TEAM MEMBERS
Anurag Ramteke - 150101010
Sarthak Tripathi - 150101057

Layerwise Relevance Propagation on Graph Convolutional Networks

Abstract

We will be tackling the problem of interpretability of a specific kind of network called as [2]Graph Convolutional Network *(GCN)*. There has been very little work done concerning the interpretability of GCNs. We intend to use [1]Layer-wise Relevance Propagation *(LRP)* which is a type of interpretability algorithm used to interpret CNN input features with heatmap[3] on the GCN to achieve our aim. So our primary focus is to interpret how different features of our dataset affects the result of the *(GCN)*. This network is different from normal Convolutional Neural Network because here instead of a set of layers, a cascade of graphs are used for the training. We decided to use GCN because GCNs are very powerful neural network architecture for machine learning on graphs. [2]In fact, these neural networks are so powerful that even a randomly initiated 2-layer GCN can produce useful feature representation of nodes in networks.***(something about results and conclusion --will be written at the end after writing the report)*.**

Introduction

Deep neural network excels in many things related to classification as well as recognition to name a few. However, utilization of the graph consisting of network of nodes instead of simply uncorrelated nodes in a layer gives rise to another type of neural network which is GCN. GCN will be able to use directly the data which is in the form of graph or involves relation between different nodes into the network. This poses another problem for interpretability of different kind of neural network which is more complicated than a convolutional neural network. However, the application of GCN in real-world datasets like knowledge graphs, social networks, the

World Wide Web, protein-interaction networks which consist of only a small number of application of GCN motivates us to interpret its decision making. We will be discussing about the interpretability of the GCN in this report.

We will be using Layer-wise relevance Propagation (LRP) to interpret the amount of contribution of each feature of the node towards its classification done by network. Any conclusion drawn or classification done by humans are based on reasoning. However, in case of neural network, the reasoning done by network is not clear. So, interpreting the neural network to find the reasoning behind its working plays an important role in many applications including object classification, facial recognition, etc. However, the interpretability plays a major part especially when it involves the sensitive work in medical field like diagnosing the disease. We will be doing visual interpretation of the features of the data using heat-map which will give the relevance of the data towards its contribution. The lighter color shows more weightage while the darker shades interpret the diminishing or less importance of the corresponding features towards the said classification of the data by the GCN.

The rest of the report has been structured as follows. ………………………..***(at the end)***

Background

Our Algorithm as its name suggests “Interpretability of GCN with LRP” deals with two major parts. One being our neural network which is Graph Convolutional Network (GCN) and another is our extended Layerwise relevance propagation which is our interpretability algorithm. To continue with our proposal of algorithm, a little background of both GCN and normal LRP is required. Following the background of the said, we will be continuing with our proposal.

1) Graph Convolutional Networks

They introduced a simple and well-behaved layer-wise propagation rule for neural network models which operate directly on graphs and showed how it can be motivated from a first-order approximation of spectral graph convolutions.

$$H^{(l+1)} = \mathrm{ReLU}(D'^{-0.5} A' D'^{-0.5} H^{(l)} W^{(l)})$$

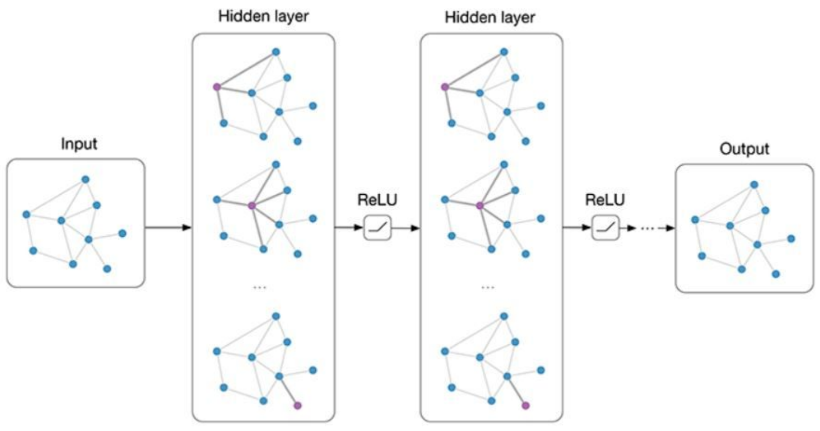
Where $A' = A + I$, $D'_{ii} = \Sigma_j (A'_{ij})$

