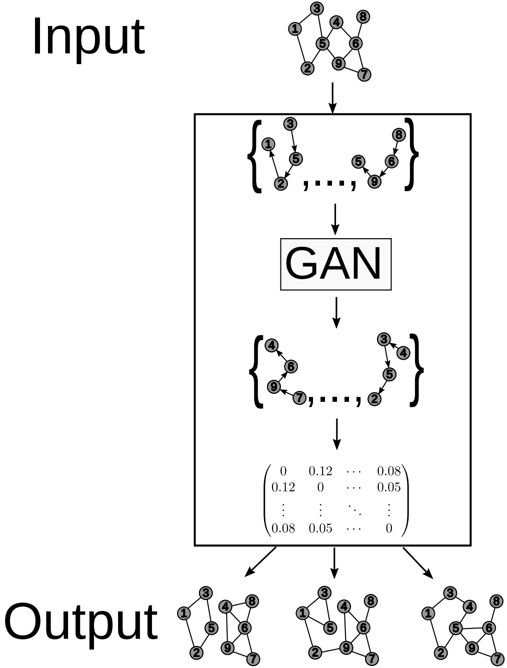
NetGAN without GAN: From Random Walks to Low-Rank Approximations

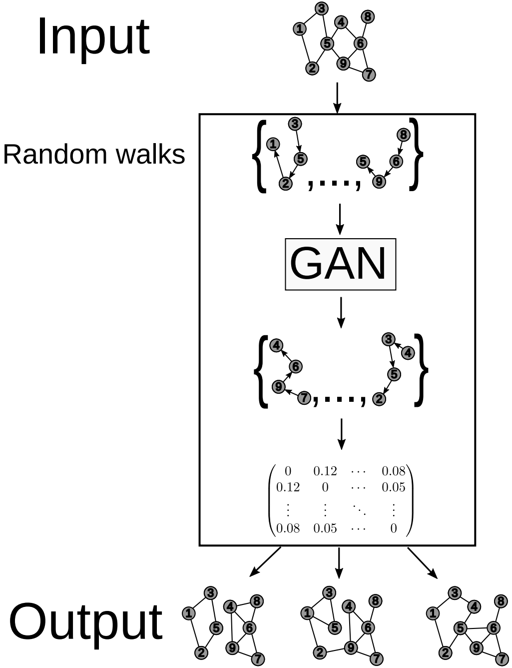
Luca Rendsburg, Holger Heidrich, Ulrike von Luxburg

