Graph Embedding VAE: A Permutation Invariant Model of Graph Structure

Tony Duan
tonyduan@cs.stanford.edu*

Juho Lee
juho@aitrics.com

Abstract

Generative models of graph structure have applications in biology and social sciences. The state of the art is GraphRNN, which decomposes the graph generation process into a series of sequential steps. While effective for modest sizes, it loses its permutation invariance for larger graphs. Instead, we present a permutation invariant latent-variable generative model relying on graph embeddings to encode structure. Using tools from the random graph literature, our model is highly scalable to large graphs with likelihood evaluation and generation in $O(|V| + |E|)$.

1 Method

We focus on learning a generative model of un-directed graph structure without node labels. Let $G = (V, E)$ denote a graph represented by a symmetric adjacency matrix $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$. Note that we are interested in the *inductive* (across graphs) setting rather than transductive (single graph).

Permutation invariance. We begin with a few useful definitions (Zaheer et al., 2017).

Definition 1. A function $f : \mathcal{X}^n \to \mathcal{Y}$ is *permutation-invariant* if and only if it satisfies $f(\pi x) = f(x)$ for any permutation $\pi \in S_n$, the set of permutations of indices $\{1, \ldots, n\}$.

Definition 2. A function $f : \mathcal{X}^n \to \mathcal{Y}^n$ is *permutation-equivariant* if and only if it satisfies $f(\pi x) = \pi f(x)$ for any permutation $\pi \in S_n$, the set of permutations of indices $\{1, \ldots, n\}$.

Within the context of graphs, the Message Passing Neural Network (MPNN) (Gilmer et al., 2017) is the most popular permutation-equivariant model. In its simplest form, a MPNN acts on a set of node features $\mathbf{X}$ given a fixed adjacency matrix $\mathbf{A}$ by layers of message passing, described by

$$\text{MPNNLayer}(\mathbf{X}; \mathbf{A}) = \sigma(\mathbf{AXW}), \qquad \text{where } \sigma \text{ is a non-linearity.}$$

Another permutation-equivariant model is the Set Transformer, which is composed of layers called the Induced Self-Attention Block (ISAB). Based on multihead attention (Vaswani et al., 2017), an ISAB computes the pairwise interactions between the n elements in $\mathbf{X}$.

$$\text{ISAB}(\mathbf{X}) = \text{MultiheadAttention}(\mathbf{X}, \text{MultiheadAttention}(\mathbf{I}, \mathbf{X})),$$

where $\mathbf{I}$ is a set of m trained inducing points. Instead of computing self-attention directly on $\mathbf{X}$ requiring $O(n^2)$ time complexity, ISAB indirectly compares the elements in $\mathbf{X}$ via the reference points $\mathbf{I}$, thus reducing the time-complexity to $O(nm)$. It is worth noting that an ISAB is a special case of a MPNN with a fully connected adjacency matrix $\mathbf{A}$.

Joint permutation invariance. To satisfy permutation invariance with respect to arbitrary node re-orderings, we want to learn a likelihood model p_θ such that:

$$p_\theta(\mathbf{PAP}^\top) = p_\theta(\mathbf{A}), \text{for all permutation matrices } \mathbf{P}.$$

*Work done as an intern at AITRICS.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

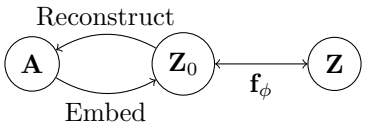

Figure 1: Graphical model summarizing our approach. We use a permutation-equivariant graph embedding $\mathbf{A} \to \mathbf{Z}_0$ to encode adjacency matrices into latent space, then apply a normalizing flow $\mathbf{Z}_0 \leftrightarrow \mathbf{Z}$. We rely on an approximate posterior for reconstruction $\mathbf{Z}_0 \to \mathbf{A}$.

