Machine Learning for Graphs and Sequential Data

Graphs – Generative Models

Lecturer: Prof. Dr. Stephan Günnemann

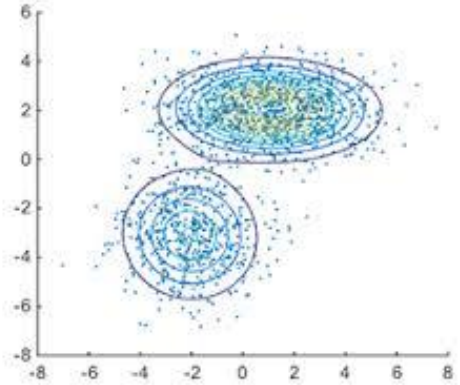
www.daml.in.tum.de

Summer Term 2020

Data Analytics and
Machine Learning

Recap: Generative Models

- Generative model: statistical model to describe the data distribution
  - for unsupervised learning, e.g., $p(\boldsymbol{x})$
  - can also be used for generating data (hence the name)

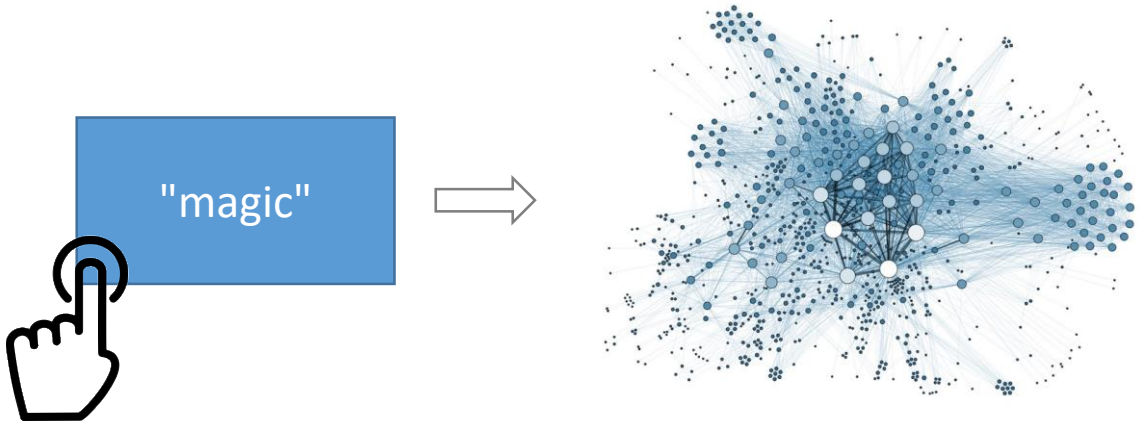


- Typical example: Gaussian Mixture Models (GMMs)

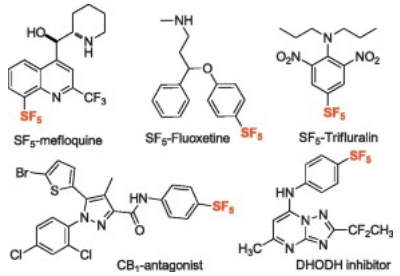
- Generative process of a GMM:
  1. Specify prior probability of each cluster k is $\pi_k > 0,\ \sum_k \pi_k = 1$
  2. For each sample i (you want to generate)
     a. Draw the cluster indicator $\mathrm{z_i} \sim Cat(\boldsymbol{\pi})$
        - $z_i = k$ means that the current datapoint i belongs to cluster k
     b. Draw the sample $\boldsymbol{x_i} \sim \mathcal{N}(\boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i})$

Generative Models for Graphs

- How to artificially generate realistic graphs?
  - Generative models for graphs
  - Challenge: What are the latent factors influencing a graph?



