Graphite: Iterative Generative Modeling of Graphs

Aditya Grover*
Stanford University
adityag@cs.stanford.edu

Aaron Zweig*
Stanford University
azweig@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

Graphs are a fundamental abstraction for modeling relational data. However, graphs are discrete and combinatiorial in nature, and learning representations suitable for machine learning tasks poses statistical and computational challenges. In this work, we propose *Graphite* an algorithmic framework for unsupervised learning of representations over nodes in a graph using deep latent variable generative models. Our model is based on variational autoencoders (VAE), and differs from existing VAE frameworks for data modalities such as images, speech, and text in the use of spectral graph convolutions for parameterizing *both* the generative model (a.k.a. decoder) and inference model (a.k.a. encoder). The use of graph convolutions directly incorporates inductive biases specific to graphs, such as permutation invariance to node orderings and locality preference for clustering node representations, in the generative model. We demonstrate empirically that Graphite outperforms state-of-the-art approaches for representation learning over graphs for the task of link prediction on benchmark datasets.

1 Introduction

Latent variable generative modeling is an effective approach for unsupervised representation learning of high-dimensional data [11]. In recent years, representations learned by latent variable models parameterized by deep neural networks have shown impressive performance on many tasks such as semi-supervised learning and structured prediction [6, 16]. However, these successes have been restricted to specific data modalities such as images and speech. In particular, current deep generative models cannot be directly applied to graph-structured data. Graphs arise in a wide variety of domains in physical sciences, information sciences, and social sciences, and consequently algorithms for unsupervised representation learning of such data have several downstream applications, including node classification, link prediction, and community detection [3, 10, 4].

In recent works, the notion of *spectral graph convolutions* has been the guiding principle for designing neural network architectures that can directly operate on graphs [1, 2]. A spectral graph convolution is defined as the multiplication of a signal (*i.e.*, feature matrix associated with the nodes) with a parameterized filter, in the Fourier space of a graph. Graph convolution networks (GCN), in particular, efficiently compute local first-order approximations to spectral graph convolutions, and have been successfully applied across several graph mining tasks such as semi-supervised learning and relational learning [9]. Such tasks involve *encoding* an input graph to a final output representation (such as the labels associated with the nodes) with potentially more intermediate layers. The inverse problem of learning to *decode* a hidden representation into a graph, as in the case of a latent variable generative model, is to the best of our knowledge largely an open question that we address in this work.

We propose Graphite, a framework for iteratively learning latent variable generative models of graphs. Specifically, we learn a directed model expressing a joint distribution $P_\theta(\mathbf{A}, \mathbf{Z})$ over the adjacency matrix of a graph $\mathbf{A} \in \{0,1\}^{n \times n}$ and a latent feature matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$ such that every row corresponds to a feature vector for a node in the graph. The multi-layer iterative decoding

*Equal Contribution.

Workshop on Deep Learning for Physical Sciences (DLPS 2017), NIPS 2017, Long Beach, CA, USA.

process for specifying the conditional distribution $P_\theta(\mathbf{A}|\mathbf{Z})$ first constructs an intermediate graph using an inner-product operation followed by a sequence of graph convolutional layers on this intermediate graph. We learn the model parameters θ by maximizing a variational lower bound to the log-likelihood assigned by the model to the observed graph. Our empirical evaluations show that Graphite outperforms competing algorithms for the task of link prediction on benchmark datasets.

2 Learning framework

We first define some notation and give a brief background on spectral graph convolutions before presenting Graphite. Consider an undirected graph $G = (V, E)$ where V and E denote the set of nodes and edges respectively. We represent the graph using an adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ where $n = |V|$ and diagonal entries $A_{ii} = 1$. Additionally, we denote the feature matrix associated with the graph as $\mathbf{X} \in \mathbb{R}^{n \times m}$ for an m-dimensional signal associated with each node in the graph. If there are no explicit node features, we set $\mathbf{X} = \mathbf{I}_n$ (identity). Let $\mathbf{D}$ be the diagonal degree matrix such that $D_{ii} = \sum_{(i,j) \in E} A_{ij}$ and hence the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$. We define $\tilde{\mathbf{A}}$ to be the symmetric normalization of A given by $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.

