UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

NATHALIA SATIE GOMAZAKO

Exploration of neural network techniques to detect community structure in complex networks

São Carlos
2021

NATHALIA SATIE GOMAZAKO

Exploration of neural network techniques to detect community structure in complex networks

Monograph presented to the Graduate Program in Physics at the Instituto de Física de São Carlos, Universidade de São Paulo to obtain the degree of Bachelor.

Advisor: Prof. Dr. Gonzalo Travieso

São Carlos
2021

Gomazako, Nathalia
Exploration of neural network techniques to detect community structure in complex networks / Nathalia Gomazako; advisor Gonzalo Travieso -- São Carlos 2021.
32 p.

Course conclusion work (Bachelor's degree - Undergraduate Program in Computational Physics) -- Instituto de Física de São Carlos, Universidade de São Paulo - Brasil , 2021.

1. Complex networks. 2. Community. 3. Neural network. I. Travieso, Gonzalo, advisor. II. Title.

ABSTRACT

Recently, complex networks have been successfully used for modeling real-world systems as sets of elements (the nodes) with pairwise connections (the links). The topology of this network frequently contains important information about the system. One topological feature that is common in complex networks is its community structure. A community is a set of nodes that is more strongly connected among themselves than with nodes outside the community. Detecting communities by trying different combinations of nodes is not viable for large networks, and thus heuristic methods are generally used. Finding a division of the nodes of a network in communities corresponds to assigning to each node a label, the number of the community to which it belongs. This means that community detection can be seen as a classification problem. Neural networks are generally a useful tool for classification. This suggests the use of neural networks for the detection of communities in complex networks, which is the purpose of this work. We train and test a neural network with two different models of complex networks with community structure and evaluate the accuracy of the community structure detected. We also compare the results of the neural network with some well established community detection algorithms. We find that the neural network has good results for the simple models used, comparable or even superior to that of well established algorithms, but the accuracy decreases when a more flexible community generation model is used. Also, the accuracy is strongly affected by the model used for training (that is, by the kind of community structure present in the training set), which is expected, but implies that careful attention must be given to the training set.

Keywords: Complex networks. Community structure. Neural network.

RESUMO

Recentemente, redes complexas têm sido usadas com sucesso para modelar sistemas do mundo real como conjuntos de elementos (os nós) com conexões pareadas (os links). A topologia desta rede frequentemente contém informações importantes sobre o sistema. Uma característica topológica comum em redes complexas é sua estrutura de comunidade. Uma comunidade é um conjunto de nós que está mais fortemente conectado entre si do que com nós fora da comunidade. Detectar comunidades tentando diferentes combinações de nós não é viável para grandes redes e, portanto, métodos heurísticos são geralmente usados. Encontrar uma divisão dos nós de uma rede em comunidades corresponde a atribuir a cada nó um rótulo, o número da comunidade a que pertence. Isso significa que a detecção da comunidade pode ser vista como um problema de classificação. As redes neurais são geralmente uma ferramenta útil para classificação. Isso sugere o uso de redes neurais para a detecção de comunidades em redes complexas, que é o objetivo deste trabalho. Treinamos e testamos uma rede neural com dois modelos diferentes de redes complexas com estrutura de comunidade e avaliamos a precisão da estrutura de comunidade detectada. Também comparamos os resultados da rede neural com alguns algoritmos de detecção de comunidade bem estabelecidos. Constatamos que a rede neural apresenta bons resultados para os modelos simples utilizados, comparáveis ou até superiores aos de algoritmos bem estabelecidos, mas a precisão diminui quando é utilizado um modelo de geração de comunidades mais flexível. Além disso, a precisão é fortemente afetada pelo modelo usado para o treinamento (ou seja, pelo tipo de estrutura da comunidade presente no conjunto de treinamento), o que é esperado, mas implica que atenção cuidadosa deve ser dada ao conjunto de treinamento.

Palavras-chave: Redes complexas. Comunidade. Rede neural.

LIST OF FIGURES

LIST OF TABLES

SUMMARY

1 INTRODUCTION

Graph theory is the study of graphs, which are mathematical structures used to model relationships between objects. They can represent processes and relations between data from diverse study fields like physics, biology, sociology and others.

One possible characteristic for a graph is its separation in communities. Communities are subgraphs that contain nodes that are strongly connected to each other, possibly because they share similarities. Community detection is a costly activity that aims to identify the modules and its hierarchical organization from graph structures.

Computer evolution has provided enough computational resources to process and analyze data and regarding classification problems, there are improvements in analysis using artificial intelligence. Classification algorithms are responsible for identifying categories of new observations, through supervised learning. Machine learning presents a series of algorithms for classification and one of its methods is neural networks. A neural network is a structure inspired by the human brain that reflects the behavior of neurons and its connections, representing a network in which signals are processed, providing excellent results for many pattern recognition problems.

The objective of this work is to analyze whereas neural networks as a supervised machine learning model are viable tools for this classification task by proposing a neural network for this problem, analyzing its results for two graph generation methods and presenting a comparison with well known community detection methods.

1.1 GRAPHS

A graph is a data representation composed of a set of nodes and its links, where nodes represent objects and the links between them are called edges. It is used to model complex systems in diverse areas and it can be fully described by an adjacency matrix.