- Applications: Forecast user behavior, large-scale analysis of algorithms, construct new molecules,…



Roadmap

- **Chapter: Graphs**
  1. Graphs & Networks
  2. **Generative Models**
     - **Models assuming (conditional) independent edges**
     - Preferential Attachment Models
     - Deep Generative Models
  3. Clustering
  4. Node Embeddings
  5. Ranking
  6. Semi-Supervised Learning
  7. Limitations of GNNs

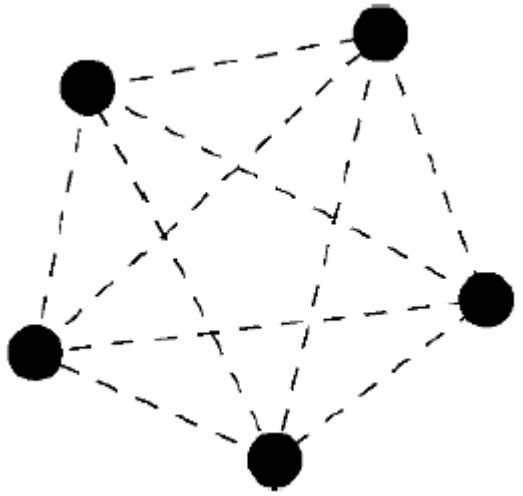
Generative Models for Graphs

- As you know, several laws apply for real world networks
- Goal: Generate synthetic graphs matching these criteria

- Seen before: Erdös-Renyi Random Graph Model

- Very simple generative process:
  Given $p \in [0,1]$ the edges are generated i.i.d. with

  $A_{ij} \sim Bernoulli(p)$



Erdös-Renyi Random Graph Model: Properties

- **Degree distribution**
  - probability of a vertex having degree k is $p_k = \binom{N-1}{k} \cdot p^k \cdot (1-p)^{N-1-k} \approx \frac{z^k e^{-z}}{k!}$ $with\ z = p(N-1)$ → corresponds to a Poisson distribution
  - But: In real world data we observe power-law distributions ☹
- **Diameter**
  - The diameter concentrates around $\log(N)/\log(z)$, where z is the average node degree in the graph
  - → The diameter grows slowly with the number of nodes
  - But: In real data we observe small (constant) or even shrinking diameters ☹
- **Clustering Coefficient**
  - The clustering coefficient is equal to the connection probability $p = z/(N-1)$
  - → No community structure and dependent on number of overall nodes
  - But: Real world data looks totally different! ☹

Generative Model for Graphs with Communities

- How do we define a probabilistic model for graphs that captures community structure?

- Observation: In real graphs nodes from the same community are more likely to connect than nodes from different communities
  - using the same probability p for all edges doesn't make sense!

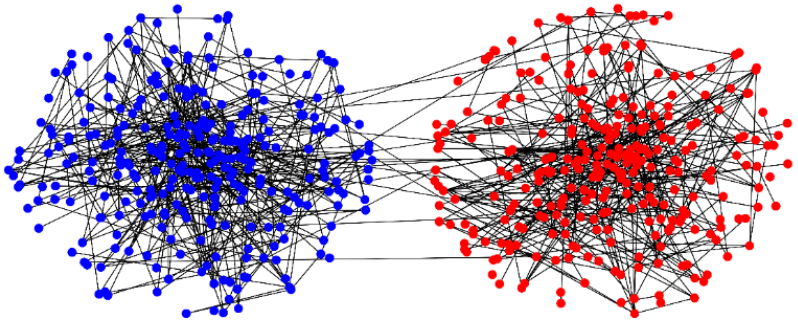
- Idea: Generalization of an Erdos-Renyi graph
  - nodes from the same community connect with probability p
  - nodes from different communities connect with probability q, where $p > q$

Planted Partition Model (PPM)

- We start with a set of nodes V, partitioned into 2 communities C_1, C_2
  - denote community assignment of node i as $z_i \in \{-1, 1\}$ – latent variables
- We generate an edge between every pair of nodes with probability

$$Pr(A_{ij} = 1 | z_i, z_j) = \begin{cases} p & \text{if } z_i = z_j \\ q & \text{if } z_i \neq z_j \end{cases}$$

- Here we consider undirected, unweighted graphs, but the model can easily be extended to other cases as well.