Note that this is a different type of symmetry than the permutation equivariance we've already described, because the rows *and* columns of $\mathbf{A}$ need to be re-ordered when the permutation is applied. Following Bloem-Reddy and Teh (2019), we call this type of symmetry *joint invariance*.

Definition 3. A function $f : \mathcal{X}^{n\times n} \to \mathcal{Y}$ is *jointly permutation-invariant* if and only if it satisfies $f(\pi x \pi^\top) = f(x)$ for any permutation $\pi \in S_n$, the set of permutations of indices $\{1, \ldots, n\}$.

Definition 4. A function $f : \mathcal{X}^{n\times n} \to \mathcal{Y}^n$ is *jointly permutation-equivariant* if and only if it satisfies $f(\pi x \pi^\top) = \pi f(x)$ for any permutation $\pi \in S_n$, the set of permutations of indices $\{1, \ldots, n\}$.

Latent-variable models. Instead of learning the adjacency matrix $p_\theta(\mathbf{A})$ directly, we introduce a latent variable $\mathbf{Z}$. Following the standard formulation of variational auto-encoders (Kingma and Welling, 2014), we introduce a variational approximation to model the intractable posterior.

$$p_\theta(\mathbf{A}) = \int_{\mathbf{Z}} p(\mathbf{Z}) p_\theta(\mathbf{A}|\mathbf{Z}) \qquad q_\phi(\mathbf{Z}|\mathbf{A}) \approx p_\theta(\mathbf{Z}|\mathbf{A})$$

This puts our method in the same line of work as VGAE (Kipf and Welling, 2016) and Graphite (Grover et al., 2019). Both are latent-variable models with permutation-equivariant decoders $p_\theta(\mathbf{A}|\mathbf{Z}, \mathbf{X})$ and permutation-equivariant encoders $q_\phi(\mathbf{Z}|\mathbf{A}, \mathbf{X})$, instantiated as message-passing neural networks (Gilmer et al., 2017). However, these prior works rely on the availability of node features $\mathbf{X}$ for message passing. In the absence of node features an arbitrary ordering of nodes is used to learn initial embeddings (by setting $\mathbf{X} = \mathbf{I}_n$), resulting in the loss of permutation-equivariance.

Graph embeddings. How do we break symmetries between nodes in the absence of node features? We explore use of a graph embedding to encode structure of the graph. Formally, we employ a function $\mathrm{Embed} : \{0,1\}^{|V|\times|V|} \to \mathbb{R}^{|V|\times P}$ that satisfies joint permutation-equivariance. The textbook example of a graph embedding method is the Laplacian Eigenmap (Belkin and Niyogi, 2003; Verma and Zhang, 2017), defined via the eigendecomposition of the Laplacian matrix.

$$\mathrm{Embed}(\mathbf{A}) = \mathbf{\Phi}^\top, \qquad \mathbf{D} - \mathbf{A} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top.$$

The Laplacian Eigenmap is our canonical example of an embedding method, though in experiments we investigate Locally Linear Embeddings as well (Roweis and Saul, 2000). More recently developed deep learning embeddings built on stochastic random walks could theoretically be employed as well (Perozzi et al., 2014; Grover and Leskovec, 2016; Abu-El-Haija et al., 2018). However, we note that such methods are typically invariant to permutations of the embedding dimensions, resulting in a different type of symmetry, so we leave their investigation to future work.

Encoder. We begin our variational posterior with a graph embedding, then apply a normalizing flow to improve expressivity (Rezende and Mohamed, 2015). Letting $\mathbf{f}_\phi$ denote a differentiable invertible transformation (potentially composed of a chain of simpler such tranformations), we have

$$\mathbf{Z}_0|\mathbf{A} \sim \mathrm{Normal}(\mathrm{Embed}(\mathbf{A}), \sigma^2\mathbf{I}) \qquad \mathbf{Z} = \mathbf{f}_\phi(\mathbf{Z}_0)$$

$$\log q_\phi(\mathbf{Z}|\mathbf{A}) = \log q(\mathbf{Z}_0|\mathbf{A}) - \log \left| \det \frac{\partial \mathbf{f}_\phi}{\partial \mathbf{Z}_0} \right|.$$

