDA tool that we used in the GRL framework for community detection.

Theoretically, this technique allows us to identify clusters that are stable under small perturbations of the input [44, 65, 66].

Table 2 Descriptive statistics of log-returns

	Mean	Std. dev.	Skew	Kurt
PALOVERDE	0.00016	0.23	-0.38	19.84
NEPOOL	-0.00016	0.17	0.51	9.22
SP15	0.00007	0.19	0.29	12.84
PJM	-0.00042	0.15	-0.20	16.02

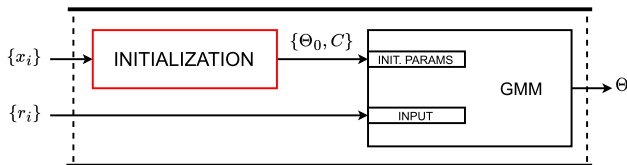


Fig. 6 The distribution estimation section

Table 3 Graph embedding experimental setup

Framework	Embedding method	Additional attributes	Embedding dimension
GRL	Diff2Vec	-	128
GRL	GraphWave	-	400
GRL	ASNE	{r _i }	128
GRL	NEU_Diff2Vec	-	128
GRL	NEU_GraphWave	-	400
GRL	NEU_ASNE	{r _i }	128

3 The experiment

In this section, Gaussian mixture models are used to perform an experiment on US wholesale electricity prices time series to assess the performance of the GA and GRL frameworks. Our data collection is composed by daily prices observed in the time interval January 1, 2017, to December 31, 2021. Time series data can be freely downloaded from www.eia.gov/electricity/wholesale. Observed time series are shown in Fig. 3.

Volatility and extreme unpredictability characterize market price movements, which are frequently accompanied by sharp spikes and jumps generated by changes in the supply–demand balance. All these time series show irregular sampling (as a result of weekends, holidays and other missing data due to market specific reasons) as well as trend and seasonality. Typically, electricity prices may be higher in winter and in summer, and the seasonal component must account for this semiannual periodicity. A trend must be included to account for expected inflation and

possibly for a real power price escalation rate (positive or negative). Data preprocessing was first conducted to fill in time series gaps and remove observable trend and seasonality over time.

3.1 Data preprocessing

Let us denote by $\{y_t^{ob}\}$ the time series of daily prices defined as a function of the incomplete observed raw time base $\{\bar{t}\}$, and $\{x_t^{ob}\}$ its natural logarithm transform, i.e., $x_t^{ob} = \log y_t^{ob}$. The time series $\{x_t^{ob}\}$ is sampled irregularly (not evenly time interval) since weekends, holidays, and other sporadic missing days are not included in it. Imputation of missing data (gap filling) can be very informative and convey important knowledge [67], counteracting the phenomenon known as informative missingness [68]. In the presence of missing data, different time periods may have different information content. In fact, even when the market is closed, new information can emerge that can influence price dynamics [69, 70].

The first step, therefore, involves the gap filling of the time series. To do this, we employ a complete daily grid and an imputation technique called missForest [71], a ML algorithm for data imputation that is completely agnostic about the data distribution. MissForest is employed to compute a value for each missing point. In this way, following the same procedure proposed in [29], we extending the raw time series $\{x_t^{ob}\}$ on a complete daily time base $\{t\}$,

$$\{x_t^{ob}\} \xrightarrow{\mathcal{F}} \{x_t^f\}, \tag{14}$$

where \mathcal{F} is the application that transforms $\{\bar{t}\}$ in $\{t\}$ and computes missing values using the missForest algorithm to fill the daily-complete grid, mapping $\{x_t^{ob}\}$ to $\{x_t^f\}$.

In the second step, we look for temporal trend and seasonality, hereinafer, trend, that must be eliminated in order to reveal the stochastic process driving the market dynamics. LOWESS (LOcally WEighted Scatterplot Smoothing) is used to perform this task. [72, 73]. LOWESS is a versatile method for removing the trend by fitting basic polynomial models to restricted portions of data. The key benefit of LOWESS over other methods is that it does not need the specification of a global function or the assumption that the data must conform to a certain distribution shape [74]. Figure 4 shows log-price time series, $\{x_t^f\}$, and superimposed the trend, $\{x_t^{tr}\}$, for each market under investigation.

Once detected, the trend can be eliminated from the filled time series, thus obtaining,

$$x_i = x_i^f - x_i^{tr}, \quad i = 1, 2, \dots, N, \tag{15}$$

where we assumed, without loss of generality, that the

Table 4 PALOVERDE

Method	Strategy	Comps	Grid parameters	BIC	ε_4 (%)
NEU_Diff2Vec	Q	3	(40, 13)	-1246.06	37.16
Diff2Vec	Q	3	(10, 81)	-1245.85	33.39
NEU_Diff2Vec	N	3	(10, 8)	-1245.58	35.05
ASNE	N	3	(42, 32)	-1244.91	33.72
Diff2Vec	N	3	(8, 14)	-1244.09	34.53
GraphWave	Q	3	(72, 62)	-1243.59	43.04
ASNE	Q	3	(8, 52)	-1242.80	35.25
NEU_ASNE	Q	3	(8, 79)	-1242.37	35.34
NEU_GraphWave	Q	3	(54, 17)	-1241.96	43.59
GA framework	N	3	(12, 265)	-1241.43	44.66
GA framework	Q	3	(6, 395)	-1241.23	44.17
NEU_ASNE	N	3	(2, 19)	-1240.86	45.67
GraphWave	N	3	(80, 11)	-1240.23	42.70
NEU_GraphWave	N	3	(90, 13)	-1238.91	47.40

