Utilisation of machine learning algorithms incorporating Laplacian and unsupervised methods for the detection of enzymes from protein graphs

Final Report Submission

Jyotishka DAS
CentraleSupélec
jyotishka.das@student-cs.fr

Arun MANIVANNAN
CentraleSupélec
arun.jegathesh@student-cs.fr

Victor HONG
CentraleSupélec
victor.hong@student-cs.fr

Hao XU
CentraleSupélec
hao.xu@student-cs.fr

Abstract

Graph classification in network analysis involves predicting the class label of a graph based on its structure. A graph is defined by a set of nodes and edges that represent a complex system, for example social networks, computer networks, biological pathways, or molecular structures. Graph classification has been addressed using various methods, including but not limited to, graph kernels, neural networks, and random walks. In all cases, the objective is to arrive at accurate and efficient algorithms that accurately map graph structures that represent large-scale datasets with a variety of structures.

In our examination of literature on the subject, we were keen to explore efficient yet accurate algorithms to perform graph classification, and we arrived at incorporating the use of the spectral decomposition of graph Laplacian to perform graph classification. In summarised terms, the graph Laplacian is a matrix that captures the connectivity and structure of a graph. Eigenvalues and eigenvectors of the Laplacian matrix are used to decompose a graph into spectral components. Each eigenvector corresponds to a specific frequency of the graph that can be used to identify substructures within the graph. By selecting a subset of these eigenvectors, the dimensionality of the graph can be reduced and used as input features for machine learning models.

Keywords: Protein, Enzymes, Laplacian, Node2vec, Ensemble Learning

1 Introduction and Motivation

Biological structures can be visualized as a graph using nodes and edges. Nodes can be molecular structures (atoms) and edges can represent the connections between these structures through chemical bonds or spatial relationships. Recent advances in machine learning has enabled us to use graph learning methods to accurately classify biological structures, specifically chemical compounds. Examples of these chemical structures include molecular structures, proteins and enzymes.
A potential application of being able to classify molecular structures is to predict if a protein is enzyme or non-enzyme from its graphical representation. Our goal in this project is to determine if incorporating embedding techniques using Laplacian eigenvalues or random walk based methods would improve the accuracy of detection of enzymes using the Protein Full (PF) dataset, while achieving stable results across a range of machine learning techniques to prove its reliability.

2 Problem Definition

Our goal is to explore the graph properties of protein networks and use different classification techniques to detect enzymes. However, certain difficulties arise when we try to approach the problem. The first difficulty we encountered is how to deal with the complexity and diversity of biological structures, which can affect the accuracy of our classifiers. In addition, taking the database we used as an example, the biological structure of a protein is not static over time, it may change its structure due to external factors: temperature, thereby affecting its characteristics and functionalities. Due to the complexity of the compound structure itself, the accuracy of our model is limited to a certain level.

We wish to study Laplacian-based methods for generating embeddings from the graph to further the classification task. This method aims to minimise the cost function that comes from the Laplacian matrix. The cost function penalises the discrepancy between labels of nodes in the neighborhood. By doing so, the Laplacian method can find a smooth label that is consistent in terms of neighboring connections. The Laplacian method is robust to the noise and the outliers in the datasets. However, the Laplacian method is sensitive to how we present the graph features and demands high computational power.

Moreover, we wish to study a second embedding method to generate representations for the graph classification task: node2vec, which can represent the biological structures as vectors with lower dimensions. By using the node2vec method, it can extract the structural information.

The intention of this project is to explore and compare the performances of the graph classification task using these two different embedding techniques, namely Laplacian-based and unsupervised node2vec methods. Also, the performances of using different machine learning classifier models like Random Forest, XGBoost, SVM, and CatBoost with gridsearch were studied that can generate the final classification results from the embeddings.

3 Related Work

From our research on literature on the topic, graph classification for chemical compounds and molecular structure through network analysis has a recent history that generated serious interest in the early-2000s. The first machine learning models that were developed focused on molecular identification, which is the means to represent chemical compounds as a series of binary or numerical values that record structural or chemical properties. This 'fingerprint' was a mathematical description of molecular structures that can then be used to compare and classify other molecules based on their numerical attributes.