ICML 2020

Input

Output

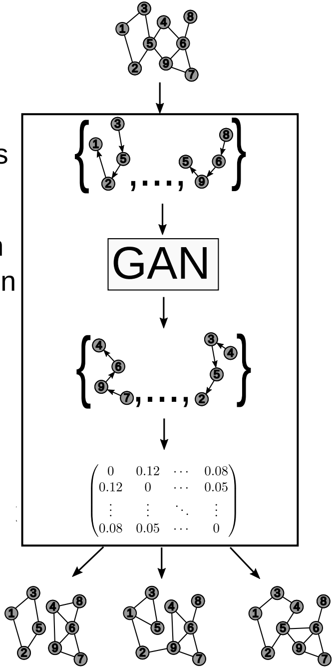

Input
GAN
0 0.12 ⋯ 0.08
0.12 0 ⋯ 0.05
0.08 0.05 ⋯ 0
Output


Input
Random walks
GAN
0 0.12 ··· 0.08
0.12 0 ··· 0.05
0.08 0.05 ··· 0
Output

Input

Random walks

Learn random walk distribution



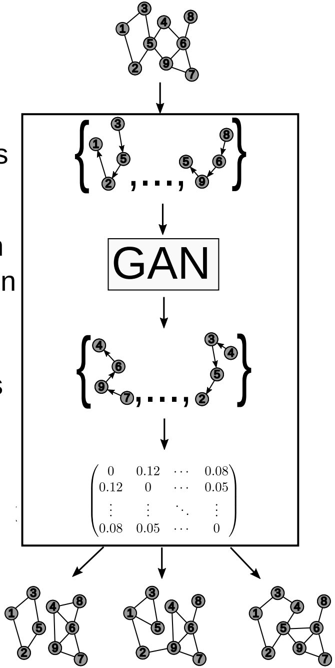
Output

Input

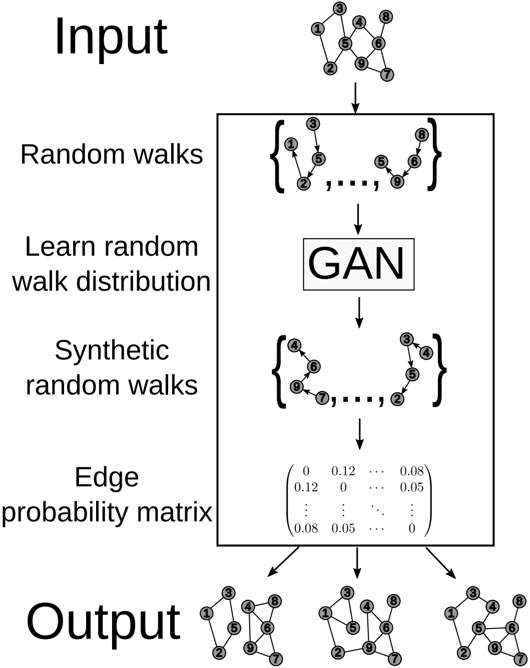
Random walks

Learn random walk distribution

Synthetic random walks



Output


Input
Random walks
Learn random
walk distribution
GAN
Synthetic
random walks
Edge
probability matrix
Output

Our contribution

Conceptual analysis

Our contribution

Conceptual analysis

- Inductive bias of NetGAN

Our contribution

Conceptual analysis

- Inductive bias of NetGAN
- Bypass sampling random walks

Our contribution

Conceptual analysis

- Inductive bias of NetGAN
- Bypass sampling random walks

Simplified version (no GAN, no sampling): "Cross-Entropy Low-rank Logits (CELL)"

Our contribution

Conceptual analysis

- Inductive bias of NetGAN
- Bypass sampling random walks

Simplified version (no GAN, no sampling): "Cross-Entropy Low-rank Logits (CELL)"

- Higher transparency

Our contribution

Conceptual analysis

- Inductive bias of NetGAN
- Bypass sampling random walks

Simplified version (no GAN, no sampling): “Cross-Entropy Low-rank Logits (CELL)”

- Higher transparency
- Comparable generalization performance

Our contribution

Conceptual analysis

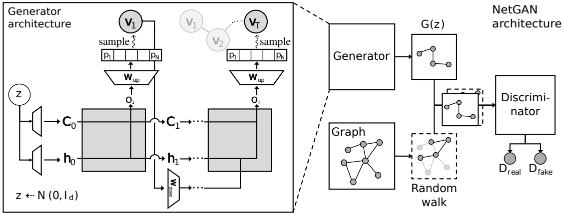
- Inductive bias of NetGAN
- Bypass sampling random walks

Simplified version (no GAN, no sampling): "Cross-Entropy Low-rank Logits (CELL)"

- Higher transparency
- Comparable generalization performance
- Huge speedup

NetGAN: learning step

- **Goal:** learn random walk distribution

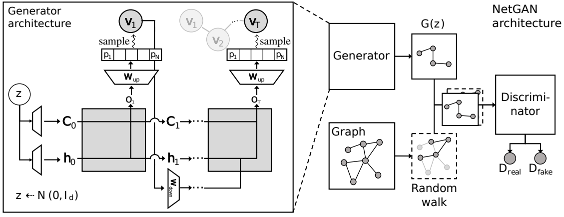


(a) Generator architecture (b) NetGAN architecture

Figure from Bojchevski et al. [2018]