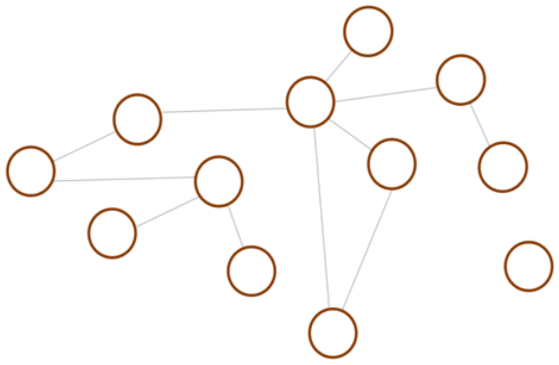


Figure 1 - Graph visual representation.
Source: By the author.

Definition 1 (Graph) An unweighted graph (or network) is represented as G = (V, E), where V and E denote the set of nodes and edges.

Definition 2 (Adjacency Matrix) The adjacency matrix A of a graph G = (V, E) is a NxN matrix, whose elements are a_{ij}. Each element aji is defined as $a_{ij} = 1$ if node i and j are connected, otherwise, $a_{ij} = 0$. If a graph contains no loops, the diagonal elements of the adjacency matrix are all zero. For an undirected graph A is a symmetric matrix.

The distribution of edges can present higher concentrations within groups of vertices than between these groups. This feature is called community structure and it can be studied in order to analyze clustering characteristics of graphs.

Communities are known as locally dense connected subgraphs or clusters of nodes.[1] In order to determine which nodes form a community, there are two specifications: nodes in a community present more connections between them and nodes outside a community present less. It means that a node is clustered into the community Ci by presenting higher connections between this node and its community node neighborhoods in comparison with connections between this node and node from other communities .

Definition 3 (Community) Communities are the subgraphs in a network where the nodes share more connections in comparison with nodes from different subgraphs. C = {C1, C2, ..., Ck} denotes a set of k communities in a network G, where i represents the i-th community from this network partition and Ci = {V1, V2, …, Vl} is the set of nodes in this community.

Definition 4 (Modularity[2]) Modularity is a property of a network that measures how well defined the communities are. It is calculated with $Q = \frac{1}{2m}\Sigma\frac{a_{ij}-k_ik_j}{2m}\delta(C_i, C_j)$, where k_i is the degree of node i and C_i is the community that contains this node and $m = \frac{1}{2}\Sigma a_{ij}$.

There are many algorithms to detect communities.[3] The Girvan–Newman algorithm[4], for example, detects communities by progressively removing edges, through its modularity. The remaining graphs expose the communities.

Another community detection strategy is through label propagation. Label propagation is a semi-supervised machine learning algorithm that adds labels to previously unlabeled data points. It works with subsets of the data points that have labels which then are propagated to other points.

Communities can also be detected through greedy algorithms. This algorithm starts each node in different communities, and then, it unites communities in pairs. Those pairs are chosen by calculating the unions that would provide the great increase in modularity. When uniting groups, the modularity will not increase when connecting nodes that are not connected to the same edges.

1.2 Neural networks

Definition 5 (Neural Network) Neural Networks is a series of algorithms in Machine Learning that aims to describe the relationship between data. They are composed of nodes that are responsible for handling the data through functions. The nodes compose layers and each layer contains nodes with weights and threshold values, which are responsible for passing information. A network has an input layer, one or more hidden layers, and an output layer. The data passes through each layer if the output of a node is above the specified threshold, meaning that a node has been activated.

Definition 6 (Activation function) The activation function is a function that represents how the weighted sum of the data is transformed.

Through an iterative process, neural networks create a model that is trained to recognize patterns by processing inputs. The input data is carried alongs the layers and is transformed with different weight associations.

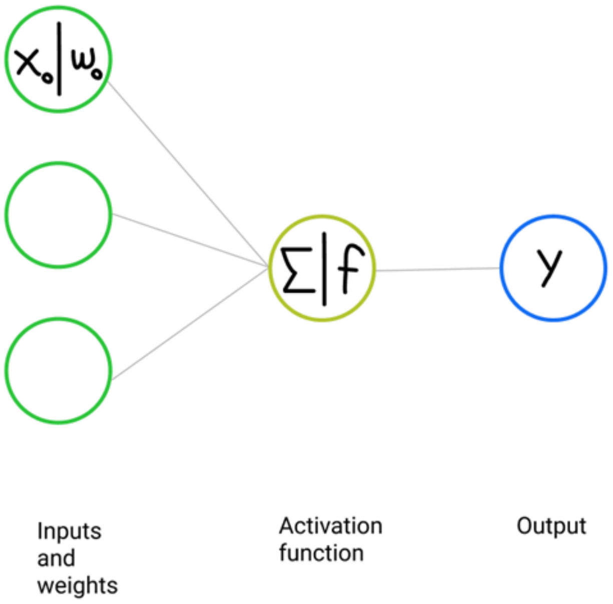


Figure 2 - An artificial neuron for neural networks.
Source: By the author.

Definition 7 (Accuracy) Accuracy is defined as the proportion between the predicted correct labels to the number of predicted and actual labels.

Definition 8 (Loss) The loss is responsible for representing the error for a single training sample.

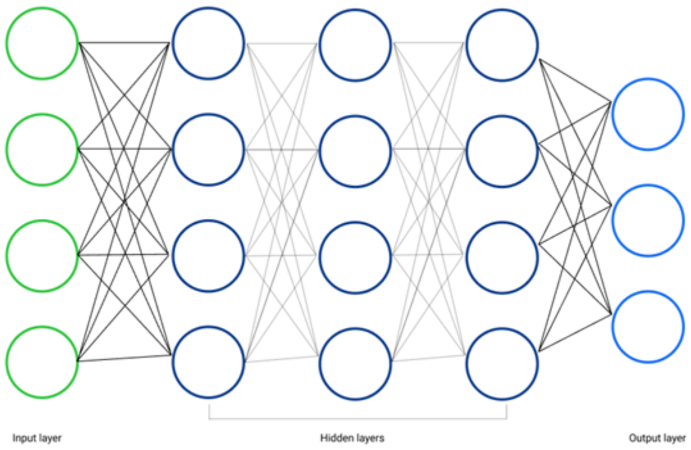


Figure 3 - Neural network visual representation.
Source: By the author.