complete time grid $\{t\}$ is represented by the first N natural numbers, $\{1, 2, \dots, N\}$. Then, log-returns are computed as the difference between two successive daily log-prices, i.e.,

$$r_i = x_{i+1} - x_i, \quad i = 1, 2, \dots, N - 1. \quad (16)$$

Fig. 5 depicts the log-return time series, $\{r_t\}$, for each market under investigation. Table 2 depicts the descriptive statistics of log-returns. In all four investigated markets, empirical log-return distributions exhibit heavy tails, as indicated by the high values of the kurtosis. The occurrence of jumps and spikes in electricity prices significantly increases the value of the fourth central moment of empirical log-return distributions. From this perspective, it is crucial that a given probabilistic model captures the first four central moments of the empirical distribution of log-returns, especially for financial applications [75, 76].

3.2 Distribution estimation: the experimental setup

In this section, Gaussian mixture models are estimated on market log-returns time series, r_t , by maximum likelihood using the EM algorithm. The initialization will be performed using both the GA and GRL frameworks. Let us focus on the Initialization blocks. Figure 6 shows a magnified version of the distribution estimation section and summarizes the calibration procedure. The Initialization blocks produce both the number of the model component, C , and the initial parameter set, Θ_0 . Indeed, the number of components is set equal to the number of the detected communities, and the initialization parameters are computed through community membership according to

Eqs. (8–10). Then, a first M-step is performed and, subsequently, the EM algorithm is used to estimate the GMM parameter set, Θ .

With respect to the GA framework, the feasibility of the GRL framework is facilitated by the embedding techniques that enable to work directly on a *not reduced* graph. In the GA framework, we performed the model estimation for each couple of values (Q, S) defined on a suitable grid. The number of bins, Q , is made to vary from 2 to 100 in increments of 2 for both quantile and normal binning strategies; the parameter S which defines the dimension of the reduced adjacency matrix \bar{M} is made to vary from 5 to 400 in increments of 5. In the GRL framework, the parameters of the four embedding methods used in this study are the default parameters of the Karate Club framework. Moreover, the ToMATo clustering techniques needs to set three parameters, namely the neighborhood information of the point cloud, the density estimator, and the merging parameter. We used a ToMATo cluster function contained in the Python library `tomaster`² which needs only the neighborhood information as input and automates the other two parameters. Since ToMATo relies heavily on neighborhood information, a popular choice to model neighborhood seems to be the K -nearest neighbors algorithm. As well documented in the literature, this ML algorithm performs well, recovering the correct clusters under an appropriate choice of the parameter K [44]. For the GRL framework the model estimation is, therefore, performed by exploring a suitable two-dimensional grid for the couple (Q, K) . In this grid, the number of bins, Q , is

² `tomaster`: Topological Mode Analysis on Steroids. Github repository at <https://github.com/louisabraham/tomaster>.

Table 5 NEPOOL

Method	Strategy	Comps	Grid parameters	BIC	ε_4 (%)
NEU_Diff2Vec	N	3	(50, 12)	-1896.41	7.38
NEU_Diff2Vec	Q	3	(6, 99)	-1896.15	3.66
NEU_GraphWave	Q	3	(98, 62)	-1896.09	9.46
Diff2Vec	N	3	(6, 97)	-1895.89	8.80
ASNE	Q	3	(2, 19)	-1895.76	7.49
Diff2Vec	Q	3	(68, 29)	-1895.61	4.52
GraphWave	Q	3	(30, 92)	-1895.56	6.18
ASNE	N	3	(16, 30)	-1895.24	4.17
NEU_ASNE	Q	3	(2, 51)	-1894.91	2.49
NEU_ASNE	N	3	(18, 43)	-1893.61	4.79
GraphWave	N	3	(12, 76)	-1893.23	7.47
NEU_GraphWave	N	3	(12, 93)	-1893.23	7.47
GA framework	N	3	(22, 95)	-1891.19	16.15
GA framework	Q	3	(26, 175)	-1889.73	18.91

Table 6 SP15

Method	Strategy	Comps	Grid parameters	BIC	ε_4 (%)
ASNE	N	3	(98, 100)	-1671.45	9.58
NEU_ASNE	N	3	(2, 30)	-1671.39	12.54
NEU_Diff2Vec	Q	3	(56, 29)	-1671.27	6.25
Diff2Vec	Q	3	(6, 75)	-1671.23	15.26
NEU_ASNE	Q	3	(2, 24)	-1671.04	8.77
NEU_Diff2Vec	N	3	(20, 8)	-1670.90	16.78
ASNE	Q	3	(38, 56)	-1670.48	12.44
GA framework	Q	3	(34, 100)	-1667.58	22.70
Diff2Vec	N	3	(20, 10)	-1667.47	0.60
GraphWave	Q	3	(14, 75)	-1667.13	23.03
NEU_GraphWave	Q	3	(82, 46)	-1666.67	23.05
GA framework	N	3	(16, 105)	-1665.84	21.91
GraphWave	N	3	(88, 11)	-1661.61	21.39
NEU_GraphWave	N	3	(42, 22)	-1645.86	33.32

made to vary from 2 to 100 in increments of 2 for both quantile and normal binning strategies; the parameter K from 2 to 100 in increments of 1.

