Journal of Physics: Complexity

TOPICAL REVIEW • OPEN ACCESS

Zoo guide to network embedding

To cite this article: A Baptista *et al* 2023 *J. Phys. Complex.* **4** 042001

IOP Publishing J. Phys. Complex. 4 (2023) 042001 https://doi.org/10.1088/2632-072X/ad0e23

Journal of Physics: Complexity

OPEN ACCESS

RECEIVED
10 July 2023

REVISED
26 October 2023

ACCEPTED FOR PUBLICATION
20 November 2023

PUBLISHED
29 November 2023

TOPICAL REVIEW

Zoo guide to network embedding

A Baptista[1,2,*], R J Sánchez-García[2,3,4], A Baudot[5,6,7] and G Bianconi[1,2,*]

[1] School of Mathematical Sciences, Queen Mary University of London, London E1 4NS, United Kingdom
[2] The Alan Turing Institute, The British Library, London NW1 2DB, United Kingdom
[3] School of Mathematical Sciences, University of Southampton, Southampton SO17 1BJ, United Kingdom
[4] Institute for Life Sciences, University of Southampton, Southampton SO17 1BJ, United Kingdom
[5] Aix-Marseille Univ, INSERM, MMG, Marseille, France
[6] Barcelona Supercomputing Center, Barcelona, Spain
[7] CNRS, Marseille, France
[*] Authors to whom any correspondence should be addressed.

E-mail: anthony.baptista@qmul.ac.uk and ginestra.bianconi@gmail.com

Keywords: network embedding, representation learning, network analysis, higher-order network

Abstract

Networks have provided extremely successful models of data and complex systems. Yet, as combinatorial objects, networks do not have in general intrinsic coordinates and do not typically lie in an ambient space. The process of assigning an embedding space to a network has attracted great interest in the past few decades, and has been efficiently applied to fundamental problems in network inference, such as link prediction, node classification, and community detection. In this review, we provide a user-friendly guide to the network embedding literature and current trends in this field which will allow the reader to navigate through the complex landscape of methods and approaches emerging from the vibrant research activity on these subjects.

1. Introduction

Networks are simple yet powerful and versatile models to represent and analyse complex data and systems across a wide variety of user domains and research fields [1, 2]. In social sciences, social networks are useful for different tasks, such as items or friends classifications, friends recommendations or targeted advertising [3]. Using social networks, community detection or link prediction can help to better understand the spreading process of rumours or epidemics [4]. In biology, link prediction in biological networks is commonly used for predicting new interactions between proteins, new therapeutic applications for existing drugs or new gene–disease associations [5, 6]. Overall, the study of networks as mathematical models has developed into the fully established discipline of network science [1, 2].

Networks are intrinsically combinatorial objects (i.e. interconnected nodes, where certain pairs of nodes are connected by links), with no *a priori* ambient space, nor node geometric information such as 'coordinates'. Network embedding (also known as representation learning) is the process of assigning such an ambient space (called the *latent* or *embedding space*) to a network. This is typically done by mapping the nodes to a geometric space, such as a Euclidean space $\mathbb{R}^n$, while preserving some properties of the nodes, links, and/or network [7]. Overall, network embedding methods are used for learning a low-dimensional vector representation from a high-dimensional (as measured by the number of nodes) network. The relationships between nodes in the network are represented by their distance in the low-dimensional embedding space. Then, the low-dimension vector representation can be used for visualisation, and in a wide variety of downstream analyses, from network inference or link prediction to node classification or community detection. Moreover, network embedding can provide insights into the geometry of the underlying data. For instance, by using measures defined through diffusion process which preserve the geometry of networks [8] or multilayer networks [9]. These insights can be useful for performance improvement, by working on a lower dimensional space, or exploiting the richer geometry of the embedding space. Finally, some downstream analyses, such as machine learning techniques, require a vector representation of the network. In this context, embedding network into a vector space is a prerequisite [10].

Network embedding raises many challenges. First, a fundamental question is which network properties should be preserved by the embedding. For instance, the embedding space may preserve the intra-community similarity, the structural role similarity, or the similarity between nodes labels. A second challenge is related to the choice of dimension. The dimension of the embedding space will be a trade-off between two competing requirements: preserving the information encoded in the original network (favours high-dimensional space representations) and reducing the complexity or noise in the original network (favours low-dimensional space representations). Third, the scalability of network embedding methods is important: embedding methods applied to real networks face low parallelisability and data sparsity issues. Methods need to be efficient for networks of the size of typical modern-day network data sets, that is, up to several million nodes and edges [11–13]. Lastly, the interpretation of the results of network embedding can be difficult [14].

For decades, dimensionality reduction methods based on factorisation matrices appeared as a relevant way to encode topological network information [15–17]. These methods provided an initial set of network embedding techniques due to the success and ubiquity of network models. Over the past few years, there has been a significant surge in the number of embedding methods, making it challenging to navigate this fast-evolving field. The purpose of this review is to provide an overview based on a novel taxonomy that extends previous ones [18, 19] and describe current trends in network embedding.

First, we introduce the basic concept of network embedding and the state-of-the-art taxonomies of these methods. Next, we present our own taxonomy based on the common mathematical processes that underlie the embedding methods. This taxonomy aims to assist readers in navigating the field. We describe the two well-established classes of methods: the shallow embedding methods, and the deep learning methods. In addition to these two classical approaches, we include two sections dedicated to the higher-order network embedding methods and the emerging network embedding methods. These sections highlight current trends in the field, although the taxonomy is broad enough to integrate these new methods. Finally, we illustrate the wide range of network embedding applications, with one section devoted to the classical applications that include a user guideline and another section dedicated to the emerging applications that are currently growing in popularity.

2. Definitions and preliminaries

A *network*, defined formally as a pair $G = (V, E)$, consists of a non-empty set V of *vertices* (or *nodes*), and a set of *edges* (or *links*) E connecting certain pairs of nodes. In the case of undirected networks, we can define E as a subset of $\{\{u, v\} \mid u, v \in V\}$, and call $\{u, v\} \in E$ an *undirected edge between vertices u and v*, so that $\{u, v\} = \{v, u\}$. In the case of directed networks, we can define $E \subseteq V \times V$, and call $(u, v) \in E$ a *directed edge from vertex u to vertex v*, so that $(u, v) \neq (v, u)$. If we agree on a labelling of the vertices, $V = \{v_1, \ldots, v_n, \ldots\}$, we can write $e_{ij} \in E$ for a vertex between v_i and v_j (undirected case) or from v_i to v_j (directed case). Depending on the network model, we can also add node or edge weights and types (see below).

In its simplest form, a network embedding maps each node of a network to a space X, typically a Euclidean vector space $X = \mathbb{R}^d$ with $d \ll n$ the number of nodes. This space is called the *latent space* or *embedding space*. In the latent space, certain properties (of the nodes, edges, or the whole network) are preserved. Hence, a network embedding (into $X = \mathbb{R}^d$) is a mapping function

$$\begin{aligned} f\colon V &\to \mathbb{R}^d \\ v_i &\mapsto z_i. \end{aligned}$$

The embedding vector z_i is expected to capture the topological properties of the original network while reducing the network dimension n. Network embedding methods can embed different components of the network. The previous definition is describing the most common embedding, namely node embedding method. In node embedding methods, each node of the network is embedded into an embedding space X, typically a reduced vectorial representation, that is, a mapping function $V \to X$. However, some methods handle edge embeddings $E \to X$, where each edge of the network is embedded into an embedding space X. Other embedding methods target subgraph or whole-network embedding, where the whole network, or some of its parts, are projected into an embedding space, such as a vector space.

To design efficient network embedding methods, several criteria need to be considered:

(i) **Adaptability**: embedding methods need to be applicable to different data and task, without, for instance, repeating a learning step.
(ii) **Scalability**: embedding methods need to process large-scale networks in a reasonable time.

(iii) **Topology awareness**: the distance between nodes in latent space should reflect the connectivity and/or homophily (similar nodes in a network will be close in the embedding space) of the nodes in the original network. The homophily is the tendency of nodes to be connected to similar nodes.
(iv) **Low dimensionality**: embedding methods should reduce the dimension of the network, by mapping a network with n nodes to a d-dimensional space, with $d \ll n$.
(v) **Continuity**: the latent space should be continuous, which is beneficial in some tasks like classification [20].

As mentioned, while reducing the dimension, the embedding space should preserve some node, edge, and/or network properties. Focusing on node properties, the most common properties preserved by network embedding methods include:

- **The first-order similarity** between two vertices, which is the pairwise similarity between the vertices. In other words, the weight of the edge between vertices defines a first-order similarity measure. Let s_{v_i} (respectively s_{v_j}) be the first-order vector similarity associated with the node v_i (resp. v_j) to every other node in the network.
- **The second-order similarity** between two vertices, which considers the similarity of vertices in terms of neighbourhood structures. The second-order similarity between the nodes v_i and v_j is defined as the similarity between the first-order vectors s_{v_i} and s_{v_j}. Higher-order similarities are based on the same idea. These second or higher-order similarities define structural equivalence between nodes.
- **The regular equivalence similarity**, which defines the similarity between vertices that share common roles in their neighbourhood, i.e. that have similar local network structures. For instance, if a node is a bridge between two communities, or if a node belongs to a clique. The regular equivalence aims to unveil the similarity between distant vertices which share common roles, in contrast to to common neighbourhood.
- **The intra-community similarity**, which defines the similarity between vertices in the same community. The intra-community similarity aims to preserve the cluster structure information of the network.

Embedding methods are often designed to use specific types of networks as input. These network types include:

- **Homogeneous networks**, which correspond to the standard definition of networks mentioned above $G = (V, E)$, where V is a non-empty set of vertices (nodes) and E a set of (directed, or undirected) edges (links). A more general setup, which allows multi-edges, is $G = (V, E, s, t)$ where $V \neq \emptyset$ and E are arbitrary (vertex, edge) sets, and $s, t\colon E \to V$ are the source, respectively target, functions. Homogeneous networks can also be weighted: an edge, respectively a vertex, a weight function is a function $w_E\colon E \to X_E$, respectively $w_V\colon V \to X_V$, where X_E and X_V are weight sets, typically numeric $X_V = X_E = \mathbb{R}$.
- **Heterogeneous networks.** In homogeneous networks, the nodes and the edges are all the same type. In a heterogeneous network, nodes and edges can have types. Formally, a heterogeneous network is a network $G = (V, E)$, associated with two type functions $\phi : V \to A$ and $\psi : E \to R$. These functions associate each node (respectively edge) to its type. More precisely, we define $A = \{a_1, a_2, \ldots, a_\alpha\}$, with α the number of node types, and $R = \{r_1, r_2, \ldots, r_\beta\}$, with β the number of edge types. If $|A| = |R| = 1$, the network is homogeneous.
- **Signed networks.** This is a particular case of a weighted homogeneous network with weights ± 1. Namely, $G = (V, E)$ is a network, and $\tau : E \to \{-1, 1\}$ is a mapping function that associates a sign to each edge.
- **Multilayer networks.** A multilayer network is a type of heterogeneous network where the nodes are grouped into layers, and the edges can connect nodes in the same, or different, layers. Formally, a multilayer network is a triplet $\mathcal{M} = (Y, G, \mathcal{G})$, where Y is the layer index set, $G = \{G_\alpha \mid \alpha \in Y\}$ are (homogeneous) networks $G_\alpha = (V_\alpha, E_\alpha)$, and $\mathcal{G} = \{\mathcal{G}_{\alpha\beta} \mid \alpha, \beta \in Y\}$ are bipartite networks $\mathcal{G}_{\alpha\beta} = (V_\alpha, V_\beta, E_{\alpha\beta})$ encoding the inter-layer connectivity. There is a rich literature on multilayer networks, with different special cases such as multiplex or temporal networks [21–26]. The interested reader can refer to [21] for an extended overview.
- **Temporal networks.** Temporal networks are specific cases of multilayer networks where the layers are ordered by time, that is, they represent the evolution of a graph over time [21, 27, 28].
- **Knowledge networks.** Knowledge graphs are defined as a set of triples $(u, r, v) \in V \times R \times V$, where the nodes u and v belong to the nodes set V, and they are connected by edges of type $r \in R$.

3. Existing taxonomies of network embedding methods

The huge amount and variety of embedding methods [29] make their classification into a unified taxonomy a difficult task. The methods can indeed be sorted according to several criteria. We will briefly present some of the most common taxonomies.

A first way to classify network embedding methods is based on the type of networks used as input. Some authors thereby distinguish the methods designed for homogeneous or heterogeneous networks [29]. The same strategy can be used to classify embedding methods designed for static and temporal networks, or single and multilayer networks. Based on this taxonomy, it is possible to add a layer of complexity by considering the type of component that the methods embed in the vectorial space, i.e. the nodes, the edges, the subgraphs, or the whole network. Other authors use some properties of the network embedding process to classify the different methods [30]. For instance, the network embedding methods may be classified depending on the network properties they intend to preserve, in addition to the part of the network the methods is focused (on the nodes, edges, or the whole network). Focusing on nodes, three different types of property preservations can be defined, at different scales: microscopic, mesoscopic, and macroscopic properties. Methods preserving microscopic properties retain structural equivalences between nodes, such as first-order, second-order, or high-order similarities. They hence seek to preserve the homophily existing in the original network. Methods preserving mesoscopic properties focus on the regular equivalence between nodes, on intra-community similarity, or, more generally, on properties that are in between the close node neighbourhood and the whole network. Finally, methods preserving macroscopic properties tend to preserve whole network properties, like the scale-freeness [31]. Based on the same idea, a different taxonomy has been adopted by Cui *et al* [7]. In this work, the authors discriminate the network embedding based on the information they wish to encode. The first class of methods, called structure and property preserving methods, preserves structural information like the neighbourhood or the community structures. The second class, called information-preserving methods, constructs the embedding using complementary information like node labels and types (for heterogeneous networks) or edge attributes. The third class, called advanced information-reserving method, gathers supervised methods that propose an end-to-end solution (learning process where all parameters are trained jointly) and use various complementary information to learn the embedding space.

In conclusion, multiple taxonomies have been proposed, based on several criteria: the properties preserved by the network embedding methods [7, 30], the type and the properties of input networks [20, 29], or based on mathematical considerations [19, 32–34]. Our approach for the review is based on mathematical considerations and similar to those in Chami *et al* [19]. However, while Chami *et al* extended the encoder–decoder framework of Hamilton *et al* [18] (see section 4) to include deep-learning methods as a special case. Herein, we have defined a more flexible approach to organise these methods that are not constraints by the encoder–decoder framework, while this framework can be a useful tool for understanding the methods, we believe that a more flexible approach will offer easier integration of new methods into this taxonomy. Significantly, our review includes higher-order network embedding methods that were not covered in [19]. Note that our review presents a wide range of methods akin to a diverse set of 'species' coexisting within a 'zoo', hence the chosen title for our review. In this way, the objective of this new taxonomy is to be fine-grained, to offer a consensual view, and to be easily extended to integrate novel methods. In addition, our approach is independent of the scientific domain of development and application of the embedding methods.

