RECOVERING BARABÁSI-ALBERT PARAMETERS OF GRAPHS THROUGH DISENTANGLEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Classical graph modeling approaches such as *Erdős-Rényi random graphs (ER graphs)* or *Barabási Albert graphs (BA graphs)*, here referred to as stylized models, aim to reproduce properties of real-world graphs in an interpretable way. While useful, graph generation with stylized models requires domain knowledge and iterative trial and error simulation. Stoehr et al. (2019) suggest addressing these issues by learning the generation process from graph data, using a disentanglement-focused deep autoencoding framework, more specifically, a β-Variational Autoencoder (β-VAE). While they successfully recover the generative parameters of ER graphs through the model's latent variables, their model performs badly on sequentially generated graphs such as BA graphs, due to their oversimplified decoder. We focus on recovering the generative parameters of BA graphs by replacing the decoder in Stoehr et al. (2019)'s β-VAE with a sequential one. We first learn the generative BA parameters in a supervised fashion using a Graph Neural Network (GNN) and a Random Forest Regressor, by minimizing the squared loss between the true generative parameters and the latent variables. Next, we train a β-VAE model, combining the GNN encoder from the first step with an LSTM-based decoder with a customized loss.

1 INTRODUCTION

Knowing the generative procedure of a graph and its parameters allows predicting, controlling and understanding its driving principles (Brugere et al., 2018; Barabási & Pósfai, 2016; Newman, 2010). One way to uncover the underlying generative parameters of such a procedure is through disentanglement – an approach which has recently gathered a large interest in the research community. The aim is to find a representation of the data where the variables are disentangled in the sense that each variable corresponds to at most one generative parameter. Disentangled representations have been successfully found for image data (Burgess et al., 2019; van Steenkiste et al., 2019; Leeb et al., 2020; Besserve et al., 2019). However, disentanglement of graph data is largely unexplored, with a few exceptions (Ma et al., 2019; Guo et al., 2020; Stoehr et al., 2019).

Stoehr et al. (2019) propose a disentanglement framework with which they recover the generative parameters of graphs generated by stylized graph models. Stylized graph generation models are graph generation algorithms that typically follow a fixed procedure in order to capture certain characteristics of real-world graphs. While useful, these models require pre-specified parameters. Furthermore, finding the appropriate values of these parameters typically demands either domain knowledge or time-consuming simulations. With a disentanglement framework however, one can find these generative parameters unsupervised from a given dataset.

A commonly used stylized model is the Erdős-Rényi (ER) random graph generator, which defines the number of nodes n and the uniform linking probability p between any two nodes in the graph. In this generation process the individual edges are sampled independently of each other and the graph can thus be generated accurately "one-shot", meaning that the procedure can be non-iterative. However, most real-world graph datasets are generated through a sequential process. Therefore, the ER model does not capture many of the properties typically observed in real-world graphs, such as the tendency of nodes to link to more connected nodes, a phenomenon known as preferential attachment. Graphs that exhibit this property are called scale-free, and they evolve

dynamically, continuously growing with the addition of new nodes. The Barabási-Albert (BA) Preferential Attachment model (Barabási & Albert, 1999) is a stylized model that was devised in order to more accurately model such graphs, in which edges are generated sequentially. As the properties of scale-free graphs are inherently the result of an iterative process, we are not aware of a method for generating them "one-shot". The β-Variational Autoencoder (β-VAE) architecture (Higgins et al., 2017) suggested by Stoehr et al. (2019) is capable of both generating graphs and recovering compact, disentangled representations of their generative parameters. While their approach performs well when applied to ER graphs, their decoder generates the graphs one-shot which hinders their performance on the scale-free BA graphs.