The embedding method, during the exploration of the grid, operates always on the complete graph G , never using an approximate form like the coarsening method that makes use of a reduced adjacency matrix of dimension S . This is one of the most important differences between GA and GRL frameworks. Table 3, reports the dimension of the Euclidean space for each embedding method. Such a parameter has been chosen according to the default one provided by the library. In addition, Table 3 shows, as additional attributes, the log-return time series in

correspondence of two well defined embedding methods, namely the ASNE and NEU_ASNE methods.³ As we will see in the next section, this feature can greatly improve the accuracy of the estimation procedure.

3.3 Results

This section outlines and discusses the empirical findings of the experiment. We will analyze the outcomes of six embedding methods for the GRL framework (see Table 3)

³ We extended the applicability of the NEU meta-embedding to the ASNE approach as well. The method is contained in our repository <https://bit.ly/3zFtbQY>.

Table 7 PJM

Method	Strategy	Comps	Grid parameters	BIC	ε_4 (%)
ASNE	N	3	(10, 59)	-2379.52	23.01
NEU_Diff2Vec	Q	3	(2, 13)	-2379.51	20.67
NEU_ASNE	N	3	(2, 39)	-2379.46	21.81
Diff2Vec	Q	3	(42, 7)	-2378.98	24.11
NEU_ASNE	Q	3	(8, 11)	-2378.45	23.70
ASNE	Q	3	(84, 99)	-2378.27	20.53
NEU_Diff2Vec	N	3	(78, 16)	-2378.11	18.13
NEU_GraphWave	Q	3	(48, 65)	-2378.11	24.72
GraphWave	Q	3	(96, 8)	-2376.66	26.99
Diff2Vec	N	3	(2, 12)	-2368.06	33.19
GA framework	N	3	(14, 255)	-2367.62	3.71
GA framework	Q	3	(36, 385)	-2361.95	43.88
GraphWave	N	3	(8, 43)	-2361.71	32.34
NEU_GraphWave	N	3	(8, 33)	-2361.71	32.34

Table 8 *K*-means results

	Comps	BIC	ε_4 (%)
PALOVERDE	3	-1236.74	36.24
NEPOOL	2	-1871.05	33.28
SP15	4	-1638.10	3.31
PJM	4	-2337.67	8.05

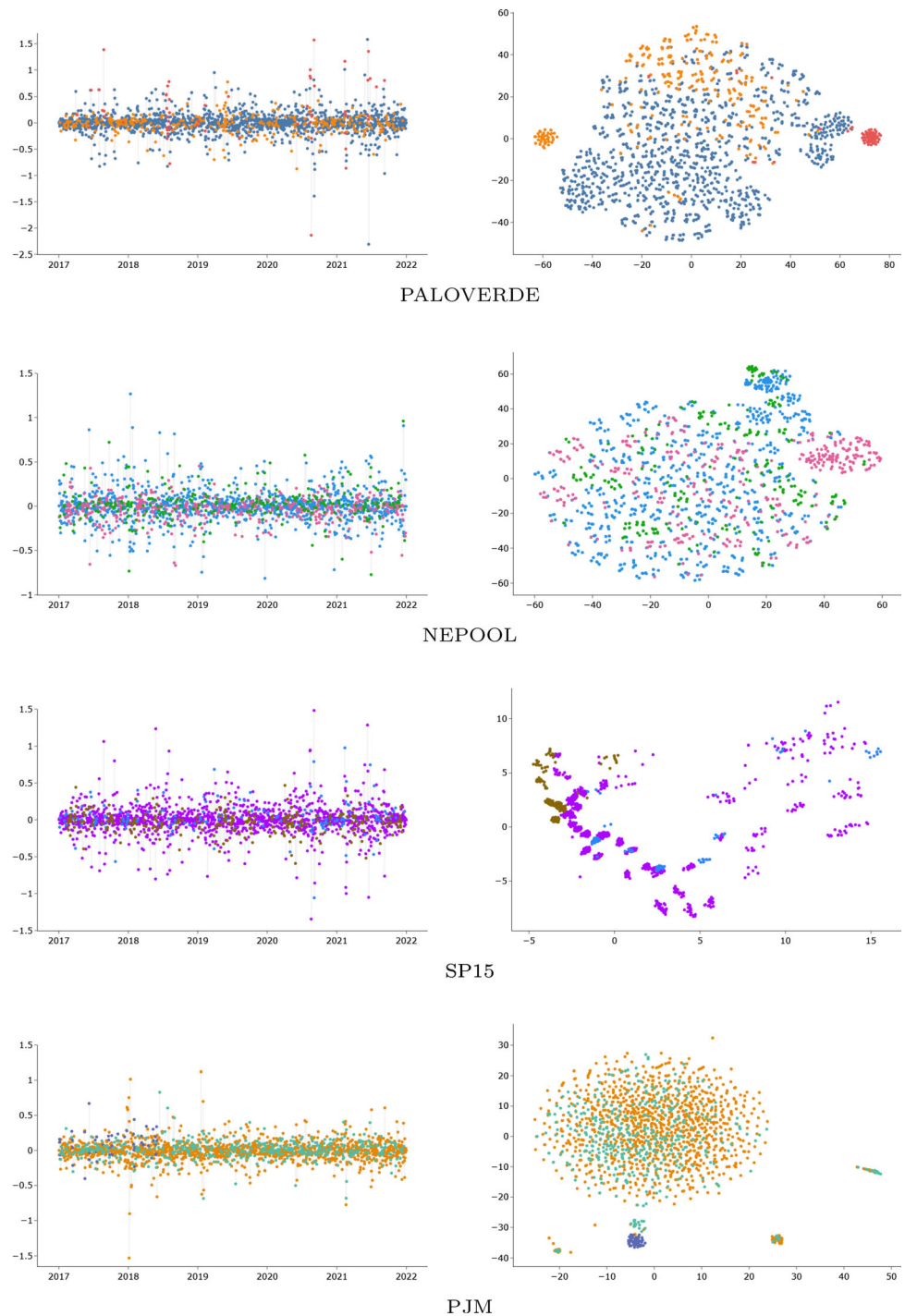
and one coarsening method for the GA framework. Due to the fact that each method includes two different binning strategies (Quantile and Normal), the comparison will involve 14 outcomes corresponding to 14 different combinations, namely 6 GRL-Q, 6 GRL-N, 1 GA-Q, 1 GA-N combinations. For each of these 14 combinations, we determined the best performing model according to the BIC (Bayesian information criterion) [77, 78]. By model we mean the combination of method and strategy, the grid parameters and the number of Gaussian components determined by the partitioning technique. Therefore, each model uniquely identifies the initialization point of the maximum likelihood estimation procedure. By varying the values of the grid parameters, the best performing model is the one that minimizes the BIC value. We recall that the use of the BIC allows us to resolve the model selection problem by introducing a penalty term for the number of parameters, dealing with the trade off between the quality of fit and the simplicity of the model. The penalty discourages overfitting problems because increasing the number of parameters in the model always improves the quality of the fit. Tables 4, 5, 6 and 7 show, for each combination (method+strategy), the best performing model (method+strategy+numbers of Gaussian

