Unveiling the Power and Limitations of Unsupervised Node Embeddings

Charalampos E. Tsourakakis

Joint works with:

Dan Chanpuriya
UMass→ UIUC→ Meta

Cameron Musco
UMass

Konstantinos Sotiropoulos
BU→ CMU →Meta

ML on Graphs is on the rise!

- **Node Classification**

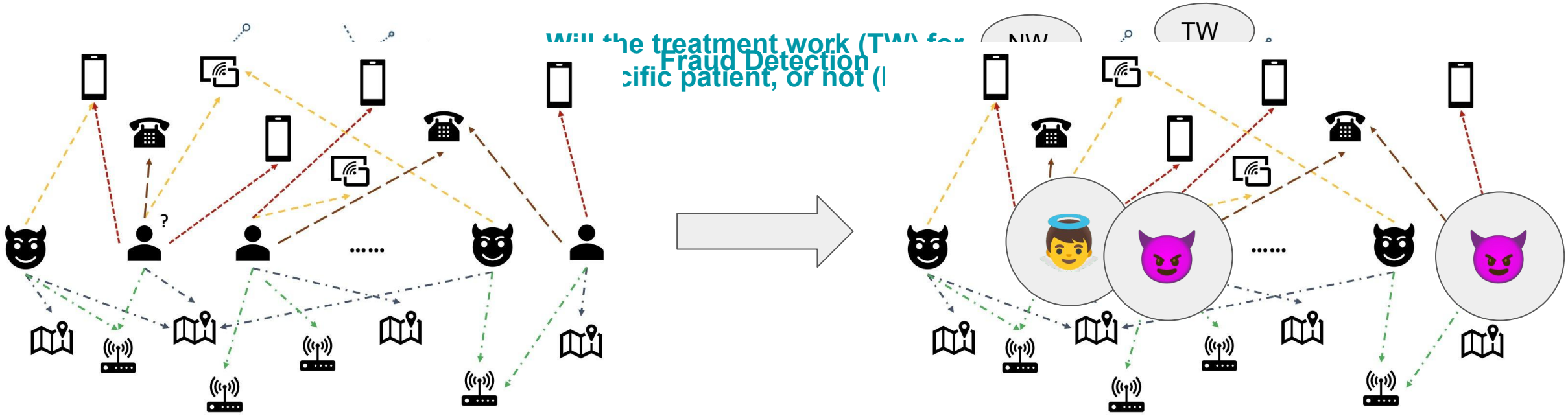
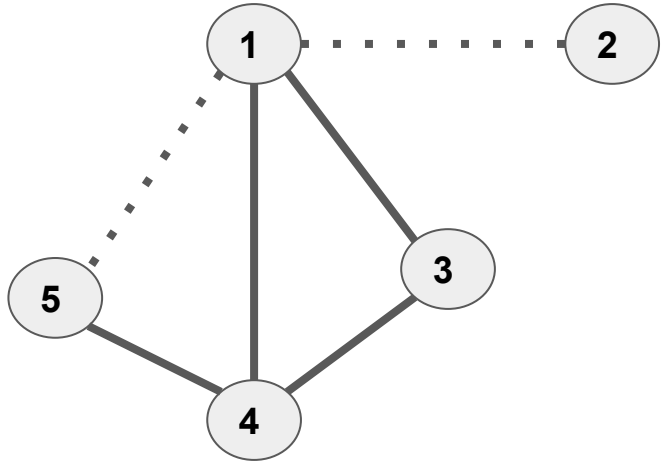


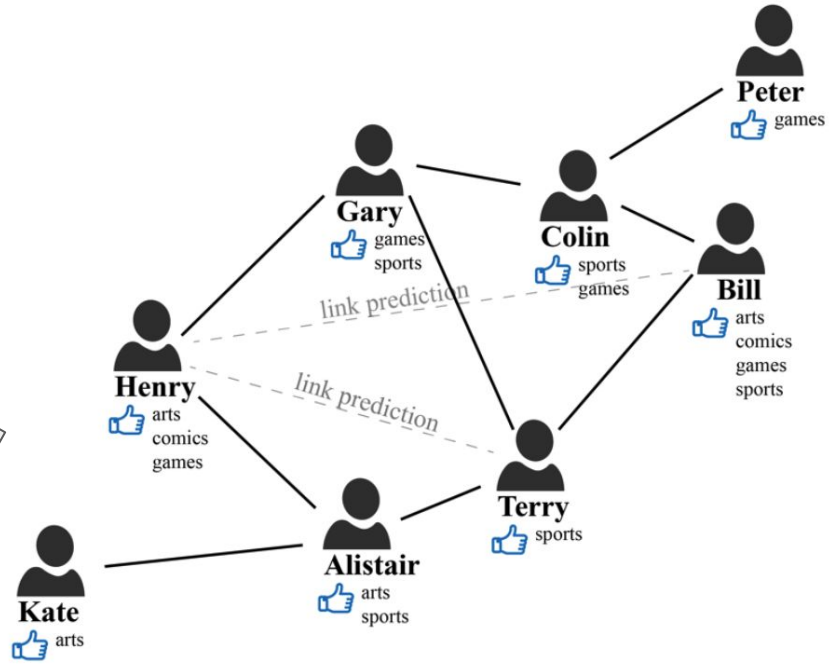
Image Source: https://aws.amazon.com/

ML on Graphs is on the rise!

- **Link Prediction**



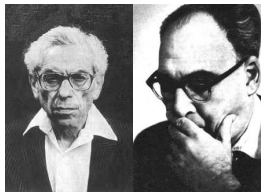
  - Friend Recommendation
  - Entity Resolution
  - Improve Marketing Strategies
  - …



[Source: AWS Amazon]

ML on Graphs is on the rise!

- Graph Generation
  - Model evolution of networks
  - Test null models (e.g. statistical significance of motifs)
  - Evaluate graph algorithms
  - Privacy
  - Discover new chemical and molecular structures



Erdős & Rényi, 1960

Watts-Strogatz, 1998

Chung-Lu, 2002

GraphVAE

GraphRNN

NetGAN

VGAE

Gilbert, 1959

Barabási-Albert-Bollobás-Riordan, 1999

What these tasks have in common?

- **(Unsupervised) Node Embeddings** are widely used as tools in <u>all</u> these tasks.
- Representation of nodes in a low-dimensional space $d \ll n$

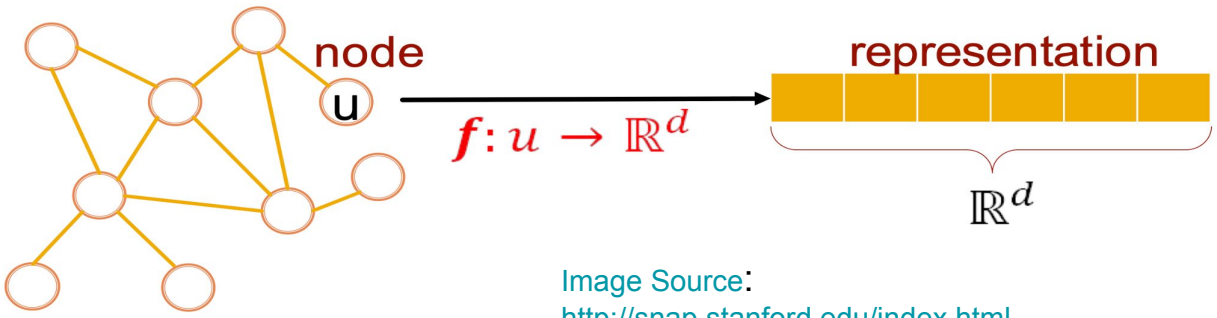
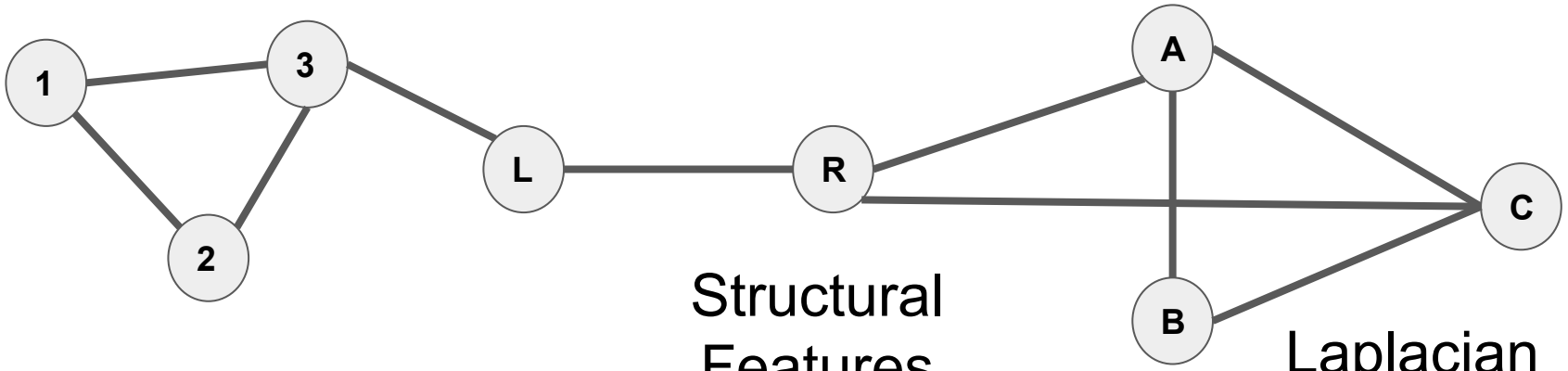


Image Source:
http://snap.stanford.edu/index.html

- **Why Unsupervised?**
  - Labels are not always available when we need them, e.g., in fraud detection. (Starnini et al., 2022)
  - Need to *better understand* unsupervised embeddings before we move to supervised.

