Joint sparsity-biased variational graph autoencoders

Journal of Defense Modeling and Simulation
18(X):1–6

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Lane Lawley[1,2] and Will Frey[2] and Patrick Mullen[2] and Alexander D. Wissner-Gross[3,4]

Abstract

To bring the full benefits of machine learning to defense modeling and simulation, it is essential to first learn useful representations for sparse graphs consisting of both key entities (vertices) and their relationships (edges). Here we present a new model, the Joint Sparsity-Biased Variational Graph AutoEncoder (JSBVGAE), capable of learning embedded representations of nodes from which both sparse network topologies and node features can be jointly and accurately reconstructed. We show that our model outperforms the previous state of the art on standard link-prediction and node-classification tasks and achieves significantly higher whole-network reconstruction accuracy, while reducing the number of trained parameters.

Keywords

Machine learning, variational autoencoders, graph neural networks, representation learning

Introduction

Conventional defense, in which kinetic effects tend to be localized near forces and capabilities, is three-dimensional. In contrast, all-domain defense is intrinsically high-dimensional in the sense that flipping a bit on one side of the planet can affect mission outcomes on the other, and thus challenging for humans to plan, direct, monitor, and assess. Therefore, optimally effective and efficient defense must leverage modern machine-learning techniques to analyze and reduce high-dimensional spaces to lower-dimensional representations that human commanders, planners, operators, and analysts can rapidly understand and manage. The entities and relations that comprise such high-dimensional spaces, with many features, as well as multi-modal spaces, with multiple types of data, can be naturally represented by graphs of varying density, with nodes representing actors, assets, and targets, and links representing effects and relationships. For example, militarily relevant graphs commonly include social and sensor networks. Graph embedding techniques are often used to make such networks available to downstream machine learning tasks[1,2]. Motivated by the insight that such networks are of varying density[3], here we introduce a new model for learning embedded representations of sparse graph vertices and edges that can enable key entity and relationship information to be learned jointly.

Related Work

A popular approach for unsupervised learning of graph embeddings is the original variational autoencoder model for graphs proposed by Kipf and Welling[4]. The authors introduced a spectral graph convolution (GC) layer that combined vertex feature information with graph topology. After generating vertex embeddings with stacked GC layers, they reconstructed the graph topology using a unary matrix inner product and passing the result through a sigmoidal activation function. However, as we show in this paper, this decoder model, which was only evaluated on a balanced link-prediction task, does not generalize well to sparse graphs.

Tran[5] introduced LoNGAE, the local neighborhood graph encoder—and αLoNGAE, with a feature-augmented adjacency matrix—with the goal of performing well on both link-prediction and node-classification tasks. This model used a symmetric encoder and decoder with shared parameters to learn node embeddings based on their local network neighborhoods. Xie et al.[6] generalized this multi-task-oriented approach to graph representation learning with the multi-task network representation learning (MTNRL) framework, meant to allow graph representation models to train so as to optimize performance an arbitrary set of downstream tasks. They implemented this framework on multi-task graph attention networks (MT-GATs), optimized to perform link prediction and clustering.

Lerique et al.[7] introduced the Attributed Network to Vector (AN2VEC) model, capable of decoding both node features and network topology from the same embeddings using two decoder heads: a weighted bilinear decoder for topology and a two-layer feed-forward neural network for node features. The topological and node embeddings were allowed to share some parameters and keep some parameters independent in the final embeddings; this ratio was controllable before training. Although our model uses a similar dual-headed decoder, it has two key differences:

[1]University of Rochester
[2]Two Six Labs
[3]Gemedy
[4]Harvard University

Corresponding author:
Lane Lawley, University of Rochester, Department of Computer Science
Email: llawley@cs.rochester.edu

first, instead of a weighted bilinear adjacency matrix decoder, we use a simpler inner-product decoder with an additive bias term; and second, we do not feature-engineer the allocation of any parameters in the node embeddings to solely represent topology or node features.