Focusing on data generated from the non-linear extension of BA (Onody & de Castro, 2004), the main contribution of this work is creating a graph VAE with a sequential decoder, for finding compact graph representations corresponding to the generative parameters of BA graphs. In this paper we first show that it is possible to recover the generative parameters, by learning them in a supervised fashion using a Graph Neural Network (GNN) and a Random Forest (RF) model, using the mean squared error (MSE) loss between the true generative parameters and the predictions. Next, we train a β-VAE model, combining the GNN encoder from the previous step with an LSTM-based decoder with a customized loss, in order to obtain a disentangled representation of the generative parameters. Creating a sequential graph decoder network is especially challenging, as it should mimic a sequential random generation process, with random sampling at each step. Such a process is not naturally differentiable with respect to the latent variables, and typically requires tricks to enable backpropagation. As such, we adapt the LSTM model to the graph setting for this task.

2 Models and Methods

We use a synthetic dataset that is generated with the BA model. The modeling approach is split into two steps: parameter prediction via supervised training and disentanglement via unsupervised training. In the first part we predict the generative factors of the BA graphs using both a Random Forest and a GNN. In the second part, we implement a β-VAE with a sequential LSTM decoder network, to find disentangled latent representations of the generative parameters of the BA graphs.

BA Graph Generation The non-linear extension of the BA model is parameterized by the number of nodes n, an edge generation integer m and the exponential of the node degree α. In the standard BA model (Barabási & Albert, 1999), we set $\alpha = 1$ and get a linear dependency on the node degrees. In real-world scale-free graphs, preferential attachment can also be non-linearly dependent on the node degrees and the extended BA model (Onody & de Castro, 2004) accounts for these scenarios through the parameter α. The description of the generative process can be found in Appendix A.1 and illustrations of BA graphs with different values of α are shown in Figure 2.

2.1 Predicting Generative Parameters

In this stage, we learn the generative parameters in a supervised manner. We train a model to output predictions for the generative parameters, and take the MSE between the model's output and the true generative parameters to be our training loss. As a first step, we want to verify that the generative parameters are learnable from the data. To do so, we extract feature vectors from the input graphs, and then train a Random Forest (RF) Regressor to predict the parameters from these vectors. However, a non-neural model requires manual feature extraction, whereas a GNN can automatically extract features from the graphs.

As such, we proceed to implement a GNN to predict the generative parameters, being adapted from Stoehr et al. (2019). The outputs of the GNN can be interpreted as the latent variables in an autoencoder model, with a BA generator as the decoder. The BA generator is non-differentiable and lacks learnable parameters however, which prevents unsupervised training via minimizing the reconstruction loss. A high-level architecture can be seen in the top part of Figure 1.

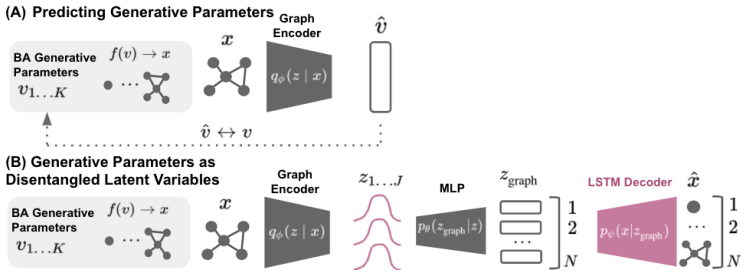

Figure 1: (A) Predicting generative parameters in a supervised setting. (B) Disentangling latent variables to recover generative parameters in an unsupervised setting.

2.2 Generative Parameters as Disentangled Latent Variables

To obtain a VAE, we keep the GNN encoder used in the prediction stage, and implement a differentiable decoder in the form of an LSTM. The latent variables produced by the encoder are passed through a deconvolutional layer to obtain N graph representations, N being the number of generations steps. These are then passed to the LSTM decoder. The model is trained with the standard β-VAE objective, composed of a scaled KL-divergence term enforcing the variational posterior to be similar to an isotropic Gaussian, and a reconstruction term comparing the input and output graphs. We further extend the reconstruction part to include a term that forces the intermediate outputs generated at each time step to be adjacency matrices. This is added in order to simulate the generation process of a sequential stylized model. A high-level architecture of this step can be seen in the bottom part of Figure 1.