How can we embed nodes?



Adjacency Vectors

1	0	1	1	0	0	0	0	0
L	0	0	1	0	1	0	0	0
R	0	0	0	1	0	1	0	1
C	0	0	0	0	1	1	1	0

Structural Features

(Degree, Triangles,Betweenness Centrality)

2	1	0
2	0	0.57
3	1	0.57
3	2	0.12

Laplacian Eigenvectors

0.09	0.28	0.45
0.38	-0.74	0.07
-0.65	-0.32	-0.23
-0.26	0.39	-0.40

Deep Learning

Word2Vec (Mikolov et al., 2003)

Image Source: Metropolitan Museum of Art

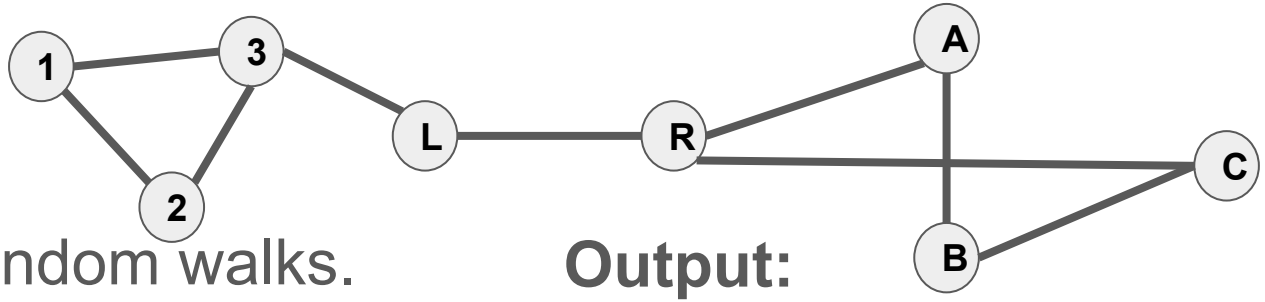
- **Skip-Gram model** (Mikolov et al., 2003) for word embeddings.
- **Input:** Collection of documents.

- **Target**: Embed words in a low-dimensional space & preserve semantic similarity.
- Words that co-appear within a window of size w in a corpus of sentences should have close representations.

- **Output**: https://projector.tensorflow.org/

DeepWalk (Perozzi, Al-Rfou & Skiena, 2014)

- **Document**:



- **Sentences:** Random walks.

  1, 3, 2, 3, L, R

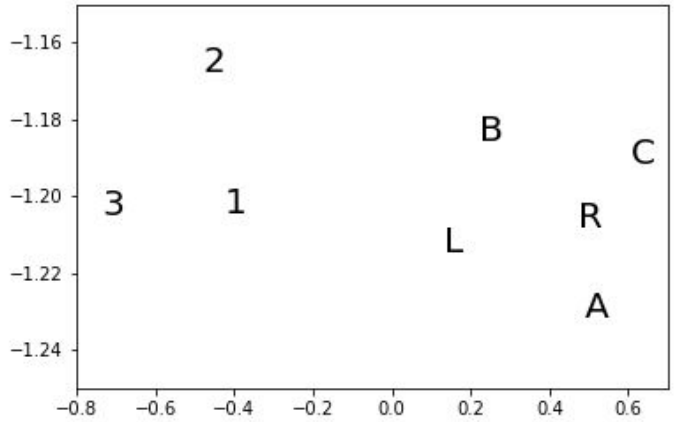
  3, 1, 2, 1, 2, 3

  R, A, R, B, C, R

  L , R, L, 3, 2, 1

  …

Output:



Node2Vec (n2v) (Leskovec & Grover, 2016)

- Uses higher-order random walks.
- Explores graph in a BFS or DFS fashion (tunable parameters).

NetMF (Qiu, Dong, Ma, Li, Wang & Tang, 2018)

- Provides a unifying framework for many embedding methods as matrix factorization.
  - Among them DeepWalk and Node2Vec.
- DeepWalk implicitly factorizes the following matrix:

$$PMI(G, \mathrm{w}) = M_{\mathrm{w}} = \log\left(\frac{\mathrm{vol}_G}{\mathrm{w}} \sum_{r=1}^{\mathrm{w}} \left(D^{-1} A\right)^r D^{-1}\right)$$

w: window size
D: diagonal matrix of degrees
vol_G: Volume of graph

- Why PMI? Pointwise Mutual Information from Information Theory.

$$\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \to \log\left(\frac{\mathrm{vol}_G}{\mathrm{w}} \sum_{r=1}^{\mathrm{w}} \left(D^{-1} A\right)^r D^{-1}\right)$$

- Very important tool to analyze node embeddings through the lens of low-rank matrix factorization!

Node embedding methods

- Spectral Clustering (Alon & Milman, 1985, Shi & Malik, 2000, Ng et al., 2002, …)
- IsoMap (Tenenbaum et al., 2000)
- Graphlets (Pržulj 2009), (Twittermancer)
- Locally Linear Embeddings (LLE) (Roweis & Saul, 2000)
- Laplacian Eigenmaps (Belkin & Niyogi, 2003)
- DeepWalk (Perozzi et al., 2014)
- node2vec (Grover & Leskovec, 2016)
- NetMF (Qiu et al., 2018)
- LINE (Tang et al.,, 2015)
- HARP (Chen et al., 2018)
- GraREP (Cao et al., 2015)
- ….

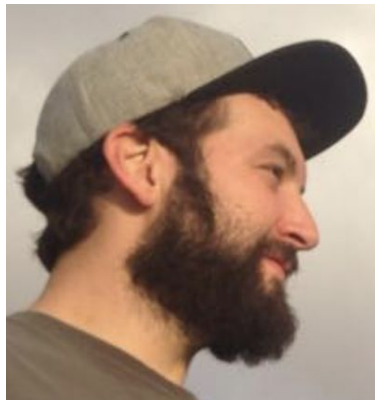
Deepwalk: Online learning of social representations
B Perozzi, R Al-Rfou, S Skiena - Proceedings of the 20th ACM SIGKDD …, 2014 - dl.acm.org
… (**DeepWalk**) that learns **social representations** of a … **learn social representations** with the following characteristics: • Adaptability - Real **social** networks are constantly evolving; new **social** …
☆ Save 99 Cite Cited by 8857 Related articles All 22 versions

node2vec: Scalable feature learning for networks
A Grover, J Leskovec - Proceedings of the 22nd ACM SIGKDD …, 2016 - dl.acm.org
… **node2vec**, an algorithmic framework for learning continuous feature representations for nodes in networks. In **node2vec**, … We demonstrate the efficacy of **node2vec** over existing state-of…
☆ Save 99 Cite Cited by 9323 Related articles All 24 versions