As time progressed, with the increase of computational power and available data, more sophisticated models were developed, including the use of graph convolutional neural networks that were able to create and capture complex representations between atoms and molecules to increase accuracy for molecular property prediction.

Later models became increasingly more sophisticated, with the use of graph neural networks with increased performance. New architectures were developed beyond graph convolutional neural networks, such as graph attention networks (GATs) and graph isomorphism networks (GINs). Newer models include functionalities that enable multiple properties of a molecule to be predicted at the same time, and also transfer learning approaches where already pre-trained models can be fine-tuned for specific predictions.

Within this body of research, we were interested to determine if models existed whereby "simpler" or less computationally intensive models existed and were able to achieve a respectable degree of accuracy when compared to more complex models that required higher computational resources.

The paper of Ian Barnett et al. on feature-based classification of networks gave our team deeper insights in improving the accuracy of graph classification from the domain of social network analysis, particularly through the observation that network representations that classes of networks more or less possess similar features and thus networks of similar purposes of construction within the same class can be classified based on features at the structural level [1]. The paper goes on to describe their approach of combining manual selection of features with existing classification models to improve accuracy of predictions.

Leonardo Gutierrez Gomez et al's paper on dynamics based features for graph classification on the other hand (as opposed to structural features) offered insights on how dynamic features, and not structural features could be used in classifying networks [2].

Thomas Bonald et al's paper gave us a primer and provided supplemental details, both on the theory as well as practical aspects of implementing graph Laplacian, to facilitate our own implementation [3].

Jiaxuan You et. al's paper was written in conjunction with Jure Leskovec and offered insights on the implementation of deep autoregressive models, particularly on the challenges of implementation. We primarily used this paper to understand the process of implementing a graph neural network to serve as a benchmark against our own implementation of a less complex model [4].

Normalized Laplacian eigenvectors are a popular strategy in machine learning applications for graph embedding. They are derived from a graph's normalized Laplacian matrix, which is a matrix representation of the graph structure. The eigenvectors are computed by dividing the Laplacian matrix by the degree matrix and computing the eigenvalues and eigenvectors of the normalized Laplacian matrix.

The Protein Full dataset, which provides a set of protein structures represented as graphs, is a commonly used benchmark dataset in bioinformatics for graph classification applications. On this dataset, Li et al. evaluated the performance of normalized Laplacian eigenvectors with various state-of-the-art protein structure classification techniques [8].

The results indicated that employing normalized Laplacian eigenvectors as graph embeddings outperformed other state-of-the-art techniques by 86.18%. These findings imply that normalized Laplacian eigenvectors are a powerful and computationally efficient method for graph embedding in machine learning applications, particularly graph classification tasks on protein datasets.

4 Methodology

The methodology of the project revolves around two major steps, namely obtaining the embeddings from the graph and then use a classifier to obtain the final accuracy results. The two steps are elucidated below:

4.1 Obtaining Embeddings from the Graph

We studied two methods to obtain the embeddings from the graph. The first method is a Laplacian spectral-based method, and the second one is an unsupervised random walk based node2vec method. These two methods of generating the embeddings are discussed below.

4.1.1 Using the Laplacian Spectrum. The Laplacian matrix is defined as follows. Let $G = (V, E)$ be a simple graph (no loops or multiple edges) with vertex set, V and edge set E. If the adjacency matrix (a matrix that shows which pairs of nodes are connected by edges) is A and the degree matrix (a diagonal matrix that contains the degrees of each node on the diagonal) is D, then the the Laplacian matrix can be represented as:

$$L = D - A \tag{1}$$

A distinct form of the Laplacian matrix, which is called the normalized Laplacian matrix, $\mathcal{L}$ is used for our current study. The normalized laplacian matrix is obtained as:

$$\mathcal{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{2}$$

The Laplacian matrix of a graph and its eigenvalues can be used in several areas of mathematical research and have a physical interpretation in various physical and chemical theories [6]. Spectral graph theory, looking at the eigenvalues of the graph Laplacian, can tell us not just whether a graph is connected, but also how well it's connected. For example, the smallest eigenvalue of G is always zero. The eigenvalue 0 tells us whether G is connected or not. In particular, if G has k connected components, then the eigenvalue 0 has multiplicity k (i.e. k distinct non-trivial eigenvectors). Fiedler, in his landmark monograph showed that the second smallest eigenvector gives the information about the algebraic connectivity of G [7]. If this value is small, this suggests that G is nearly disconnected. In other words, the second eigenvalue gives a sort of continuous measure of how well a graph is connected. Since, the aim of the project was to extract some meaningful informations from G, without sacrificing on the time-complexity of the execution of the model, therefore, the k smallest positive eigenvalues of $\mathcal{L}$ in ascending order were used as embeddings to describe the local structure of the network, as shown in the equation 3.

$$X = (\sigma_1, \sigma_2, ..., \sigma_k) \tag{3}$$

These limited number of eigenvalues as embeddings were able to depict the local structure of graphs in a clear and effective manner, and were denoted as spectral features (SF). These attributes record information on the graph's number of edges, nodes, and connectivity. The SF were also found to be excellent at distinguishing between different types of graphs. These embeddings were then fed to a machine learning (ML) classifier to get the classification result. An overview of the model is shown in Fig.1.

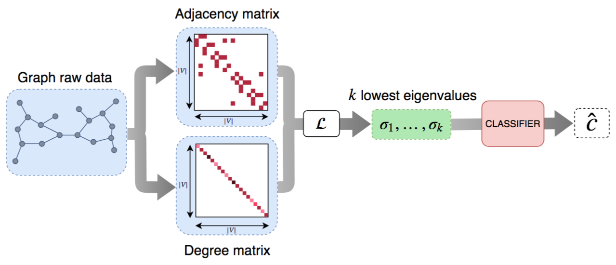

Figure 1. Modelling the Graph Classification Task

4.1.2 Using unsupervised node2vec method. Node2vec is a graph embedding technique that uses random walks to capture the structural information of a graph to develop low-dimensional representations (embeddings) of nodes. The embeddings can then be used for a variety of purposes, including graph categorization.

To explore the usage of this method, the graph was initially translated into a suitable format for use as input to node2vec in the first stage. Typically, this entails expressing the graph as an adjacency matrix or an edge list. To learn the embeddings for the nodes in the graph, we used the node2vec technique. Node2vec works by producing random walks on the graph, which are node sequences that capture the graph's local structure. These random walks are used by the method to learn embeddings that encapsulate the structural features of the graph. After generating the embeddings, they were fed into a classifier for the graph classification job.

4.2 Modelling the classification task

The choice of the classifier can significantly affect the accuracy of the final classification results. We investigated three major machine-learning based classifier for this project:

4.2.1 Random Forest (RF). Random forest is a widely supervised machine learning method for classification and regression that involves employing many decision trees and merging the predictions of the trees into an overall prediction. To train the random forest, each of its decision trees must be trained independently. Each decision tree is typically trained on a subset of the training set and may use alternative attributes for node splits.

The assumption is that the differences in how each decision tree is trained would help minimize overfitting, which is typical when only one decision tree is trained on the complete training set. The method of mixing several predictors (in this example, decision trees) is known as ensemble learning, and using various parts of the training set for each predictor is known as bootstrap aggregation or bagging.

4.2.2 XGBoost.
XGBoost (Extreme Gradient Boosting) is a machine learning technique for creating ensemble models that is tuned for gradient boosting.

It works by constructing a sequence of decision trees, each of which is trained to correct the mistakes of the preceding trees. During training, XGBoost employs a loss function to compute the difference between predicted and actual values. By adding new trees to the ensemble, the algorithm attempts to minimize the loss function.

It also incorporates various optimization approaches, including as parallel computation, regularization, and early halting, to speed up training and enhance performance. These techniques enable models to be trained on big datasets with excellent accuracy and efficiency.