4. Taxonomy of network embedding methods

Recently, important efforts have been made to produce general frameworks defining different embedding methods under a common mathematical formulation [18, 19, 35]. Notably, Hamilton *et al* [18] proposed an encoder–decoder framework to define embedding methods, following four components:

(i) A pairwise similarity function: $s_{\mathcal{G}} : V \times V \to \mathbb{R}^+$.
This function defines the similarity measure between the nodes in the original (i.e. direct) network space.
(ii) An encoder function: $\mathrm{Enc} : V \to \mathbb{R}^d$.
This function encodes the nodes into the embedding space. For instance, the node $v_i \in V$ is embedded into the vector $z_i \in \mathbb{R}^d$.
(iii) A decoder function: $\mathrm{Dec} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$.
This function associates a similarity measure in the embedding space to each pair of embedding vectors.
(iv) A loss function: $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

This function measures the quality of the pairwise reconstruction. The objective is to minimise the errors of the reconstruction as follows: $\mathrm{Dec}(\mathrm{Enc}(v_i), \mathrm{Enc}(v_j)) = \mathrm{Dec}(z_i, z_j) \approx s_G(v_i, v_j)$. Most approaches minimise an empirical loss function around a set of training nodes (noted $\mathcal{D}$) rather than the theoretical loss function

$$\mathcal{L} = \sum_{(v_i, v_j) \in \mathcal{D}} l\left(\mathrm{Dec}\left(z_i, z_j\right), s_{\mathcal{G}}\left(v_i, v_j\right)\right). \tag{1}$$

Many embedding techniques align with this taxonomy, but a few of them are inaccurately described. For instance, higher-order network embedding techniques [36–38] do not only use pairwise similarity functions (see section 4.3). While this framework has been extended and enhanced to incorporate deep learning techniques [19], it still does not cover higher-order network embedding methods.

Another general framework has been proposed by Yang *et al* [35] to classify heterogeneous network embedding (HNE) methods. The idea of this framework is to convert the homophily principle (similar nodes in a network will be close in the embedding space) into a generic objective function:

$$\mathcal{J} = \sum_{v_i, v_j \in V} w_{v_i v_j} d\left(z_i, z_j\right) + \mathcal{J}_{\mathrm{R}}. \tag{2}$$

The term $w_{v_i v_j}$ denotes the proximity weight, $d(z_i, z_j)$ is the embedding distance between the embedding vectors associated with the nodes v_i and v_j, and $\mathcal{J}_{\mathrm{R}}$ represents some additional objectives such as regularisers.

The taxonomy proposed in this work is based on a mathematical point of view, illustrated in figure 1, which splits the methods into two main classes depending on their depth: the shallow embedding methods (4.1), and the deep learning methods (4.2). We complement these two classes by including higher-order methods (4.3), which can be classified as either shallow embedding or deep learning methods, enabling us to spotlight these new types of methods. In the next sections, we adopt the notation defined in section 2 for both the network and the associated embedding. In the following, we write $\|.\|_{\mathrm{F}}$ for the Frobenius norm, and $\|.\|_2$ for the Euclidean norm.

4.1. Shallow network embedding methods

In this section we will consider the shallow network embedding methods, which are a set of methods with an encoder function that can be written as follows:

$$\mathrm{Enc}\left(v_i\right) = \mathbf{Z}\mathbf{v_i}, \tag{3}$$

where $\mathbf{Z}$ corresponds to the matrix with the embedding vectors of all nodes, and $\mathbf{v_i}$ corresponds to the indicator vector associated with each node v_i (vector of zeros except in position i, where the element is equal to 1). In this case, the objective of the embedding process is to optimise the embedding matrix $\mathbf{Z}$ in order to have the best mapping between the nodes and the embedding vectors (figure 2). We define three major classes of shallow embedding methods based on different mathematical processes: the matrix factorisation methods, the random walk methods, and the optimisation methods.

4.1.1. Matrix factorisation methods

Matrix factorisation is based on the fact that a matrix, such as the adjacency matrix or the Laplacian matrix, can fully represent a network. This fact implies that existing methods from matrix algebra, such as matrix factorisation, can be used for network embedding. Network embedding methods based on matrix factorisation are directly inspired by linear dimensionality reduction methods, such as principle component analysis (PCA) [39], linear discriminant analysis (LDA) [17], or multidimensional scaling (MDS) [20]. Other methods are inspired by non-linear dimensionality reduction methods such as Isomap [40], which is an extension of MDS [20], locally linear embedding (LLE) [41], t-distributed stochastic neighbour embedding (t-SNE) [42], or more recently uniform manifold approximation and projection (UMAP) [43]. The factorisation process depends on the properties of the matrices. For positive semi-definite matrices, like graph Laplacians, the embedding can be obtained by eigenvalue decomposition. However, for unstructured matrices, like covariance matrices, gradient descent or singular value decomposition (SVD) should be used to obtain the network embedding. Thereafter, we will describe the most common network embedding methods based on matrix factorisation.

- **Laplacian eigenmaps (LEs)** [44] aims to embed the network in such a way that two nodes close in the original network are also close in the low-dimensional embedding space, by preserving a similarity measure defined by the weight between nodes. In that way, we define the weight matrix denoted by W, where W_{ij}

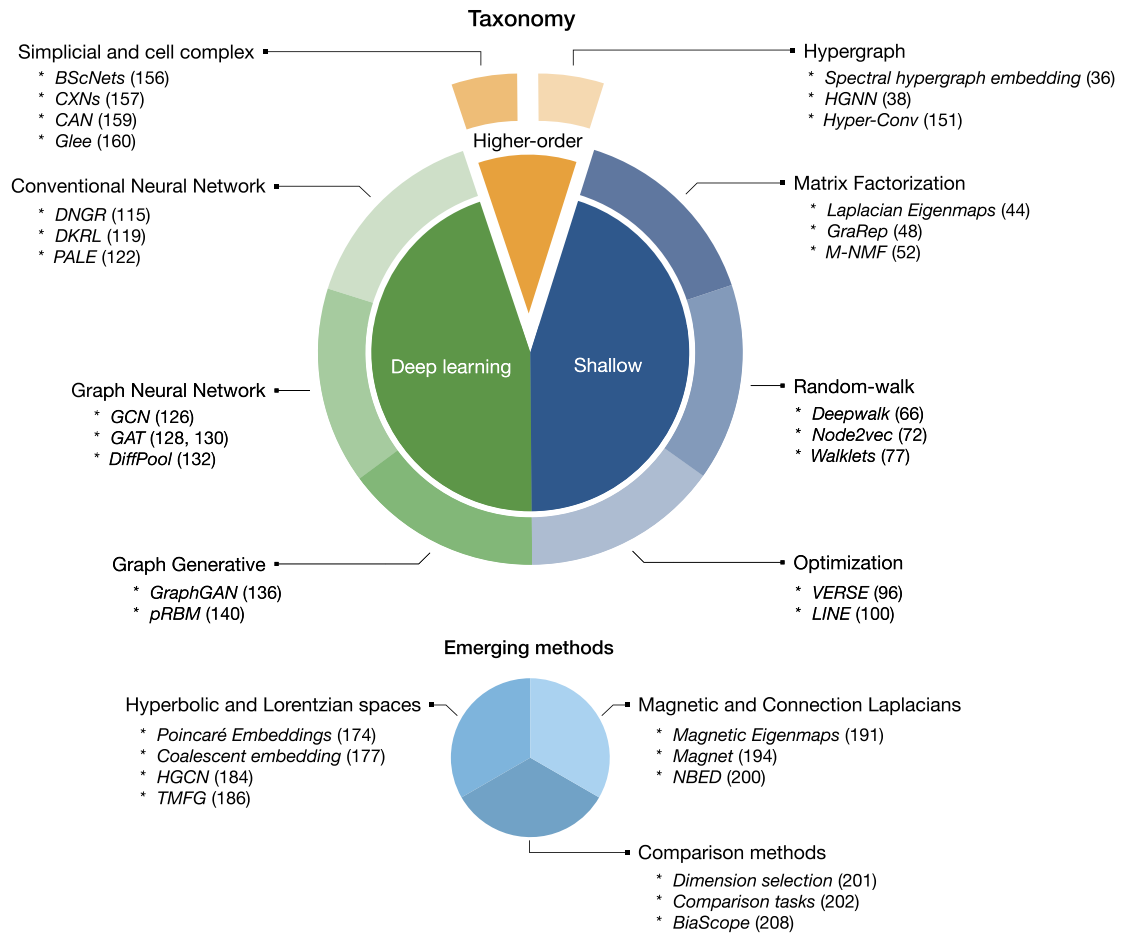

Figure 1. Pie charts describing the new taxonomy defined in this manuscript. In the top pie chart, the methods are divided into two main categories: shallow embedding methods and the deep learning methods, complemented by higher-order methods that can be either a shallow embedding or a deep learning methods. The bottom pie chart highlights the three major emerging groups of methods. Notably, these emerging groups of methods can be classified into our defined taxonomy due to its flexibility.

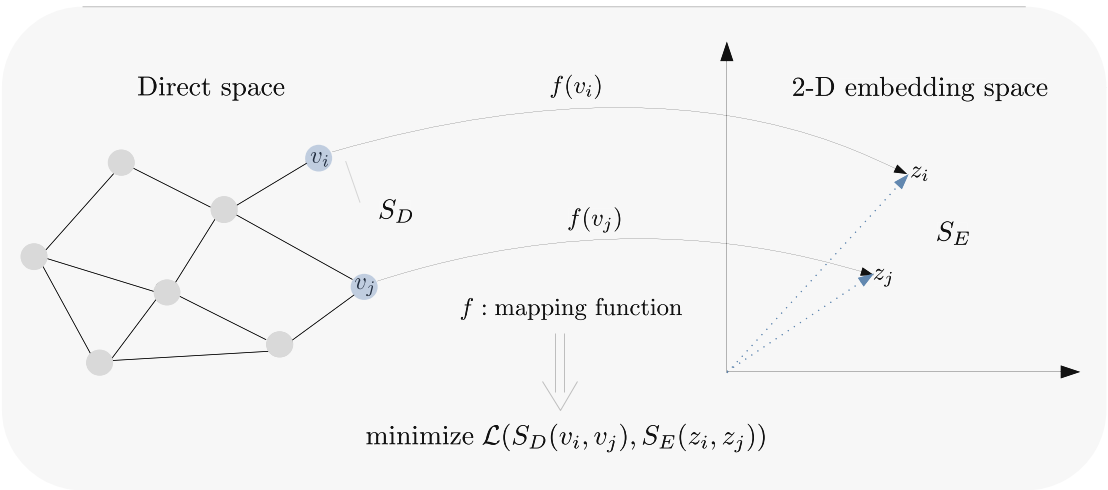

Figure 2. Shallow network embedding: to perform shallow network embedding, a network is projected into a low-dimensional vector space, such as a two-dimensional embedding space. This projection is achieved using a mapping function f that enables the mapping from the direct space to the embedding space. The mapping function f is derived by optimising a loss function $\mathcal{L}$, which aims to minimise the difference between the similarity measures of nodes in the direct space (S_D) and their equivalents in the embedded space (S_E) obtained through the decoder function.

encodes the weight between the nodes i and j. The learning process is done by optimising the following objective function:

$$\mathcal{L} = \sum_{v_i, v_j \in V} \mathrm{Dec}\left(z_i, z_j\right) \cdot s_{\mathcal{G}}\left(v_i, v_j\right), \tag{4}$$

with $\mathrm{Dec}(z_i, z_j) = \|z_i - z_j\|_2^2$, and $s_{\mathcal{G}}(v_i, v_j) = W_{ij}$.

We can introduce the Laplacian matrix L, defined as $L = D - W$, with $D_{ii} = \sum_j W_{ji}$. The equation (4) can be written as:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2} \sum_{i,j} \|z_i - z_j\|_2^2 W_{ij} \\
\mathcal{L} &= \sum_{i,i} \|z_i\|_2^2 D_{ii} + \sum_{i,j} \|z_i z_j\|_2 W_{ij} \\
\mathcal{L} &= \sum_{i,i} \|z_i z_j\|_2 L_{ij} \\
\mathcal{L} &= \mathrm{Tr}\left(Z Z^{\mathrm{T}} L\right) = \mathrm{Tr}\left(Z^{\mathrm{T}} L Z\right),
\end{aligned} \tag{5}$$

with $Z = (z_1, z_2, \ldots, z_n) \in \mathbb{R}^{d \times n}$. The loss function needs to respect the constraint $Z^{\mathrm{T}} D Z = I$ to avoid trivial solutions. The solution can be obtained by finding the matrix composed of the eigenvectors associated with the d smallest eigenvalues of the generalised eigenvalue problem $LZ = \Lambda D Z$, with $\Lambda = \mathrm{diag}([\lambda_1, \lambda_2, \ldots, \lambda_n])$ [45].

It is important to note that LEs use a quadratic decoder function. This function does not preserve the local topology because the quadratic penalty penalises the small distance between embedded nodes.

- **Cauchy graph embedding** [46] aims to improve the previous method (LEs), which does not preserve the local topology. Cauchy graph embedding use a different decoder function $\mathrm{Dec}(z_i, z_j) = \frac{\|z_i - z_j\|_2^2}{\|z_i - z_j\|_2^2 + \sigma^2} = 1 - \frac{\sigma^2}{\|z_i - z_j\|_2^2 + \sigma^2}$, with σ^2 representing the variance. Consequently, the loss function can be written as follows:

$$\mathcal{L} = \sum_{i,j} \frac{1}{\|z_i - z_j\|_2^2 + \sigma^2} W_{ij}, \tag{6}$$

with the following constraints: $\sum_i z_i = 0$, and $Z^{\mathrm{T}} Z = I$, where $Z = (z_1, z_2, \ldots, z_n) \in \mathbb{R}^{d \times n}$. The solution is obtained by an algorithm that mixes gradient descent and SVD.

- **Graph factorisation** [47] proposes a factorisation method that is designed for network partitioning. It learns an embedding representation that minimises the number of neighbouring vertices across the partition. The loss function can be written as follows:

$$\mathcal{L} = \sum_{v_i, v_j \in V} \|\mathrm{Dec}\left(z_i, z_j\right) - s_{\mathcal{G}}\left(v_i, v_j\right)\|_2^2 + \frac{\lambda}{2} \sum_{v_i \in V} \|z_i\|_2^2, \tag{7}$$

with $\mathrm{Dec}(z_i, z_j) = z_i^{\mathrm{T}} z_j$, $s_{\mathcal{G}}(v_i, v_j) = W_{ij}$, and λ a regularisation parameter. Notably, this method is scalable and can deal with networks with millions of vertices and billions of edges.

- **GraRep** [48] extends the skip-gram model [49] to capture higher-order similarity, i.e. k-step neighbours of nodes. The value of k is chosen such as $1 \leqslant k \leqslant K$, with K the highest order. GraRep is also motivated by the noise-contrastive estimation (NCE) approximation [50] which consists in learning a model that converges to the objective function. NCE trains a binary classifier to distinguish between node samples coming from the similarity distribution $s_{\mathcal{G}}$ and node samples generated by a noise distribution over the nodes. GraRep defines its k-step loss function as follows:

$$\mathcal{L}_k = \sum_{v_i \in V} \left(\sum_{v_j \in V} T_{i,j}^k \log\left(\sigma\left(x_i^{\mathrm{T}} x_j\right)\right) + \lambda \mathbb{E}_{v_j \sim p_k(V)} \left[\log\left(\sigma\left(-x_i^{\mathrm{T}} x_j\right)\right)\right] \right), \tag{8}$$

where the matrix T represents the transition matrix, defined as $T = D^{-1} A$, with A the adjacency matrix, and D the degree matrix. The vectors x_i and x_j are the vector representations of the nodes v_i and v_j in the direct space. The term $\mathbb{E}_{v_j \sim p_k(V)}$ is the expectation of the node v_j, obtained by negative sampling. The expectation follows the noise distribution over the nodes in the network, denoted by $p_k(V)$. The parameter λ indicates the number of negative samples, and $\sigma(.)$ is the sigmoid function defined as $\sigma(x) = (1 + \mathrm{e}^{-x})^{-1}$. GraRep reformulates its loss function minimisation into a matrix factorisation problem. Each k-step term is

computed from the matrix X^k defined as $X^k_{ij} = \max([\log\left(\frac{T^k_{ij}}{\sum_m T^k_{mj}}\right) - \log(\beta)], 0)$. Then, the low-dimensional representation of the matrix C^k is constructed from the SVD: SVD(X^k). Finally, the final representation is obtained by concatenating all order-term matrices, $C = [C^1, C^2, \ldots, C^K]$.