The Fourier basis of a graph is given by the eigenvector matrix of the graph Laplacian. A spectral graph convolution convolves the feature matrix $\mathbf{X}$ with a parameterized filter $F_\theta = \mathrm{diag}(\theta)$ (where $\theta \in \mathbb{R}^n$ are the learned parameters), defined as a matrix multiplication in the Fourier basis of the graph. Formally, we define the graph convolution operation between F_θ and $\mathbf{X}$ for the graph $\mathbf{A}$ as:

$$F_\theta * \mathbf{X} = \mathbf{U} F_\theta \mathbf{U}^T \mathbf{X}$$

where $\mathbf{U}$ is the eigenvector matrix for the graph Laplacian, *i.e.*, $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T$ where Λ is the diagonal matrix of eigenvalues. Since eigendecomposition of the graph Laplacian is computationally prohibitive, [9] propose a local first-order approximation:

$$F_\theta * \mathbf{X} \approx \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{X}.$$

Extending the above approximation to a neural network design with several filters applied in each layer, we get the layerwise propagation rule for $l \geq 1$ a graph convolutional network (GCN):

$$\mathbf{H}^{(l)} = \eta(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{H}^{(l-1)}\Theta^{(l)})$$

where $\mathbf{H}^{(l)}$ and $\mathbf{H}^{(l-1)}$ are the l-th and $l-1$-th layers of the GCN with the base case $\mathbf{H}^{(0)} = \mathbf{X}$, $\Theta^{(l)}$ is a matrix of learnable parameters for the current layer, and η is a suitable activation function. For brevity throughout the paper, we will denote the propagation rule as:

$$\mathbf{H}^{(l)} = \mathrm{GCN}_{\eta,l}(\mathbf{A}, \mathbf{H}^{(l-1)}).$$

An alternate interpretation of a GCN layer is performing message passing over a graph using the Weisfeiler-Lehman (WL) algorithm [18]. Intuitively, every node in the graph passes 'messages' or information about its local structure to its neighbors. For a single GCN layer, the messages are passed to neighbors at a 1-hop distance from the node. In a multi-layer GCN with L-layers, information is propagated to neighbors up till an L-hop distance from the source. Refer to [9] for more details.

2.1 Graphite Variational Autoencoder

We use upper-case symbols to denote probability distributions and assume they all admit absolutely continuous densities (denoted by the corresponding lower-case notation) on a suitable reference measure. We are interested in learning a latent variable model that specifies a joint distribution $P_\theta(\mathbf{A}, \mathbf{Z}|\mathbf{X})$ conditioned on the feature matrix $\mathbf{X}$ of a graph $\mathbf{A}$ and latent variable matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$ where every row corresponds to a latent vector for a node in the graph. Similar to a variational autoencoder [7], our learning objective maximizes an evidence lower bound (ELBO) to the log-likelihood of the observed graph $\mathbf{A}$:

$$\log p_\theta(\mathbf{A}|\mathbf{X}) \geq \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{A},\mathbf{X})}\left[\log \frac{p_\theta(\mathbf{A},\mathbf{Z}|\mathbf{X})}{q_\phi(\mathbf{Z}|\mathbf{A},\mathbf{X})}\right]$$

where $q_\phi(\mathbf{Z}|\mathbf{A}, \mathbf{X})$ is a variational approximation to the true posterior $p_\theta(\mathbf{Z}|\mathbf{A}, \mathbf{X})$, parameterized by ϕ. We specify an isotropic standard Gaussian prior over $\mathbf{Z}$ such that the joint distribution factorizes as $P_\theta(\mathbf{A}, \mathbf{Z}|\mathbf{X}) = P_\theta(\mathbf{A}|\mathbf{Z}, \mathbf{X})P(\mathbf{Z})$. The observation model $P_\theta(\mathbf{A}|\mathbf{Z}, \mathbf{X})$ and the variational posterior $Q_\phi(\mathbf{Z}|\mathbf{A}, \mathbf{X})$, also referred to as the encoders and decoders respectively, are specified using deep neural networks. In particular, we use graph convolutional networks as described below.

Table 1: Area Under the ROC Curve (AUC) scores for link prediction (* denotes dataset with features)

	PPI	Cora	Citeseer	Pubmed	Cora*	Citeseer*	Pubmed*
SC	84.2 ± 0.34	89.9 ± 0.20	91.5 ± 0.17	**94.9** ± 0.04	-	-	-
DW	68.2 ± 0.08	85.0 ± 0.17	88.6 ± 0.15	91.5 ± 0.04	-	-	-
GAE	88.8 ± 0.01	90.2 ± 0.16	92.0 ± 0.14	92.5 ± 0.06	93.9 ± 0.11	94.9 ± 0.13	96.8 ± 0.04
VGAE	89.5 ± 0.07	90.1 ± 0.15	92.0 ± 0.17	92.3 ± 0.06	94.1 ± 0.11	96.7 ± 0.08	95.5 ± 0.13
Graphite-AE	91.1 ± 0.05	**91.4** ± 0.16	92.5 ± 0.16	94.5 ± 0.05	94.4 ± 0.10	94.6 ± 0.25	**97.8** ± 0.03
Graphite-VAE	**91.2** ± 0.05	**91.4** ± 0.16	**93.0** ± 0.12	94.6 ± 0.04	**94.7** ± 0.09	**97.2** ± 0.08	97.4 ± 0.04