We parameterize $\mathbf{f}_\phi$ as a Neural Spline Flow over coupling layers (Durkan et al., 2019) for adequate expressivity. Coupling and 1×1 convolutions are performed over the P dimensions of the embeddings. To ensure permutation equivariance while allowing dependencies between nodes, the splines for each coupling layer are parameterized by a stack of ISABs. We note that stacking self-attention layers over the node embeddings results in potentially complex interactions between nodes that have typically been captured via message-passing neural networks. However, the use of ISABs requires only $O(|V|m)$ complexity instead of $O(|V|^2)$ complexity, where m is the number of inducing points.

Decoder. Our decoder applies the inverse flow $\mathbf{f}_\phi^{-1}$, an ISAB, then a Bernoulli-Exponential link.

$$\mathbf{Z} \sim \text{Normal}(\mathbf{0}, \mathbf{I}) \qquad \mathbf{Z}^* = \text{ISABStack}(\mathbf{f}_\phi^{-1}(\mathbf{Z}))$$
$$\mathbf{A}|\mathbf{Z}^* \sim \text{Bernoulli-Exponential}(\mathbf{Z}^*\mathbf{Z}^{*\top})$$

The Bernoulli-Exponential link (Zhou, 2015; Caron, 2012) is defined by augmenting the model with truncated Exponential random variables $\mathbf{M} \in [0,1]^{|V|\times|V|}$ corresponding to entries of the adjacency matrix $\mathbf{A}$. Letting $\mathbf{z}_i^*, \mathbf{z}_j^*$ denote rows of $\mathbf{Z}^*$ corresponding to the i-th and j-th nodes,

$$a_{i,j}|\mathbf{z}_i^*, \mathbf{z}_j^* \sim \text{Bernoulli}(m_{i,j} < 1) \qquad m_{i,j}|\mathbf{z}_i^*, \mathbf{z}_j^* \sim \text{Exponential}(\mathbf{z}_i^{*\top}\mathbf{z}_j^*)$$

The joint log-likelihood can then be expressed as

$$\log p_\theta(\mathbf{A}, \mathbf{M}|\mathbf{Z}^*) = \sum_{(i,j)\in E} \Big(\log(\mathbf{z}_i{*}^\top \mathbf{z}_j^*) - \mathbf{z}_i^{*\top}\mathbf{z}_j^*(m_{i,j}-1)\Big) - \frac{1}{2}\left((\sum_{i=1}^n \mathbf{z}_i^*)^\top(\sum_{i=1}^n \mathbf{z}_i^*) - \sum_{i=1}^n ||\mathbf{z}_i^*||_2^2\right).$$

The advantage of the Bernoulli-Exponential link function over a traditional (ex. logistic) link function is scalability; the joint log-likelihood can be can be calculated in $O(|E|+|V|)$ instead of $O(|V|^2)$. We sample from the analytic posterior for inference, noting that we only need to sample the $|E|$ auxiliary variables where $a_{i,j} = 1$ (δ_1 below denotes the Dirac delta function centered at 1). Since the auxiliary random variables are continuous the re-parameterization trick can be used.

$$q(\mathbf{M}|\mathbf{A}, \mathbf{Z}^*) = \prod_{i<j} q(m_{i,j}|a_{i,j}, \mathbf{z}_i^*, \mathbf{z}_j^*) \qquad q(m_{i,j}|a_{i,j}) = (1-a_{i,j})\delta_1 + a_{i,j}p(m_{i,j}|\mathbf{z}_i^*, \mathbf{z}_j^*)$$

