International Journal of Computer Applications (0975 - 8887)
Volume - No.,

Learning the Decoder for Variational Graph Autoencoder

Advanced Machine Learning and Optimization Course by Prof. Passerini

Ali Akay [224414]
Trento University
MSc.Artificial Intelligence Systems
ali.akay@studenti.unitn.it

ABSTRACT

I wish to build an Variational Graph autoencoder(VGAE) model with adopted decoder part for Citeseer dataset with VGAE and graph autoencoder(GAE).First, I implement the VGAE architecture presented by Kipf and Welling [1] using inner product decoder in order to compare the results.Then I implement proposed solution which is multi layer perceptron for the reconstruction part.
8

1. INTRODUCTION

Many complex systems in real life, such as social networks,can be abstracted into graph-structured data for analysis. The analysis results can be leveraged to guide practical applications, such as link prediction and personalized recommendation. The variational graph autoencoder proposed by Kipf and Welling [1], which extended the variational autoencoder (VAE) from the field of Euclidean structure data (e.g., image) to that of non-Euclidean data (e.g., graph-structured data), is one of the commonly utilized methods for studying graph-structured data. It could capture the distribution of the samples through a two-layer graph convolutional network and reconstruct the graph structure through a decoder.VAE and VGAE versions have increased rapidly in recent years, considerably improving the performance of the variational (graph) autoencoder on traditional machine learning tasks such as link prediction.

2. DATASET

The CiteSeer dataset consists of 3312 scientific publications classified into one of six classes.

The papers were selected in a way such that in the final corpus every paper cites or is cited by at least one other paper. There are 3312 papers in the whole corpus.

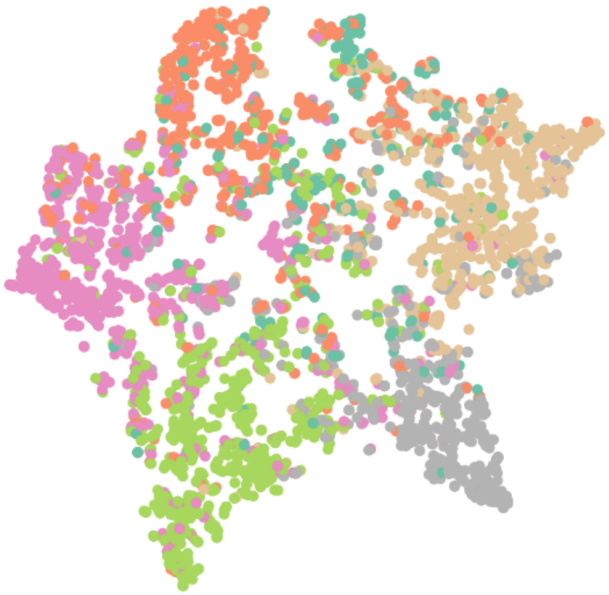

Fig. 1. Latent space of VGAE model trained on CiteSeer dataset.

After stemming and removing stopwords we were left with a vocabulary of size 3703 unique words. All words with document frequency less than 10 were removed.
Dataset contains 3327 nodes,9104 edges,3703 features and 6 classes.

3. BACKGROUND

An autoencoder is a type of artificial neural network used to learn data encodings in an unsupervised manner.
The aim of an autoencoder is to learn a lower-dimensional representation (encoding) for a higher-dimensional data, typically for dimensionality reduction, by training the network in order to understand the most important parts of the input.

Autoencoders consist of 3 parts: 1. Encoder: A module that compresses the input data into an encoded representation that is typically several orders of magnitude smaller than the input data. 2. Bottleneck: A module that contains the compressed knowledge representations and is therefore the most important part of the network. 3. Decoder: A module that helps the network"decompress" the knowledge representations and reconstructs the data back from its encoded form. The output is then compared with a ground truth. If the latent space is continuous, we call this a variational autoencoder.

While research into graphical deep learning has exponentialized, there remain challenges in autoencoding graphs, particularly in the decoding step. Specific to VAEs, it is challenging to take a discrete graphical structure, encode it into a continuous latent space, then accurately decode back to a discretized graphical structure.