NetGAN: learning step

- **Goal:** learn random walk distribution
- **Training set:** unbiased random walks $\{(v_0^{(i)}, \ldots, v_T^{(i)})\}_i$ of length T over input graph

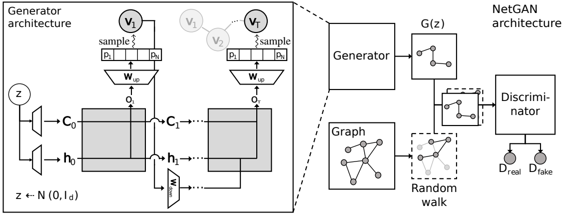


(a) Generator architecture (b) NetGAN architecture

Figure from Bojchevski et al. [2018]

NetGAN: learning step

- **Goal:** learn random walk distribution
- **Training set:** unbiased random walks $\{(v_0^{(i)}, \ldots, v_T^{(i)})\}_i$ of length T over input graph
- **Generator:** generate sequences $\{(w_0^{(i)}, \ldots, w_T^{(i)})\}_i$ of "synthetic" random walks

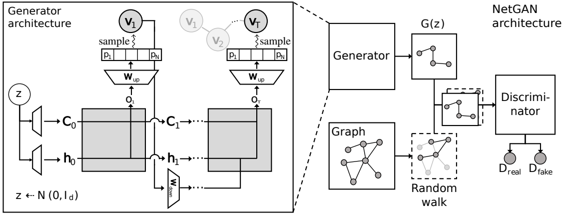


(a) Generator architecture
(b) NetGAN architecture

Figure from Bojchevski et al. [2018]

NetGAN: learning step

- **Goal:** learn random walk distribution
- **Training set:** unbiased random walks $\{(v_0^{(i)}, \ldots, v_T^{(i)})\}_i$ of length T over input graph
- **Generator:** generate sequences $\{(w_0^{(i)}, \ldots, w_T^{(i)})\}_i$ of "synthetic" random walks
- **Discriminator:** distinguish synthetic from real random walks

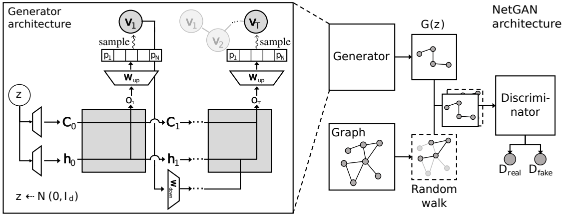


(a) Generator architecture (b) NetGAN architecture

Figure from Bojchevski et al. [2018]

NetGAN: learning step

- **Goal:** learn random walk distribution
- **Training set:** unbiased random walks $\{(v_0^{(i)}, \ldots, v_T^{(i)})\}_i$ of length T over input graph
- **Generator:** generate sequences $\{(w_0^{(i)}, \ldots, w_T^{(i)})\}_i$ of "synthetic" random walks
- **Discriminator:** distinguish synthetic from real random walks
- **Architecture:** LSTMs with Wasserstein loss



(a) Generator architecture (b) NetGAN architecture

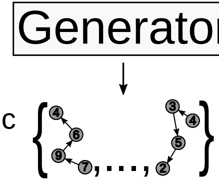
Figure from Bojchevski et al. [2018]

NetGAN: reconstruction step

Generator

NetGAN: reconstruction step

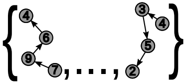
Generate *many* synthetic random walks



NetGAN: reconstruction step

$\boxed{\text{Generator}}$

Generate *many* synthetic random walks



Count transitions in *score matrix* S

$$\begin{pmatrix} 0 & 360 & \cdots & 212 \\ 307 & 0 & \cdots & 150 \\ \vdots & \vdots & \ddots & \vdots \\ 240 & 143 & \cdots & 0 \end{pmatrix}$$

