आई आई टी हैदराबाद

IIT Hyderabad

CS6890
Fraud Analytics

Cluster identification using Node2Vec, Spectral and GCN embedding

---

V Harikrishnan
CS23MTECH11008

Suryansh Gautam
CS23MTECH11020

Patel Heetkumar D.
CS23MTECH11029

Anil kumar Sharma
CS23MTECH13001

KR Anuraj
CS23MTECH13002

1. Problem Statement

Clustering is a form of unsupervised learning where data points with similar features are club together to form clusters. To detect clusters in a graph algorithms like Node2Vec, Spectral clustering and neural networks like Graph Convolutional Network (GCN) are used.

Executing this algorithms on weighted multi-graphs is a challenging task. It requires various data pre-processing to make the graph suitable for algorithm implementation.

Various techniques like K-Nearest Neighbours, K Means and Principal Component Analysis are used to bring out the relationships between the nodes.

2. Description of the Dataset

Payment : It consists of payment information where each row is a transaction described by sender, receiver, amount. Sender and receiver column are nothing but sender's id and receiver's id respectively. This id's are used to construct a graph with each id corresponding to a node. Amount column can be used to represents the edge weight in the graph. Thus the edge weight can link different nodes and the edge weight can tell the degree of proximity of the nodes.

Dataset contains the multiple payments between a sender and a receiver and an accurate representation would involve constructing a multi edge directed weighted graph.

As a pre-processing step we apply log on the amount column so as to address the skewness of the payment which is appropriately scaled using the logarithm function.

3. Algorithm Used

A. GCN

i. Graph Construction

Inputs for Graph Construction are :

1. Payment Dataset : It consists of payment information where each row is a transaction described by sender, receiver, amount.

Oututs for Graph Construction are :

1. Directed Graph g : Directed graph with edges between nodes depicting sum of log of amount.
2. Edge list : Tensor of dimension (2, no of edges) with each node relabeled from (0, no of nodes).
3. Weight list : Ordered list of corresponding weights.

Algorithm Graph Construction

Input : payment dataset

Output : directed graph g, edge list e, weight list w

1. Read each row in the dataset and add an edge from sender to the receiver with the weight being the log of amount. If the multiple edges are present than sum their weights.
2. Re-label all the nodes from (0, no of nodes).
3. Extract all edges from the graph to from edge list e.
4. Add all corresponding weights to the weight list w.
5. return g, e, w

ii. GCN

Inputs for GCN are :

1. Directed graph g : It consists of payment information where each node is a sender or a receiver and each edge weight represents sum of log of amount in payment dataset.
2. initial_embedding_matrix[ ] [ ] : Its an identity matrix which represents one hot encoding for all the nodes in the graph.
3. Edge list e
4. Weight list w

Oututs for GCN are :

1. Model : Model with GCN layers obtained after training.

Algorithm GCN

Input : directed graph g, initial_embedding_matrix[ ] [ ], edge list e, weight list w

Output : model

1. Initialize GCN model with appropriate no of layers and embedding dimension.
2. Define loss_function which is the mean of pair wise euclidean distance between the embeddings.
3. Define Optimizer (Adam or SGD).
4. **for** epoch in range(no_of_epoch)
   (a) output = model (initial_embedding_matrix [ ] [ ], e, w)

(b) loss = loss_function(output,output)

(c) update parameters based on loss.

end for

5. return model

iii. Plot Cluster

Inputs for plot cluster are :

1. model : Trained model from previous step
2. initial_embedding_matrix[ ] [ ] : Its an identity matrix which represents one hot encoding for all the nodes in the graph.

Oututs for plot cluster are :

1. plot : scatter plot to visualize different clusters.

