A Degeneracy Framework for Scalable Graph Autoencoders

Guillaume Salha[1,2], Romain Hennequin[1], Viet-Anh Tran[1], Michalis Vazirgiannis[2]

[1] Deezer Research & Development, Paris, France
[2] LIX, École Polytechnique, Palaiseau, France

I - Summary

Context:

- **Graphs** have become ubiquitous in the Machine Learning community.
- Learning **node embeddings**, i.e. low dimensional vector representations of nodes in which *similar* nodes are *close*, appears as an effective way to extract meaningful information from graph structures.
- Graph **autoencoders (AE)** and **variational autoencoders (VAE)** recently emerged as powerful node embedding methods.
- **However, existing graph AE and VAE suffer from scalability issues, and all experiments are limited to relatively small graphs (<20K nodes).**

Contributions:

- We introduce a **general framework to scale graph AE and VAE models**, leveraging **graph degeneracy** concepts (k-core decomposition).
- We apply this framework to five real-world datasets and two learning tasks. These are the **first applications of graph AE/VAE to large graphs with up to millions of nodes/edges**.
- We empirically show that our approach significantly **improves scalability** while **preserving performance**. We also achieve competitive results w.r.t. alternative node embeddings methods such as node2vec and DeepWalk.

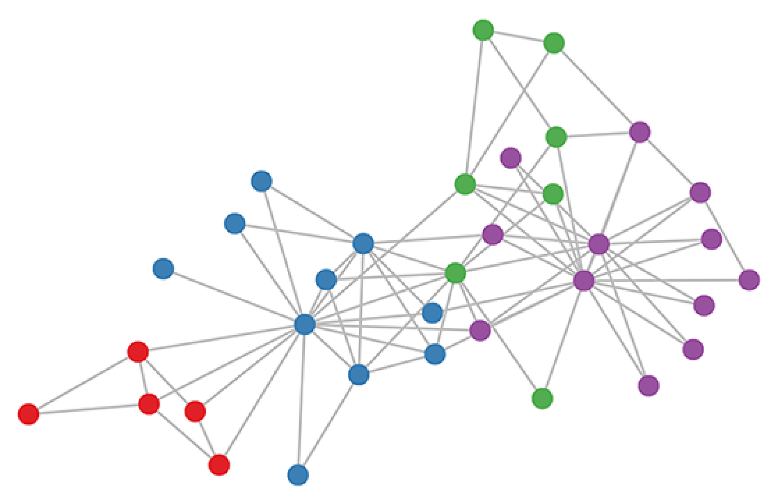
This work [3] will be presented at the **IJCAI 2019** international conference.

II - Representation Learning on Graphs

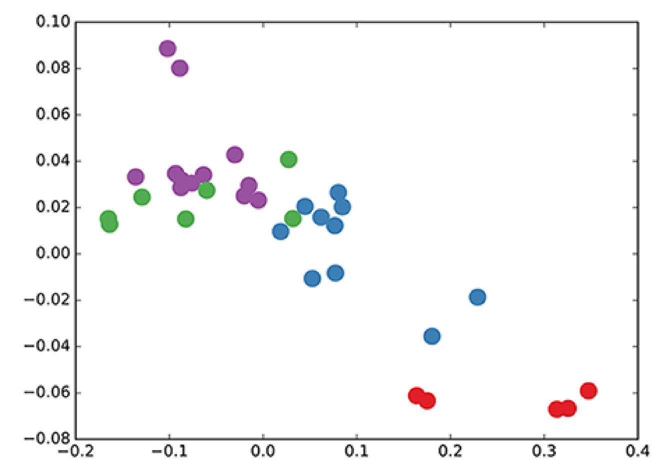
We consider an **undirected graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$ nodes, $|\mathcal{E}| = m$ edges, without self-loops. A is the $n \times n$ adjacency matrix of $\mathcal{G}$, weighted or not.

Node Embedding paradigm: instead of directly working at the graph level, map nodes into a low-dimensional vector space $\mathcal{Z}$.