Graph neurol network(GNN) uses deep learning methods to solve graph-related problems, which can be seen as an application of deep learning on graph data. One of the most popular methods of GNNs called Graph Convolutional Networks(GCN). GCNs originated with the work of Bruna et al. [2], which develops a version of graph convolutions based on spectral graph theory then generalize the convolution operation from Euclidean data to graph data by using the Fourier basis of a given graph.
Variational graph autoencoder, a framework for unsupervised learning on graph-structured data based on the variational auto-encoder. This model makes use of latent variables and is capable of learning interpretable latent representations for undirected graphs.

The encoder (inference model) of VGAE consists of GNNs.It is convolutional, because filter parameters are typically shared over all locations in the graph (or a subset thereof as in).
For these models, the goal is then to learn a function of signals/features on a graph G=(V, E) which takes as input:

—A feature description xi for every node i; summarized in a $N \times D$ feature matrix X (N: number of nodes, D: number of input features)

—A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix A (or some function thereof)

The first GCN layer generates a lower-dimensional feature matrix. It is defined as

$$\overline{X} = GCN(X, A) = ReLU(\tilde{A}XW_0) \tag{1}$$

$$\tilde{A} = D^{-1/2}AD^{-1/2} \tag{2}$$

A-tilde is the symmetrically normalized adjacency matrix.
The second GCN layer generates μ and logσ^2, where

$$\mu = GCN_\mu(X, A) = \tilde{A}\overline{X}W_1 \tag{3}$$

$$\sigma = GCN_\sigma(X, A) = \tilde{A}\overline{X}W_1 \tag{4}$$

Now if we combine the math of two-layer GCN together, we get the formula which generate μ and logσ^2

$$GCN(X, A) = \tilde{A}ReLU(\tilde{A}XW_0)W_1 \tag{5}$$

Then we can calculate Z using parameterization trick

$$Z = \mu + \sigma * \epsilon \tag{6}$$

The decoder (generative model) is defined by an inner product between latent variable Z. The output of our decoder is a reconstructed adjacency matrix A-hat, which is defined as

$$\hat{A} = \sigma(zz^T) \tag{7}$$

After the latent variable Z, learning the similarity of each row in the latent variable (because one row represents one vertex) to generate the output adjacency matrix. Inner product could calculate the cosine similarity of two vectors, which is useful when we want a distance measure that is invariant to the magnitude of the vectors. Therefore, by applying the inner product on the latent variable Z and Z^T, we can learn the similarity of each node inside Z to predict our adjacency matrix.

The loss function and learning for variational graph autoencoder is pretty much the same as before. The first part is the reconstruction loss between the input adjacency matrix and the reconstructed adjacency matrix. More specifically, it is the binary cross-entropy between the target (A) and output (A')logits. The second part is the KL-divergence between $q(Z|X, A)$ and $p(Z)$, where $p(Z) = N(0, 1)$. It measures how closely our $q(Z|X, A)$ matches to $p(Z)$.

$$L = E_{q(Z|X,A)}[\log(A|Z] - KL[q(Z|X, A)||p(Z)] \tag{8}$$

4. PROPOSED APPROACH FOR LINK PREDICTION

Link prediction, which is used to predict whether two nodes are connected in a graph, is widely applied in various fields, such as recommendation, knowledge graph, and social network. The traditional approaches mainly focus on graph topology.Topology-based methods compute node similarity score as the likelihood of connections. [3] [4] [5] Graph embedding approaches are currently popular to resolve link prediction problems.Representative methods such as DeepWalk [6], Node2vec [7] are trying to learn parameter-free low-dimensional node embeddings from the observed graph nodes and then use these embeddings to predict links.Recently,GNNs has shown impressive performance on link prediction.

4.1 Encoder

Encoder part of the project is the same with the paper of Kipf and Welling.

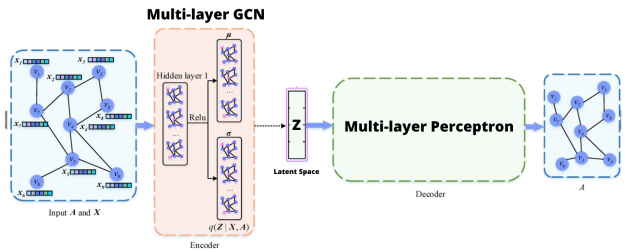

Fig. 2. VGAE + MLP Structure

There is one graph convolution layer and 2 output for mean and standard deviation as you can see in the [Fig. 2.]