Summary. The high-level idea is summarized in Figure 1. We fit by optimizing the ELBO,

$$\begin{aligned}
\log p_\theta(\mathbf{A}) &\geq \mathbb{E}_{q_\phi(\mathbf{Z},\mathbf{M}|\mathbf{A})}[\log p_\theta(\mathbf{A}, \mathbf{Z}, \mathbf{M}) - \log q_\phi(\mathbf{Z}, \mathbf{M}|\mathbf{A})] \\
&= \mathbb{E}_{q(\mathbf{Z}_0,\mathbf{M}|\mathbf{A})}[\log p(\mathbf{Z}) + \log p_\theta(\mathbf{A}, \mathbf{M}|\mathbf{Z}) - \log q_\phi(\mathbf{Z}|\mathbf{A}) - \log q(\mathbf{M}|\mathbf{A}, \mathbf{Z})] \\
&= \mathbb{E}_{q(\mathbf{Z}_0,\mathbf{M}|\mathbf{A})}\left[\log p(\mathbf{Z}) + \log p_\theta(\mathbf{A}, \mathbf{M}|\mathbf{Z}_0) - \log q(\mathbf{Z}_0|\mathbf{A}) - \log q(\mathbf{M}|\mathbf{A}, \mathbf{Z}) + \log\left|\det \frac{\partial \mathbf{f}_\phi}{\partial \mathbf{Z_0}}\right|\right]
\end{aligned}$$

We call our jointly permutation-invariant generative model the Graph Embedding VAE (GE-VAE).

2 Related Work

Deep generative models of graphs. So far the most successful inductive model of graph structure has been GraphRNN (You et al., 2018), which fits an auto-regressive model to sequences of node and edge formations derived from $\mathbf{A}$. The factorization implied by each sequence is dependent on chosen node orderings, so the model is not permutation invariant. However, by amortizing over sampled breadth-first orderings the model is approximately permutation invariant for modest graph sizes $|V|$. Graphite (Grover et al., 2019) is a variational auto-encoder model with permutation-equivariant MPNNs for encoding and decoding, but in the absence of node features relies on an arbitrary node ordering by setting node features $\mathbf{X} = \mathbf{I}$. Graph Normalizing Flow (GNF) (Liu et al., 2019) uses permutation equivariant MPNNs to parameterize coupling layers in a normalizing flow model. They initialize node embeddings by sampling $\mathbf{X} \sim \text{Normal}(\mathbf{0}, \sigma^2\mathbf{I})$, which is invariant to node re-orderings. However, they require a separate decoder for generating samples $\mathbf{A}|\mathbf{Z}$, by reverse message-passing. We note that without $\mathbf{X}$ both Graphite and GNF require message-passing over fully connected $\mathbf{A}$ to sample new graphs, a step which we replace with the Set Transformer.

Permutation invariant and equivariant models. Zaheer et al. (2017) first introduced permutation invariance and equivariance in the context of deep models. Herzig et al. (2018) introduced *graph permutation-invariance* which is similar to our notion of joint permutation-equivariance under permutations of $\mathbf{A}$, but assumes the presence of unique node features $\mathbf{X}$ to break symmetry. Hartford et al. (2018) discuss *exchangeable matrix invariance*, which reflects separate symmetries in separate permutations of rows and columns of $\mathbf{A}$, but not joint permutations. Bloem-Reddy and Teh (2019) provide a review of the above definitions that capture the symmetry in graphs.

			Graph Embedding VAE				GraphRNN		
Dataset	$\max \lvert V \rvert$	$\max \lvert E \rvert$	bits/dim	degree	cluster	orbit	degree	cluster	orbit
Community	160	1945	0.297	0.011	0.056	0.002	0.014	0.002	0.039
Ego	399	1071	0.155	0.116	0.711	0.163	0.077	0.316	0.030
Grid	361	684	0.071	0.779	0.026	0.509	10^{-5}	0	10^{-4}
Protein	500	1575	0.114	0.591	1.563	0.451	0.034	0.935	0.217

Table 1: Test set MMD and log-likelihood graph generation statistics comparing our method GE-VAE with GraphRNN. Results for GraphRNN are reported from You et al. (2018).

