ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Improving Graph Variational Autoencoders with Multi-Hop Simple Convolutions

Erik Jhones F. do Nascimento[1], Amauri H. Souza[1] and Diego Mesquita[2]

1- Federal Institute of Ceará - Department of Computer Science
Fortaleza - Brazil

2- Aalto University - Department of Computer Science
Espoo - Finland

Abstract. Variational auto-encoding architectures represent one of the most popular approaches to graph generative modeling. These models comprise encoder and a decoder networks, which map back and forth between the input and latent spaces. Notably, most of the literature in variational autoencoders (VAEs) for graphs focuses on developing more efficient architectures at the expense of increased complexity. In this work, we pursue an orthogonal direction and leverage multi-hop linear graph convolutional layers to create efficient yet simple encoders, boosting the performance of graph autoencoders. Our results demonstrate that our approach outperforms popular graph VAE baselines in link prediction tasks.

1 Introduction

Variational autoencoders (VAEs) [1] have become a prominent framework for generative modeling in diverse areas, such as computer vision and natural language processing. In the context of graphs, auto-encoding architectures (including VAEs) have also been successfully applied to many challenging tasks, such as link prediction [2, 3, 4], node clustering [5, 6], matrix completion for recommendation systems [7], and generation of molecular graphs [8, 9].

In general, VAEs for graphs consist of an encoder which projects graphs into a low-dimensional latent space, and a decoder which maps the latents back to the original space. Although these VAEs come in many flavours [2, 3, 4, 9], most VAEs for graphs employ graph neural networks (GNNs) [10, 2] for the encoder.

Recently, many works [11, 12, 13, 14, 15] have shown that simple GNNs can achieve competitive performance on a variety of graph learning tasks. In the context of GraphVAEs, Salha *et al.* [16] empirically demonstrate that one-hop linear encoders perform similarly to architectures based on multi-layer GNNs [2].

In this paper, we improve GraphVAEs [2] encoders by leveraging multi-hop neighborhood information, which we extract using multiple simplified graph convolutions (SGCs) [11]. We propose two ways of combining such information: concatenating SGC outputs and summing them up. Because these GraphVAEs are mainly designed to handle reconstruction tasks (e.g., link prediction), we also extend our analysis to deterministic autoencoders (AEs).

Empirical results demonstrate that our methods outperform GraphVAEs for link prediction in popular benchmark datasets (Cora, Citeseer and Pubmed). Our methods also outperform the one-hop linear model in [16], which indicates that combining multi-hop information is beneficial for reconstruction tasks.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

2 Background

We represent a graph G with n nodes as a pair $(\boldsymbol{A}, \boldsymbol{X})$ where $\boldsymbol{A} \in \{0,1\}^{n \times n}$ is a symmetric adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is a node feature matrix. Additionally, we define the diagonal degree matrix $\boldsymbol{D}$ of G such that $D_{ii} \coloneqq \sum_j A_{ij}$. We denote the normalized graph Laplacian of G as $\boldsymbol{\Delta} \coloneqq \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$.

2.1 Graph variational autoencoders

GraphVAEs [2] comprise a decoder and an encoder network, with respective parameters $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$. The former parameterizes the likelihood function $p_{\boldsymbol{\Phi}}(G_i|\boldsymbol{Z}_i)$ of a graph G_i given its latent representation $\boldsymbol{Z}_i \in \mathbb{R}^{n_i \times d}$. The latter parameterizes the variational distribution $q_{\boldsymbol{\Theta}}(\boldsymbol{Z}_{1:S}) = \prod_{i=1}^{S} q_{\boldsymbol{\Theta}}\left(\boldsymbol{Z}_i|G_i\right)$ which approximates the Bayesian posterior:

$$p_{\boldsymbol{\Phi}}(\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_S|G_1, \ldots, G_S) \propto \prod_{i=1}^{S} p_{\boldsymbol{\Phi}}(G_i|\boldsymbol{Z}_i)p(\boldsymbol{Z}_i), \tag{1}$$

where $p(\boldsymbol{Z}_1), \ldots, p(\boldsymbol{Z}_S)$ are user-defined priors.

