Convergence and Stability of Graph Convolutional Networks on Large Random Graphs

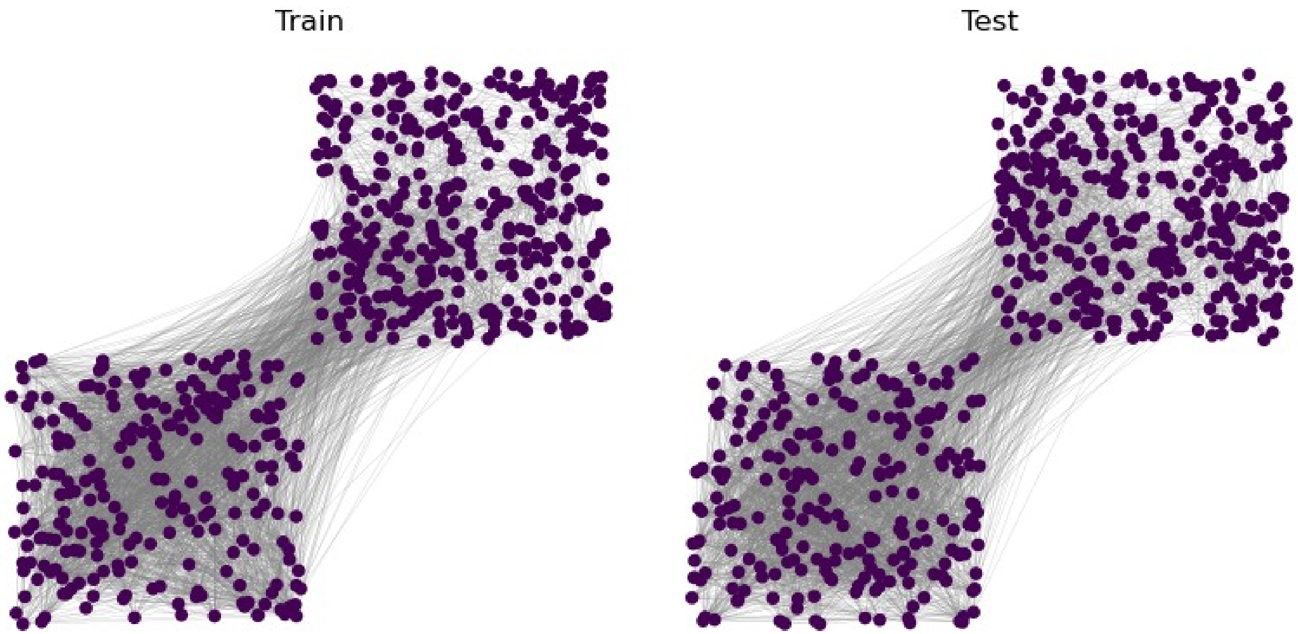
Nicolas Keriven[1], Alberto Bietti[2], Samuel Vaiter[3]

[1]CNRS, GIPSA-lab [2]NYU Center for Data Science [3]CNRS, IMB

nicolas.keriven@gipsa-lab.grenoble-inp.fr

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

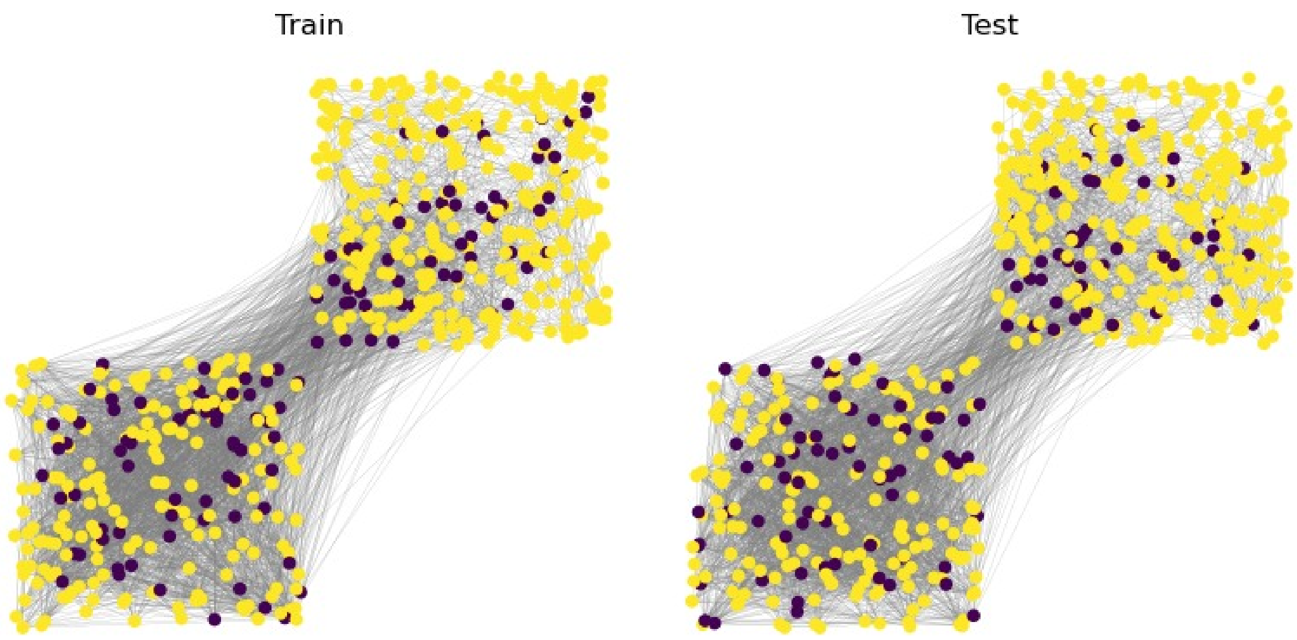


Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

Training GNNs

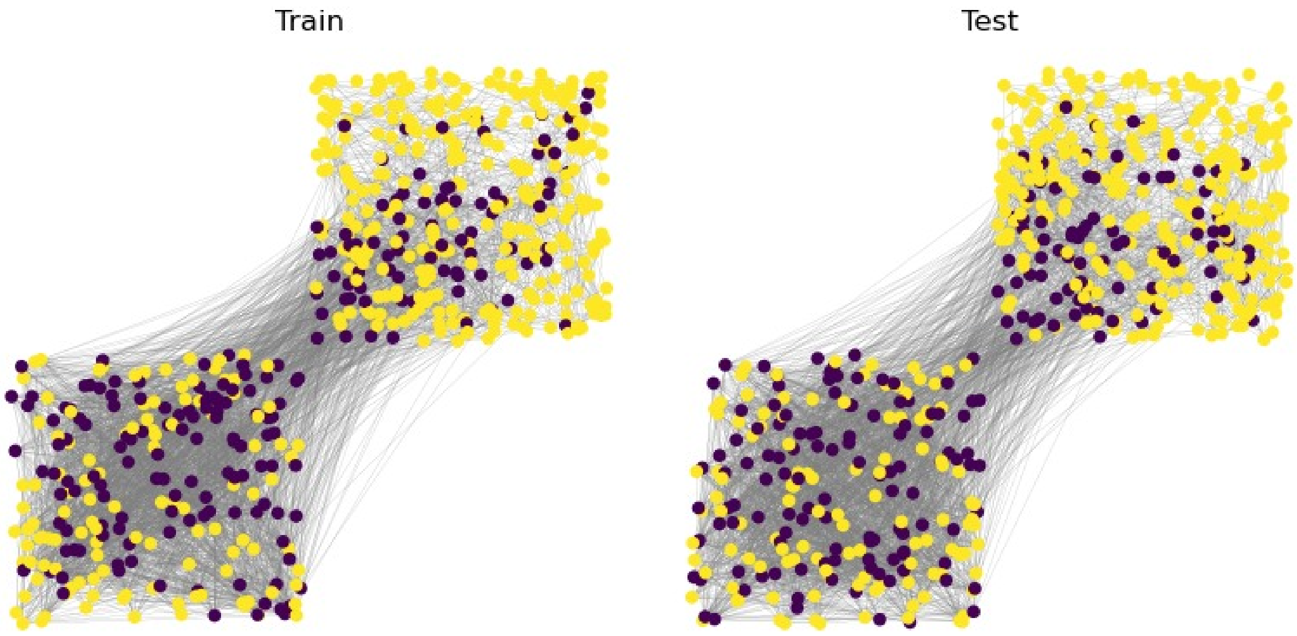
Goal: theoretical properties of GNNs on *large* graphs



*Output of a GNN with only graph structure as input, on two **different** graphs that "look the same"*

