Graph learning methods to analyze and support industrial resilience

Kévin Cortial, Adélaïde Albouy-Kissi, Frédéric Chausse

Graph learning methods to analyze and support industrial resilience

Kévin CORTIAL (kevin.cortial@doctorant.uca.fr), Adélaïde ALBOUY-KISSI (adelaide.kissi@uca.fr)
Frédéric CHAUSSE (frederic.chausse@uca.fr)

Institut Pascal, Université Clermont Auvergne, CNRS, Clermont Auvergne INP, 63000 Clermont-Ferrand, France

Abstract

Graphs are increasingly used to describe interactions between entities. They are based on simple formalism that nevertheless allows modelling of complex systems such as industrial ecosystems. Thus, a knowledge graph can be built from traditional economic variables, but also from new alternatives variables from open-source's data and big data. In this article, we review some graph learning methods and discusses latest advances in this field. Machine and deep graph learning method learn embeddings for nodes/edges in a graph to perform many tasks, such as link prediction, clustering, and nodes classification. Originality of this talk is the application on graph learning's methods to analyze and support industrial resilience. Indeed, by learning knowledge graphs which represents an industrial ecosystem, we could help for a more resilient and ecological production. In this survey, we discuss on many advantages of graph learning models and their limits. This review shows latest advances and leads to a discussion of prospects for future research.

Keywords: Graph learning, Knowledge graph, Industrial resilience

JEL codes: C45-Neural Networks and Related Topics; C15-Statistical Simulation Methods

1 - Introduction

Graphs, also called networks, can model various real-world relationships between entities. Many disciplines use graphs to formulate relationships between entities. For example, social networks, chemical and biological networks, computer networks, are examples that use graphs to model connections between certain pairs of objects. Technically, a graph is often defined by a set of nodes and a set of edges between pairs of nodes. A directed graph is a graph in which edges have an orientation. The degree of a node is the number of entering or leaving edges in each node. A directed graph is a graph in which the edges have an orientation.

Generally, graph encoding involves creation of an adjacency matrix whose values a_{ij} are the weights of the edges linking node i to node j. If the graph is non-oriented, adjacency matrix is symmetric. However, this structuring does not allow to store multiple data in nodes and edges. To build a knowledge graph of industrial ecosystems, we will use database graphs. In addition to storing the data freely, we will be able to query data without the need for multiple expensive joins that would be required in traditional relational databases. Thus, allow for very efficient updates, even for a very large data set. Triplestores store only one data's type. Triplet, consisting of start node, edge node and finish node. We do not need to create tables as in a relational database. Moreover, a triplestore is optimized for storing many triples and to retrieve these triples using the SPARQL query language.

Machine learning is a field of study in artificial intelligence that relies on mathematical and statistical approaches to give computers ability to "learn" from data. However, machine learning usually works with structured data (time series, images with pixels, texts, etc.). Classical machine learning methods will not work with graphs, as they can have arbitrary size, multimodal features, and complex topology.

Graphs can be embedded in a low-dimensional vector space where structural's information, attributes of nodes and edges can be preserved. These vectors are input data for graph learning methods. Vectorization methods can be various, depending on raw data typology of the graph (images, texts, etc.). These methods will be detailed in next chapters. According to nodes attributes, edges and subgraphs, graph learning tasks can be divided into three categories, respectively based on three ones.

(i) Classification of nodes will allow identification of key players in economic network. In fact, during node classification task, algorithm must determine labelling of a node by looking his embedding.

(ii) Grouping and detection of communities with similar properties. Technically, process of grouping together nodes that appear based on some similarity measure to be closer to each other. Identifying characteristics of each group is needed, this is the cluster analysis stage. For industrial resilience graph, clustering could identify stakeholder groups which may be economic and industrial sectors.

(iii) Prediction of links that do not yet exist between two nodes. Predicting a connection between two entities could be seen as a recommendation system. In a context of industrial resilience graph, link prediction can recommend partnerships between local industrial actors.