Recent work[5,6,8] has demonstrated the power of semi-supervised learning, wherein embedded representations are first pre-trained through unsupervised learning tasks such as autoencoding, and then later applied to or fine-tuned on narrower tasks (e.g,. link prediction or node classification) that may not be known a priori. Motivated by that goal, especially in the context of defense modeling and simulation where inference may take place at a higher classification than training, we focus here on accurately reconstructing autoencodings as a universal objective.

More broadly, in contrast to these previous studies, we expand our *evaluation* of graph autoencoder performance beyond the standard link-prediction and node-classification tasks due to their inability to generalize to sparse networks and decode node features. For example, the standard link-prediction task introduced by Kipf and Welling[4] uses an artificially balanced evaluation set consisting of equal numbers of positive and negative edges, which obscures poor performance on sparse graphs. Furthermore, none of the related studies we have mentioned evaluates whole-graph topological errors, motivating us to evaluate reconstruction of every edge, and every node feature vector, in the graph.

Model Architecture

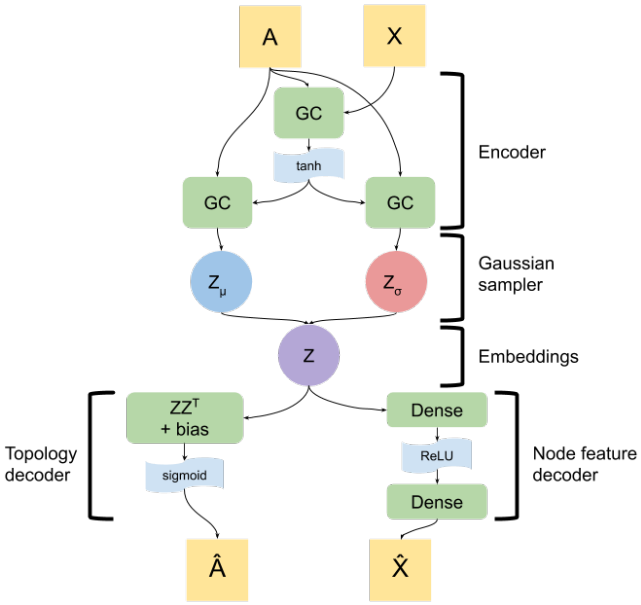

Figure 1. The architecture of our auto-encoder. Green blocks represent operations with trained parameters. $\mathbf{X}$ refers to the node features, $\mathbf{A}$ refers to the adjacency matrix, and $\hat{\mathbf{X}}$ and $\hat{\mathbf{A}}$ refer to their respective decoded counterparts. $\mathbf{Z}_\mu$ and $\mathbf{Z}_\sigma$ refer to the parameters of the Gaussian distributions from which the embeddings, $\mathbf{Z}$, are sampled at each training iteration.

Our model is derived from the original VGAE introduced by Kipf and Welling[4], with several key changes. The foremost architectural difference is our model's second decoder head, which allows reconstruction of the original node features, in addition to the adjacency-matrix-reconstruction head. We also reformulated the adjacency matrix's inner-product decoder to include a trainable bias term, to improve sparse graph reconstruction performance.

Encoder

As in the original VGAE paper, our encoder uses a single GC layer to produce a hidden representation, which is then shared by two independent GCs responsible for producing the means and standard deviations for the Gaussian distributions from which the node embeddings are sampled.

We also employ a refinement proposed by Kipf and Welling in a follow-up paper[8], where the adjacency matrix $\mathbf{A}$ is first transformed into $\tilde{\mathbf{A}} = \mathbf{A} + \lambda\mathbf{I}$, where λ is a hyperparameter, effectively weighting self-connections higher than neighbor connections in the graph convolution's Laplacian smoothing.

Topology Decoder Head

As shown in Figure 1, our model encodes an adjacency matrix $\mathbf{A}$ and a matrix of node feature vectors $\mathbf{X}$ into embeddings $\mathbf{Z}$, and then decodes $\mathbf{Z}$ along two independent paths. The first path uses pairwise vector inner products—which include implicit cosine-similarity terms—between the node embeddings to predict links in the original network.