Algorithm Cluster Plot

Input : model, initial_embedding_matrix[ ] [ ]

Output : plot

1. output = model (initial_embedding_matrix [ ] [ ]) /* model outputs the final embedding of the nodes */
2. clusters, labels = KMeans(output) /* to perfrom clustering and obtain labels for each cluster */
3. embed = PCA(output) /* get 2d projection of embeddings */
4. plots the nodes by embedding and labels

B. Node2Vec

i. Graph Construction

Read each row in the dataset and add an edge from sender to the receiver with the weight being the log of amount. Get the multi edge directed graph as a result

ii. Node2Vec

Inputs for Node2Vec are :

1. Multi edge directed graph g : It consists of payment information where each node is a sender or a receiver and each edge weight represents log of amount in payment dataset.

2. Probability{} : It's a empty dictionary to be populated with probability for each outgoing edge for every node.

3. p, q : hyperparameters to determine the type of random walk.

Oututs for Node2Vec are :

1. walks = Traverses on the graph based on the probabilities given by the weights of the edges.

Algorithm Node2Vec

Input : Multi edge directed graph g, probability {}, p ,q

Output : walks

1. computing probabilities for transition from one node to another.
```
for src in g
    for crt in g.neighbours(src)
        for target = g.neighbours(crt)
            if src = target
                prob = g[crt][target].weights * (1/p)
            else if target in g.neighbours(src)
                prob = g[crt][target].weights
            else
                prob = g[crt][target].weights * (1/q)
            end if
            append prob to probability {}
        end for
    end for
end for
```

2. generating random walks W
```
for node in g
walk = [node]
    for i in range(max_walks)
        for k in range(walk_len)
            sample a node based on the last node in the walk
            add node to walk list
        end for
        add walk to the W
    end for
end for
```

3. return walks

iii. Getting embeddings and plotting

Inputs for Getting embeddings and plotting are :

1. walks W

Oututs for Getting embeddings and plotting are :

1. embeddings = embedding for Node2Vec.
2. plot : scatter plot to visualize different clusters.

Algorithm Cluster Plot

Input : walks W

Output : embeddings, plot

1. from the random walks pass to Word2Vec algorithm and obtain the embeddings.
2. clusters, labels = KMeans(embeddings) /* to perfrom clustering and obtain labels for each cluster */
3. embed = PCA(embeddings) /* get 2d projection of embeddings */
4. plots the nodes by embedding and labels

C. Spectral Clustering

i. Graph Construction

Inputs for Graph Construction are :

1. Payment Dataset : It consists of payment information where each row is a transaction described by sender, receiver, amount.
2. k : nubmber of nearest neighbours.

Oututs for Graph Construction are :

1. Directed Graph g : Directed graph with edges only in between nodes that are close to each other (max amount transferred).
2. Adjacency matrix adj[][] : matrix of dimension n x n (no of nodes) with value 1 if edge is their for i to j else 0.
3. Degree matrix D [][] : diagonal matrix of n x n with the entries corresponding to the number of outgoing edges.

Algorithm Graph Construction

Input : payment dataset , k

Output : directed graph g , adjacency matrix adj[][], D[][]

1. Read each row in the dataset and add an edge from sender to the receiver with the weight being the log of amount. If the multiple edges are present than sum their weights.
2. Get adjacency matrix adj[][] from the graph.
3. **for** i in range(no of nodes)
   (a) Get the top k non-zero entries in the row corresponding to the node i.
   (b) Make each of the top non-zero entry 1 and rest to zero.

   end for
4. Sum each row of the adj[][] to get D[][].
5. return D[][], g, adj[][]

ii. Spectral Clustering and plotting

Inputs for Spectral Clustering and plotting are :

1. Adjacency matrix adj[][]
2. Degree matrix D[][]
3. Dimension dim : Number of eigenvectors to be considered.

Oututs for Spectral Clustering and plotting are :

1. plot : scatter plot to visualize different clusters.