3 Experiments

We experiment with several datasets, following the GraphRNN (You et al., 2018) codebase. (1) Community: 3500 two-community graphs with ER clusters. (2) Ego: 757 3-hop ego networks extracted from Citeseer (Sen et al., 2008). (3) Grid: 3500 standard 2D grid graphs. (4) Protein: 918 protein graphs over amino acids (Dobson and Doig, 2003).

All datasets were split into roughly $\frac{2}{3}$ into training and $\frac{1}{3}$ into test sets. In order to handle graphs of various sizes in a dataset, we take only the eigenvectors corresponding to the smallest P eigenvalues of each graph's unnormalized Laplacian. We implement masking in all self-attention steps, and maximize the reconstruction log-probability *per edge* (i.e. per dimension).

We evaluate by reporting Maximum Mean Discrepancy (MMD) statistics over degree distributions, clustering coefficient distributions, and orbit count statistics (Table 1). For the GE-VAE we report estimated test set log-likelihoods as well. We exhibit visualizations of generated graphs in Figure 2.

Overall we find that the GE-VAE is competitive with GraphRNN on the Community and Ego datasets, but outperformed by GraphRNN on the Grid and Protein datasets. We suspect that this is due to the extremely multimodal nature of these graphs that are difficult to capture with a latent-variable model.

4 Discussion

Embedding limitations. The primary limitation of our method is heavy dependence on graph embeddings to encode the structure of the adjacency matrix in a jointly permutation-equivariant way. There is a heavy upfront computation cost to calculating embeddings for large graphs. Moreover, graph embeddings (such as the Laplacian Eigenmap) are not generally scale-invariant; as graph size increases, the representations encoded each dimension of the node embeddings do not straightforwardly map between graphs (see Figure 3 in the Appendix).

Meta-learning embeddings. A meta-learned graph embedding model that is simultaneously trained along with our latent-variable model would allow for more flexible representations. Such a process would need to be permutation-equivariant, and also able to break symmetries by being position-aware (a simple MPNN would not suffice) (You et al., 2019). We leave this for future work.

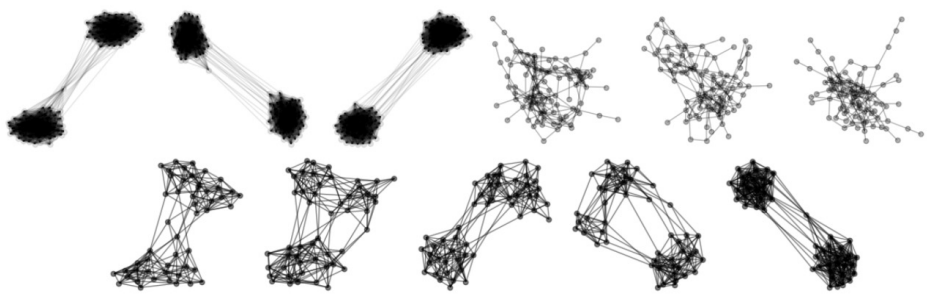

Figure 2: (Top) Generated graphs for the Community and Ego datasets. (Bottom) Interpolation in latent space between a four and two community graph structure.

References

Abu-El-Haija, S., Perozzi, B., Al-Rfou, R., and Alemi, A. A. (2018). Watch Your Step: Learning Node Embeddings via Graph Attention. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 9180–9190. Curran Associates, Inc.

Belkin, M. and Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396.

Bloem-Reddy, B. and Teh, Y. W. (2019). Probabilistic symmetry and invariant neural networks. Technical report. arXiv: 1901.06082.

Caron, F. (2012). Bayesian nonparametric models for bipartite graphs. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 2051–2059. Curran Associates, Inc.

Dobson, P. D. and Doig, A. J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural Spline Flows. Technical report. arXiv: 1906.04032.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural Message Passing for Quantum Chemistry. In *International Conference on Machine Learning*, pages 1263–1272.