Definition 9 (Optimization) The optimization of a neural network is a function that changes weights in order to reduce the loss.

The difference between the input and the output is the error. The network adjusts the weights according to the error value in order to increase the matching between the input and the output values, through an optimization function. The measurement that compares input and output values is called accuracy.

Definition 10 (Learning rate) Learning rate is a hyperparameter that is responsible for controlling how much a model can change in order to update weights.

Definition 11 (Early stop) The neural network improves its model and continues to train while its performance improves. If the performance stops improving or decays, the training stops in order to avoid accuracy drops.

Each round of this training is called epoch. An epoch has a limited rate of how much the data can be processed. With this learning rate, when a neural network cannot produce more models with higher accuracy, it stops by the neural network early stop configuration. At the training stage, the neural network is used to predict the behavior of the testing data. Then, the predicting stage produces other values of accuracy, but this time for the data that is meant to be evaluated.

2 Development

For this work, it is necessary to create graphs with its community structure in order to provide data for the input and output of the neural networks. It is also necessary to create neural networks as well as result visualization. For those objectives, Python is a great language for data science programming purposes. It is easy to use, has simple syntax and contains plenty of libraries. At this phase, the concern was to establish which libraries to use, and generate the input and output data for the analysis as well as the neural network to handle the classification.

2.1 Graph generation

Tool 1 (NetworkX[5]) NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Tool 2 (Random Partition Graph[6]) A partition graph is a graph of communities with predefined sizes, in which nodes in the same group are connected with probability $p_{in}$and nodes of different groups are connected with probability p_{out}.

The network is generated using the function random_partition_graph, which creates graphs for a number of partitions, where C = $\{C_1, C_2, \ldots, C_l\}$ where C_i represents the number of nodes in the i-th partition and the sums of C represents the total number N of nodes, connecting them according to the linking probabilities $p_{in}$and p_{out}.

This method returns the generated graph G, and this object contains a list of communities C and the number of communities is k. From the graph information, it is possible to generate its adjacency matrix.

When generating a graph, the probability p_{in} for the connections within groups and the probability p_{out} for the connections between groups enforces the community characteristic by the relationship $p_{in} \gg p_{out}$.

This matrix is the input data for the neural network and the chosen output data is a sorted list of communities, which means that communities are assigned with an ascending label value and the nodes are represented by their index in the list. Fulfilling the need for multiple graphs for a neural network to learn, we studied the effects of the number of graphs needed to achieve a neural network with great accuracy.

Tool 3 (Random Communities Model) A new model allowing a flexible communities number with randomized nodes in each partition.

Since random_partition_graph has a fixed number of communities and a fixed number of nodes in each community as well, another network generation method was conceived to perform studies. This method creates a network with community structure with N nodes with at most k_{max} communities, following the required rules for probabilities, each node being randomly assigned to a community.

Tool 4 (LFR[7]) Lancichinetti–Fortunato–Radicchi is an algorithm that generates benchmark networks.[8]

Another graph generation method is Lancichinetti–Fortunato–Radicchi. This is a graph built for performing community detection with heterogeneity in the distributions of the community sizes and its node degrees.

Tool 5 (Mutual Information) The mutual information of the two sets of labels represents the amount of information that the first set label provides about the second set.

Tool 6 (Normalized Mutual Information[9]) Normalized Mutual Information (NMI) is a normalization of the Mutual Information (MI) score to scale the results from 0 and 1, with 0 representing no mutual information and 1 representing perfect correlation.

There are measurements to detect the success of classification between the partitions. One of them is the normalized mutual information. It measures the dependence between two variables and exhibits how much the knowledge of a variable reduces the uncertainty about the other.

2.2 Neural network generation

Tool 7 (TensorFlow) TensorFlow is a Python library for machine learning.

Tool 8 (Keras) Keras is a deep learning API running on top of the machine learning platform TensorFlow.

Regarding neural network generation, we start by defining layers. The first layer is called the input layer. For a NxN graph, the input layer has format NxN, in order to accommodate the adjacency matrix for a graph with N nodes. The last layer is called the output layer and it contains the shape Nxk, where k is the maximum number of communities and it represents the possible labels for each node. The layers in between are called hidden layers and its configuration reflects on the accuracy results.

Definition 12 (Overfitting) Overfitting occurs when a model becomes too rooted to the input data, losing its plasticity to classify results.

Definition 13 (Dropout rate) Dropout rate is used to randomly set input units to 0.

The study started by using one of the simplest neural network models called Feed Forward Network (FFN). FFN is a network formed with three layers: batch normalization layer, dense layer and dropout layer. The batch normalization layer is designed for normalizing the input of its layer which aims to accelerate training and provide regularization in order to reduce error. The dense layer is designed for applying activation functions for nodes. The dropout layer is defined with the dropout rate used to randomly set input units to 0 to prevent overfitting. The output layer provides an array of possible labels for each node, and the position with greatest value represents the predicted label.

Tool 9 (KerasTuner[10]) KerasTuner is an easy-to-use, scalable hyperparameter optimization framework in Python that solves the pain points of hyperparameter search.

In order to discover the best parameters for the FNN, KerasTuner was used. KerasTuner executes multiple neural network algorithms with a combination of values in order to fulfill an objective of optimizing accuracy or optimizing loss. The algorithm trains a large number of models and carries forward the ones with best performance.