Training GNNs

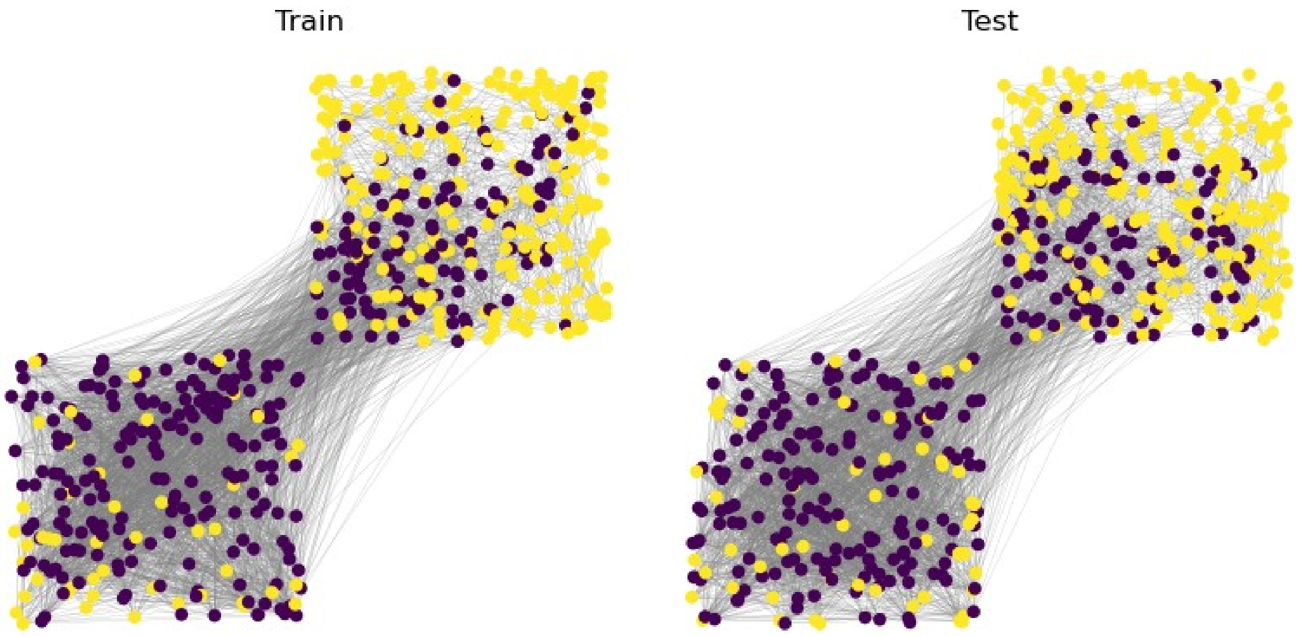
Goal: theoretical properties of GNNs on *large* graphs



Output of a GNN with only graph structure as input,
*on two **different** graphs that "look the same"*

Training GNNs

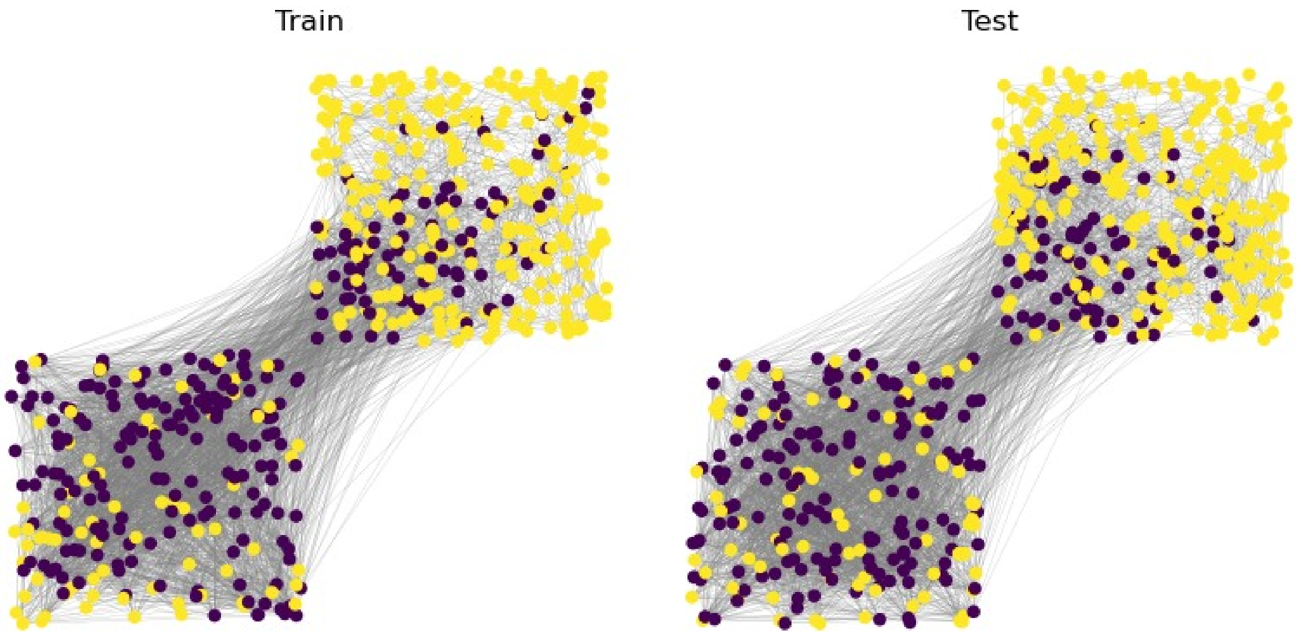
Goal: theoretical properties of GNNs on *large* graphs



Output of a GNN with only graph structure as input,
*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

