Rule-based GNN explanation with cooperative games

Vincent Elouan
(supervisor: Marc Plantevit)

Technical Report n^ooutput, June 2023
revision

Recent works have proposed to explain GNNs using activation rules. Activation rules allow to capture specific configurations in the embedding space of a given layer that is discriminant for the GNN decision. In the other hand, cooperative games allow to compute importance score in a set of players collaborating for a common result, allowing to give scores to features using some specific characteristic function in various machine learning problem. In the report, we propose a method that use cooperative games, especially Shapley value, to rank activation rule importance in the embedding of a node to give more information about what the GNN really captures in its hidden layers.

Des travaux récents ont proposé d'expliquer les GNN en utilisant des règles d'activation. Les règles d'activation permettent de capturer des configurations spécifiques dans l'espace d'intégration d'une couche donnée qui est discriminante pour la décision GNN. D'autre part, les jeux coopératifs permettent de calculer le score d'importance dans un ensemble de joueurs collaborant pour un résultat commun, permettant de donner des scores aux fonctionnalités en utilisant une fonction caractéristique spécifique dans divers problèmes d'apprentissage automatique. Dans le rapport, nous proposons une méthode qui utilise des jeux coopératifs, en particulier la valeur de Shapley, pour classer l'importance de la règle d'activation dans l'intégration d'un nœud pour donner plus d'informations sur ce que le GNN capture réellement dans ses couches cachées.

Keywords
Cooperative games, Activation rules, GNN, xAI, GNN explanability, SHAP, Shapley

Laboratoire de Recherche de l'EPITA
14-16, rue Voltaire – FR-94276 Le Kremlin-Bicêtre CEDEX – France
Tél. +33 1 53 14 59 22 – Fax. +33 1 53 14 59 13
elouan.vincent@epita.fr – http://www.lre.epita.fr/

Copying this document

Contents

Chapter 1

Introduction

Artificial intelligences are built to help human beings to perform various cognitive tasks, especially classification which is one the most used. Helping human on these tasks can be done in two ways, either assist them in decision making process and let the final decision and the control to the human, either by replacing them in less critical tasks.

Recent years have witnessed the advent of deep learning models for the general public. Although their power intrigues and excites, the need to regulate artificial intelligences is felt and leads to discussion of regulatory text like the AI Act.

One of the needs that comes out of these is the need to make the models interpretable, to be able to explain the decisions made by the model. There are several motivations that could lead to wanting to understand the decision of a model, be it ethical, for detection and correction of bias, or extract knowledge learned by the model.

Some models are easily interpretable, but when we talk about deep learning models, models act like black boxes As the fields of application of this kind of model can be critical (health, justice), it is not desirable to allow a model to make decisions without the ability to explain them.

In computer vision or natural language processing, more and more methods are emerging to solve this interpretability problem. This is not the case for neural graph networks that learn representations of nodes in a latent space and therefore capture a lot of information that is uninterpretable by humans.

In this report, we address the problem of GNN interpretability by trying to extract more information about internal representation of nodes learned by the model using cooperative games to compute score of some interesting activation pattern inside the hidden layers of a GNN.

First, in section 2 we will talk about the state of the art of the explanation of GNN, and we will present the tools we will use such as activation rules and cooperative games. Then, we will present our method in Section 3 using cooperative games to compute activation rules scores. Finally, we will perform some experimentation in section 4 to evaluate and discuss our method.

Chapter 2

Preliminaries

2.1 Graph Neural Networks

A graph $\mathcal{G}$ consists of a set of nodes V and a set of edges E. We represent graphs as $\mathcal{G} = (V, X, A)$, where $X \in \mathbb{R}^{n \times d}$ is a matrix of node features with n nodes and d dimensions, and $A \in \{0, 1\}^{n \times n}$ is the adjacency matrix representing the connections between nodes. Graph Neural Networks (GNNs) leverage message-passing techniques to learn node representations and perform various tasks on graphs, such as node classification, graph classification, link prediction, or clustering.

In GNNs, the representation of each node $u \in V$ is updated using message-passing and aggregation of information from its neighboring nodes. This update process occurs across multiple layers of the GNN, allowing nodes to gather information from their multi-hop neighbors and generate structure-aware representations.