- **High-order proximity preserved embedding (HOPE)** [51] has been developed to encode higher-order similarity of large-scale networks while also capturing the asymmetric transitivity, i.e. going from node v_i to node v_j can be different from going from node v_j to node v_i. HOPE can hence deal with directed networks. The loss function is equal to:

$$\mathcal{L} = \sum_{v_i, v_j \in V} \|\mathrm{Dec}\left(z_i, z_j\right) - s_{\mathcal{G}}\left(v_i, v_j\right)\|_2^2, \tag{9}$$

with $\mathrm{Dec}(z_i, z_j) = z_i^{\mathrm{T}} z_j$ and $s_{\mathcal{G}}(v_i, v_j)$ denoting any similarity measure between v_i and v_j. The authors of HOPE introduce a general factorisation in which the similarity measure can be factorised in one matrix associated with the global similarity M_{g} and another matrix associated with the local similarity M_{l}. So, the similarity matrix can be expressed as $S = M_{\mathrm{g}}^{-1} M_{\mathrm{l}}$, where both local and global similarities are polynomial sparse matrices. This also enables HOPE to use efficient SVD decomposition for embedding large-scale networks. The authors considered different similarity measures such as Katz index ($S^{\mathrm{katz}} = (I - \beta A)^{-1}(\beta A)$), rooted PageRank ($S^{\mathrm{RPR}} = (I - \alpha T)^{-1}((1-\alpha)I)$), common neighbours ($S^{\mathrm{CN}} = I(A^2)$), or Adamic–Adar ($S^{\mathrm{AA}} = I(ADA)$), where A indicates the adjacency matrix, T represents the transition matrix, α a value $\in [0, 1)$, and β a value less than the spectral radius of the adjacency matrix.
- **Modularised nonnegative matrix factorisation (NMF)** [52] aims to obtain an embedding representation aware of the community structure of the original network while maintaining the microscopic information from the first-order and second-order similarities. Let us define the similarity measure $S = S^{(1)} + \eta S^{(2)} \in \mathbb{R}^{n \times n}$, where $S^{(1)}$ is the first-order similarity matrix, for instance, $S^{(1)}_{ij} = A_{ij}$ with A the adjacency matrix, and $S^{(2)}$ is the second-order similarity matrix, defined as $S^{(2)}_{ij} = \frac{\mathcal{N}_i \mathcal{N}_j}{\|\mathcal{N}_i\|_2 \|\mathcal{N}_j\|_2}$, with $\mathcal{N}_i = (S^{(1)}_{i1}, S^{(1)}_{i2}, \ldots, S^{(1)}_{in})$ the first-order similarity vector of the node i. The parameter η is the weight of the second-order term (often chosen equal to 5 [52]). The embedding of the microscopic structure can be expressed in the NMF framework as the following optimisation problem:

$$\min_{M,U} \ \|S - MU^{\mathrm{T}}\|_F^2; \quad M > 0\,,\ U > 0, \tag{10}$$

with $M \in \mathbb{R}^{n \times d}$ and $U \in \mathbb{R}^{n \times d}$ two non-negative matrices; U_i is the embedding of the node i.

The community structure is obtained with modularity maximisation, which is expressed for two communities as $Q = \frac{1}{4m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) h_i h_j$, with k_i the degree of the node i, h_i is equal to 1 if the node i belongs to the first community, otherwise is equal to -1, and m is the total number of edges. Let us define B such as $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$, so the modularity becomes $Q = \frac{1}{4m} h^{\mathrm{T}} B h$, where $h \in \mathbb{R}^n$. The generalisation of the modularity optimisation problem for k communities is defined as:

$$\min_{H} \ -\beta\,\mathrm{Tr}\left(H^{\mathrm{T}} B H\right)\,; \quad \mathrm{Tr}\left(H^{\mathrm{T}} H\right) = n, \tag{11}$$

with $H \in \mathbb{R}^{n \times k}$, β is a positive parameter. The second equation imposes the association of each node to one community. The two models are combined using a term that uses the community structure to guide the node representation learning process. Formally, we define $C \in \mathbb{R}^{k \times d}$ as the community representation matrix; C_r as the representation of the community r, and $U_i C_r$ represents the propensity of the node i to belong to the community r. So the last term to optimise is equal to $\alpha \|H - UC^{\mathrm{T}}\|_{\mathrm{F}}^2$, with the constraint that $C > 0$, and α a positive parameter. Finally, the equation to be optimised is the following one:

$$\begin{aligned} \min_{M,U,H,C} \ & \|S - MU^{\mathrm{T}}\|_{\mathrm{F}}^2 - \beta\,\mathrm{Tr}\left(H^{\mathrm{T}} B H\right) + \alpha \|H - UC^{\mathrm{T}}\|_{\mathrm{F}}^2 \\ & M > 0\,,\ U > 0\,,\ C > 0\,,\ \mathrm{Tr}\left(H^{\mathrm{T}} H\right) = n\,. \end{aligned} \tag{12}$$

Due to the non-convex behaviour of the previous function, a non-trivial optimisation process has been developed [52].
- **Text-associated deepwalk (TADW)** [53] aims to integrate text data information into the network embedding process. The authors first prove that the learning process used in the deepwalk embedding method (see the section about random walk network embedding methods) is equivalent to the optimisation of a matrix

factorisation problem, $M = W^{\mathrm{T}}H$, with $M \in \mathbb{R}^{n\times n}$ the matrix of the original network, $W \in \mathbb{R}^{d\times n}$ the weight matrix, and $H \in \mathbb{R}^{d\times n}$ the factor matrix. The factorisation matrix problem is the following:

$$\min_{W,H} \|M - W^{\mathrm{T}}H\|_{\mathrm{F}}^2 + \frac{\lambda}{2}\left(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2\right). \tag{13}$$

The idea of TADW is to take into account a text factor matrix T into the decomposition, such that $M = W^{\mathrm{T}}HT$, with $M \in \mathbb{R}^{n\times n}$, $W \in \mathbb{R}^{d\times n}$, $H \in \mathbb{R}^{d\times k}$, and $T \in \mathbb{R}^{k\times n}$. The new factorisation matrix problem is:

$$\min_{W,H} \|M - W^{\mathrm{T}}HT\|_{\mathrm{F}}^2 + \frac{\lambda}{2}\left(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2\right). \tag{14}$$

The optimisation process is obtained with the gradient descent algorithm introduced by Yu *et al* [54].

- **Other matrix factorisation methods**: the methods detailed above are some of the most common ones, and there are used as basis for alternative or extended methods. Notably, several strategies propose variations of the LEs. For instance, the locality preserving properties method [55] uses a linear approximation of LE. The method structure-preserving embedding [56] extends LE by including connectivity structure similarity as a constraint during the learning process. Similarly, augmented relational embedding [57] modifies the Laplacian matrix to integrate feature information.

  Spectral techniques such as label informed attributed network embedding [58] are also promising for preserving node structure similarities and the correlations between their labels.

  Some methods dedicated to multi-class node classification have also been developed. These methods can be seen as variations of the TADW method. For instance, the method homophily, structure, and content augmented [59] adds a regularisation term to the objective function of TADW to enforce the structure homophily existing between nodes in the network. Max-margin deepwalk [60] adds a multi-class support vector machine (SVM) to integrate labelling information of the nodes. Discriminative matrix factorisation [61] uses a linear classifier trained on labelled nodes to complement the TADW objective function.

  A large number of other embedding methods based on matrix factorisation have been developed. For instance, several embedding methods applied to knowledge graphs, use matrix factorisation (or tensor factorisation). These methods can be defined as relation learning methods. We can mention some of the most common ones, such as RESCAL [62], DistMult [63] and ComplEx [64]. DistMult is a special case of RESCAL developed to reduce overfitting and ComplEx extents DistMult to complex matrices.

  Finally, matrix factorisation methods can extract network embeddings from a time-dependent node similarity measure inspired by dynamical systems and control theory notions [65].

4.1.2. Random walk methods

The idea behind random walk embedding is to encode the scores of the random walk into an embedding space. Most methods use ideas initially developed in the DeepWalk paper [66]. In this section, we describe the most common methods and some of their extensions.

- **DeepWalk** [66] is a scalable network embedding method that uses local information obtained from truncated random walks to learn latent representations. DeepWalk treats the walks as the equivalent of sentences. The process is inspired by the famous word2vec method [49, 67], in which short sequences of words from a text corpus are embedded into a vectorial space. The first step of DeepWalk consists in generating sequences of nodes from truncated random walks on the network. Then, the update procedure consists in applying the skip-gram model [49] on the sequences of nodes, in order to maximise the probability of observing a node neighbour conditioned on the node embedding. The loss function is defined as follows:

$$\min_{\phi} -\log\left(\mathbb{P}\left(\{v_{i-w},\ldots,v_{i+w}\} \setminus v_i \mid \phi(v_i)\right)\right), \tag{15}$$

with w indicating the window size (in terms of node sequence), and $\phi : V \to \mathbb{R}^d$ indicating the mapping function. We can also see $\phi \in \mathbb{R}^{n\times d}$ as the matrix of the embedding representation of the nodes. The skip-gram model transform the equation (15) as follows:

$$\min_{\phi} -\log\left(\prod_{j=i-w}^{i+w} \mathbb{P}(v_j|\ \phi(v_i)\right). \tag{16}$$

Then, the hierarchical softmax function [68] is applied to approximate the joint probability distribution as:

$$\mathbb{P}\left(v_j \mid \phi\left(v_i\right)\right) = \prod_{l=1}^{\log(n)} \mathbb{P}\left(b_l \mid \phi\left(v_i\right)\right) = \prod_{l=1}^{\log(n)} \frac{1}{1+\exp\left(-\phi\left(v_i\right)\psi\left(b_l\right)\right)}, \tag{17}$$

where v_j is defined by a sequence of tree nodes $(b_0, b_1, \ldots, b_{\log(n)})$, with b_0 the root of the tree, and $b_{\log(n)}$ the node v_j. Notably, similarly to the TADW method, Deepwalk is equivalent to the following matrix factorisation problem: $M = W^{\mathrm{T}}H$ [53, 69], with $W \in \mathbb{R}^{d\times n}$ the weight matrix, and $H \in \mathbb{R}^{d\times n}$ the factor matrix. Several extensions of the deepwalk have been adapted for multilayer networks [70, 71].

- **node2vec** [72] is a modified version of DeepWalk, with two main changes. First, node2vec uses a negative sampling instead of a hierarchical softmax for normalisation. This choice improves the running time. Second, node2vec uses a biased random walk that offers more flexible learning with control parameters. The biased random walk can be described as:

$$\mathbb{P}\left(c_i = x \mid c_{i-1} = y\right) = \begin{cases} \frac{\pi_{yx}}{Z} & \text{if } (y,x) \in E \\ 0 & \text{otherwise} \end{cases}, \tag{18}$$

where π_{yx} is the unnormalised transition probability between node y and node x, and Z is the normalising constant. The variable π is defined as follows:

$$\pi_{yx} = \begin{cases} \frac{1}{p}\omega_{yx} & \text{if } d_{tx} = 0 \\ \omega_{yx} & \text{if } d_{tx} = 1 \\ \frac{1}{q}\omega_{yx} & \text{if } d_{tx} = 2 \end{cases}, \tag{19}$$

where ω_{yx} is the weight of the edge between the node y and the node x, and d_{tx} is the shortest path between the node x and the node t, which is the node reached before the node y. The parameters p and q are two control parameters of the random walk. The return parameter p controls the likelihood of immediately revisiting a node in the walk, while the in-out parameter q controls the likelihood of visiting a node in the neighbourhood of the node that was just visited. Both parameters control if the random walk follows a breadth-first sampling strategy or a depth-first sampling strategy. The first strategy preserves the structural equivalence of the nodes, the second one preserves their homophily. Recently, multinode2vec [73] and PMNE [74], two extensions of node2vec, adapted the random walk process to multilayer networks.
- **HARP** [75] is an algorithm that was developed to improve the DeepWalk and node2vec embedding methods. The idea is to capture the global structure of an input network by recursively coalescing edges and nodes of the network into smaller networks with similar structures (see also section 5.2 on network compression). The hierarchy of these small networks is an appropriate initialisation for the network embedding process, because it directly express a reduced-dimension version of the input network while preserving its global structure. The final embedding is obtained by propagating the embedding of the smallest network through the hierarchy.
- **Discriminative deep random walk (DDRW)** [76] is particularly suitable for the network classification task. It can be seen as a DeepWalk extension that considers the label information of nodes. To do so, DDRW jointly optimises the DeepWalk embedding loss function and a classification loss function. The final loss function to optimise is defined as:

$$\mathcal{L} = \eta\mathcal{L}_{\mathcal{DW}} + \mathcal{L}_{\mathcal{C}}, \tag{20}$$

$$\mathcal{L}_{\mathcal{C}} = C\sum_{i=1}^{n}\left(\sigma\left(1 - y_i\beta^{\mathrm{T}}z_i\right)\right)^2 + \frac{1}{2}\beta^{\mathrm{T}}\beta, \tag{21}$$

where η is a weight parameter, and σ the Heaviside function, i.e. $\sigma(x) = x$ for $x > 0$ and $\sigma(x) = 0$ otherwise. Moreover, z_i is the embedding vector of the node v_i, y_i is the label of the node v_i, C is the regulariser parameter, and β the subsequent classifier.
- **Walklets** [77]. Given the observation that DeepWalk can be derived from a matrix factorisation containing the powers of the adjacency matrix [78], it appears that DeepWalk is biased towards lower powers of the adjacency matrix, which correspond to short walks. This can become a limitation when higher-order powers are the most appropriate representations, for instance to embed the regular equivalence between nodes. To bypass this issue, Walklets propose to learn the embedding from a multi-scale representation.

This multi-scale representation is sampled from successive higher powers of the adjacency matrix obtained from random walks. Then, after partitioning the representation by scale, Walklets learns the representation of each node generated for each scale.

- **Struct2vec** [79] aims to capture the regular equivalence between nodes in a network. In other words, two nodes that have identical local network structures should have the same embedding representation. The construction of the embedding representation is based on different steps. The first step is to determine the structural similarity between each pair of nodes for different neighbourhood sizes. The structural similarity between the nodes v_i and v_j, when considering their k-hop neighbourhoods (all nodes at a distance less or equal to k and all edges among them), is defined as follows:

$$d_k\left(v_i, v_j\right) = d_{k-1}\left(v_i, v_j\right) + g\left(s\left(R_k\left(v_i\right)\right), s\left(R_k\left(v_j\right)\right)\right) \; ; \\ k \geqslant 0 \;\text{ and }\; \left|R_k\left(v_i\right)\right|, \left|R_k\left(v_j\right)\right| > 0 \,, \tag{22}$$

where $R_k(v_i)$ is the set of nodes at a distance less or equal to k from the node v_i, $s(S)$ is the ordered degree sequence of a set of nodes S, and $g(S_1, S_2)$ is a distance measure between the two ordered degree sequences S_1 and S_2. The distance used is the dynamic time warping [80] and by convention $d_{-1} = 0$.

This procedure produces a hierarchy of structural similarities between nodes of the network. The hierarchy is used to create a weighted multi-layer network, in which layers represent node similarities for different levels of the hierarchy. The edge weights between node pairs are inversely proportional to their structural similarity. After that, a biased random walk process is applied to the multilayer network to generate sequences of nodes. The sequence of nodes are used to learn a latent representation with the skip-gram process.

- **Other random walk methods**: some methods are designed to integrate additional information in the learning process. For instance, SemiNE [81] is a semi-supervised extension of DeepWalk that takes into account node labels. GENE [82] also integrates node labels. Node labels, as well as additional node contents, are also integrated into TriDNR [83].

  SNS [84] is another method that aims to preserve structural similarity in the embedding representation. SNS measures the regular equivalence between nodes by representing them as a graphlet degree vectors: each element of the graphlet degree vector represents the number of times a given node is touched by the corresponding orbit of graphlets.