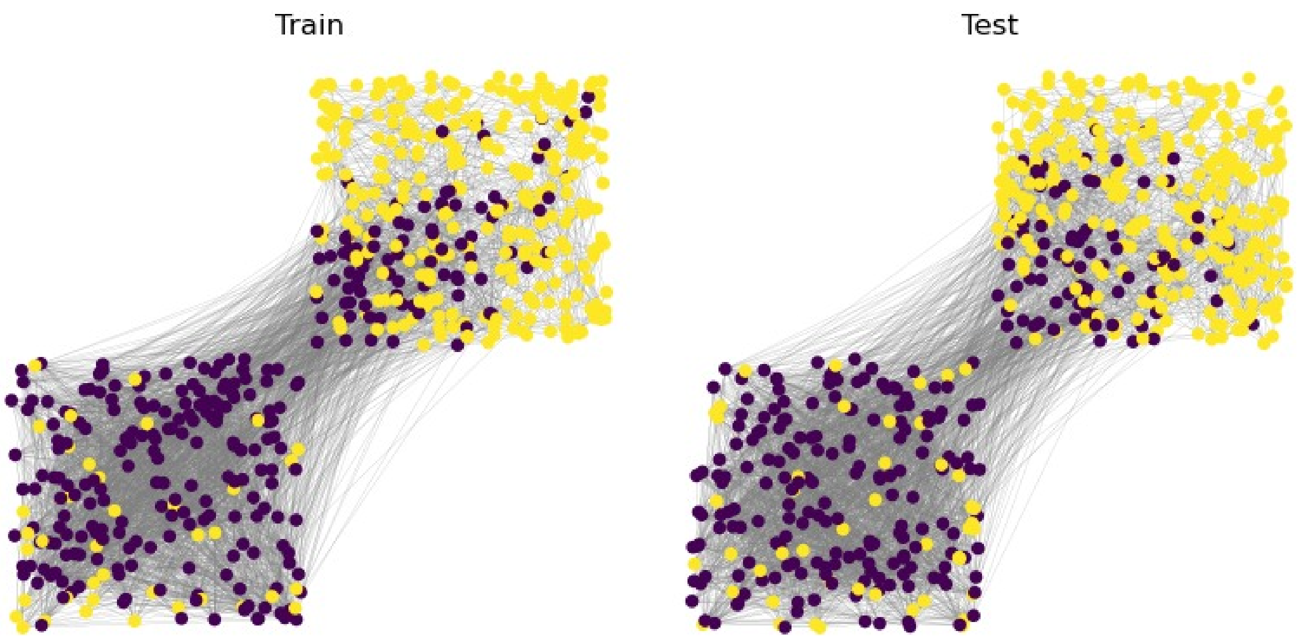


Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

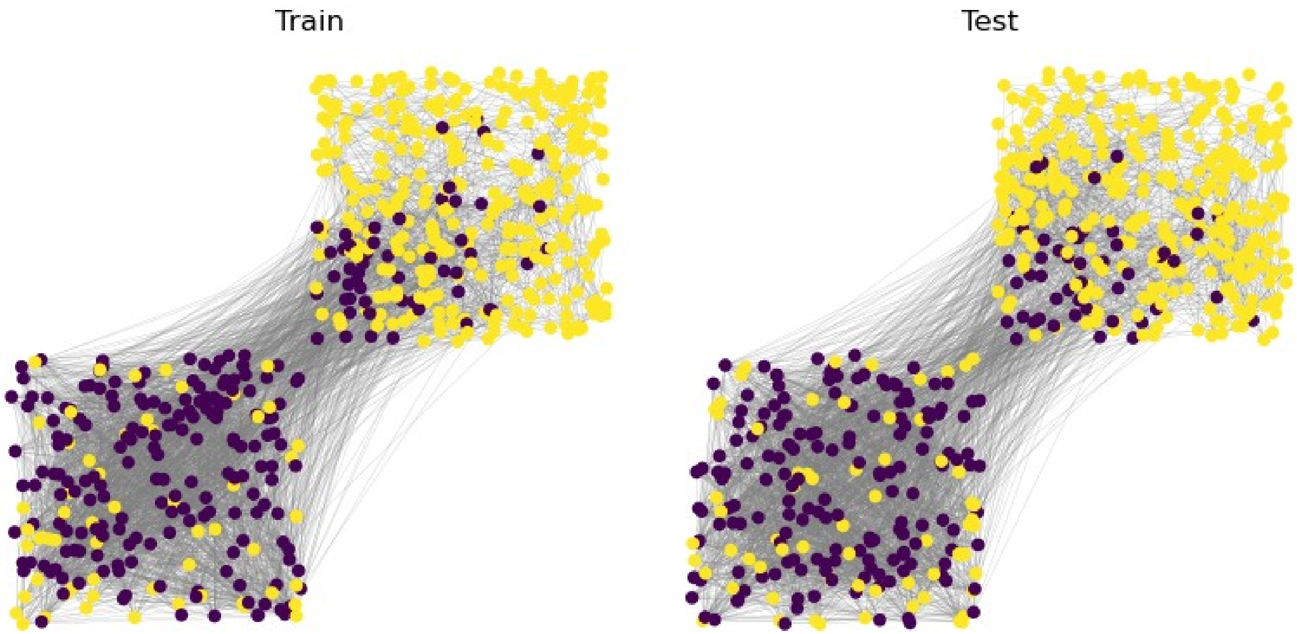


Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

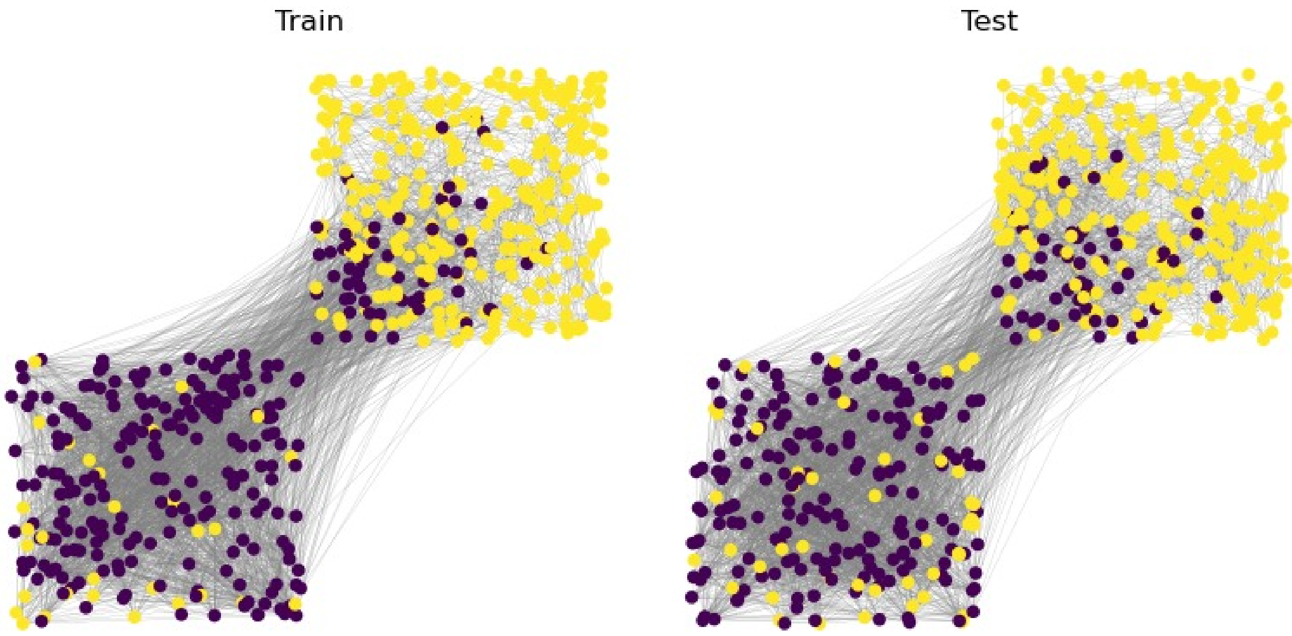


Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

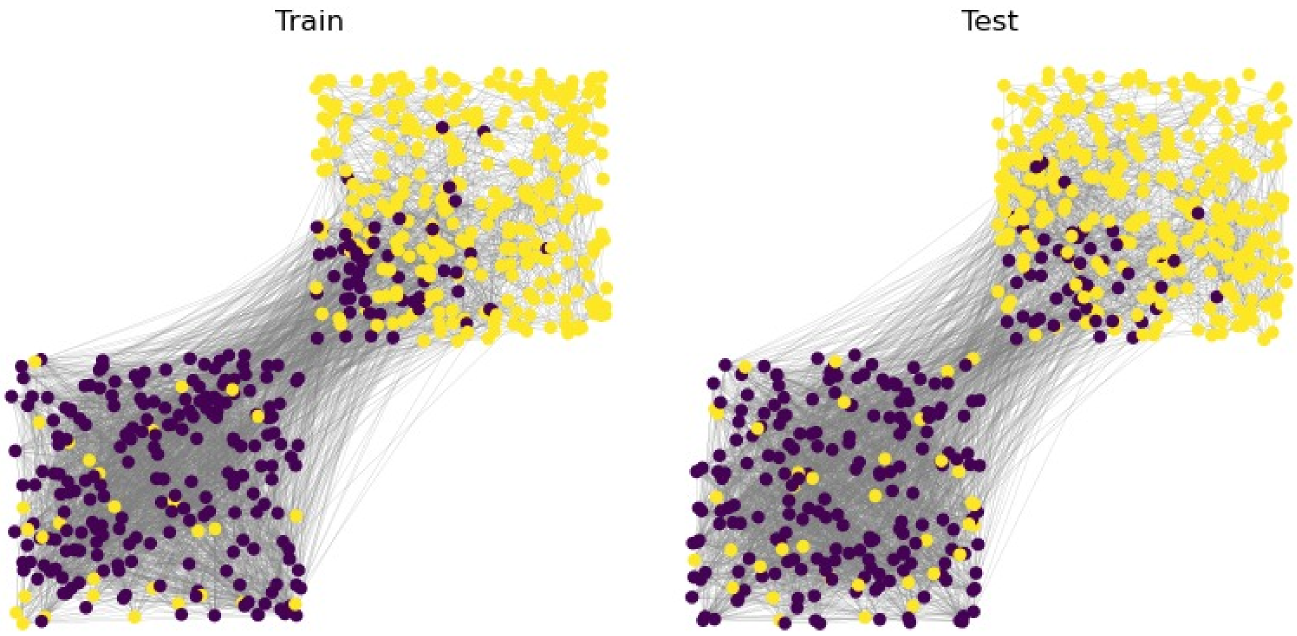


Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

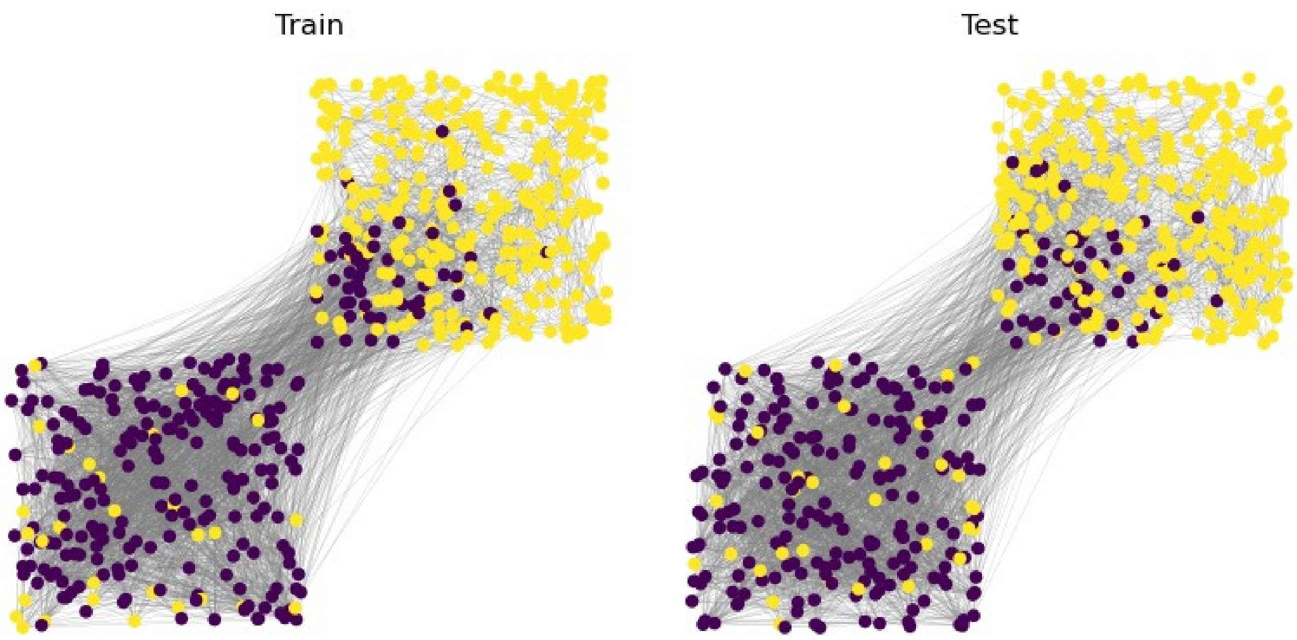


Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

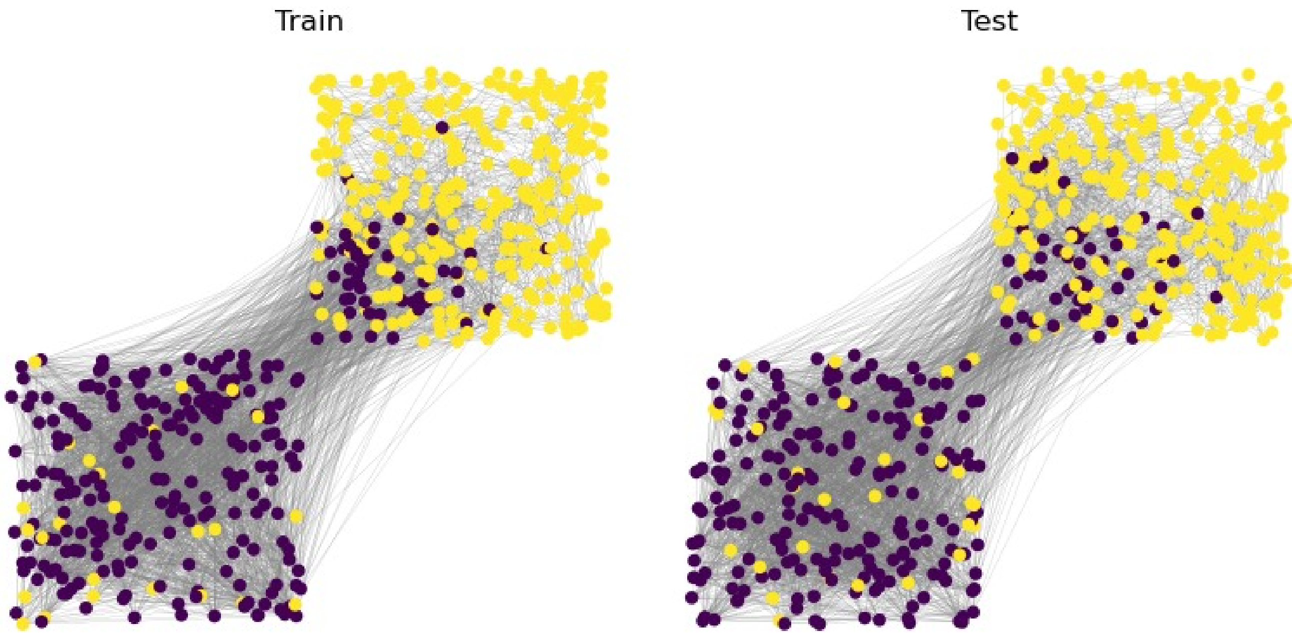


Output of a GNN with only graph structure as input,

on two ***different*** *graphs that "look the same"*

Training GNNs

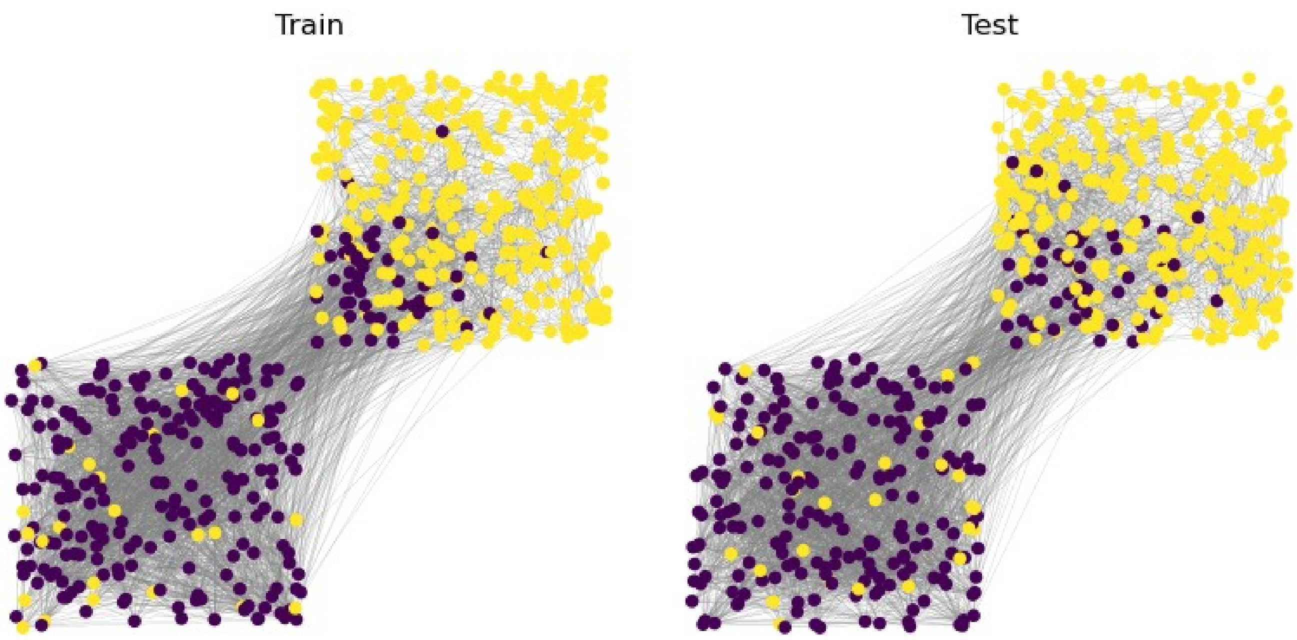
Goal: theoretical properties of GNNs on *large* graphs



Output of a GNN with only graph structure as input,
*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

