Report Project GNN

Arthur Desbiaux and Valentin Cuzin-Rambaud

Abstract In this report, we explore several methods to make links prediction on graph. The Dataset used is an airport graph, the objective is to compare models. With an ablation study, this report, demonstrate that our custom model can perform link prediction on this dataset.

Introduction

The prediction link in a graph have several applications. In fact, a prediction link is a classification problem, where giving 2 nodes, we predict if there is a link between them or not. In our case, we have an airport graph of the world. It contains 3363 nodes and, 13547 edges. A node is defined by is longitude, latitude, population, country, city name. In the next part, we are going to explain our methodology for making models, test and tune them, after what we are comparing retained models. Finally, we're showing to you an ablation study of the best model from the comparison.

Methodology

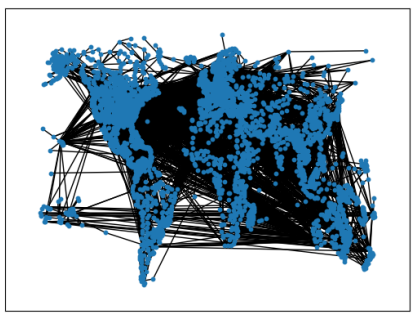
For training and evaluating models, we need to generate data from the graph. We question ourselves about whether using more information from nodes can make better classifications. So we are going to extract data from the graph in 2 ways: the first way is by using only latitude and longitude, and the second way is by adding population size information. The second question is: In which proportion we separate our data into 3 parts for training, validate and test ? We are going to use 0.05 of edge for validation and, for the test proportion, 0.1 and 0.2. When we are doing this proportion in test, it's come to the same as masking 0.1 or 0.2 of the edge. For training, we have deduced that most of the time we need 500 epochs for converging, so we fix this value.

We are interested in evaluating methods for comparing our models. We have selected AUC and AP metrics because they are largely used for link prediction problems. Furthermore, we have studied the visualization of an output of a decoder, but this kind of model predicts many edges, and even if we filter edges that are certain at 0.98% like in the figure1, we can't decide ourselves which limit is good, so it's not relevant.

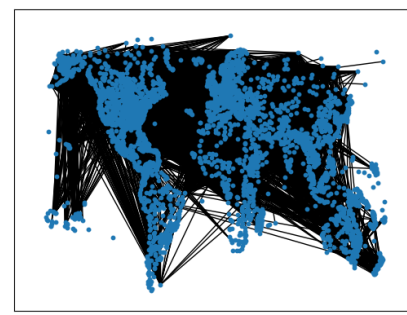
We have studied 3 kinds of models. The GAE with the GAE encoder, VGAE with the VGAE encoder and finally VGAE with the GAT encoder. About the implementation, we are in a transductive case because we have a whole graph during the training, we're just hiding some labels. According to the documentation, we cached the first computation on each layer, which can help.

For hyperparameter tuning, we have chosen to tune the learning rate and the output value of the encoder, because they have a great influence on the effectiveness of the model. We have also been tuning the number of neurons in the hidden layer of the GAT encoder. Here are the chosen values:

- learning rate in 0.01, 0.005, 0.001, 0.0001
- output in 5, 7, 9, 11, 13, 15
- hidden number in 60, 90 (only for GAT)



Input graph



Reconstruct the graph from the decoder of GAE, filter > 0.98

FIGURE 1. *Visualization of graph, and it's prediction of link by GAE*

Analyze the comparison

First we will a split of 85% train, 10% test, 5% validation

Also, we will group node attributes with the 'lon', 'lat' and 'population' we have in table 1 the best hyperparameters for each type of model. For the GAE, regardless of the hyperparameters, the results remain at 0.5 AUC and AP. The population information means that the model didn't learn from the data. The GAT encoder does not perform well for this task too, but with hyperparameters, we have seen little improvement. Only VGAE is relevant, so we detail results on the figure2.