  Random walk approaches are widely used for HNE. Examples of HNE method include MRWNN [85], SHNE [86], HHNE [87], GHE [88], JUST [89], HeteSpaceyWalk [90], and TapEm [91]. The interested reader can refer to Yang *et al* [35] for a detailed review of HNE. Finally, random walks are also often used for metapath-based methods, another set of methods relevant for network embedding. This set of methods includes Metapath2vec [92], HIN2vec [93], HINE [94], or, more recently, HERec [95].

4.1.3. Optimisation methods

The two previous groups of methods involve two distinct mathematical processes: matrix factorisations, and random walks. (Matrix factorisation is a common mathematical operation, while random walk encompasses several methods that share a common principle.) However, there are additional embedding techniques that do not fit into either category, but they do share a common objective of optimising a loss function. In essence, these methods use a broad range of mathematical processes but ultimately involve an optimisation step, which is usually achieved through gradient descent. As a result, these approaches can be viewed as hybrid methods that all utilise a shared optimisation step.

The most important step in optimisation methods is to define a loss function that encodes all the properties that should be preserved through the embedding. This loss function often gathers similarities between nodes in the direct space, together with some regulariser terms that depend on network features that we want to preserve. The embedding representation is obtained based on the optimisation of this loss function. We will present the most common optimisation-based network embedding methods and some of their extensions.

- **VERtex similarity embeddings (VERSE)** [96] is a versatile network embedding method that accepts any network similarity measure. Let G be a network with an associated similarity measure $s_G : V \times V \to \mathbb{R}^+$. The VERSE method constructs the embedding representation of the network G, noted $Z \in \mathbb{R}^{d \times n}$, associated with a similarity measure in the embedding space $\mathrm{Dec} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$. Each column of the matrix Z is the embedding vector z_i of the node v_i. The embedding representation is based on the optimisation of a loss function, noted $\mathcal{L}$, corresponding to the Kullback–Leibler divergence between the similarity matrix in the direct space (i.e. original network) and the similarity matrix in the embedding space:

$$\mathcal{L} = \sum_{i=1}^{n} s_{\mathcal{G}}\left(v_i, .\right) \cdot \ln\left(\frac{s_{\mathcal{G}}\left(v_i, .\right)}{\mathrm{Dec}\left(v_i, .\right)}\right)$$
$$= -\sum_{i=1}^{n} s_{\mathcal{G}}\left(v_i, .\right) \cdot \ln\left(\mathrm{Dec}\left(v_i, .\right)\right) + C, \tag{23}$$

with $s_{\mathcal{G}}(v_i, .)$ (resp. $\mathrm{Dec}(v_i, .)$) the similarity vector between the node v_i and all the other nodes of the network in the direct space (resp. embedding space). Notably, $\sum_{j=1}^{n} s_{\mathcal{G}}(v_i, v_j) = \sum_{j=1}^{n} \mathrm{Dec}(v_i, v_j) = 1$. Moreover, $C = \sum_{i=1}^{n} s_{\mathcal{G}}(v_i, .) \cdot \ln(s_{\mathcal{G}}(v_i, .))$ defines a constant that does not affect the optimisation algorithm, and can therefore be neglected. The vector $s_{\mathcal{G}}(v_i, .)$ corresponds to the vector associated with the node v_i in the similarity matrix defined in the direct space. As stated above, the similarity matrix in the direct space (network) can be defined by several measures. The authors proposed three different similarity matrices: the adjacency matrix, the SimRank similarity matrix [97], and a similarity matrix based on random walk with restart. The vector $\mathrm{Dec}(v_i, .)$ corresponds to the vector associated with the node v_i in the similarity matrix defined in the embedding space. The vector $\mathrm{Dec}(v_i, .)$ can also be seen as the similarity vector between the vectors z_i and z_j with $j \neq i, j \in [1, n]$, where n is the number of nodes in the network. The vectors gathered in the similarity matrix in the embedding space are defined by the following equation:

$$\mathrm{Dec}\left(v_i, .\right) = \frac{\exp\left(z_i^{\mathrm{T}} Z\right)}{\sum_{j=1}^{n} \exp\left(z_i^{\mathrm{T}} z_j\right)}. \tag{24}$$

The node embedding is obtained by optimising the loss function with a gradient descent algorithm. Usually, the embedding vectors are initialised with a normal distribution with a mean equal to zero. Because the Kullback–Leibler optimisation is a time-consuming process, a negative sampling procedure, such as NCE [50, 98] is often used. Recently, an extension of VERSE to heterogeneous multiplex networks, named MultiVERSE, has been developed [99].

- **Large scale information network embedding (LINE)** [100] aims to embed both first-order and second-order similarities.
  - The embedding space of the first-order similarity can be obtained by the following optimisation algorithm.
    Let us define the theoretical expected probability as:

$$p_1\left(v_i, v_j\right) = \frac{1}{1 + \exp\left(-z_i^T z_j\right)} \; ; \; z_i \in \mathbb{R}^d, \tag{25}$$

    and the empirical probability as:

$$p_1^*\left(v_i, v_j\right) = \frac{w_{ij}}{W} \; ; \; W = \sum_{(i,j) \in E} w_{ij}. \tag{26}$$

    The main goal of LINE is to minimise the error between the theoretical expected probability p_1 and the empirical probability p_1^*. To do so, the loss function O_1 minimises the distance $d(p_1^*(.,.), p_1(.,.))$ by using the Kullback–Leibler divergence. Hence, O_1 can be written as follows:

$$O_1 = -\sum_{(i,j) \in E} w_{ij} \log\left(p_1\left(v_i, v_j\right)\right). \tag{27}$$

  - The embedding space of the second-order similarity is obtained with the optimisation process described as follows. Let us define the theoretical expected probability as:

$$p_2\left(v_j \mid v_i\right) = \frac{\exp\left(z_j^{\mathrm{T}} z_i\right)}{\sum_{k=1}^{n} \exp\left(z_k^{\mathrm{T}} z_i\right)} \; ; \; z_i \in \mathbb{R}^d. \tag{28}$$

    The empirical probability is defined as:

$$p_2^*\left(v_j \mid v_i\right) = \frac{w_{ij}}{d_i} \; ; \; d_i = \sum_{i \in N(i)} w_{ik}, \tag{29}$$

    where $N(i)$ is the neighbourhood of the node i, and d_i defines the out-degree of the node i. The idea is again to minimise the error between the theoretical expected probability and the empirical probability.

To do so, the loss function O_2 minimises the distance $d(p_2^*(.\mid v_i), p_2(.\mid v_i))$ by using the Kullback–Leibler divergence. Hence, O_2 can be written as follows:

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log\left(p_2\left(v_j \mid v_i\right)\right). \tag{30}$$

The first and second-order node representations are computed separately, and both first and second-order embedding representations are concatenated for each node.

- **Transductive LINE (TLINE)** [101] is a transductive version of LINE that also uses node labels to train an SVM classifier for node classifications. Both node embedding and the SVM classifier are optimised simultaneously, in order to make full use of the label information existing in the network. Notably, TLINE as LINE permits fast embedding of large-scale networks by using edge sampling and negative sampling in the stochastic gradient descent process. The method optimises a loss function O_{T} composed of the same loss functions as LINE, O_1 and O_2 to embed both first and second-order similarity, and the SVM loss function O_{SVM}, that is,

$$O_{\mathrm{T}} = O + \beta O_{\mathrm{SVM}}, \tag{31}$$

$$O_{\mathrm{SVM}} = \sum_{i=1}^{n}\sum_{k=1}^{K} \max\left(0, 1 - y_i^k w_k^{\mathrm{T}} z_i\right) + \lambda \|w_k\|^2, \tag{32}$$

where β is a trade-off parameter between LINE and SVM, n the number of nodes, K the number of label types in the network, z_i the embedding vector representation of the node v_i, w_k the parameter vector of the label class k, and y_i^k is equal to 1 if the node v_i is in the class k.

- **Other optimisation methods**: a wide range of optimisation methods are applied to heterogeneous networks. Many of them are similar to LINE and optimise first and second-order similarities. In predictive text embedding [102], the loss function is divided into several loss functions, each associated with one network of the heterogeneous network. The asymmetric proximity preserving [103] network embedding method is similar to VERSE, and captures both asymmetric and high-order similarities between node pairs thanks to an optimisation process over random walk with restart results.

  Recently, many network embedding methods have been adapted or designed for multilayer networks, and a significant portion of them are based on an optimisation process [104–108]. In addition, several embedding methods applied to knowledge graphs are optimisation methods. These methods are often called relation learning methods. We can mention some of the most common ones, like the translation-based methods first defined by Bordes *et al* [109]. This method, named TransE, embeds multi-relational data that uses directed graphs. Edges can be defined by three elements: the head node (h), the tail node (t), and the edge label (l). The embedding vector of the tail node t should be close to the embedding vector of the head node h, plus some vector that depends on the relationship l. This approach constructs the embedding representation by optimising a loss function that integrates these three elements. This method has given rise to several alternative methods: TransH [110] that improves TranSE for reflexive/one-to-many/many-to-one/many-to-many relationships, TransR [111] that builds entity and relation embeddings in separate entity space and relation space (in contrast to the two previous methods), and TransD [112], which is an improvement of TransR for large-scale networks. Recently, the RotatE method has been developed [113]. RotatE is a knowledge graph embedding method that can model and infer various relation patterns such as symmetry, inversion, and composition.

4.2. Deep learning methods

In recent years, the use of deep learning for data analysis has increased steadily, and network analysis, including network embedding [114], is no exception. The success of deep learning methods can be explained by their ability to capture complex features and non-linearity among input variables. We define three major classes of deep learning embedding methods: the conventional neural networks based methods, the graph neural networks (GNNs) based methods, and the graph generative methods. All three of these classes of methods are based on different philosophies and mathematical formulations of deep learning.

4.2.1. Conventional neural networks

The first network embedding methods based on deep learning methods use conventional deep learning techniques. We can cite the following class of deep learning architectures:

- **Autoencoder:** autoencoders and their various variants been widely used for feature extraction. This capability has been utilised by several high-performing network embedding methods such as: deep neural networks

for learning graph representations (DNGRs) [115], structural deep network embedding (SDNE) [116], variational graph auto-encoders [117]. In the case of SDNE, nodes are represented by their high-dimensional neighbourhood vectors, which are then fed into the autoencoder. This approach allows SDNE to preserve high-order proximity while also incorporating the LEs proximity measure in order to maintain 1st-order proximity. On the other hand, DNGR achieves node embeddings by applying an autoencoder to the positive pointwise mutual information matrix, which is a matrix derived using random surfing.

- **Convolutional neural networks (CNNs):** CNN methods have shown high performance [118] for detecting the significant features, this ability has been used by various network embedding methods. Description-embodied knowledge representation learning [119], PATCHY-SAN [120]. As mentioned in the case of autoencoders, the type of input of the deep learning methods can be very diverse. In the case of a neural network, not only the weights of the neural layers [121] determine the node coordinates of the embedded network, but the chosen type of input network representation may also affect the embedding space.
- **Other neural networks:** for instance, a multi-layer perceptron is used by the method predicting anchor links via embedding [122], and a recurrent neural network is used by the method DeepCas [123].

4.2.2. GNNs

Recently, an important class of deep learning methods for network embedding has been developed: GNNs [124]. GNNs generalise the notion of CNNs (typically applied to image datasets, with an image seen as a lattice network of pixels) to arbitrary networks. GNNs encode high-dimensional information about each node neighbourhood into a dense vector embedding. GNNs algorithms can be divided into two main components. The encoder, which maps a node v_i into a low-dimensional embedding vector z_i, based on the local neighbourhood and the attributes of the node, and a decoder, which extracts user-specified predictions from the embedding vector. This kind of method is suitable for end-to-end learning and offers state-of-the-art performance [124, 125]. GNNs and their application to network embedding can be divided into different classes of methods.

- **Graph convolutional networks (GCNs):** GCNs is the generalisation of CNNs to graphs [126]. The basic idea behind CNN is to apply convolutional operations during the learning process to capture local properties of the input data, recognising identical features regardless of the spatial locations. Several similar successful approaches have been developed, and we can mention Chebyshev networks [124] and SAGE [127].
- **Graph attention network (GAT):** a well-known shortcoming of the graph convolutions procedure is that they consider every node neighbour as having the same importance. GATs are neural networks architectures that leverage masked self-attentional layers to address this shortcoming [128–130].
- **Other GNNs:** several other embedding methods based on alternative architectures of GNNs exist [131–133]. The interested reader can refer to the review of Zhou *et al* on GNNs for more details [134].

4.2.3. Graph generative methods

Graph generative methods are other deep learning methods mostly known for generative adversarial networks (GANs) [135]. The principle of GANs is based on two components: a generator, and a discriminator. The idea of GANs is to train a generator until it is efficient enough to mislead the discriminator. The discriminator is misled when it cannot discriminate real data from the data generated by the generator. Based on this idea, several embedding methods appeared, including GraphGAN [136], adversarial network embedding [137], and ProGAN [138]. In the case of GraphGAN, for a given vertex, the generator tries to fit its underlying true connectivity distribution over all other vertices and produces 'fake' samples to fool the discriminator, while the discriminator tries to detect whether the sampled vertex is from the ground truth or generated by the generative model. An alternative method to GAN is the restricted Boltzmann machine [139], which inspired different embedding methods (pRBM [140]).

4.3. Higher-order network methods

We have seen in section 4.1 that shallow embedding methods use pairwise similarity functions. This choice is imposed by the structure of the graphs, which by definition connect nodes by pairwise interactions. However, generalisations of graphs that can encode higher-order interactions, such as hypergraphs and simplicial complexes, are increasingly being studied [141–144]. Hypergraphs encode arbitrary relations between any number of nodes, that is, edges are generalised to hyper-edges which can contain any number of nodes, not just two. Simplicial complexes generalise graphs by allowing triangles, tetrahedrons, and higher-dimensional 'cliques' to be represented, and are closely related to topology, particularly topological data analysis [145, 146]. Note that simplicial complexes are a type of hypergraphs, so, at least in principle, hypergraph methods also apply to simplicial complexes. Recently, network embedding methods have been extended to consider these new types of higher-order networks. In the case of higher-order networks, the adjacency matrix, which

describes pairwise interactions between nodes, cannot uniquely characterise their structure. This is not surprising, as an $n \times n$ matrix can only represent n^2 pairwise interactions between n objects. Instead, the incidence matrix (hypergraphs) or the boundary matrices (simplicial complexes) are chosen to characterise higher-order networks [141]. These are $n \times m$ matrices where n is the number of nodes and m the number of higher-order relations (of certain dimension, for a simplicial complex). Some of the existing network embedding methods described in sections 4. 1-2 have been extended to hypergraphs, using the incidence matrix of the hypergraph instead of the adjacency matrix. Methods capable of embedding hypergraphs include shallow embedding methods: learning hypergraph-regularised [147] (matrix factorisation method), spectral hypergraph embedding [36] (random walk method), LBSN2Vec++ [148] and hyperedge-based embedding [37, 149] (optimisation methods). There are also methods derived from deep learning processes: deep hypergraph network embedding [150] (autoencoder), hypergraph neural networks [38] (GNN), hypergraph attention embedding [151] (GAT). As explained above, simplicial complexes are a special type of hypergraphs which are amenable to be treated by powerful tools based on algebraic topology [152], including topology data analysis [145, 146]. The literature on simplicial complexes embedding and simplicial neural networks is rapidly growing [152]. It includes simplicial and cell complex neural networks [153–159] and geometric LEs embedding [160]. Simple graphs can also be described as higher-order set-of-sets formed by node neighbourhoods, for which recently a new graph embedding has been proposed (HATS) [161].

