GATTO: Can Topological Information Improve Node Classification via GAT?

Final Report for Learning from Network's project

Francesco Biscaccia Carrara *(2120934)*, Riccardo Modolo *(2123750)*,
Alessandro Viespoli *(2120824)*

Master Degree in Computer Engineering
University of Padova

***Abstract*— In this report we study how topological features of a node can affect its prediction on GAT (Graph Attention Network). We introduce the dataset, the type of feature we computed and we compare our result with the original GAT paper.**

I. INTRODUCTION

Node classification is an important reasearch and business topic. Our framework GATTO (Graph ATtention network with TOpological information) want to improve the classic GAT prediction using node feature coming from the graph or from its embedding. The main idea is to using topological features to imporve prediction of GAT, where each node already have features in it.

II. DATA

We're going to use the same dataset of GAT[1], with features to each node. the Dataset of GAT are: *Cora and Citeseer*. Each node have only one label and the graph are both directed, unweighted and without self loop.

Network	Nodes	Edges	labels	features
Cora	2708	5429	7	1443
Citeseer	3327	4732	6	3703

The feature we intended to compute and assign to each node for all these graphs are:

- degree centrality
- betweenness centrality
- closeness centrality
- suggested label

The *suggested label* parameter is the result of a clustering made on the embedding of the Graph. We use Node2Vec[2] to produce the embedding, and for clustering we use k-mean++ method.

III. IMPLEMENTATION

The practical implementation[3] respect the nature of the the framework concept. We have two blocks:

- **Precomputation Module**: the class that compute every needed or requested features from the graph or from it's embedding, and return it as feature matrix
- **GAT Module**: the GAT implementation for train and predict node labels

A. Precomputational Module

For precomputational module, we have used the python packages: NetworkX[4] and Node2Vec[5]. Since the framwork is meant to be scalable after the phase of building feature, the script produce a file where computed features are stored. In this way GATTO can be used as normal GAT network if any feature computation are needed.

B. GAT Module

In this section we want to deep dive in how we setup the Network. We use the same configuration presented in the GAT paper for the **Transductive Learning**. Ntwork have two layer

GAT model, the first one, with $K = 8$ attention heads computing $F' = 8$ features each, followed by an ELU (exponential linear unit). The second layer is used for classification: a single attention head compute C features (C number of classes), followed by a softmax activation. We use a dropout of $p = 0.5$ for layers' input, as well as to the *normalized attention coefficients*. For the loss function we have used *cross-entropy loss*.

The data inserted inside GAT are binary value, for that reason we *binarize* our data before the GAT training. after the training we test the GAT returning the accuracy and the confusion matrix.

IV. TESTS

Since we want to compare the performance of GATTO with respect to GAT baseline algorithm, for each dataset, we perform 10 runs for each technique, to estimate the following parameter:

- **Accuracy**: The proportion of correctly predicted instances (both true positives and true negatives) out of the total instances.
- **Precision**: The proportion of true positive predictions out of all positive predictions. It measures exactness.
- **Recall**: The proportion of true positive predictions out of all actual positive cases. It measures completeness.
- **F1 Score**: The harmonic mean of precision and recall, balancing the two.

The algorithms have been executed on 2 different machines: *CAPRI High-Performance Computing* to perform the feature matrix extraction and a general-purpose computer with *Nvidia RTX 2060* to train GAT and analyze its performance.
To accomplish such comparison, we utilize a set of statistical tests:

1) Since all the scores are values in $[0,1]$, we utilize the Shapiro-Wilk Test to assert the normality of the scores.
2) If the test "passes", we perform:
   - Two-Sample t-Test, with the assumption that the variances are equal
   - Two-Sample t-Test, with the assumption that the variances are different
   - Wilcoxon Signed-Rank Test
3) If the test "fails", we only utilize the Wilcoxon Signed-Rank Test, since we have hard evidence that we are not in a normal scenario.

The described tests are implemented and available on Github repository as "Results.R" file in the section "code/results".

V. RESULTS

The following tables show the averages and the variances of each score, for each algorithm, based on the test described before.

	Accuracy	Precision	Recall	F1 Score
GAT	0.888	0.891	0.888	0.888
GATTO	0.890	0.893	0.890	0.890

TABLE I: Averages of the scores for GAT and GATTO (on Cora dataset)

	Accuracy	Precision	Recall	F1 Score
GAT	2.566e−5	2.693e−5	2.566e−5	2.564e−5
GATTO	1.823e−5	1.611e−5	1.823e−5	1.837e−5

TABLE II: Variances of the scores for GAT and GATTO (on Cora dataset)

	Accuracy	Precision	Recall	F1 Score
GAT	0.737	0.732	0.737	0.731
GATTO	0.736	0.731	0.736	0.730

TABLE III: Averages of the scores for GAT and GATTO (on CiteSeer dataset)

	Accuracy	Precision	Recall	F1 Score
GAT	2.580e−5	2.819e−5	2.580e−5	2.911e−5
GATTO	1.011e−5	9.231e−6	1.011e−5	9.936e−6

TABLE IV: Variances of the scores for GAT and GATTO (on CiteSeer dataset)

From a naive prospective, it seems that GATTO works a little bit better in terms of both average values and variance values. However, by running the statistical tests, as explained in the test section, we obtain the tables (link to table). Since we want to bound the Type-I error $\alpha \leq 0.05$ and the p-values returned by the tests, we can say that there is no statistically evidence that

the algorithm perform equally, at least on small dataset. This may be due to the fact that adding a small additive "piece of information" to the node (4 features over 1447 for Cora, 4 features over 3707 for CiteSeer) doesn't change the overall perfomance of the GAT network. By such considerations, we can get rid of Precomputation Module since it's computationally expensive and it doesn't provide an actual perfomance improve, even if, in absolute terms, it provides an overall $0.14 - 0.54\%$ boost.