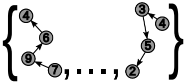
$S_{i,j} = \#\text{ transitions from } i \text{ to } j$

NetGAN: reconstruction step

$\boxed{\text{Generator}}$

↓

Generate *many* synthetic random walks



↓

Count transitions in *score matrix* S

$$\begin{pmatrix} 0 & 360 & \cdots & 212 \\ 307 & 0 & \cdots & 150 \\ \vdots & \vdots & \ddots & \vdots \\ 240 & 143 & \cdots & 0 \end{pmatrix}$$

$$S_{i,j} = \# \text{ transitions from } i \text{ to } j$$

↓

Transform S into edge probability matrix $A^\dagger$

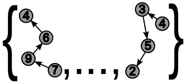
$$\begin{pmatrix} 0 & 0.12 & \cdots & 0.08 \\ 0.12 & 0 & \cdots & 0.05 \\ \vdots & \vdots & \ddots & \vdots \\ 0.08 & 0.05 & \cdots & 0 \end{pmatrix}$$

$$A^\dagger_{i,j} = \frac{\max\{S_{i,j}, S_{j,i}\}}{\sum_{k,l} \max\{S_{k,l}, S_{l,k}\}}$$

NetGAN: reconstruction step

$\boxed{\text{Generator}}$

Generate *many* synthetic random walks



Count transitions in *score matrix* S

$$\begin{pmatrix} 0 & 360 & \cdots & 212 \\ 307 & 0 & \cdots & 150 \\ \vdots & \vdots & \ddots & \vdots \\ 240 & 143 & \cdots & 0 \end{pmatrix}$$

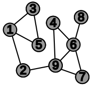
$$S_{i,j} = \# \text{ transitions from } i \text{ to } j$$

Transform S into edge probability matrix $A^\dagger$

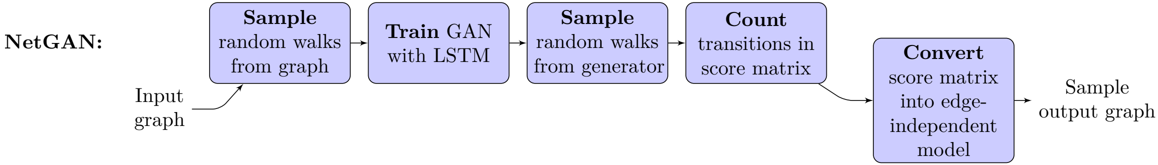
$$\begin{pmatrix} 0 & 0.12 & \cdots & 0.08 \\ 0.12 & 0 & \cdots & 0.05 \\ \vdots & \vdots & \ddots & \vdots \\ 0.08 & 0.05 & \cdots & 0 \end{pmatrix}$$

$$A^\dagger_{i,j} = \frac{\max\{S_{i,j}, S_{j,i}\}}{\sum_{k,l} \max\{S_{k,l}, S_{l,k}\}}$$