Table 2: Average Precision (AP) scores for link prediction (* denotes dataset with features)

	PPI	Cora	Citeseer	Pubmed	Cora*	Citeseer*	Pubmed*
SC	88.9 ± 0.21	92.8 ± 0.12	94.4 ± 0.11	**96.0** ± 0.03	-	-	-
DW	69.0 ± 0.09	86.6 ± 0.17	90.3 ± 0.12	91.9 ± 0.05	-	-	-
GAE	89.4 ± 0.05	92.4 ± 0.12	94.0 ± 0.12	94.3 ± 0.5	94.3 ± 0.12	94.8 ± 0.15	96.8 ± 0.04
VGAE	89.6 ± 0.05	92.3 ± 0.12	94.2 ± 0.12	94.2 ± 0.04	94.6 ± 0.11	97.0 ± 0.08	95.5 ± 0.12
Graphite-AE	92.1 ± 0.05	92.4 ± 0.17	93.5 ± 0.19	95.7 ± 0.06	94.6 ± 0.11	94.3 ± 0.26	**97.7** ± 0.03
Graphite-VAE	**92.2** ± 0.06	**93.1** ± 0.13	**94.6** ± 0.12	**96.0** ± 0.03	**95.1** ± 0.08	**97.3** ± 0.08	97.4 ± 0.04

Encoder. The encoder $Q_\phi(\mathbf{Z}|\mathbf{A}, \mathbf{X})$ is a factorized multivariate Gaussian distribution specified using a multi-layer GCN, referred as EGCN (Encoder-GCN). In our experiments, we use two layers for the encoder and a multivariate Gaussian posterior with diagonal covariance. Hence, the forward pass is given by:

$$\begin{aligned} \mathbf{H}_e^{(1)} &= \mathrm{EGCN}_{\eta_1,1}(\mathbf{A}, \mathbf{X}) \\ \mu_q, \log \sigma_q &= \mathrm{EGCN}_{\eta_2,2}(\mathbf{A}, \mathbf{H}_e^{(1)}) \end{aligned}$$

where η_1 is any activation function and η_2 is set to identity (no activation) to search the full range of possible parameters for the Gaussian posterior.

Decoder. The decoder $P_\theta(\mathbf{A}|\mathbf{Z}, \mathbf{X})$ is a factorized Bernoulli distribution also specified using a multi-layer GCN, referred as a DGCN (Decoder-GCN). However, we do not have access to an explicit graph to perform graph convolutions unlike an EGCN that could directly access $\mathbf{A}$. We resolve this shortcoming by generating intermediate graphs using a similarity matrix that is a function of the latent feature matrix $\mathbf{Z} \sim Q_\phi(\mathbf{Z}|\mathbf{A}, \mathbf{X})$. The forward pass for the two-layer decoder is specified as:

$$\begin{aligned} \widehat{\mathbf{A}} &= \sigma(\mathbf{Z}\mathbf{Z}^T) \\ \mathbf{H}_d^{(1)} &= \mathrm{DGCN}_{\eta_1,1}(\widehat{\mathbf{A}}, [\mathbf{Z}|\mathbf{X}]) \\ \mathbf{Z}_* &= \mathrm{DGCN}_{\eta_2,2}(\widehat{\mathbf{A}}, \mathbf{H}_d^{(1)}) \end{aligned}$$

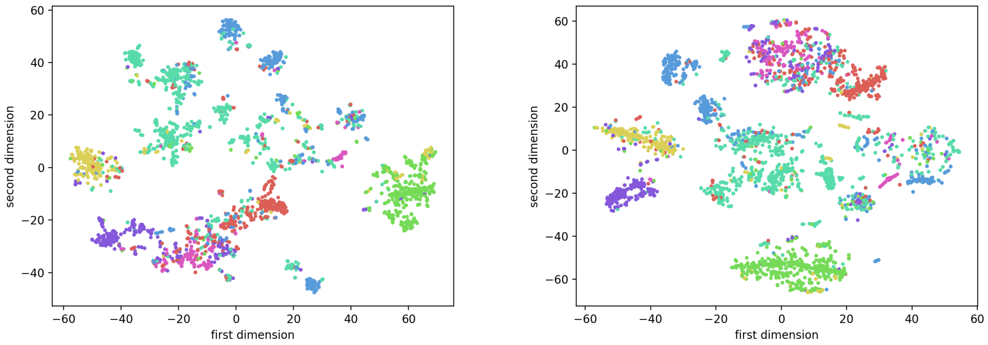
where η_1 is any suitable activation function and η_2 is the identity (such that $\mathbf{Z}$ and $\mathbf{Z}_*$ can take the same range of values) and the vertical bars denote concatenation of the original feature matrix $\mathbf{X}$ with $\mathbf{Z}$. For the final step of the decoding, we obtain the desired conditional distribution $p_\theta(\mathbf{A}|\mathbf{Z}, \mathbf{X})$ with an inner product over a feature matrix specified as the convex combination of $\mathbf{Z}$ and $\mathbf{Z}_*$.