For these models, the goal is then to learn a function of signals/features on a graph
G=(V,E) which takes as input:

- A feature description x_i for every node i; summarized in a nXd matrix (n is the number of nodes and d is the number of input features for each node).
- A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix A.

and produces a node-level output Z (nXf matrix where f is the number of output features per node).

We could use $H^{(l+1)} = \text{ReLU}(A H^{(l)} W^{(l)})$ but this has limitations. Multiplication with A excludes the node itself, which is corrcted by self loops. Another limitation is normalization. A changes the scale of feature vectors. We could use $D^{-1} A$ to normalize but in practice $D'^{-0.5} A' D'^{-0.5}$ this works better.

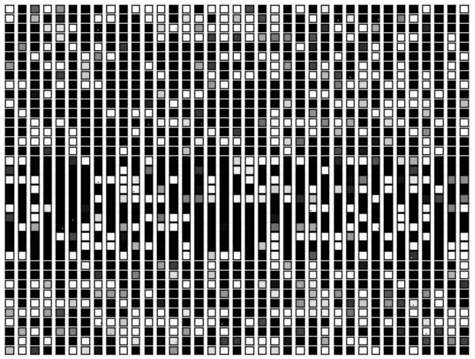


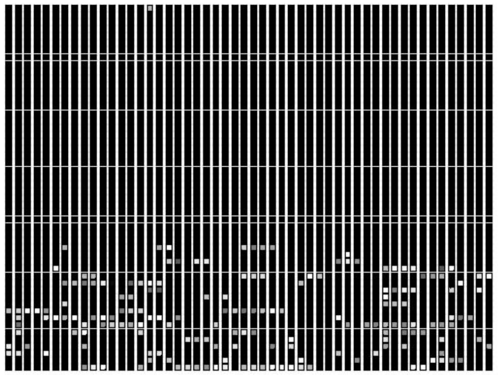
Multi-layer Graph Convolutional Network (GCN) with first-order filters.

2) Layerwise Relevance Propagation(LRP)

LRP is a graph interpretability algorithm which basically computes the score for every feature denoting the impact of those features on the prediction of our trained classifier. This interpretability can also be visualised by carving a heatmap denoting the weight of every feature by the brightness of the corresponding block in the heatmap. However, we will be using extended LRP which will be introduced in further sections.

A trained deep neural network is simply a feed forward neural network where the input is given and the network uses its weight to classify the input. So, LRP exploits this property of the neural network. The following equation $x_j^{(l+1)}=g(0,\Sigma_i x_i^{(l)} w_{ij}^{(l,l+1)}+b_j^{(l+1)})$ is the feed-forward equation. The same equation shows that it can be used to redistribute the relevance f(x) at the output of the network onto feature-wise relevance score $\{R_p^{(l)}\}$, by using a local redistribution rule $R_i^{(l)}=\Sigma_j((z_{ij}/(\Sigma_{i'} z_{i'j}))R_j^{(l+1)})$ where $z_{ij}=x_i^{(l)}w_{ij}^{(l,l+1)}$ where i is the index of a neuron at some particular layer l, and where Σ_j runs over all upper-layer neurons to which neuron i contributes. After classification a backward pass of this rule generates a heat map that satisfies the property $\Sigma R_p^{(1)}=f(x)$. The heat map generated will help us to interpret the major deciding features in the output of the classifier by seeing the weight shown by brightness of the pixel where higher brightness representing more weight. The figures below show the heatmap for the label 3 of the cora dataset Where the former image shows the heatmap for nodes for each feature and later shows the heatmap for features for each node as generated by our algorithm.