components+grid parameters). The models are sorted by increasing BIC value, so that the first row of each table shows the top-performing model (the model with the lowest BIC value) among the 14 selected models. For each model, the kurtosis absolute percentage error, ε_4 is also reported in order to measure the capability of the model to capture also the heavy tail phenomenon. ε_4 is calculated as the absolute percent difference between the kurtosis of the empirical log-return distribution and the kurtosis computed by the model. In this regard, we recall that the moments of the GMM distribution can be calculated exactly. It should be emphasized that in all four markets under consideration, the best fitting model according to the BIC is a three-component model in both the GA and GRL frameworks.

Table 8 shows the estimation results obtained using a *K*-means based initialization technique. Our methodology outperforms this more conventional approach. In fact, for each market the BIC value obtained with the this initialization technique is greater than the maximum BIC value reported in the last row of Tables 4, 5, 6 and 7.

Figure 7 shows for each market the log-return time series in which log-returns belonging to the communities identified to initialize the EM algorithm in the case of the top-performing model are represented with the same color (left panel). In addition, a two-dimensional reduction of the multidimensional embedding of the top-performing model is also shown (right panel). The reduction has been performed by t-distributed stochastic neighbor embedding (t-SNE) which is a well-suited technique for the visualization of high-dimensional datasets [79]. The parameters of the top-performing models are provided in Table 9. The same table also shows the colors of the communities and the

Fig. 7 Community representations of the top-performing models. Colored log-return time series (left panel). Two-dimensional reduction of the embeddings (right panel) (Color figure online)



number of log-returns belonging to the same community (count).

Table 10 displays for each market, a selection of three-components models chosen among the models that minimize the BIC value in grid exploration. The model with the best BIC is listed, of course, in the first row of each sub-table. The models listed in the other rows were chosen to provide a trade-off between the value of BIC and the value

of the kurtosis error coefficient ε_4 . The significance of this trade-off should be noted. The first row provides a better match according to the BIC, reproducing well the first three moments of the empirical log-return distributions. In contrast, the last row of each sub-table provides a very interesting compromise reproducing well also the fourth moment.