We learn $(\boldsymbol{\Theta}, \boldsymbol{\Phi})$ by minimizing the Kullback-Leibler divergence between the variational distribution and the posterior in Equation 1:

$$\begin{aligned}(\boldsymbol{\Theta}^\star, \boldsymbol{\Phi}^\star) &= \underset{\boldsymbol{\Theta},\boldsymbol{\Phi}}{\operatorname{argmin}}\, D_{\mathrm{KL}}\left[q_{\boldsymbol{\Theta}}\left(\boldsymbol{Z}_{1:S}|G_{1:S}\right) \| p_{\boldsymbol{\Phi}}(\boldsymbol{Z}_{1:S}|G_{1:S})\right] \\ &\triangleq \underset{\boldsymbol{\Theta},\boldsymbol{\Phi}}{\operatorname{argmin}}\, \mathbb{E}_{q_{\boldsymbol{\Theta}}}\left[\log \frac{q_{\boldsymbol{\Theta}}\left(\boldsymbol{Z}_{1:S}|G_{1:S}\right)}{p_{\boldsymbol{\Phi}}(\boldsymbol{Z}_{1:S}|G_{1:S})}\right]\end{aligned}, \tag{2}$$

which is seemingly intractable but can be efficiently solved by maximizing the Evidence Lower-BOund (ELBO):

$$\mathcal{L}\left(\boldsymbol{\Theta}, \boldsymbol{\Phi}\right) = \sum_{i=1}^{S} \mathbb{E}_{q_{\boldsymbol{\Theta}}(\boldsymbol{Z}_i|G_i)}\left[-\log p_{\boldsymbol{\Phi}}\left(G_i|\boldsymbol{Z}_i\right)\right] + D_{\mathrm{KL}}\left[q_{\boldsymbol{\Theta}}\left(\boldsymbol{Z}_i|G_i\right) \| p(\boldsymbol{Z}_i)\right] \tag{3}$$

Kipf and Welling [2] model $q_\Theta(\boldsymbol{Z}_i|G_i)$ as a Gaussian $\mathcal{N}(\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i))$ and use two two-layer Graph Convolutional Networks (GCNs) [17] for the encoder. One encoder network computes $\boldsymbol{\mu}_i \in \mathbb{R}^{n_i \times d}$ given G_i, while the other outputs $\boldsymbol{\sigma}_i \in \mathbb{R}^{n_i}$. The likelihood $p_{\boldsymbol{\Phi}}(G_i|\boldsymbol{Z}_i)$ is modeled as an element-wise Bernoulli distribution over the adjacency $\boldsymbol{A}_i$, with logits $\boldsymbol{Z}_i\boldsymbol{Z}_i^\intercal$.

2.2 Simple Graph Convolutions

We can use the eigendecomposition of the graph Laplacian matrix to define convolutions in graph domains [18]. Let $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ denote the eigenvectors and eigenvalues of the graph Laplacian $\boldsymbol{\Delta}$, respectively. The graph Fourier transform of a d-channel signal $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ on the vertices of the graph is given by $\hat{\boldsymbol{X}} =$

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

$\boldsymbol{U}^\intercal \boldsymbol{X}$, and its inverse is $\boldsymbol{X} = \boldsymbol{U}\hat{\boldsymbol{X}}$. Using these operations, we define the graph convolution between a signal $\boldsymbol{X}$ and a filter $\boldsymbol{g}$ as

$$\boldsymbol{g} \star \boldsymbol{X} = \boldsymbol{U}\left((\boldsymbol{U}^\intercal \boldsymbol{g}) \odot (\boldsymbol{U}^\intercal \boldsymbol{X})\right) = \boldsymbol{U}\hat{\boldsymbol{G}}\boldsymbol{U}^\intercal \boldsymbol{X} \tag{4}$$

where $\hat{\boldsymbol{G}} = \mathrm{diag}(\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_n)$ comprises the spectral filter coefficients $\hat{g}_i$.

In practice, the non-parametric approach in Equation 4 is undesirable, since it comprises a large number of parameters and does not produce localized filters [19]. A solution is to approximate the filter with polynomials of the Laplacian eigenvalues, leading to polynomials of the Laplacian since $\boldsymbol{U}\boldsymbol{\Lambda}^k\boldsymbol{U}^\intercal = \boldsymbol{\Delta}^k$.