The impossibility of low-rank representations for triangle-rich complex networks

C. Seshadhri[a,1], Aneesh Sharma[b], Andrew Stolman[a], and Ashish Goel[c]

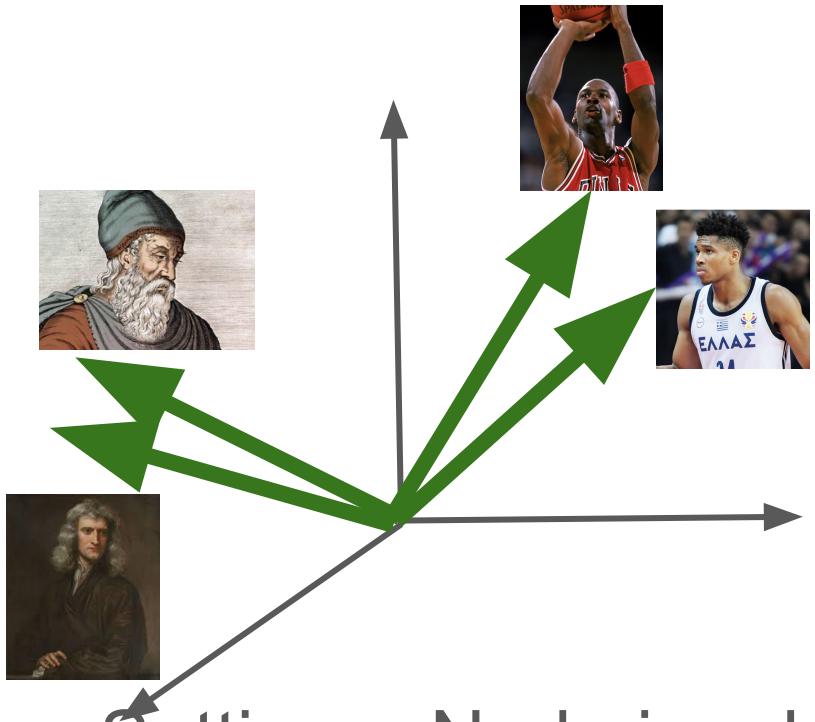


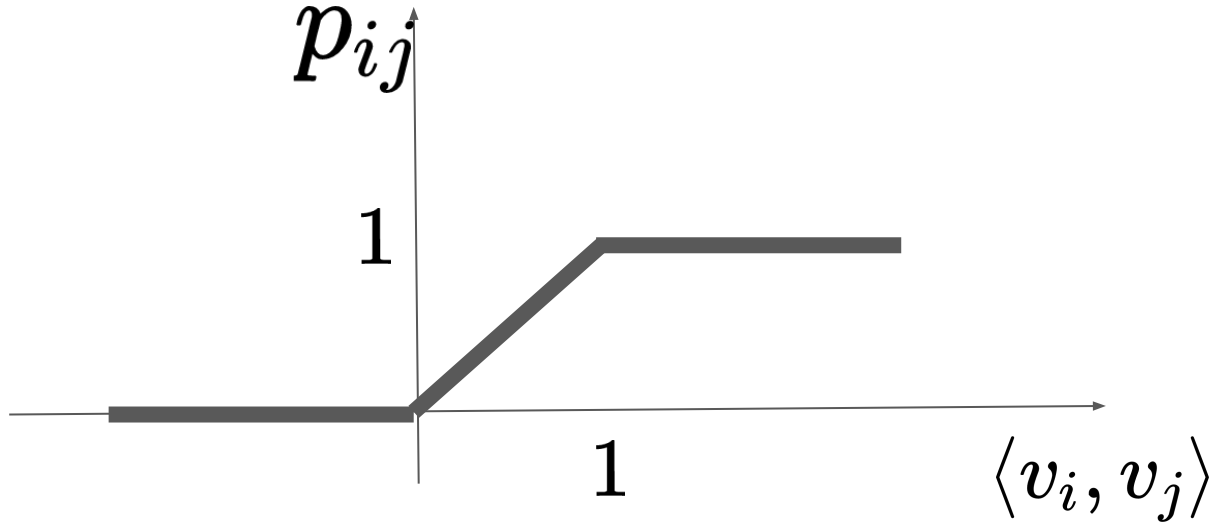



Seshadhri et al. took a different research path.

- What are the limitations of node embeddings?

Setting



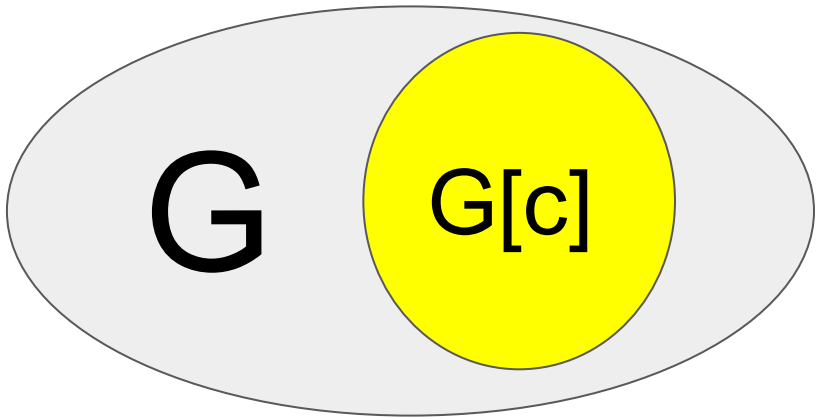


Setting: - Node i embedded as a vector v_i in d dimensions. In practice d<<n.

- Connection probability of i, j by the truncated dot product (TDP)

$$p_{ij} = \max(0, \min(\langle v_i, v_j \rangle, 1))$$

(c,Δ) triangle foundation [Seshadhri et al.]

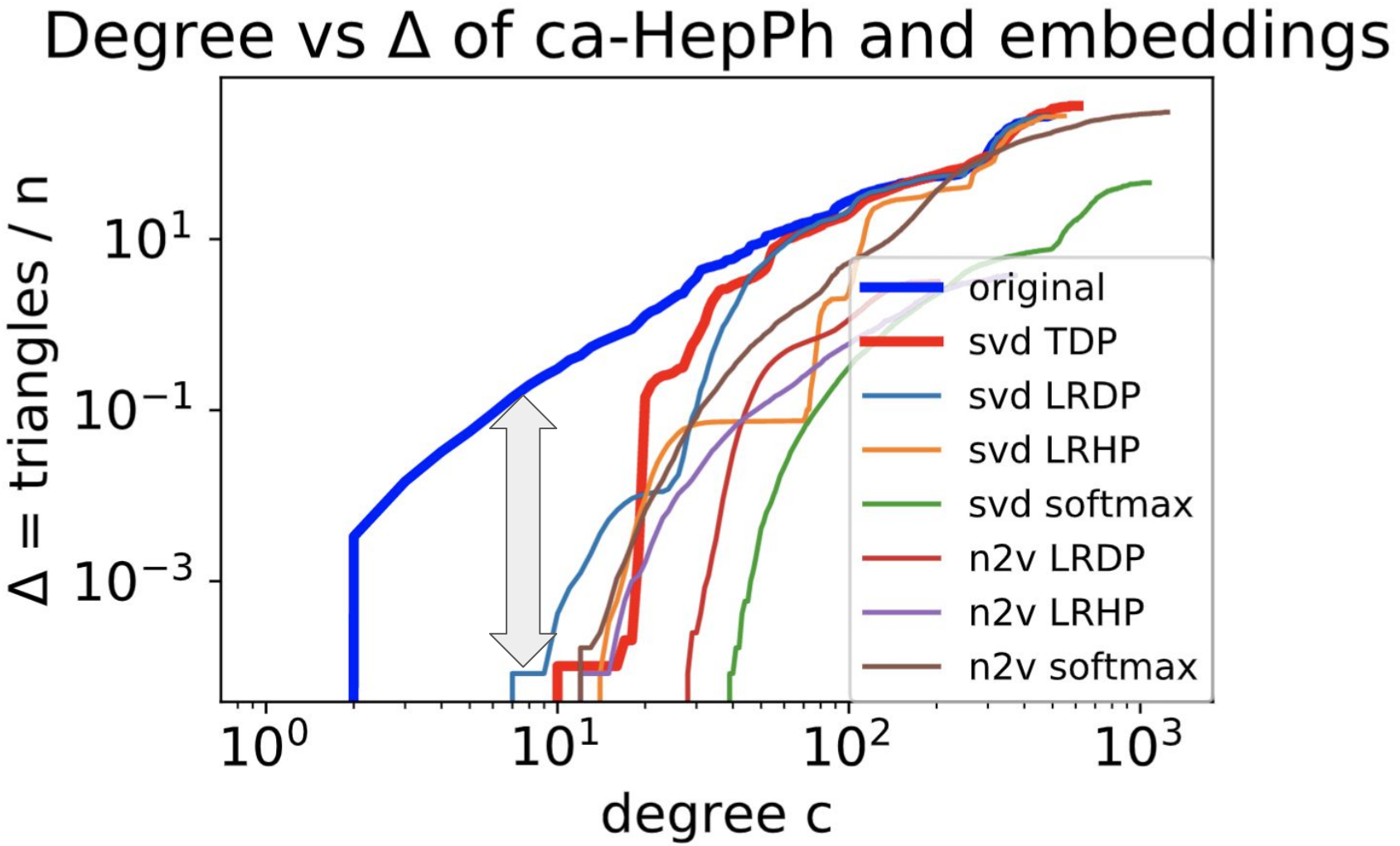


G[c] = subgraph induced by all nodes whose degree is at most c

n = #number of nodes in G

Define $\Delta(c) = \frac{\#\triangle s \text{ in } G[c]}{n}$

Empirical observation: linear number of triangles among constant degree vertices.



