Variational Graph Auto-Encoders

Kipf, T.N.; Welling, M.

Publication date
2016

Document Version
Final published version

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
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.

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

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

\[
q(Z | X, A) = \prod_{i=1}^{N} q(z_i | X, A), \quad \text{with} \quad q(z_i | X, A) = \mathcal{N}(z_i | \mu_i, \text{diag}(\sigma_i^2)).
\]

Here, \( \mu = \text{GCN}_\mu(X, A) \) is the matrix of mean vectors \( \mu_i \); similarly \( \log \sigma = \text{GCN}_\sigma(X, A) \). The two-layer GCN is defined as \( \text{GCN}(X, A) = \tilde{A} \text{ReLU}(\tilde{AX}W_0)W_1 \), with weight matrices \( W_0 \). \( \text{GCN}_\mu(X, A) \) and \( \text{GCN}_\sigma(X, A) \) share first-layer parameters \( W_0 \). \( \text{ReLU}(\cdot) = \max(0, \cdot) \) and \( \tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \) is the symmetrically normalized adjacency matrix.

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

\[
p(A | Z) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij} | z_i, z_j), \quad \text{with} \quad p(A_{ij} = 1 | z_i, z_j) = \sigma(z_i^T z_j),
\]

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

Learning  We optimize the variational lower bound \( \mathcal{L} \) w.r.t. the variational parameters \( W_1 \):

\[
\mathcal{L} = \mathbb{E}_{q(Z | X, A)} \left[ \log p(A | Z) \right] - \text{KL}[q(Z | X, A) \parallel p(Z)],
\]
we compare models based on their ability to correctly classify edges and non-edges. 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 \( Z \). We use Eq. 4 (left
side) to calculate scores for elements of the reconstructed adjacency matrix. We omit recent variants

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.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUC</td>
<td>AP</td>
<td>AUC</td>
</tr>
<tr>
<td>SC [5]</td>
<td>84.6 ± 0.01</td>
<td>88.5 ± 0.00</td>
<td>80.5 ± 0.01</td>
</tr>
<tr>
<td>DW [6]</td>
<td>83.1 ± 0.01</td>
<td>85.0 ± 0.00</td>
<td>80.5 ± 0.02</td>
</tr>
<tr>
<td>GAE*</td>
<td>84.3 ± 0.02</td>
<td>88.1 ± 0.01</td>
<td>78.7 ± 0.02</td>
</tr>
<tr>
<td>VGAE*</td>
<td>84.0 ± 0.02</td>
<td>87.7 ± 0.01</td>
<td>78.9 ± 0.03</td>
</tr>
<tr>
<td>GAE</td>
<td>91.0 ± 0.02</td>
<td>92.0 ± 0.03</td>
<td>89.5 ± 0.04</td>
</tr>
<tr>
<td>VGAE</td>
<td><strong>91.4 ± 0.01</strong></td>
<td><strong>92.6 ± 0.01</strong></td>
<td><strong>90.8 ± 0.02</strong></td>
</tr>
</tbody>
</table>

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