4.2 Multi-Layer Perceptron Decoder

In the decoding step instead of using only an inner product decoder like Kipf and Welling, I use a multi-layer perceptron(MLP) with a sigmoid activation function outputting $\tilde{A}$. [Fig. 2.] MLP compose for 3 layers.Loss functions are the same with baseline models.The output is mapped by a MLP to a scalar indicating the probability of having an edge.

4.3 Sequence Models Decoder

The key idea of the approach is to represent graphs orderings as sequences, and then to build an autoregressive generative model on these sequences. [8]

In the theory,this approach does not suffer from some drawbacks. it can naturally generalize to graphs of varying size, and requires training on all possible node permutations or specifying a canonical permutation.But vector representation based models which represent G by flattening A into vector can not generalize.

I build a model with same encoder part of the paper then build a RNN+Linear layer for output.However,I couldn't find proper result for the sequence model for decoder.

4.4 Benchmark with State of the Art

First of all,I have done hyperparameter tuning for MLP decoder model with some parameters for 100 epoch. In [Fig. 3.] you can see that result of some models.In terms of the graph, the best model with parameters; LR:0.01,Optimizer:Adam with embedding size of 128.

Here in [Fig. 4.],you can see the graph of the dataset and in [Fig. 5.] you can see the reconstructed one. The output of MLP decoder produce more then 400000 link while it is 9104 edge link in the dataset.

I compare models of the paper and results of the project based on their ability to correctly classify edges and non-edges.The proposed model is compared in [Table 1.] with SEAL (Zhang and Chen 2018);variational graph auto-encoder (Kipf and Welling 2016b);graph auto-encoder (Kipf and Welling 2016b);adversarially regularized variational graph auto-encoder (ARVGA) (Pan et al. 2018); and adversarially regularized graph auto-encoder (ARGA) (Pan et al. 2018).

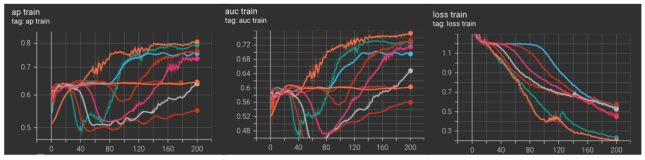

Fig. 3. Hyperparameter Tuning

Model	Auc	AP
SEAL	92.8	93.4
ARVGA	94.4	95.7
ARGA	93.5	95
GAE	89.5	89.9
VGAE	90.8	92
VGAE+MLP	0.74	0.8159

Table 1. Benchmark for CiteSeer Dataset

In particular, ARVGA and ARGA are modified from VGAE and GAE, respectively, by adding a discriminator that are trained adversarially.

The paper of Kipf and Welling, train for 200 iterations using Adam optimizer with a learning rate of 0.01. We use a 32-dim hidden layer and 16-dim latent variables in all experiments.

I compare the results the parameters that gives me the best results. You can see the results for the CiteSeer dataset.For performance evaluation,the area under the ROC curve (AUC) and average precision (AP) are used as in (Kipf and Welling).

5. CONCLUSION

In the end the project aim to invastigate the application for the decoder for VGAE.It can be seen that inner product still have good result when we compare the models for scalability and complexity.On the other hand,there has been better applications in order to improve the performance of the decoder using flexiable generative models and sequence models.

I did not see a significant result in outcomes by using MLP in the decoder. However, I would like to try this experiment again on VGAE with sequence models for decoder in order to get better results.

