EGO NETS GRAPH CLASSIFICATION

MLNS Final Project Report

Swapnil PADE, Sanjana, GUPTA, Sibiranjith NAGESH, Solomon BIDJA, Kenza BADDOU

ABSTRACT

This study aims to investigate the effectiveness of different machine learning algorithms for classifying ego-net graphs. Ego-net graphs are used in various industries, making the task of graph classification increasingly important. The study will experiment with different graph classification approaches to determine the most suitable methodology for the task.

INTRODUCTION & MOTIVATION

Ego-net graphs are a valuable tool for studying social networks. In these graphs, individuals are depicted as nodes, and their connections with others are represented as edges. These graphs are particularly useful for analysing the relationships and behaviour of individuals within a specific community or social network. Many industries, such as social media sites, marketing companies, law enforcement, finance, healthcare, and gaming, rely on ego-net graphs to analyse their users. As social network analysis gains more importance in various fields, the use of ego-net graphs is becoming more widespread.

Our goal is to investigate various machine learning algorithms that can classify ego-net type networks and determine the most effective methodology for achieving this. This involves experimenting with different approaches to graph classification and identifying the most suitable one for the task at hand.

RELATED WORK

There has been some similar work on ego-net graphs, and one such work has been published in a paper called "Graph Classification Using Structural Attention" by Zhang and colleagues in 2018. The study presents a new graph classification method that employs graph neural networks and structural attention mechanisms to improve accuracy and efficiency. The paper evaluated the proposed method using different benchmark datasets, including bioinformatics data and social networks, and demonstrated its superior performance compared to existing methods. This paper aligns with our research topic as it explores the use of machine learning algorithms to enhance graph classification, specifically highlighting the effectiveness of GNNs. Unlike the paper by Zhang and his colleagues, we have used a variety of ensemble bagging and boosting methods to provide a broader understanding of the performance of various machine learning algorithms in classifying ego-net graphs, which can be useful for actual industry applications that require efficient and accurate graph classification.

THE DATASET

For this project, we will be utilising the Twitch dataset [4], which consists of ego-nets of users who took part in the partnership program in April 2018. This dataset is composed of nodes that represent individual users and links that indicate friendships. To have a better understanding of this dataset, following are some properties of this dataset:

• Number of graphs: 127,094
• Directed: No.
• Node features: No.
• Edge features: No.
• Graph labels: Yes.
• Binary-labelled.
• Temporal: No.

The dataset we are working with consists of two categories of users: those who play single player games and those who engage in multiplayer games. The ego-net of users who play single player games are more densely connected. Our objective for this project is to train our machine learning models using binary graph classification to predict whether a user plays single player or multiplayer games based on their ego-net. This information can be utilised by gaming companies to gain insight into their target audience.

We are also provided with a “target.csv” which contains binary information for each graph in the egonet network. “0” representing people engaged in single player games and “1” representing those engaged in multiplayer games.

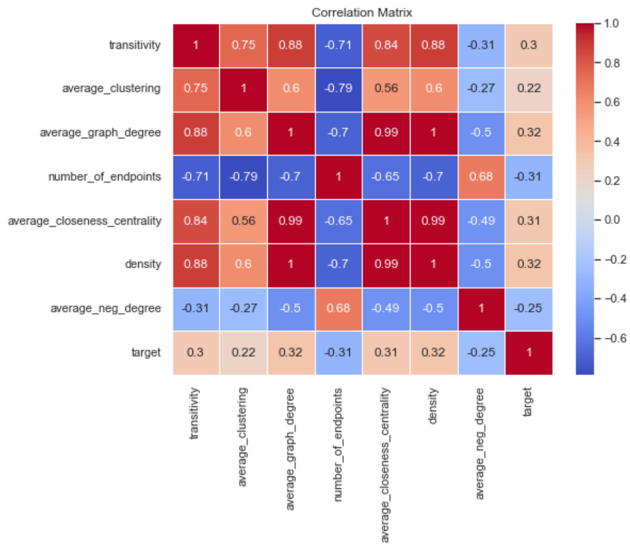
METHODOLOGY

As mentioned earlier, we have employed different types of classifiers and conducted a comparative study for the graph classification. We have used accuracy and precision as metrics to determine the optimal classification method for data consisting of user ego-nets. In addition to exploring different classifiers, we aim to identify the most relevant features that would contribute to accurate classification. The following paragraphs detail the preprocessing and the modelling steps used to analyse the data.