In this study, KerasTuner analyzed the learning rate for values rate $[1e^{-2}, 1e^{-3}, 1e^{-4}]$, focusing on maximizing the value of accuracy. KerasTuner was also used for the number of neural network nodes for the dense layer that had a variation between 20k and 100k, as well as the number of the dropout rate with range between 0.1 and 0.9. With the best results, we were able to build our neural network model.

Tool 10 (SparseCategoricalAccuracy[11]) The SparseCategoricalAccuracy is an accuracy achieved by dividing the number of successful ones by the total predictions.

Tool 11 (SparseCategoricalCrossentropy[12]) The SparseCategoricalCrossentropy is a loss function used when the output is composed by more than one label class.

Tool 12 (Adam optimization[13]) Adam optimization is an adaptive method that uses first-order and second-order moments by calculating the exponential moving average of the gradient and the squared gradient with the objective to find better learning parameters for the neural network.

The result accuracy can be calculated by various methods. In this work, the accuracy was calculated with the SparseCategoricalAccuracy, the loss function selected was

SparseCategoricalCrossentropy and the optimization was the Adam Optimization, for faster and computationally effective learning.

3 Results

The first step is to generate graphs with pronounced communities. This is achieved by selecting $p_{in} \gg p_{out}$ values. The initial values for the study were $p_{in} = 0.01$ and $p_{out} = 0.25$.

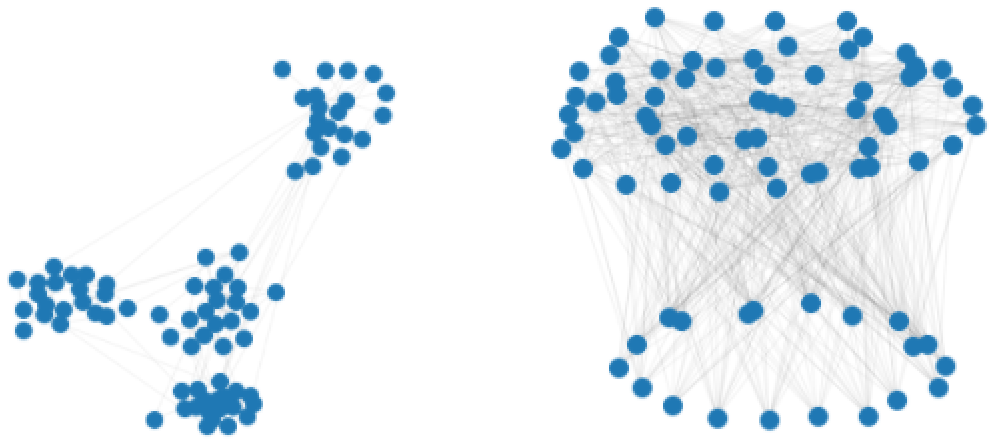


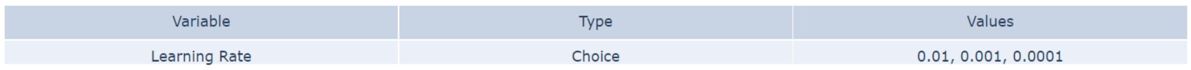
Figure 4 - On the left, there is a graph with $p_{in} = 0.25$ and $p_{out} = 0.01$. On the right, there is a graph with $p_{in} = 0.01$ and $p_{out} = 0.25$. A "good" cluster configuration is characterized by $p_{in} \gg p_{out}$.
Source: By the author.

The amount of nodes was set for N = 100 nodes and the number of graphs used for the input was g = 100, since it is necessary to use an expressive amount of data in the training stage.

After these initial parameters, analyses were made about the effects of p_{in} and p_{out}, the number of communities, the number of graphs and the number of nodes.

For the neural network generation, the first variable selected for the KerasTuner analysis is the Learning Rate. It is challenging to choose because smaller values may result in an unnecessary longer training process, whereas larger values might result in a suboptimal change of weights.

Table 1 - KerasTuner schema for the variable Learning Rate.



Variable	Type	Values
Learning Rate	Choice	0.01, 0.001, 0.0001

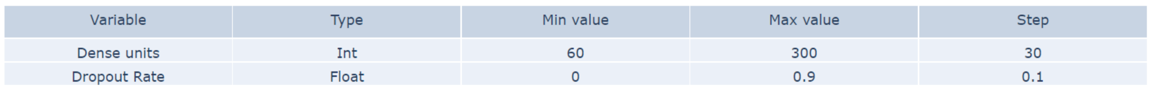
Source: By the author.

The other variables analyzed were related to the neural network size. The Dense Units variable is the shape of the layer with the activation function of a Gaussian Error Linear Units

(GELU)[14] used for the dense layer. The number of units per layer is related to the amount of time that the algorithm takes to run.

The Dropout Rate is a variable from 0 to 0.9 used for the dropout layer. A higher dropout rate adds confusion to the system, allowing the neural network to have softer learning models, causing the neural network to be more adaptable.

Table 2 - KerasTuner schema for the variable Dense Units and Dropout Rate.



Variable	Type	Min value	Max value	Step
Dense units	Int	60	300	30
Dropout Rate	Float	0	0.9	0.1

Source: By the author.