4.4. Emerging methods

Here we highlight some key emerging methods for network embeddings which deserve particular attention. Specifically, we discuss key results in the rapidly growing literature focusing on network embeddings in non-Euclidean spaces, including hyperbolic and Lorentzian spaces. Moreover, we cover the very vibrant research activity on a new generation of neural networks using magnetic and connection Laplacians which provide powerful tools to treat directed networks and to improve the explainability of the algorithms. Finally, we discuss the important research direction aiming at comparing different embedding algorithms.

4.4.1. Network embedding in hyperbolic and Lorentzian spaces

Embedding in hyperbolic spaces [162] offers advantages for representing hierarchical network data [163, 164] with applications ranging from the navigability of the internet [165] to the representation of natural systems including the olfactory system and neuronal circuits [166, 167]. Among the benefits of hyperbolic spaces, the most relevant one is probably the fact that hyperbolic spaces are natural spaces to embed trees having a number of nodes growing exponentially with the distance from the root, and in general to embed small-world networks. In the case of non-Euclidean space, two approaches can be distinguished: vector-based and point-based representations. In vector-based embeddings, nodes are represented as vectors, and the similarity is calculated using vector functions like the inner product. On the other hand, in point-based embeddings, each node is mapped to a point in (not necessarily a vector) space, and similarity is determined by the distance between these points. In Euclidean spaces, these approaches are equivalent, as any vector is uniquely defined by its endpoint starting from the origin. In non-Euclidean spaces, point-based representations are often favoured due to their ease of comparing two points, which is particularly useful for hyperbolic embeddings. Although vector-based representations are possible, comparing vectors from nodes located at different points can be challenging and may involve complex parallel transport techniques. We distinguish three major approaches to embedding networks in hyperbolic spaces:

- **Embedding based on the complex hyperbolic network models [168, 169] in the $\mathbb{H}^2$ plane.** According to the spatial hyperbolic network [168] and popularity-similarity optimisation(PSO) [169] models, the radial coordinate of the node embedding is determined by the degree of the nodes and the angular coordinate of the node embedding is determined by a similarity metric. The hyperbolic embedding can be used to formulate a greedy algorithm for network navigability [164, 170], to predict missing edges [171], and also to relate the clusters found along the angular coordinates to network communities [172, 173]. For a general review of this approach see [162]. The original embedding algorithm HyperMap [170], used to determine the angular coordinates of the nodes and revisited in [174], maximises the likelihood that the data is drawn from the model. Mercator [175] improves this algorithm by initialising the position of the nodes using LEs and, for each optimisation step, the angular positions of the nodes is updated by choosing, among a set of several possible moves, the one that optimises the likelihood. The possible new moves are drawn from a Gaussian distribution centred on the mean angles among the neighbour nodes. This algorithm has computational complexity $O(n^2)$ on sparse networks. A fast and efficient alternative to this approach is provided by the *noncentered minimum curvilinear embedding* (ncMCE) [176] which provides a machine learning pipeline including three main steps: (i) a pre-weighting procedure which identifies the network backbone; (ii) the extraction of the matrix **D** of nodes similarities (distances) measured on this network backbone (iii) a

dimensionality reduction of the matrix **D**. The ncMCE has been further extended in [177] to further reduce the loss calculated according to the PSO loglikelihood. Recently, an exact and rapid one-dimensional embedding algorithm [178] based on dynamic programming has been proposed. This algorithm can determine the angular coordinates of the hyperbolic embedding in $\mathbb{H}^2$ and, more generally, extract other types of one-dimensional embeddings. Hyperbolic latent space contains geometric information associated with the original network; in the context of $\mathbb{H}^2$, a recent study showed that shortest paths in the hyperbolic latent space are aligned along geodesic curves connecting endpoint nodes, and this alignment is sufficiently strong to allow the identification of shortest path nodes even in the case of substantially incomplete networks [179]. Finally, note that the hyperbolic embedding in the case of directed networks has been addressed in [180].

- **Embedding of hierarchical data.** Hyperbolic spaces allow the embedding of hierarchical data and in particular trees without distortion in spaces of low dimension [181] while the same data would require a high dimensional Euclidean embedding if low distortion is desired. This fundamental property of hyperbolic spaces has been exploited in [182] to first embed a network in a tree and then embed the tree into hyperbolic spaces achieving a fast and reliable hyperbolic embedding. This approach has been extended to knowledge graphs in [183] while in [184] hyperbolic graph CNNs have been proposed to learn the hyperbolic embedding. Note that embedding a network into another network (which might not in general be a tree) is a generalised embedding problem tackled also in [185].
- **Filtering of networks generating the triangulated maximally filtered graph (TMFG) [186].** The TMFG generalises the minimal spanning tree and has the topology of a 'fat tree' formed by d connected $(d+1)$-cliques (d-simplices). The structure of TMFG reduces to the structure of the model *network geometry with flavour* [187] with natural hyperbolic embedding in $\mathbb{H}^d$ [188]. Therefore TMFGs have a hyperbolic geometry while the previously proposed maximally filtered planar graphs [189] have a natural $\mathbb{R}^2$ embedding.

Finally, we point out also that network embeddings in Lorentzian (Minkowskian) spaces have been proposed [190] and applied to the study of directed acyclic networks as citation networks. This embedding can be used for paper recommendation, identifying missing citations, and fitting citation models.

4.4.2. Network embeddings using magnetic and connection Laplacians

Despite many networks being directed, machine learning methods are typically developed for undirected networks. One challenge in directed networks is that they are naturally encoded by asymmetric adjacency matrices with a complex spectrum while machine learning techniques usually require a loss function that is real and positive definite. In order to address this challenge, the use of magnetic Laplacians is attracting growing attention. Magnetic Laplacians are Hermitian matrices (hence having a real and non-negative spectrum) that encode the direction of edges through a complex phase. Using a magnetic Laplacian, it is possible to formulate Eigenmap embeddings [191–193] that can detect non-trivial periodic structure in the network such as three or more communities whose connection pattern is cyclic. Additionally, in [194], the magnetic Laplacian is also used to propose *magnet*, a novel and efficient neural network for directed networks. This vibrant research activity on the magnetic Laplacian is indicative of the recent interest in network structure with complex weights [195]. The magnetic Laplacian can be considered a special case of the connection Laplacian, which can be used for vector diffusion maps [196]. Moreover, the connection Laplacian is used for formulating Sheaf neural networks [197, 198], which are a new generation of neural networks obtaining excellent performance on several machine learning tasks. Finally, the non-backtracking matrix that identifies non-backtracking cycles and efficiently detects the network communities [199], has been recently proposed for embedding oriented edges (non-backtracking embedding dimensions) [200].

4.4.3. Comparison of different algorithms

An important question is how to choose among the different network embedding algorithms and how to select their hyperparameters. For instance, the selection of the embedding dimension is a crucial hyperparameter for network embedding algorithms. Gu *et al* [201] provides a method to determine the value of the embedding dimension that constitutes the best trade-off between favouring low dimensions and requiring the generation of a network representation close to the best representation that the chosen algorithm can achieve. The comparison between different embedding algorithms [202] and the investigation of network embedding algorithms as opposed to alternative inference approaches such as community detection [203] is attracting increasing attention and can guide the choice of the most suitable algorithm. A conventional approach to evaluating the results of network embedding methods is to compare their performance across various tasks, such as link prediction, network reconstruction, or node clustering. Most of these tasks rely on cross-validation or the computation of an $F1$ score, which assesses the quality of predictions made by the given embedding method. Finally, we note that a unifying approach of node embeddings and structural graph representation has been recently proposed [204].

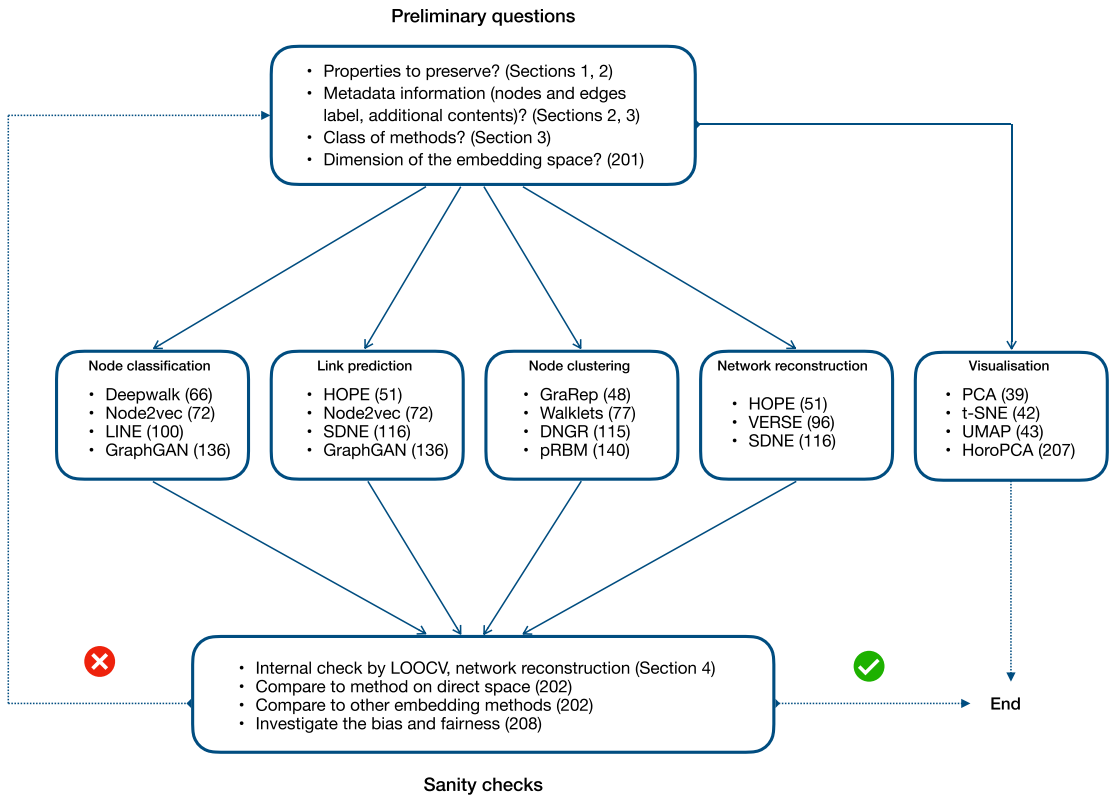

Figure 3. Workflow for choosing a network embedding method. The application of network embedding starts with some preliminary questions (top box). Depending on the answers, and the task to be performed, different network embedding methods can be applied. Here, we list the most common methods associated with the most common network embedding tasks. Once the embedding representation has been obtained, the workflow could further perform some 'sanity checks' to measure the efficiency of the network embedding setup (bottom box). Depending on the results of these sanity checks, we can either stop the development or go back to the preliminary questions in order to improve the workflow. These improvements are usually based on adding complementary information, tuning the parameters, or redefining the properties to be preserved by the embedding representation.

5. Applications

A wide variety of applications exist for network embedding methods. Some methods have been designed focusing on specific task(s) but others admit versatile applications. In the following sections, we will present the most common applications of network embedding methods as well as some emergent applications offering promising results. We associate each application with some widely used embedding methods in order to give the interesting reader a guideline for applying network embedding in different tasks (figure 3).

5.1. Classical applications of network embedding

- **Node classification:** the aim is to predict labels for unlabelled nodes based on the information learned from the labelled nodes. Network embedding methods embed nodes into vectors which can be used in an unsupervised setting. In this case, the nodes associated with similar node embedding vectors will have similar labels. In a supervised setting, the classifier is trained with the vectors associated with the labelled nodes. The classifier is then applied to predict the labels of unlabelled nodes.
- **Link prediction:** the aim is to infer new interactions between pairs of nodes in a network. The similarity between the nodes encodes the propensity of the nodes to be linked. The similarity can be computed, for instance, with an inner product or a cosine similarity between each pair of node embedding vectors. In the embedding space, several operators exist to infer edges between pairs of embedding vectors [72]. These operators can be, for instance, binary operators (table 1), or, heuristic scores (table 2). Link prediction can extend to non-Euclidean latent spaces, such as hyperbolic latent spaces, which exhibit several compelling properties, hinting at their potential as more powerful tools for link prediction; an example is the HYPERLINK embedder [171], an optimised embedding method specifically designed for link prediction.
- **Node clustering/community detection:** The aim is to determine a partition of the network such that the nodes belonging to a given cluster are more similar to each other than to the nodes belonging to other clusters. In practice, any classic clustering method can be directly applied in the latent space to cluster the

Table 1. Binary operators to compute infer edges between pairs of embedding vectors. The variable z_i defines the embedding vector associated with the node v_i, and $z_i(k)$ defines the kth element of the embedding vector z_i.

Operator	Definition
Average	$\frac{z_i(k)+z_j(k)}{2}$
Hadamard	$z_i(k) \odot z_j(k)$
Weighted-L1	$\lvert z_i(k) - z_j(k)\rvert$
Weighted-L2	$\lvert z_i(k) - z_j(k)\rvert^2$
Cosine	$\frac{z_i(k)\cdot z_j(k)}{\lVert z_i\rVert \lVert z_j\rVert}$

Table 2. Heuristic scores are used to predict edges between pairs of nodes in the direct space. The variable $\mathcal{N}(v_i)$ defines the neighbour set of nodes associated with the node v_i.

Score	Definition
Common neighbours	$\lvert\mathcal{N}(v_i) \cap \mathcal{N}(v_j)\rvert$
Jaccard's coefficient	$\lvert\frac{\mathcal{N}(v_i)\cap\mathcal{N}(v_j)}{\mathcal{N}(v_i)\cup\mathcal{N}(v_j)}\rvert$
Adamic–Adar score	$\sum_{t\in\mathcal{N}(v_i)\cap\mathcal{N}(v_j)} \frac{1}{\ln\lvert\mathcal{N}(v_i)\rvert}$
Preferential attachment	$\lvert\mathcal{N}(v_i)\rvert\lvert\mathcal{N}(v_j)\rvert$

nodes. K-means [205] is often used for this purpose. However, it is useful to notice that some embedding methods are designed specifically for this task [47]. A recent study [206], illustrate that classical network embedding methods, without non-linear activation, such as DeepWalk [66], node2vec [72], or LINE [100] encode community structure as well as, or even better than, spectral embedding methods for both dense and sparse networks, with and without degree and community size heterogeneity. Furthermore, this result suggest that the two common components of deep learning embedding, the 'deep' layers and the non-linear activation are not necessary to perform community detection in the embedding space context.

- **Network reconstruction:** the aim here is to reconstruct the whole network based on the learned embedding representation. Let us define n as the total number of nodes in the network. The reconstruction imposes $n(n-1)/2$ evaluations to test each potential edge, and each evaluation is equivalent to a link prediction.
- **Visualisation:** network embedding methods, as other reduction dimension methods, can be used to visualise high-dimensional data in a lower-dimensional space. We expect similar nodes to be close to each other in the visualisation. However, network embedding methods that were not designed for this specific task show poor results when directly projected into a two-dimensional embedding space [30, 100]. To bypass this shortcoming, the visualisation of the result of network embedding methods is frequently projected into a d-dimensional embedding space ($2 < d \ll n$), and then into a two-dimensional space. The two-dimensional space is obtained with a dimension reduction method suitable for visualisation, like PCA [15, 39], t-SNE [42], or UMAP [43]. Recently, an extension of PCA for data lying in hyperbolic spaces has been developed [207].

  Finally, visualisation is a powerful tool to investigate the results obtained by embedding methods. The interpretability of results can be enhanced through visualisation, and a recent method has been proposed to address questions related to bias and fairness in results [208].