Mutual Information Gap Among the many metrics for assessing the quality of disentangled latent representations[1], we choose MIG as it captures axis-alignment and is unbiased (Chen et al., 2018). The MIG of latent variables $z_{1,\ldots,J}$ with respect to generative factors $v_{1,\ldots,K}$ is defined as:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{H(v_k)}(I(z_{j^{(k)}};v_k) - \max_{j\neq j^{(k)}} I(z_j;v_k))$$

where I is the mutual information, H is the entropy and $j^{(k)} = \arg\max_j I(z_j;v_k)$. Intuitively, the MIG averages the difference between the largest mutual information and the second largest mutual information for each generative parameter with respect to the latent variables. It is then normalized by the entropy to be between 0 and 1, where a larger MIG value represents better disentanglement.

3 Results

We assess the performance of the supervised GNN model in predicting the generative parameters, for both linear and non-linear BA graphs. The training and test sets are of size $16,000$ and $4,000$ respectively. The number of nodes n is fixed to be 50, m is sampled as an integer uniformly at random between 1 and 49, and for the non-linear case α is sampled uniformly at random between $1/3$ and 3 in order to capture graphs of both sub-linear and super-linear degree distribution[2]. The results of the two learning tasks are shown in Table 1 together with the results from the RF validation model.

The GNN learns the parameters of the linear BA model with a satisfactory performance, but does not manage to learn α, while the RF model manages to learn α with a significantly lower MSE and a Pearson correlation coefficient of 0.89 between the true and predicted α, indicating that α is in fact learnable from the graph data. The predicted parameter values can be used to generate new graphs, using the BA graph generator. These can be compared to the original graphs for performance assessment, see Appendix A.2 for more details.

[1]See (Locatello et al., 2019) and (van Steenkiste et al., 2019) for a comparison of different disentanglement metrics in fairness and abstract visual reasoning respectively.

[2]Further implementation details can be found in the code, which is to be published with the camera-ready version.

Task & Data	MSE on test set		
	n	**m**	α
GNN & Linear BA	0.0005	0.128	-
GNN & $\alpha \sim \mathcal{U}(\frac{1}{3}, 3)$	0.005	0.257	0.590
Random Forest & $\alpha \sim \mathcal{U}(0, 3)$	0.000	1.823	0.194

Table 1: Model performance in supervised setting.

Model	Data	MIG
LSTM-VAE	Linear BA	0.26
LSTM-VAE	Linear BA (m/n)	0.29
LSTM-VAE	Non-linear BA	0.12

Table 2: MIG scores in unsupervised setting.

For the LSTM-based β-VAE, we compute the MIG score between the generative parameters and the latent variables. The model is trained on two BA datasets, a linear one with $\alpha = 1$, and a non-linear one with $\alpha \in (\frac{1}{3}, 3)$. Both datasets consist of 200 graphs, with node sizes varying from 3 to 12, and the model is trained for 200 epochs. The MIG scores are presented in Table 2. We see that when trained on linear BA data the model displays the best performance, as expected since this is an easier task. We further note that the parameters n and m in the BA model are not independent. Since independence is enforced by the KL-term when training a β-VAE we also examine the MIG with respect to the parameter $\frac{m}{n}$ instead of m. This parameter is independent from n, but together with n completely describes m. We observe in Table 2 that the MIG scores for $\frac{m}{n}$ is indeed higher.

4 Discussion

Encoder Architectures In the prediction setting, we achieve satisfactory results for learning the parameters with a simple Random Forest model that uses manually extracted features as input. We do not manage to learn α with the GNN, which also contributes to poor results in later stages, where we use this architecture as the encoder. The performance of the RF model attests that the graph data holds sufficient information about the α parameter for a machine learning model to learn it. We thus believe that other GNN architectures should be experimented with to obtain better performance on this task. In particular, the graph convolutional layer used in our GNN requires adjacency matrices as input and output and as such depends on a fixed number of nodes of the graph[3]. There exist various graph convolutional layers that are invariant to the number of nodes, outputting embeddings on node level instead of graph level (Bacciu et al., 2020), that could be used in future work.