For each analysis, KerasTuner had to reconstruct parameters to adjust itself for graphs parameters. Here is the result of a KerasTuner analysis with the constructed neural network for graphs created through the random_partition_graph, with N = 100 nodes and k = 2 communities:

```
 Layer (type)                Output Shape              Param #
=================================================================
 input_features (InputLayer) [(None, 100, 100)]        0

 ffn_block1 (Sequential)     (None, 100, 140)          14540

 logits (Dense)              (None, 100, 2)            282
=================================================================
Total params: 14,822
Trainable params: 14,622
Non-trainable params: 200
```

When the size of the data is large, running an algorithm might take a considerable amount of time to complete every iteration. This can be done in smaller steps to save resources and the collection of data of reduced size is called batch.

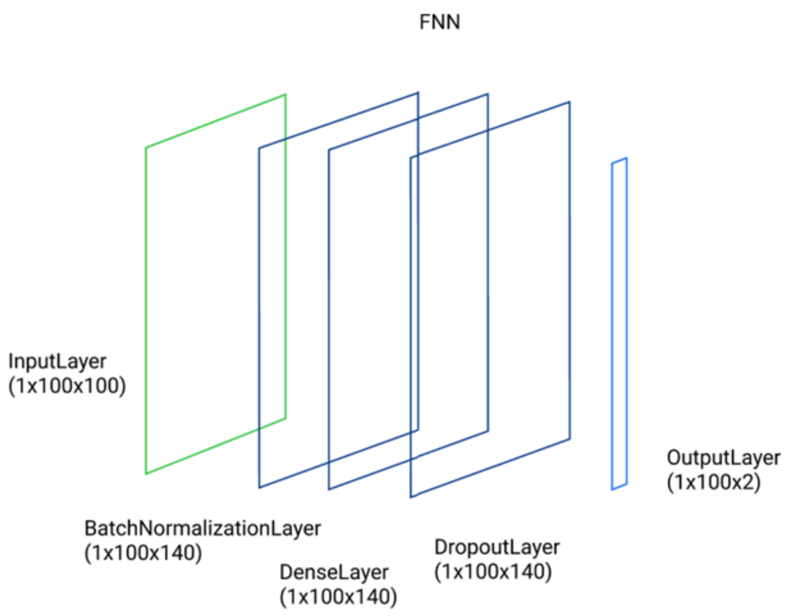


Figure 5 - Representation of a neural network with the same configuration as the one from the KerasTuner results.
Source: By the author.

Through the results, we can observe that the input layer contains an output shape that represents the size of the collection of data. As batches can contain multiple sizes depending on the amount of data, it is represented as None. The next itens of the output shape represents the format of the data, that in this case, it was the shape of the adjacency matrix 100x100.

The hidden layer of this neural network contains the output shape (None, 100, 140) where None is the variable size of the batch, 100x210 is the shape of nodes in this layer, where 100 is the number of nodes and 210 is the variable set from KerasTuner. The variable selected for the dropout rate in the hidden layer was 0.5 and the learning rate was 0.01.

The output layer is (None, 100, 2) where None is the variable size of the batch and 100x2 is the shape of the nodes in this layer and 2 is the number of communities.

3.1 Effect of connection probabilities within the graph

From literature, it is explained that in order to ensure clustering, $p_{in} \gg p_{out}$. Here is an analysis of the relationship between those variables. The FNN used was the one generated for N =100 nodes, k = 2 communities and g = 100 graphs as the input data. The analysis consisted of fixing values for p_{in} and variating values for p_{out}, and then fixing for $p_{out}$and with variations for p_{in}. The fixed value selected for p_{in} in the first experiment was p_{in}= 0.25, on the second experiment, the fixed value is p_{out}= 0.01.

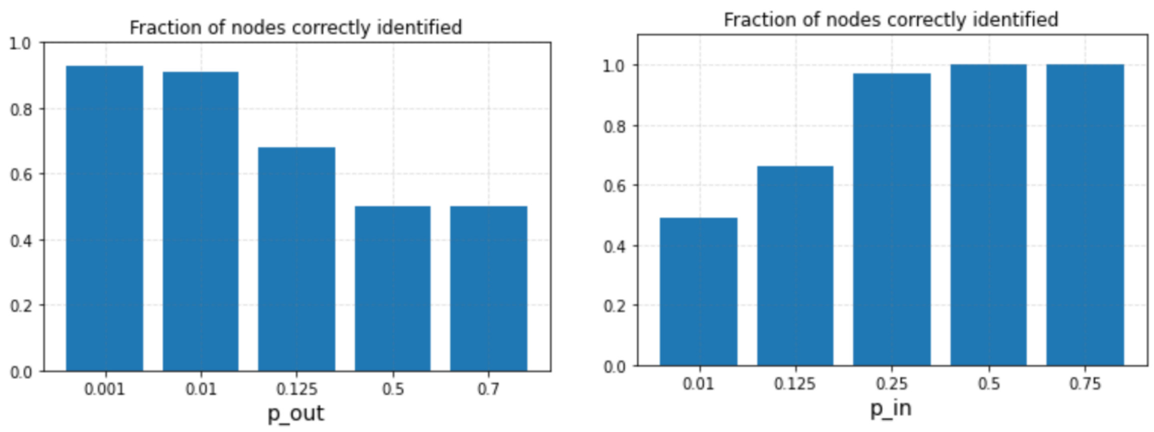


Figure 6 - On the left, there are results for p_{in}= 0.25 and p_{out} variations. On the right, there are results for p_{out} = 0.01 and p_{in} variations.
Source: By the author.

As p_{out} gets greater than p_{in}, the neural network is incapable of detecting communities correctly. It can be seen that the results are better for $p_{in} > 0.125$ with low variance on the assertiveness.

3.2 Accuracy per number of communities and per number of graphs

In order to analyze accuracy per number of communities, we vary the number of communities in the interval^{2,10}. It means that the random_graph_partition would contain k communities and each partition would have approximately $\frac{N}{k}$ nodes. For the other graph generation method, random_communities_model, k would mean the maximum number of communities possible k_{max}, and each community could have a different number of nodes, reducing the quality of the partitions.