DP: dot product
HP: Hadamard product

T: truncated
LR: logistic regression

svd: singular value decomposition
n2v: node2vec

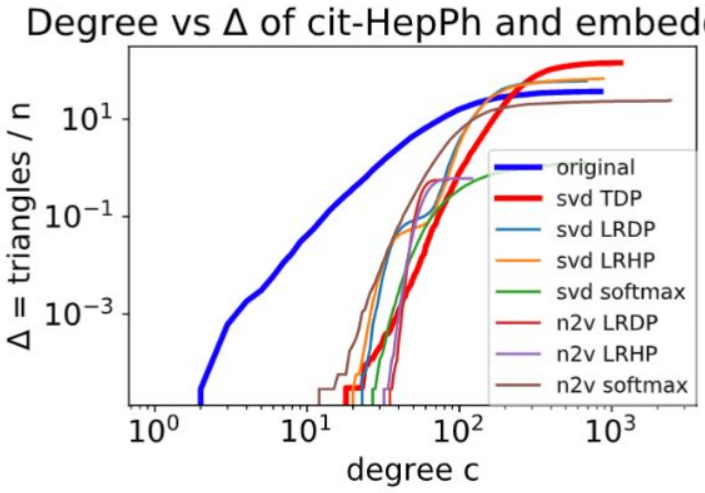
Theorem

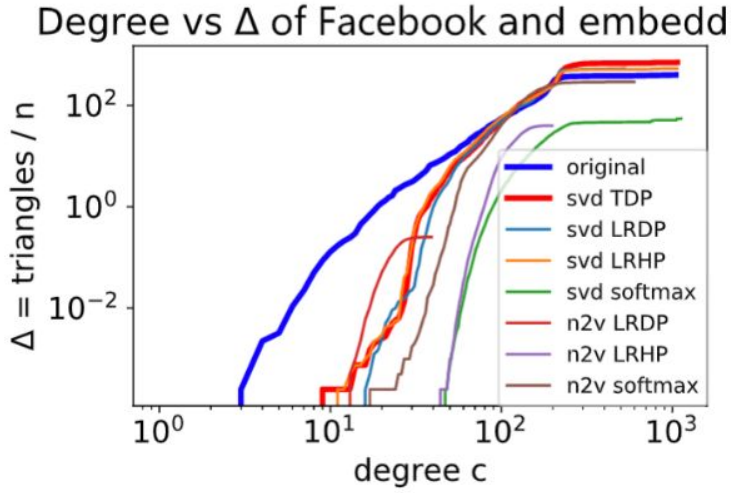
Fix c>4, Δ>0. Suppose the expected number of triangles in G[c] is at least Δn. Then, the dimensionality d is at least

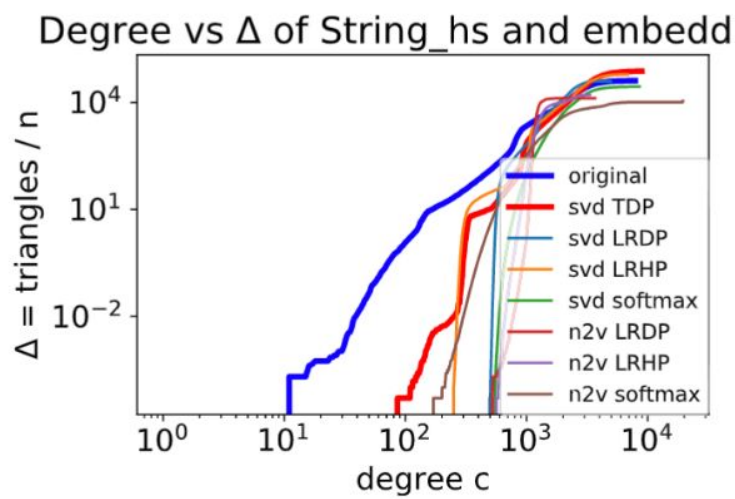
$$\min\left(1, poly\left(\frac{\Delta}{c}\right)\right)\frac{n}{\log^2 n}$$

- In other words, graphs generated from low-dimensional embeddings cannot contain many triangles only on low-degree vertices

Theory agrees with practice







- 100 dimensions and 100 graph samples, plotting the best among them. Still, the gap is evident!
- Equivalently, inherente limitations in representing triangle dense graphs if the embedding is given by $X \in \mathbb{R}^{n \times d}$ and the expected adjacency matrix is given

$$A = clip\left(XX^T\right), \text{where } clip \text{ clips entries to } [0,1].$$

Possibility Result

- (Seshadhri, Sharma, Stolman & Goel, PNAS 2020):

  “Low-dimensional embeddings can not generate graphs that have many triangles among low-degree nodes (sparse and triangle rich)”

- (Chanpuriya, Musco, Sotiropoulos & T, NeurIPS 2020):

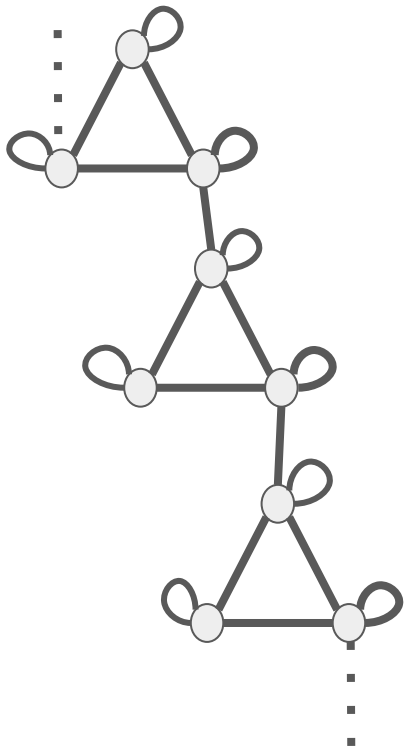
  “Low-dimensional embeddings can not only generate graphs that have many triangles among low-degree nodes (sparse and triangle rich), but exactly encode the graph!”

- How can both statements be true?

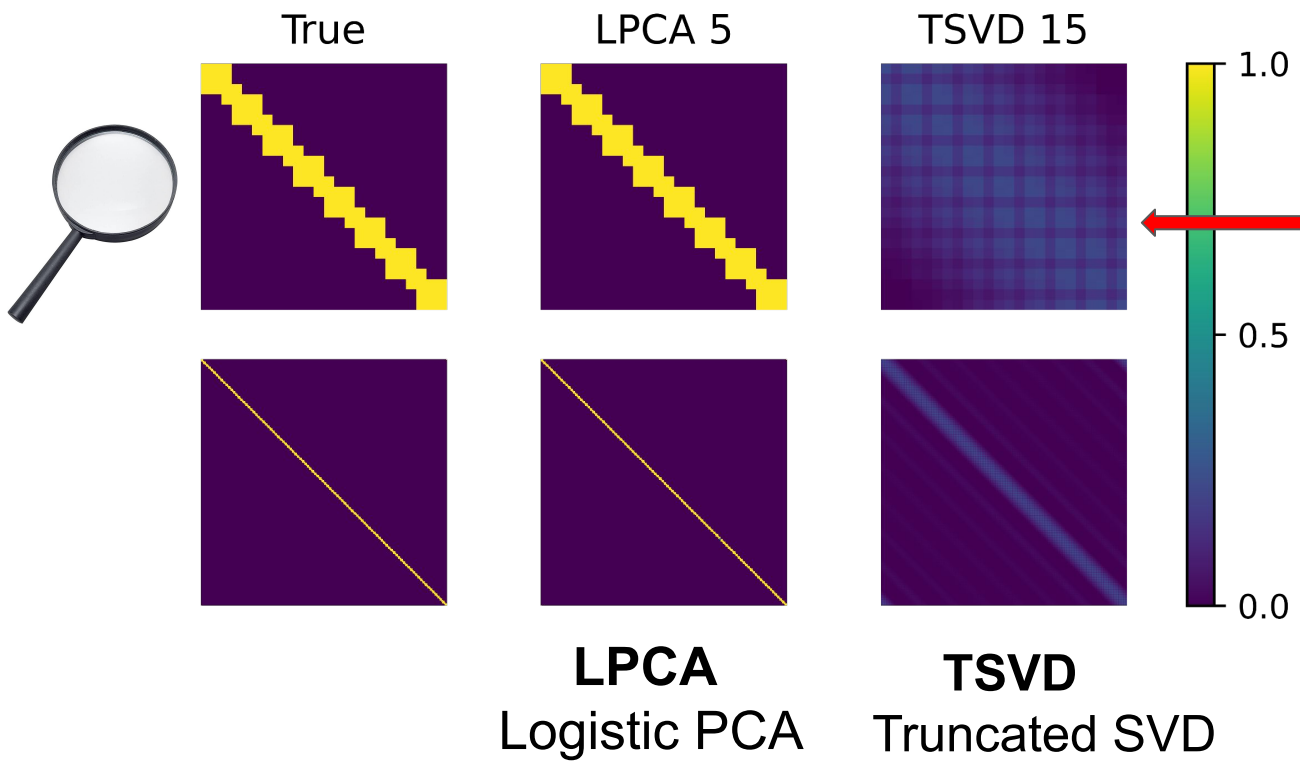
[CMST, NeurIPS 2020]

