Paper's abstract

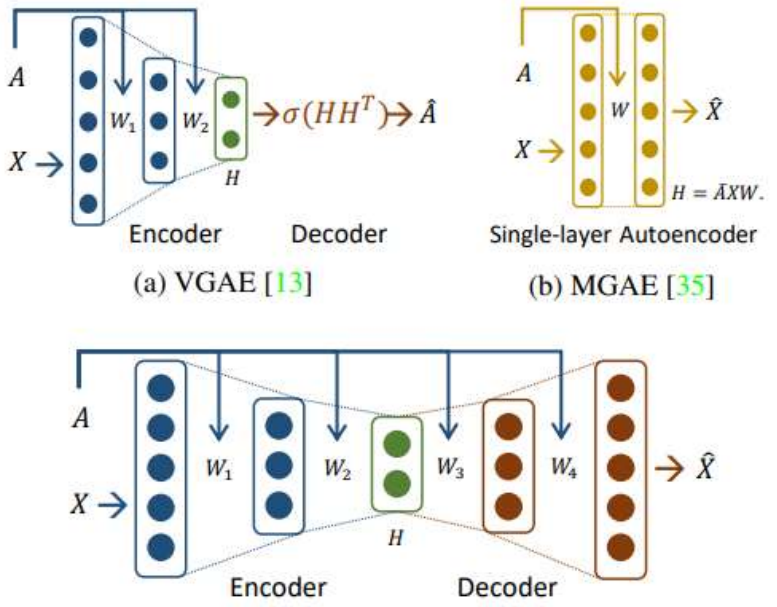
We propose a symmetric graph convolutional autoencoder which produces a low-dimensional latent representation from a graph. In contrast to the existing graph autoencoders with asymmetric decoder parts, the proposed autoencoder has a newly designed decoder which builds a completely symmetric autoencoder form. For the reconstruction of node features, the decoder is designed based on Laplacian sharpening as the counterpart of Laplacian smoothing of the encoder, which allows utilizing the graph structure in the whole processes of the proposed autoencoder architecture. In order to prevent the numerical instability of the network caused by the Laplacian sharpening introduction, we further propose a new numerically stable form of the Laplacian sharpening by incorporating the signed graphs. In addition, a new cost function which finds a latent representation and a latent affinity matrix simultaneously is devised to boost the performance of image clustering tasks. The experimental results on clustering, link prediction and visualization tasks strongly support that the proposed model is stable and outperforms various state-of-the art algorithms.

paper link

Introduction

Problem Statement

- In recent years, an emerging field called geometric deep learning, generalizes deep neural network models to non-Euclidean domains such as meshes, manifolds, and especially **graphs**.
- Among them, **finding deep latent representations of geometrical structures of graphs** using an auto-encoder framework is getting growing attention.
- However, conventional graph auto-encoder (GAE) methods have limitations.
  - Some of them were designed to reconstruct **the affinity matrix A instead of node feature**.
  - Some of them's **decoder part cannot be learnable**.
  - Some of them are **too shallow** to map high-dimensional node features.



(a) VGAE [13] (b) MGAE [35]

(c) Proposed autoencoder

Figure 1: Architectures of existing graph convolutional autoencoders and proposed one. A, X, H and W denote the affinity matrix (structure of graph), node attributes, latent representations and the learnable weight of network respectively.

Proposed Solution and Contribution Points

- In order to make symmetric structure like convolutional auto-encoder, the authors propose Laplacian sharpening layer which is the counterpart of Laplacian smoothing (Graph convolutional layer).
- In order to improve image clustering performance (**node clustering without ground truth affinity matrix**), the authors propose subspace clustering cost function.

Methods

Laplacian sharpening layer

- The authors were inspired by Laplacian **smoothing**, and design Laplacian **sharpening** layer.
- Laplacian sharpening (b) is almost same with smoothing (a), but only difference is **the sign of affinity matrix**.

$$H^{(m+1)} = \xi(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(m)}\Theta^{(m)}) \quad \text{(a)}$$

$$H^{(m+1)} = \xi((2I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}})H^{(m)}\Theta^{(m)}) \quad \text{(b)}$$

- In order to make Laplacian sharpening stable, they apply **renormalization trick** like smoothing.

$$\hat{A} = 2I_n - A \text{ and } \hat{D} = 2I_n + D$$

$$H^{(m+1)} = \xi(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(m)}\Theta^{(m)})$$

- In order to improve clustering performance, the authors propose the clustering cost function.

$$\min_{\bar{X},H} \underbrace{\frac{1}{2}\|X - \bar{X}\|_F^2}_{\text{Recon. cost}} + \underbrace{\frac{\mu\lambda}{2}tr((\mu I_k + \lambda HH^T)^{-1}HH^T)}_{\text{Clustering cost}}$$

