DA 324 : Course Project Report

Nityam Pareek & Shantanu Chaudhari
Roll No 210150021 & 210150023

Objective

In this project, we are given data about a graph of 11952 nodes. Our objective is to cluster the nodes into 10 different classes. We are given data through the following three files :

1. *adjacency.csv* : This file contains the adjacency matrix (binary) of the graph.
2. *attributes.xlsx* : This file contains a 10-dimensional attribute vector for each node.
3. *seed.xlsx* : This file gives class information about 30 nodes, 3 per class.

1 Data Visualisation

We start with an exploratory data analysis of the attributes. On applying PCA and reducing the attributes from 103-dimensions to 2-dimensions, we get the plot in Figure 1.

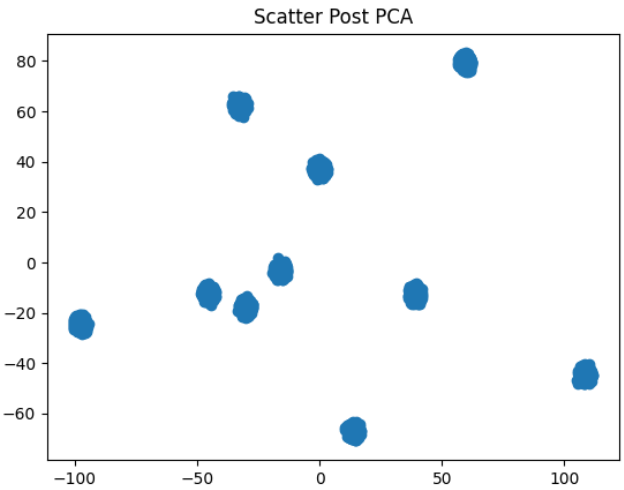

Figure 1: Plotting attributes after applying PCA.

We also apply non-linear dimensionality reduction using the Isomap algorithm. The results are seen in Figure 2.

To find out the more important features and in an attempt to denoise the data, we also calculated a matrix containing pairwise cosine similarities between attributes. We then applied spectral decomposition and observed the following distribution in top 10 eigenvalues as shown in Figure 3.