Reconstructing a sparse and triangle-dense graph

Graph

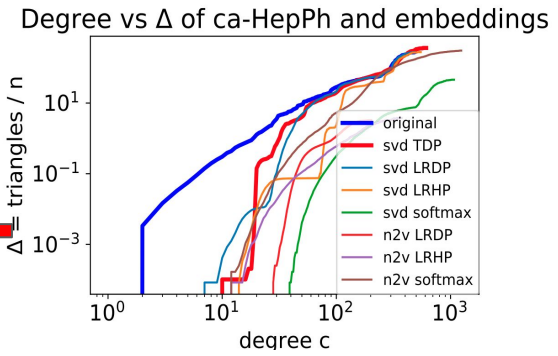


Reconstructions from Embeddings



LPCA
Logistic PCA

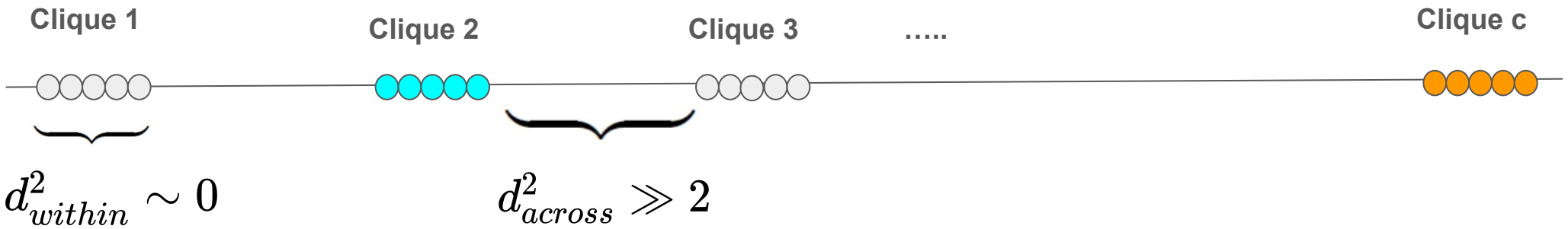
TSVD
Truncated SVD



Relaxing the PSD constraint (2 embeddings per node)

- **Observation**: By relaxing the PSD constraint of the dot-product model we can exactly factorize a sparse and triangle dense graph using only 3 dimensions.

Example: Let A be the adjacency matrix of a union of n/c cliques.



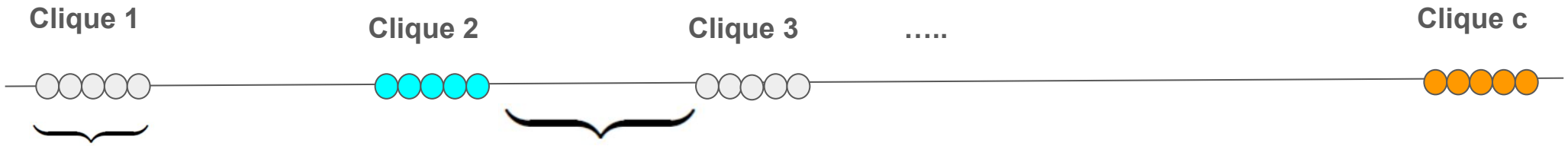
The matrix D_{dist} of squared euclidean distances has rank $\leq$ d+2 (d=1 in our case).

Consider $\tilde{A} = 2 - D_{dist}$.

$\tilde{A}$ has rank 3 and $A = clip\left(\tilde{A}\right) = clip\left(XY^T\right)$ where $X, Y \in \mathbb{R}^{n \times 3}$

Relaxing the PSD constraint (2 embeddings per node)

In the 1D case it is easy to see the rank-3 factorization of



$d^2_{within} \sim 0$ $\quad d^2_{across} \gg 2$

$$D_{dist} = \begin{bmatrix} 0 & & & & \\ & 0 & & & \\ & & \ddots & (x_i - x_j)^2 & \\ & & (x_i - x_j)^2 & 0 & \\ & & & & 0 \end{bmatrix}$$

Define $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, y = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_n^2 \end{bmatrix}, \mathbb{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

$$\mathbb{1}y^T + y\mathbb{1}^{\mathbb{T}} - 2xx^T = \begin{bmatrix} x_1^2 & \cdots & x_n^2 \\ \vdots & \ddots & \vdots \\ x_1^2 & \cdots & x_n^2 \end{bmatrix} + \begin{bmatrix} x_1^2 & \cdots & x_1^2 \\ \vdots & \ddots & \vdots \\ x_n^2 & \cdots & x_n^2 \end{bmatrix} - 2\begin{bmatrix} x_1^2 & \cdots & x_1 x_n \\ \vdots & \ddots & \vdots \\ x_1 x_n & \cdots & x_n^2 \end{bmatrix} = D_{dist}$$

Connection with sign-rank

$$A = \text{clip}(XX^T) \rightarrow A = \text{clip}(XY^T)$$

1. Matrix is PSD. Relax this by considering XY^T.
2. We apply a function that clips entries to [0,1]. Think of A as {-1,1} instead of {0,1}. We are interested in a low-rank factorization that ***matches A sign-wise***.

Definition [Sign-rank]: The sign rank of a real matrix A, with entries {-1,1} is the least rank of a matrix B, such that $A_{ij} \cdot B_{ij} > 0$ for all i,j.

Studied in TCS due to connections with circuit complexity (Razborov & Sherstov, 2010), (Bun & Thaler, 2016), communication complexity (Alon et al., 1985), (Linial et al., 2007), learning theory (Alon et al., 2016).

Key Theorem

[CMST, NeurIPS 2020]

For any graph G with maximum degree c, there exist factors $X, Y \in \mathbb{R}^{n\times(2c+1)}$ of dimension 2c+1 , such that $A = clip\left(XY^T\right)$

Proof sketch: Let $V^{nx(2\Delta+1)}$ be the Vandermonde matrix evaluated at 1,..n

$$V^{n\times(2c+1)} = \begin{bmatrix} 1^0 & 1^1 & \cdots & 1^{2c} \\ 2^0 & 2^1 & \cdots & 2^{2c} \\ 3^0 & 3^1 & \cdots & 3^{2c} \\ 4^0 & 4^1 & \cdots & 4^{2c} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

Notice that for any vector x^{2c+1}, $(Vx)[t] = \sum_{i=0}^{2c} x_i t^i$

Key Theorem

- Let a_i be the i-th row of the adjacency matrix A.
  - It has at most c ones.

- For each node i, we find a polynomial of degree 2c which is positive at all integers t with $a_i(t)=1$ and negative at all t with $a_i(t)=0$.
  - Straight-forward interpolation
  - The coefficients X give us $A=\text{sign}(VX^T)$

Corollary: preferential attachment graphs have maximum degree $O(\sqrt{n})$ with high probability, so it admits sublinear embedding.

Follow-up result: $O(\alpha^2)$ dims suffice, α=arboricity

Logistic PCA (LPCA)

- We also provide an algorithm that learns embeddings $X, Y \in \mathbb{R}^{n \times k}$ such that $A = clip\left(XY^T\right)$ using gradient-descent on the following logistic loss:

$$loss = -\sum_{i,j} \sigma\left(\tilde{A_{ij}} \cdot \left[XY^T\right]_{ij}\right), \text{ where } \sigma \text{ is the sigmoid.}$$

- We can exactly factorize real-world networks using very few dimensions.

Dataset	# Nodes	# Edges	$2\Delta + 1$	**Exact Empirical Factorization Dimension**
Pubmed	19 581	43 847	343	**48**
ca-HepPh	11 204	117 634	983	**32**
BlogCatalog	10 312	333 983	7 985	**128**
Citeseer	3 327	4 552	199	**16**
Cora	2 708	5278	337	**16**

[CMS**T**, ICML 2021]

Are the embeddings of DeepWalk exact?

- LPCA can exactly encode the input graph.

- It is not clear what DeepWalk (or NetMF) embeddings encode.

Question:
“What information of the input graph
do some of the most popular
modern embeddings encode?”