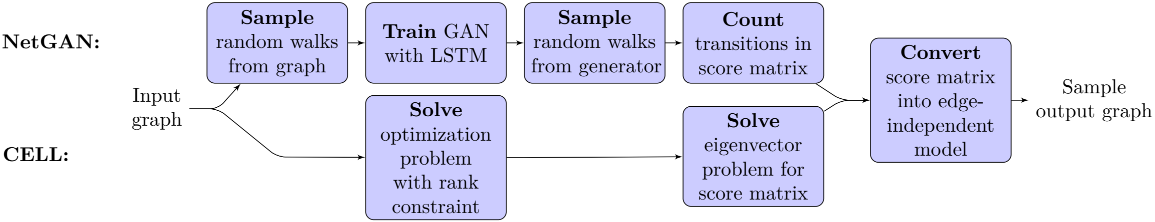
Sample edges without replacement



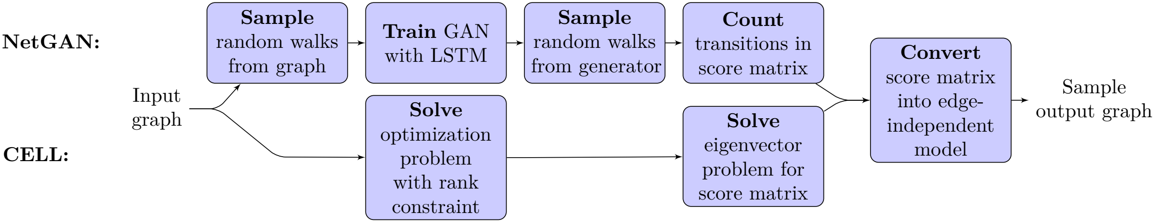
Overview of simplifications



Overview of simplifications

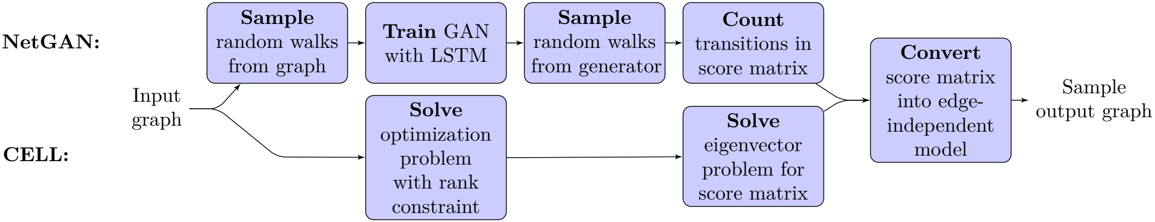


Overview of simplifications



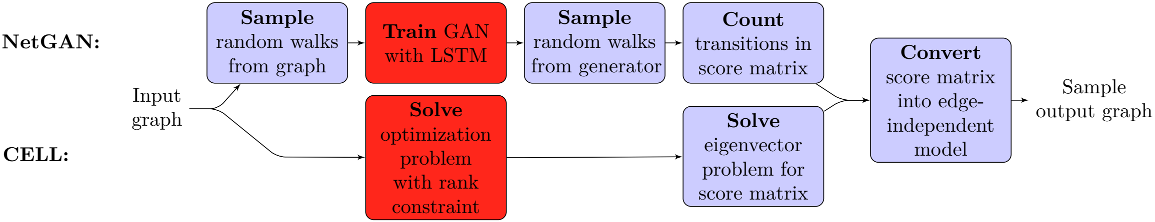
1. Replace GAN with rank-constrained optimization problem

Overview of simplifications



1. Replace GAN with rank-constrained optimization problem
2. Bypass random walk sampling

Replacing the GAN (1)



1. Replace GAN with rank-constrained optimization problem
2. Bypass random walk sampling

Replacing the GAN (2)

What causes the generalization of NetGAN?

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

The GAN?

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

The GAN?

✗ Perfectly learning random walk distribution simply memorizes graph

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

The GAN?

✗ Perfectly learning random walk distribution simply memorizes graph

The LSTM?

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

The GAN?

✗ Perfectly learning random walk distribution simply memorizes graph

The LSTM?

✗ No long-term dependencies in random walks

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

The GAN?

✗ Perfectly learning random walk distribution simply memorizes graph

The LSTM?

✗ No long-term dependencies in random walks

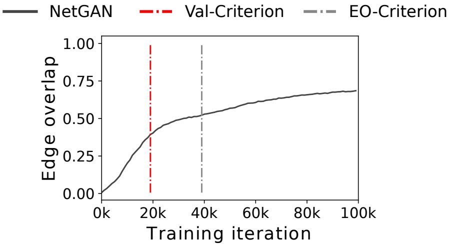


Figure from Bojchevski et al. [2018]

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

The GAN?

✗ Perfectly learning random walk distribution simply memorizes graph

The LSTM?

