Assignment 1: Identify clusters using (Node2Vec Embedding, Spectral, and GCN) embeddings(Assignment)

Kummitha Jhanavi
CS21BTECH11032

Potta Vennela
CS21BTECH11046

Vinjam Aswitha
MA21BTECH11018

Pundi Bindusree
CS21BTECH11048

K Vivek Kumar
CS21BTECH11026

May 2, 2024

Abstract

Keywords: Synthetic Data Generation, Data analysis, Fraud Analytics

This paper presents a solution for Clustering data based on Node2Vec, Spectral and GCN embeddings on a dataset provided for Assignment 1 in the Fraud Analytics course.

---

1 Introduction to Clustering and Embeddings

Clustering is a fundamental unsupervised learning technique used to group similar data points together based on their inherent characteristics. Embeddings play a crucial role in clustering by transforming raw data into a lower-dimensional representation that captures essential features and relationships. Node2Vec, Spectral, and GCN embeddings are popular techniques for embedding graph-structured data, each offering unique advantages in capturing different aspects of the data's topology and structure.

2 Problem Statement

Given a dataset `Payments.csv`, we have to cluster similar types of customers who send and receive amounts of data. Clustering should be done using three ways of embedding data, Node2Vec, Spectral and GCN Embeddings.

3 Datasets

Dataset `Payments.csv` consists of three columns namely, Sender, Receiver and Amount. Sender sends the amount to the receiver via a transaction known to be payment in the `Payments.csv` row

4 Approach (Algorithm Used)

4.1 Node2Vec Embeddings

1. **Import Necessary Packages:** Import the necessary packages including matplotlib, scikit-learn, numpy, pandas, networkx, node2vec, torch, and others required for data manipulation, visualization, and clustering.

2. **Load the Dataset:** Load the dataset 'Payments.csv' using pandas, which contains information about payments including sender, receiver, and amount.

3. **Initialize Graph:** Initialize a directed graph using networkx.

4. **Populate Graph:** Populate the graph with nodes, edges, and corresponding weights using the data from the dataset. Each payment transaction is represented as an edge between the sender and receiver nodes with the transaction amount as the weight.

5. **Generate Node Embeddings:** Generate node embeddings using the Node2Vec algorithm. Configure parameters such as dimensions, walk length, number of walks, and workers for the Node2Vec model.

6. **Fit Node2Vec Model:** Fit the Node2Vec model to the graph to learn node embeddings. Retrieve node IDs and corresponding embeddings.

7. **Perform KMeans Clustering:** Perform KMeans clustering on the node embeddings to group nodes into clusters. Choose the number of clusters, and assign each node to a cluster.

8. **Visualize Node Embeddings:** Visualize the node embeddings in 2D space using a scatter plot, coloring the points based on their assigned cluster labels. Save the plot as 'node2vec.png'.

4.2 Spectral Embeddings

1. **Importing Necessary Packages:** This step involves importing all the required libraries and packages needed for the analysis, including `matplotlib` for plotting, `sklearn` for clustering algorithms, `numpy` and `pandas` for data manipulation, `networkx` for graph operations, `node2vec` for node embeddings, `torch` for neural network operations, and various other utilities.

2. **Loading the Datasets:** The dataset containing payment information is loaded from a CSV file named `Payments.csv` using `pandas`. This dataset will be used to construct a graph representing transactions between senders and receivers.

3. **Initializing a Networkx Graph:** A directed graph (`DiGraph`) is initialized using the `networkx` library. This graph will represent the payment transactions, where nodes represent users (senders and receivers), edges represent transactions, and edge weights represent the transaction amounts.

4. **Building the Graph:** The dataset is iterated row by row, and for each transaction, an edge is added to the graph connecting the sender to the receiver, with the transaction amount as the weight of the edge.

5. **Compute Spectral Embeddings:** Spectral embeddings are computed for the graph using the `SpectralEmbedding` class from `scikit-learn`. This technique transforms the graph into a low-dimensional vector space while preserving its structure.

6. **Dimensionality Reduction with PCA:** The dimensionality of the spectral node embeddings is reduced to two dimensions using Principal Component Analysis (PCA). This step is performed to visualize the embeddings in a 2D space while retaining most of the variance in the data.

7. **Applying KMeans Algorithm:** The KMeans clustering algorithm is applied to the reduced-dimensional spectral embeddings to group similar nodes together into clusters. The number of clusters is specified by the variable `num_clusters`.

8. **Plotting the Clusters:** Finally, the clusters obtained from KMeans clustering are visualized in a scatter plot. Each point represents a node in the reduced-dimensional space, colored according to its assigned cluster. This visualization helps in understanding the structure and distribution of clusters in the dataset.

4.3 GCN Embeddings

1. **Data Preparation and Graph Construction:**
   - Load the dataset `Payments.csv` into a pandas DataFrame.
   - Initialize a directed graph using networkx.
   - Build the graph with nodes, edges, and corresponding weights from the DataFrame.

2. **Node Features and Edge Indices:**
   - Create node features using sender and receiver IDs from the DataFrame.
   - Map sender and receiver IDs to node indices.
   - Create edge indices tensor for the graph.

3. **GCN Model Definition and Training:**
   - Define a Graph Convolutional Network (GCN) model class using PyTorch.
   - Initialize the GCN model with input, hidden, and output channels.
   - Define the optimizer and loss function for training.
   - Train the GCN model in a loop over a specified number of epochs.
   - Perform clustering on the output embeddings using KMeans clustering.
   - Convert cluster labels to PyTorch tensor format.