Table 9 Top-performing models: the parameters of the Gaussian mixture distribution













		α_c	μ_c	σ_c^2	Count
PALOVERDE		0.57114370	-0.00051137	0.02063759	1440
		0.28607423	-0.00428902	0.00195742	327
		0.14278207	0.01178404	0.27665900	58
NEPOOL		0.46221626	-0.01529096	0.02193517	1099
		0.39300628	0.00168726	0.00228042	447
		0.14477746	0.04316237	0.11832773	279
SP15		0.56324985	-0.00037009	0.02051393	1398
		0.12448871	0.00743187	0.20415022	267
		0.31226144	-0.00207479	0.00150556	160
PJM		0.59942954	0.00103331	0.01657649	1326
		0.07124184	0.00647185	0.16030051	444
		0.32932862	-0.00456474	0.00206121	55

Table 10 The trade-off between BIC and ε_4

Method	Strategy	Grid params	BIC	ε_4 (%)	Mean	Standard deviation	Skewness	Kurtosis
<i>PALOVERDE</i>								
NEU_Diff2Vec	Q	(40, 13)	-1246.06	37.16	1.6e-04	0.23	0.11	12.47
NEU_Diff2Vec	Q	(42, 95)	-1215.23	8.29	1.6e-04	0.23	0.32	18.20
NEU_Diff2Vec	Q	(42, 94)	-1214.12	6.53	1.6e-04	0.23	0.34	18.54
GA framework	N	(6, 335)	-1212.52	2.38	1.6e-04	0.23	0.39	19.37
NEU_Diff2Vec	N	(40, 6)	-1207.47	1.48	1.6e-04	0.23	0.35	19.55
<i>NEPOOL</i>								
NEU_Diff2Vec	N	(50, 12)	-1896.41	7.38	-1.6e-04	0.17	0.37	8.54
NEU_Diff2Vec	Q	(2, 34)	-1895.94	2.04	-1.6e-04	0.17	0.41	9.04
NEU_Diff2Vec	Q	(6, 96)	-1895.58	0.72	-1.6e-04	0.17	0.41	9.16
NEU_Diff2Vec	N	(34, 13)	-1895.35	0.15	-1.6e-04	0.17	0.42	9.24
NEU_Diff2Vec	N	(32, 23)	-1894.76	0.10	-1.6e-04	0.17	0.39	9.22
<i>SP15</i>								
ASNE	N	(98, 100)	-1671.45	9.58	7e-05	0.19	0.07	11.61
NEU_Diff2Vec	Q	(56, 29)	-1671.27	6.24	7e-05	0.19	0.07	12.04
NEU_ASNE	Q	(2, 22)	-1670.45	3.58	7e-05	0.19	0.06	12.38
NEU_Diff2Vec	Q	(6, 80)	-1668.98	0.64	7e-05	0.19	0.05	12.92
NEU_Diff2Vec	Q	(64, 92)	-1668.64	0.08	7e-05	0.19	0.05	12.86
<i>PJM</i>								
ASNE	N	(10, 59)	-2379.52	23.01	-4e-04	0.15	0.08	12.33
ASNE	N	(14, 13)	-2374.44	9.04	-4e-04	0.15	0.08	14.57
GA framework	N	(14, 255)	-2367.62	3.71	-4e-04	0.15	0.11	15.42
GA framework	N	(12, 290)	-2363.15	1.72	-4e-04	0.15	0.13	15.74
GA framework	N	(12, 390)	-2362.55	0.15	-4e-04	0.15	0.14	15.99

Finally, we remark that the NEU approach performs very well, matching several rows in Table 10, in particular when it is combined with the Diff2Vec embedding method. The ASNE approach that makes use of extra information carried on by log-return time series as additional attribute,

also provides good results. However, the GA framework remains a plausible option, providing interesting results for Paloverde and PJM log-return distribution estimates.

4 Concluding remarks

We presented a superframework to handle the initialization problem of the EM algorithm for mixture models. Our innovative method emerges from a comparison with a prior study [27], which we transferred and expanded in a general superframework based on graph machine learning. We applied the proposed methodology to estimate Gaussian mixture models on US wholesale electricity market prices using two different configurations of the superframework, namely the GA and the GRL frameworks. Electricity market prices offer an excellent example of a complex system that transcends simple modeling. Using complex networks approaches that link time series and graphs, we were able to exploit graph-structured information derived from electricity market data. We have shown that the GA framework performs better than conventional initialization methods, such as k -means and random-based techniques [27]. In this paper, we demonstrated that the GRL framework provides even more accurate results. Indeed, the GRL framework, incorporating data-driven and unsupervised graph embedding approaches in conjunction with TDA-based clustering, has less information loss and a more conservative design than the GA framework. We found that structural information can definitely capture the behavior of time series and the nonlinear relationships between different observations. In particular, in the GRL framework we highlight how the combination of NEU meta-embedding applied to Diff2Vec, and the ASNE method stand out as especially effective embedding models.

The proposed superframework proves to be a very flexible tool of analysis that can be further developed. Indeed, the ability to incorporate a wide variety of methods within its functional building blocks makes it extremely adaptable to very different empirical situations. From this point of view, three main directions of our future research will be: (i) since the output of the graph encoding block using MTF can be interpreted as an image [30, 31], we can exploit an image classification transfer learning technique [80] to generate the embedding in the graph compression step and use an image pretrained VGG-16 deep convolutional neural network [81] for this feature extraction; (ii) combine embedding and coarsening procedures in the graph compression block, as it is done in other contexts [82, 83], and test the accuracy of the results; (iii) use the initialization blocks as a self-supervised learning framework to discover communities and assign them to different dynamics [84, 85]. We have left these topics for future study.

Funding Open access funding provided by Università degli Studi G. D'Annunzio Chieti Pescara within the CRUI-CARE Agreement.

Data Availability The US wholesale electricity prices time series data that support the findings of this study are available from repository Wholesale Electricity and Natural Gas Market Data, <https://www.eia.gov/electricity/wholesale/>.