$$\begin{aligned} \mathbf{Z}' &= \lambda \mathbf{Z} + (1 - \lambda)\mathbf{Z}_* \\ p_\theta(\mathbf{A}|\mathbf{Z}, \mathbf{X}) &= \Pi_{i=1}^n \Pi_{j=1}^n p_\theta(A_{ij}|\mathbf{Z}, \mathbf{X}) \\ \text{where } p_\theta(A_{ij}|\mathbf{Z}, \mathbf{X}) &= \sigma(\mathbf{Z}_i' \mathbf{Z}_j'). \end{aligned}$$

Here, $\mathbf{Z}_i'$ denotes the i-th row of $\mathbf{Z}'$ and $\lambda \in [0, 1]$ is a tunable hyperparameter.

2.2 Graphite Autoencoder

Similar to the differences between a standard autoencoder (AE) and a variational autoencoder, the EGCN here directly represents $\mathbf{Z}$ (instead of a variational posterior). Additionally, the learning objective minimizes the cross-entropy between the reconstructed graph probabilities and true graphs.



(a) Graphite-AE

(b) Graphite-VAE

Figure 1: t-SNE embeddings of the latent feature vectors for the Cora dataset. Colors denote labels.

3 Experimental Evaluation

We consider the task of link prediction [11]. Even though Graphite learns a distribution over graphs, it can be used for predictive tasks within a *single* graph. In the case of link prediction, we learn a model for a random, connected training subgraph of the true graph. For validation and testing, we add positive and negative (false) edges to the original graph and evaluate the performance of the model based on the reconstruction probabilities assigned to the validation and test edges (similar to denoising of the input graph). In our experiments, we held out a set of 5% edges for validation, 10% edges for testing, and train all models on the remaining subgraph.

We evaluate performance based on the Area Under the ROC Curve (AUC) and Average Precision (AP) metrics. Additionally, the validation and testing sets also each contain an equal number of non-edges. We compared across four standard benchmark datasets: a Protein-Protein Interaction (PPI) network for Homo Sapiens [17, 5] with proteins as nodes and interactions as edges, and three citation networks Cora, Citeseer, and Pubmed with papers as nodes and citations as edges [15]. For the citation networks, the text in the papers can be synthesized into optional node features.

For Graphite-AE and Graphite-VAE, we used an architecture of 32-32 units for the encoder and 16-32-16 units for the decoder trained using the Adam optimizer [7] with a learning rate of 0.01. The dropout rate (for edges) and λ were tuned as hyperparameters on the validation set to optimize the AUC. Additionally, we trained every model for 500 iterations and used the model checkpoint with the best validation loss for testing. We evaluated Graphite-VAE and Graphite-AE against competing methods for representation learning on graphs. The baselines we consider are Spectral Clustering (SC) [1], DeepWalk (DW) [14], Graph Autoencoder [8], and Variational Graph Autoencoder [8].

We used the SC implementation from [13] and public implementations for others made available by the authors. For SC, we used a dimension size of 128. For DW which uses a skipgram like objective on random walks from the graph, we used the same dimension size and default settings used in the paper of 10 random walks of length 80 per node and a context size of 10. For GAE and VGAE, we used the same architecture as VGAE and Adam optimizer with learning rate of 0.01. Note that SC and DW do not provide the ability to incorporate node features while learning embeddings, and hence we evaluate them only on the featureless datasets. GAE and VGAE are a special case of Graphite, using only a single inner-product decoder (*i.e.*, $\lambda = 1$).

The AUC and AP results (along with standard errors) are shown in Table 1 and Table 2 respectively averaged over 50 random train/validation/test splits. On both metrics, Graphite-VAE gives the best performance overall. Graphite-AE also gives good results, generally outperforming its closest competitor GAE. We further visualize the embeddings generated by a 2D t-SNE projection [12] of the latent feature vectors (given as rows for $\mathbf{Z}'$ with $\lambda = 0.5$) on the Cora dataset in Figure 1. Even without any access to label information for the nodes during training, the Graphite models are able to cluster the nodes (papers) as per their labels (paper categories).

References

[1] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2013.

[2] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[3] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[4] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[5] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[6] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

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

[8] T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

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

[10] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[11] J. C. Loehlin. *Latent variable models: An introduction to factor, path, and structural analysis*. Lawrence Erlbaum Associates Publishers, 1998.

[12] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[14] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

[15] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[16] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

[17] C. Stark, B. J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. Biogrid: A general repository for interaction datasets. *Nucleic Acids Research*, 34:535–539, 2006.

[18] B. Weisfeiler and A. Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.