Grover, A. and Leskovec, J. (2016). Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA. ACM. event-place: San Francisco, California, USA.

Grover, A., Zweig, A., and Ermon, S. (2019). Graphite: Iterative Generative Modeling of Graphs. In *International Conference on Machine Learning*, pages 2434–2444.

Hartford, J., Graham, D., Leyton-Brown, K., and Ravanbakhsh, S. (2018). Deep Models of Interactions Across Sets. In *International Conference on Machine Learning*, pages 1909–1918.

Herzig, R., Raboh, M., Chechik, G., Berant, J., and Globerson, A. (2018). Mapping Images to Scene Graphs with Permutation-Invariant Structured Prediction. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 7211–7221. Curran Associates, Inc.

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*. arXiv: 1312.6114.

Kipf, T. N. and Welling, M. (2016). Variational Graph Auto-Encoders. Technical report. arXiv: 1611.07308.

Kumar, A., Poole, B., and Murphy, K. (2019). Learning Generative Samplers using Relaxed Injective Flow. Technical report.

Liu, J., Kumar, A., Ba, J., Kiros, J., and Swersky, K. (2019). Graph Normalizing Flows. Technical report. arXiv: 1905.13177.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA. ACM. event-place: New York, New York, USA.

Rezende, D. and Mohamed, S. (2015). Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, pages 1530–1538.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective Classification in Network Data. *AI Magazine*, 29(3):93–93.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, , and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Verma, S. and Zhang, Z.-L. (2017). Hunt For The Unique, Stable, Sparse And Fast Feature Learning On Graphs. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 88–98. Curran Associates, Inc.

You, J., Ying, R., and Leskovec, J. (2019). Position-aware Graph Neural Networks. In *International Conference on Machine Learning*, pages 7134–7143.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. (2018). GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *International Conference on Machine Learning*, pages 5708–5717.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep Sets. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc.

Zhou, M. (2015). Infinite Edge Partition Models for Overlapping Community Detection and Link Prediction. In *Artificial Intelligence and Statistics*, pages 1135–1143.

5 Appendix

Injective flow perspective. Suppose we instead replace the first step of the encoder as

$$\mathbf{Z}_0|\mathbf{A} \sim \cancel{\text{Normal}(\text{Embed}(\mathbf{A}), \sigma^2\mathbf{I})} \qquad \mathbf{Z}_0|\mathbf{A} = \text{Embed}(\mathbf{A}).$$

Then applying $\mathbf{f}_\phi \circ \text{Embed}$ results in a series of transformations of $\mathbf{A}$ into the latent variable $\mathbf{Z}$. Crucially, graph embedding methods are not invertible, since there will always exist embeddings that do not correspond to any adjacency matrices – so we cannot interpret the composition as a series of invertible flows. However, when the graph embedding is injective (such as the case when the Laplacian Eigenmap is used (Verma and Zhang, 2017)), the result is an injective flow (Kumar et al., 2019).

$$\log p_\phi(\mathbf{A}) = \log p(\mathbf{f}_\phi \circ \text{Embed} \circ \mathbf{A}) + \frac{1}{2}\log\left|\det J_{\mathbf{f}_\phi \circ \text{Embed}}(\mathbf{A})^\top J_{\mathbf{f}_\phi \circ \text{Embed}}(\mathbf{A})\right|$$

Log-likelihood evaluation. We compute test set log-likelihood by Monte Carlo importance sampling with the variational posterior, using 128 samples.