Proposal

We propose the application of Layerwise Relevance Propagation(LRP) on Graph Convolutional Networks(GCNs). This will help understand how the features contribute towards the final classification of the input graph and help in reason the output given by the Graph Convolutional Networks in terms of the input features. The amount of contribution of the each input features act as a reliable explanation for the network's output because feature represents the decision factors which are real world human perceivable attribute. The feature which amounts to the human perception which are weighted according to the relevance towards the neural network's output works as a good explainability behind the network's outcome.

Algorithm

We first define the formula for calculation of relevances of a layer from its previous layer. For that we define z_{ijkl} which is the contribution of j^{th} feature of the i^{th} node to the l^{th} feature of k^{th} node of the next layer.

For GCN

$$X^{(l+1)} = \text{ReLU}(\ D'^{-0.5}\ A'\ D'^{-0.5}\ X^{(l)}\ W^{(l)}\ ) = \ \text{ReLU}(\ C\ X^{(l)}\ W^{(l)}\ )$$

Where $A'=A+I$, $D'_{ii} = \Sigma_j(A'_{ij})$, $C = D'^{-0.5}\ A'\ D'^{-0.5}$

$$z_{ijkl}=c_{ik}\ x_{kl}\ w_{lj}$$

Where c_{ik} is from the matrix C, x_{kl} is from the input X and w_{lj} is from the weight matrix W, all for that layer.

C is an nXn matrix (n is the number of nodes), X is a nXd matrix(d is the number of features in input layer) and W is the dXf matrix(f is the number of features in output layer).

After expanding the matrix we get

$$x_{ij}^{(l+1)} = \Sigma_l \left(\left(\Sigma_k\, c_{ik}\, x_{kl}^{(l)}\right) . w_{lj}\right) = \Sigma_k \left(c_{ik} . \left(\Sigma_l\, x_{kl}^{(l)}\, w_{lj}\right)\right)$$

Thus from $x_{kl}^{(l)}$ there is contribution of $c_{ik}\, x_{kl}\, w_{lj}$ to $x_{ij}^{(l+1)}$

$$R_{kl}^{\,l} = \Sigma_i \Sigma_j \left(z_{ijkl} / \left(\Sigma_{k'}\, \Sigma_{l'}\, z_{ijk'l'}\right)\right) R_{ij}^{\,l+1}$$

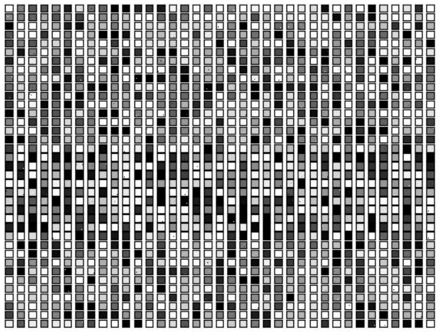
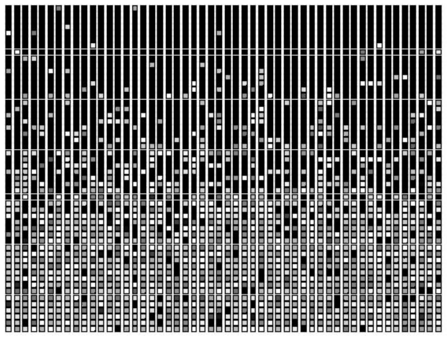
Where $R_{kl}^{\,l}$ is the relevance for kth feature of lth node and $R_{ij}^{\,l+1}$ is the relevance for ith feature of jth node of the previous layer.
This formula can be derived simply by proportioning. If a node has relevance x and we need to propagate the relevance to previous layer we use proportions. So if a node in previous layer contributes c_i it would get (c_i / c) x relevance where $c = \Sigma_i\, c_i$. Similarly we apply for nodes and features.