Although Kipf and Welling[4] originally defined the adjacency matrix edge probabilities as $\sigma(\mathbf{Z}\mathbf{Z}^T)$, we observed that the inner product $\mathbf{Z}\mathbf{Z}^T$ contains few values less than zero, resulting in edge probabilities largely above 0.5 after applying the sigmoid function. This is perhaps unsurprising given that if $\mathbf{Z}$ is invertible then $\mathbf{Z}\mathbf{Z}^T$ is positive-definite. As a consequence, the decoded adjacency matrices are dense. We addressed this by adding a trainable bias term b to the inner-product decoder: $\sigma(\mathbf{Z}\mathbf{Z}^T + b)$. This term allows the topology decoder head to re-center the inner product before normalizing it to reconstruct the level of sparseness of the input graph, effectively tuning the average element magnitude of the reconstructed adjacency matrix.

Node Feature Decoder Head

We also augment the original Kipf and Welling model with a second decoder head, trained to reconstruct the original node features $\mathbf{X}$ from the embeddings $\mathbf{Z}$.

In choosing an architecture for the node feature decoder head, we observed that a simple feed-forward layer with a sigmoid activation is a special case of the symmetrical adjacency matrix graph convolution rule given by Kipf and Welling[8]: $\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{D}}^{\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$, where $\hat{\mathbf{A}} = \mathbf{A} + I$, $\hat{\mathbf{D}}$ is the diagonal node degree matrix of $\hat{\mathbf{A}}$, and $\mathbf{W}^{(l)}$ and $\mathbf{H}^{(l)}$ are the weights and inputs, respectively, of the l-th hidden layer of the graph convolutional network. If $\hat{\mathbf{A}}$ is the identity matrix, i.e. no nodes neighbor any other nodes, then its symmetrically normalized Laplacian, $\hat{\mathbf{D}}^{\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}$, simplifies to the identity matrix $\mathbf{I}$, and the propagation rule simplifies to $\mathbf{H}^{(l+1)} = \sigma(\mathbf{H}^{(l)}\mathbf{W}^{(l)})$, which is the propagation rule for a simple feed-forward neural network.

Interpreted this way, our architecture's two-layer, feed-forward node feature decoder renders the end-to-end feature reconstruction pathway of our model symmetrical: two

encoder GC layers convolve the input features with $\mathbf{A}$, and two decoder GC layers convolve the hidden features with the identity matrix before decoding them into their original form. By ignoring topology the decoder GC layers, we can avoid calculating symmetrically normalized Laplacians and accelerate training.

Training

Objective

As our model is a variational autoencoder, it assumes that the adjacency matrix $\mathbf{A}$ and the node feature matrix $\mathbf{X}$ are sampled from a probability distribution $p(\mathbf{A}, \mathbf{X}|\mathbf{Z})$, conditioned on latent variables $\mathbf{Z}$. The model's decoder defines the factorized generative model $p(\mathbf{A}, \mathbf{X}|\mathbf{Z}) = p(\mathbf{A}|\mathbf{Z})p(\mathbf{X}|\mathbf{Z})$, assuming that $\mathbf{A}$ and $\mathbf{X}$ are conditionally independent given the latent variables $\mathbf{Z}$. The model's encoder defines $q(\mathbf{Z}|\mathbf{A}, \mathbf{X})$, which is trained to approximate the true posterior distribution $p(\mathbf{Z}|\mathbf{A}, \mathbf{X})$. We assume an isotropic Gaussian prior $p(\mathbf{Z}) = \prod_{i=1}^{|\mathbf{Z}|} \mathcal{N}(\mathbf{z_i}|\mathbf{0}, \mathbf{I})$

We train our model using full-batch gradient descent to maximize the variational lower bound

$$\mathcal{L} = \underbrace{\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})]}_{\text{Topology reconstruction loss}} + \underbrace{\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{X}|\mathbf{Z})]}_{\text{Node feature reconstruction loss}} - \underbrace{\beta D_{KL}[q(\mathbf{Z}|\mathbf{X}, \mathbf{A})||p(\mathbf{Z})]}_{\text{Weighted KL divergence regularizer}} \quad (1)$$

where

$$p(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(\mathbf{A}_{i,j}|\mathbf{z_i}, \mathbf{z_j}) \quad \text{with } p(A_{i,j} = 1|\mathbf{Z_i}, \mathbf{Z_j}) = \sigma(\mathbf{Z_i^T Z_j} + b) \quad (2)$$

and $p(\mathbf{X}|\mathbf{Z})$ is defined by the two-layer neural network in the decoder, shown in Figure 1.