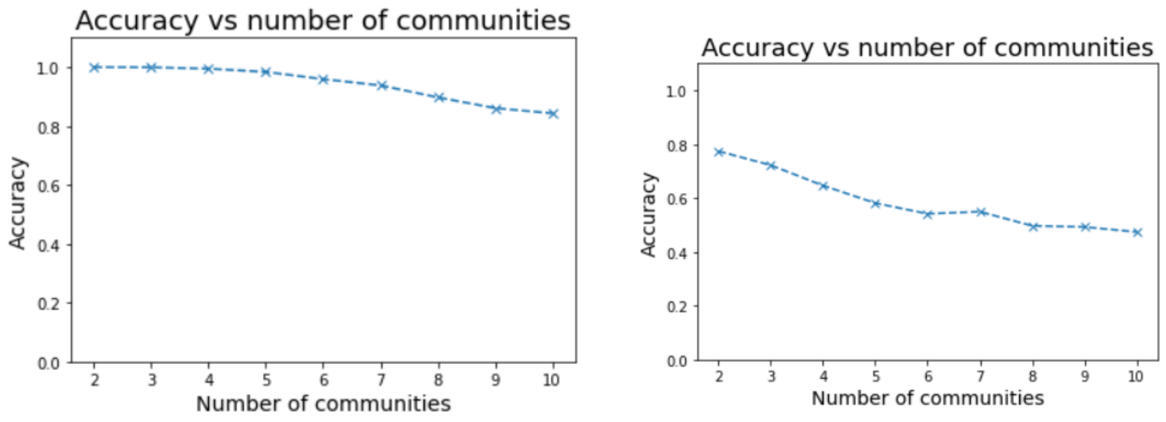


Figure 7 - On the left, there are results for the random_partition_graph accuracy for multiple numbers of communities. On the right, there are results for the random_communities_model accuracy.
Source: By the author.

We can see that the accuracy decreases in both analyzes when the number of communities is higher. The difference between accuracies are pronounced in the random_communities_model generation method because it contains communities that are diverse in size so many communities may remain undetected, especially the smaller ones.

In order to analyze accuracy per number of nodes, we variate the number of nodes used for input and output, increasing the number of nodes per graph for each neural network model from g = 100 to g = 1000, with steps of 100. The intention of this analysis is to understand how many nodes it is necessary to achieve a good accuracy.

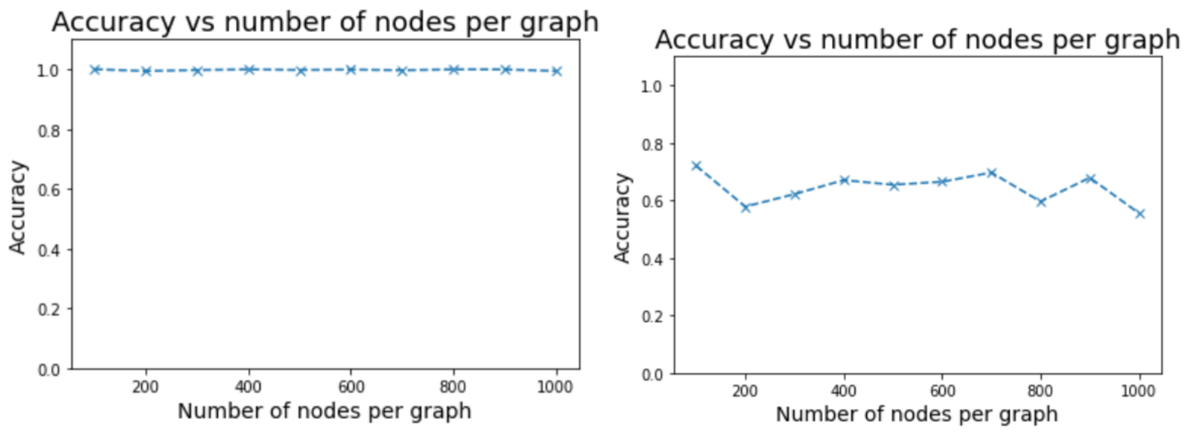


Figure 8 - On the left, there are results for the random_partition_graph accuracy for multiple numbers of nodes per graph. On the right, there are results for the random_communities_model accuracy.
Source: By the author.

We observe that the number of nodes per graph doesn’t provide significant changes to the accuracy for either of the graph generation methods.

3.3 Comparison with other community detection algorithms and graph generation methods

The comparison between multiple community detection algorithms started by creating a set of g = 100 graphs using the random_partition_graph. Each graph had N = 100 nodes and k = 3 communities groups. From each graph, NMI was calculated after detecting communities through the neural network, the givarn newman, the label propagation and the greedy modularity algorithms.