[CMST, ICML 2021]

Our contribution

- Invert Node Embeddings:
  - We invert NetMF embeddings by learning graphs that have similar embeddings to the input graph.
  - Study the properties of these graphs to understand what is encoded in low-dimensions.
  - How is this information relevant to downstream ML tasks?

- **Problem Formulation**
  Given an embedding algorithm $\mathcal{E} : \mathcal{G} \rightarrow \mathbb{R}^{n \times d}$ and the embedding $\mathcal{E}(\mathcal{G})$ for some $G \in \mathcal{G}$, produce $G \in \tilde{\mathcal{G}}$ such that $\left|\left|\mathcal{E}\left(\tilde{G}\right) - \mathcal{E}(G)\right|\right|$ is small. How close are $G, \tilde{G}$ in terms of, e.g., edges, degrees, triangles, community structure?

Algorithm 1 for d=n, w⟶ inf - Solve a linear system!

- In the limit, as $\mathbf{w} \to \infty$, using a result from Chanpuriya & Musco (2020) we know that:

$$M_{\infty}^{n \times n} = \mathrm{vol}_G D^{-1/2} \left(L^{\dagger}_{normalized} - I \right) D^{-1/2} + \mathbb{1}\mathbb{1}^T$$

- Solving a linear system yields the pseudo-inverse
- Cons, in practice:
  1. w is never set to be infinite, instead typically a small value
  2. The embeddings in practice are low dimensional, not full/high dimensional.

Algorithm 2 - Practical PMI optimization algorithm

- For typical settings, w=10 and low-rank, we employ an optimization-based approach.
- Given the low-rank embedding $M_{\mathrm{w},d}$ of a graph G, our algorithm finds a graph $\tilde{G}$ of same expected volume with a similar embedding.

$$\min_{\tilde{G}} \left|\left| PMI\left(\tilde{G}, \mathrm{w}\right) - M_{\mathrm{w},d} \right|\right|_F^2$$

- We apply this algorithm to invert embeddings of popular benchmark graphs in node classification tasks.

Experimental Setting

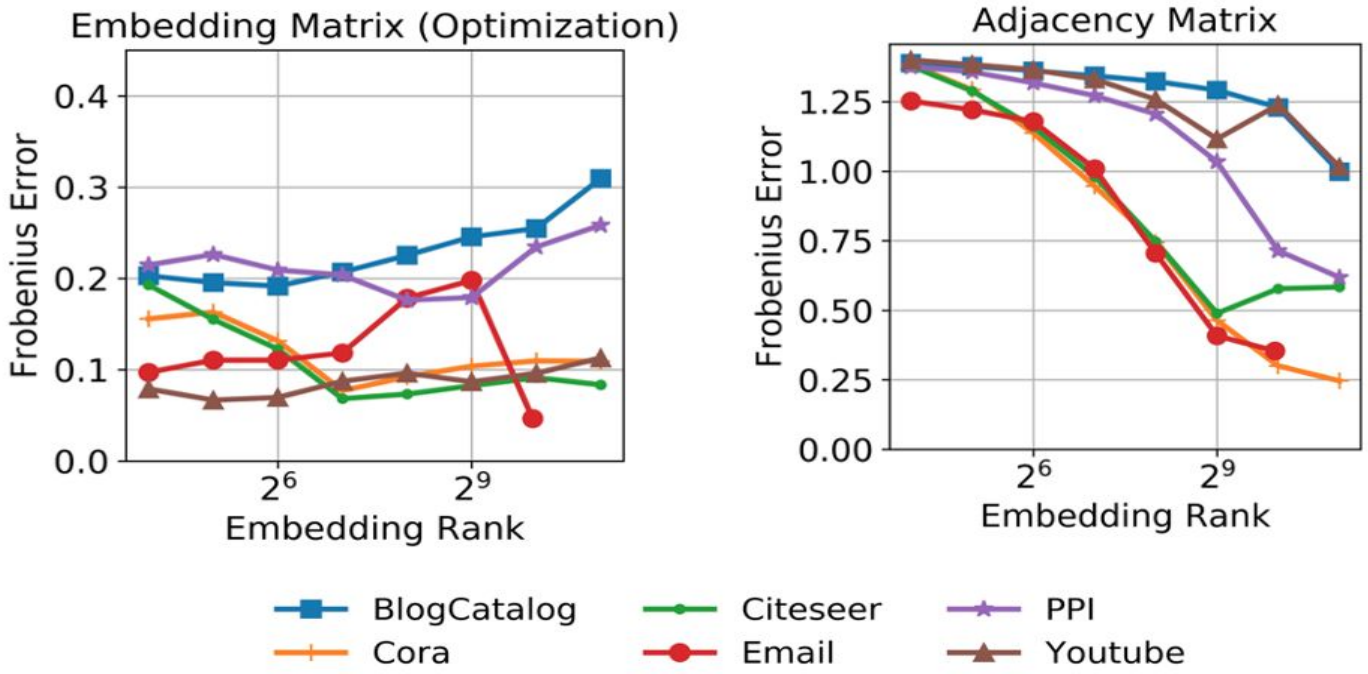
- We use 6 popular benchmark datasets in node classification (#labels): BlogCatalog **(**39), Cora (7), Citeseer (6), Email (42), PPI (50), YouTube (20)
- We use PyTorch, auto-differentiation and SciPy for the optimization algorithm.
- Reconstruction errors for embeddings and adjacency matrices $\tilde{M}_r$ and $\tilde{A}_r$ (rank r) are reported using the (relative) Frobenius Error w.r.t. the input graph M and A.

$$\frac{\left|\left|X - \tilde{X}\right|\right|_F}{\left|\left|X\right|\right|_F}$$

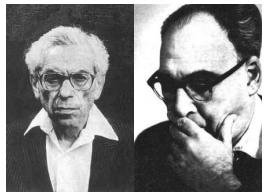
- We evaluate embeddings in node classification tasks. We use one-vs-rest logistic regression classifier and report the Micro-F1 score (global F1 score).

Results

- In the y-axis we report the Frobenius Error as we vary the embedding dimensions (x-axis).
  1) We observe the embedding error is small (and not sensitive to dimension).
  2) Despite having similar embeddings, reconstructed networks are very different in low ranks.



Graph Generative Models


Erdős & Rényi, 1960


Watts-Strogatz, 1998


Chung-Lu, 2002

GraphVAE

GraphRNN

NetGAN

VGAE

Gilbert, 1959

Barabási-Albert-Bollobás-Riordan, 1999

Our contributions [CMS**T** NeurIPS ‘21]:

- Embeddings used to form an edge independent model are limited, there is an inherent trade-off between expressivity and memorization.
- “*Study graph generative models under the lens of how capable they are to mimic the original network while generating “edge diverse” samples.*”

Graph Generative Models

More recently we see deep-learning based models.

- Given an input graph G, …

- **… learn** a distribution and sample new graphs *similar* to G.
  - NetGAN (Bojchevski et al., 2018)
  - VGAE (Kipf & Welling, 2016)
  - GVAE (Simonovski & Komodakis 2018)
  - GraphRNN (You et al., 2018)
  - ….

Evaluation of these models

- We want generated graphs to be “similar” to the input graph.
- Match statistics of the input graph:
  - Degree distribution
  - Triangle count
  - Path length
  - Clustering coefficient (transitivity)
  - ...
- We can simply return the input graph, but…
  - We want a graph similar, but not identical to the input graph.
  - We also want samples to be edge-diverse.
- **Target**: Match graph statistics, while generating edge-diverse samples.

Motivation

- Complex Deep-Learning based models, like NetGAN (Bojchevski et al., 2018) or CELL (Rendsburg et al, 2020) learn a matrix $P \in [0,1]^{n \times n}$ and sample edges independently.
- They stop training before sampled graphs overlap a lot with the input graph.
- Generated graphs usually have few triangles!

GRAPH		TRIANGLE COUNT
CORA-ML		2,802
NETGAN	(54% EO)	1,461
CELL	(53% EO)	1,396

GRAPH		TRIANGLE COUNT
WEB-EDU		4491
NETGAN	(53% EO)	1,244
CELL	(54% EO)	1,176

- Is this a coincidence? Namely, are there some limitations of edge independent models?

A few Preliminaries

- Our focus: Undirected, unweighted & labeled graphs (V=[n]).
- Let $\mathcal{U}, \mathcal{V}$ be distributions over $\binom{n}{2}$-dimensional vectors.

  Inner Product: $\langle \mathcal{U}, \mathcal{V} \rangle := \mathop{\mathbb{E}}_{\mathbf{u} \sim \mathcal{U},\, \mathbf{v} \sim \mathcal{V}} [\mathbf{u} \cdot \mathbf{v}]$

- Suppose $\mathcal{A}$ is a distribution over adjacency matrices of undirected graphs on n nodes.