The topology reconstruction loss term in Equation 1 is defined as the binary cross-entropy loss between the reconstructed adjacency matrix $\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^\mathbf{T} + b)$ and the input adjacency matrix $\mathbf{A}$. The node feature reconstruction loss term is defined as the mean squared error between the reconstructed node features $\hat{\mathbf{X}}$ and the input node features $\mathbf{X}$.

Datasets

To empirically investigate our technique's suitability to defense, which frequently centers on globally sparse, multi-topic, information-sharing networks[3], we selected three well-studied citation-network datasets as open-source proxies for analysis.

We performed experiments on our model trained on each of the CORA[9], Citeseer[10], and Pubmed[11] datasets, where nodes represent papers and links represent citations. CORA and Citeseer nodes have associated feature vectors consisting of binary, multi-hot encodings indicating the presence or absence of vocabulary words in paper titles. The Pubmed node feature vectors are TF/IDF-weighted word frequencies for the entire document. The CORA dataset contains 2,708 nodes with 5,429 links, the Citeseer dataset contains 3,312 nodes with 4,732 links, and the Pubmed dataset contains 19,717 nodes with 44,338 links.

The three standard citation networks on which we evaluated our model are representative of a large variety of other militarily relevant attributed graphs, whose node features are also encoded as sparse binary vectors, including user attributes[12] and natural language text[13] in social media networks, and sensor output data in sensor networks[14,15].

We performed link-prediction and node-classification experiments for each dataset, and evaluated reconstruction of the node feature vectors and network topology for the CORA dataset. We also include a discussion of observed useful properties of the learned latent space for the CORA dataset.

Preprocessing

For training, we used the versions of these datasets provided by the Deep Graph Library[16] (DGL). For numerical stability, before training, we normalize the graph's node features $\mathbf{X}$ to be distributed with a mean of 0 and a variance of 1. Apart from this normalization, we made no other changes to the DGL datasets.

Hyperparameters

We used a hidden dimension of 1,024 in our encoder graph convolutions and the two-layer feature decoder head, and an embedding dimension of 512 for $\mathbf{Z}$ on the CORA dataset.* Our KL divergence weight term, β, was set to 0.4. We performed a hyperparameter search for the optimal value of the encoder's $\lambda \in \{2, 4, 8, 16, 32\}$, which yielded a value of $\lambda = 8$ in all three encoder graph convolutions that minimized node feature reconstruction loss. Using the ADAM optimizer and a learning rate of 0.001, we trained for 23,000 epochs.

Results

Link Prediction

We measured our autoencoder's performance on a standard link-prediction task with the CORA, Citeseer, and Pubmed citation network datasets. Using the same protocol as Kipf and Welling[4], we trained our model on 85% of the edges, and evaluated link prediction using a validation set of 10% of held-out edges and a test set of the remaining 5%. We report the results in Table 1. Our model outperforms the others we tested on the CORA and Citeseer datasets, and performs competitively on the Pubmed dataset. However, we believe that this task is an incomplete measure of graph reconstruction ability, as we discuss in the next section.

Node Classification

To more fully illustrate our model's applicability to standard downstream tasks, we evaluated the ability of a shallow classifier to classify our model's node embeddings into labeled document classes, which were not provided during the initial training process. We used a one-versus-rest SVM

*For the Citeseer dataset, we used the same dimensions. For the larger and denser Pubmed dataset, we used a hidden dimension of 2,048 and an embedding dimension of 1,024.