2 - Related Work

There are several studies on deep and machine learning methods on graphs. In contrast to these studies, our objective is to provide a comparison of graph learning methods, which can be applicable in economic context: in particular, knowledge extraction from graphs that model the industrial ecosystem. For our knowledge, this application is not covered by other literature's reviews.

Xia and al. compared graph learning methods with other methods which are more statistical models, such as graph signal processing, random walk, and matrix factorization. Nickel and al. also discuss these statistical models, but their use is focused on knowledge graphs and their automatic construction. Zhou and al. only offer a comparison of deep learning models for GNN graphs and their variants by mentioning several fields of application without mentioning economics. Kosasih and Brintrup present GNN, however, in the case of supply chains graph the purpose is to detect unknown potential links. In addition, they use gradient embedding to improve explainability by highlighting the input features which influence the decisions of their deep learning model. Cai and al. summarized graph embedding methods, this work allows us to preprocess our graphs for the models we will present in the following chapters.

In summary, no existing studies provide a comprehensive overview graph learning's application to the learning of economic graphs modelling an industrial ecosystem. Originality of our paper is the application of methods in a field where is a lack of literature review on the application of graph learning in this economic context. Our paper will present the most recent machine learning techniques for graph data that could be applied to this context.

3 - Methods

In this section, several methods which allow to analyze knowledge graphs of industrial ecosystems modeling will be introduced. These statistical or machine learning tools extract knowledge from graph topologies. In a first step, the most classical methods of statistical analysis of graphs will be presented. Then, this chapter will ended by presenting deep learning methods that have been recognized as the most powerful techniques for graph's analysis with complex data in nodes and edges (Zhang, Cui, and Zhu 2020).

a) Modularity

Modularity is a measure of the quality partitioning of the nodes for a graph. This principle implies many intra-community edges and a few inter-community edges. In other words, there are more connections between nodes of the same community than between nodes of different communities (Newman 2006). The modularity score compares, for a group of nodes, the number of actual edges to the number of expected edges (in an equivalent graph where the edges are randomly placed like the Erdös-Rényi model). If there are more actual edges than expected, then one could have a community. This gives a division score of a graph defined by the following formula:

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - P_{ij}\right)\delta\left(C_i, C_j\right)$$

Where A_{ij}, is the value ij of the adjacency matrix. m is the number of edges. $2m$ is the total number of half-edges. δ is the Kronecker delta: $\delta\left(C_i, C_j\right) = 1$ if i and j are in the same community C, i.e., $C_i = C_j$. Otherwise $\delta\left(C_i, C_j\right) = 0$. P_{ij} is the probability of the number of connections between i and j, under a null Erdös-Rényi model, which produces a homogeneous graph.

Thus, maximizing the modularity Q means looking for groups of nodes with an abnormally high number of connections between them.

Modularity is the fundamental basis of community extraction applicable to graphs. the Louvain method (Blondel and al. 2008) is particularly well suited to handle large volumes of data.

The first phase of the Louvain method consists in finding small communities by local optimization of modularity on each node. In a second step, the nodes of the same community are grouped into a single node. First phase is repeated on the newly obtained network. Iterations are repeated until no modularity's increase is possible.

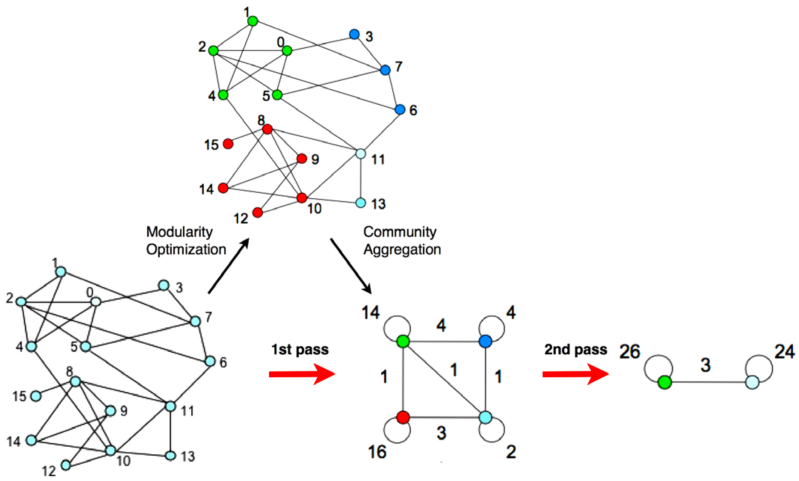

Figure 1 : Visualization of the steps of the Louvain method (Blondel and al. 2008)