Graph generated by a PPM with
$N = 600, p = 6/600, q = 0.1/600$
$z_i = -1$ for blue nodes, $z_i = 1$ for red nodes

Limitations of the PPM

- PPM is an improvement over ER graph generator, but we would like to
  - generate graphs with an arbitrary number of communities
  - generate communities with different edge densities
  - generate graphs with “more interesting” structure than just dense communities + few edges between communities

- Can we generalize PPM even further to achieve these properties?

Stochastic Block Model (SBM)

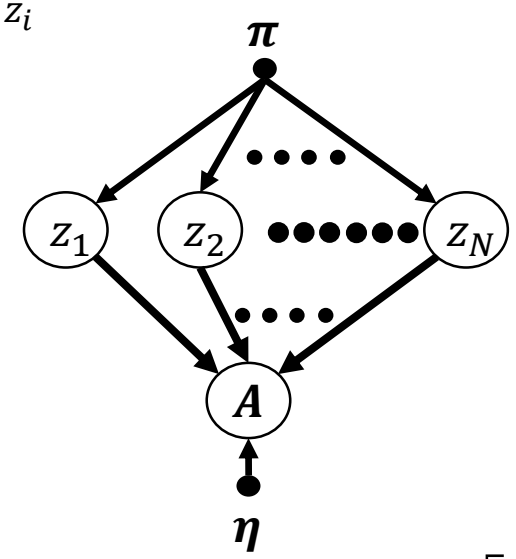
- **Stochastic block model** generalizes the PPM to graphs with arbitrary numbers and sizes of communities, and varying edge densities.

$$\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_k]$$

- Random variables:
  - $z_i \in \{1, \ldots, K\}$: node i belongs to block/community z_i
  - $\boldsymbol{A} \in \{0,1\}^{N \times N}$: adjacency matrix

- Model parameters:
  - $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$: community proportions
  - η_{uv}: edge probability between two nodes that are in communities u and v.

- Conditional distributions:
  - $\Pr(z_i = k) = \pi_k$
  - $\Pr(A_{ij} | z_i, z_j) = \text{Bernoulli}(\eta_{z_i z_j})$