Algorithm Cluster Plot

Input : adj[][], D[][], dim

Output : plot

1. Do D[][] - adj[][].
2. Calculate the inverse square root of Degree matrix D^-(1/2).
3. Find normalize laplacian L_norm = I_n - D^-(1/2) * adj[][] * D^-(1/2)
4. Find eigenvectors and eigenvalues of L_norm.
5. Take the dim number of lowest eigenvalues and store the corresponding eigenvectors in embeddings.
6. clusters, labels = KMeans(embeddings) /* to perfrom clustering and obtain labels for each cluster */
7. embed = PCA(embeddings) /* get 2d projection of embeddings */
8. plots the nodes by embedding and labels

4. Results

A. GCN

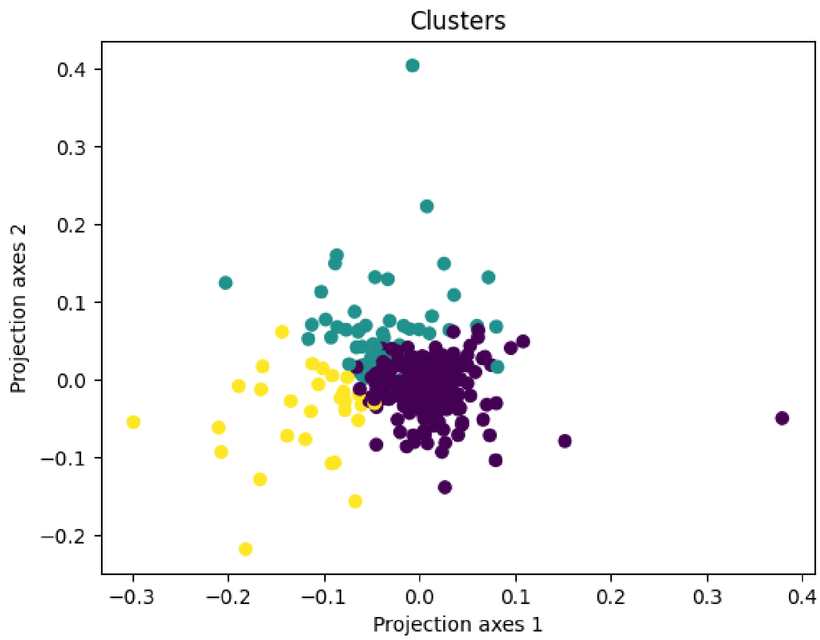

Figure 1: GCN clusters

B. Node2Vec

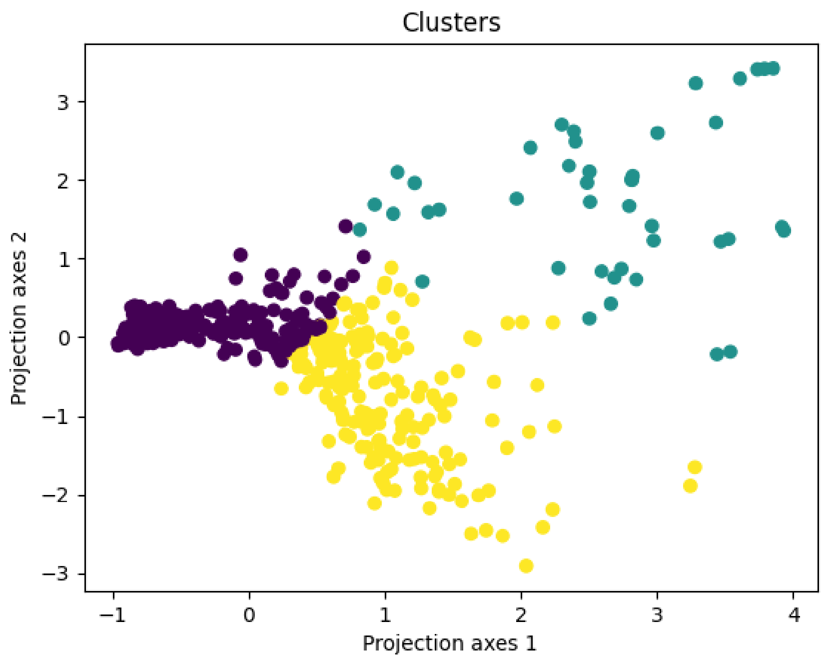

Figure 2: Node2Vec clusters with p = 2 and q = 5 (structural equivalence)