Let $\boldsymbol{H}^{(0)} = \boldsymbol{X}$; at layer ℓ, a generic polynomial spectral GNN computes

$$\boldsymbol{H}^{(\ell)} = \mathrm{ReLU}\left(\sum_{k=0}^{K} \boldsymbol{\Delta}^k \boldsymbol{H}^{(\ell-1)} \boldsymbol{\Theta}_k^{(\ell)}\right), \tag{5}$$

where $\boldsymbol{\Theta}^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ are the coefficients of the spectral filters. The output $\boldsymbol{H}^{(L)} \in \mathbb{R}^{n \times d_L}$ after L layers comprises representations for each node in G.

Wu *et al.* [11] simplify the layer in Equation 5 in four ways: i) adopting $K = 1$; ii) setting $\boldsymbol{\Theta}_0^{(\ell)} = \boldsymbol{0}$ and $\boldsymbol{\Theta}_1^{(\ell)} = \boldsymbol{I}$; iii) removing the ReLU function; iv) replacing $\boldsymbol{\Delta}$ by the normalized adjacency matrix $\tilde{\boldsymbol{A}} = \boldsymbol{I} - \boldsymbol{\Delta}$ after adding self-loops to the graph. Stacking L layers and applying a linear transformation, we obtain the *simplified graph convolution* (SGC):

$$\boldsymbol{H} = \tilde{\boldsymbol{A}}^L \boldsymbol{X} \boldsymbol{\Theta}. \tag{6}$$

3 Multi-hop linear graph (variational) autoencoder

To improve the performance of GraphVAEs, we modify their encoders by i) replacing the two-layer GCNs with SGCs; and ii) using SGCs of different order to effectively extract multi-hop information. We refer to this method as *multi-hop linear graph variational autoencoder* (ML-GVAE). In addition, we propose combining the multiple SGCs in two-ways: summing and concatenating their outputs. The resulting linear encoders are:

$$\text{ML-GVAE (sum):} \qquad \boldsymbol{\mu}_i = \sum_{c=1}^{C} \tilde{\boldsymbol{A}}_i^c \boldsymbol{X}_i \boldsymbol{\Theta}_\mu^{(c)}, \quad \log \boldsymbol{\sigma}_i = \sum_{c=1}^{C} \tilde{\boldsymbol{A}}_i^c \boldsymbol{X}_i \boldsymbol{\theta}_\sigma^{(c)}; \tag{7}$$

$$\text{ML-GVAE (cat):} \qquad \boldsymbol{\mu}_i = \Big\|_{c=1}^{C} \tilde{\boldsymbol{A}}_i^c \boldsymbol{X}_i \boldsymbol{\Theta}_\mu^{(c)}, \quad \log \boldsymbol{\sigma}_i = \Big\|_{c=1}^{C} \tilde{\boldsymbol{A}}_i^c \boldsymbol{X}_i \boldsymbol{\theta}_\sigma^{(c)}; \tag{8}$$

where $\|$ denotes concatenation.

Despite the linear nature of our encoder, it is worth mentioning that optimizing ML-GVAEs is still a non-convex problem. The main issue here is arguably the relationship between the (Gaussian) variational distribution and $\boldsymbol{\sigma}_i$'s. To control the loss landscape, we also propose deterministic versions of ML-GVAE,

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

which does not depend on noise injection:

$$\text{ML-GAE (sum):} \qquad \boldsymbol{Z}_i = \sum_{c=1}^{C} \tilde{\boldsymbol{A}}_i^c \boldsymbol{X}_i \boldsymbol{\Theta}_\mu^{(c)}; \tag{9}$$

$$\text{ML-GAE (cat):} \qquad \boldsymbol{Z}_i = \Big\|_{c=1}^{C} \tilde{\boldsymbol{A}}_i^c \boldsymbol{X}_i \boldsymbol{\Theta}_\mu^{(c)}. \tag{10}$$

Another usual alternative to control the loss landascape is to replace $\boldsymbol{\sigma}_i$'s by a scalar hyper-parameter σ and use isotropic Gaussians for the variational distribution. In this case, we only need a single encoder network. Note that, unlike VAEs, deterministic autoencoders can not generate samples. Nonetheless, this is not an impairment for link prediction, in which reconstruction is the goal.