Results

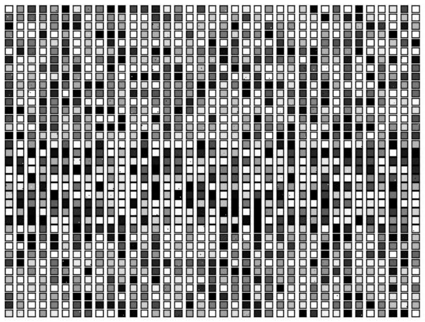
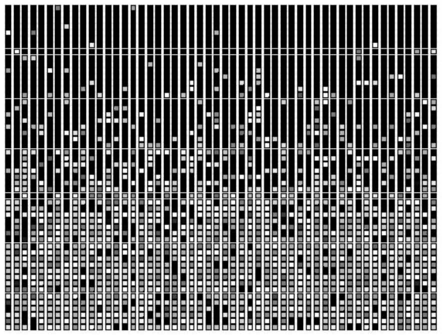
We ran GCN on cora dataset and then after the model trained, we ran relevance propagation in backward direction (starting from output layer). Initially the relevance for last layer is same as output (Similar to the model used in [1]).

The relevance propagation thus obtained contributes to all the node classification. To obtain how the relevance of a node's features affects its classification we only allow it to remain non-zero. Thus we turn all others off.

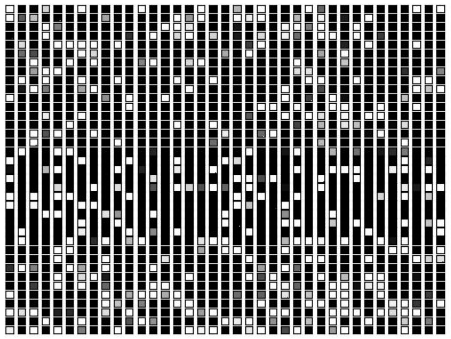
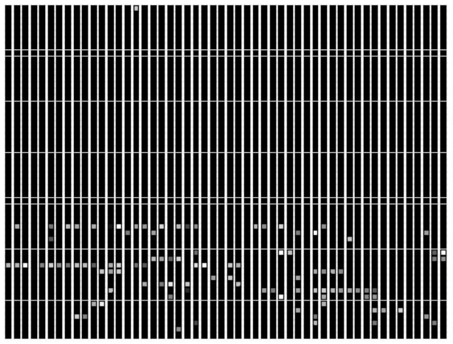
To analyse the relevances, we convert the the features of a node to a square and plot it as a heatmap. We then print such heatmaps for all the nodes. Also, we try to see how features affect nodes. Thus for each feature, we plot the heatmaps of its relevances for all nodes.

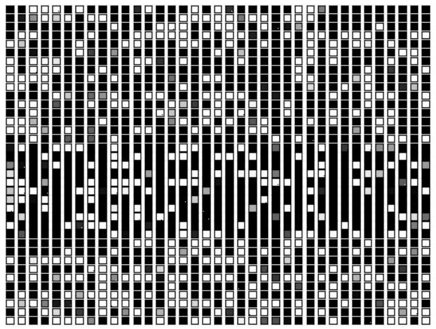
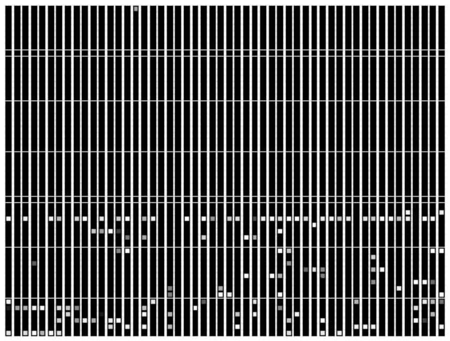
Using all the predicted outputs