Creation of these communities would make possible to identify industrial sectors. We could use knowledge graph of productive skills of Pachot and al. to group local productive partners and secure their supplies, thus promote distributed manufacturing.

b) Stochastic block model (SBM)

SBMs are generative models for random graphs. These models produce graphs containing communities, with subsets of nodes characterized by their connections to each other with edge densities. To generate a random graph, the SBM needs three parameters:

- n the number of nodes.
- A partition of the set of nodes into communities according to a multinomial distribution. $Z_i \sim \mathcal{M}\left(1, \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)\right)$ where K is the number of communities and α is the probability belongs to the community.
- The probability matrix μ_{kl} where diagonal elements correspond to the probability that nodes connect internally, within the community. Other elements correspond to the probability of connection between communities (Faskowitz and al. 2018).

Graph generation with an SBM model is simple. However, estimating all parameters of the SBM model such as μ_{kl} and α is a challenge. Indeed, standard optimization algorithms, such as the expectation maximization algorithm (to obtain maximum likelihood parameters), cannot be derived. To solve this problem, variational and stochastic approximations exist, such as variational Bayesian methods (Latouche, Birmele, and Ambroise 2010)

SBM methods and modularity score can create communities, but their results may be different. Indeed, modularity can illustrate communities whose nodes are strongly connected internally. On the other hand, there are few connections between nodes of different communities. The SBM identifies different clusters providing new information as hubs, which have many degrees per node. These nodes are loosely connected to each other but regularly connected to others forming star graphs. The strength of the SBM model is to provide probabilities of connections μ_{kl} between different nodes according to their cluster assignment. Probabilities of connections between groups of nodes can be considered in multiobjective recommendation systems such as those of Pachot and al.

c) Graph Neural Networks (GNN)

Deep learning tools usually work on structured data. However, graphs could have arbitrary size, multimodal features, and complex topology. Thus, Graph Neural Networks (GNN) are deep learning algorithms which induce features from graph data. These inferences may be used directly for predictions, classifications, or other approaches. (Wu and al. 2021)

To exploit graph's data, GNNs have made the hypothesis that many pieces of node's information reside in its neighborhoods. To store this data, we use "Node Embedding" which gathers the neighborhoods information with multilayer neural networks (MLPs).

Thus, a GNN uses an MLP on each node's neighbor: we call this a GNN layer. In the Figure 2, there are therefore two layers: a first layer for A neighbor's, which in turn is derived from a second layer concerning the neighbors of the neighbors. For the node information set (its embedding) we apply the MLP (grey/black blocks in the Figure 2) and

get a new learned node vector. The same will be done for the edges when they contain information (several edge types, values, weights, etc.).

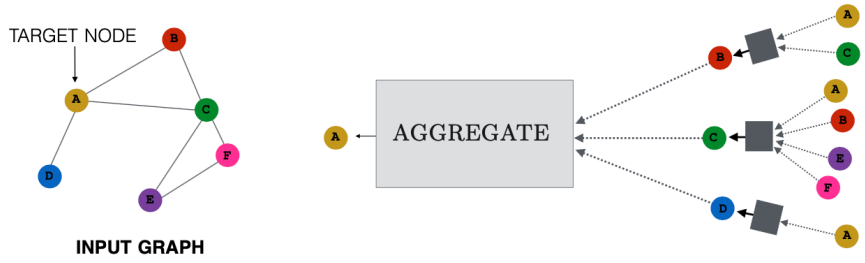

Figure 2 : GNN encoding with neighborhood aggregation methods (Hamilton, Ying, and Leskovec 2018)