We define $h_i^{(l)}$ as the representation of node i at iteration (layer) l, and $AGGR(.,.)$ as the aggregation method. The update rule for node representations can be expressed as follows:

$$h_u^{(l)} = AGGR(h_u^{(l-1)}, \{h_i^{(l-1)} | i \in \mathcal{N}(u)\}) \tag{2.1}$$

Here, $h_u^{(l-1)}$ represents the previous layer's representation of node u, and $\{h_i^{(l-1)} | i \in \mathcal{N}(u)\}$ denotes the set of representations of node u's neighbors, where $\mathcal{N}(u)$ is the set of neighbors of node u. The aggregation function combines these representations to compute the updated representation $h_u^{(l)}$ for node u at layer l.

GNN models then used these nodes representation to either classify nodes, graphs or predict edges. In this report, we will mainly address the problem of graph classification problem.

2.2 GNN explanation

Yuan et al. (2022) presents a good taxonomy of the GNN explanation methods. This study is beginning to date and therefore there are missing several new methods for explaining GNNs, but the types of methods are almost always the same and this taxonomy remains relevant. It presents two categories of explanation of GNNs: instance-level and model-level methods.

2.2.1 Instance level methods

Given an input graph, instance-level methods provide input-dependent explanations by identifying important input characteristics that the model used for its prediction. Among instance-level methods, we can distinguish four different families of methods: The gradient/features-based methods which use the gradients or hidden feature map values to compute the importance of the input features. The perturbation-based methods which learn a mask by studying the prediction when perturbing the input graphs. The surrogate methods explain an input graph by sampling its neighborhood and learning an interpretable model. The decomposition-based methods start by decomposing the prediction score to the neurons in the last hidden layer and back propagate these scores layer by layer until reaching the input space.

2.2.2 Model-level methods

XGNN (Yuan et al. (2020)) consist in training a graph generator to maximize the predicted probability for a certain class and uses such graph patterns to explain this class, but it cannot be applied to complex phenomena because it is based on the assumption that each class can be explained by a single graph.

A type of method is not stated in this taxonomy because it is more recent, it is the methods that rely on the activation rules.

2.2.3 Rule-based methods

DISCERN (Veyrin-Forrer et al. (2021b)) methods use a Monte Carlo Tree Search to find a realistic subgraph whose embedding is similar to an activation rule, which are groups of vector components that are mostly activated together in graphs having the same GNN decision. Starting with an empty graph, it aims to maximize the score of targeted activation rule for the graph by adding an edge to the graph. A realism factor is computed each step at avoid generating a graph which has an embedding close to an activation rule without being realistic and very different from graphs in the dataset. This method uses three different metrics for the score: Cosine, Cross-Entropy, Relative-CE.

Kamal et al. (2023) extract the median graph of the set of ego-graphs to build a representative graph for a given rule, guaranteeing the realism of the product graph

2.3 Activation rules

Activation rules are groups of vector components that are mostly activated together for a same decision of the GNN. An activation rule $A^l \rightarrow c$ is composed of a binary vector A^l of size K and $c \in \{0, 1\}$ a decision class of the GNN. A graph $g_i = (V_i, E_i, L_i) \in G$ activates the rule if there is a node v in V_i such that for $\hat{H}^l$ the activation matrix, $\hat{H}^l[v, k] = (A^l)_k, \forall k = 1...K$. It is denoted $\mathbf{Act}(\mathbf{A}^l \rightarrow c, v)$. Moreover, on a dataset, we are able to extract all the graphs that activate a rule, so that, we can compute the median graph on the set of graphs which activates a targeted rule and be sure that this graph is realistic and activate the rule. Veyrin-Forrer et al. (2021a) introduce the notion of activation pattern and present a way to mine activation rules from a GNN.

Activation rules of interest are the one for which $supp(A^l \rightarrow c, G)$ is high and $supp(A^l \rightarrow \bar{c}, G)$ is low. There are different ways to calculate activation rules, Veyrin-Forrer et al. (2021a) propose to use the subjective interestingness framework De Bie, Tijl (2011). Therefore, for binary graph classification, an activation rule of a model is defined by an hidden layer, a set of activated

component in this layer, a targeted class ($c \in \{0, 1\}$), a global interestingness score, and one for each class. When extracted, we can of course also know which nodes of which graph activate a given rule.