- Node $i \in \mathcal{V} \longrightarrow$ **latent vector** z_i of size $d \ll n$.
- convenient for challenging tasks, e.g. **missing link prediction** and **node clustering** [1].



→



(Images from https://tkipf.github.io/).

III - Graph Autoencoders (AE) and Variational Autoencoders (VAE)

Graph AE and VAE recently emerged as powerful node embedding methods [2]
→ *successful applications to link prediction, node clustering, recommendation, graph generation...*

Graph AE: unsupervised learning of a node embedding (**encoding**) from which reconstructing the graph (**decoding**) is possible.

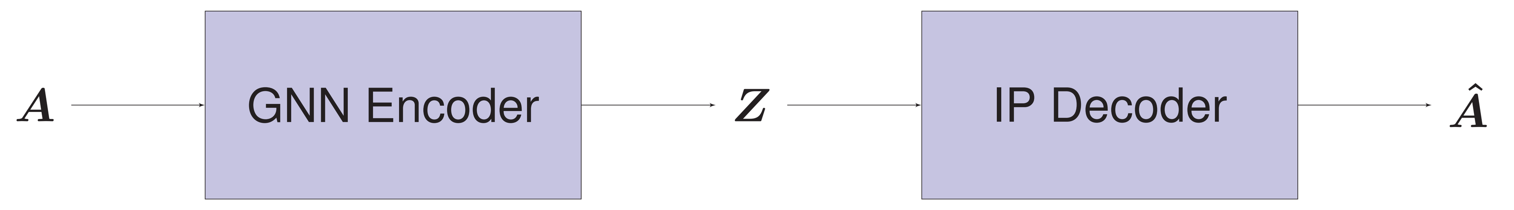
- Encoder step: learn $n \times d$ embedding matrix Z, stacking up latent vectors z_i.
- Usually the output of a **Graph Neural Network (GNN)**:

$$Z = \textbf{GNN}(A)$$

- Decoder step : reconstruct A using **inner products** between latent variables with sigmoid activation:

$$\hat{A} = \sigma(ZZ^T)$$

- Training: minimize **reconstruction loss** $\|A - \hat{A}\|_F$, by **stochastic gradient descent**.



Graph VAE: assume a **probabilistic model** on the graph structure involving some latent variables z_i of length d for each node, interpreted as latent representations.

- **Inference model** (encoder):

$$q(Z|A) = \prod_{i=1}^{n} q(z_i|A) \text{ where } q(z_i|A) = \mathcal{N}(z_i|\mu_i, \text{diag}(\sigma_i^2)).$$

- Gaussian parameters are learned using two GNNs: $\mu = \textbf{GNN}_\mu(A)$ and $\log \sigma = \textbf{GNN}_\sigma(A)$.
- **Generative model** (decoder):

$$p(A|Z) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(A_{ij}|z_i, z_j), \text{ where } p(A_{ij} = 1|z_i, z_j) = \sigma(z_i^T z_j).$$

- Training: maximize a tractable **lower bound of the model's likelihood (ELBO)**:

$$\mathcal{L} = \mathbb{E}_{q(Z|A)}\Big[\log p(A|Z)\Big] - \mathcal{D}_{KL}(q(Z|A)\|p(Z))$$

by gradient descent, with a Gaussian prior $p(Z) = \prod_i p(z_i) = \prod_i \mathcal{N}(z_i|0, I)$. $\mathcal{D}_{KL}(\cdot,\cdot)$ is the Kullback-Leibler divergence.