The following formula indicates how the input information (from the target node h_i^t and its neighbors h_j^t) will be aggregate by neural networks and produce a new representation h_i^{t+1}. This is the mechanism found within the gray and black blocks in the Figure 2.

$$h_i^{t+1} = \sigma\left(h_i^t W + \sum_{j \in N(i)} \frac{1}{c_{ij}} h_j^t U\right)$$

- h_i^t is the initial vector containing the information of the node h_i.
- h_i^{t+1} is the updated vector containing the information of the neighbors but also graph's topology. This gives more information and a better representation of the h_i node within the graph.
- W is the matrix of weights from a neural network (MLP) that identifies most important elements to keep within the initial vector h_i^t.
- U is also a weight matrix from a neural network but will process vectors from neighboring nodes h_j^t.
- $\sum_{j \in N(i)} \frac{1}{c_{ij}}$ is an aggregation function, this sum normalizes the neighboring representations $N(i)$. This requires a function that is invariant by permutation of its variables because we must be insensitive to the order of the neighbors to have the same result, whatever the order of the neighbors.
- σ is an activation function.

Applying this function iteratively provides better representations of the nodes within their environment, i.e., in the graph. Thanks to these new embeddings, link prediction between nodes, clustering of nodes or subgraphs could be established. This method was popularized by the work of Kipf and Welling 2017.

d) Graph Attention Networks (GAT)

Like GNN, Graph Attention Networks (GAT) use data contained in these neighbors to create embeddings for each node. The difference in GATs is the use of the attention mechanism (Vaswani and al. 2017) to select the importance to be given to each neighbor.

$$\text{Attention}\ (Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Computation of attention coefficient is done by the above formula, where the three vectors are multiplied by the embedding of the input sequence. Here, Q is the search query, K would be the features (texts, images, etc.) associated with the best fitting results V. The attention weights are divided by the square root of the dimension of key vectors $\sqrt{d_k}$ to stabilize gradients during training, and then passed through a SoftMax function that normalizes weights to select the V values to retain. These steps are the heads of attention which have queries and keys as input. In practice, the attention function is computed on a set of queries simultaneously, grouped in a Q matrix. Keys and values are also grouped in matrices K and V (Vaswani and al. 2017).

Advantages of multi-head attention are that it provides a diversification of the results during several self-attention/attention heads (h times), as the vectors Q, K and V are randomly initialized. The chaining of these attention's heads makes learning more robust and parallelizable.

GAT aggregates information present in the neighborhood of a node by a weighted sum as an attention mechanism. First, GAT calculates an attention coefficient for each neighbor of the node in question (node features). Then, GAT must integrate edge data according to their number (centrality). Finally, if the nodes position in the graph is important,

GAT will consider this position data (spatial encoding). Indeed, for our knowledge graph which models industrial ecosystems, the geographical location of the companies (nodes) must be considered. The adaptation of the GAT architecture, presented above, requires the addition of all graph's information at self-attention layer (Ying and al. 2021).

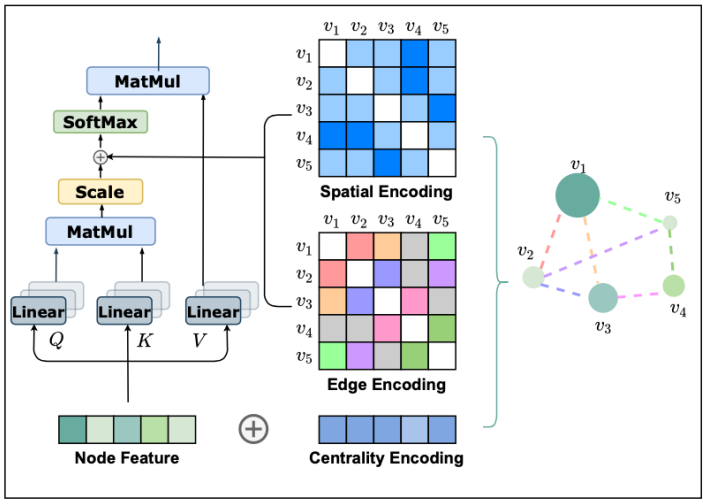

Figure 3 : Illustration of the encoding of graph information in the attention mechanism (Ying and al. 2021)