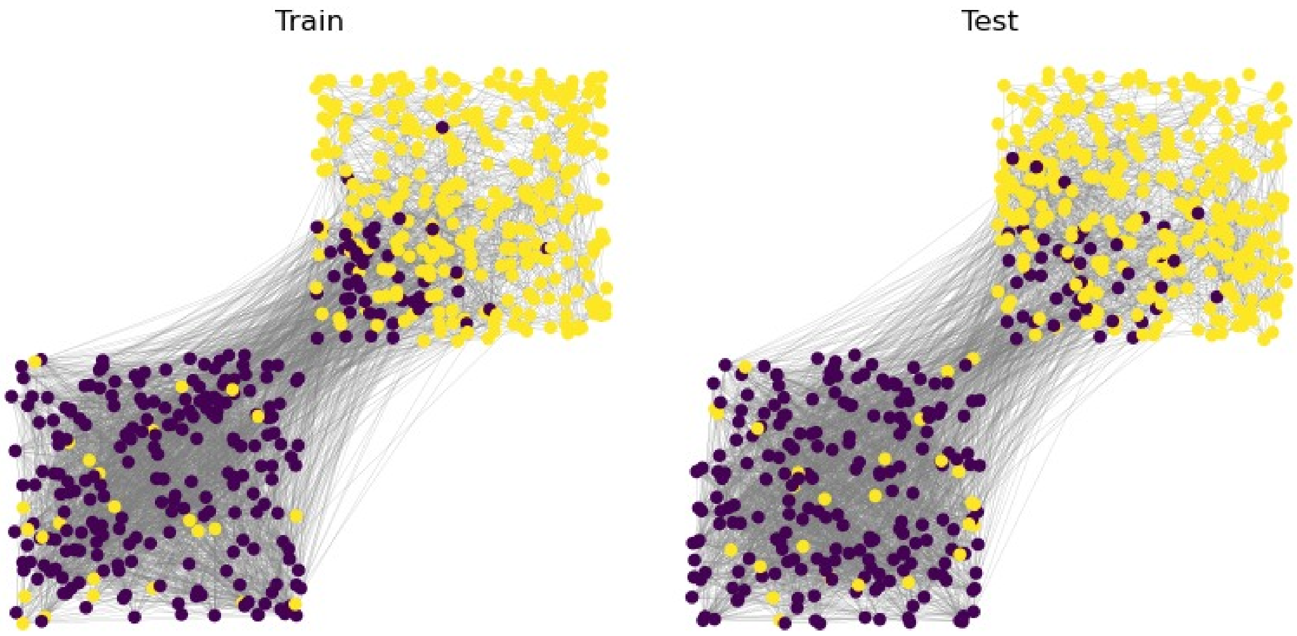


Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

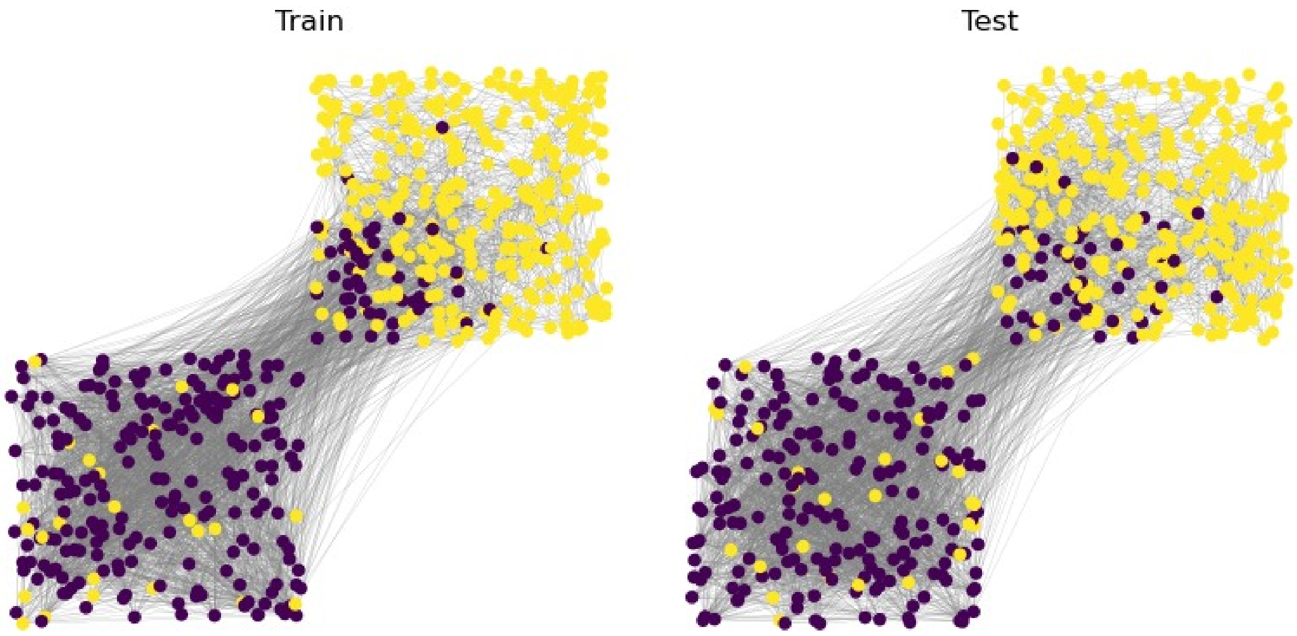


Output of a GNN with only graph structure as input,

*on two **different** graphs that “look the same”*

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs

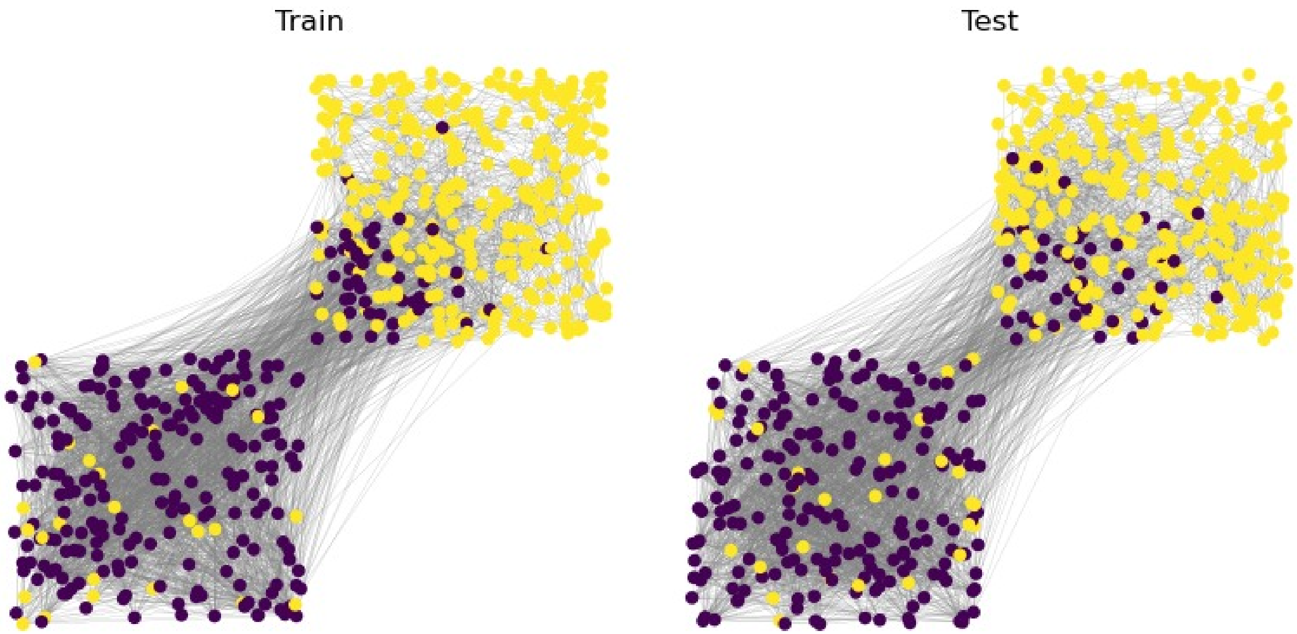


Output of a GNN with only graph structure as input,
*on two **different** graphs that "look the same"*

- When are two large graphs "similar" ? Does a GNN give "similar" outputs ?

Training GNNs

Goal: theoretical properties of GNNs on *large* graphs



Output of a GNN with only graph structure as input,

*on two **different** graphs that "look the same"*

- When are two large graphs "similar" ? Does a GNN give "similar" outputs ?
- In this talk: (some) properties of GNNs on large **random latent-position models** of graphs

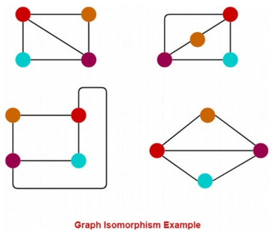
Background on GNNs

(Some) background in GNN theoretical analysis...

Background on GNNs

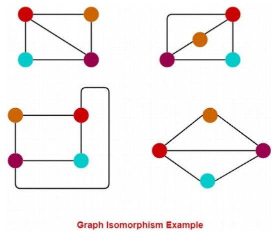
(Some) background in GNN theoretical analysis...

- As powerful as WL-test to distinguish **graph isomorphism** *[Xu et al. 2018]*
  - Can be made as powerful as k-WL-test… *[Maron et al. 2019]*



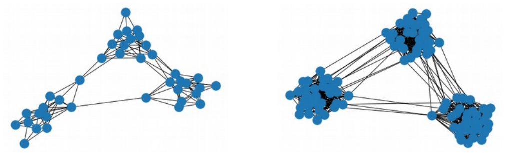
Background on GNNs

(Some) background in GNN theoretical analysis...



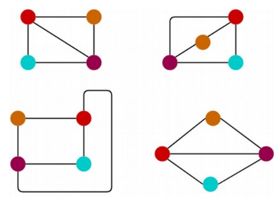
- As powerful as WL-test to distinguish **graph isomorphism** *[Xu et al. 2018]*
  - Can be made as powerful as k-WL-test… *[Maron et al. 2019]*

What about "large" graphs ?
(never isomorphic...)



Background on GNNs

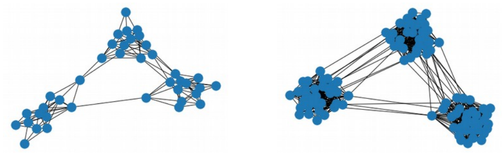
(Some) background in GNN theoretical analysis...