IV - Scaling-Up Graph AE and VAE with Degeneracy

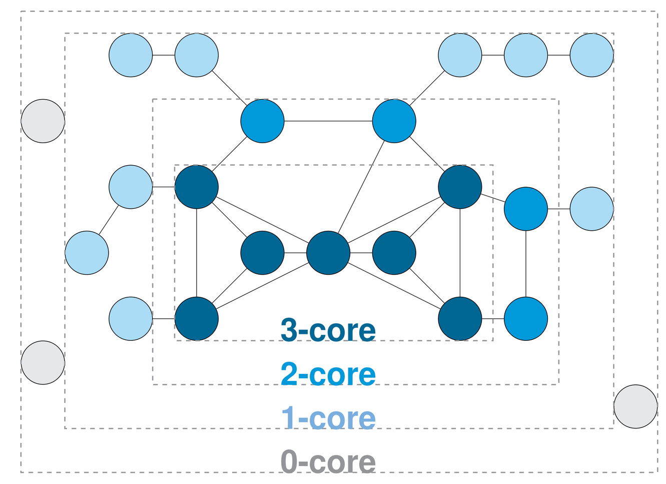
Despite promising results, graph AE and VAE suffer from scalability issues:

- Inner product decoding suffers from a $O(dn^2)$ complexity
- Training complex GNN encoders (e.g. spectral models, see [3]) might also be costly!

→ We introduce a framework to scale graph AE and VAE models to large graphs.
Key idea: optimize loss from a ***subset of nodes****, instead of using the entire graph* $\mathcal{G}$.

Step 1 - Identify dense parts of $\mathcal{G}$ by computing its core decomposition.

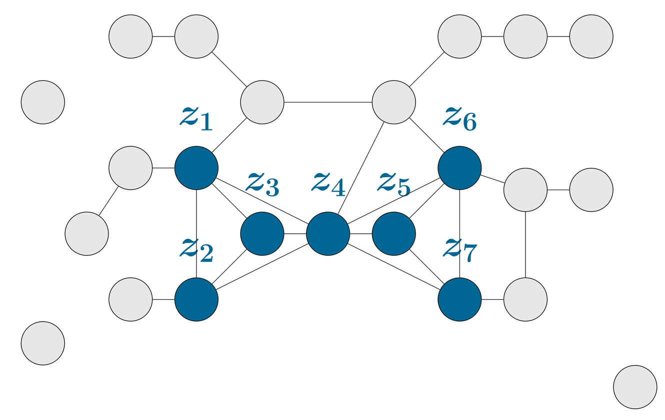
- **k-core** or **k-degenerate version of** $\mathcal{G}$ = largest subgraph of $\mathcal{G}$ for every node has a degree $\geq k$ within the subgraph.
- **Fast $O(m)$ computation** for undirected graphs.
- Effective tool to extract representative subgraphs [3].



↓

Step 2 - Train (V)AE on k-degenerate subgraph of $\mathcal{G}$.

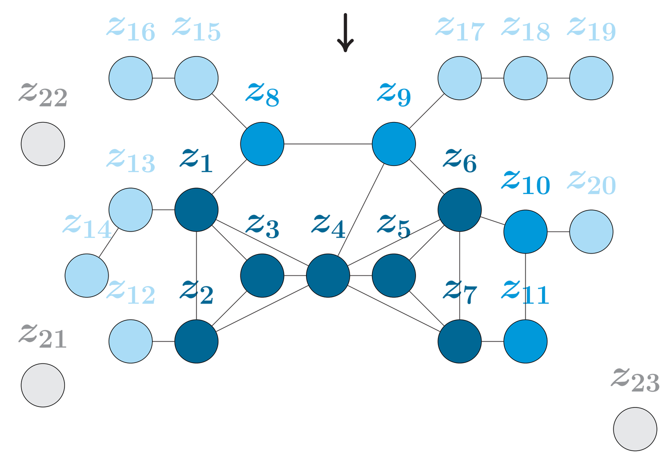
- We only derive latent vectors for this subgraph.
- Graph AE/VAE: still complex, but **now the input subgraph is much smaller than** $\mathcal{G}$.
- k is a parameter to tune (perf./speed trade-off).



↓

Step 3 - Infer other vectors using a simple propagation heuristic.

- We introduce a theoretically founded propagation scheme *(see paper [3] for technical details)*.
- Linear comb. of already learned latent vectors.
- Propagation in $O(m)$ **time complexity**.



