Node prediction in citation networks - a comparison of classifiers

Saskia de Wit - s1060762

November 2022

1 Introduction

The roles of nodes in a network can be very different. If we can identify the role of different nodes we might be able to better control the outbreak of epidemics, prevent outages in power grids or predict successful scientific publications based on citations. Other than that, if we have more information about a node we might be better able to build recommendation and question-answering systems. But what is the best way to predict the label of a node? Often there are features available for every node. Are the features of a node enough to predict the true class of a node, or is edge information more valuable? To answer this question, a research has been conducted. In this assignment node labels for three selected data sets were predicted using three different classifiers. One of the classifiers purely takes the features of the nodes into consideration, one of the classifiers only checks the node labels of the linked nodes and the last classifier takes both node features as well as the graph structure into consideration. The experiments were conducted on citation networks. The repository can be found on github.

2 Related work

Recently, the focus of network science has shifted from discovering statistical regularities such as small-worldness and scale-free networks to finding the structural organisations such as communities. The interest here is in the roles of the nodes in the network and the meaning of links between nodes [5]. Traditional methods (centrality based or machine learning methods) only consider either network structures or node features to evaluate the role of a node. The central problem here is the incorporation of information about the graph structure into a machine learning model. This is due to the fact that machine learning methods have no straightforward method to encode the high-dimensional information about the graph structure into a feature vector [4]. When using machine learning methods, structural information about the graphs can be added into a feature vector, such as the degree, centrality measures and kernel functions.

These approaches are however limited, as they are inflexible and can be time-consuming to gather.

[1] assumes a partially encoded graph in which a part of the nodes is not labelled yet. They use the information that is already encoded to predict these labels based on machine learning classification tasks: a classifier is trained based on the nodes with labels and applied to predict the labels of nodes that are unlabeled. To do this, features of nodes have to be decided upon for classification (just like in traditional classification tasks). Sometimes these are available (like age, gender and location for social networks), but the presence of a link makes the classification task different. This is due to the fact that in traditional classification tasks independence is assumed, but in networks this independence is disrupted due to links between nodes.

[7] uses Graph Convolutional Networks to predict the class of a node, using both the features of a node as well as the structure of the network to predict the node class.

3 Experimental setup

For this assignment, three data sets were gathered and three methods are described to predict the label of a node. The data sets that were conducted are all citation networks, in which there are papers or URLs linking to each other and there is information available about these nodes. These data sets were chosen because they are similar in their context, but vary in their graph structure. Three methods were chosen to predict the node labels. We thought it would be interesting to see if the difference in results would be big if all classifiers have a different focus. One of the classifiers only looks at the node features, one only looks at the graph structure and one takes both of these attributes into consideration. We would expect that the last method would outperform the other methods. In the rest of this section the data sets and the methods will be discussed.

3.1 Selected datasets

In this section the datsets that were used to conduct the experiments will be discussed.

Cora dataset The Cora datset [6] contains 2708 nodes and 5429 edges. Every node represents a scientific publication and every edge is directed and represents a citation. The scientific publications are classified into one of seven classes (Case Based (298 nodes), Genetic Algorithms (418 nodes), Neural Networks (818 nodes), Probabilistic Methods (426 nodes), Reinforcement Learning (217 nodes), Rule Learning (180 nodes) and Theory (351 nodes)). The classes are not completely balanced. Besides that, a dictionary is made, holding 1433 unique words. For every publication this a vector is made, assigning a 1 if the word from the dictionary is in the publication and 0 otherwise.

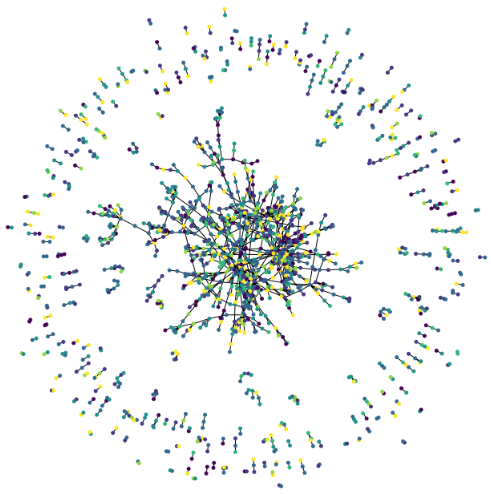

Figure 1: Spring graph of the Cora data. Every color represent a label

In Figure 1 the spring graph of the Cora data set is visualized. The spring graph is like a normal graphical representation of the data, but the nodes with the highest amount of links are placed in the middle. This graph shows one big cluster with a lot of connections, but also a lot of single-connections between two nodes.