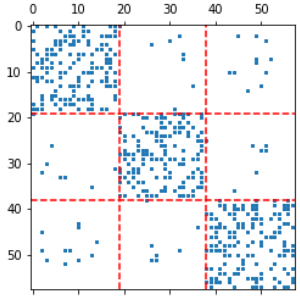
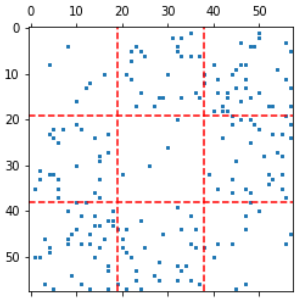
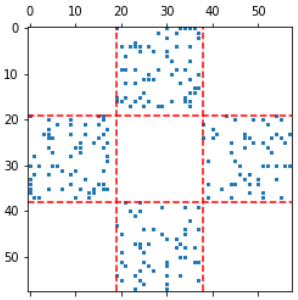
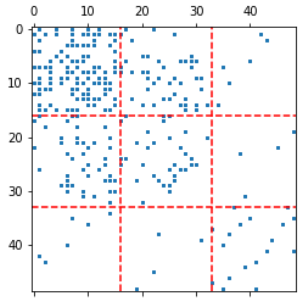
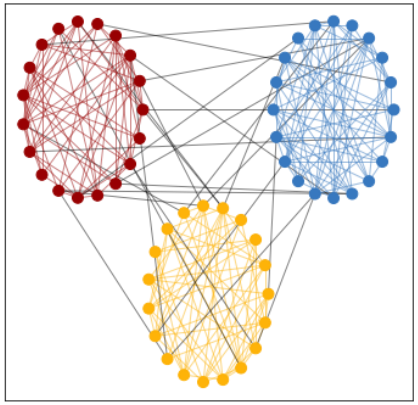
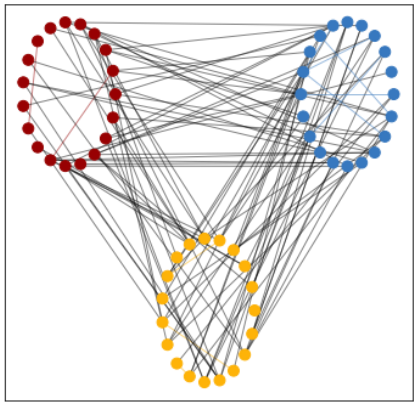
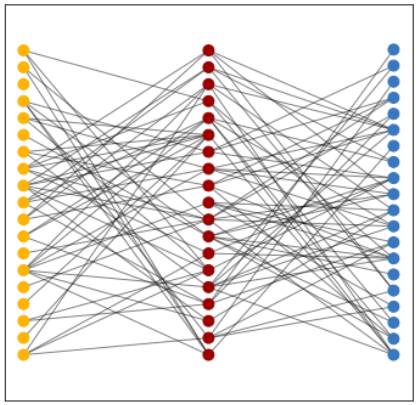
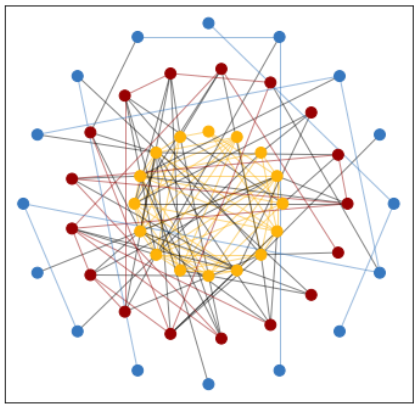
$$\boldsymbol{\eta} = \begin{bmatrix} \eta_{11} & \cdots & \eta_{1K} \\ \vdots & \ddots & \vdots \\ \eta_{K1} & \cdots & \eta_{KK} \end{bmatrix}$$

Planted Partition Model as a Stochastic Block Model

- Planted partition model can be viewed as a special case of the stochastic block model with the following parameters

$$\boldsymbol{\pi} = [0.5, 0.5] \qquad \boldsymbol{\eta} = \begin{bmatrix} p & q \\ q & p \end{bmatrix}$$

Some Types of Graphs Produced by SBM

	Assortative	Disassortative	Ordered	Core-periphery
η	$\begin{bmatrix} 0.4 & 0.02 & 0.02 \\ 0.02 & 0.4 & 0.02 \\ 0.02 & 0.02 & 0.4 \end{bmatrix}$	$\begin{bmatrix} 0.02 & 0.08 & 0.08 \\ 0.08 & 0.02 & 0.08 \\ 0.08 & 0.08 & 0.02 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.15 & 0.15 \\ 0.15 & 0 & 0.15 \\ 0.15 & 0.15 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.4 & 0.15 & 0.03 \\ 0.15 & 0.15 & 0.03 \\ 0.03 & 0.03 & 0.03 \end{bmatrix}$
A				
				

Limitations of SBM

- Stochastic block model is an elegant and well-studied model for graphs with communities, but it doesn't capture all patterns of real networks
  - real graphs have power-law degree distribution → Degree-Corrected SBM
    Karrer B. and Newman M. E. J.: Stochastic Blockmodels and Community Structure in Networks, in Physical Review E 83, 2011
  - real communities have more triangles → Geometric Block Model
    Galhotra S. et al.: The Geometric Block Model, in AAAI 2018
  - real communities are overlapping → Community-Affiliation Graph Model
    Yang J. and Leskovec J.: Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach, in WSDM 2013
- For an overview of recent advances in SBM see [Abbe2018]
  Abbe E.: Community Detection and Stochastic Block Models: Recent Developments, in JMLR 18, 2018