✗ No long-term dependencies in random walks

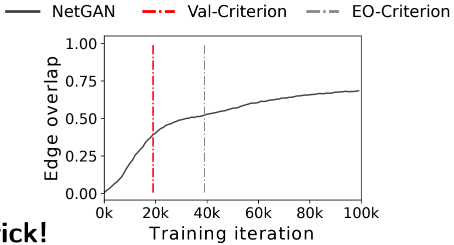


Figure from Bojchevski et al. [2018]

Computational trick!

Replacing the GAN (2)

What causes the generalization of NetGAN?

The random walks?

✗ Random walk distribution determines graph

The GAN?

✗ Perfectly learning random walk distribution simply memorizes graph

The LSTM?

✗ No long-term dependencies in random walks

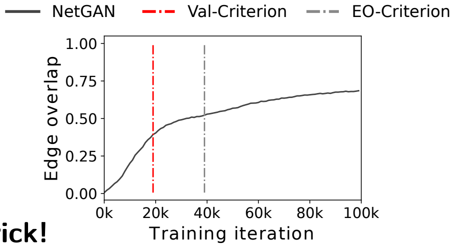


Figure from Bojchevski et al. [2018]

Computational trick!

✓ Low-dimensional bottleneck

Replacing the GAN (3)

- Number of nodes N, set of random walks $\mathcal{R}$, low rank H, row-wise softmax σ_{rows}

Replacing the GAN (3)

- Number of nodes N, set of random walks $\mathcal{R}$, low rank H, row-wise softmax σ_{rows}
- Generator learns random walk distribution by **learning random walk transition matrix** from parametric family

$$\mathcal{P} = \{\sigma_{\text{rows}}(W) \in \mathbb{R}^{N \times N} : W \in \mathbb{R}^{N \times N}, \operatorname{rank}(W) \leqslant H\}\,.$$

Replacing the GAN (3)

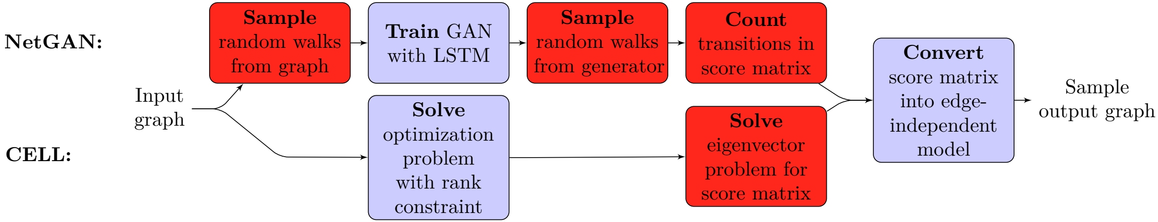
- Number of nodes N, set of random walks $\mathcal{R}$, low rank H, row-wise softmax σ_{rows}
- Generator learns random walk distribution by **learning random walk transition matrix** from parametric family

$$\mathcal{P} = \left\{\sigma_{\text{rows}}(W) \in \mathbb{R}^{N\times N} : W \in \mathbb{R}^{N\times N}, \operatorname{rank}(W) \leqslant H\right\}.$$

- Do this directly with **maximum likelihood estimation**

$$\boxed{\begin{aligned} &\min_{W\in\mathbb{R}^{N\times N}} -\sum_{(i,j)\in\mathcal{R}} \log \sigma_{\text{rows}}(W)_{i,j} \,, \\ &\text{s.t.} \quad \operatorname{rank}(W) \leqslant H \,. \end{aligned}}$$

Bypassing sampling (1)



1. Replace GAN with rank-constrained optimization problem
2. Bypass random walk sampling

Bypassing sampling (2)

- NetGAN samples *many* random walks from input graph & from generator

Bypassing sampling (2)