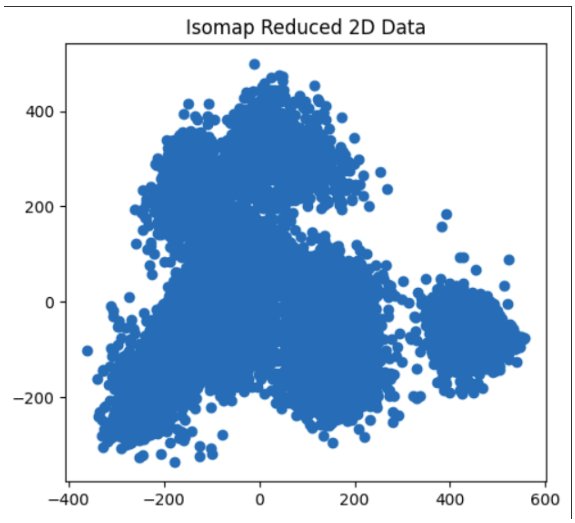

Figure 2: Visualisation of the projected data after applying ISOMAP

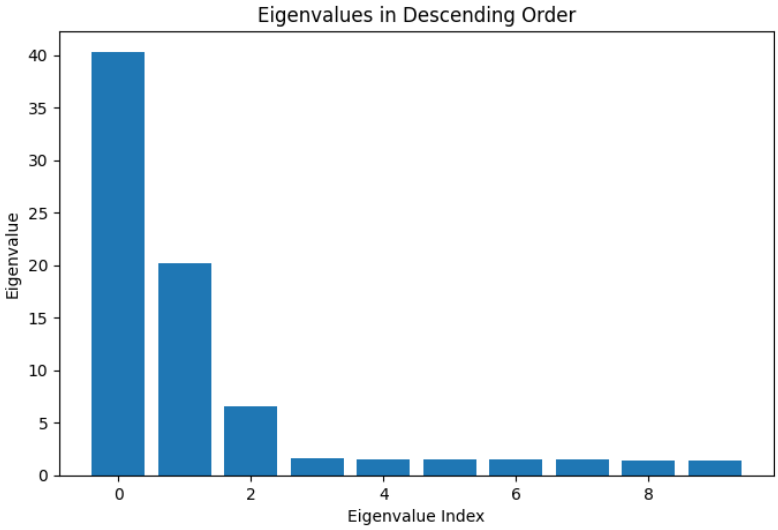

Figure 3: Eigenvalues of Cosine Similarity Matrix (Top 10).

We see that only the first 3 eigenvalues are significant, meaning we can reconstruct data using only the first 3 eigenvectors without losing much information. More visualisations are shown in the below sections as and when required.

2 Model Description

2.1 Assigning Labels to First 10952 Nodes

2.1.1 Extracting graph structure using Deepwalk

We needed a robust way to extract embeddings from graph structure; the two most popular approaches based on random walks for the same are Node2Vec and Deepwalk. We chose Deepwalk because of its computational efficiency on large graphs. At each node we initialized 5 random walks each with length 10. We used Word2Vec with window size four and skipgram to calculate the final embeddings. The embeddings were of 100 dimensions each.

Various other approaches were also used to get the embedding, for example, Spectral embedding, and pairwise distances. Those are listed in section 3

2.1.2 Combining Features

To cluster the nodes, we needed to combine the features obtained from Deepwalk and the *attributes.xlsx*. We approached it in various ways, such as adding the attributes to the graph and vice versa. In the end, the following approach worked the best in our case.

The features obtained after Deepwalk were normalized along with the attributes. They were concatenated to obtain the final set of features to cluster.

2.1.3 Clustering

Observing the feature space obtained after combining features, we used KMeans to cluster the nodes. To ensure we assign the correct label index to the nodes, we used the seeds available in seed.xlsx to initialize the clustering.

2.2 Assigning Labels to Last 1000 Nodes

Here, we explore using GraphSAGE, a specific GNN technique, for this purpose. GraphSAGE offers several advantages that make it well-suited for this task. Firstly, unlike some GNNs, GraphSAGE allows for inductive learning. This means it can generalize to unseen nodes, making predictions on entirely new graphs not present during training. Secondly, GraphSAGE is scalable, efficiently processing large networks by focusing on local neighborhoods of nodes. Finally, GraphSAGE's learnable aggregation function captures complex relationships between nodes within the network, leading to more informative embeddings. Overall, GraphSAGE's ability to handle unseen data, scalability, and flexibility in capturing network structure make it a compelling choice for learning node embeddings for tasks like node classification.

The approach leverages these strengths in a two-step process. Firstly, the GraphSAGE model is used to learn node representations by considering both the network structure and the nodes' existing feature vectors. This captures the context of each node within the network. Secondly, a logistic regression classifier is trained on these learned node embeddings and the previously labeled 10952 nodes to predict the class of unseen 1000 nodes.

2.3 Results

This approach gave us the best results out of all the others, giving us an accuracy of **0.24564**, as shown in the screenshot of Kaggle.

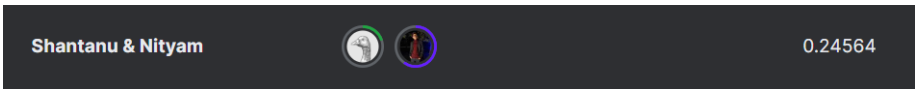

Figure 4: Best Submission on Kaggle.