For the exploration step of our method, we search for a subgraph that is the closest to an activation rule, to that end, we need to define a measure of proximity between embedding and activation rule. Veyrin-Forrer et al. (2021b) propose three different measures to evaluate the proximity between an ego-graph embedding h_v^l centered at node v and a layer l and an activation rule $A^l \rightarrow c$, with $A^l = \{a_1, ..., a_K\}, a_i \in 0, 1$. They define $E_g = \{\epsilon_1, ..., \epsilon_K\}$ the ego-graph embedding truncated to fit the interval $[0, 1]$; $\epsilon = min(max(0, (h_v^l)_k), 1)$.The similarity between E_g and A^l can be evaluated by the cosine similarity measure :

$$Cosine(E_g, A^l) = \frac{E_g.A^l}{||E_g||\ ||A_l||} = \frac{\sum a_i \epsilon_i}{\sqrt{\sum a_i^2}\sqrt{\sum \epsilon_i^2}} \tag{2.2}$$

Or by the cross-entropy measure (equivalent the log-likehood):

$$CrossEntropy(E_g, A^l) = \sum a_i log(\epsilon_i) \tag{2.3}$$

And the Relative-Cross-Entropy:

$$Relative\_CE(E_g, A^l, R) = CrossEntropy(E_g, A^l) + \max_{r \in R}(CrossEntropy(E_g, r)) \tag{2.4}$$

2.4 Cooperative games

Cooperative game theory is a branch of mathematics that focuses on studying how rational individuals or agents can cooperate to achieve mutually beneficial outcomes. Unlike non-cooperative game theory, which primarily deals with strategic interactions where players act independently, cooperative game theory explores situations where players can form coalitions and work together towards common goals.

In cooperative games, players recognize the potential benefits of cooperation and aim to distribute the resulting gains in a fair and efficient manner. These games involve scenarios where players can negotiate, form alliances, and make binding agreements to achieve better outcomes than what they could achieve individually.

The central concept in cooperative game theory is the formation of coalitions, which are groups of players who collaborate and coordinate their actions. These coalitions can range from small subsets of players to the grand coalition, which includes all players in the game. The cooperative nature of these games allows players to consider not only their individual strategies but also how their actions affect the overall welfare of the group.

To analyze cooperative games, various solution concepts have been developed. These solution concepts aim to provide guidelines for allocating the joint payoff generated by the coalition among the individual players. Prominent solution concepts include the Shapley value, the Nash bargaining solution, and the Core, each with its own set of properties and principles for fair distribution.

Cooperative game theory finds applications in diverse fields such as economics, political science, computer science, and operations research. It helps us understand how coalitions are formed, how resources are shared, and how cooperation can lead to more desirable outcomes for all participants.

2.4.1 Problem definition

In game theory, a cooperative game is represented by a pair (N, v), where N is a set of players denoted as $\{1, \ldots, n\}$, and v is the characteristic function that maps subsets of players (coalitions) $S \subseteq N$, with $v(\emptyset) = 0$ representing the empty coalition.

A solution function ϕ is a mapping that assigns each cooperative game (N, v) a vector $\phi(N, v) \in \mathbb{R}^n$. This vector, known as a solution, represents an allocation of the total payoff $v(N)$ generated by all players to each individual. The i-th coordinate of $\phi(N, v)$, denoted as $\phi_i(N, v)$, corresponds to the payoff attributed to player i. The solution function $\phi(N, v)$ is also referred to as the "value" of the game when it satisfies certain properties. Various solution concepts have been proposed, each with distinct properties, to characterize different allocation schemes for cooperative games, such as the Shapley value and the Hamiache-Navarro values.

2.4.2 Shapley value

The Shapley value is a widely used solution concept for cooperative games that addresses the problem of allocating a fair share of the total payoff to each player. It takes into account that each player can interact with all other players in a symmetric manner. The Shapley value calculates the marginal contribution of each player based on their interactions with different coalitions.

To compute the Shapley value for a player i, we consider all possible coalitions S that can be formed without player i. For each coalition S, we evaluate the difference between the characteristic function's value for the coalition $S \cup i$ and the characteristic function's value for the coalition S . This difference captures the marginal contribution of a player i when joining the coalition S. Mathematically, it can be expressed as $m(i, S) = v(S \cup \{i\}) - v(S)$.