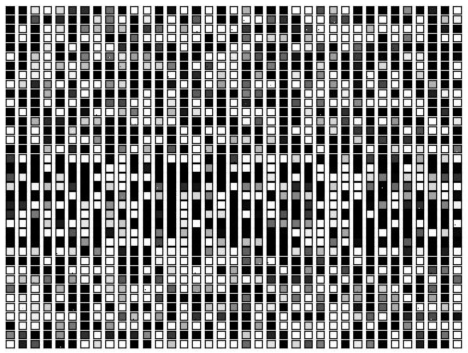
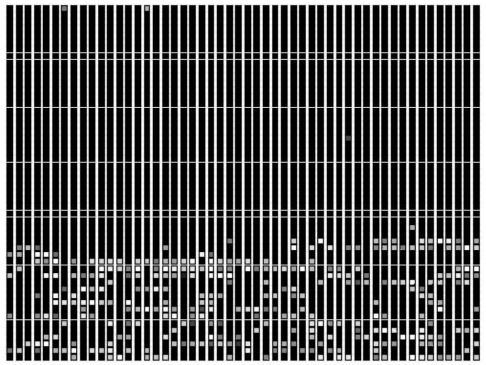
Using only those outputs whose class label is 0

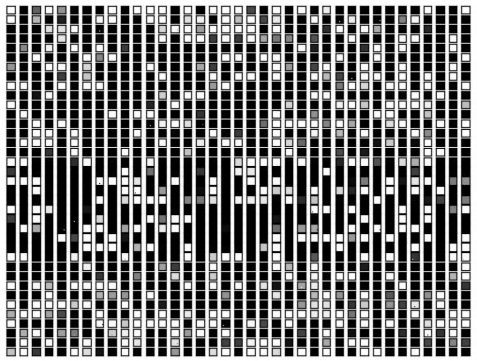
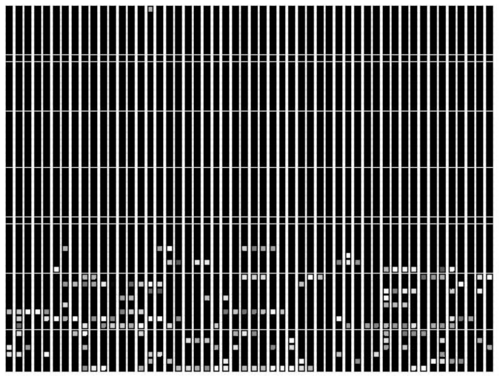
Using only those outputs whose class label is 1

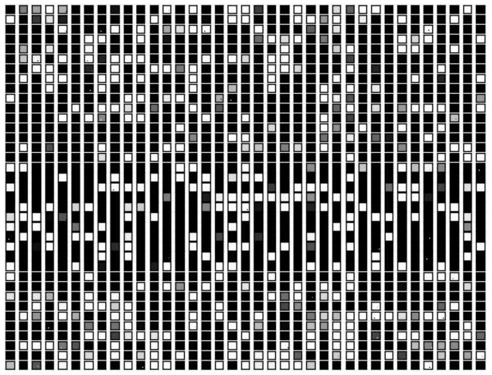
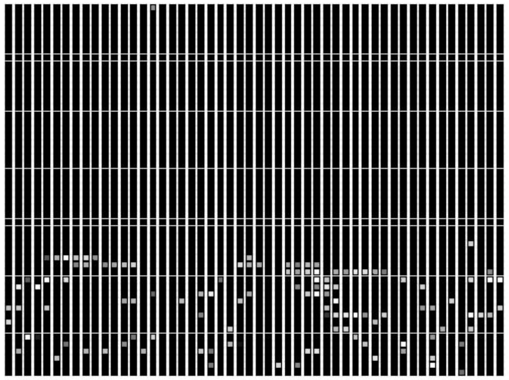
Using only those outputs whose class label is 2