Graph Isomorphism Example

- As powerful as WL-test to distinguish **graph isomorphism** *[Xu et al. 2018]*
  - Can be made as powerful as k-WL-test… *[Maron et al. 2019]*

What about "large" graphs ?
(never isomorphic...)



- **Stability** to input change
  - **CNN** (translation-invariant): robustness to deformation *[Mallat, Bruna, Bietti, Mairal]*

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$

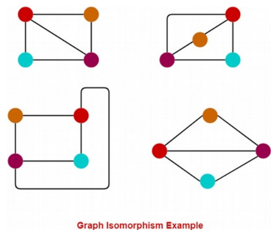
  - **GNN**: stability to **discrete graph metrics** *[Gama et al. 2019]*

$$\|\Phi_G(x) - \Phi_{G'}(x)\| \leq d(G, G')$$

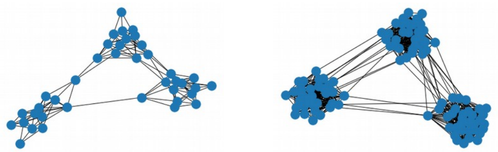
Background on GNNs

(Some) background in GNN theoretical analysis...

- As powerful as WL-test to distinguish **graph isomorphism** *[Xu et al. 2018]*
  - Can be made as powerful as k-WL-test… *[Maron et al. 2019]*



What about "large" graphs ? (never isomorphic...)



- **Stability** to input change
  - **CNN** (translation-invariant): robustness to deformation *[Mallat, Bruna, Bietti, Mairal]*

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$

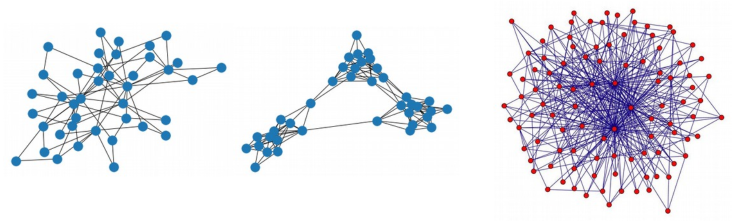
  - **GNN**: stability to **discrete graph metrics** *[Gama et al. 2019]*

$$\|\Phi_G(x) - \Phi_{G'}(x)\| \leq d(G, G')$$

What is a meaningful "deformation" for a graph ?

Random Graphs

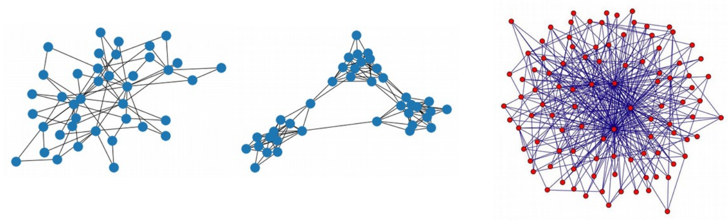
Long history of modeling large graphs with
random generative models



Bollobas. *Random Graphs (2001)*
Chung and Lu. *Complex Graphs and Networks (2004)*
Penrose. *Random Geometric Graphs (2008)*
Lovasz. *Large networks and graph limits (2012)*
Frieze and Karonski. *Introduction to random graphs (2016)*

Random Graphs

Long history of modeling large graphs with **random generative models**



Bollobas. *Random Graphs (2001)*
Chung and Lu. *Complex Graphs and Networks (2004)*
Penrose. *Random Geometric Graphs (2008)*
Lovasz. *Large networks and graph limits (2012)*
Frieze and Karonski. *Introduction to random graphs (2016)*

Latent position model

(W-random graphs, kernel random graphs...)

$$x_i \sim P \in \mathbb{R}^d \qquad z_i = f_0(x_i)$$

Latent variables *Node features*

$$a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$$

*Connectivity **kernel***

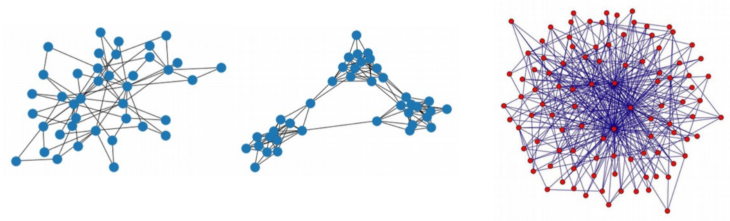
***Sparsity** level:*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$

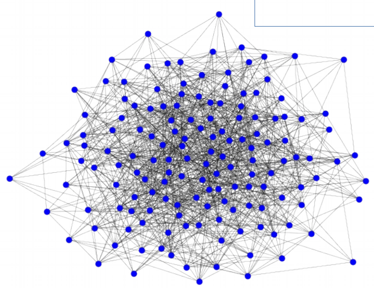
Random Graphs

Long history of modeling large graphs with **random generative models**



Bollobas. *Random Graphs (2001)*
Chung and Lu. *Complex Graphs and Networks (2004)*
Penrose. *Random Geometric Graphs (2008)*
Lovasz. *Large networks and graph limits (2012)*
Frieze and Karonski. *Introduction to random graphs (2016)*

- Erdos-Rényi



Latent position model
(W-random graphs, kernel random graphs...)

$$x_i \sim P \in \mathbb{R}^d \qquad z_i = f_0(x_i)$$

Latent variables *Node features*

$$a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$$

*Connectivity **kernel***

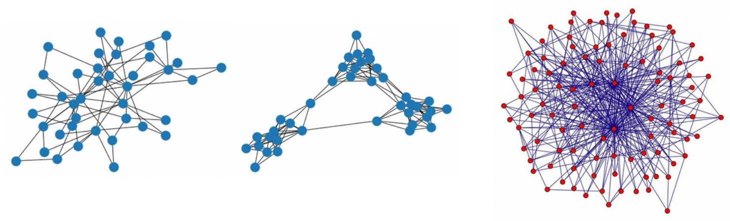
***Sparsity** level:*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$

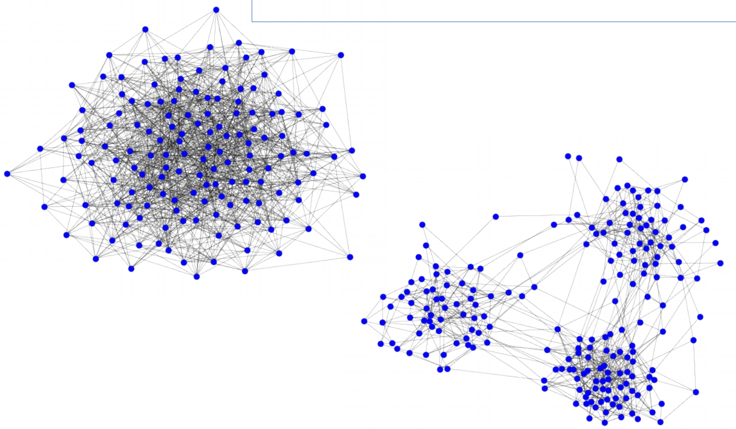
Random Graphs

Long history of modeling large graphs with
random generative models



Bollobas. *Random Graphs (2001)*
Chung and Lu. *Complex Graphs and Networks (2004)*
Penrose. *Random Geometric Graphs (2008)*
Lovasz. *Large networks and graph limits (2012)*
Frieze and Karonski. *Introduction to random graphs (2016)*

- Erdos-Rényi
- Stochastic Block Models



Latent position model
(W-random graphs, kernel random graphs...)

$$x_i \sim P \in \mathbb{R}^d \qquad z_i = f_0(x_i)$$

Latent variables *Node features*

$$a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$$

Connectivity ***kernel***

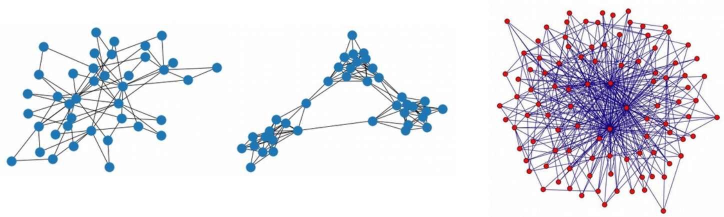
Sparsity *level:*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$

Random Graphs

Long history of modeling large graphs with **random generative models**



Bollobas. *Random Graphs (2001)*
Chung and Lu. *Complex Graphs and Networks (2004)*
Penrose. *Random Geometric Graphs (2008)*
Lovasz. *Large networks and graph limits (2012)*
Frieze and Karonski. *Introduction to random graphs (2016)*

- Erdos-Rényi
- Stochastic Block Models
- Gaussian kernel

Latent position model

(W-random graphs, kernel random graphs...)