5.2. Emerging applications

- **Network compression and coarsening:** the aim of network compression is to convert a large network into a smaller network containing a reduced number of nodes and edges. This compression is expected to store the network more efficiently and to allow running the network algorithms faster. Network coarsening is often used as a preliminary step in the network embedding process to produce a compression by collapsing pairs of nodes and edges with appropriate criteria. One example is network compression using symmetries. Real-world networks have a large number of symmetries [209, 210] and this can be exploited in practice for compression or coarsening, as well as for speeding up calculations [211, 212].
- **Network classification:** the aim is to associate a label to a whole network. Network classification can easily be applied in the context of whole network embedding. A wide range of applications has been proposed, such as classifying molecular networks according to their properties [120, 213–215], predicting therapeutic effects of candidate drugs based on molecular networks [215], or classifying images that have been converted into networks representation [216].
- **Applications to knowledge graphs:** let us consider a knowledge graph defined by a set of triples $(u, r, v) \in V \times R \times V$. There are three main applications of embedding for knowledge graphs. Link prediction is used to infer the interaction between u and v. Triplet classification, which is a standard binary classification task,

determines if a given triplet (u, r, v) is correct. And finally, knowledge completion aims to determine the missing element in a triplet where only two of the pieces of information are known [29, 217].

- **Illustration of network embedding with biological applications:** network embedding is an active research topic in bioinformatics [34], and all the classical applications previously mentioned also flourish in the context of biological networks. Some applications emerge as particularly relevant to network biology. We describe here some of them and the interested reader can refer to [10, 34] for detailed reviews. A first important application in biology is related to network alignment, which aims to find correspondences between nodes in different networks. This can be useful to reveal similar subnetworks, and thereby biological processes, in different species, by aligning their protein–protein interaction networks [218, 219]. Another important application pertains to network denoising, which consists of projecting a graph into an embedding space to reduce noise by preserving the most relevant properties of the original network. For instance, high-order structures of the original networks can be preserved by diffusion processes. Network embedding methods can also be used to predict the functions of proteins [104], or to detect modules in chromosome conformation networks [10]. LEs have also been used to map the niche space of bacterial ecosystems from genomic information [220]. In genomics, UMAP embedding has been shown to be useful to identify overlooked sub-populations and characterise fine-scale relationships between geography, genotypes, and phenotypes in a given population [221]. Moreover, the prediction of edges (also known as associations in the biological context) is a common task in biological network embedding. For instance, a recent method named GCN-MF [222] used GCNs and matrix factorisation to predict gene–disease associations. Another method, named NEDTP [108], used a multiplex network embedding based on an optimisation method to predict drug–target associations. Finally, knowledge graphs are also used in biomedical contexts. For example, electronic health record can be represented as a knowledge graph and embedded jointly with other networks integrating proteins, diseases, or drug information, to predict patient outcomes [223, 224].

6. Conclusion

Network embedding methods are powerful tools for transforming high-dimensional networks into low-dimensional vector representations that can be used for visualisation and a wide range of downstream analyses, such as network inference, link prediction, node classification, and community detection. The resulting low-dimensional representation also enables the use of deep-learning analysis, which is often not possible directly on the original network as it is inherently a combinatorial object. Moreover, the latent space generated through the embedding process can highlight relevant features of the original network and filter out the noise inherent in the dataset used to construct the network. This helps to reveal the underlying structure and organisation of the network, making it easier to analyse and interpret its properties. Furthermore, employing network embedding methods in non-Euclidean spaces may offer more relevant representations for nonlinear data. For instance, hyperbolic geometry provides a more natural space for representing scale-free networks and hierarchical structures.

More generally, let us mention the large body of literature in discrete and computational geometry about practical and theoretical methods to embed a general metric space (such as a graph with a choice of metric) into a normed or a metric space of low dimension with minimal distortion (a formal way of comparing the difference between the source and the target metrics in an embedding), see e.g. [225–229]. The focus of this approach is often on the optimal (minimum) distortion possible from one type of space into another (see e.g. table 8.4.1 in [229]); this, beyond a guiding principle, may be less relevant on practical applications, where the typical performance in practical real-world scenarios is decisive. Therefore, we have focused on practical embedding network embedding methods in this review.

Despite the numerous advantages of network embedding methods, there are still significant questions surrounding their use, particularly with regard to selecting appropriate methods and assessing their properties, interpreting the results, and ensuring adaptability across a range of contexts. These questions are especially critical as datasets become more complex and require more sophisticated tools to achieve the desired level of adaptability and interpretability, particularly in the biological contexts, where datasets may be particularly noisy, leading to errors in the learning process.

Another relevant aspect is quantifying the quality of the embedding by comparing it to methods operating directly in the original space. For example, Nelson *et al* [10] showed that, in the context of biological networks, working in the latent space is not consistently superior to working in the original space, with the exception of clustering. This observation can be explained by the fact that most embedding methods preserve the neighbourhood structure, which significantly influences node clustering. However, when addressing tasks like link prediction or network alignment, this question needs careful consideration. We therefore advocate more awareness to this issue in current and future network embedding methods.

In recent years, there has been a substantial increase in the number of network embedding methods, making it challenging to stay up-to-date in this rapidly evolving field. Therefore, it is crucial to have access to a comprehensive review of the state-of-the-art methods. Our network embedding review provides not only a summary of the current state-of-the-art, but also lays the foundation for future developments by presenting a flexible yet rigorous mathematical perspective-based taxonomy of embedding methods. Despite the many advancements in the field, challenges remain regarding the selection and interpretability of these methods. To address these issues, we offer a set of guidelines for practical applications that take into account these considerations and are aligned with the latest developments in the field.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

We acknowledge interesting discussions with Filippo Radicchi and funding from the Roche-Turing Partnership (A Baptista, R S-G, G B) and the 'Investissements d'Avenir' French Government program managed by the French National Research Agency (ANR-16-CONV-0001 and ANR-21-CE45-0001-01) (A Baudot). We also acknowledge Galadriel Brière for interesting discussions.

ORCID iDs

A Baptista https://orcid.org/0000-0002-8514-0250
A Baudot https://orcid.org/0000-0003-0885-7933
G Bianconi https://orcid.org/0000-0002-3380-887X

References

[1] Newman M 2018 *Networks* (Oxford University Press)
[2] Barabási A L 2013 Network science *Phil. Trans. R. Soc.* A **371** 20120375
[3] Borgatti S P, Everett M G and Johnson J C 2018 *Analyzing Social Networks* (SAGE)
[4] Pastor-Satorras R, Castellano C, Van Mieghem P and Vespignani A 2015 *Rev. Mod. Phys.* **87** 925
[5] Junker B H and Schreiber F 2011 *Analysis of Biological Networks* (Wiley)
[6] Alm E and Arkin A P 2003 Biological networks *Curr. Opin. Struct. Biol.* **13** 193–202
[7] Cui P, Wang X, Pei J and Zhu W 2019 A Survey on Network Embedding *IEEE Trans. Knowl. Data Eng.* **31** 833–52
[8] De Domenico M 2017 Diffusion geometry unravels the emergence of functional clusters in collective phenomena *Phys. Rev. Lett.* **118** 168301
[9] Bertagnolli G and De Domenico M 2021 Diffusion geometry of multiplex and interdependent systems *Phys. Rev.* E **103** 042301
[10] Nelson W, Zitnik M, Wang B, Leskovec J, Goldenberg A and Sharan R 2019 To embed or not: network embedding as a paradigm in computational biology *Front. Genet.* **10**
[11] *The KONECT Project* (available at: http://konect.cc/) (Accessed 22 March 2023)
[12] *Netzschleuder Network Catalogue* (available at: https://networks.skewed.de/) (Accessed 22 March 2023)
[13] *UCI Network Data Repository* (available at: http://networkdata.ics.uci.edu/) (Accessed 22 March 2023)
[14] Chari T and Pachter L 2023 The specious art of single-cell genomics *PLoS Comput. Biol.* **19** 8
[15] Jolliffe I T and Cadima J 2016 Principal component analysis: a review and recent developments *Phil. Trans. R. Soc.* A **374** 20150202
[16] Robinson S L and Bennett R J 1995 A typology of deviant workplace behaviors: a multidimensional scaling study *Acad. Manage. J.* **38** 555–72
[17] Ye J, Janardan R and Li Q 2005 Two-dimensional linear discriminant analysis *Advances in Neural Information Processing Systems* vol 17, ed L Saul, Y Weiss and L Bottou (MIT Press) (available at: https://proceedings.neurips.cc/paper/2004/file/86ecfcbc1e9f1ae5ee2d71910877da36-paper.pdf)
[18] Hamilton W L, Ying R and Leskovec J 2018 Representation learning on graphs: methods and applications (arXiv:1709.05584)
[19] Chami I, Abu-El-Haija S, Perozzi B, Ré C and Murphy K 2022 *J. Mach. Learn. Res.* **23** 1–64
[20] Chen H, Perozzi B, Al-Rfou R and Skiena S 2018 *CoRR* (arXiv:1808.02590)
[21] Bianconi G 2018 *Multilayer Networks: Structure and Function* (Oxford University Press)
[22] Boccaletti S, Bianconi G, Criado R, del Genio C I, Gómez-Gardeñes J, Romance M, Sendiña-Nadal I, Wang Z and Zanin M 2014 *Phys. Rep.* **544** 1–122
[23] De Domenico M, Granell C, Porter M A and Arenas A 2016 The physics of spreading processes in multilayer networks *Nat. Phys.* **12** 901–6
[24] Kivelä M, Arenas A, Barthelemy M, Gleeson J P, Moreno Y and Porter M A 2014 *J. Complex Netw.* **2** 203–71
[25] De Domenico M, Solé-Ribalta A, Cozzo E, Kivelä M, Moreno Y, Porter M A, Gómez S and Arenas A 2013 *Phys. Rev.* X **3** 041022
[26] De Domenico M, Solé-Ribalta A, Gómez S and Arenas A 2014 Navigability of interconnected networks under random failures *Proc. Natl Acad. Sci.* **111** 8351–6
[27] Masuda N and Lambiotte R 2016 *A Guide to Temporal Networks* (World Scientific)
[28] Holme P and Saramäki J 2012 *Phys. Rep.* **519** 97–125
[29] Li B and Pi D 2020 *Neural Comput. Appl.* **32** 16647–79
[30] Zhang D, Yin J, Zhu X and Zhang C 2020 Network Representation Learning: A Survey *IEEE Trans. Big Data* **6** 3–28
[31] Feng R, Yang Y, Hu W, Wu F and Zhuang Y 2018 arXiv:1711.10755

[32] Goyal P and Ferrara E 2018 Graph embedding techniques, applications, and performance: A survey *Knowl.-Based Syst.* **151** 78–94
[33] Chen F, Wang Y C, Wang B and Kuo C C J 2020 *APSIPA Trans. Signal Inf. Process.* **9** e15
[34] Li M M, Huang K and Zitnik M 2022 Graph representation learning in biomedicine and healthcare *Nat. Biomed. Eng.* **6** 1353–69
[35] Yang C, Xiao Y, Zhang Y, Sun Y and Han J 2020 *IEEE Trans. Knowl. Data Eng.* **34** 1843–55
[36] Zhou D, Huang J and Schölkopf B 2006 Learning with hypergraphs: clustering, classification and embedding *Proc. 19th Int. Conf. on Neural Information Processing Systems (NIPS '06)* (MIT Press) pp 1601–8
[37] Gui H, Liu J, Tao F, Jiang M, Norick B and Han J 2016 Large-scale embedding learning in heterogeneous event data *2016 IEEE 16th Int. Conf. on Data Mining (ICDM)* pp 907–12
[38] Feng Y, You H, Zhang Z, Ji R and Gao Y 2019 Hypergraph neural networks *Proc. AAAI Conf. on Artificial Intelligence* vol 33 pp 3558–65
[39] Wold S, Esbensen K and Geladi P 1987 Principal component analysis *Chemometr. Intell. Lab. Syst.* **2** 37–52
[40] Samko O, Marshall A and Rosin P 2006 Selection of the optimal parameter value for the Isomap algorithm *Pattern Recognit. Lett.* **27** 968–79
[41] Roweis S T and Saul L K 2000 Nonlinear Dimensionality Reduction by Locally Linear Embedding *Science* **290** 2323–6
[42] van der Maaten L and Hinton G 2008 *J. Mach. Learn. Res.* **9** 2579–605
[43] McInnes L, Healy J, Saul N and Großberger L 2018 UMAP: Uniform Manifold Approximation and Projection *J. Open Source Softw.* **3** 861
[44] Belkin M and Niyogi P 2003 Laplacian Eigenmaps for Dimensionality Reduction and Data Representation *Neural Comput.* **15** 1373–96
[45] Ghojogh B, Karray F and Crowley M 2019 Eigenvalue and generalized eigenvalue problems: tutorial (arXiv:1903.11240)
[46] Luo D, Ding C H Q, Nie F and Huang H 2011 Cauchy graph embedding *Int. Conf. on Machine Learning* pp 553–60 (available at: https://icml.cc/2011/papers/353_icmlpaper.pdf)
[47] Ahmed A, Shervashidze N, Narayanamurthy S, Josifovski V and Smola A J 2013 Distributed large-scale natural graph factorization *Proc. 22nd Int. Conf. on World Wide Web (WWW '13)* (Association for Computing Machinery) pp 37–48
[48] Cao S, Lu W and Xu Q 2015 GraRep: learning graph representations with global structural information *Proc. 24th ACM Int. on Conf. on Information and Knowledge Management (CIKM '15)* (Association for Computing Machinery) pp 891–900
[49] Mikolov T, Chen K, Corrado G and Dean J 2013 Efficient estimation of word representations in vector space *1st Int. Conf. on Learning Representations (ICLR 2013) (Scottsdale, Arizona, USA, 2–4 May 2013)* (*Workshop Track Proc.*) (arXiv:1301.3781)
[50] Gutmann M and Hyvärinen A 2010 Noise-contrastive estimation: a new estimation principle for unnormalized statistical models *Proc. 13th Int. Conf. on Artificial Intelligence and Statistics* (*Proc. Machine Learning Research* vol 9) ed Y W Teh and M Titterington (PMLR) pp 297–304 (available at: https://proceedings.mlr.press/v9/gutmann10a.html)
[51] Ou M, Cui P, Pei J, Zhang Z and Zhu W 2016 Asymmetric transitivity preserving graph embedding *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '16)* (Association for Computing Machinery) pp 1105–14
[52] Wang X, Cui P, Wang J, Pei J, Zhu W and Yang S 2017 Community preserving network embedding *Proc. 31st AAAI Conf. on Artificial Intelligence (AAAI '17)* (AAAI Press) pp 203–9
[53] Yang C, Liu Z, Zhao D, Sun M and Chang E Y 2015 Network representation learning with rich text information *Proc. 24th Int. Conf. on Artificial Intelligence (IJCAI '15)* (AAAI Press) pp 2111–7
[54] Yu H F, Jain P, Kar P and Dhillon I 2014 Large-scale multi-label learning with missing labels *Proc. 31st Int. Conf. on Machine Learning* (*Proc. Machine Learning Research* vol 32) ed E P Xing and T Jebara (PMLR) pp 593–601 (available at: https://proceedings.mlr.press/v32/yu14.html)
[55] He X and Niyogi P 2004 *Advances in Neural Information Processing Systems* vol 16 pp 153–60
[56] Shaw B and Jebara T 2009 Structure preserving embedding *Proc. 26th Annual Int. Conf. on Machine Learning (ICML '09)* (Association for Computing Machinery) pp 937–44
[57] Lin Y Y, Liu T L and Chen H T 2005 Semantic manifold learning for image retrieval *Proc. 13th Annual ACM Int. Conf. on Multimedia (MULTIMEDIA '05)* (Association for Computing Machinery) pp 249–58
[58] Huang X, Li J and Hu X 2017 Label informed attributed network embedding *Proc. 10th ACM Int. Conf. on Web Search and Data Mining (WSDM '17)* (Association for Computing Machinery) pp 731–9
[59] Zhang D, Yin J, Zhu X and Zhang C 2016 Homophily, structure and content augmented network representation learning *2016 IEEE 16th Int. Conf. on Data Mining (ICDM)* pp 609–18
[60] Tu C, Zhang W, Liu Z and Sun M 2016 Max-margin deepwalk: discriminative learning of network representation *Proc. 25th Int. Joint Conf. on Artificial Intelligence (IJCAI '16)* (AAAI Press) pp 3889–95
[61] Zhang D, Yin J, Zhu X and Zhang C 2016 Collective classification via discriminative matrix factorization on sparsely labeled networks *Proc. 25th ACM Int. on Conf. on Information and Knowledge Management (CIKM '16)* (Association for Computing Machinery) pp 1563–72
[62] Nickel M, Tresp V and Kriegel H P 2012 Factorizing YAGO: scalable machine learning for linked data *Proc. 21st Int. Conf. on World Wide Web (WWW '12)* (Association for Computing Machinery) pp 271–80
[63] Yang B, tau Yih W, He X, Gao J and Deng L 2015 *CoRR* (arXiv:1412.6575)
[64] Trouillon T, Welbl J, Riedel S, Gaussier E and Bouchard G 2016 Complex embeddings for simple link prediction *Proc. 33rd Int. Conf. on Int. Conf. on Machine Learning (ICML '16)* vol 48 (JMLR.org) pp 2071–80
[65] Schaub M T, Delvenne J C, Lambiotte R and Barahona M 2019 Multiscale dynamical embeddings of complex networks *Phys. Rev.* E **99** 062308
[66] Perozzi B, Al-Rfou R and Skiena S 2014 DeepWalk: online learning of social representations *Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '14)* (Association for Computing Machinery) pp 701–10
[67] Mikolov T, Sutskever I, Chen K, Corrado G S and Dean J 2013 Distributed representations of words and phrases and their compositionality *Advances in Neural Information Processing Systems* vol 26, ed C J C Burges, L Bottou, M Welling, Z Ghahramani and W (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-paper.pdf)
[68] Mnih A and Hinton G E 2008 A scalable hierarchical distributed language model *Advances in Neural Information Processing Systems* vol 21
[69] 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 *Proc. 11th ACM Int. Conf. on Web Search and Data Mining (WSDM '18)* (Association for Computing Machinery) pp 459–67