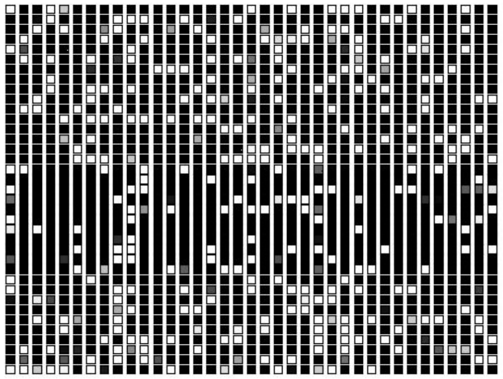
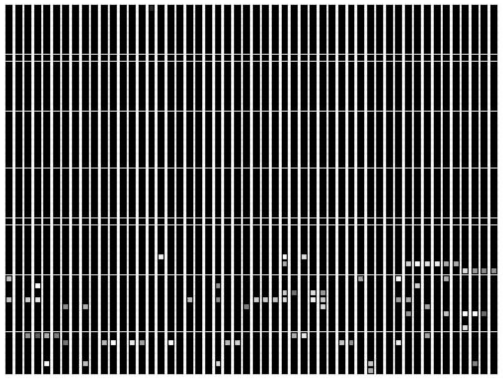
Using only those outputs whose class label is 3

Using only those outputs whose class label is 4

Using only those outputs whose class label is 5

Using only those outputs whose class label is 6

Where the first image is heatmap for nodes for each feature and second is the heatmap for features for each node.

