Attributed Graph Clustering:

A Deep Attentional Embedding Approach

Published in IJCAI 2019

Problem Definition

- Data is represented in graph format rather than flat-table or vector format
- Many real-world tasks rely on graph-data mining skills
- The comlpexity of graph has imposed challenges on these tasks, including graph clustering

Problem Definition

Attributed Graph

$G = (V, E, X)$

- **Nodes,** $V = \{v_i\}_{i=1,\cdots,n}$
- **Edges,** $E = \{e_{ij}\}$
- **Adjacency Matrix,** A (IF $(v_i, v_j) \in E$, $A_{i,j} = 1$, ELSE $A_{i,j} = 0$)
- **Attribute Values,** $X = \{x_1; \cdots ; x_n\}$
  $x_i \in R^m$ is real-value attribute vector associated with v_i

Problem Definition

Graph Clutering

Aims to partition the nodes in G into k disjoint groups $\{G_1, G_2, \cdots, G_k\}$.
The nodes within the same cluster:

- close to each other in terms of graph structure
- more likely to have similar attribute values

The key problem is how to capture the structural relationship and exploit the node content information

Prior works

Use deep learning techniques to learn compact representation to exploit the rich information of both the content and structure data.
Apply simple clustering algorithms based on the learned graph embedding.

Autoencoder is a mainstream soluton.

But these embedding-based methods are **two-step**.
The learned embedding may not be the best fit for the graph clustering task.

Proposed Method

Deep Attentional Embedded Graph Clustering, DAEGC

Graph Attentional Autoencoder

- Input: Attribute values & Graph structure
- Output: Embedding
- By Minimizing the reconstruction loss

Self-training Clustering

- Input: Embedding
- Output: Soft Assignment
- By minimizing the clustering loss

Joint Embedding and Clustering Optimization

Graph Attentional Autoencoder

A Unified Framework:
Graph Structure A + Node Content X $\rightarrow$ **Graph Encoder** $\rightarrow$ Representations

Variant of the graph attention network [Velickovic et al., 2017]

The idea is to learn hidden representations of each node by attending over its neighbors, to combine the attribute values with the graph structure in the latent representation.

Graph Attentional Autoencoder

$$z_i^{l+1} = \sigma(\sum_{j \in N_i} \alpha_{ij} W z_j^l)$$

- z_i^{l+1}, the output representation of node i
- N_i, the neighbors of i
- α_{ij}, attention coefficient

  indicates the importance of neighbor node j to node i
- W, weight matrix
- σ, nonlinearity function

How to calculate the attention coefficient α_{ij}?

- **Attribute Values**

  c_{ij}, single-layer feedforward neural network on the concatennation of x_i and x_j with weight vector $\vec{a} \in R^{2m'}$

$$c_{ij} = \vec{a}^T [Wx_i || Wx_j]$$

- **Topological Distance**

  Proximity Matrix

$$M = (B + B^2 + \cdots + B^t)/t$$

  B, transition matrix, IF $e_{ij} \in E$ $B_{ij} = 1/d_i$ ELSE $B_{ij} = 0$
  d_i, degree of node i
  M_{ij}, the topological relevance of node j to node i up to t orders

How to calculate the attention coefficient α_{ij}?

$$\alpha_{ij} = \text{softmax}_j(c_{ij}) = \frac{\exp(c_{ij})}{\sum_{r \in N_i} \exp(c_{ir})}$$

$$\downarrow$$

$$\alpha_{ij} = \frac{\exp(\delta M_{ij}(\vec{a}^T [W x_i || W x_j]))}{\sum_{r \in N_i} \exp(\delta M_{ir}(\vec{a}^T [W x_i || W x_r]))}$$

- δ, LeakyReLU

Graph Attentional Autoencoder

Stack Two Graph Attention Layers

$$
\begin{gathered}
x_i = z_i^0 \\
\downarrow \\
z_i^{(1)} = \sigma\Big(\sum_{j \in N_i} \alpha_{ij} W^{(0)} x_j\Big) \\
\downarrow \\
z_i^{(2)} = \sigma\Big(\sum_{j \in N_i} \alpha_{ij} W^{(1)} z_j^{(1)}\Big) \\
\downarrow \\
z_i = z_i^{(2)}
\end{gathered}
$$

Reconstruction

Inner Prodcut Decoder

Adopt a simple inner product decoder to predict the links between nodes:

$$\hat{A}_{ij} = \text{sigmoid}(z_i^T z_j)$$

Reconstruction Loss

$$L_r = \sum_{i=1}^{n} loss(A_{ij}, \hat{A}_{ij})$$

Self-optimizing Embedding

Soft Assignment (Probability of assigning sample i to cluster u):