4 - Experimentations and results

There are several dataset and benchmarks used to evaluate performance of graph learning approaches for various tasks, such as link prediction and node classification. For subgraph clustering or community detection, evaluation's metrics depend on results interpretation according to study contexts. Thus, in this section, we compare the results of models for link prediction and node classification.

For link prediction, the FB15k dataset contains knowledge base relation triples and textual mentions of Freebase entity pairs. It has a total of 592,213 triplets with 14,951 entities and 1,345 relationships. FB15K-237 is a variant of the original dataset where inverse relations are removed, since it was found that many test triplets could be obtained by inverting triplets in the training set (Bordes and al. 2013). On this dataset, Wang and al. proposed a variant of GAT. Their work on a GAT based on attenuated attention shows best results in comparison with other state-of-the-art methods on FB15k-237 for link prediction.

WN18RR is a link prediction dataset contains 93,003 triples with 40,943 entities and 11 relation types. WN18RR is a link prediction dataset contains 93,003 triples with 40,943 entities and 11 relation types. It is designed to produce an intuitively usable dictionary and thesaurus. Its entities correspond to word senses, and relationships define lexical relations between them (Bordes and al. 2013). On this dataset, Li, Yi, and He proposed a variant of BERT model (Devlin and al. 2019) for link prediction. This BERT model uses attention mechanism on the textual data of the nodes and edges of this dataset to predict links. The work of Li, Yi, and He had the best state-of-the-art results on the WN18RR data. They improved by 5% over previous state-of-the-art result on the WN18RR dataset.

For node classification, the Cora dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links (Mccallum 2000). The best model for classification into the seven Cora classes is the model of Izadi and al. They used GNN algorithms using natural gradient information in the optimization process.

5 - Conclusion and perspectives

This paper provides an overview of graph learning and compares most recent graph learning methods. We have presented existing graph learning methods including modularity, SBM, GNN and GAT. Originality of our work is the application of graph learning methods to support industrial resilience. Indeed, by learning knowledge graphs that represent an industrial ecosystem, we could contribute to a more resilient and ecological production. Experiments and results from different benchmarks show that deep learning models (GNN and GAT) have better performances on clustering, classification, and link prediction. These results confirm our intention to use graph learning methods for our future research on knowledge graphs to model industrial ecosystems. We will work on the graph constructed with potential productive links between firms of Pachot and al. Graph learning is currently a very rapidly evolving field. We hope that this study will help researchers and practitioners in their research and development's works on graph learning and related fields.

Bibliography

Blondel, Vincent D., Jean-Loup Guillaume, Renaud Lambiotte, et Etienne Lefebvre. 2008. « Fast Unfolding of Communities in Large Networks ». Journal of Statistical Mechanics: Theory and Experiment 2008(10): P10008.

Bordes, Antoine et al. « Translating Embeddings for Modeling Multi-Relational Data ». : 9.

Cai, Hongyun, Vincent W. Zheng, et Kevin Chen-Chuan Chang. 2018. « A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications ». arXiv:1709.07604 [cs]. http://arxiv.org/abs/1709.07604.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, et Kristina Toutanova. 2019. « BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding ». arXiv:1810.04805 [cs]. http://arxiv.org/abs/1810.04805.

Erdös, P., et A. renyi. 1961. « On the Strength of Connectedness of a Random Graph ».

Faskowitz, Joshua, Xiaoran Yan, Xi-Nian Zuo, et Olaf Sporns. 2018. « Weighted Stochastic Block Models of the Human Connectome across the Life Span ». Scientific Reports 8(1): 12997.