Table 1. Citation dataset link-prediction performance, both area under the ROC curve (AUC) and average precision score (AP). The best score of each column is bolded.

	CORA		Citeseer		Pubmed	
Model	AUC	AP	AUC	AP	AUC	AP
GAE	0.910	0.920	0.895	0.899	0.964	0.965
VGAE	0.914	0.926	0.908	0.920	0.944	0.947
LoNGAE	0.896	0.915	0.860	0.892	0.926	0.930
αLoNGAE	0.943	0.952	0.956	0.964	0.960	0.963
MT-GAT	0.930	0.963	0.931	0.963	**0.968**	0.970
AN2VEC	0.930	0.935	0.949	0.951	0.931	0.931
JSBVGAE *(ours)*	**0.984**	**0.977**	**0.972**	**0.971**	**0.968**	**0.974**

classifier architecture, with a radial basis function kernel and a misclassification cost of $C = 32$. We trained the classifier on 90 percent of the nodes from links held out of the initial training process and validated it on the remaining 10 percent.

The validation accuracy results are reported in Table 2. While our model outperformed all other evaluated models on node classification in all three datasets, this result is perhaps unsurprising given our comparatively high node embedding dimensions of 512 or 1024, which we selected for accurate node feature reconstruction.

Table 2. Citation dataset node classification accuracy. The best score of each column is bolded.

Model	CORA	Citeseer	Pubmed
αLoNGAE	0.783	0.716	0.794
GCN [8]	0.815	0.703	0.790
Planetoid [17]	0.757	0.647	0.772
ICA [18]	0.751	0.691	0.739
DeepWalk [19]	0.672	0.432	0.653
JSBVGAE *(ours)*	**0.838**	**0.736**	**0.812**

Topology Reconstruction

As Kipf and Welling [4] describe the link-prediction task now standard in graph autoencoding literature, there are an equal number of positive and negative links evaluated to produce the final score. Given that most graphs have unbalanced adjacency matrices, we believe that a more representative measure of adjacency matrix autoencoding accuracy is the F1 score of the full reconstructed adjacency matrix.

Although the model presented by Kipf and Welling [4] achieved an Area under the ROC Curve (AUC) of 0.914 and an average precision of 0.926 on the CORA dataset, their reconstruction (Figure 2c) had three orders of magnitude more positive edges than the original graph (Figure 2a), and its F1 score on full CORA adjacency matrix reconstruction was only 0.007, while our model achieved an F1 score of 0.851 (Figure 2b).

Node Feature Reconstruction

CORA node features are 1,433-dimensional, binary-valued, multi-hot word vectors in which each dimension indicates the presence or absence of the corresponding vocabulary word in the document the node represents. Each feature vector is divided by the sum of its positive dimensions in the original dataset. Due to this, and to our own pre-training node feature normalization, even perfectly reconstructed features are not directly interpretable as a binary vector.