- Let $\mathcal{F}$ be the distribution that always returns the complete graph K_n.

We define the expected volume as $\mathrm{Vol}(\mathcal{A}) := \langle \mathcal{A}, \mathcal{F} \rangle$

Overlap

- Suppose $\mathcal{A}$ is a distribution over adjacency matrices of labeled, undirected graphs on n nodes. We define the overlap of $\mathcal{A}$ as:

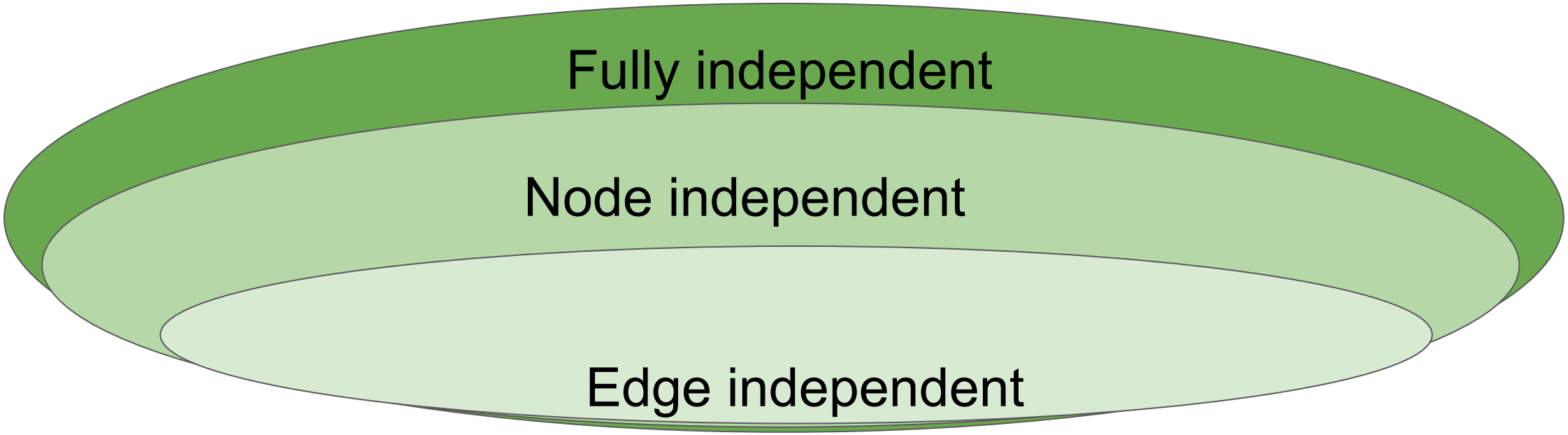
$$\boxed{\mathrm{Ov}(\mathcal{A}) \ := \frac{\langle \mathcal{A}, \mathcal{A} \rangle}{\mathrm{Vol}(\mathcal{A})}}$$

- **Meaning**: Expected fraction of edges that will overlap among two samples from the same distribution.
- Connection between Overlap and Volume (from Cauchy-Schwartz):

$$\textbf{Lemma 3: } \mathrm{Vol}(\mathcal{A}) \leq \binom{n}{2} \cdot \mathrm{Ov}(\mathcal{A})$$

*"**Question:** For a given overlap, how well can different graph generative models mimic properties of the input graph, e.g., number of triangles?"*

A Hierarchy of Graph Generative Models

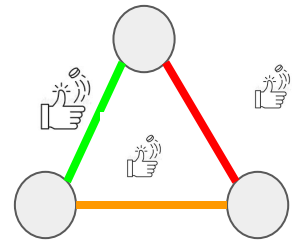


The node independent model instead of embedding each node to a vector, it gives to each node an independent distribution over embedding vectors.

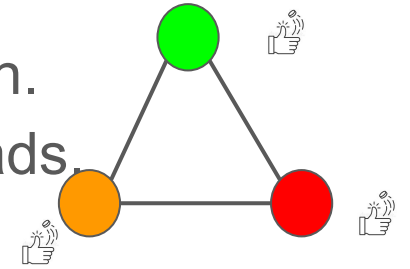
Examples

Coin tosses are independent!

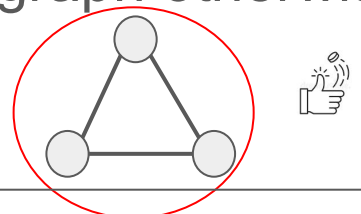
- **Edge Independent Model $\mathcal{A}_{E.I.}$**
  - Toss a coin with probability p for each edge in the graph.
  - Probability of an edge = $\mathrm{Ov}(\mathcal{A}_{E.I.})$ = p.
  - Number of triangles $\approx n^3 \cdot \mathrm{Ov}(\mathcal{A}_{E.I.})^3$



- **Node Independent Model $\mathcal{A}_{N.I.}$**
  - Toss a coin with probability $\sqrt{p}$ for each node in the graph.
  - Create a clique among all nodes that coin tosses are heads.
  - Probability of an edge = $\mathrm{Ov}(\mathcal{A}_{N.I.})$ = p.
  - Number of triangles $\approx n^3 \cdot \mathrm{Ov}(\mathcal{A}_{E.I.})^{3/2}$



- **Fully Dependent Model $\mathcal{A}_{F.D.}$**
  - With probability p create a complete graph, or empty graph otherwise.
  - Probability of an edge = $\mathrm{Ov}(\mathcal{A}_{F.D.})$= p.
  - Number of triangles $\approx n^3 \cdot \mathrm{Ov}(\mathcal{A}_{F.D.})$



A Hierarchy of Graph Generative Models

- **Edge Independent Models** [Edges are independent]
  A distribution $\mathcal{A}$ over symmetric adjacency matrices **A**, such that for some $\mathbf{P} \in [0,1]^{n \times n}$, $\mathbf{A}_{ij} = \mathbf{A}_{ji} = 1$ with probability $\mathbf{P}_{ij}$ independently for all i,j.

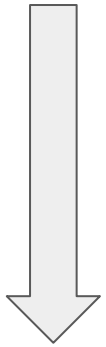
- **Node Independent Models** [Models dependencies at node level]
  A distribution $\mathcal{A}$ over symmetric adjacency matrices $A = \left[XY^T > 0\right] \in \{0,1\}^{n \times n}$ where **X**,**Y** are random matrices and the rows of [XY] are mutually independent.
  (*[M>0] sets all positive entries to 1, and 0 otherwise.)*

- **Fully Dependent Models** [Models any kind of dependencies]
  Any distribution $\mathcal{A}$ over symmetric adjacency matrices.

Summary of Results

- Turns out these examples provide (asymptotically) tight bounds for the expected number of triangles, graphs generated from these models can have.

Model	Upper Bound in #triangles
Edge Independent Model	$n^3 \cdot \mathrm{Ov}(\mathcal{A})^3$
Node Independent Model	$n^3 \cdot \mathrm{Ov}(\mathcal{A})^{3/2}$
Fully Dependent Model	$n^3 \cdot \mathrm{Ov}(\mathcal{A})$

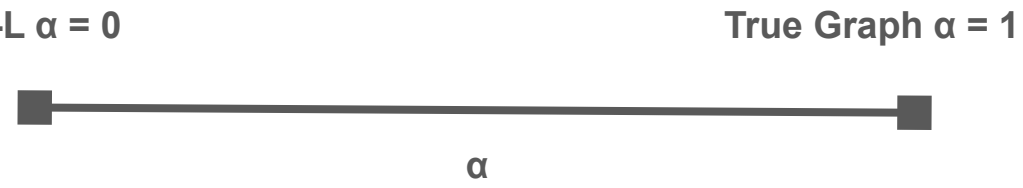


Recall Ov(A)<=1.

Putting these ideas in practice.

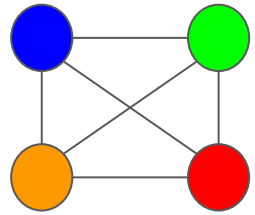
- We design a simple baseline for each model in the hierarchy.
- Desiderata:
  - Easy to range overlap.
  - Be able to reproduce (in the limit) the input graph.

- **CCOP (E.I)**: Edge independent model.
  Convex combination of input graph and random graph that preserves degree sequence on expectation.

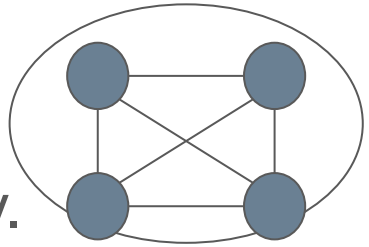


Putting these ideas in practice.