Declarations

Conflict of interest The authors declare no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Albert R, Barabási A (2002) Statistical mechanics of complex networks. *Rev Modern Phys* 74(1):47–97. <https://doi.org/10.1103/RevModPhys.74.47>
2. Vespignani A (2018) Twenty years of network science. *Nature* 558:528–529
3. Hyttinen A, Plis S, Järvisalo M, Eberhardt F, Danks D (2017) A constraint optimization approach to causal discovery from subsampled time series data. *Int J Approx Reason* 90:208–225. <https://doi.org/10.1016/j.ijar.2017.07.009>
4. Xie Y, Chen C, Gong M, Li D, Qin AK (2021) Graph embedding via multi-scale graph representations. *Inf Sci* 578:102–115. <https://doi.org/10.1016/j.ins.2021.07.026>
5. Laengle S, Lobos V, Merigó JM, Herrera-Viedma E, Cobo MJ, De Baets B (2021) Forty years of fuzzy sets and systems: a bibliometric analysis. *Fuzzy Sets Syst* 402:155–183. <https://doi.org/10.1016/j.fss.2020.03.012>
6. Newman ME (2003) The structure and function of complex networks. *SIAM Rev* 45(2):167–256
7. Newman M (2010) *Networks: an introduction*. Oxford University Press, Oxford, pp W677–W682
8. Yang Y, Yang H (2008) Complex network-based time series analysis. *Phys Statist Mech Appl* 387(5–6):1381–1386
9. da Fontoura Costa L, Rodrigues FA, Travieso G, Boas PRV (2005) Characterization of complex networks: a survey of measurements. *Adv Phys* 56:167–242
10. Zou Y, Donner RV, Marwan N, Donges JF, Kurths J (2019) Complex network approaches to nonlinear time series analysis. *Phys Rep* 787:1–97. <https://doi.org/10.1016/j.physrep.2018.10.005>. (Complex network approaches to nonlinear time series analysis)
11. Silva VF, Silva ME, Ribeiro P, Silva F (2021) Time series analysis via network science: concepts and algorithms. *WIREs Data Min Knowl Discov* 11(3):1404. <https://doi.org/10.1002/widm.1404>
12. Tofallis C (2008) Selecting the best statistical distribution using multiple criteria. *Comput Ind Eng* 54(3):690–694. <https://doi.org/10.1016/j.cie.2007.07.016>

