# Variational Graph Auto-Encoders

**Thomas N. Kipf**
University of Amsterdam
T.N.Kipf@uva.nl

**Max Welling**
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

## 1 A latent variable model for graph-structured data

We introduce the *variational graph auto-encoder* (VGAE), a framework for unsupervised learning on graph-structured data based on the *variational auto-encoder* (VAE) [2, 3]. This model makes use of latent variables and is capable of learning interpretable latent representations for undirected graphs (see Figure 1).

We demonstrate this model using a *graph convolutional network* (GCN) [4] encoder and a simple *inner product* decoder. Our model achieves competitive results on a link prediction task in citation networks. In contrast to most existing models for unsupervised learning on graph-structured data and link prediction [5, 6, 7, 8], our model can naturally incorporate node features, which significantly improves predictive performance on a number of benchmark datasets.

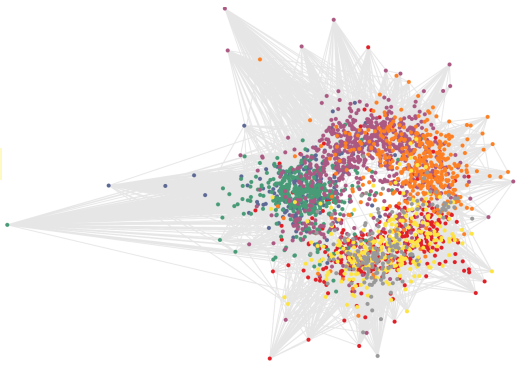

Figure 1: Latent space of unsupervised VGAE model trained on Cora citation network dataset [1]. Grey lines denote citation links. Colors denote document class (not provided during training). Best viewed on screen.

**Definitions** We are given an undirected, unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes. We introduce an adjacency matrix $\mathbf{A}$ of $\mathcal{G}$ (we assume diagonal elements set to 1, i.e. every node is connected to itself) and its degree matrix $\mathbf{D}$. We further introduce stochastic latent variables $\mathbf{z}_i$, summarized in an $N \times F$ matrix $\mathbf{Z}$. Node features are summarized in an $N \times D$ matrix $\mathbf{X}$.

**Inference model** We take a simple inference model parameterized by a two-layer GCN:

$$q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A}) \,, \quad \text{with} \quad q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i \,|\, \boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2)) \,. \tag{1}$$

Here, $\boldsymbol{\mu} = \mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ is the matrix of mean vectors $\boldsymbol{\mu}_i$; similarly $\log \boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$. The two-layer GCN is defined as $\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\,\mathrm{ReLU}\big(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0\big)\mathbf{W}_1$, with weight matrices $\mathbf{W}_i$. $\mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ and $\mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$ share first-layer parameters $\mathbf{W}_0$. $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ and $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix.

**Generative model** Our generative model is given by an inner product between latent variables:

$$p\left(\mathbf{A} \,|\, \mathbf{Z}\right) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{ij} \,|\, \mathbf{z}_i, \mathbf{z}_j\right) \,, \quad \text{with} \quad p\left(A_{ij} = 1 \,|\, \mathbf{z}_i, \mathbf{z}_j\right) = \sigma(\mathbf{z}_i^{\top} \mathbf{z}_j) \,, \tag{2}$$

where $A_{ij}$ are the elements of $\mathbf{A}$ and $\sigma(\cdot)$ is the logistic sigmoid function.

**Learning** We optimize the variational lower bound $\mathcal{L}$ w.r.t. the variational parameters $\mathbf{W}_i$:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\big[\log p\left(\mathbf{A} \,|\, \mathbf{Z}\right)\big] - \mathrm{KL}\big[q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) \,||\, p(\mathbf{Z})\big] \,, \tag{3}$$

where $\mathrm{KL}[q(\cdot)||p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. We further take a Gaussian prior $p(\mathbf{Z}) = \prod_i p(\mathbf{z_i}) = \prod_i \mathcal{N}(\mathbf{z}_i \,|\, 0, \mathbf{I})$. For very sparse $\mathbf{A}$, it can be beneficial to re-weight terms with $A_{ij} = 1$ in $\mathcal{L}$ or alternatively sub-sample terms with $A_{ij} = 0$. We choose the former for the following experiments. We perform full-batch gradient descent and make use of the *reparameterization trick* [2] for training. For a featureless approach, we simply drop the dependence on $\mathbf{X}$ and replace $\mathbf{X}$ with the identity matrix in the GCN.