Citeseer dataset The Citeseer dataset [3] is similar to the Cora data set. It contains 3312 nodes, which are scientific publications and 4732 edges which describe citations. The publications are classified into one of six balanced classes (AI (249 nodes), Agents (596 nodes), DB (701 nodes), HCI (508 nodes), IR (668 nodes) and ML(590 nodes)). Besides that each publication contains a word vector which is valued by either 0 or 1 of the publication contains a word. The dictionary consists of 3703 unique words.

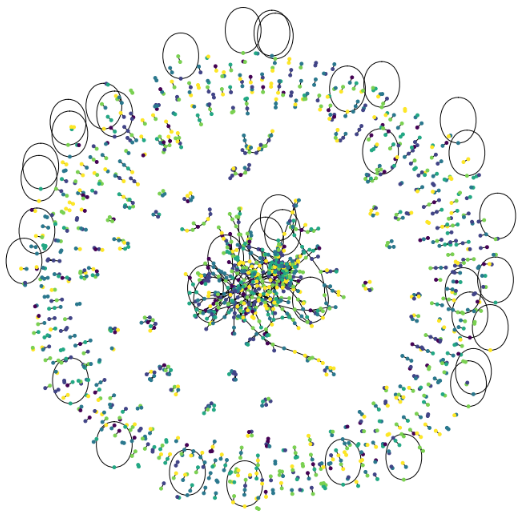

Figure 2: Spring graph of the Citeseer data. Every color represent a label

In Figure 2 the spring graph of the Citeseer data set is visualized. This graph shows one big cluster with a lot of connections in the middle, but also a lot of short-path connections between just a few nodes. On top of this there are some circles visible. These circles represent link from one nodes to itself.

WebKB dataset The WebKB dataset [2] contains 877 web pages represented as nodes. These web pages are categorized into 5 categories (Student (415 nodes), Faculty (125 nodes), Staff (39 nodes), Course (218 nodes), Project (80 nodes)). These categories are imbalanced. An edge between two web papers describes an outgoing link from one web page to another. This dataset contains web pages from computer science departments of universities from Cornell, Texas, Washington and Wisconsin. For the purpose of this exercise the data was merged. On the source of the download of the dataset the description states that the dataset contains 4518 web pages, but when loading the data only 877 appear in the set. These 877 were used for the node classification task.

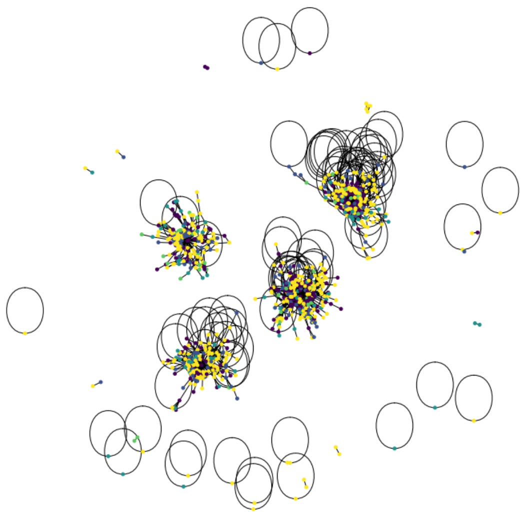

Figure 3: Spring graph of the WebKB data. Every color represent a label

In Figure 3 the spring graph of the WebKB data set is visualized. This graph shows four clusters with a lot of connections in the middle and a few short-path connections between just a few nodes. The four clusters that are visible in the middle represent the different universities and there seem to be no connections between links of different universities. There are also a few nodes that are not connected to one of the components. These mainly show connections between another link or a connection to itself (hence the circle).

3.2 Selected methods

In this section the various methods that were applied are discussed. For every method a train set of 80% of the data was selected and a test set of 20 % was selected.

Nearest-Neighbour-Classifier Also called the K-NN classifier, this method predicts the labels of the nodes purely based on the labels of the K-nearest neighbours.This method does not take the node features into consideration. For this method, we set K equal to 1, which means that only the direct neighbors of the node of interest are considered. For the node of interest, the incoming- and outgoing edges are considered. For every linked node the label is checked and the label that occurs in most of the direct neighbours of the node of interest is assigned to this node.