The Shapley value considers all possible permutations of player arrivals and calculates the average marginal contribution over all permutations. This ensures fairness in the distribution of the total payoff among players by giving each player their rightful share based on their individual contributions.

By computing the Shapley value for each player, we obtain a vector that represents the fair allocation of the total payoff. Each component of the vector corresponds to the share attributed to a specific player, reflecting their unique contributions when interacting with different coalitions.

The Shapley value compute a "fair" distribution in the sens that it is the only distribution with certain desirable properties. More formally, according to the Shapley value, the amount that the player j gets is

$$\phi_j(v) = \phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(M - |S| - 1)!}{M!}(v(S \cup \{j\}) - v(S)), j = 1, .., n \tag{2.5}$$

Therefore, the Shapley value is a weighted mean over all subset S of players that does not contain j, including $S = \emptyset$.

The properties that the Shapley values satisfies are the following:

Efficiency:

$$\sum_{j=0}^{n} \phi_j = v(N) \tag{2.6}$$

It means that all the total gain is distributed among players.

Symmetry: If for every subset S that does not contain i and j, $v(S \cup \{j\}) = v(S \cup \{i\})$, then the Shapley value of i and j are identical: $\phi_i = \phi_j$. Players i and j contribute equally to the game.

Dummy player: If $v(S \cup \{j\}) = v(S)$ for a player j and all coalition $S \subseteq N \setminus \{j\}$, then $\phi_j = 0$.

Linearity: If two coalition games described by gain functions v and w are combined, then the distributed gains correspond to the gains derived from v and the gains derived from w: $\phi_i(v+w) = \phi_i(v)+\phi_i(w)$, for every i. Also, for any real number, x we have that $\phi_i(av) = a\phi_i(v)$.

In machine learning, the Shapley value can be used to perform an individual prediction's explanation Aas et al. (2020). It was firstly used in classical machine learning model, dealing with tabular data and clearly identifiable features, but then it has been applied to deep learning, including GNN explanation. Perotti et al. (2022) and Duval and Malliaros (2021) both use the Shapley value in different ways to build instance-level explanation.

2.4.3 SHAP

The goal of SHAPLundberg and Lee (2017) is to explain the prediction of an instance x by computing the contribution of each feature to the prediction. The SHAP explanation method computes Shapley values from coalitional game theory. The feature values of a data instance act as players in a coalition. Shapley values tell us how to fairly distribute the "payout" (the prediction) among the features. A player can be an individual feature value, e.g. for tabular data. A player can also be a group of feature values. For example to explain an image, pixels can be grouped to superpixels and the prediction distributed among them. One innovation that SHAP brings to the table is that the Shapley value explanation is represented as an additive feature attribution method, a linear model. That view connects LIME and Shapley values. SHAP specifies the explanation as:

$$g(x') = \phi_0 + \sum_{j=1}^{n} \phi'_j \tag{2.7}$$

where g is the explanation model, x' a perturbation of the instance x using coalitions.

The KernelSHAP estimates for an instance x the contributions of each feature value to the prediction, by sampling some coalitions instead of computing the Shapley value using all coalition $S \subseteq N$. To get from coalitions of features z' to valid data instances, we need a function $h_x(z') = z$ where $h_x : \{0,1\}^n \to \mathbb{R}^p$. The function h_x maps 1's to the corresponding value from the instance x that we want to explain. For tabular data, it maps 0's to the values of another instance that we sample from the data. This means that we equate "feature value is absent" with "feature value is replaced by random feature value from data".

SHAP weights the sampled instances according to the weight the coalition would get in the Shapley value estimation. Small coalitions (few 1's) and large coalitions (i.e. many 1's) get the largest weights. To achieve a Shapley compliant weighting, Lundberg and Lee (2017) propose the SHAP kernel:

$$\pi_{x'}(z') = \frac{n-1}{\binom{n}{|z'|}|z'|(n-|z'|)} \tag{2.8}$$

with $|z'|$ the number of present features in z'. Lundberg and Lee (2017) sho that a linear regression with this kernel weight yields Shapley values.

KernelSHAP estimates the Shapley value by computing the value function v on various coalition and then compute the weighted regression using π_x to optimize the following loss function:

$$L(f, g, \pi_{x'}) = \sum_{z' \in Z} [f(h^{-1}(z')) - g(z')]^2 \pi_{x'}(z') \tag{2.9}$$

with f is the to be explain model.