Categorical Latent Variables We use Gaussian distributions as priors on the β-VAE's latent variables. However, both n and m are categorical variables, and α is bounded to a fixed-length interval, so Gaussians do not provide an optimal fit. In the linear BA case, we could sample from a discrete distribution, for instance using the Gumbel-Softmax trick (Jang et al., 2017). Implementing this in the non-linear case is not as straightforward, since α is not discrete.

Generation Model A major challenge is the backpropagation through the decoder model, which essentially requires a graph generation process that both sequentially generates a graph and is differentiable with respect to the latent variables. The LSTM model can be considered sequential as it takes a graph representation as input in every time step. However, it conditions on the full generation history in the memory cell, which is expensive and not necessary in order to simulate a sequential graph generating process. Other drawbacks of this model are long training times as well as it not being a natural fit for graph data, requiring tricks to output an adjacency matrix. We note the existence of sequential GNN architectures (Li et al., 2018; Liu et al., 2018). However, the former conditions on the full generation history and the latter is tailored specifically towards molecule generation.

5 Conclusion

We recovered the generative parameters of the Barabási-Albert model and its non-linear extension through disentanglement. To this end, we emphasized the differences between sequential and non-sequential graph generators, and the scarcity of interpretable neural graph generators adhering to a sequential generation process. We hope to point out a promising and important research direction with plenty of room for future improvement.

[3]For the experiments varying n the work-around used was to zero-pad the elements of adjacency matrices of some specified maximum size.

REFERENCES

Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, Sep 2020.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

Albert-László Barabási and Márton Pósfai. *Network Science*. Cambridge University Press, Cambridge, 2016.

Michel Besserve, Arash Mehrjou, Rémy Sun, and Bernhard Schölkopf. Counterfactuals uncover the modular structure of deep generative models, 2019.

Ivan Brugere, Brian Gallagher, and Tanya Y. Berger-Wolf. Network structure inference, a survey: Motivations, methods, and applications. *ACM Computing Survey*, 51(2), 2018.

Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation, 2019.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in Neural Information Processing Systems 31*, pp. 2610–2620, 2018.

Xiaojie Guo, Liang Zhao, Zhao Qin, Lingfei Wu, Amarda Shehu, and Yanfang Ye. Interpretable deep graph generation with node-edge co-disentanglement. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, Jul 2020.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 1, pp. 370–378, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017.

Felix Leeb, Yashas Annadani, Stefan Bauer, and Bernhard Schölkopf. Structural autoencoders improve representations for generation and transfer, 2020.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *Proceedings of the 35th International Conference on Machine Learning*, 35: 444–450, 2018.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, volume 31, pp. 491–499. Curran Associates, Inc., 2018.

Francesco Locatello, Gabriele Abbati, Tom Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the fairness of disentangled representations. *CoRR*, abs/1905.13662, 2019.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4212–4221. PMLR, 09–15 Jun 2019.

Mark Newman. *Networks: An Introduction*. Oxford University Press, New York, USA, 2010.

Roberto N. Onody and Paulo A. de Castro. Nonlinear barabási–albert network. *Physica A: Statistical Mechanics and its Applications*, 336(3):491 – 502, 2004.

Niklas Stoehr, Marc Brockschmidt, Jan Stuehmer, and Emine Yilmaz. Disentangling interpretable generative parameters of random and real-world graphs. *CoRR*, abs/1910.05639, 2019.

Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? *CoRR*, abs/1905.12506, 2019.

A APPENDIX

A.1 BARABÁSI-ALBERT GENERATION PROCEDURE

The non-linear extension of BA (Onody & de Castro, 2004) is parameterized by the number of nodes n, an edge generation integer m and the exponential of the node degree α. The generative process is defined as follows:

1. Initialize a graph with m nodes.
2. At each step, until graph has n nodes, add a new node and connect it to m nodes randomly sampled from the following categorical distribution:

$$p(k_i) = \frac{k_i^\alpha}{\sum_j k_j^\alpha} \tag{1}$$

   where k_i is the degree of node i, and $p(k_i)$ is the probability that the new node connects to node i.