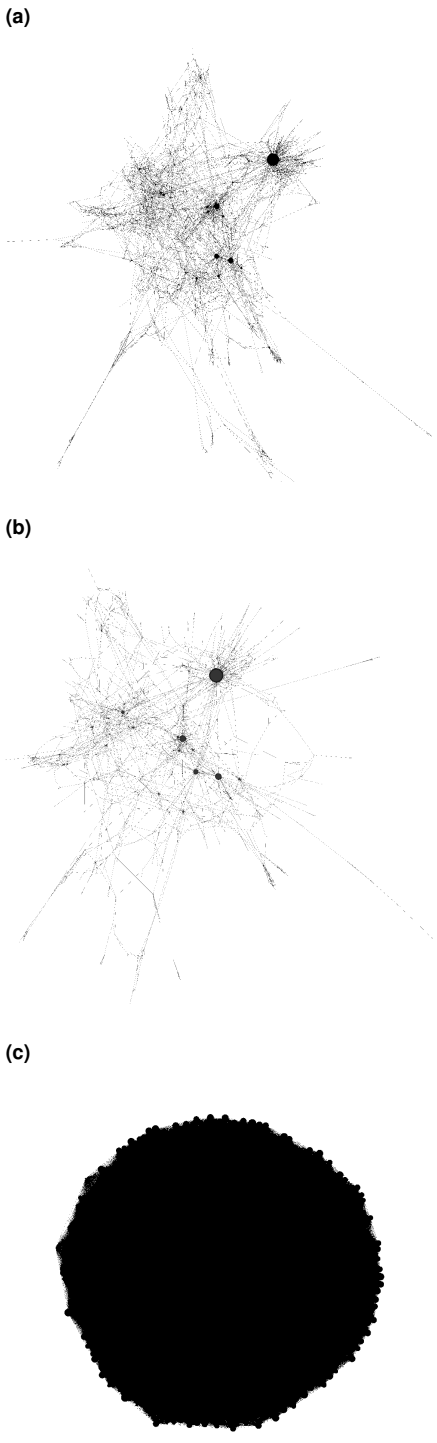

Figure 2. Comparison between original CORA citation network topology (2a), our topological reconstruction (2b), and the Kipf autoencoder [4] reconstruction (2c). All three networks are visualized with the ForceAtlas2 algorithm [20], with node radius scaled by degree. By visual inspection, our model can reconstruct the sparse citation network, including its high-degree hub nodes, while the Kipf reconstruction is sufficiently dense that the original topology cannot be easily discerned.

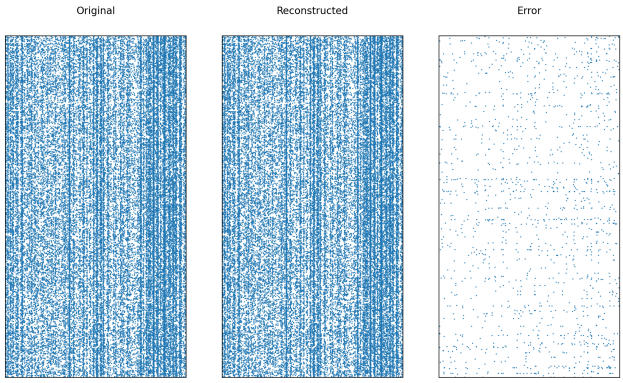

Figure 3. CORA's feature matrix, our reconstruction, and the error. Each row represents a document as a binary vector indicating the presence of each of 1,433 words.

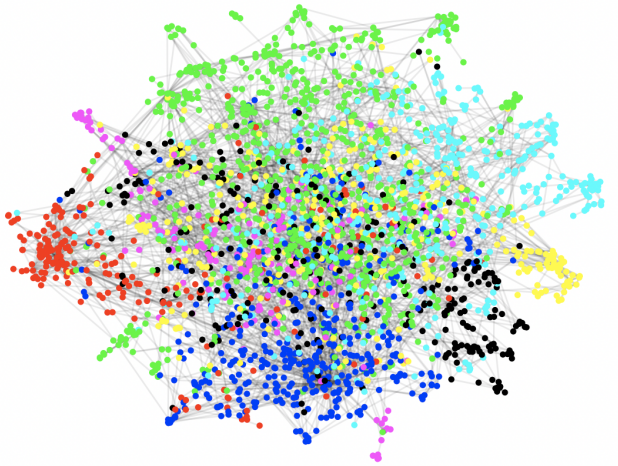

Figure 4. A visualization of our encoder's latent representation of CORA, constructed using the Uniform Manifold Approximation and Projection algorithm (UMAP)[21]. Colors indicate canonical document class (not provided during training). Gray lines indicate edges in the graph. The learned latent space has clustered similar document types without knowledge of those labels.

To convert the real-valued output, $\hat{\mathbf{X}}$, of the node feature decoder head into a binary vector, we clustered each dimension of $\hat{\mathbf{X}}$ according to a k-means analysis with $k = 2$. The dimensions in the larger-valued cluster are interpreted as 1, and the rest as 0.