13. Wang Y, Yam RCM, Zuo MJ (2004) A multi-criterion evaluation approach to selection of the best statistical distribution. *Comput Ind Eng* 47(2–3):165–180. <https://doi.org/10.1016/j.cie.2004.06.003>
14. Li L, Kumar Damarla S, Wang Y, Huang B (2021) A gaussian mixture model based virtual sample generation approach for small datasets in industrial processes. *Inf Sci* 581:262–277. <https://doi.org/10.1016/j.ins.2021.09.014>
15. Chen Y, Cheng N, Cai M, Cao C, Yang J, Zhang Z (2021) A spatially constrained asymmetric gaussian mixture model for image segmentation. *Inf Sci* 575:41–65. <https://doi.org/10.1016/j.ins.2021.06.034>
16. Ramos-López D, Masegosa AR, Salmerón A, Rumí R, Langseth H, Nielsen TD, Madsen AL (2018) Scalable importance sampling estimation of gaussian mixture posteriors in bayesian networks. *Int J Approx Reason* 100:115–134. <https://doi.org/10.1016/j.ijar.2018.06.004>
17. Quost B, Deneux T (2016) Clustering and classification of fuzzy data using the fuzzy em algorithm. *Fuzzy Sets Syst* 286:134–156. <https://doi.org/10.1016/j.fss.2015.04.012>. (Theme: Images and Clustering)
18. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the em algorithm. *J Royal Statist Soc SerB (Methodolog)* 39(1):1–22. <https://doi.org/10.1111/j.2517-6161.1977.tb01600.x>
19. Baudry J-P, Celeux G (2015) Em for mixtures: initialization requires special care. *Statist Comput*. <https://doi.org/10.1007/s11222-015-9561-x>
20. McLachlan GJ, Peel D (2000) *Finite mixture models series in probability and statistics*. Wiley, New York
21. Hipp J, Bauer D (2006) Local solutions in the estimation of growth mixture models: Correction to hipp and bauer (2006). *Psycholog Methods* 11:305–305. <https://doi.org/10.1037/1082-989X.11.3.305>
22. Shireman E, Steinley D, Brusco M (2015) Examining the effect of initialization strategies on the performance of gaussian mixture modeling. *Behav Res Methods*. <https://doi.org/10.3758/s13428-015-0697-6>
23. Steinley D, Brusco M (2011) Evaluating mixture modeling for clustering: recommendations and cautions. *Psycholog Methods* 16:63–79. <https://doi.org/10.1037/a0022673>
24. Biernacki C, Celeux G, Govaert G (2003) Choosing starting values for the em algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computat Statist Data Anal* 41(3–4):561–575
25. Yu L, Yang T, Chan AB (2019) Density-preserving hierarchical em algorithm: simplifying gaussian mixture models for approximate inference. *IEEE Trans Patt Anal Mach Intell* 41(6):1323–1337. <https://doi.org/10.1109/TPAMI.2018.2845371>
26. Panić B, Klemenc J, Nagode M (2020) Improved initialization of the em algorithm for mixture model parameter estimation. *Mathematics* 8:373. <https://doi.org/10.3390/math8030373>
27. Mari C, Baldassari C (2022) Unsupervised expectation-maximization algorithm initialization for mixture models: a complex network-driven approach for modeling financial time series. *Inf Sci* 617:1–16. <https://doi.org/10.1016/j.ins.2022.10.073>
28. Voit J (2013) *The statistical mechanics of financial markets. Theoretical and mathematical physics*. Springer, Berlin, pp 08–52
29. Mari C, Baldassari C (2021) Ensemble methods for jump-diffusion models of power prices. *Energies*. <https://doi.org/10.3390/en14082084>
30. Campanharo A, Sire M, Malmgren R, Ramos F, Amaral L (2011) Duality between time series and networks. *PloS one* 6:23378. <https://doi.org/10.1371/journal.pone.0023378>
31. Pineda AM, Ramos FM, Betting LE, Campanharo AS (2020) Quantile graphs for eeg-based diagnosis of alzheimer’s disease. *PloS one* 15(6):0231169
32. Zhang R, Zheng F, Min W (2018) Sequential behavioral data processing using deep learning and the markov transition field in online fraud detection. *arXiv preprint arXiv:1808.05329*
33. Hansen F, Elliott H (1982) Image segmentation using simple markov field models. *Comput Graph Image Process* 20(2):101–132
34. Cai C, Wang D, Wang Y (2021) Graph coarsening with neural networks. *arXiv preprint arXiv:2102.01350*
35. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. KDD ’14*, pp. 701–710. ACM, New York, NY, USA. <https://doi.org/10.1145/2623330.2623732>
36. Sun Z, Deng Z, Nie J-Y, Tang J (2019) Rotate: knowledge graph embedding by relational rotation in complex space. *arXiv preprint ArXiv:abs/1902.10197*
37. Rozemberczki B, Kiss O, Sarkar R (2020) Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In: *Proceedings of the 29th ACM international conference on information and knowledge management (CIKM ’20)*, pp. 3125–3132. ACM
38. Girvan M, Newman ME (2002) Community structure in social and biological networks. *Proceed Natl Acad Sci* 99(12):7821–7826
39. Rozemberczki B, Sarkar R (2020) Fast sequence-based embedding with diffusion graphs. *CoRR* [ArXiv:abs/2001.07463](https://arxiv.org/abs/2001.07463)
40. Donnat C, Zitnik M, Hallac D, Leskovec J (2018) Learning structural node embeddings via diffusion wavelets. pp 1320–1329. <https://doi.org/10.1145/3219819.3220025>
41. Liao L, He X, Zhang H, Chua T-S (2018) Attributed social network embedding. *IEEE Trans Knowl Data Eng* 30(12):2257–2270. <https://doi.org/10.1109/tkde.2018.2819980>
42. Yang C, Sun M, Liu Z, Tu C (2017) Fast network embedding enhancement via high order proximity approximation, 3894–3900. <https://doi.org/10.24963/ijcai.2017/544>
43. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. *J Statist Mech Theory Exp* 2008(10):10008. <https://doi.org/10.1088/1742-5468/2008/10/p10008>
44. Chazal F, Guibas LJ, Oudot SY, Skraba P (2013) Persistence-based clustering in riemannian manifolds. *J ACM*. <https://doi.org/10.1145/2535927>
45. Skrlj B, Kralj J, Lavrac N (2019) Embedding-based silhouette community detection. *CoRR* [ArXiv:abs/1908.02556](https://arxiv.org/abs/1908.02556)
46. Cohen-Steiner D, Edelsbrunner H, Harer J (2007) Stability of persistence diagrams. *Discrete Computat Geom* 37(1):103–120. <https://doi.org/10.1007/S00454-006-1276-5>
47. Chazal F, De Silva V, Glisse M, Oudot S (2016) *The structure and stability of persistence modules*, vol 10. Springer, Berlin
48. Cohen-Steiner-Marc FC, Oudot GG (2008) Proximity of persistence modules and their diagrams
49. Smyth P (2021) Mixture Models and the EM Algorithm. https://www.ics.uci.edu/~smyth/courses/cs274/notes/mixture_models_EM.pdf
50. Liu L, Wang Z (2018) Encoding temporal markov dynamics in graph for visualizing and mining time series. In: *workshops at the Thirty-Second AAAI conference on artificial intelligence*
51. Chen J, Saad Y, Zhang Z (2022) Graph coarsening: from scientific computing to machine learning. *SeMA J* 79(1):187–223
52. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K,

- Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with NumPy. *Nature* 585(7825):357–362. <https://doi.org/10.1038/s41586-020-2649-2>
53. Faouzi J, Janati H (2020) pyts: a python package for time series classification. *J Mach Learn Res* 21(46):1–6
 54. Cui P, Wang X, Pei J, Zhu W (2018) A survey on network embedding. *IEEE Trans Knowl Data Eng* 31(5):833–852
 55. Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y, Jaiswal S (2017) graph2vec: Learning Distributed Representations of Graphs. <https://doi.org/10.48550/ARXIV.1707.05005>. [arXiv.org/abs/1707.05005](https://arxiv.org/abs/1707.05005)
 56. Ye F, Chen C, Zheng Z (2018) Deep autoencoder-like nonnegative matrix factorization for community detection pp 1393–1402. <https://doi.org/10.1145/3269206.3271697>
 57. Perozzi B, Kulkarni V, Chen H, Skiena S (2017) Don't walk, skip!: Online learning of multi-scale network embeddings. *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*
 58. Rozemberczki B, Allen C, Sarkar R (2021) Multi-scale attributed node embedding. *J Complex Netw* 9:14
 59. Yang C, Sun M, Liu Z, Tu C (2017) Fast network embedding enhancement via high order proximity approximation, 3894–3900. <https://doi.org/10.24963/ijcai.2017/544>
 60. Leskovec J, Sosič R (2016) Snap: a general-purpose network analysis and graph-mining library. *ACM Trans Intell Syst Technol (TIST)* 8(1):1
 61. Peixoto T (2014). The graph-tool python library. <https://doi.org/10.6084/M9.FIGSHARE.1164194.V13>
 62. Lancichinetti A, Fortunato S (2009) Community detection algorithms: a comparative analysis. *Phys Rev E* 80:056117. <https://doi.org/10.1103/PhysRevE.80.056117>
 63. Edelsbrunner H, Letscher D, Zomorodian A (2000) Topological persistence and simplification. *Discr Computat Geom* 28:511–533
 64. Zomorodian A, Carlsson G (2005) Computing persistent homology. *Discr Computat Geom* 33(2):249–274
 65. Koontz WLG, Narendra PM, Fukunaga K (1976) A graph-theoretic approach to nonparametric cluster analysis. *IEEE Trans Comput* 25(9):936–944
 66. Koontz WLG, Narendra PM, Fukunaga K (1975) A branch and bound clustering algorithm. *IEEE Trans Comput* C 24(9):908–915. <https://doi.org/10.1109/T-C.1975.224336>
 67. Owen M (2007) *Practical signal processing*. Cambridge University Press, Cambridge
 68. Che Z, Purushotham S, Cho K, Sontag D, Liu Y (2018) Recurrent neural networks for multivariate time series with missing values. *Sci Rep* 8:6085
 69. French KR (1980) Stock returns and the weekend effect. *J Financ Econom* 8:55–69
 70. Mantegna RN, Stanley HE (1999) *Introduction to econophysics: correlations and complexity in finance*. Cambridge University Press, Cambridge
 71. Stekhoven DJ, Bühlmann P (2011) MissForest-non-parametric missing value imputation for mixed-type data. *Bioinformatics* 28(1):112–118. <https://doi.org/10.1093/bioinformatics/btr597>
 72. Cleveland WS (1979) Robust locally weighted regression and smoothing scatterplots. *J Am Statist Assoc* 74(368):829–836. <https://doi.org/10.1080/01621459.1979.10481038>
 73. Cleveland WS, Devlin SJ (1988) Locally weighted regression: an approach to regression analysis by local fitting. *J Am Statist Assoc* 83:596–610. <https://doi.org/10.1080/01621459.1988.10478639>
 74. Dagum EB, Bianconcini S (2016) *Seasonal adjustment methods and real time trend-cycle estimation*. Statistics for social and behavioral sciences. Springer, Berlin
 75. Geman H (2005) *Commodities and commodity derivatives: modeling and pricing for agricultural metals and energy*. The Wiley Finance Series, Wiley
 76. Geman H, Roncoroni A (2006) Understanding the fine structure of electricity prices. *J Bus.* <https://doi.org/10.1086/500675>
 77. Aho K, Derryberry D, Peterson T (2014) Model selection for ecologists: the worldviews of aic and bic. *Ecology* 95(3):631–636. <https://doi.org/10.1890/13-1452.1>
 78. Boubaker O, Jafari S (2018) *Recent advances in chaotic systems and synchronization: from theory to real world applications*, 1st edn. Academic Press Inc, USA
 79. Van der Maaten L, Hinton G (2008) Visualizing data using t-sne. *J Mach Learn Res* 9(11):2579
 80. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q (2020) A comprehensive survey on transfer learning. *Proceed IEEE* 109(1):43–76
 81. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*
 82. Fahrback M, Goranci G, Peng R, Sachdeva S, Wang C (2020) Faster graph embeddings via coarsening. In: *international conference on machine learning*, pp 2953–2963. PMLR
 83. Liang J, Gurukur S, Parthasarathy S (2021) Mile: a multi-level framework for scalable graph embedding. *Proceed Int AAAI Conf Web Soc Media* 15:361–372
 84. Ma Q, Li S, Zhuang W, Li S, Wang J, Zeng D (2021) Self-supervised time series clustering with model-based dynamics. *IEEE Trans Neural Netw Learn Syst* 32(9):3942–3955. <https://doi.org/10.1109/TNNLS.2020.3016291>
 85. Lovrić M, Milanović M, Stamenković M (2014) Algorithmic methods for segmentation of time series: an overview. *J Contem Econom Bus Issues* 1(1):31–53