TABLE 1. *Best models for each type, 10% of the test, with population*

Models	*hidden neurons*	*learning rate*	*output number*	*AUC*	*AP*
GAE Encoder	/	0.0005	9	0.5	0.5
GAT Encoder	90	0.001	11	0.562	0.533
VGAE Encoder	/	0.001	7	**0.726**	**0.647**

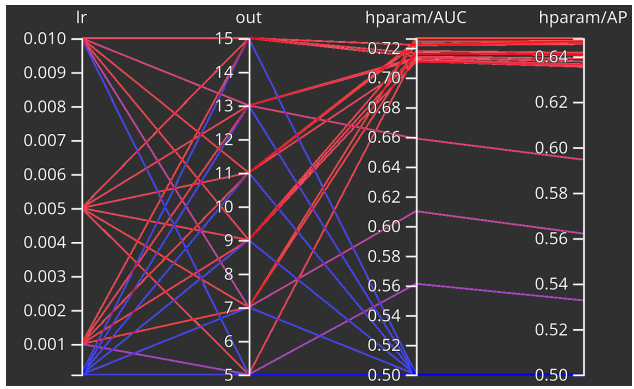


FIGURE 2. *VGAE with VGAE encoder, 10% of the test, with population*

As we can see, on the result of the table 1, the highest performance was achieved with VGAE, though the scores remain relatively modest (0.724 AUC, 0.649 AP). We deduce that grouping, using population information, biases the model classification. We will try to remove one of the node attributes to see if the results change. We will now group by 'lon' and 'lat'.

The table 2 compares all models, showing only the top 10 best AUC. It seems that GAE performs really well, with VGAE emerging as a strong competitor. High learning rate combine with high output result in good performances for GAE and VGAE. GAT isn't in the 10 best models but reach 0.899 AUC with a learning rate of 0.001, 11 out and 60 hidden neurons.

TABLE 2. *The bests model, 10% of test, without population*

Models	*learning rate*	*output number*	*AUC*	*AP*
GAE	0.005	11	0.948	0.941
GAE	0.001	15	0.948	0.933
VGAE	0.005	11	0.95	0.936
GAE	0.01	7	0.951	0.945
GAE	0.01	11	0.953	0.945
GAE	0.005	13	0.953	0.948
VGAE	0.01	15	0.953	0.948
GAE	0.01	13	0.955	0.948
GAE	0.005	15	0.955	**0.95**
GAE	0.01	15	**0.955**	0.949

Now we will try a split of 75% train, 20% test, 5% validation.

In this part, we will only test with the node attributes 'lon' and 'lat'. As we can see on the table 3, by reducing the train percentage, we got results that are similar. From 0.955 AUC with GAE to 0.951 AUC, we can also observe the same idea with the AP going from 0.949 to 0.941. In conclusion, we can't say that this decrease is caused by reducing the train part, it can be caused by the random weight.

TABLE 3. *Best models for each type, 20% of test, without population*

Models	*hidden neurons*	*learning rate*	*output number*	*AUC*	*AP*
GAE Encoder	/	0.01	7	**0.9510**	0.9363
GAT Encoder	90	0.001	13	0.8958	0.8694
VGAE Encoder	/	0.01	15	0.9483	**0.9415**

Ablation study

According to the previous basic comparison, we assume that we are going to create 8 models derivated from the best one, so derivate from GAE with a 0.01 learning rate, and 15 outputs. We introduce you to the GAE Extended encoder. GAEE contains 4 layers of GCNconv, with 2 relu activation, and after we apply a sigmoid for the third layer. You can see in table4 that the GAEE encoder outperforms the state of the art for this dataset.

TABLE 4. *Ablation Study, 10% of test, without population*

Models	*AUC*
Basic GAE	0.9457
Basic Inductive GAE	0.9372
GAE with only 1 layer	0.8320
GAE with 3 layers	0.9139
GAE with softmax	0.9625
GAE with sigmoid	0.9623
GAE with 3 layers, last with sigmoid	0.9675
GAE with 5 layers, last with sigmoid	0.9410
GAE with 4 layers, last with softmax	0.9378
GAEE	**0.9722**

We perform that analyze by running 5 times each models and computing the means, for by sure that the difference between two models is not responsible for the random initialization of weight. We have verified that Inductive methods are not revenant in our case. Less or More layers is useless, and using a soft max function isn't better than the sigmoid.

Conclusion

In this report, we explored various methods for link prediction on an airport graph dataset. We compared different models, including GAE, VGAE, and GAT encoders, and conducted a comparison study to evaluate their performance. Our findings indicate that the GAE model with specific hyperparameters (0.01 learning rate, 15 output) performs best, achieving high AUC and AP scores. Additional, using latitude and longitude as node attributes, without population information, yields the best results.

We conclude that our custom model, GAEE, which extends the GAE with additional layers and sigmoid activation functions on the third layer, outperforms the state-of-the-art models for this dataset. This study demonstrates the effectiveness of graph encoder in link prediction tasks and highlights the importance of careful hyperparameter tuning and model selection.