Our autoencoder was able to recreate the 1,433-dimensional CORA node features with an average F1 score of 0.986 per record, and 0.984 across the concatenated matrix of all records. The latter reconstruction, along with the error matrix, is visualized in Figure 3.

Latent Space Properties

Our biased inner-product decoder appears to avert the issue observed by Kipf and Welling[4] wherein embeddings were pushed away from the origin. As seen in Figure 4, our latent space is loosely clustered by document type, with significant variance in two dimensions owing to the projection from a 512-dimensional space, many of which dimensions are used to encode node features.

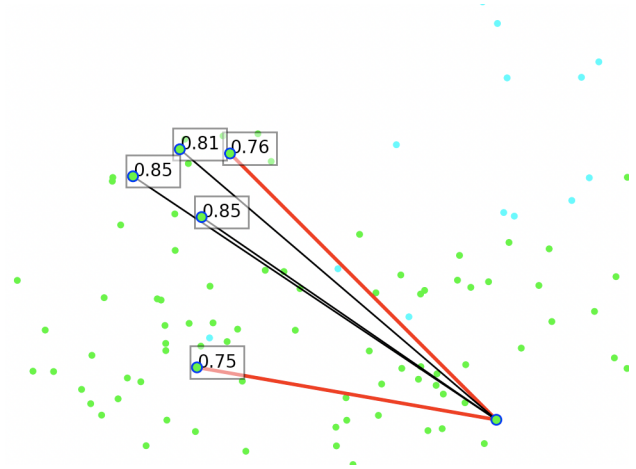

Figure 5. Lines showing the five nearest neighbors of an embedded document in the CORA latent space. Red lines indicate that the nearest neighbor is also connected by an edge in the graph. The lowest scores tend to go to topological neighbors, with the remaining nearest neighbors being unconnected documents of the same class.

As exemplified in Figure 5, the nearest neighbors of a node in embedding space are trained to be the vertices it is connected to in the input graph, but the next few nearest neighbors tend to be documents that are of the same class, but not directly connected by an edge. This hierarchical property of the distance metric, which captures both topological and feature similarity but weights the former higher, may reflect the known tendency of link data in citation networks to cluster more tightly than text data[22].

Discussion

Our model, although promising, has a number of limitations that we will now review, together with potential model generalizations that address them. First, due to its scalar bias term, our model's network topology decoder implicitly favors globally consistent sparsity. By incorporating an additive bias matrix, we could generalize our tunable sparsity to accommodate localized density variations. Second, although our model's variationality makes it a generative graph model, the naive generative method of sampling and decoding latent variables from unit Gaussians did not perform well for our model. To generalize our model by taking advantage of its theoretical generative capabilities, our model could be trained on more graphs in an inductive setting. Third, our model assumes that pre-trained embeddings of node features are provided as input, rather than raw features. By extending our model to also train node feature encoders, we could apply our model directly to raw node data and learn optimal feature embeddings for reconstruction. Fourth, while our model's nodes are attributed (i.e., featureful), the links only encode one bit of information. In the future, our model could be generalized to encode and reconstruct arbitrary link features in a similar way. Finally, our model does not yet apply to dynamic graphs that change over time. By performing stochastic gradient descent in an online- or lifelong-learning setting, we could potentially update our model continuously as graph mutations stream in.

More broadly, our results suggest a number of avenues for followup work. Because of our learned latent space's ability to hierarchically encode multiple distinct distance metrics, future work with our model might investigate the reconstruction of heterogeneous graphs with multiple distinct node and link types. Additionally, the performance of our model on other well-studied attributed graphs[23, 24], as well as on smaller networks, should be examined. Finally, the potential of our model to extrapolate large, unseen regions of partially observable graphs warrants further study.