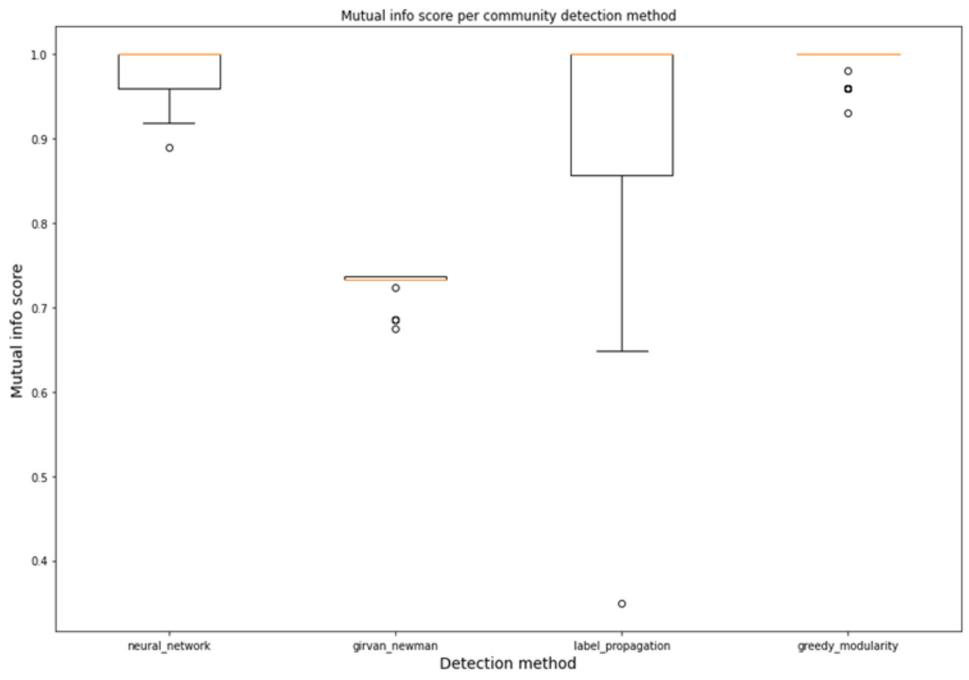


Figure 9 - Normalized mutual info score for multiple community detection methods.
Source: By the author.

We can observe that the neural network model created for the random_partition_graph is as effective as the greedy_modularity algorithm, whether the Girvan-Newman method presented a lower NMI.

It is noticeable that the Girvan-Newman method took a longer time to perform the community detection, taking around minutes while other methods were finished in approximately seconds. The label propagation method had a high NMI as well, but it presented a lot of variation, which resulted in a poor performance.

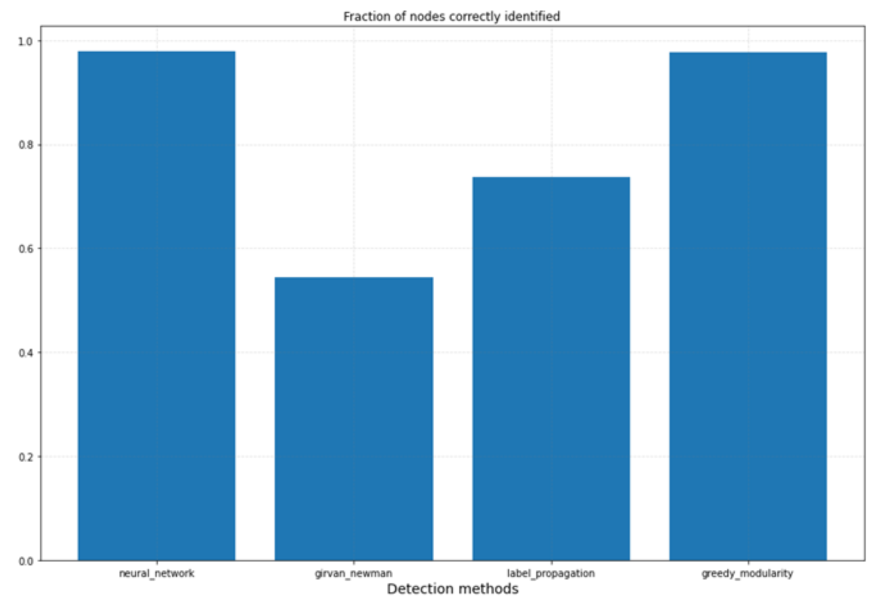


Figure 10 - Results of the fraction of nodes correctly identified for multiple community detection methods.
Source: By the author.

Regarding the usage of the neural network for different network generation methods, the analysis below presents the results for the use of a neural network trained for the random_partition_graph. With the resulting neural network, predictions were made to the other graph generation models.

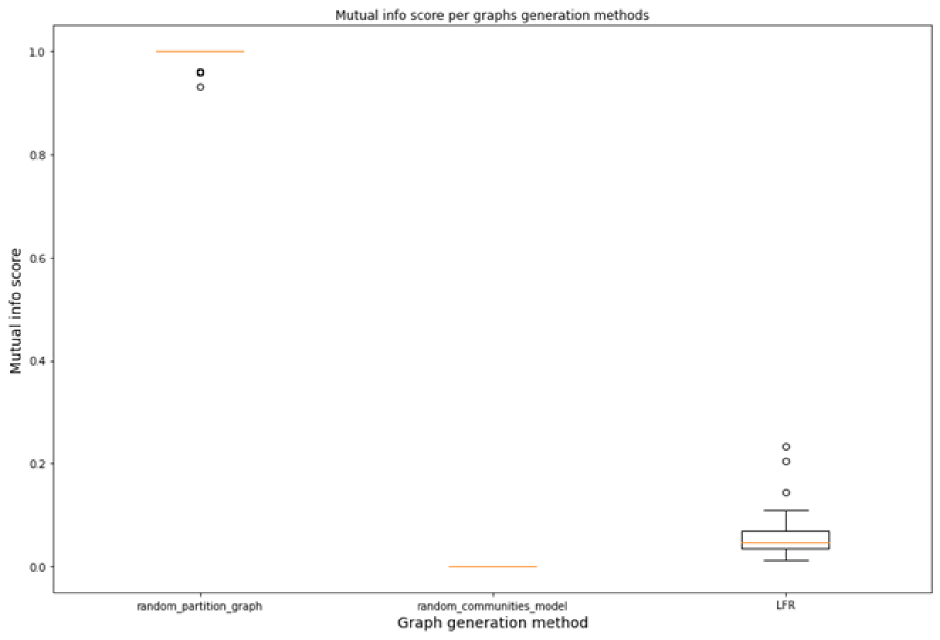


Figure 11 - Normalized mutual info score for multiple graph generation methods.
Source: By the author.