Use Student's t-distribution to measure the similarity between embedding and cluster center embedding:

$$q_{iu} = \frac{(1 + ||z_i - \mu_u||^2)^{-1}}{\sum_k (1 + ||z_i - \mu_k||^2)^{-1}}$$

- handle different scaled clusters
- computationally convenient

Self-optimizing Embedding

Refine the clusters by learning from high confidence assignments with the help of auxiliary target distribution.

$$L_c = KL(P||Q) = \sum_i \sum_u p_{iu} log \frac{p_{iu}}{q_{iu}}$$

The choice of target distributuion P is crucial for clustering algorithm's performance.

Self-optimizing Embedding

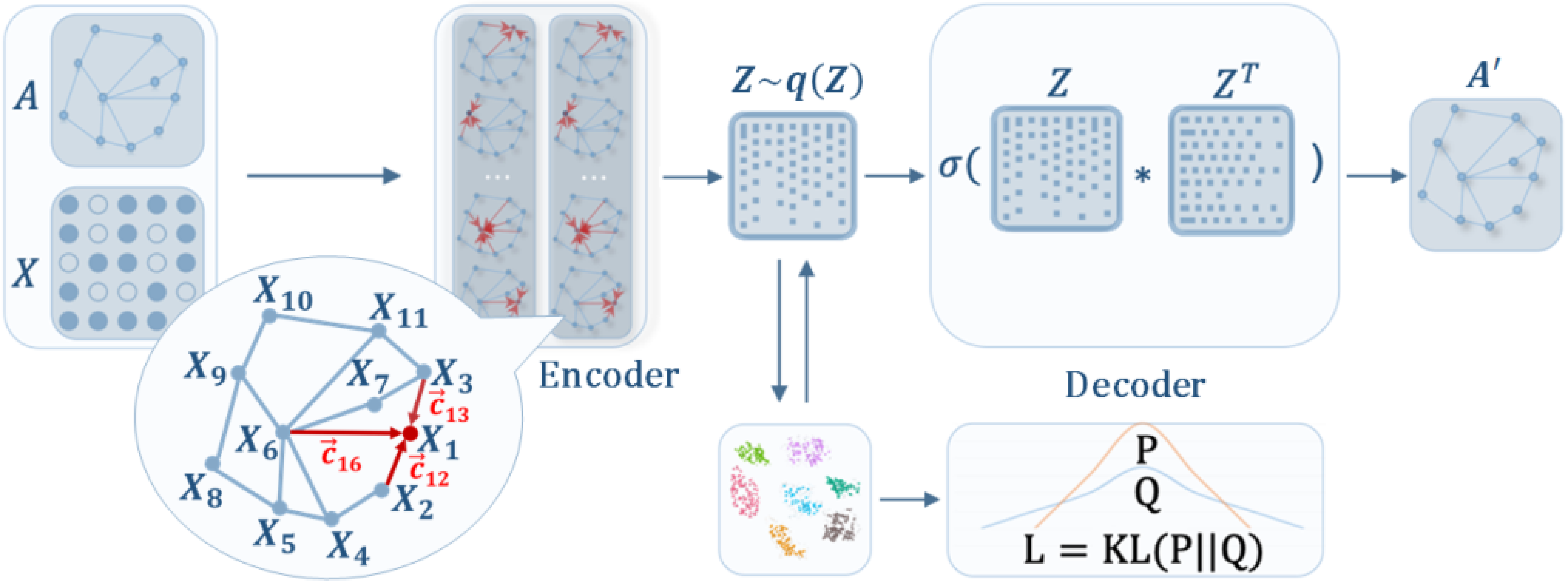
Target Distribution

- Strengthen predictions
- put more emphasis on data points with high confidence
- normalize loss contribution of center to prevent large clusters from distorting the latent space

$$p_{iu} = \frac{q_{iu}^2 / \sum_i q_{iu}}{\sum_k (q_i k^2 / \sum_i q_{ik})}$$

Joint Embedding and Clustering Optimization

$$L = L_r + \gamma L_c$$



Algorithm DAEGC

- Update the autoencoder by minimizing L_r to obtain embedding Z
- Compute the initial cluster centers μ by k-means based on Z
- for $l = 0$ to $Iter - 1$ do:
    - Calculate soft assignment distribution Q with Z and μ according to t-distribution
      if l then
        - Calculate target distribution P with Q
    - Calculate clustering loss L_c
    - Update the whole framework by minimizing L
- Get the clustering results

END