Data Preprocessing

Data preprocessing is always an important step to be carried out before diving into data analysis since it helps us users ensure that the data is clean, consistent and complete. Additionally, it also prepares the data to be used in ML models by removing irrelevant sections and identifying useful patterns. The preprocessing steps used for this project are as follows:

1. The target data was loaded (with the ‘y’ variable set to the ‘target’ column), and a separate data frame was created which had just the ‘id’ column sorted in ascending order.
2. The Twitch ego network data was also loaded (JSON file) and converted into a dictionary containing NetworkX graphs with nodes sorted in ascending order. The dictionary was then saved as a pickle file and loaded into memory for faster execution in the future.
3. A list of graph metric feature functions is defined - average_closeness_centrality, transitivity, average_clustering, density, average_neg_degree. A loop is run over each feature function for feature extraction, computing the feature value for each graph in the dictionary. The resulting values were stored in a new column in the dataframe created earlier.
4. The 'target' column from the original target data was added to the saved CSV file of features.
5. Lastly we plotted a correlation heat map matrix to compute the correlation between all the features identified in the graph.



	transitivity	average_clustering	average_graph_degree	number_of_endpoints	average_closeness_centrality	density	average_neg_degree	target
transitivity	1	0.75	0.88	-0.71	0.84	0.88	-0.31	0.3
average_clustering	0.75	1	0.6	-0.79	0.56	0.6	-0.27	0.22
average_graph_degree	0.88	0.6	1	-0.7	0.99	1	-0.5	0.32
number_of_endpoints	-0.71	-0.79	-0.7	1	-0.65	-0.7	0.68	-0.31
average_closeness_centrality	0.84	0.56	0.99	-0.65	1	0.99	-0.49	0.31
density	0.88	0.6	1	-0.7	0.99	1	-0.5	0.32
average_neg_degree	-0.31	-0.27	-0.5	0.68	-0.49	-0.5	1	-0.25
target	0.3	0.22	0.32	-0.31	0.31	0.32	-0.25	1

Fig 1.: Correlation Heatmap Matrix between Graph Features

6. We then did feature selection, drop the irrelevant features with close correlation and feed the relevant features- transitivity, average_clustering, number_of_endpoints, average_closeness_centrality and average_neg_degree in our x_data and the target in the y_data.

Data Modelling

To accommodate the size of the data, we used an 80:20 split for training and testing data, respectively. Two ensemble boosting models and an ensemble bagging model were created and trained on the clean and appropriately split data:

XGBoost classifier- When it comes to categorising networks of the ego-net kind, which are made up of a core node (the ego) and its close neighbours (the alters), XGBoost classifiers thrive in this particular application. Class imbalance is a common feature of networks of the ego-net type, where the proportion of positive and negative samples may not be uniform. XGBoost contains built-in tools to manage imbalanced data. This makes it easier for XGBoost to produce accurate predictions for both positive and negative classes and learn from ego-net data that is imbalanced. A crucial function offered by XGBoost is "feature importance," which enables users to choose the features that are most pertinent for categorization. Finding the salient characteristics that lead to the classification can be extremely important since in ego-net type networks, the relationships between the ego node and its alters might be complex. The most relevant features are found using XGBoost's feature significance analysis, which can help in understanding the dynamics and structure of the network.

CATBoost classifier- Categorical features, such as characteristics of the ego node and its alters, such as gender, employment, or interests, are frequently included in networks of the ego-net type. Without the requirement for complex feature engineering or one-hot encoding, categorical features can be handled smoothly by CATBoost. It has the ability to process categorical features directly, which enables it to capture the natural connections and interactions between many categorical variables that are frequently present in ego-nets. The feature importance analysis offered by CATBoost aids in locating the most essential features for classification. This makes it easier to read the model and comprehend the value of various categorical features in the ego-net data. In order to better grasp the connections and interactions between the ego node and its alters and to gain important insights for making decisions, interpretability is essential in ego-net analysis.

RandomForestClassifier- The accuracy of classification may be impacted by noise or missing values in ego-net data. In order to handle missing values and noisy features, RandomForest classifiers do not require considerable data preprocessing or imputation. As a result, they are robust to noisy data. For ego-net classification tasks, RandomForest is a good option because of its ensemble of decision trees, which can also reduce the influence of outliers and noisy data points. The link between features in ego-net data may not necessarily be linear, and there may be intricate nonlinear interactions between the ego node and its alters. Without making any assumptions about the distribution of the underlying data, RandomForest can capture intricate patterns and interactions between features. Because of its adaptability, it can capture nonlinear interactions in ego-nets that may not be suitable for linear classifiers.

Hyperparameter Tuning

We have improved the generalizability of our results and prevented overfitting by incorporating validation within the training dataset. To achieve this, we have used cross-validation techniques. Due to this crucial step, our classification models can effectively classify the ego-net network data, while avoiding any overfitting that may occur due to using a limited training dataset.

XGBoost Classifier- Ego-net networks frequently have a modest size since they only include the ego node and its close neighbours. Due to the lack of sufficient data, the classifier may get overfit and memorise the training set rather than generalising adequately to new data. The regularisation strategies offered by XGBoost, such as "max_depth", "min_child_weight", “learning_rate” and "gamma", aid in managing the model's complexity and avoiding overfitting. To use this to our full advantage we tuned our XGBoost model to search the best parameters using RandomSearchCV by performing 5 fold cross validation.