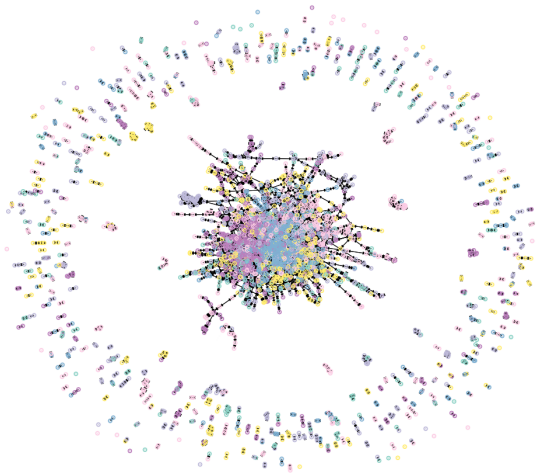

Fig. 4. CiteSeer Dataset

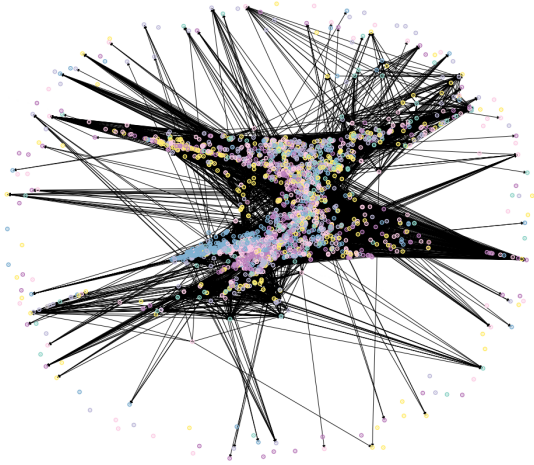

Fig. 5. MLP Decoder Reconstruction Output

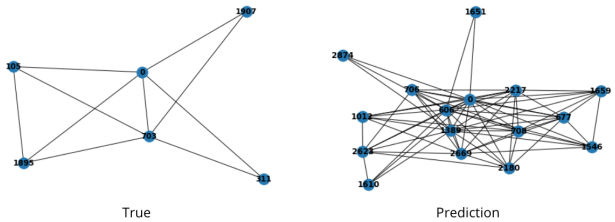

Fig. 6. Comparison for index 0

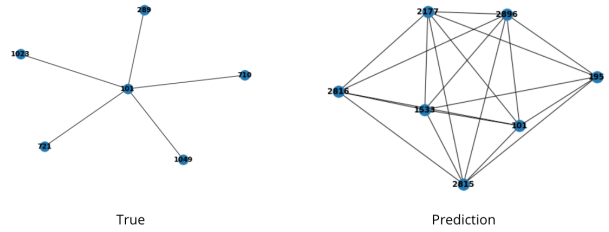

Fig. 7. Comparison for index 101

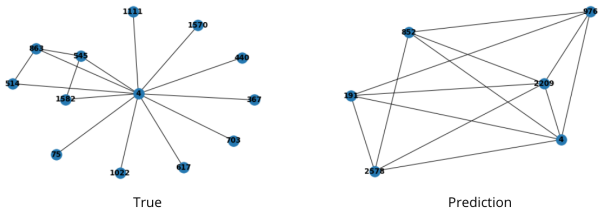

Fig. 8. Comparision for index 4

6. REFERENCES

[1] Thomas N. Kipf, Max Welling *Variational Graph Auto-Encoders.*

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. *Spectral networks and locally connected networks on graphs..*

[3] Gobinda G Chowdhury *Introduction to modern information retrieval. Facet publishing..*

[4] Leo Katz. *A new status index derived from sociometric analysis..*

[5] Lada A Adamic and Eytan Adar *Friends and neighbors on the Web. Social Networks.*

[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena *Deepwalk: Online learning of social representations.*

[7] Aditya Grover and Jure Leskovec. *node2vec: Scalable feature learning for networks.*

[8] Jiaxuan You *Graphrnn: Generating realistic graphs with deep auto-regressive model.*

[9] Simonovsky and Komadakis *graphVAE: Towards the Generation of Small Graphs Using Variational Autoencoders.*

[10] William L Hamilton, Rex Ying, and Jure Leskovec. *Inductive representation learning on large graphs*

[11] Baole Ai,Zhou Qin *Structure Enhanced Graph Neural Networks for Link Prediction.*

[12] Muhan Zhang,Yixin Chen *Link Prediction Based on Graph Neural Networks.*