In this report, we will inspire ourself from Shapley value and the KernelSHAP to adapt it to our problem.

Chapter 3

Method

In previous work, we introduce a method to extract important subgraphs for the activation of an activation rule, allowing to better understand what each activation rule captures. In this report, we propose a method to compute contribution of each activation rule to the model prediction using Shapley value. In our case, the set of players N is the set of activation rules extracted of the model by InsideGNN, and the result for which we need to compute the contribution is the model predictions on the dataset. SHAP use the Shapley value to compute contribution of features. It's perturbs the instance x to estimate an importance score of features. The challenge here is that we want to compute importance score of some specific parts of our model, therefore we perturb the model instead of perturbing one instance. With this come the need to adapt the Shapley value and the SHAP to our problem.

3.1 Exclude a player from the game

To compute the value function for a coalition S, we need to "exclude" players from the game. When we compute $v(S)$, we "exclude" every player that are not in the coalition S. When we are dealing with tabular data, with features as players, we can replace an excluded feature with a default value, the mean, a random value from other instances. We can extend this on all kind of data, when we are computing the contribution of a portion of an instance to the prediction of a model, there is a lot of ways to exclude the player by perturbing the instance. In our case, to exclude a player means excludes some activation rules of the model. As an activation rule is a set of neurons that are commonly activated together, we can choose to deactivate concerned neurons of a rule by multiplying their output by 0. Thereby, the neurons concerned by the activation rule will not transmit any of the information they captured from the instance. It is efficient since it can be done by just multiplying the output of a forward pass with a binary mask, but it might be excessive. Another approach we consider is to replace the actual output of the neuron by the mean output of the whole dataset for this neuron. Even if it seems more fair than multiplying the output of a neuron by 0, it is also more costly. First, it needs to compute the mean output for each neuron, implying a forward pass on the whole dataset. Then, replacing the output values in the forward pass when the model is perturbed is also less efficient than a vector multiplication. The process described in this subsection is similar to the function h_x described in SHAP, which is used to transform the set of players to a coalition.

3.2 Value function

To adapt Shapley value to our problem, we also need to define an appropriate value function. One solution is to use the mean prediction for a targeted class c:

$$v(S) = \frac{\sum_{x \in \mathcal{D}} [f_S(x)]^c}{m}$$

where $\mathcal{D}$ is the dataset, m its size, and f_S the perturbed model. This implies that we can compute the contribution of activation rules for only one class for each defined game. The second option is to consider the difference between the perturbed model and the original model:

$$v(S) = \frac{\sum_{x \in \mathcal{D}} [f(x)]^c - [f_S(x)]^c}{m}$$

The intuition between both value function proposed here is that, if an activation rule is actually important for the model prediction of a given class, the fact to exclude it of the game should involve an important variation of the probability prediction of the model, if not, it is either because other neurons captures similar information as those include in this activation rule, either because this activation rule is not that significant in the model prediction.

Moreover, since extraction of the activation rules already target a specific class, we can split the activation rule set A in as much subset as there is class for the instances. Thus, we can consider two different games (in case of binary classification) , one for each class, in which we will only consider activation of the targeted class. This allows to reduce the number of players in each game and compare activation rules on their contribution of their targeted class. Computing the contribution to class 0 of an activation rules that has been extracted by targeting class 1 is not necessary.

So far, we have considered that we need to compute the importance score of an activation rule of the model for the whole dataset to try to extract interesting information what the model learned in general. We can also consider the same method to apply it for each instance independently and compute the contribution of each activation rule for a given instance x. This way, we could build a set of important activation rule used by the model to make a decision.

To summarize the method, here are the important step of the process, with the model, dataset and activation already extracted:

1. Sample k coalitions
2. Get prediction for each coalition S using f_S
3. Compute the weight of each coalition using π
4. Fit the weighted linear model
5. Return the estimated contribution, which are the coefficients of the linear model

Chapter 4

Experiments

To evaluate the method and the different strategy we described, we train graph classification models on *Mutag* and *AIDS* datasets from Morris et al. (2020) which are two real-world dataset. The MUTAG dataset is 'a collection of nitroaromatic compounds and the goal is to predict their mutagenicity on Salmonella typhimurium'. It contains 188 graphs with a maximum size of 28 nodes. The AIDS dataset is a dataset containing compounds checked for evidence of anti-HIV activity, containing 2000 graphs with an average number of 15 nodes. We train one model for each dataset with three hidden layers with a dimension of 128 each. Once trained, we extracted activation rules using InsideGNN method, which extract 46 rules for MUTAG and 53 for AIDS.