	Measure	**Cora**	**Citeseer**	**WebKB**
K-NN	*Accuracy*	0.84	0.69	0.13
	F1-Score	0.83	0.67	0.12
Decision tree	*Accuracy*	0.66	0.60	0.69
	F1-Score	0.64	0.55	0.53
GCN	*Accuracy*	0.78	0.75	0.85
	F1-Score	0.77	0.67	0.65

Table 1: Results

Decision tree The decision tree classifier predicts the node labels only based on the features of the nodes. It therefore does not take the edges between nodes into consideration. The decision tree classifier builds a tree based on the training data and afterwards predicts the node labels for the test data. In our data sets the features consist of binary vectors representing words, so the classification is done based on the words in the publication/on the website.

Graph Convolutional Network The Graph Convolutional Network is selected, because it combines both the features of the nodes as well as the edges between the nodes. The Graph Convolutional Networks is applicable to graphs as it adds a graph convolutional layer, which is similar to a convolutional dense layer, but augmented by the adjacency matrix. It therefore also carries information about the node connections. The Graph Convolutional Network is for all data sets trained with a learning rate of 0.01, a dropout rate of 0.5 (0.7 for Citeseer), 300 epochs (but early stopping is also implemented) and d batch size of 256. The Adam optimizer was used for optimization, the loss used is Categorical Cross entropy.

3.3 Evaluation metrics

To elaborate on the effectiveness of the methods, multiple evaluation metrics were selected. The node classification tasks will give as output a predicted label, which can be compared to the true label. To elaborate on the effectiveness of the methods, the accuracy and F1-score were calculated. To gain more information in the prediction, the accuracy matrix was also created. This can be used to see with which class the classifiers struggles.

4 Results

In this section the results of the different methods on the data sets are discussed. In Table 4 the results for the different methods on the different data sets can be found. The results are described in accuracy and in the F1-score. The confusion matrices are not given in this report, but can be found in the github.

5 Conclusion

The results in the previous section show various things. First of all, the Graph Convolutional Network does not always outperform the other methods. For the Cora data set, the K-NN classifier produces the best results, followed by the Graph Convolutional Network. This might tell us that the features of this data set tell us more than the links in the network. The topics that are discussed in this topic are all closely related and have many links to each other.

For the Citeseer data set however, the Graph Convolutional Network outperforms the other methods. As the spring graph of this network shows, there is one 'big' cluster, bus also many smaller clusters. The combination of using features as well as edges could give the result of Graph Concolutional Network outperforming the other methods. We see that the other methods perform decent as well.

Finally, for the WebKB data set the Graph Convolutional Network achieves a high accuracy, but a somewhat lower F1-Score. The performance of the K-NN classifier is however worse than random guessing for this data set. This might be due to the fact that there are multiple smaller components and nodes that do not have any links. These are more difficult to predict for the classifier with no additional information.

6 Discussion

In this work three classifiers for node classification were tested on three different data sets. The data sets show different graph structures and therefore the classifiers show different results. The K-NN classifier performs well in a network in which nodes have many links. The Decision Tree classifier works well in a network in which the nodes have sufficient features and might not have many edges. Finally, the Graph Convolutional Network gives a trade-off between the two. If performs well in a network with a sufficient amount of features as well as link. The overall performance on the WebKb data set was not very good. This might be due to the fact that this data set was smaller and more imbalanced than the other data sets.

In future work it could be interesting to test the different models on other data sets, and see if e.g. social networks show the same results. It could also be meaningful the run the K-NN classifiers for K = 2, so you also look at the links of nodes that are one step further away. Finally, graph data such as degree could be added to the features for the decision tree, the give more information about the graph structure. This was also suggested in literature. Finally, the Graph Convolutional Network could be optimized using diffent settings.

References

[1] Bhagat, S., Cormode, G., and Muthukrishnan, S. Node classification

in social networks. In *Social network data analytics*. Springer, 2011, pp. 115–148.

[2] García-Plaza, A. P., Fresno, V., Unanue, R. M., and Zubiaga, A. Using fuzzy logic to leverage html markup for web page representation. *IEEE Transactions on Fuzzy Systems 25*, 4 (2016), 919–933.

[3] Giles, C. L., Bollacker, K. D., and Lawrence, S. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries* (1998), pp. 89–98.

[4] Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[5] Lü, L., Chen, D., Ren, X.-L., Zhang, Q.-M., Zhang, Y.-C., and Zhou, T. Vital nodes identification in complex networks. *Physics Reports 650* (2016), 1–63.

[6] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval 3*, 2 (2000), 127–163.

[7] Zhao, G., Jia, P., Zhou, A., and Zhang, B. Infgcn: Identifying influential nodes in complex networks with graph convolutional networks. *Neurocomputing 414* (2020), 18–26.