$$\begin{aligned}
\log p_\theta(\mathbf{A}) &= \log \int_{\mathbf{Z}} p_\theta(\mathbf{A}, \mathbf{Z}) d\mathbf{Z} \\
&= \log \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{A})}\left[\frac{p_\theta(\mathbf{Z}, \mathbf{A})}{q_\phi(\mathbf{Z}|\mathbf{A})}\right] \\
&= \log \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{A})}\left[\frac{p(\mathbf{Z})p_\theta(\mathbf{A}|\mathbf{Z})\left|\det \frac{\partial \mathbf{f}_\phi}{\partial \mathbf{Z}_0}\right|}{q_\phi(\mathbf{Z}_0|\mathbf{A})}\right]
\end{aligned}$$

We take the sum of the upper triangular entries of $p_\theta(\mathbf{A}|\mathbf{Z})$ and then divide by $\frac{1}{2}|V|(|V|-1)$ to calculate the number of bits per dimension, independent of the number of nodes in the graph.

Generating large-scale graphs We can generate large-scale graphs efficiently by interpreting the Bernoulli-Exponential link as augmented Poisson random variables instead of augmented Exponential random variables. Recalling that the marginal $p(a_{i,j} = 1) = 1 - e^{-\mathbf{z}_i^{*\top}\mathbf{z}_j^*}$, let

$$a_{i,j}|\mathbf{z}_i^*, \mathbf{z}_j^* \sim \text{Bernoulli}(m_{i,j} = 0) \qquad m_{i,j}|\mathbf{z}_i^*, \mathbf{z}_j^* \sim \text{Poisson}(\mathbf{z}_i^{*\top}\mathbf{z}_j^*).$$

It follows that the total number of edges is distributed as

$$\begin{aligned}
E = \sum_{i<j} a_{i,j} &\sim \text{Poisson}\left(\sum_{i<j} \mathbf{z}_i^{*\top}\mathbf{z}_j^*\right) \\
&\sim \text{Poisson}\left(\frac{1}{2}\left((\sum_{i=1}^n \mathbf{z}_i^*)^\top(\sum_{i=1}^n \mathbf{z}_i^*) - \sum_{i=1}^n ||\mathbf{z}_i^*||_2^2\right)\right) \\
&\sim \sum_{d=1}^P \text{Poisson}\left(\frac{1}{2}(\sum_{i=1}^n z_{i,d}^*)(\sum_{i=1}^n z_{i,d}^*) - \sum_{i=1}^n z_{i,d}^{*2}\right) \\
&\stackrel{\text{def}}{=} \sum_{d=1}^P E_d.
\end{aligned}$$

So we can first sample the total number of edges E by sampling E_d (corresponding to each dimension), then sample the nodes (i, j) corresponding to each edge by picking

$$p(i)|E_d \propto z_{i,d}^*.$$

Additional figures.

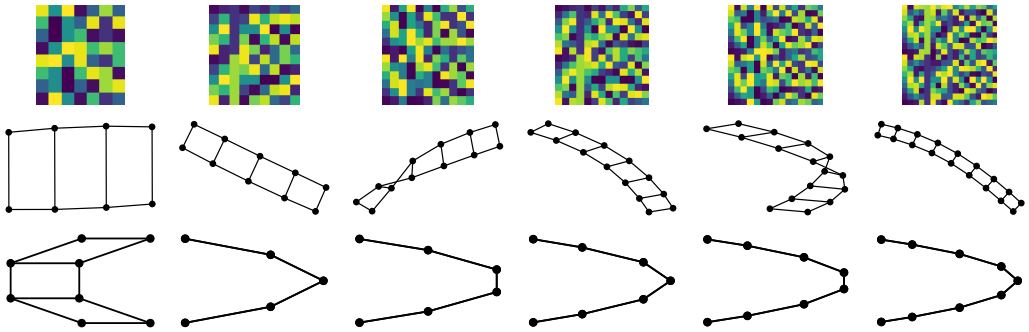

Figure 3: Laplacian Eigenmap embeddings for a sequence of ladder graphs of increasing size. The top row shows the embeddings and the bottom row shows the projections onto the first two columns (i.e. eigenvectors). As the size of the graph grows, the number of dimensions necessary to different betweeen the two columns of the ladder increases.