Conclusion

We have presented a novel graph autoencoder architecture that: (1) can accurately reconstruct the node features and link structures of the entire original networks, including held-out links from partially observed networks; (2) learns hierarchical node embeddings in which directly connected nodes are nearest neighbors, and nodes with similar features are next-nearest neighbors; and (3) achieves state-of-the-art performance on existing benchmarks for standard datasets. In particular, our architecture's ability to represent both direct network links and node-feature-based similarities in a single vector space enables it to reveal important patterns in high-dimensional spaces. In the context of defense, this ability promises to enable commanders, planners, operators, and analysts to more rapidly classify featureful entities and sparse relationships in tactical, operational, and strategic graphs for unsupervised detection of anomalies and similarities, and semi-supervised estimation of node criticality, accessibility, recuperability, vulnerability, effect, and recognizability[25].

References

1. Parno B, Perrig A and Gligor V. Distributed detection of node replication attacks in sensor networks. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, 2005. IEEE, pp. 49–63.
2. Yang Z, Zhang Y and Dai Y. Defending against social network sybils with interaction graph embedding. In *2018 IEEE Conference on Communications and Network Security (CNS)*, 2018. IEEE, pp. 1–9.
3. Kott A, Swami A and West BJ. The internet of battle things. *Computer* 2016; 49(12): 70–75.
4. Kipf TN and Welling M. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*, 2016.
5. Tran PV. Learning to make predictions on graphs with autoencoders. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018. IEEE, pp. 237–245.
6. Xie Y, Jin P, Gong M et al. Multi-task network representation learning. *Frontiers in Neuroscience* 2020; 14.
7. Lerique S, Abitbol JL and Karsai M. Joint embedding of structure and features via graph convolutional networks. *Applied Network Science* 2020; 5(1): 1–24.
8. Kipf TN and Welling M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017.
9. Sen P, Namata G, Bilgic M et al. Collective classification in network data. *AI magazine* 2008; 29(3): 93–93.
10. Getoor L. Link-based classification. In *Advanced methods for knowledge discovery from complex data*. Springer, 2005. pp. 189–207.
11. Namata G, London B, Getoor L et al. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, 2012.
12. Tian Y, Niu Y, Yan J et al. Inferring private attributes based on graph convolutional neural network in social networks. In *2019 International Conference on Networking and Network Applications (NaNA)*, 2019. IEEE, pp. 186–190.
13. Ding X, Liu T, Duan J et al. Mining user consumption intention from social media using domain adaptive convolutional neural network. In *AAAI*, volume 15, 2015. pp. 2389–2395.
14. Medrano J and Lin FJ. Enabling machine learning across heterogeneous sensor networks with graph autoencoders. In *European Conference on Ambient Intelligence*, 2019. Springer, pp. 153–169.
15. Cartwright M, Cramer J, Salamon J et al. Tricycle: Audio representation learning from sensor network data using self-supervision. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019. IEEE, pp. 278–282.
16. Wang M, Yu L, Zheng D et al. Deep graph library: Towards efficient and scalable deep learning on graphs. *arXiv preprint* 2019; (arXiv:1909.01315).
17. Yang Z, Cohen W and Salakhudinov R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 2016. PMLR, pp. 40–48.
18. Lu Q and Getoor L. Link-based classification. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, 2003. ICML'03, AAAI Press. ISBN 1577351894, p. 496–503.
19. Perozzi B, Al-Rfou R and Skiena S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014. KDD '14, New York, NY, USA: Association for Computing Machinery. ISBN 9781450329569, p. 701–710. DOI:10.1145/2623330.2623732.
20. Jacomy M, Venturini T, Heymann S et al. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one* 2014; 9(6): e98679.
21. McInnes L, Healy J, Saul N et al. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software* 2018; 3(29): 861. DOI:10.21105/joss.00861.
22. Xu Z and Ke Y. Effective and efficient spectral clustering on text and link data. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016. CIKM '16, New York, NY, USA: Association for Computing Machinery. ISBN 9781450340731, p. 357–366. DOI:10.1145/2983323.2983708.
23. Chunaev P. Community detection in node-attributed social networks: a survey. *Computer Science Review* 2020; 37: 100286.
24. Hamilton WL. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning* 2020; 14(3): 1–159.
25. Bennett BT. *Understanding, assessing, and responding to terrorism: Protecting critical infrastructure and personnel*. John Wiley & Sons, 2018.