[70] Cen Y, Zou X, Zhang J, Yang H, Zhou J and Tang J 2019 Representation learning for attributed multiplex heterogeneous network *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '19)* (Association for Computing Machinery) pp 1358–68
[71] Dursun C, Smith J R, Hayman G, Kwitek A E and Bozdag S 2020 NECo: a node embedding algorithm for multiplex heterogeneous networks *2020 IEEE Int. Conf. on Bioinformatics and Biomedicine (BIBM)* (IEEE Computer Society) pp 146–9
[72] Grover A and Leskovec J 2016 node2vec: scalable feature learning for networks (arXiv:1607.00653)
[73] Wilson J D, Baybay M, Sankar R and Stillman P E 2018 arXiv:1809.06437
[74] Liu W, Chen P Y, Yeung S, Suzumura T and Chen L 2017 Principled multilayer network embedding *2017 IEEE Int. Conf. on Data Mining Workshops (ICDMW)* pp 134–41
[75] Chen H, Perozzi B, Hu Y and Skiena S 2018 HARP: hierarchical representation learning for networks *32nd AAAI Conf. on Artificial Intelligence*
[76] Li J, Zhu J and Zhang B 2016 Discriminative deep random walk for network classification *Proc. 54th Annual Meeting of the Association for Computational Linguistics*
[77] Perozzi B, Kulkarni V, Chen H and Skiena S 2017 Don't walk, skip! Online learning of multi-scale network embeddings *Proc. 2017 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining 2017 (ASONAM '17)* (Association for Computing Machinery) pp 258–65
[78] Yang C and Liu Z 2015 arXiv:1501.00358
[79] Ribeiro L F, Saverese P H and Figueiredo D R 2017 *struc2vec*: learning node representations from structural identity *Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '17)* (Association for Computing Machinery) pp 385–94
[80] Salvador S and Chan P 2007 Toward accurate dynamic time warping in linear time and space *Intell. Data Anal.* **11** 561–80
[81] Li C, Li Z, Wang S, Yang Y, Zhang X and Zhou J 2017 Semi-supervised network embedding *Database Systems for Advanced Applications* ed S Candan, L Chen, T B Pedersen, L Chang and W Hua (Springer International Publishing) pp 131–47
[82] Chen J, Zhang Q and Huang X 2016 Incorporate group information to enhance network embedding *Proc. 25th ACM Int. on Conf. on Information and Knowledge Management (CIKM '16)* (Association for Computing Machinery) pp 1901–4
[83] Pan S, Wu J, Zhu X, Zhang C and Wang Y 2016 Tri-party deep network representation *Proc. 25th Int. Joint Conf. on Artificial Intelligence (IJCAI' 16)* (AAAI Press) pp 1895–901
[84] Lyu T, Zhang Y and Zhang Y 2017 Enhancing the network embedding quality with structural similarity *Proc. 2017 ACM on Conf. on Information and Knowledge Management (CIKM '17)* (Association for Computing Machinery) pp 147–56
[85] Wu F, Lu X, Song J, Yan S, Zhang Z M, Rui Y and Zhuang Y 2016 Learning of Multimodal Representations With Random Walks on the Click Graph *IEEE Trans. Image Process.* **25** 630–42
[86] Zhang C, Swami A and Chawla N V 2019 SHNE: representation learning for semantic-associated heterogeneous networks *Proc. 12th ACM Int. Conf. on Web Search and Data Mining (WSDM '19)* (Association for Computing Machinery) pp 690–8
[87] Wang X, Zhang Y and Shi C 2019 Heterogeneous graph attention network *Proc. AAAI Conf. on Artificial Intelligence* vol 33 pp 5337–44
[88] Chen T and Sun Y 2017 Task-guided and path-augmented heterogeneous network embedding for author identification *Proc. 10th ACM Int. Conf. on Web Search and Data Mining (WSDM '17)* (Association for Computing Machinery) pp 295–304
[89] Hussein R, Yang D and Cudré-Mauroux P 2018 Are meta-paths necessary? Revisiting heterogeneous graph embeddings *Proc. 27th ACM Int. Conf. on Information and Knowledge Management (CIKM '18)* (Association for Computing Machinery) pp 437–46
[90] He Y, Song Y, Li J, Ji C, Peng J and Peng H 2019 HeteSpaceyWalk: a heterogeneous spacey random walk for heterogeneous information network embedding *Proc. 28th ACM Int. Conf. on Information and Knowledge Management (CIKM '19)* (Association for Computing Machinery) pp 639–48
[91] Park C, Kim D, Zhu Q, Han J and Yu H 2019 Task-guided pair embedding in heterogeneous network *Proc. 28th ACM Int. Conf. on Information and Knowledge Management (CIKM '19)* (Association for Computing Machinery) pp 489–98
[92] Dong Y, Chawla N V and Swami A 2017 metapath2vec: scalable representation learning for heterogeneous networks *Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '17)* (Association for Computing Machinery) pp 135–44
[93] Fu T Y, Lee W C and Lei Z 2017 HIN2Vec: explore meta-paths in heterogeneous information networks for representation learning *Proc. 2017 ACM on Conf. on Information and Knowledge Management (CIKM '17)* (Association for Computing Machinery) pp 1797–806
[94] Huang Z and Mamoulis N 2017 arXiv:1701.05291
[95] Shi C, Hu B, Zhao W X and Yu P S 2019 Heterogeneous Information Network Embedding for Recommendation *IEEE Trans. Knowl. Data Eng.* **31** 357–70
[96] Tsitsulin A, Mottin D, Karras P and Müller E 2018 VERSE: versatile graph embeddings from similarity measures *Proc. 2018 World Wide Web Conf. (WWW '18)* (International World Wide Web Conferences Steering Committee) pp 539–48
[97] Jeh G and Widom J 2002 SimRank: a measure of structural-context similarity *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '02)* (Association for Computing Machinery) pp 538–43
[98] Mnih A and Teh Y W 2012 A fast and simple algorithm for training neural probabilistic language models *Proc. 29th Int. Coference on Int. Conf. on Machine Learning (ICML '12)* (Omnipress) pp 419–26
[99] Pio-Lopez L, Valdeolivas A and Tichit L 2020 Multiverse: a multiplex and multiplex-heterogeneous network embedding approach *Sci. Rep.* **11** 8794
[100] Tang J, Qu M, Wang M, Zhang M, Yan J and Mei Q 2015 LINE: large-scale information network embedding *Proc. 24th Int. Conf. on World Wide Web (WWW '15)* (International World Wide Web Conferences Steering Committee) pp 1067–77
[101] Zhang X, Chen W and Yan H 2016 TLINE: scalable transductive network embedding *Information Retrieval Technology* ed S Ma, J R Wen, Y Liu, Z Dou, M Zhang, Y Chang and X Zhao (Springer International Publishing) pp 98–110
[102] Tang J, Qu M and Mei Q 2015 PTE: predictive text embedding through large-scale heterogeneous text networks *Proc. 21th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '15)* (Association for Computing Machinery) pp 1165–74
[103] Zhou C, Liu Y, Liu X, Liu Z and Gao J 2017 Scalable graph embedding for asymmetric proximity *Proc. AAAI Conf. on Artificial Intelligence* vol 31
[104] Zitnik M and Leskovec J 2017 Predicting multicellular function through multi-layer tissue networks *Bioinformatics* **33** i190–8
[105] Xu L, Wei X, Cao J and Yu P S 2017 Embedding of embedding (EOE): joint embedding for coupled heterogeneous networks *Proc. 10th ACM Int. Conf. on Web Search and Data Mining (WSDM '17)* (Association for Computing Machinery) pp 741–9
[106] Zhang H, Qiu L, Yi L and Song Y 2018 Scalable multiplex network embedding *Int. Joint Conf. on Artificial Intelligence (IJCAI)*
[107] Bagavathi A and Krishnan S 2018 Multi-Net: a scalable multiplex network embedding framework *Complex Networks and Their Applications VII* (Springer International Publishing) pp 119–31

[108] An Q and Yu L 2021 *Brief. Bioinform.* **22** bbab275
[109] Bordes A, Usunier N, Garcia-Durán A, Weston J and Yakhnenko O 2013 Translating embeddings for modeling multi-relational data *Proc. 26th Int. Conf. on Neural Information Processing Systems (NIPS '13)* vol 2 (Curran Associates, Inc.) pp 2787–95
[110] Wang Z, Zhang J, Feng J and Chen Z 2014 Knowledge graph embedding by translating on hyperplanes *Proc. 28th AAAI Conf. on Artificial Intelligence (AAAI '14)* (AAAI Press) pp 1112–9
[111] Lin Y, Liu Z, Sun M, Liu Y and Zhu X 2015 Learning entity and relation embeddings for knowledge graph completion *Proc. 29th AAAI Conf. on Artificial Intelligence (AAAI' 15)* (AAAI Press) pp 2181–7
[112] Ji G, He S, Xu L, Liu K and Zhao J 2015 Knowledge graph embedding via dynamic mapping matrix *Proc. 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing (Volume 1: Long Papers)* (Association for Computational Linguistics) pp 687–96 (available at: https://aclanthology.org/P15-1067)
[113] Sun Z, Deng Z, Nie J Y and Tang J 2019 arXiv:1902.10197
[114] Chang S, Han W, Tang J, Qi G J, Aggarwal C C and Huang T S 2015 Heterogeneous network embedding via deep architectures *Proc. 21th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '15)* (Association for Computing Machinery) pp 119–28
[115] Cao S, Lu W and Xu Q 2016 Deep neural networks for learning graph representations *Proc. 30th AAAI Conf. on Artificial Intelligence (AAAI '16)* (AAAI Press) pp 1145–52
[116] Wang D, Cui P and Zhu W 2016 Structural deep network embedding *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '16)* (Association for Computing Machinery) pp 1225–34
[117] Kipf T and Welling M 2016 arXiv:1611.07308
[118] Alzubaidi L, Zhang J, Humaidi A J, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel M A, Al-Amidie M and Farhan L 2021 *J. Big Data* **8** 53
[119] Xie R, Liu Z, Jia J, Luan H and Sun M 2016 Representation learning of knowledge graphs with entity descriptions *Proc. 30th AAAI Conf. on Artificial Intelligence (AAAI '16)* (AAAI Press) pp 2659–65
[120] Niepert M, Ahmed M and Kutzkov K 2016 Learning convolutional neural networks for graphs *Proc. The 33rd Int. Conf. on Machine Learning* (*Proc. Machine Learning Research* vol 48) ed M F Balcan and K Q Weinberger (PMLR) pp 2014–23 (available at: https://proceedings.mlr.press/v48/niepert16.html)
[121] Gaier A and Ha D 2019 Weight agnostic neural networks *Advances in Neural Information Processing Systems* vol 32, ed H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox and R Garnett (Curran Associates, Inc.)
[122] Man T, Shen H, Liu S, Jin X and Cheng X 2016 Predict anchor links across social networks via an embedding approach *Proc. 25th Int. Joint Conf. on Artificial Intelligence (IJCAI '16)* (AAAI Press) pp 1823–9
[123] Li C, Ma J, Guo X and Mei Q 2017 DeepCas: an end-to-end predictor of information cascades *Proc. 26th Int. Conf. on World Wide Web (WWW '17)* (International World Wide Web Conferences Steering Committee) pp 577–86
[124] Defferrard M, Bresson X and Vandergheynst P 2016 Convolutional neural networks on graphs with fast localized spectral filtering *Proc. 30th Int. Conf. on Neural Information Processing Systems (NIPS '16)* (Curran Associates, Inc.) pp 3844–52
[125] Zitnik M, Agrawal M and Leskovec J 2018 Modeling polypharmacy side effects with graph convolutional networks *Bioinformatics* **34** i457–66
[126] Kipf T and Welling M 2017 arXiv:1609.02907
[127] Hamilton W L, Ying R and Leskovec J 2017 Inductive representation learning on large graphs *Proc. 31st Int. Conf. on Neural Information Processing Systems (NIPS '17)* (Curran Associates, Inc.) pp 1025–35
[128] Veličković P, Cucurull G, Casanova A, Romero A, Liò P and Bengio Y 2017 *6th Int. Conf. on Learning Representations*
[129] Xu Q, Wang Q, Xu C and Qu L 2017 Attentive graph-based recursive neural network for collective vertex classification *Proc. 2017 ACM on Conf. on Information and Knowledge Management (CIKM '17)* (Association for Computing Machinery) pp 2403–6
[130] Abu-El-Haija S, Perozzi B, Al-Rfou R and Alemi A A 2018 Watch your step: learning node embeddings via graph attention *Advances in Neural Information Processing Systems* vol 31, ed S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi and R Garnett (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper/2018/file/8a94ecfa54dcb88a2fa993bfa6388f9e-paper.pdf)
[131] Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi K-I and Jegelka S 2018 Representation learning on graphs with jumping knowledge networks *Int. Conf. on Machine Learning*
[132] Ying R, You J, Morris C, Ren X, Hamilton W L and Leskovec J 2018 Hierarchical graph representation learning with differentiable pooling *Proc. 32nd Int. Conf. on Neural Information Processing Systems (NIPS '18)* (Curran Associates, Inc.) pp 4805–15
[133] Zhang C, Song D, Huang C, Swami A and Chawla N V 2019 Heterogeneous graph neural network *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '19)* (Association for Computing Machinery) pp 793–803
[134] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C and Sun M 2020 *AI Open* **1** 57–81
[135] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial nets *Advances in Neural Information Processing Systems* vol 27 ed Z Ghahramani, M Welling, C Cortes, N Lawrence and K Weinberger (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-paper.pdf)
[136] Wang H, Wang J, Wang J, Zhao M, Zhang W, Zhang F, Xie X and Guo M 2018 *Proc. AAAI Conf. on Artificial Intelligence* vol 32 (available at: https://ojs.aaai.org/index.php/AAAI/article/view/11872)
[137] Dai Q, Li Q, Tang J and Wang D 2018 Adversarial network embedding *Proc. 32nd AAAI Conf. on Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conf. and 8th AAAI Symp. on Educational Advances in Artificial Intelligence (AAAI '18/IAAI '18/EAAI'18)* (AAAI Press)
[138] Gao H, Pei J and Huang H 2019 ProGAN: network embedding via proximity generative adversarial network *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '19)* (Association for Computing Machinery) pp 1308–16
[139] McClelland J L *et al* 1987 *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models* (MIT Press)
[140] Wang S, Tang J, Morstatter F and Liu H 2016 Paired restricted Boltzmann machine for linked data *Proc. 25th ACM Int. on Conf. on Information and Knowledge Management (CIKM '16)* (Association for Computing Machinery) pp 1753–62
[141] Bianconi G 2021 *Higher-Order Networks* (Cambridge University Press)
[142] Battiston F, Cencetti G, Iacopini I, Latora V, Lucas M, Patania A, Young J G and Petri G 2020 *Phys. Rep.* **874** 1–92
[143] Battiston F *et al* 2021 The physics of higher-order interactions in complex systems *Nat. Phys.* **17** 1093–8
[144] Torres L, Blevins A S, Bassett D and Eliassi-Rad T 2021 The why, how, and when of representations for complex systems *SIAM Rev.* **63** 435–85