3 Failed Attempts

3.1 K-Means Clustering

To make a baseline, we applied K-Means algorithm for clustering the data based on the attributes. We first applied Principal Component Analysis (PCA) to the 103-dimensional vectors to convert them into 2-dimensional vectors, and then visualised the clusters. We saw ten well defined clusters,

which encouraged us but upon closer inspection we see that these clusters do not indicate the various classes, because the points given in *seed.xlsx* do not all belong to the same cluster, as can be seen in the figure below. We initialised the algorithm with the first row of *seed.xslx* and let it

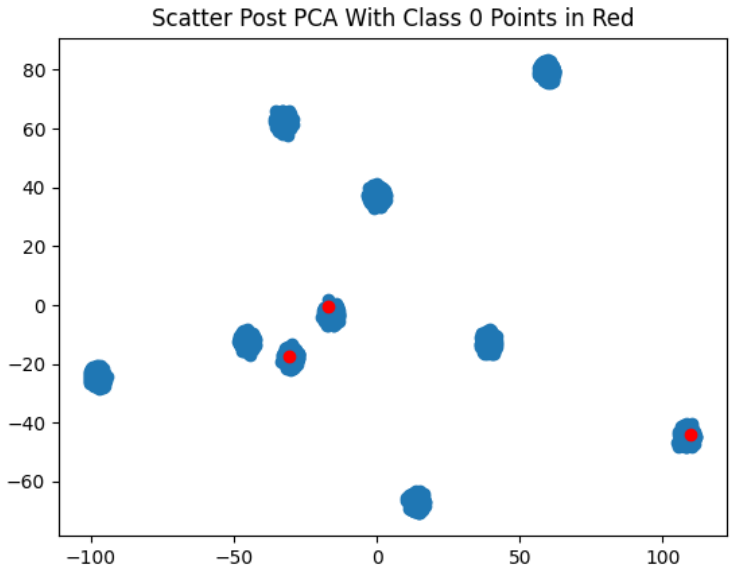

Figure 5: Plotting attributes after applying PCA, class 0 points plotted in red.

run for 1 iteration. We obtained a score of **0.12039**.

3.2 Spectral Clustering

Here, we tried to combine the attributes and the graph structure, first, we reduced the dimensions of the attributes by using PCA to 2 components. Then, we took the adjacency matrix and replaced all the 1's in the matrix with reciprocal of the distance between the nodes in the attributes. The intuition behind this approach was to make a weighted graph and then cluster the graph using spectral clustering. here we checked the distribution of assigned labels, and it was very skewed towards a single label, and there was no way to initialize the clusters using the sklearn implementation. Thus we had to abandon this approach

3.3 Spectral Embedding with non-Linear dimensionality reduction

We first calculated the spectral embedding of the adjacency matrix to capture the graph structure. We then combined the node attributes with each node's embedding to get a final attributes matrix with the attributes and the graph structure as features. To denoise the data, we applied various non-linear dimensionality reduction techniques to the data, which included ISOMAP, t-SNE, LLE, and Random Projection. We also tried projecting the data first and then combining the reduced features. Lastly, we applied K-Means to cluster using the initial seed. The results are in Table 1. Based on the results, we can conclude that non-linear dimensionality reduction didn't help much, as the scores of PCA and the rest of the methods are almost the same

3.4 Label Spreading

We tried using the label spreading algorithm provided by sklearn. Before using the algorithm, we had to preprocess our data to convert it into a usable format. We combined the node attribute data with the adjacency matrix by first calculating a matrix of pairwise euclidean distances between

Method	Result
PCA	0.13662
ISOMAP	0.12215
t-SNE	0.11629
Random Projections	0.10232
ISOMAP (Separately)	0.13662

Table 1: Results after applying non-linear dimensionality reduction

the nodes using their attributes. We then applied min-max normalization and then took reciprocal of the data. We then performed element wise multiplication with the adjacency matrix and obtained our final weighted adjacency matrix. These steps were done to indicate lower distances as having higher weights, representing a stronger edge. We then ran the algorithm with 2 different parameters - with rbf kernel, and knn kernel with 200 nearest neighbours being considered. The rbf kernel gave an accuracy of **0.08751** while the knn kernel gave an accuracy of **0.09320**. Both these algorithms perform worse than our baseline.