- NetGAN samples *many* random walks from input graph & from generator
- Example: On CORA-ML (2,810/ 7,981), sees every edge $\sim 14,000$ times on average

Bypassing sampling (2)

- NetGAN samples *many* random walks from input graph & from generator
- Example: On CORA-ML (2,810/ 7,981), sees every edge $\sim 14,000$ times on average
- Count random walk transitions in *score matrix* $S \in \mathbb{R}^{N \times N}$

Bypassing sampling (2)

- NetGAN samples *many* random walks from input graph & from generator
- Example: On CORA-ML (2,810/ 7,981), sees every edge $\sim 14,000$ times on average
- Count random walk transitions in *score matrix* $S \in \mathbb{R}^{N \times N}$
- Length of random walk T, amount of random walks n. Normalized score matrix converges

$$\frac{S}{nT} \xrightarrow[n,T\to\infty]{a.\,s.} \operatorname{diag}(\pi)P\,,$$

where P is corresponding random walk transition matrix and π its stationary distribution.

Bypassing sampling (2)

- NetGAN samples *many* random walks from input graph & from generator
- Example: On CORA-ML (2,810/ 7,981), sees every edge $\sim 14,000$ times on average
- Count random walk transitions in *score matrix* $S \in \mathbb{R}^{N\times N}$
- Length of random walk T, amount of random walks n. Normalized score matrix converges

$$\frac{S}{nT} \xrightarrow[n,T\to\infty]{a.\,s.} \mathrm{diag}(\pi)P\,,$$

where P is corresponding random walk transition matrix and π its stationary distribution.

Replace normalized score matrix with its limit

Bypassing sampling (3)

1. Learning step (with adjacency matrix A)

$$\boxed{\begin{aligned} &\min_{W \in \mathbb{R}^{N \times N}} -\sum_{(i,j) \in \mathcal{R}} \log \sigma_{\text{rows}}(W)_{i,j} \,, \\ &\text{s.t.} \quad \operatorname{rank}(W) \leqslant H \end{aligned}} \longrightarrow \boxed{\begin{aligned} &\min_{W \in \mathbb{R}^{N \times N}} -\sum_{k,l} A_{k,l} \log \sigma_{\text{rows}}(W)_{k,l} \,, \\ &\text{s.t.} \quad \operatorname{rank}(W) \leqslant H \end{aligned}}$$

Bypassing sampling (3)

1. Learning step (with adjacency matrix A)

$$\boxed{\begin{aligned} &\min_{W \in \mathbb{R}^{N\times N}} -\sum_{(i,j)\in\mathcal{R}} \log \sigma_{\text{rows}}(W)_{i,j}\,, \\ &\text{s.t.} \quad \operatorname{rank}(W) \leqslant H \end{aligned}} \rightarrow \boxed{\begin{aligned} &\min_{W \in \mathbb{R}^{N\times N}} -\sum_{k,l} A_{k,l} \log \sigma_{\text{rows}}(W)_{k,l}\,, \\ &\text{s.t.} \quad \operatorname{rank}(W) \leqslant H \end{aligned}}$$

2. Reconstruction step (with synthetic transition matrix P^*)

$$\boxed{\text{Compute } S \text{ by counting transitions of synthetic random walks}} \rightarrow \boxed{S := \operatorname{diag}(\pi^*)P^*}$$

Bypassing sampling (3)

1. Learning step (with adjacency matrix A)

$$\boxed{\begin{aligned}&\min_{W\in\mathbb{R}^{N\times N}} -\sum_{(i,j)\in\mathcal{R}} \log \sigma_{\text{rows}}(W)_{i,j}\,,\\ &\text{s.t.}\quad \operatorname{rank}(W)\leqslant H\end{aligned}} \rightarrow \boxed{\begin{aligned}&\min_{W\in\mathbb{R}^{N\times N}} -\sum_{k,l} A_{k,l} \log \sigma_{\text{rows}}(W)_{k,l}\,,\\ &\text{s.t.}\quad \operatorname{rank}(W)\leqslant H\end{aligned}}$$

2. Reconstruction step (with synthetic transition matrix P^*)

$$\boxed{\text{Compute } S \text{ by counting transitions of synthetic random walks}} \rightarrow \boxed{S := \operatorname{diag}(\pi^*)P^*}$$

Bypass sampling in both steps

Experiments (1)

Does CELL generate the same type of graphs as NetGAN?