For all nodes that have same class, we found the feature that has highest relevance. This would help to understand how each feature is affecting the classification. This was again done once for the output and rest for when output for one class is enabled.

```
Class  0 enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 25), (1263, 14), (874, 7), (1262, 6), (1177, 5)]
Top 5 Highest Relevance Features for Class  1 ->[(581, 19), (0, 14), (1263, 13), (1177, 5), (1262, 3)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 37), (1263, 13), (1262, 11), (1230, 4), (495, 4)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 66), (1263, 36), (19, 19), (1381, 7), (1177, 7)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 27), (1263, 11), (1143, 5), (1381, 4), (19, 4)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 20), (1263, 12), (1071, 4), (1397, 3), (1262, 3)]
Top 5 Highest Relevance Features for Class  6 ->[(0, 11), (1263, 8), (619, 3), (1205, 2), (1071, 2)]


Class  1 enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 124), (1263, 1), (1071, 1), (507, 1), (398, 1)]
Top 5 Highest Relevance Features for Class  1 ->[(581, 20), (0, 14), (1263, 11), (168, 5), (1262, 4)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 140), (368, 2), (625, 1), (19, 1)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 307), (1263, 2), (1397, 1), (1219, 1), (1177, 1)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 144), (455, 2), (860, 1), (774, 1), (19, 1)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 101), (1263, 1), (969, 1)]
Top 5 Highest Relevance Features for Class  6 ->[(0, 64)]


Class  2 enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 130)]
Top 5 Highest Relevance Features for Class  1 ->[(0, 89), (1263, 1), (581, 1)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 40), (1263, 12), (1262, 10), (1230, 4), (510, 4)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 316), (1263, 1), (812, 1), (1, 1)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 149)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 101), (1211, 2)]
Top 5 Highest Relevance Features for Class  6 ->[(0, 64)]


Class  3 enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 120), (1263, 4), (1376, 2), (1381, 1), (758, 1)]
Top 5 Highest Relevance Features for Class  1 ->[(0, 85), (1263, 3), (581, 1), (168, 1), (65, 1)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 139), (19, 2), (1263, 1), (1262, 1), (378, 1)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 58), (1263, 38), (19, 19), (1381, 8), (1290, 5)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 138), (1263, 2), (1345, 1), (1254, 1), (1209, 1)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 101), (1397, 1), (168, 1)]
Top 5 Highest Relevance Features for Class  6 ->[(0, 64)]


Class  4 enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 124), (1320, 1), (1290, 1), (1177, 1), (615, 1)]
Top 5 Highest Relevance Features for Class  1 ->[(0, 84), (1263, 2), (1177, 2), (581, 2), (336, 1)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 144)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 302), (1263, 2), (19, 2), (1353, 1), (1305, 1)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 21), (1263, 12), (1381, 6), (1143, 5), (19, 4)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 103)]
Top 5 Highest Relevance Features for Class  6 ->[(0, 62), (1171, 1), (464, 1)]


Class  5 enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 129), (1262, 1)]
Top 5 Highest Relevance Features for Class  1 ->[(0, 86), (1263, 3), (1177, 1), (507, 1)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 141), (814, 1), (510, 1), (507, 1)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 315), (1325, 1), (1263, 1), (812, 1), (19, 1)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 147), (393, 1), (132, 1)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 18), (1263, 11), (1071, 6), (1351, 4), (1397, 3)]
Top 5 Highest Relevance Features for Class  6 ->[(0, 57), (1263, 3), (1397, 1), (783, 1), (505, 1)]


Class  6 enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 130)]
Top 5 Highest Relevance Features for Class  1 ->[(0, 91)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 141), (814, 1), (510, 1), (507, 1)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 318), (393, 1)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 148), (393, 1)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 97), (1211, 2), (1397, 1), (1263, 1), (336, 1)]
Top 5 Highest Relevance Features for Class  6 ->[(0, 10), (1263, 9), (619, 3), (507, 3), (1262, 2)]


All Classes enabled
Top 5 Highest Relevance Features for Class  0 ->[(0, 25), (1263, 14), (874, 7), (1262, 6), (1177, 5)]
Top 5 Highest Relevance Features for Class  1 ->[(581, 20), (1263, 13), (0, 13), (1262, 4), (1177, 4)]
Top 5 Highest Relevance Features for Class  2 ->[(0, 35), (1263, 13), (1262, 10), (19, 5), (1230, 4)]
Top 5 Highest Relevance Features for Class  3 ->[(0, 51), (1263, 36), (19, 21), (1381, 7), (1177, 7)]
Top 5 Highest Relevance Features for Class  4 ->[(0, 20), (1263, 12), (1381, 6), (1143, 5), (19, 4)]
Top 5 Highest Relevance Features for Class  5 ->[(0, 15), (1263, 12), (1071, 7), (1397, 4), (1262, 3)]
Top 5 Highest Relevance Features for Class  6 ->[(1263, 9), (0, 9), (619, 3), (507, 3), (1262, 2)]
```

Where it is (Feature index, Count)

Analysis

In the heatmaps white signifies higher relevances. The black feature heatmaps for nodes could be due to disabling the node's outputs.It could also be that none of the features have particularly high relevance and contribute equally. Similar the whiteness could mean many features contribute highly towards the node's classification. Similarly for node heatmap for each feature, the black heatmap signify that, that feature is not contributing to any node's classification, or contributing to each equally. A white heatmap signifies that, that feature contributes highly to many nodes.

As seen from the counts, features 0 and 1263 contribute highest relevances frequently and affect classification. Thus these are important features for node classification. This can also be inferred from the heatmaps.

Conclusion (and future work)

We analysed the heatmaps and and counts to find the features that contribute to the node classification.

We could also analyse how the neighbour's features affect a node's classification. We can also do similar analysis for other datasets like citeseer and pubmed.

REFERENCES

1) Alexander Binder, Sebastian Bach , Gregoire Montavon , Klaus-Robert Muller , and Wojciech Samek: *Layer-wise Relevance Propagation for Deep Neural Network Architectures*. [arXiv:1604.00825v1 [cs.CV] 4 Apr 2016]
2) Thomas N. Kipf, Max Welling: *SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS*. [ICLR 2017]
3) Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert M¨uller, and Wojciech Samek, Layer-wise Relevance Propagation for Deep Neural Network Architectures