4 Related works

Kipf and Welling [2] first introduce GraphVAEs as a framework to learn node latent variables and handle reconstruction tasks. Building upon this model, Salha *et al.* [16] show that a simple linear one-hop model outperforms deeper GCN-based auto-encoding architectures. In this regard, our work extends their model by combining multi-hop information. Curiously, we found that removing the 1-hop information (the one they use) produces the best results (see Sec. 5).

The idea of combining multi-hop information to improve graph models is not novel. For instance, SIGN [20] proposes concatenating multiple pre-computed diffusion matrices for fast inference in node classification tasks. Our work leverages this idea to graph generative models in reconstruction tasks. Behind all these methods, it is the idea that we can build simple yet effective GNNs for a variety tasks. This is related to the fact the many real-world tasks fall under the *homophily* setting, in which neighboring nodes have similar features or labels. In these cases, simple low-pass filtering designs often yield good results [21, 11].

5 Experiments

Setup. We follow the same setup as Salha *et al.* [16]. Therefore, we evaluate our proposals in link prediction tasks and use the Cora, Citeseer and Pubmed citation networks as benchmarks. We run experiments on two versions of each dataset: with and without node features. For the latter, we substitute the feature matrix by an identity matrix.

We compare our methods against Graph(V)AE with GCN-based encoders [2], and Linear (V)AE with one-hop encoders [16]. Our methods combine 2-, 3-, and 4-hop neighborhood information ($C = 4$ and we do not use 1-hop information). Moreover, we use the Area Under the ROC Curve (AUC) and Average Precision (AP) as metrics. We repeat the experiments 20 times. For each method, we report mean and standard deviation as results. Our PyTorch implementation is available at: `https://github.com/ErikJhones/MLGVAE`.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Model	Cora w/ features		CiteSeer w/ features		PubMed w/ features	
	AUC	AP	AUC	AP	AUC	AP
GCN VAE ($L = 2$)	91.6±0.92	92.6±0.91	90.7±1.01	92.0±0.97	94.6±0.51	94.8±0.42
GCN VAE ($L = 3$)	90.5±0.94	91.7±0.88	88.6±0.95	90.2±0.81	92.7±1.02	93.3±0.91
GCN AE ($L = 2$)	91.2±0.78	92.4±0.71	89.7±1.39	90.3±1.62	96.2±0.36	96.2±0.25
GCN AE ($L = 3$)	89.1±1.18	90.9±1.01	87.3±1.74	89.6±1.52	94.8±0.41	95.4±0.26
Linear GVAE	92.5±0.97	93.6±0.68	90.6±0.86	91.8±0.77	94.9±1.10	94.5±0.89
Linear GAE	92.0±0.93	93.3±0.86	91.5±1.17	92.9±0.97	95.8±0.20	95.8±0.17
ML-GVAE (sum)	93.1±0.35	93.5±0.77	92.8±0.61	93.3±0.74	**98.1±0.92**	**98.3±0.16**
ML-GVAE (cat)	93.5±0.74	**94.2±0.17**	**93.6±0.83**	93.5±0.87	96.7±0.29	96.7±0.11
ML-GAE (sum)	92.4±0.72	93.4±0.67	92.2±0.96	93.2±0.72	97.1±0.67	97.2±0.12
ML-GAE (cat)	**93.7±0.37**	94.1±0.35	93.4±0.95	**94.0±0.53**	96.9±0.70	97.0±0.34

Table 1: Link prediction using node features. ML-G(V)AEs consistently outperform GCN-based Graph(V)AE and Linear G(V)AE in all cases.