In this experiment, we will compare execution time between deactivation and replacement of output of a neuron, we will also compare our score with interestingness scores computed by InsideGNN, and study the distribution of rules score when we consider each instance indenpendently.

The figure 4.1 represent the execution time of the execution time need to compute contribution of rule for both class on the mutag dataset, showing the difference between the two strategies: replacing or multiplying by 0 the output of neurons. As presented before, the implementation required to replace output leads to a slower execution time. Is this experiment we use $k \in \{100, 500, 1000\}$, if we want to use more coalition for the contribution estimation, the method "replace" seems to not be efficient at all since its computation time become quickly too important for this task compared to the "deactivate" method.

The figure 4.2 presents the contribution to class 0 and 1 for each activation of the mutag model for both player exclusion strategy. The expected and perfect result would be two clusters, one for each class, that maximize the contribution score for the class that the activation rule target according to InsideGNN. The results are far from this, the method even gives negative contribution to class 1 for rules that target class 1. If the targeted class given by InsideGNN are relevant, the result of our method (with both strategy) are not meeting our expectation. Moreover, the results from both strategy are significantly different. Both strategy have results that are not convincing, thus it is difficult to make conclusion about the difference of execution time.

Figure 4.3 shows the contribution to prediction for each activation rule using "deactivate" strategy with $k = 100$ with interestingness computed by InsideGNN as color. Again, we can see that some rules with a high interestingness score got a contribution score close to 0 for both class, even though some other important rule got a good contribution score on one of the two classes. To confirm this, the figure 4.4 shows the lack of correlation between the contribution computed by our method and the interestingness score of InsideGNN.

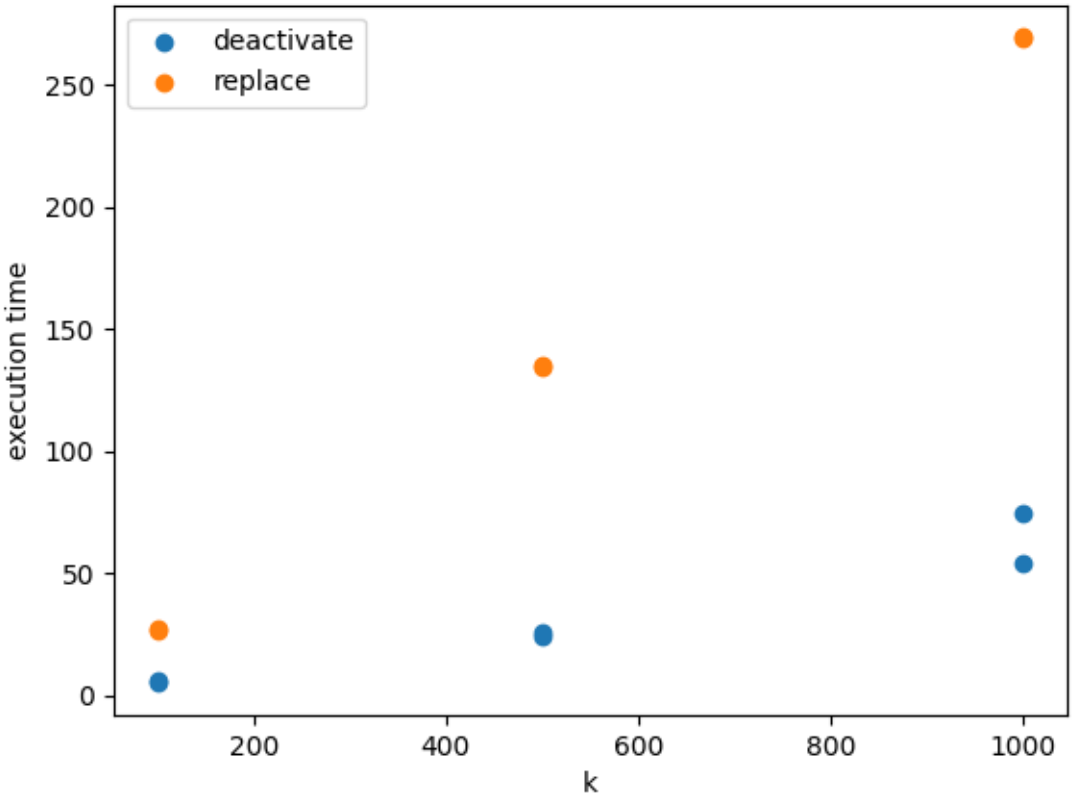

Figure 4.1: Execution time for our method for different number of coalition sampled (k)