4.2.3 CatBoost.
Catboost is a boosted decision tree machine learning algorithm. It works in the same way as other gradient boosted algorithms such as XGBoost but provides support out of the box for categorical variables, has a higher level of accuracy without tuning parameters and also offers GPU support to speed up training.

CatBoost uses two types of regularization: L2 regularization to avoid overfitting and bagging (randomly sampling data subsets) to reduce variance. This makes it robust against overfitting and helps improve the generalization performance of the model.

CatBoost also incorporates several other features that can help to improve model performance, including the ability to handle missing values, advanced techniques for feature selection, and a novel method for optimizing the learning rate during the training process.

4.2.4 LightGBM.
LightGBM (Light gradient-boosting machine) is a gradient boosting framework developed by Microsoft that uses tree based learning algorithms.

It is being widely-used in many winning solutions of machine learning competitions. Comparison experiments on public datasets show that LightGBM can outperform existing boosting frameworks on both efficiency and accuracy, with significantly lower memory consumption. Distributed learning experiments show that LightGBM can achieve a linear speed-up by using multiple machines for training in specific settings.

4.2.5 Support Vector Machine (SVM).
Support Vector Machine (SVM) is a supervised learning machine learning algorithm that can be used mostly for classification task. In the SVM algorithm, each data item is plotted as a point in n-dimensional space (where n is the number of features you have), with the value of each feature being the value of a particular coordinate.

The classification is performed by finding the optimal hyper-plane that differentiates the two classes very well. Support Vectors are simply the coordinates of individual observation, and a hyper-plane is a form of SVM visualization. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/line).

5 Evaluation
The steps undertaken for the evaluation of the models are discussed below.

5.1 Dataset preparation
The publicly available Proteins Full (PF) network dataset was used for this work. It can be downloaded as a .csv file from [13]. It is a dataset of proteins that are classified as enzymes or non-enzymes. When visualised as a graph, it has 43.5K nodes and 162.1K edges. The maximum degree is 50, whereas the minimum degree is 2. Nodes represent the amino acids and two nodes are connected by an edge if they are less than 6 Angstroms apart. Since many recent literatures have used the dataset to report their results, we felt that it is good idea to use the same publicly available dataset as a means of benchmarking.

5.2 Experimental setup and Results
The dataset was divided into ten folds such that the class proportions were preserved in each of the folds. These folds were then utilized for cross-validation, with one section acting as the testing set and the other nine as the training set.

Table 1. Comparison of the classification accuracy obtained using different approaches

Accuracy	**Laplacian eigenvalues**	**Node2vec embeddings**
Random Forest	**0.75**	0.68
XGBoost	0.72	0.67
CatBoost	0.73	0.69
SVM	**0.75**	0.69
LightGBM	0.73	0.66

For each of the ten folds, the results were averaged across all testing sets. In order to obtain repeatable results, the random seed was set to 1. The embedding dimension was set to the average number of nodes in the dataset, and a unique set of classifier hyper-parameters was utilized.

We used scikit-learn's random forest classifier with "class-weights": balanced. Apart from the random forest classifier, we included XGBoost, CatBoost, LightGBM , and SVM with grid search to obtain the best hyper-parameters. The second embedding method we utilized was node2vec, which extracted the biological structures and represented them with lower dimensions.

The additional non-default hyper parameters were chosen using randomized cross validation across many datasets. Experiments were also carried out to ensure the robustness of our model in relation to some of its hyper-parameters. The model was simulated in a Jupyter notebook file hosted in Google Colab using K80 GPU and 12 GB of RAM.

The results were obtained extremely quickly. The embedding from the graph took about 50 seconds to generate, while training and testing the random forest classifier took about 40 seconds with the best hyper-parameters. As a result, taking Random Forest as an example, the total time required to execute all of the stated trials was roughly 1.5 minutes. However, obtaining the hyper-parameters that give the best performance, may require much computing power when running the grid search for each classifier. This demonstrates that employing normalized Laplacian eigenvalues as embeddings for a graph classification problem can offer extremely good accuracy while the whole process can be performed in a fraction of time to that of more complex algorithms.