Experimental Results

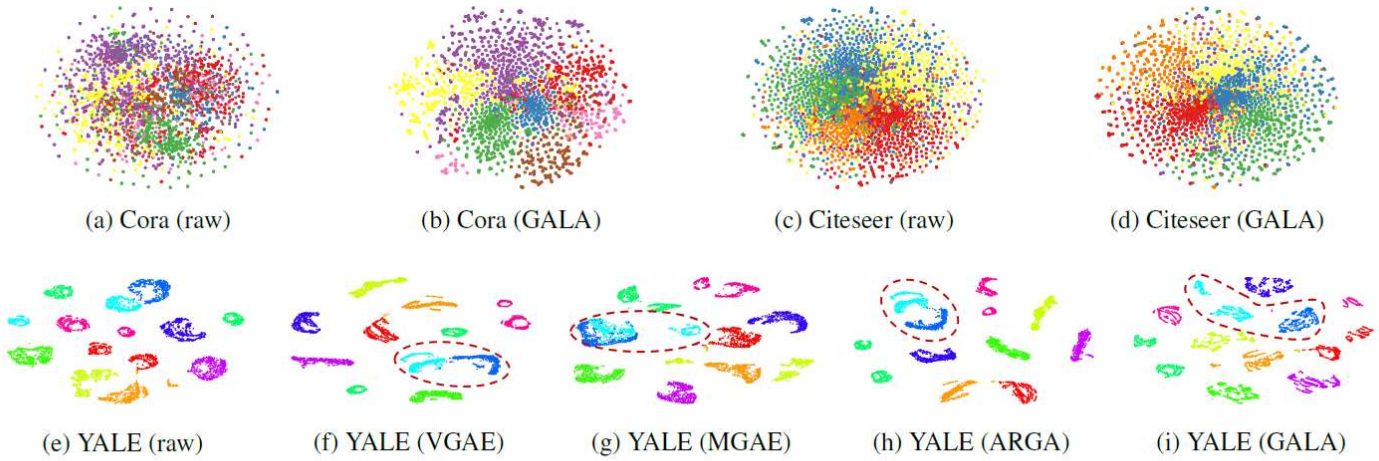
- Note that four network datasets (Cora, Citeseer, Wiki, and Pubmed) provide affinity matrix but image datasets (COIL20, YALE, and MNIST) don't provide affinity matrix.
- In all the experiments, the proposed methods outperform than others.

Table 2: Experimental results of node clustering

	Cora			Citeseer			Wiki		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Kmeans[21]	0.4922	0.3210	0.2296	0.5401	0.3054	0.2786	0.4172	0.4402	0.1507
Spectral[26]	0.3672	0.1267	0.0311	0.2389	0.0557	0.0100	0.2204	0.1817	0.0146
Big-Clam[38]	0.2718	0.0073	0.0011	0.2500	0.0357	0.0071	0.1563	0.0900	0.0070
DeepWalk[28]	0.4840	0.3270	0.2427	0.3365	0.0878	0.0922	0.3846	0.3238	0.1703
GraEnc[33]	0.3249	0.1093	0.0055	0.2252	0.0330	0.0100	0.2067	0.1207	0.0049
DNGR[3]	0.4191	0.3184	0.1422	0.3259	0.1802	0.0429	0.3758	0.3585	0.1797
Circles[17]	0.6067	0.4042	0.3620	0.5716	0.3007	0.2930	0.4241	0.4180	0.2420
RTM[4]	0.4396	0.2301	0.1691	0.4509	0.2393	0.2026	0.4364	0.4495	0.1384
RMSC[36]	0.4066	0.2551	0.0895	0.2950	0.1387	0.0488	0.3976	0.4150	0.1116
TADW[37]	0.5603	0.4411	0.3320	0.4548	0.2914	0.2281	0.3096	0.2713	0.0454
VGAE[13]	0.5020	0.3292	0.2547	0.4670	0.2605	0.2056	0.4509	0.4676	0.2634
MGAE[35]	0.6844	0.5111	0.4447	0.6607	0.4122	0.4137	0.5146	0.4852	0.3490
ARGA[27]	0.6400	0.4490	0.3520	0.5730	0.3500	0.3410	0.3805	0.3445	0.1122
ARVGA[27]	0.6380	0.4500	0.3740	0.5440	0.2610	0.2450	0.3867	0.3388	0.1069
GALA	**0.7459**	**0.5767**	**0.5315**	**0.6932**	**0.4411**	**0.4460**	**0.5447**	**0.5036**	**0.3888**

Table 4: Experimental results of image clustering

	COIL20			YALE			MNIST		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Kmeans[21]	0.6118	0.7541	0.5545	0.7450	0.8715	0.7394	0.5628	0.5450	0.4213
Spectral[26]	0.6806	0.8324	0.6190	0.5793	0.7202	0.4600	0.6496	0.7204	0.5836
GAE[13]	0.6632	0.7420	0.5514	0.8520	0.8851	0.8122	0.7043	0.6535	0.5534
VGAE[13]	0.6847	0.7465	0.5627	0.9157	0.9358	0.8873	0.7163	0.7149	0.6154
MGAE[35]	0.6507	0.7889	0.6004	0.8203	0.8550	0.7636	0.5807	0.5820	0.4362
ARGA[27]	0.7271	0.7895	0.6183	0.9309	0.9394	0.8961	0.6672	0.6759	0.5552
ARVGA[27]	0.7222	0.7917	0.6240	0.8727	0.8803	0.7944	0.6328	0.6123	0.4909
GALA	0.8000	0.8771	0.7550	0.8530	0.9486	0.8647	0.7384	0.7506	0.6469
GALA+SCC	**0.8229**	**0.8851**	**0.7579**	**0.9933**	**0.9860**	**0.9854**	**0.7426**	**0.7565**	**0.6675**



(a) Cora (raw) (b) Cora (GALA) (c) Citeseer (raw) (d) Citeseer (GALA)

(e) YALE (raw) (f) YALE (VGAE) (g) YALE (MGAE) (h) YALE (ARGA) (i) YALE (GALA)

Figure 3: The two-dimensional visualizations of raw features of each node and the latent representations of compared methods and GALA for Cora, Citeseer and YALE are presented. The same color indicates the same cluster.

Conclusion and Personal Opinions.

- The authors propose a novel graph convolutional auto-encoder framework.