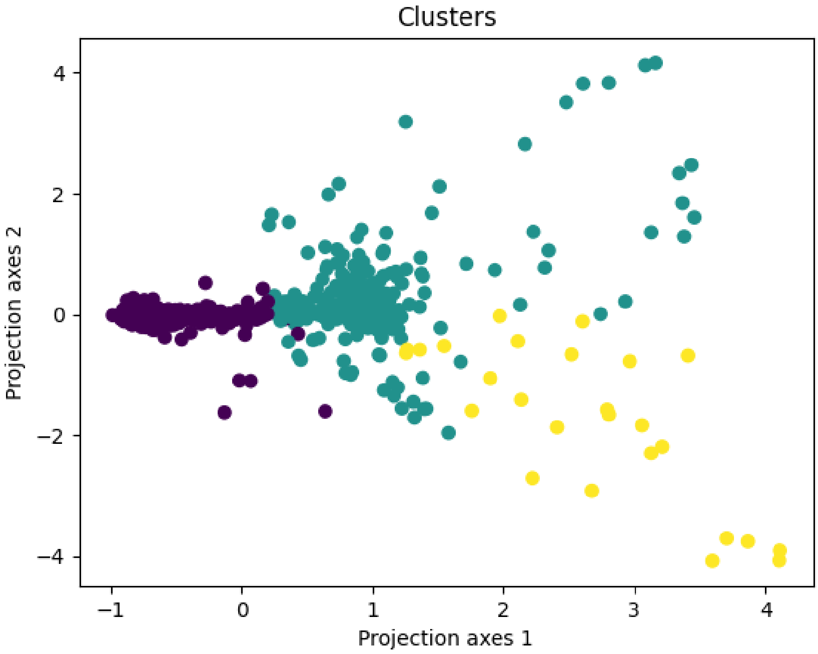

Figure 3: Node2Vec clusters with p = 5 and q = 2 (homophily)

B. Node2Vec

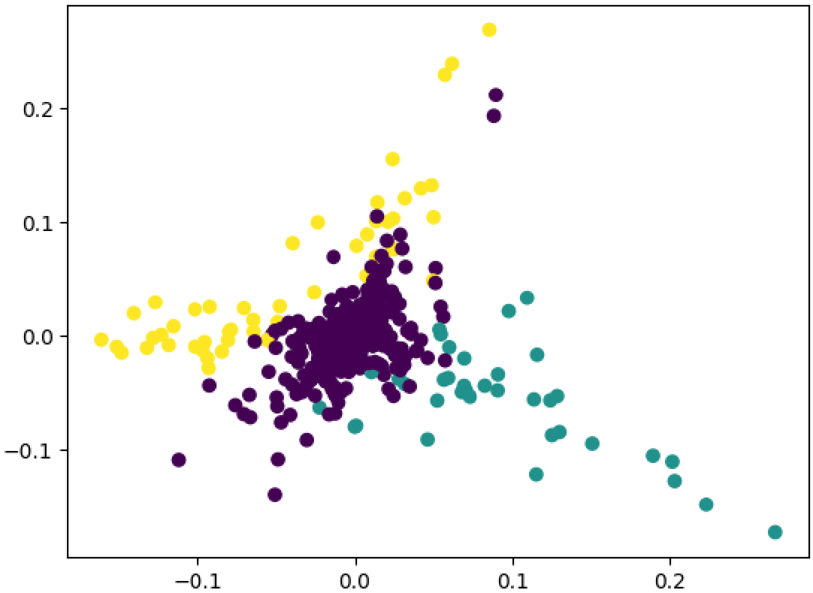

Figure 4: spectral clustering