[145] Salnikov V, Cassese D and Lambiotte R 2018 Simplicial complexes and complex systems *Eur. J. Phys.* **40** 014001
[146] Zomorodian A 2012 Topological data analysis *Advances in Applied and Computational Topology* vol 70 (American Mathematical Society) pp 1–39
[147] Huang S, Elhoseiny M, Elgammal A and Yang D 2015 Learning hypergraph-regularized attribute predictors *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 409–17
[148] Yang D, Qu B, Yang J and Cudre-Mauroux P 2020 *IEEE Trans. Knowl. Data Eng.* **34** 1843–55
[149] Gui H, Liu J, Tao F, Jiang M, Norick B, Kaplan L and Han J 2017 Embedding Learning with Events in Heterogeneous Information Networks *IEEE Trans. Knowl. Data Eng.* **29** 2428–841
[150] Tu K, Cui P, Wang X, Wang F and Zhu W 2018 Structural deep embedding for hyper-networks *Proc. 32nd AAAI Conf. on Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conf. and 8th AAAI Symp. on Educational Advances in Artificial Intelligence (AAAI '18/IAAI '18/EAAI '18)* (AAAI Press)
[151] Bai S, Zhang F and Torr P H 2021 Hypergraph convolution and hypergraph attention *Pattern Recognit.* **110** 107637
[152] Papillon M, Sanborn S, Hajij M and Miolane N 2023 arXiv:2304.10031
[153] Bodnar C, Frasca F, Wang Y, Otter N, Montufar G F, Lio P and Bronstein M 2021 Weisfeiler and Lehman go topological: message passing simplicial networks *Int. Conf. on Machine Learning* (PMLR) pp 1026–37
[154] Bodnar C, Frasca F, Otter N, Wang Y, Lio P, Montufar G F and Bronstein M 2021 *Advances in Neural Information Processing Systems* vol 34 pp 2625–40
[155] Ebli S, Defferrard M and Spreemann G 2020 arXiv:2010.03633
[156] Chen Y, Gel Y R and Poor H V 2022 BScNets: block simplicial complex neural networks *Proc. AAAI Conf. on Artificial Intelligence* vol 36 pp 6333–41
[157] Hajij M, Istvan K and Zamzmi G 2020 arXiv:2010.00743
[158] Schaub M T, Zhu Y, Seby J B, Roddenberry T M and Segarra S 2021 *Signal Process.* **187** 108149
[159] Giusti L, Battiloro C, Testa L, Di Lorenzo P, Sardellitti S and Barbarossa S 2022 arXiv:2209.08179
[160] Torres L, Chan K S and Eliassi-Rad T 2020 GLEE: Geometric Laplacian Eigenmap Embedding *J. Complex Netw.* **8** cnaa007
[161] Meng C, Yang J, Ribeiro B and Neville J 2019 HATS: a hierarchical sequence-attention framework for inductive set-of-sets embeddings *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining* pp 783–92
[162] Boguna M, Bonamassa I, De Domenico M, Havlin S, Krioukov D and Serrano M A 2021 Network geometry *Nat. Rev. Phys.* **3** 114–35
[163] Aste T, Di Matteo T and Hyde S 2005 Complex networks on hyperbolic surfaces *Physica* A **346** 20–26
[164] Kleinberg R 2007 Geographic routing using hyperbolic space *IEEE INFOCOM 2007—26th IEEE Int. Conf. on Computer Communications* (IEEE) pp 1902–9
[165] Boguna M, Krioukov D and Claffy K C 2009 Navigability of complex networks *Nat. Phys.* **5** 74–80
[166] Zhou Y, Smith B H and Sharpee T O 2018 Hyperbolic geometry of the olfactory space *Sci. Adv.* **4** eaaq1458
[167] Sharpee T O 2019 An argument for hyperbolic geometry in neural circuits *Curr. Opin. Neurobiol.* **58** 101–4
[168] Krioukov D, Papadopoulos F, Kitsak M, Vahdat A and Boguná M 2010 Hyperbolic geometry of complex networks *Phys. Rev.* E **82** 036106
[169] Papadopoulos F, Kitsak M, Serrano M A, Boguná M and Krioukov D 2012 Popularity versus similarity in growing networks *Nature* **489** 537–40
[170] Boguná M, Papadopoulos F and Krioukov D 2010 Sustaining the Internet with hyperbolic mapping *Nat. Commun.* **1** 62
[171] Kitsak M, Voitalov I and Krioukov D 2020 Link prediction with hyperbolic geometry *Phys. Rev. Res.* **2** 043113
[172] Faqeeh A, Osat S and Radicchi F 2018 Characterizing the Analogy Between Hyperbolic Embedding and Community Structure of Complex Networks *Phys. Rev. Lett.* **121** 098301
[173] Zuev K, Boguná M, Bianconi G and Krioukov D 2015 Emergence of Soft Communities from Geometric Preferential Attachment *Sci. Rep.* **5** 9421
[174] Nickel M and Kiela D 2017 *Advances in Neural Information Processing Systems* vol 30
[175] García-Pérez G, Allard A, Serrano M A and Boguñá M 2019 Mercator: uncovering faithful hyperbolic embeddings of complex networks *New J. Phys.* **21** 123033
[176] Muscoloni A, Thomas J M, Ciucci S, Bianconi G and Cannistraci C V 2017 Machine learning meets complex networks via coalescent embedding in the hyperbolic space *Nat. Commun.* **8** 1615
[177] Kovács B and Palla G 2021 Optimisation of the coalescent hyperbolic embedding of complex networks *Sci. Rep.* **11** 8350
[178] Patania A, Allard A and Young J G 2023 arXiv:2301.10403
[179] Kitsak M, Ganin A, Elmokashfi A, Cui H, Eisenberg D A, Alderson D L, Korkin D and Linkov I 2023 Finding shortest and nearly shortest path nodes in large substantially incomplete networks by hyperbolic mapping *Nat. Commun.* **14** 186
[180] Kovács B and Palla G 2023 Model-independent embedding of directed networks into Euclidean and hyperbolic spaces *Commun. Phys.* **6** 28
[181] Sarkar R 2012 Low distortion delaunay embedding of trees in hyperbolic plane *Graph Drawing: 19th Int. Symp., (GD 2011) (Eindhoven, The Netherlands, 21–23 September 2011)* (*Revised Selected Papers* vol 19) (Springer) pp 355–66
[182] Sala F, De Sa C, Gu A and Ré C 2018 Representation tradeoffs for hyperbolic embeddings *Int. Conf. on Machine Learning* (PMLR) pp 4460–9
[183] Chami I, Wolf A, Juan D C, Sala F, Ravi S and Ré C 2020 arXiv:2005.00545
[184] Chami I, Ying Z, Ré C and Leskovec J 2019 *Advances in Neural Information Processing Systems* vol 32
[185] Fernández-Gracia J and Onnela J P 2019 Flexible model of network embedding *Sci. Rep.* **9** 11710
[186] Massara G P, Di Matteo T and Aste T 2016 *J. Complex Netw.* **5** 161–78
[187] Bianconi G and Rahmede C 2016 Network geometry with flavor: From complexity to quantum geometry *Phys. Rev.* E **93** 032315
[188] Bianconi G and Rahmede C 2017 *Sci. Rep.* **7** 1–9
[189] Tumminello M, Aste T, Di Matteo T and Mantegna R N 2005 A tool for filtering information in complex systems *Proc. Natl Acad. Sci.* **102** 10421–6
[190] Clough J R and Evans T S 2017 Embedding graphs in Lorentzian spacetime *PLoS One* **12** e0187301
[191] Fanuel M, Alaíz C M, Fernández A and Suykens J A 2018 Magnetic eigenmaps for the visualization of directed networks *Appl. Comput. Harmon. Anal.* **44** 189–99
[192] Gong X, Higham D J and Zygalakis K 2021 Directed network Laplacians and random graph models *R. Soc. Open Sci.* **8** 211144
[193] Gong X, Higham D J and Zygalakis K 2023 Generative hypergraph models and spectral embedding *Sci. Rep.* **13** 540

[194] Zhang X, He Y, Brugnone N, Perlmutter M and Hirn M 2021 *Advances in Neural Information Processing Systems* vol 34 pp 27003–15
[195] Böttcher L and Porter M A 2022 arXiv:2212.06257
[196] Singer A and Wu H T 2012 Vector diffusion maps and the connection Laplacian *Commun. Pure Appl. Math.* **65** 1067–144
[197] Bodnar C, Di Giovanni F, Chamberlain B P, Liò P and Bronstein M M 2022 arXiv:2202.04579
[198] Barbero F, Bodnar C, de Ocáriz Borde H S, Bronstein M, Veličković P and Liò P 2022 Sheaf neural networks with connection Laplacians *Topological, Algebraic and Geometric Learning Workshops 2022* (PMLR) pp 28–36
[199] Krzakala F, Moore C, Mossel E, Neeman J, Sly A, Zdeborová L and Zhang P 2013 Spectral redemption in clustering sparse networks *Proc. Natl Acad. Sci.* **110** 20935–40
[200] Torres L, Suárez-Serrato P and Eliassi-Rad T 2019 Non-backtracking cycles: length spectrum theory and graph mining *Appl. Netw. Sci.* **4** 1–35
[201] Gu W, Tandon A, Ahn Y Y and Radicchi F 2021 Principled approach to the selection of the embedding dimension of networks *Nat. Commun.* **12** 3772
[202] Zhang Y J, Yang K C and Radicchi F 2021 Systematic comparison of graph embedding methods in practical tasks *Phys. Rev.* E **104** 044315
[203] Tandon A, Albeshri A, Thayananthan V, Alhalabi W, Radicchi F and Fortunato S 2021 Community detection in networks using graph embeddings *Phys. Rev.* E **103** 022316
[204] Srinivasan B and Ribeiro B 2019 arXiv:1910.00452
[205] Macqueen J 1967 Some methods for classification and analysis of multivariate observations *5th Berkeley Symp. on Mathematical Statistics and Probability* pp 281–97
[206] Kojaku S, Radicchi F, Ahn Y Y and Fortunato S 2023 Network community detection via neural embeddings (arXiv:2306.13400)
[207] Chami I, Gu A, Nguyen D P and Re C 2021 HoroPCA: hyperbolic dimensionality reduction via horospherical projections *Proc. 38th Int. Conf. on Machine Learning* (*Proc. Machine Learning Research* vol 139) ed M Meila and T Zhang (PMLR) pp 1419–29 (available at: https://proceedings.mlr.press/v139/chami21a.html)
[208] Rissaki A, Scarone B, Liu D, Pandey A, Klein B, Eliassi-Rad T and Borkin M A 2022 BiaScope: visual unfairness diagnosis for graph embeddings *2022 IEEE Visualization in Data Science (VDS)* pp 27–36
[209] MacArthur B D, Sánchez-García R J and Anderson J W 2008 Symmetry in complex networks *Discrete Appl. Math.* **156** 3525–31
[210] MacArthur B D and Sánchez-García R J 2009 Spectral characteristics of network redundancy *Phys. Rev.* E **80** 026117
[211] Sánchez-García R J 2020 Exploiting symmetry in network analysis *Commun. Phys.* **3** 87
[212] Wang J, Huang Y, Wu F X and Pan Y 2012 Symmetry compression method for discovering network motifs *IEEE/ACM Trans. Comput. Biol. Bioinform.* **9** 1776–89
[213] Dai H, Dai B and Song L 2016 Discriminative embeddings of latent variable models for structured data *Proc. 33rd Int. Conf. on Int. Conf. on Machine Learning (ICML '16)* vol 48 (JMLR.org) pp 2702–11
[214] Duvenaud D K, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A and Adams R P 2015 Convolutional networks on graphs for learning molecular fingerprints *Advances in Neural Information Processing Systems* vol 28, ed C Cortes, N Lawrence, D Lee, M Sugiyama and R Garnett (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-paper.pdf)
[215] Kearnes S, McCloskey K, Berndl M, Pande V and Riley P 2016 Molecular graph convolutions: moving beyond fingerprints *J. Comput.-Aided Mol. Des.* **30** 595–608
[216] Bruna J, Zaremba W, Szlam A D and LeCun Y 2014 *CoRR* (arXiv:1312.6203)
[217] Feng J, Huang M, Yang Y and Zhu X 2016 GAKE: graph aware knowledge embedding *Proc. COLING 2016, the 26th Int. Conf. on Computational Linguistics: Technical Papers* (The COLING 2016 Organizing Committee) pp 641–51 (available at: https://aclanthology.org/C16-1062)
[218] Fan J, Cannistra A, Fried I, Lim T, Schaffner T, Crovella M, Hescott B and Leiserson M D 2018 *bioRxiv Preprint* (22 March 2023)
[219] Heimann M, Shen H, Safavi T and Koutra D 2018 REGAL: representation learning-based graph alignment *Proc. 27th ACM Int. Conf. on Information and Knowledge Management (CIKM '18)* (Association for Computing Machinery) pp 117–26
[220] Fahimipour A K and Gross T 2020 Mapping the bacterial metabolic niche space *Nat. Commun.* **11** 4887
[221] Diaz-Papkovich A, Anderson-Trocmé L, Ben-Eghan C and Gravel S 2019 UMAP reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts *PLoS Genet.* **15** e1008432
[222] Han P, Yang P, Zhao P, Shang S, Liu Y, Zhou J, Gao X and Kalnis P 2019 GCN-MF: disease-gene association identification by graph convolutional networks and matrix factorization *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '19)* (Association for Computing Machinery) pp 705–13
[223] Rotmensch M, Halpern Y, Tlimat A, Horng S and Sontag D 2017 Learning a Health Knowledge Graph from Electronic Medical Records *Sci. Rep.* **7** 5994
[224] Wu T, Wang Y, Wang Y, Zhao E, Yuan Y and Yang Z 2019 arXiv:1910.02574
[225] Graham R L and Winkler P M 1985 On isometric embeddings of graphs *Trans. Am. Math. Soc.* **288** 527–36
[226] Linial N, London E and Rabinovich Y 1995 The geometry of graphs and some of its algorithmic applications *Combinatorica* **15** 215–45
[227] Linial N 2003 arXiv:math/0304466
[228] Indyk P, Matoušek J and Sidiropoulos A 2017 Low-distortion embeddings of finite metric spaces *Handbook of Discrete and Computational Geometry* (Chapman and Hall/CRC) pp 211–31
[229] Agrawal A, Ali A and Boyd S 2021 Minimum-Distortion Embedding *Found. Trends Mach. Learn.* **14** 211–378