A comparison of the classification accuracy obtained using different approaches are tabulated in Table 1. The rows headers of the table correspond to the algirithms used to generate the embeddings while the column headers correspond to the choice of the machine learning classifier.

We have tried to benchmark our model for the detection of enzymes from protein graphs against many recent state-of-the-arts deep learning models utilising Graph neural network methods on the same Protein Full (PF) dataset in Table 2. As

Table 2. Comparison of the performance of the classification accuracy with recent "deep learning" based graphical models on the Protein Full dataset

Algorithm	Accuracy	Reference
Graph Isomorphism Network (GIN)	0.84	Xu et al [12]
Deep Graph Convolutional Neural Network (DGCNN)	0.81	Zhang et al [11].
Graph Convolutional Network (GCN)	0.76	Kipf & Welling [9]
Graph Attention Network (GAT)	0.75	Velickovic et al [10].
Laplacian embeddings with Random Forest/SVM	**0.75**	**Proposed Method**

seen in the Table 2, the proposed method gives comparable performance in terms of accuracy when compared with a Graph Attention Network (GAN) or a Graph Convolutional Network (GCN), whereas more complex methods like Graph Isomorphism Network (GIN) or Deep Graph Convolutional Neural Network (DGCNN) gives a better classification accuracy. On the contrary, these deep networks take huge computation time to train their networks. We were unable to find a comparison of the time taken to train the models in the available literatures as a means of benchmarking, but available literatures show that these deep learning algorithms take considerable time for training owing to the repeated number of trials to arrive at the right set of hyperparameters. Also, these deeper networks extract the embeddings from the graph automatically, and as such it puts a question on their human interpretability.

Therefore, the authors feel that if the proposed method can provide near-equivalent performance in terms of accuracy, provides interpretibility, and leads to a significant gain in terms of time needed for training, it is worthwhile to consider it.

6 Conclusions

In summary, Laplacian-based methods do improve the accuracy of classification of graph methods. Laplacian methods outperformed the team's implementation using Node2vec consistently within a 4-13% difference across a range of classifiers (Random Forest, XGBoost, CatBoost, SVM & LightGBM).

One note for improvement the team would have liked to have undertaken is to implement more sophisticated neural network algorithms and compare performance in both accuracy and computational resources (in both computing power and time taken) between neural networks and Laplacian models.

7 References

[1] Ian Barnett, Nishant Malik, Marieke L Kuijjer, Peter J Mucha, and Jukka-Pekka Onnela. Feature-based classification of networks. arXiv preprint arXiv:1610.05868, 2016.

[2] Leonardo Gutierrez Gomez, Benjamin Chiem, and Jean-Charles Delvenne. Dynamics based features for graph classification. arXiv preprint arXiv:1705.10817, 2017.

[3] Thomas Bonald, Alexandre Hollocou, and Marc Lelarge. Weighted spectral embedding of graphs. arXiv:1809.11115, 2018.

[4] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. arXiv preprint arXiv:1802.08773, 2018.

[5] Nathan de Lara, Edouard Pineau. A Simple Algorithm for Graph Classification. arXiv: 1810.09155, 2018.

[6] Xiao-Dong Zhang. The laplacian eigenvalues of graphs: a survey. arXiv preprint. arXiv:1111.2897, 2011

[7] Miroslav Fiedler. Algebraic connectivity of graphs. Czechoslovak mathematical journal 23(2), 298–305, 1973

[8] Li, Y., Zhang, J., Hu, H., Li, Y., Zhu, J. Multi-Scale Graph Convolutional Networks for Protein Structure Classification. IEEE Transactions on Neural Networks and Learning Systems, 1-12. DOI: 10.1109/TNNLS.2021.3064141, 2021

[9] Kipf, T. N., Welling, M. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations, 2017

[10] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y. Graph attention networks. In International Conference on Learning Representations, 2018

[11] Zhang, J., Zhang, X., Wang, C., Qi, H. An end-to-end deep learning architecture for graph classification. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018

[12] Xu, K., Hu, W., Leskovec, J., Jegelka, S. How powerful are graph neural networks? In International Conference on Learning Representations, 2018

[13] Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI, https://networkrepository.com, 2015