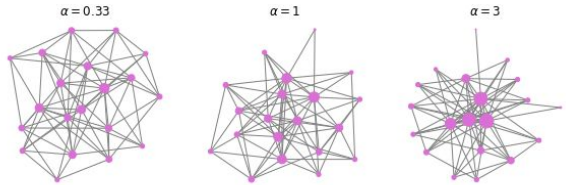

Figure 2: BA graphs with 20 nodes and $m = 5$ under different values of α. **Left**: in the sub-linear case, graphs are similar to random ER graphs, with no presence of hubs. **Center**: BA model. **Right**: Nodes with higher degree are more attractive, leading to fewer but larger hubs. Nodes with higher degrees are displayed as larger.

A.2 PERFORMANCE ASSESSMENT OF PARAMETER PREDICTION

For each test graph, we compute the standard deviation of degree and betweenness centrality for $1,000$ generated graphs and compare to those of the input graph. We consider the model's predictions of parameters reasonable if the input graph properties fall within the distribution given by the generated graphs. The plots for a sample of three test graphs from the linear & non-linear BA models are shown in Figure 3 & 4 respectively. As seen in these examples, the measures of graphs generated from the linear BA model more closely follow their induced graphs' distribution.

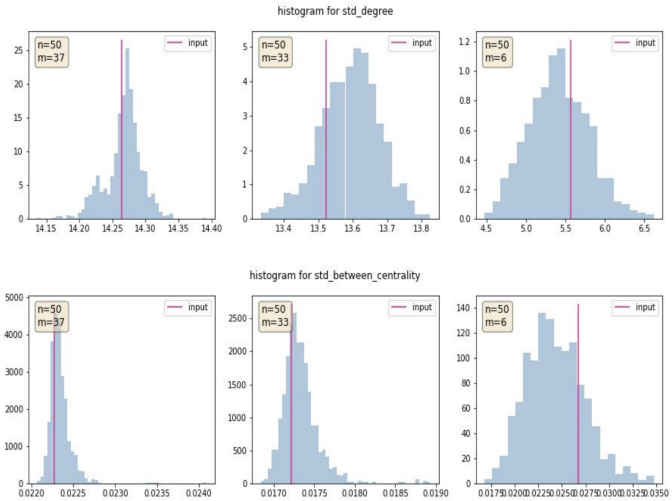

Figure 3: Graph measures of test graphs compared to $1,000$ generated graphs from linear BA model, with true generative parameters of the test graphs.

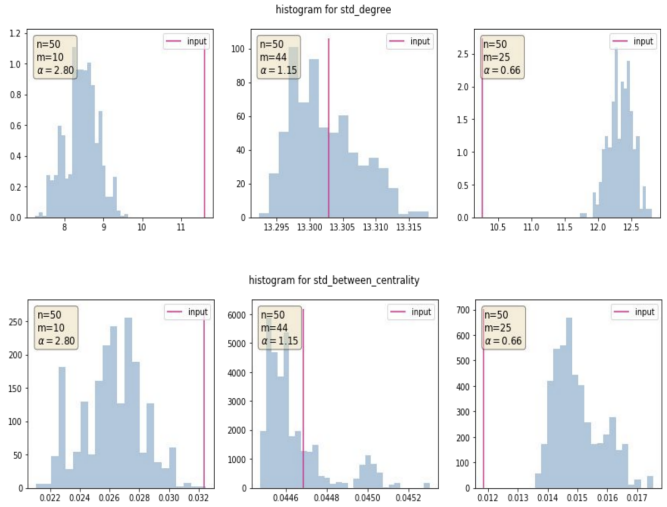

Figure 4: Graph measures of test graphs compared to $1,000$ generated graphs from non-linear BA extension model, with true generative parameters of the test graphs.