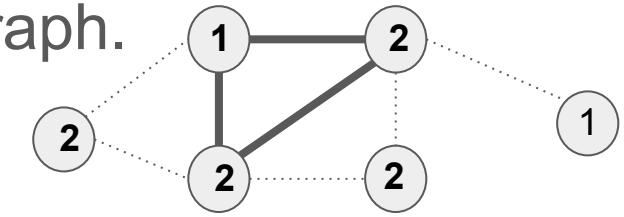
- Common idea for models with dependency.
- List all maximal cliques of the input graph.
  - **MCPROP (N.I.)**
    For each maximal clique, retain each node with probability p, independently.



  - **MCPLOP (F.D.)**
    Include each maximal clique with probability p, independently.



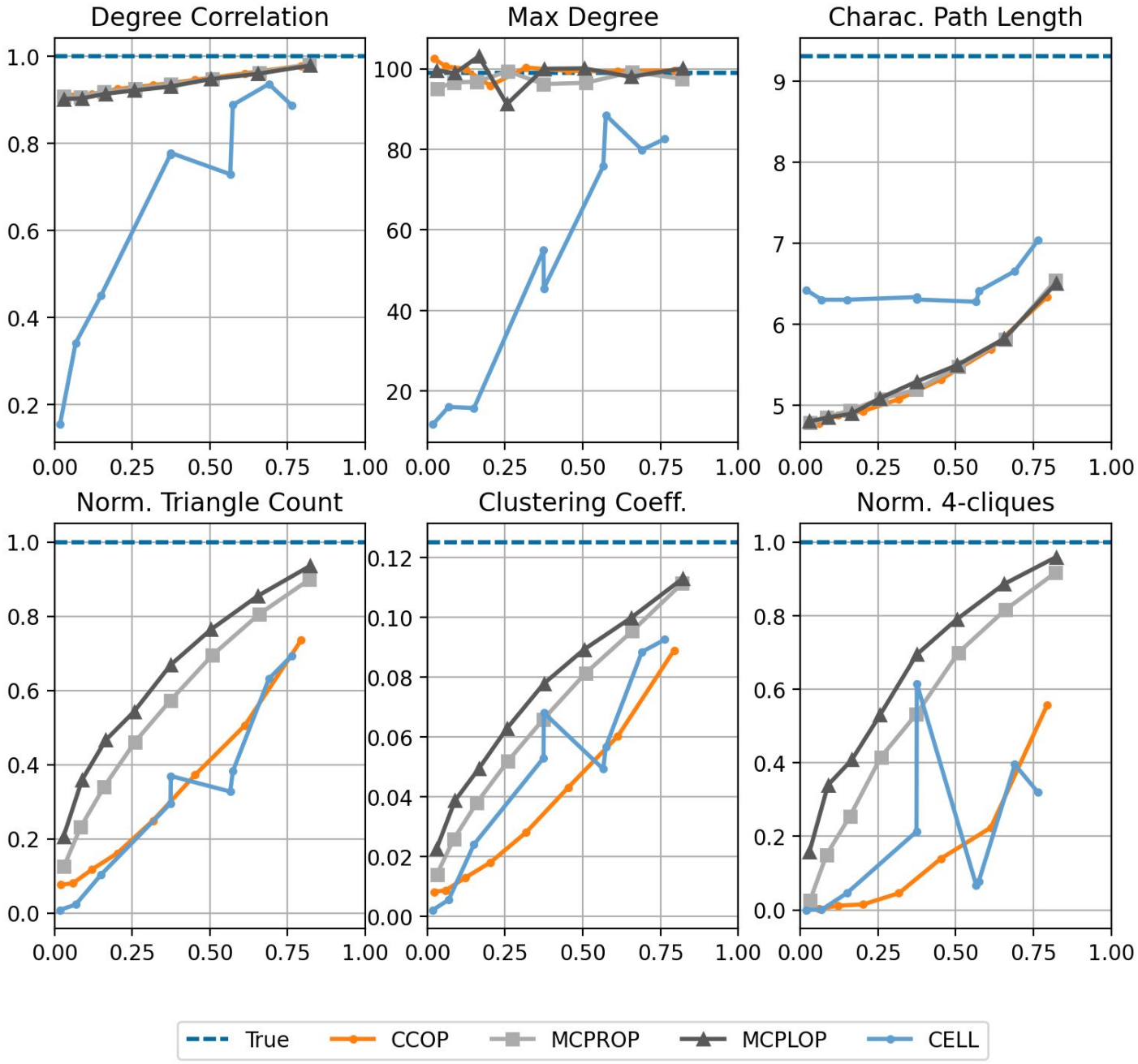
- Fit a random graph model that preserves degree sequence in expectation on the residual graph.



Experimental Evaluation

- We use 5 publicly available datasets:
  - Citeseer, Cora, PolBlogs, PPI, Web-Edu
- Datasets range from more sparse (e.g., Citeseer with 2.1K nodes and 7.3K edges) to more dense (e.g., PolBlogs with 1.2K nodes and 33.4K edges).
- We compare against CELL, a variant of the popular NetGAN method that applies then GAN framework on random walks.
  - Following the authors of CELL, we perform early-stopping to prevent from simply reproducing the input graph.

- We observe our baselines are competitive to the CELL method, and usually outperform it in matching counts of dense subgraphs in low overlap.



Citeseer Dataset

x-axis: Overlap,
y-axis: graph statistics

- Our baselines, especially those introducing dependencies, achieve higher triangle density in low overlap compared to CELL.
- By design, they also do well in matching degree sequence.
- CELL does better in average path length.

Importance of some of our key results

- Exact encoding of a graph with low dimensional embeddings
  - Sign rank as a key tool for understanding exact compression.
  - Motivates the use of mixtures of embeddings.

- We can go back from embeddings to graphs
  - Studying salient and non-salient characteristics.
  - Raises question about the privacy of node embeddings.

- We provide a unique lens of studying graph generative models.
  - Concept of overlap for refined random graph model evaluation.
  - We give a theoretical explanation to empirical findings on edge independent models.
  - Novel and simple baselines that are competitive to state-of-the-art models.

Open Questions

- How can we convert our deeper understanding to efficient practical methods?
  - Mixtures of embeddings?
- What is a good choice for the number of dimensions given the dense subgraph structure of the graph?
- How do we extend our findings to unlabeled graphs and directed graphs?

References

(Broder et al., 2000) Broder, Andrei, et al. "Graph structure in the web." *Computer networks* 33.1-6 (2000): 309-320.

(Mikolov et al., 2013) Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *NIPS 2013*.

(Perozzi et al., 2014) Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." *KDD* 2014.

(Grover & Leskovec, 2016) Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." *KDD* 2016.

(Qiu et al., 2018) Qiu, Jiezhong, et al. "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec." *WSDM* 2018.

(Starnini et al., 2021) Starnini, Michele, et al. "Smurf-based anti-money laundering in time-evolving transaction networks." *ECML PKDD 2021.*

*(*Erdős & Rényi, 1960) Erdős, Paul, and Alfréd Rényi. "On the evolution of random graphs." *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960): 17-60.

(Gilbert, 1959) Gilbert, Edgar N. "Random graphs." *The Annals of Mathematical Statistics* 30.4 (1959): 1141-1144.

References (2)

(Watts & Strogatz, 1998) Watts, Duncan J., and Steven H. Strogatz. "Collective dynamics of ‘small-world’networks." *nature* 393.6684 (1998): 440-442.

(Barabási and Albert, 1999) Barabási, Albert-László, and Réka Albert. "Emergence of scaling in random networks." *science* 286.5439 (1999): 509-512.

(Chung & Lu, 2002) Chung, Fan, and Linyuan Lu. "Connected components in random graphs with given expected degree sequences." *Annals of combinatorics* 6.2 (2002): 125-145.

(Simonovsky & Komodakis, 2018) Simonovsky, Martin, and Nikos Komodakis. "Graphvae: Towards generation of small graphs using variational autoencoders." *ICANN*, 2018.

(Kipf & Welling, 2016) Kipf, Thomas N., and Max Welling. "Variational graph auto-encoders." *arXiv preprint arXiv:1611.07308* (2016).

(You et al., 2018) You, Jiaxuan, et al. "Graphrnn: Generating realistic graphs with deep auto-regressive models." *International conference on machine learning*. PMLR, 2018.

(Bojchevski et al., 2018) Bojchevski, Aleksandar, et al. "Netgan: Generating graphs via random walks." *International conference on machine learning*. PMLR, 2018.

References (3)

(Rendsburg et al, 2020) Rendsburg, Luca, Holger Heidrich, and Ulrike Von Luxburg. "Netgan without gan: From random walks to low-rank approximations." *International Conference on Machine Learning*. PMLR, 2020.