Experiments (1)

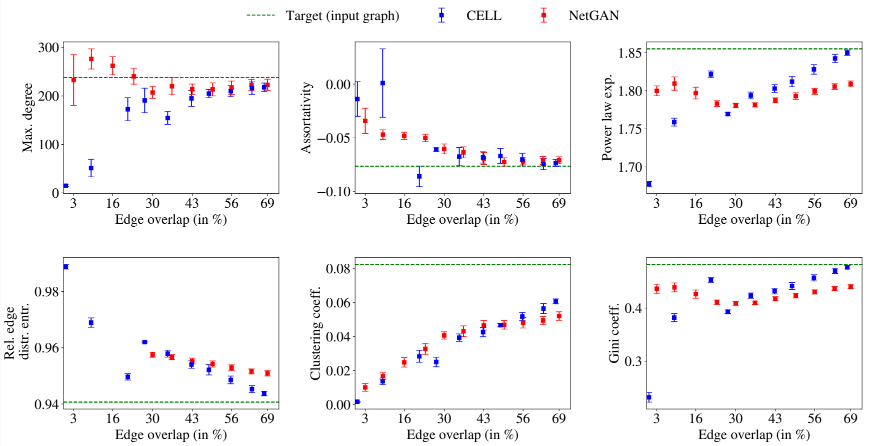
Does CELL generate the same type of graphs as NetGAN?

Yes!

Experiments (1)

Does CELL generate the same type of graphs as NetGAN? Yes!

Graph: CORA-ML citation network (2,810/ 7,981)



Experiments (2)

CELL is significantly faster

Experiments (2)

CELL is significantly faster

Table: Training time (in seconds) for NetGAN and CELL on a variety of networks. NetGAN requires a GPU, while CELL runs on a CPU.

Data set (Nodes/ Edges)		NetGAN	CELL
CORA-ML	(2,810/ 7,981)	7,478	21
Citeseer	(2,110/ 3,668)	4,654	10
PolBlogs	(1,222/ 16,779)	55,276	15
RT-GOP	(4,687/ 5,529)	14,800	23
Web-EDU	(3,031/ 6,474)	11,000	16

Conclusion

NetGAN

Conclusion

NetGAN

- Successful graph generative model

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

- Uncover inductive bias: low-rank assumption

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

- Uncover inductive bias: low-rank assumption
- Starting point to better understand inductive bias

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

- Uncover inductive bias: low-rank assumption
- Starting point to better understand inductive bias
- Bypass sampling by using a limit argument

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

- Uncover inductive bias: low-rank assumption
- Starting point to better understand inductive bias
- Bypass sampling by using a limit argument
- Propose simplified algorithm with comparable generalization performance

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

- Uncover inductive bias: low-rank assumption
- Starting point to better understand inductive bias
- Bypass sampling by using a limit argument
- Propose simplified algorithm with comparable generalization performance

Future work

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

- Uncover inductive bias: low-rank assumption
- Starting point to better understand inductive bias
- Bypass sampling by using a limit argument
- Propose simplified algorithm with comparable generalization performance

Future work

- Explain contribution of low-rank assumption

Conclusion

NetGAN

- Successful graph generative model
- Complicated and not transparent

Our contribution: conceptual analysis

- Uncover inductive bias: low-rank assumption
- Starting point to better understand inductive bias
- Bypass sampling by using a limit argument
- Propose simplified algorithm with comparable generalization performance

Future work

- Explain contribution of low-rank assumption