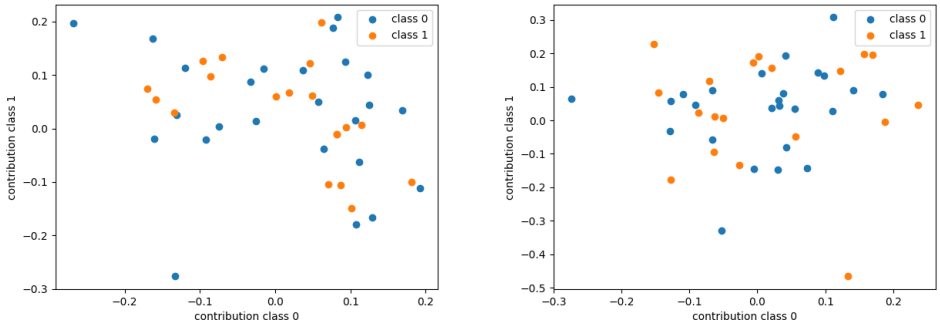

Figure 4.2: Contribution to class 0 and 1 computed by our method for $k = 1000$ with "deactivate" strategy (left) and "replace" strategy (right) with class targeted by the rule as color.

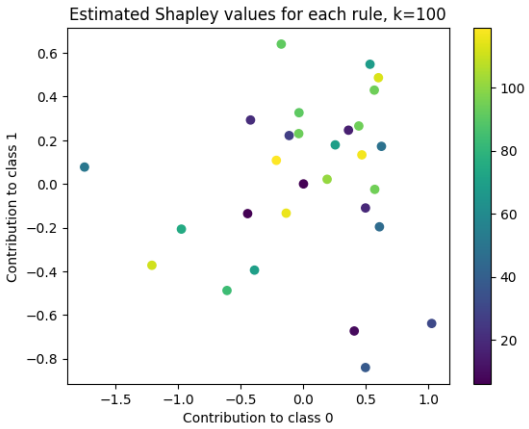

Figure 4.3: Contribution to class 0 and 1 for each activation rule on the mutag dataset with interestingness of InsideGNN as color

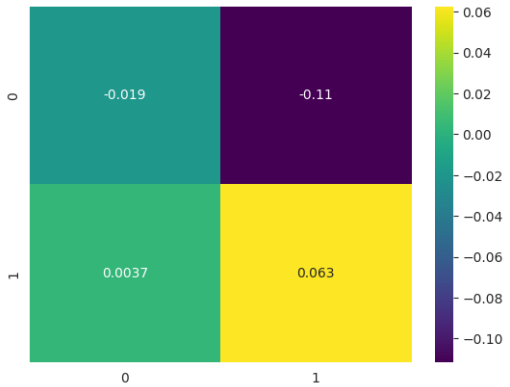

Figure 4.4: Correlation between contribution computed by our method and the interestingness score computed by InsideGNN ("deactivate" strategy, k=100)

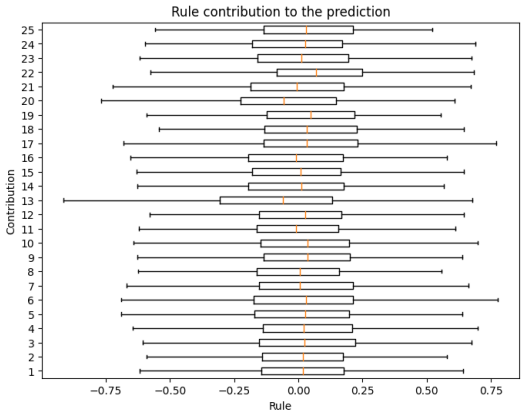

Figure 4.5: Distribution of rule score for each instance of the dataset for each rule using our contribution score method for the class 0

Finally, we run our method on each instance of the dataset instead of the whole dataset all in once to get the rule score for each instance, as an instance-level explanation method would do it. As shown in figure 4.5, most of the median score for each rule is close to 0, which can be expected almost half of the instances of the dataset are classified as class 1, which is not the target of the 25 rule studied here. However, for some instances, some rules seem to be really important with either a negative or a positive contribution score that tends to 1.