VI. Future works

A lot of improvement can be done in future works. First of all changing or trying other embeddings or clustering to get the best accuracy for the suggested cluster feature (not analyze in this essay for time problem). After that parameters can be change and add or also the GAT parameters. Another point is to try this model for graphs without features, in this way we can use GAT for more general family of graphs.

References

[1] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. URL: https://arxiv.org/abs/1710.10903.

[2] Aditya Grover and Jure Leskovec. *node2vec: Scalable Feature Learning for Networks*. 2016. arXiv: 1607.00653 [cs.SI]. URL: https://arxiv.org/abs/1607.00653.

[3] Francesco Biscaccia Carrara Alessandro Viespoli Riccardo Modolo. *GATTO: GitHub Implementation*. URL: https://github.com/RickSrick/GATTO.

[4] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[5] Aditya Grover and Jure Leskovec. *node2vec: GitHub Implementation*. 2016. URL: https://github.com/aditya-grover/node2vec.

Work Report

In this section we want to describe the distribution of the work and detailed contribution of each member.

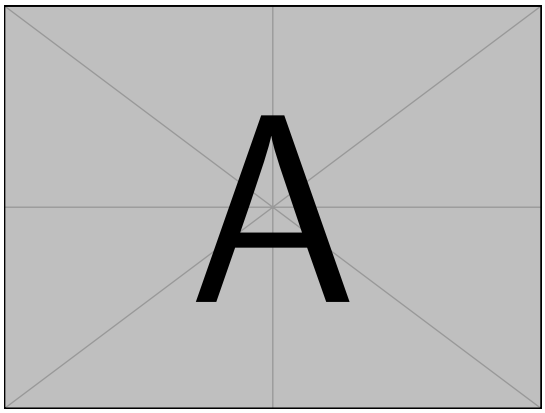
- *Francesco Biscaccia Carrara (2120934)*: **40% of the work**. Produce the code compute features and Runnig test on CAPRI
- *Alessandro Viespoli (2120824)*: **20% of the work**. Produce the code for the GAT and the code for plotting results
- *Riccardo Modolo (2123750)*: **40% of the work**. Produce the code to retrieve data, review code, write Proposal, Midterm and final Paper

	2S-T-Test (same variance)	2S-T-Test (diff variance)	Wilcoxon-Test
Accuracy	0.546	0.546	0.670
Precision	0.529	0.530	0.570
Recall	0.546	0.546	0.670
F1 Score	0.524	0.525	0.677

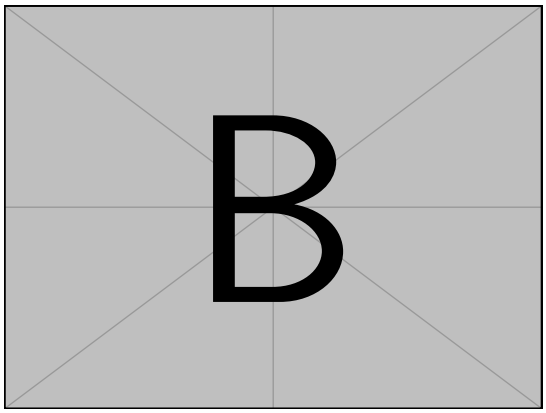
TABLE V: P-values for each test (on Cora dataset)

	2S-T-Test (same variance)	2S-T-Test (diff variance)	Wilcoxon-Test
Accuracy	×	×	0.908
Precision	0.567	0.569	0.969
Recall	×	×	0.908
F1 Score	0.583	0.585	0.969

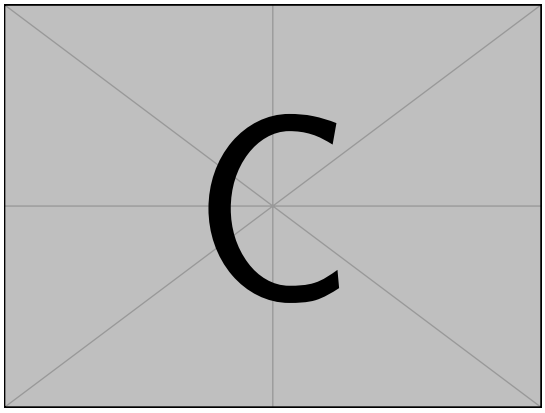
TABLE VI: P-values for each test (on Cora dataset)



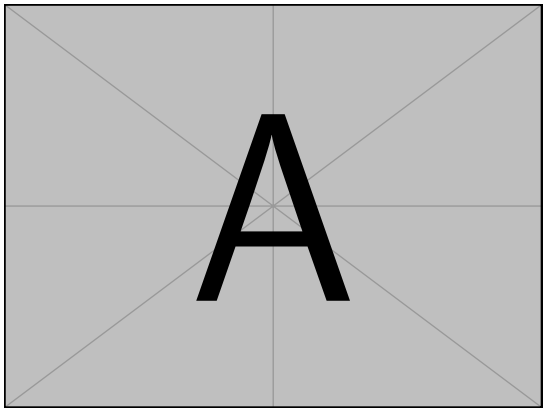
(a) **CORA**: confusion matrix without additional features



(b) **CORA**: confusion matrix with additional features



(c) **CiteSeer**: confusion matrix without additional features



(d) **CiteSeer**: confusion matrix with additional features