$$x_i \sim P \in \mathbb{R}^d \qquad z_i = f_0(x_i)$$

Latent variables *Node features*

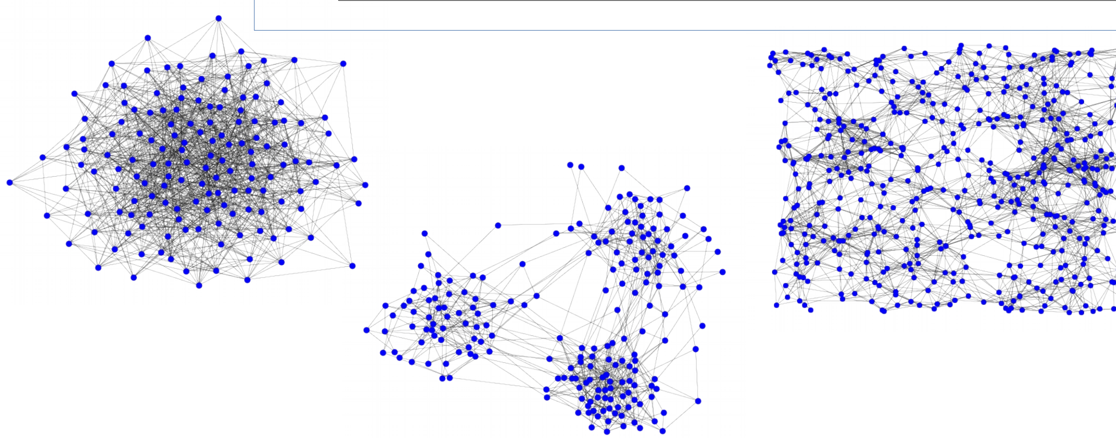
$$a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$$

*Connectivity **kernel***

***Sparsity** level:*

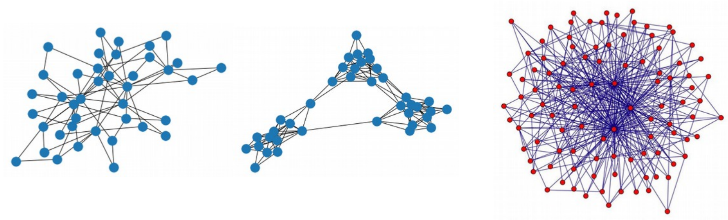
Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$



Random Graphs

Long history of modeling large graphs with **random generative models**



Bollobas. *Random Graphs (2001)*
Chung and Lu. *Complex Graphs and Networks (2004)*
Penrose. *Random Geometric Graphs (2008)*
Lovasz. *Large networks and graph limits (2012)*
Frieze and Karonski. *Introduction to random graphs (2016)*

- Erdos-Rényi
- Stochastic Block Models
- Gaussian kernel
- Epsilon-graphs
- Etc...

Latent position model

(W-random graphs, kernel random graphs...)

$$x_i \sim P \in \mathbb{R}^d \qquad z_i = f_0(x_i)$$

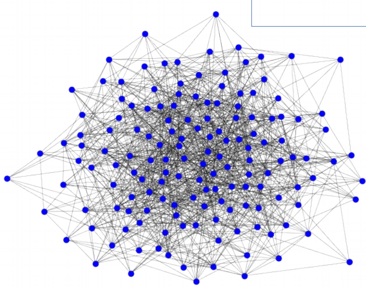
Latent variables *Node features*

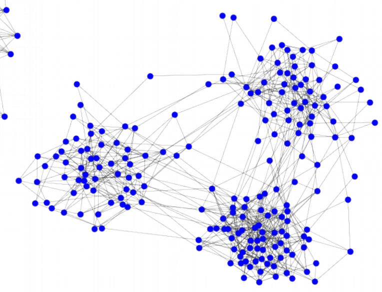
$$a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$$

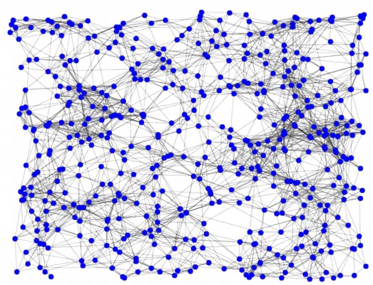
Connectivity ***kernel***

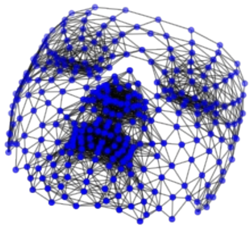
Sparsity *level:*

Dense	$\alpha_n \sim 1$	*Sparse*	$\alpha_n \sim 1/n$
Relatively sparse	$\alpha_n \sim (\log n)/n$		



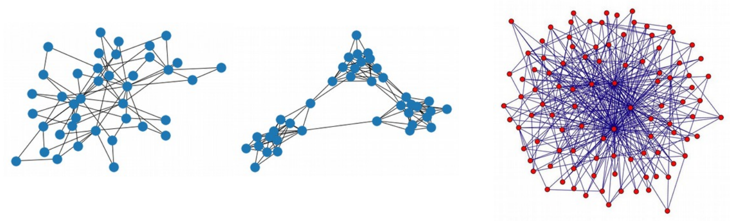






Random Graphs

Long history of modeling large graphs with **random generative models**



Bollobas. *Random Graphs (2001)*
Chung and Lu. *Complex Graphs and Networks (2004)*
Penrose. *Random Geometric Graphs (2008)*
Lovasz. *Large networks and graph limits (2012)*
Frieze and Karonski. *Introduction to random graphs (2016)*

- Erdos-Rényi
- Stochastic Block Models
- Gaussian kernel
- Epsilon-graphs
- Etc...

Latent position model

(W-random graphs, kernel random graphs...)

$$x_i \sim P \in \mathbb{R}^d \qquad z_i = f_0(x_i)$$

Latent variables — *Node features*

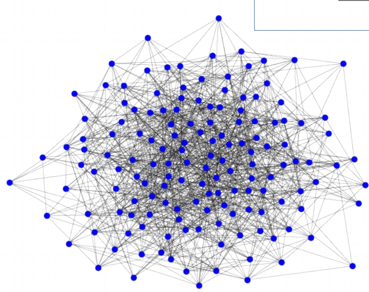
$$a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$$

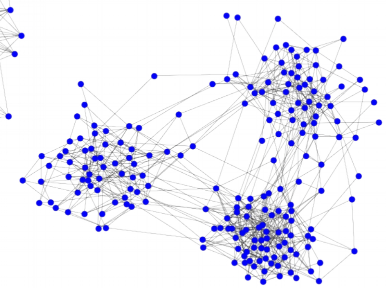
*Connectivity **kernel***

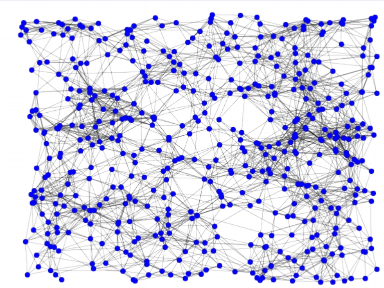
***Sparsity** level:*

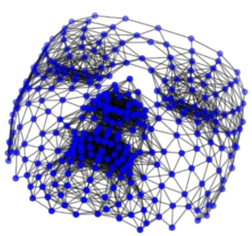
Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$









NB: all the above can be formulated with ***translation-invariant*** **kernels**

GNN convergence

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Polynomial graph filters
with normalized Laplacian $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

By default ***equivariant***, final pooling for ***invariant***

GNN convergence

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Polynomial graph filters with normalized Laplacian $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

By default ***equivariant***, final pooling for ***invariant***

Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Filters with normalized **Laplacian operator** $\mathcal{L} = \int \frac{W(\cdot, x)}{\sqrt{d_W(\cdot) d_W(x)}} f(x) dP(x)$

By default "***continuously***" ***equivariant***, final integration for ***invariant***

GNN convergence

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Polynomial graph filters with normalized Laplacian $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

By default ***equivariant***, final pooling for ***invariant***

Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Filters with normalized **Laplacian operator** $\mathcal{L} = \int \frac{W(\cdot, x)}{\sqrt{d_W(\cdot) d_W(x)}} f(x) dP(x)$

By default "***continuously***" ***equivariant***, final integration for ***invariant***

Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$, with probability $1 - n^{-r}$, the deviation between discrete and continuous GNN is at most $\boxed{O(dn^{-1/2} + (\alpha_n n)^{-1/2})}$

GNN convergence

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Polynomial graph filters with normalized Laplacian $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