**Non-probabilistic *graph auto-encoder* (GAE) model** For a non-probabilistic variant of the VGAE model, we calculate embeddings $\mathbf{Z}$ and the reconstructed adjacency matrix $\hat{\mathbf{A}}$ as follows:

$$\hat{\mathbf{A}} = \sigma\big(\mathbf{Z}\mathbf{Z}^\top\big), \quad \text{with} \quad \mathbf{Z} = \mathrm{GCN}(\mathbf{X}, \mathbf{A}). \tag{4}$$

## 2 Experiments on link prediction

We demonstrate the ability of the VGAE and GAE models to learn meaningful latent embeddings on a link prediction task on several popular citation network datastets [1]. The models are trained on an incomplete version of these datasets where parts of the citation links (edges) have been removed, while all node features are kept. We form validation and test sets from previously removed edges and the same number of randomly sampled pairs of unconnected nodes (non-edges).

We compare models based on their ability to correctly classify edges and non-edges. The validation and test sets contain $5\%$ and $10\%$ of citation links, respectively. The validation set is used for optimization of hyperparameters. We compare against two popular baselines: *spectral clustering* (SC) [5] and *DeepWalk* (DW) [6]. Both SC and DW provide node embeddings $\mathbf{Z}$. We use Eq. 4 (left side) to calculate scores for elements of the reconstructed adjacency matrix. We omit recent variants of DW [7, 8] due to comparable performance. Both SC and DW do not support input features.

For VGAE and GAE, we initialize weights as described in [9]. We train for 200 iterations using Adam [10] with a learning rate of $0.01$. We use a 32-dim hidden layer and 16-dim latent variables in all experiments. For SC, we use the implementation from [11] with an embedding dimension of $128$. For DW, we use the implementation provided by the authors of [8] with standard settings used in their paper, i.e. embedding dimension of $128$, $10$ random walks of length $80$ per node and a context size of $10$, trained for a single epoch.

**Discussion** Results for the link prediction task in citation networks are summarized in Table 1. GAE* and VGAE* denote experiments without using input features, GAE and VGAE use input features. We report *area under the ROC curve* (AUC) and *average precision* (AP) scores for each model on the test set. Numbers show mean results and standard error for 10 runs with random initializations on fixed dataset splits.

Table 1: Link prediction task in citation networks. See [1] for dataset details.

| **Method** | **Cora** | | **Citeseer** | | **Pubmed** | |
|---|---|---|---|---|---|---|
| | AUC | AP | AUC | AP | AUC | AP |
| SC [5] | $84.6 \pm 0.01$ | $88.5 \pm 0.00$ | $80.5 \pm 0.01$ | $85.0 \pm 0.01$ | $84.2 \pm 0.02$ | $87.8 \pm 0.01$ |
| DW [6] | $83.1 \pm 0.01$ | $85.0 \pm 0.00$ | $80.5 \pm 0.02$ | $83.6 \pm 0.01$ | $84.4 \pm 0.00$ | $84.1 \pm 0.00$ |
| GAE* | $84.3 \pm 0.02$ | $88.1 \pm 0.01$ | $78.7 \pm 0.02$ | $84.1 \pm 0.02$ | $82.2 \pm 0.01$ | $87.4 \pm 0.00$ |
| VGAE* | $84.0 \pm 0.02$ | $87.7 \pm 0.01$ | $78.9 \pm 0.03$ | $84.1 \pm 0.02$ | $82.7 \pm 0.01$ | $87.5 \pm 0.01$ |
| GAE | $91.0 \pm 0.02$ | $92.0 \pm 0.03$ | $89.5 \pm 0.04$ | $89.9 \pm 0.05$ | $\mathbf{96.4} \pm 0.00$ | $\mathbf{96.5} \pm 0.00$ |
| VGAE | $\mathbf{91.4} \pm 0.01$ | $\mathbf{92.6} \pm 0.01$ | $\mathbf{90.8} \pm 0.02$ | $\mathbf{92.0} \pm 0.02$ | $94.4 \pm 0.02$ | $94.7 \pm 0.02$ |

Both VGAE and GAE achieve competitive results on the featureless task. Adding input features significantly improves predictive performance across datasets. A Gaussian prior is potentially a poor choice in combination with an inner product decoder, as the latter tries to push embeddings away from the zero-center (see Figure 1). Nevertheless, the VGAE model achieves higher predictive performance on both the Cora and the Citeseer dataset.

Future work will investigate better-suited prior distributions, more flexible generative models and the application of a stochastic gradient descent algorithm for improved scalability.

**Acknowledgments**

We would like to thank Christos Louizos, Mart van Baalen, Taco Cohen, Dave Herman, Pramod Sinha and Abdul-Saboor Sheikh for insightful discussions. This project was funded by SAP Innovation Center Network.

## References

[1] 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.

[2] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[3] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning (ICML)*, 2014.

[4] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[5] L. Tang and H. Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710. ACM, 2014.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. ACM, 2015.

[8] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[10] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.