4. **Cluster Visualization:**
   - Plot the clusters using the output embeddings.
   - Visualize the clusters in a scatter plot with two embedding dimensions.
   - Save the plot as `gcn.png`.

5 Results

On plotting the embeddings as per their clusters in the following plots below we obtained the following values. We took the total number of clusters to be 5 and obtained these graphs.

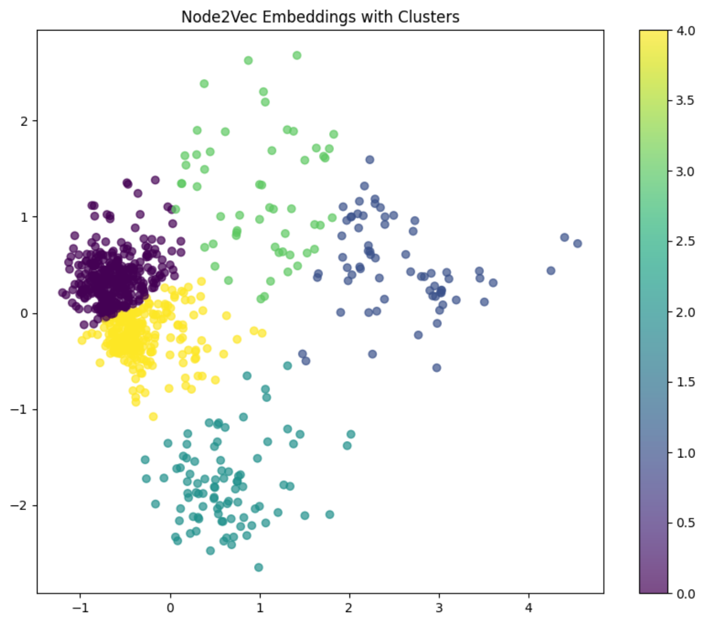

Figure 1: Node2Vec Embeddings

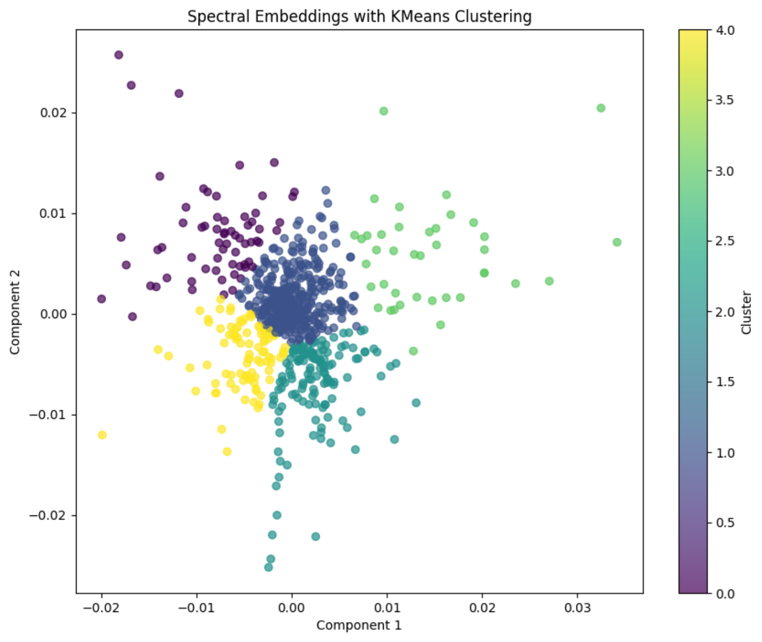

Figure 2: Spectral Embeddings

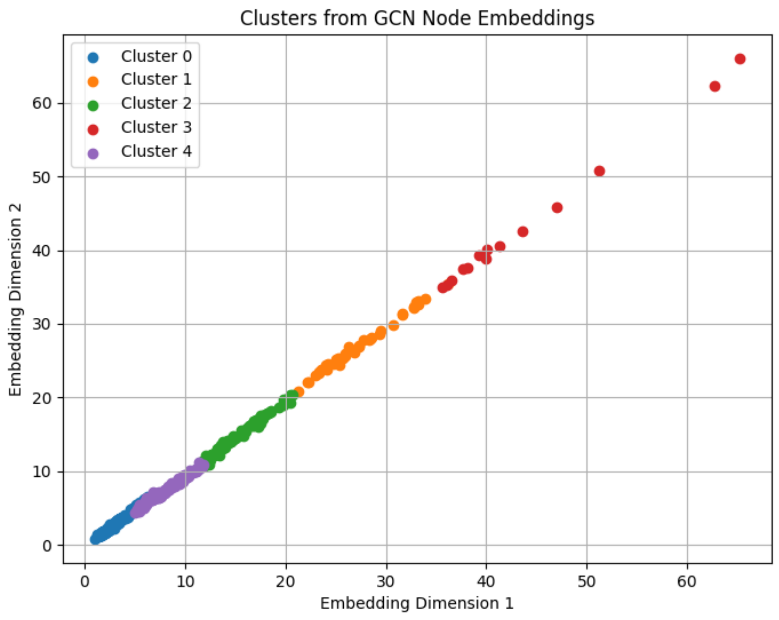

Figure 3: GCN Embeddings