V - Experimental Analysis

We provide an in-depth **evaluation of our framework** on:

- **5 real-world graphs:** Cora, Citeseer, Pubmed, Google, Patent (2.7K to 3M nodes).
- **10 variants of graph AE and VAE** models from existing literature.
- **2 graph learning tasks:** Link Prediction and Node Clustering

Model	Size of input k-core (nb nodes)	Mean Perf. on Test Set (in %)		Mean Running Times (in sec.)				
		AUC	AP	k-core dec.	Model train	Propagation	Total	Speed gain
VAE on $\mathcal{G}$	-	83.02 ± 0.13	**87.55 ± 0.18**	-	710.54	-	710.54	-
on 2-core	9 277 ± 25	**83.97 ± 0.39**	85.80 ± 0.49	1.35	159.15	0.31	160.81	x 4.42
on 3-core	5 551 ± 19	**83.92 ± 0.44**	85.49 ± 0.71	1.35	60.12	0.34	61.81	x 11.50
on 4-core	3 269 ± 30	82.40 ± 0.66	83.39 ± 0.75	1.35	22.14	0.36	23.85	x 29.79
on 5-core	1 843 ± 25	78.31 ± 1.48	79.21 ± 1.64	1.35	7.71	0.36	9.42	x 75.43
...	...	...	...	...	...	...	...	...
on 8-core	414 ± 89	67.27 ± 1.65	67.65 ± 2.00	1.35	1.55	0.38	**3.28**	x **216.63**
on 9-core	149 ± 93	61.92 ± 2.88	63.97 ± 2.86	1.35	1.14	0.38	**2.87**	x**247.57**
Spectral emb. *(best baseline)*	-	83.14 ± 0.42	86.55 ± 0.41	-	31.71	-	31.71	-

Table: Link Prediction on Pubmed graph (n=20K, m=44K), using graph VAE model from [2] on all cores

Model	Size of input k-core (nb nodes)	Mean Perf. on Test Set (in %)	Mean Running Times (in sec.)			
		Mutual Information	k-core dec.	Model train	Propagation	Total
VAE on 14-core	46 685	**25.22 ±1.51**	507.08	6 390.37	120.80	7 018.25 (1h57)
on 15-core	35 432	24.53 ± 1.62	507.08	2 589.95	123.95	3 220.98 (54min)
on 16-core	28 153	24.16 ± 1.96	507.08	1 569.78	123.14	2 200.00 (37min)
on 17-core	22 455	24.14 ± 2.01	507.08	898.27	124.02	1 529.37 (25min)
on 18-core	17 799	22.54 ± 1.98	507.08	551.83	126.67	**1 185.58 (20min)**
node2vec *(best baseline)*	-	**24.10 ±1.64**	-	26 126.01	-	26 126.01 (7h15)

Table: Node Clustering on Patent graph (n = 3M, m = 14M), using graph VAE model from [2] on 14 to 18 cores (over 64)
Note: the graph is too large to compare to "VAE on $\mathcal{G}$"... however, our approaches are competitive w.r.t. baselines

Main takeaways:

- Significant **scalability** improvement, while **performance** preserved for largest cores.
- Scaled AE/VAE are competitive w.r.t. DeepWalk, node2vec, LINE (+ spectral embedding for medium-size graphs)

Next steps:

- **Extending the framework to attributed graphs?** *See our experiments in [3]*
- **Extending graph AE/VAE to directed graphs?** *See our recent preprint [4]*
- **Current works in progress:**
  - Towards theoretical guarantees for k-core approximations
  - Graph AE/VAE for dynamic graphs
  - Graph AE/VAE for large-scale music recommendation

References

[1] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
[2] T. N. Kipf and M. Welling. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*, 2016.
[3] G. Salha, R. Hennequin, V. A. Tran, and M. Vazirgiannis. A degeneracy framework for scalable graph autoencoders. *IJCAI*, 2019.
[4] G. Salha, S. Limnios, R. Hennequin, V. A. Tran, and M. Vazirgiannis. Gravity-inspired graph autoencoders for directed link prediction. *Arxiv*, 2019.