CS6890: Fraud Analytics - Assignment 5 Finding Clusters using Node2Vec embedding

Akshay Santoshi - CS21BTECH11012
Nitya Bhamidipaty - CS21BTECH11041

April 2025

1 Introduction

In this assignment, we have a payments dataset, which we represented by graph, where nodes represent individuals, and directed edges represent financial transactions between them. We use Node2Vec for walk-based feature extraction and train a Skip-Gram with Negative Sampling model to learn embeddings. These embeddings are then clustered using KMeans, and visualized using PCA.

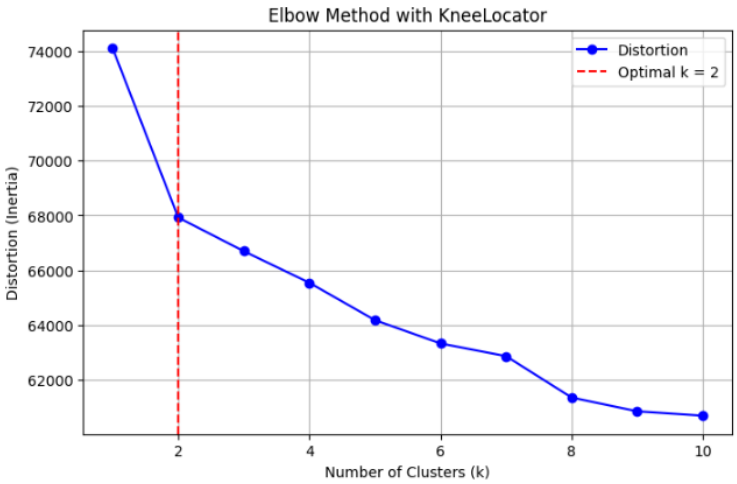
2 Approach

- We first read the Payments.xlsx file which contains transactions between senders and receivers. Next we combine all senders and receivers to form a unique list of nodes. Next, we create an adjacency list where each node maps to its neighbours and aggregated transaction amounts are used as edge weights. We have removed duplicate edges and combined them into a single edge. Next we mapped nodes to indices for tensor computation.
- Next we have defined a *generate_node2vec_walks* function. It simulates biased random walks on the graph. Node2Vec's parameters p and q control the walk bias. 'p' is the return parameter. If it is higher, it means that it is less likely to revisit. We have taken its value as 0.5. 'q' is the in-out parameter. It tells about the exploration vs local-focus. We have taken its value to be 2.0.
- Next, we have defined a *generate_skipgram_pairs* function. It converts each walk into center-context word pairs using a window of size context_size = 5. It mimics the way Word2Vec is trained on text.
- The class *SkipGramNegDataset* prepares training samples for the neural network. For each center-context pair, it gnerates num-negatives = 5 negative samples. These are random nodes and are not related to context.

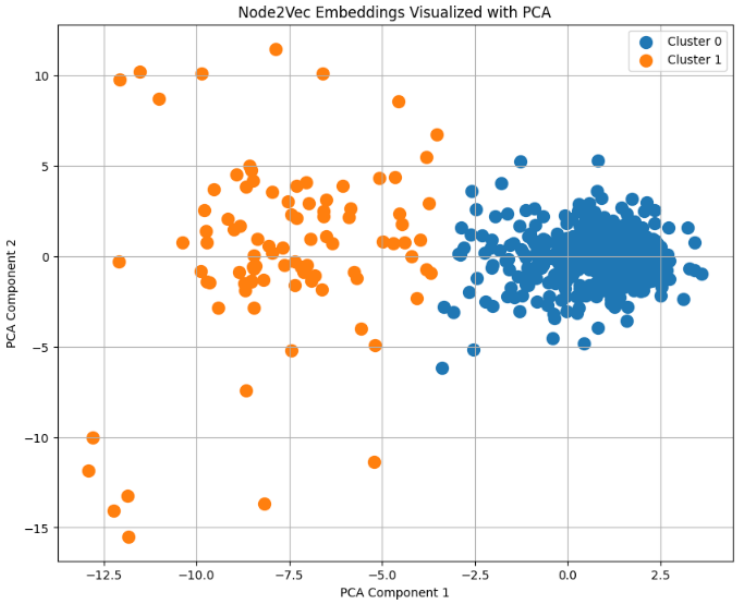
- Next we define the class *SkipGramNegSampling*. It consists of a neural network with two embedding matrices: one for input nodes and one for output/context nodes. It computes the positive loss, which is the dot product of center and context embeddings. And also a negative loss, which is the dot product with negative samples. This should be low.
- The function *train_skipgram_negative_sampling* instantiates the model, creates a dataloader. It trains over several epochs using Adam optimizer and at the end, it returns node embeddings as a Numpy array.
- Next we use KMeans clustering for multiple values of k to find the best number of clusters using elbow method (with KneeLocator). Then it fits KMeans on the final optimal_k.
- Next we apply PCA to reduce dimensions from 64 to 2. We visualize the clusters using scatter plot. We also print the cluster contents with original node IDs grouped by cluster.

3 Results

- Each node in the graph is represented as a 64-dimensional vector.
- Using elbow method, we find that the optimal value of k is 2.



- The PCA visualization gives a 2D scatter plot which shows the realtive positions of nodes. This helps us in interpreting clustering results visually.



- With this, we show that Node2Vec successfully gave these 2 clusters by preserving neighbourhood similarity in the learned embedding space.