CATBoost Classifier- Regularization methods like "l2_leaf_reg", “subsample”, “colsample_bylevel” and "max_depth" are among the built-in procedures provided by CATBoost to prevent overfitting. With the use of these regularisation strategies, the model's complexity can be managed, and by preventing it from memorising the training data, it becomes more robust and generalizable to new data, which is crucial for ego-net classification with sparse data. To use this to our full advantage we tuned our CATBoost model to search the best parameters using RandomSearchCV by performing 5 fold cross validation.

RandomForestClassifier- RandomForestclassifiers can be a good choice for classifying ego-net type of networks due to their ability to handle high-dimensional data, robustness to noisy data, non-parametric and non-linear modelling, ensemble of decision trees, and interpretability to some extent. By leveraging the strengths of RandomForest classifier through tuning it and finding the best parameters through RandomSearchCV for “max_depth”, “min_sample_split”, “min_sample_leaf” and “max_features” by performing 5 fold cross validation it is possible to build accurate and robust models for ego-net analysis.

EVALUATION

After running our model on various machine learning classifiers we got the following accuracies for each tuned and untuned models of the classifiers [Table1]:

Classifiers	**No tuning (%)**	**Hyperparameter tuning (%)**
XGBoost Classifier	**70.44**	**70.64**
CATBoost Classifier	**70.59**	**70.63**
RandomForest Classifier	68.01	70.38

Table 1: Accuracy scores on test data of each classifier

We can see that the XGBoost Classifier and CATBoost Classifier perform fairly well and give almost same results. The RandomForest Classifier also gives almost similar results. Our best model is the hyperparameter tuned model of the XGBoost Classifier giving the highest accuracy of 70.64%.

CONCLUSION

In conclusion, many machine learning classifiers including XGBoost, CATBoost, and RandomForest can successfully conduct ego-net classification. For ego-net classification tasks, these classifiers have a number of benefits, including the capacity to handle categorical features, high prediction accuracy, robustness to overfitting, interpretability to some extent, scalability, and the capacity to capture complex relationships and interactions within ego-net data.

Ego-net classification tasks are well suited for XGBoost amongst the three chosen due to its great performance in handling huge datasets and complicated feature interactions. On the other hand, CATBoost is a potential option for ego-nets because it is specifically made to handle categorical features, which are frequently present in them. RandomForest can be useful for ego-net classification applications due to its capacity for handling high-dimensional data, robustness to noisy data, and ensemble of decision trees.

Overall, the selection of a classifier is influenced by the unique traits of the ego-net data and the specifications of the classification task. By utilising the advantages of these classifiers, accurate and reliable models for ego-net analysis can be created. This allows for numerous applications in social network analysis, recommendation systems, targeted marketing, and other fields, as well as insights into the relationships and interactions within ego-nets.

Future Scope

The binary graph classification of egonets has room for improvement, according to the report's findings. Investigating the use of graph neural networks (GNNs) for this task is one possible direction for future research.

GNNs are a particular class of neural network created specifically to operate on graph data directly, making them ideal for tasks involving graph analysis and classification. GNNs have demonstrated promising results in a variety of graph-related tasks, including node classification, link prediction, and graph classification, by incorporating the structural information of graphs into their models.

GNNs could be used to capture the intricate relationships between nodes and the subgraph structures within egonets in the context of egonet classification. This could perform better than the conventional machine learning models used in the report and probably result in more accurate classification.

Making and fine-tuning a GNN specifically for the egonet classification task is one possible strategy. Designing a GNN architecture that can efficiently capture the structural data of egonets and training it on a sizable dataset of labelled egonets would be required to accomplish this. To achieve optimal performance, this would necessitate a sizable amount of data preprocessing and engineering, as well as extensive hyperparameter tuning.

Exploring the use of transfer learning and pre-trained GNN models to increase egonet classification accuracy is also a possible future direction. Overall, the use of GNNs for egonet classification could hold a lot of promise for future study and is a worthwhile area for field researchers to investigate.

REFERENCES

1. Tranmer M, Steel D, Browne WJ. Multiple-membership multiple-classification models for social network and group dependences. J R Stat Soc Ser A Stat Soc. 2014 Feb;177(2):439-455. doi: 10.1111/rssa.12021. Epub 2014 Aug 9. PMID: 25598585; PMCID: PMC4282334.
2. Maurya, S.K., Liu, X., Murata, T.: Feature selection: key to enhance node classification with graph neural networks. CAAI Trans. Intell. Technol. 1– 15 (2023). https://doi.org/10.1049/cit2.12166
3. Yang, Rendong & Bai, Yun & Qin, Zhaohui & Yu, Tianwei. (2014). EgoNet: Identification of human disease ego-network modules. BMC genomics. 15. 314. 10.1186/1471-2164-15-314
4. Dataset - https://snap.stanford.edu/data/twitch_ego_nets.html