Hamilton, William L., Rex Ying, et Jure Leskovec. 2018. « Representation Learning on Graphs: Methods and Applications ». arXiv:1709.05584 [cs]. http://arxiv.org/abs/1709.05584.

Izadi, Mohammad Rasool, Yihao Fang, Robert Stevenson, et Lizhen Lin. 2020. « Optimization of Graph Neural Networks with Natural Gradient Descent ». arXiv:2008.09624 [cs, stat]. http://arxiv.org/abs/2008.09624.

Kipf, Thomas N., et Max Welling. 2017. « Semi-Supervised Classification with Graph Convolutional Networks ». arXiv:1609.02907 [cs, stat]. http://arxiv.org/abs/1609.02907.

Kosasih, Edward Elson, et Alexandra Brintrup. 2021. « A Machine Learning Approach for Predicting Hidden Links in Supply Chain with Graph Neural Networks ». International Journal of Production Research: 1-14.

Latouche, Pierre, Etienne Birmele, et Christophe Ambroise. 2010. « Variational Bayesian Inference and Complexity Control for Stochastic Block Models ». arXiv:0912.2873 [stat]. http://arxiv.org/abs/0912.2873.

Li, Da, Ming Yi, et Yukai He. 2022. « LP-BERT: Multi-Task Pre-Training Knowledge Graph BERT for Link Prediction ». arXiv:2201.04843 [cs]. http://arxiv.org/abs/2201.04843.

Mccallum, Andrew Kachites. « Automating the Construction of Internet Portals with Machine Learning ». : 37.

Newman, M. E. J. 2006. « Modularity and Community Structure in Networks ». Proceedings of the National Academy of Sciences 103(23): 8577-82.

Nickel, Maximilian, Kevin Murphy, Volker Tresp, et Evgeniy Gabrilovich. 2016. « A Review of Relational Machine Learning for Knowledge Graphs ». Proceedings of the IEEE 104(1): 11-33.

Pachot, Arnault, Adélaïde Albouy-Kissi, Benjamin Albouy-Kissi, et Frédéric Chausse. 2021a. « Multiobjective Recommendation for Sustainable Production Systems ». http://rgdoi.net/10.13140/RG.2.2.23257.44649/1.

Pachot, Arnault, Adelaide Albouy-Kissi, Benjamin Albouy-Kissi, et Frederic Chausse. 2021b. « Production2Vec: A Hybrid Recommender System Combining Semantic and Product Complexity Approach to Improve Industrial Resiliency ». In 2021 2nd International Conference on Artificial Intelligence and Information Systems, Chongqing China: ACM, 1-6. https://dl.acm.org/doi/10.1145/3469213.3469218.

Pachot, Arnault, Adélaïde Albouy-Kissi, Benjamin Albouy-Kissi, et Frédéric Chausse. « Decision Support System for Distributed Manufacturing Based on Input–Output Analysis and Economic Complexity ». : 9.

Vaswani, Ashish et al. 2017. « Attention Is All You Need ». arXiv:1706.03762 [cs]. http://arxiv.org/abs/1706.03762.

Wang, Rui et al. 2020. « Knowledge Graph Embedding via Graph Attenuated Attention Networks ». IEEE Access 8: 5212-24.

Wu, Zonghan et al. 2021. « A Comprehensive Survey on Graph Neural Networks ». IEEE Transactions on Neural Networks and Learning Systems 32(1): 4-24.

Xia, Feng et al. 2021. « Graph Learning: A Survey ». IEEE Transactions on Artificial Intelligence 2(2): 109-27.

Ying, Chengxuan et al. 2021. « Do Transformers Really Perform Bad for Graph Representation? » arXiv:2106.05234 [cs]. http://arxiv.org/abs/2106.05234.

Zhang, Ziwei, Peng Cui, et Wenwu Zhu. 2020. « Deep Learning on Graphs: A Survey ». arXiv:1812.04202 [cs, stat]. http://arxiv.org/abs/1812.04202.

Zhou, Jie et al. 2020. « Graph Neural Networks: A Review of Methods and Applications ». AI Open 1: 57-81.