Note that we only discuss here the result on the MUTAG dataset, but the same conclusions can be drawn from the AIDS dataset.

Chapter 5

Discussion

In this report, we present a new method that use cooperative games, especially the Shapley value, to compute the rule importance for a given class in a graph classification problem. To do so, we propose to adapt the SHAP method to consider activation rules as players using several strategies for the player exclusion and the value function of the cooperative games. The experiments show the experiments show unsatisfactory results, that are not correlated to the interestingness score extracted by the algorithm which computes activation rules of a model. Considering this method to perform instance-level-like rule importance shows slightly interesting results, but it is not the goal of our approach.

The problem of our method could be either the exclusion method, which involve deactivating a neuron or replacing its output by another value, or from the value function. Another possible problem is the fact that activation rule could overlap each other. One component of the activation vector could be common in some activation rules. This can perturb the exclusion of a rule if a component is shared with a rule that is in the coalition and another that is not. To face this issue, we choose to keep all neurons from rules in the coalition activated, but this might be not the best solution. In future work, we could try to randomly deactivate neurons that are common between two rules that should not be activated together in the perturbed model. Otherwise, the fact that rules (player of the game) can overlap might be inconsistent with cooperative games, which that Shapley value does not work in our case. Therefore, trying to model the problem in another way to avoid overlap between players could be a good solution to study. Due to the complexity of the hidden behavior of a deep learning model, it is difficult to verify if our method satisfies the property of the Shapley value. Another future work would be to study extensively the impact of the hyperparameter k that define the number of coalition to sample to perform the contribution estimation.

Chapter 6

Bibliography

Aas, K., Jullum, M., and Løland, A. (2020). Explaining individual predictions when features are dependent: More accurate approximations to shapley values. (page 9)

De Bie, Tijl (2011). An information theoretic framework for data mining. In *17th ACM SIGKDD international conference on Knowledge discovery and data mining, Proceedings*, pages 564–572. ACM. (page 6)

Duval, A. and Malliaros, F. D. (2021). Graphsvx: Shapley value explanations for graph neural networks. (page 9)

Kamal, A., Vincent, E., Plantevit, M., and Robardet, C. (2023). Improving the quality of rule-based gnn explanations. In Koprinska, I., Mignone, P., Guidotti, R., Jaroszewicz, S., Fröning, H., Gullo, F., Ferreira, P. M., Roqueiro, D., Ceddia, G., Nowaczyk, S., Gama, J., Ribeiro, R., Gavaldà, R., Masciari, E., Ras, Z., Ritacco, E., Naretto, F., Theissler, A., Biecek, P., Verbeke, W., Schiele, G., Pernkopf, F., Blott, M., Bordino, I., Danesi, I. L., Ponti, G., Severini, L., Appice, A., Andresini, G., Medeiros, I., Graça, G., Cooper, L., Ghazaleh, N., Richiardi, J., Saldana, D., Sechidis, K., Canakoglu, A., Pido, S., Pinoli, P., Bifet, A., and Pashami, S., editors, *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 467–482, Cham. Springer Nature Switzerland. (page 6)

Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. (page 9)

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. (2020). Tu-dataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. (page 12)

Perotti, A., Bajardi, P., Bonchi, F., and Panisson, A. (2022). Graphshap: Motif-based explanations for black-box graph classifiers. (page 9)

Veyrin-Forrer, L., Kamal, A., Duffner, S., Plantevit, M., and Robardet, C. (2021a). On GNN explanability with activation patterns. working paper or preprint. (page 6)

Veyrin-Forrer, L., Kamal, A., Duffner, S., Plantevit, M., and Robardet, C. (2021b). What does my GNN really capture? On the exploration of GNN internal representations. working paper or preprint. (pages 6 and 7)

Yuan, H., Tang, J., Hu, X., and Ji, S. (2020). XGNN: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. ACM. (page 6)

Yuan, H., Yu, H., Gui, S., and Ji, S. (2022). Explainability in graph neural networks: A taxonomic survey. (page 5)