It is noticed that the results showed that the neural network is heavily attached to the configuration of its input data, presenting a low capacity prediction for other graph generation methods that weren’t used in the training stage.

4 Conclusion

In this work, we have investigated community structure in networks, analyzing the viability of the usage of neural networks for detecting such structure. This study provides analyses for different graph generation methods and network parameters.

The results indicate that our algorithm is a sensitive and accurate method for extracting community structure. The algorithms perform well for networks with a small number of communities, and the number of nodes used for the learning don't present substantial differences in the accuracy. This impacts positively on the running time of the KerasTuner analysis, since it presents the most time consuming. Many measures were taken in order to prevent overfitting, for example the use of a simple neural network model, using a large quantity of input data, using regularization and dropout layers, and using an early stop technique.

Overall, detecting communities using neural networks presented great success when analyzed for the same graph generation law, knowing that neural networks are adaptive to the input data. It means that the neural network is very sensitive to the formation of the graph used as well as its number of communities.

The creation of neural network models with higher flexibility is challenging[15], but it is possible to improve the result of the neural network by creating more complex models with the side effect of increasing the execution time and computer process.

References

1. FORTUNATO, S.; HRIC, D. Community detection in networks: a user guide. **Physics Reports**, v. 659, p. 1-44, 04 Nov. 2016. Available from: https://arxiv.org/pdf/1608.00163.pdf. Accessible at: 28 Mar. 2022.

2. FORTUNATO, S. Community detection in graphs. **Physics Reports**, v.486,n.3-5,p.75-174 Feb. 2010. DOI: 10.1016/j.physrep.2009.11.002. Available from: https://arxiv.org/abs/0906.0612. Accessible at: 28 Mar. 2022.

3. CLAUSET, A.; NEWMAN, M. E. J.; MOORE, C. Finding community structure in very large networks. **Physical Review E**, v. 70, n. 6, p. 066111, 09 Aug. 2004. DOI: 10.1073/pnas.122653799. Available from: https://arxiv.org/abs/cond-mat/0408187. Accessible at: 28 Mar. 2022.

4. GIRVAN, Michelle; NEWMAN, Mark EJ. Community structure in social and biological networks. **Proceedings of the National Academy of Sciences**, v. 99, n. 12, p. 7821-7826, 2002. DOI: 10.1073/pnas.122653799. Available from: https://www.pnas.org/doi/abs/10.1073/pnas.122653799. Accessible at: 28 Mar. 2022.

5. HAGBERG, A.; SWART, P.; SCHULT, D. Exploring network structure, dynamics, and function using NetworkX. *In*: Python IN SCIENCE CONFERENCE. 7. Los Alamos. **Proceedings [...]** , Los Alamos, Pasadena: National Lab.(LANL), 2008. p. 11-15. Available from: https://www.osti.gov/biblio/960616. Accessible at: 28 Mar. 2022.

6. NETWORK X: random_partition_graph. Network Analysis in Python, [*S. l.*], c2022. Available from: https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.random_partition_graph.html. Accessible at: 28 Mar. 2022.

7. NETWORK X: LFR_benchmark_graph. Network Analysis in Python, [*S. l.*], c2022. Available from: https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.LFR_benchmark_graph.html. Accessible at: 28 Mar. 2022.

8. LANCICHINETTI, A.; FORTUNATO, S.; RADICCHI, F. Benchmark graphs for testing community detection algorithms. **Physical Review E**, v. 78, n. 4, p. 046110, 24 Out. 2008. DOI: 10.1103/PhysRevE.78.046110. Available from: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.78.046110. Accessible at: 28 Mar. 2022.

9. SCIKIT Learn: sklearn.metrics.normalized_mutual_info_score. [*S. l.*], c2022. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html. Accessible at: 28 Mar. 2022.

10. KERAS-tuner. **Keras-Team**, github.com, [*S. l.*], 24 Mar. 2022. Available from: https://github.com/keras-team/keras-tuner. Accessible at: 28 Mar. 2022.

11. TENSOR Flow: tf.keras.metrics.SparseCategoricalAccuracy. [*S. l.*], c2022a. Available from:
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy. Accessible at: 28 Mar. 2022.

12. TENSOR Flow: tf.keras.losses.SparseCategoricalCrossentropy. [*S. l.*], c2022b. Available from:
https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy. Accessible at: 28 Mar. 2022.

13. KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, p. 1-15, 30 Jan. 2017. DOI: 10.48550/arXiv.1412.6980. Available from: https://arxiv.org/abs/1412.6980. Accessible at: 28 Mar. 2022.

14. HENDRYCKS, D.; GIMPEL, K. Gaussian error linear units (gelus). **arXiv preprint arXiv:1606.08415**, p. 1-9, 08 Jul. 2020. Available from: https://arxiv.org/pdf/1606.08415.pdf. Accessible at: 28 Mar. 2022.

15. LIU, F. *et al*. Deep learning for community detection: progress, challenges and opportunities. **arXiv preprint arXiv:2005.08225**, p.1-7, 23 Sept. 2020. DOI: 10.24963/ijcai.2020/693. Available from: https://arxiv.org/pdf/2005.08225.pdf. Accessible at: 28 Mar. 2022.