Model	Cora		CiteSeer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
GCN VAE ($L = 2$)	84.5±1.05	88.8±1.04	77.5±1.24	82.8±0.91	84.1±0.31	88.0±0.22
GCN VAE ($L = 3$)	84.5±1.42	87.6±1.08	79.3±1.78	83.7±1.13	84.1±0.47	88.2±0.31
GCN AE ($L = 2$)	84.8±1.10	88.4±0.82	78.2±1.69	83.8±1.24	82.5±0.64	87.4±0.38
GCN AE ($L = 3$)	84.6±1.22	87.6±1.11	78.6±1.74	82.8±1.43	83.3±0.98	87.6±0.68
Linear GVAE	84.7±1.24	88.2±1.02	78.9±1.34	83.3±0.99	84.0±0.28	87.9±0.25
Linear GAE	83.2±1.13	87.5±0.95	77.0±1.81	83.0±1.25	81.8±0.32	87.5±0.28
ML-GVAE (sum)	85.5±0.21	89.1±0.12	78.6±0.35	83.5±0.43	**90.1±0.42**	**91.5±0.54**
ML-GVAE (cat)	**86.3±0.51**	**89.8±0.61**	**79.9±0.49**	**84.7±0.71**	86.8±0.22	88.1±0.34
ML-GAE (sum)	85.2±0.16	88.9±0.58	78.3±0.16	83.4±0.05	82.1±0.56	87.9±0.18
ML-GAE (cat)	84.8±0.12	89.1±0.32	77.8±0.18	83.2±0.11	83.7±0.58	88.1±0.63

Table 2: Link prediction without node features. Our multi-hop linear encoders obtain greater AUC and AP for all datasets. The performance gap is particularly greater on Pubmed.

Results. Table 1 and Table 2 show that ML-GVAE (sum) and ML-VAE (cat) outperform the other methods for all datasets. The performance gap between ML-G(V)AEs and competitors is particularly noticeable in the PubMed dataset, with and without features. A possible explanation is that our methods are the only to leverage information from neighborhoods of size three and four, since GCN-based (V)AEs and Linear G(V)AEs employ two-layer GCNs as encoders.

6 Conclusion

This papers proposes ML-GVAE, an improved version of GraphVAE [2]. By combining multiple SGCs [11] for the encoder, our method is capable of leveraging multi-hop information. Results show that ML-GVAE consistently outperforms GraphVAEs and its simplified variant [16], Linear GVAE, in link prediction tasks. Additionally, our method is arguably faster than GraphVAEs, since we can pre-compute propagated features before training/test.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

In future work, we will evaluate ML-GVAE in a larger number of large-scale benchmarks. Additionally, Ghosh *et al.* [22] show that the noise injection ($\boldsymbol{\sigma}_i$'s) in VAEs can be replaced by directly regularizing the latent embeddings. Therefore, we believe that combining ML-GVAE with similar tricks can lead to efficient models with simple loss landscapes. We will explore this direction in future work.

References

[1] D. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[2] T. Kipf and M. Welling. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

[3] M. Simonovsky and N. Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *ICANN*, 2018.

[4] A. Grover, A. Zweig, and S. Ermon. Graphite: Iterative generative modeling of graphs. In *ICML*, 2019.

[5] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv:1802.04407*, 2018.

[6] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *CIKM*, 2017.

[7] R. Berg, T. Kipf, and M. Welling. Graph convolutional matrix completion. *arXiv:1706.02263*, 2017.

[8] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *KDD*, 2016.

[9] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, 2018.

[10] A. Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[11] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

[12] Q. Huang, H. He, A. Singh, S. Lim, and A. Benson. Combining label propagation and simple models out-performs graph neural networks. In *ICLR*, 2021.

[13] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In *ICML*, 2020.

[14] F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. In *ICLR*, 2020.

[15] D. Mesquita, A. H. Souza, and S. Kaski. Rethinking pooling in graph neural networks. In *NeurIPS*, 2020.

[16] G. Salha, R. Hennequin, and M. Vazirgiannis. Simple and effective graph autoencoders with one-hop linear models. In *ECML-PKDD*, 2020.

[17] T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[18] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Mag.*, 34(4):18–42, 2017.

[19] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[20] E. Rossi, F. Frasca, B. Chamberlain, D. Eynard, M. M. Bronstein, and F. Monti. SIGN: scalable inception graph neural networks. *Arxiv:2004.11198*, 2020.

[21] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *NeurIPS*, 2020.

[22] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. Black, and B. Scholkopf. From variational to deterministic autoencoders. In *ICLR*, 2020.