By default ***equivariant***, final pooling for ***invariant***

Continuous Graph Neural Networks

- Propagate **function over latent space**

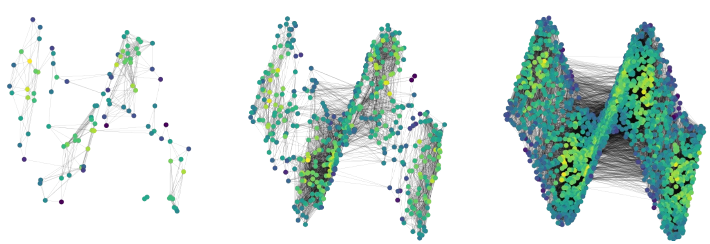
$$f_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

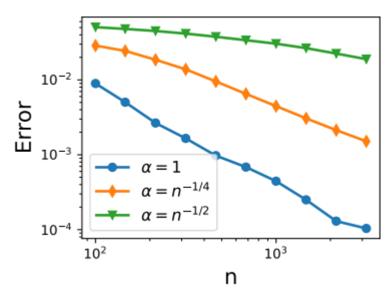
Filters with normalized **Laplacian operator** $\mathcal{L} = \int \frac{W(\cdot,x)}{\sqrt{d_W(\cdot) d_W(x)}} f(x) dP(x)$

By default "***continuously***" ***equivariant***, final integration for ***invariant***

Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$, with probability $1 - n^{-r}$, the deviation between discrete and continuous GNN is at most $\boxed{O(dn^{-1/2} + (\alpha_n n)^{-1/2})}$





GNN convergence

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Polynomial graph filters with normalized Laplacian $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

By default ***equivariant***, final pooling for ***invariant***

Continuous Graph Neural Networks

- Propagate **function over latent space**

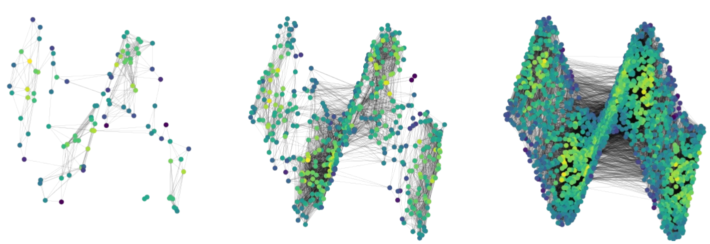
$$f_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

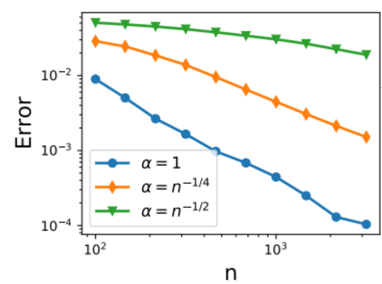
Filters with normalized **Laplacian operator** $\mathcal{L} = \int \frac{W(\cdot, x)}{\sqrt{d_W(\cdot) d_W(x)}} f(x) dP(x)$

By default "***continuously***" ***equivariant***, final integration for ***invariant***

Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$, with probability $1 - n^{-r}$, the deviation between discrete and continuous GNN is at most $O(dn^{-1/2} + (\alpha_n n)^{-1/2})$

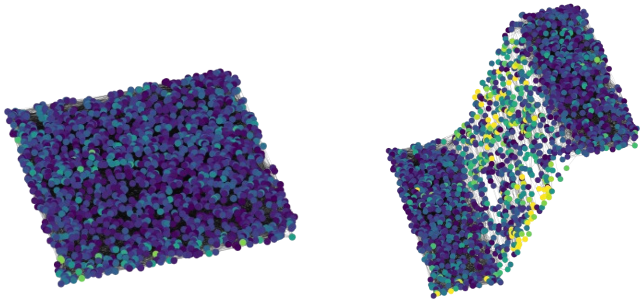




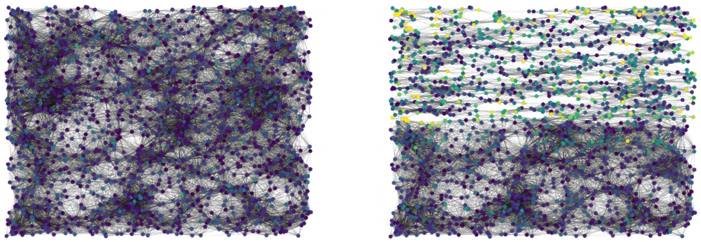
NB: Thanks to normalized Laplacian, the limit does ***not*** depend on α_n but the rate of convergence does...

Random Graphs: stability

Latent position models allow to define **intuitive geometric deformations**



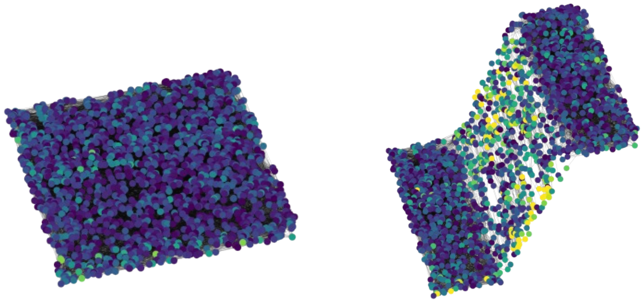
Deformation of distribution



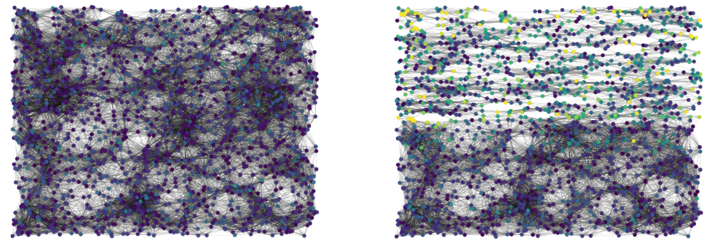
Deformation of kernel

Random Graphs: stability

Latent position models allow to define **intuitive geometric deformations**



Deformation of distribution



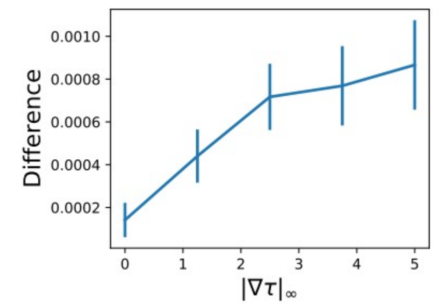
Deformation of kernel

Thm (Stability, simplified)

For **translation-invariant** kernels, if:

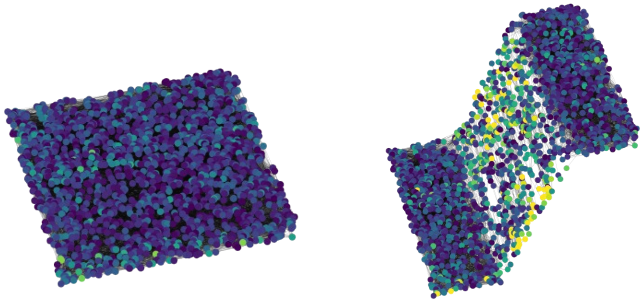
- W is replaced by $W(x-\tau(x), x'-\tau(x'))$, or
- P is replaced by $(Id-\tau)\sharp P$ (and f_0 is translated), or
- f_0 is replaced by $f_0 \circ (Id-\tau)$

Then, the deviation of c-GNN is bounded by $\|\nabla\tau\|_\infty$

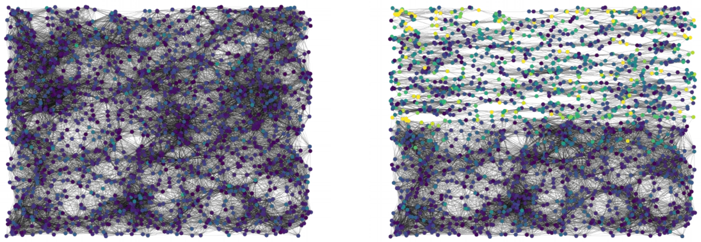


Random Graphs: stability

Latent position models allow to define **intuitive geometric deformations**



Deformation of distribution



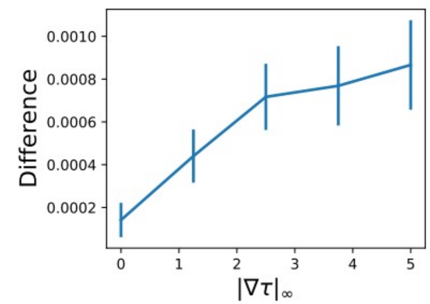
Deformation of kernel

Thm (Stability, simplified)

For **translation-invariant** kernels, if:

- W is replaced by $W(x - \tau(x), x' - \tau(x'))$, or
- P is replaced by $(Id - \tau)\sharp P$ (and f_0 is translated), or
- f_0 is replaced by $f_0 \circ (Id - \tau)$

Then, the deviation of c-GNN is bounded by $\|\nabla \tau\|_\infty$



Outlooks: approximation power, generalization, optimization, other RG models...