Roadmap

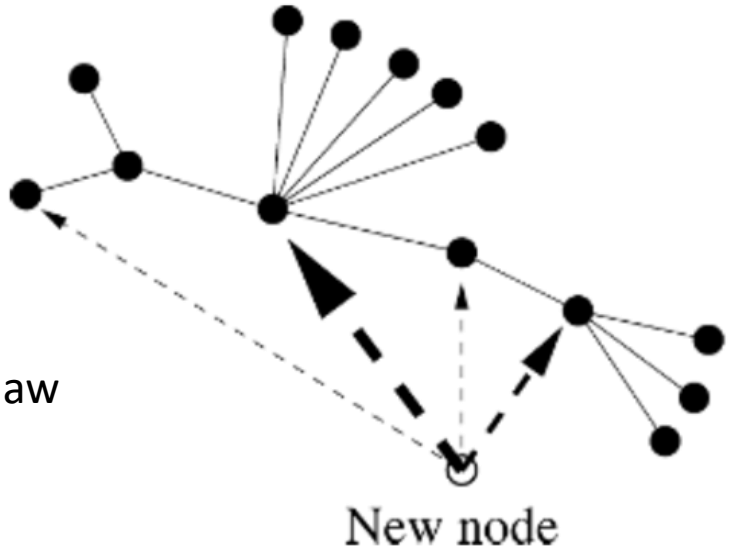
- **Chapter: Graphs**
  1. Graphs & Networks
  2. **Generative Models**
     - Models assuming (conditional) independent edges
     - **Preferential Attachment Models**
     - Deep Generative Models
  3. Clustering
  4. Node Embeddings
  5. Ranking
  6. Semi-Supervised Learning
  7. Limitations of GNNs

Preferential Attachment Models

- ER/PPM/SBM assume that the **edges** are generated **independently**
  - *all* nodes are given at the beginning; each potential edge corresponds to a Bernoulli distribution (independent of the others)

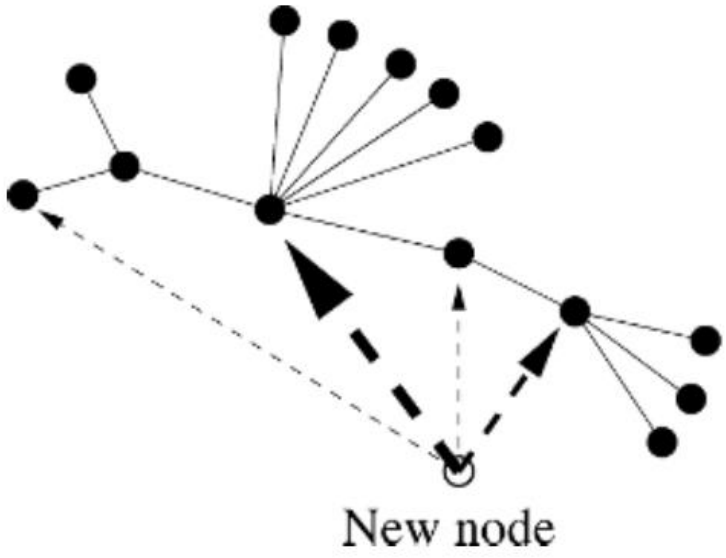
- Now: Generate network based on two processes
  - **Growth**: Instead of starting with all nodes, start with a small set of nodes and let the network grow over time by adding new nodes and edges
  - **Preferential attachment**:
    „rich get richer“ idea; probability of connecting nodes is proportional to the current degree of the nodes

→ „the rich get richer“ principle leads to a power law in the degree distribution



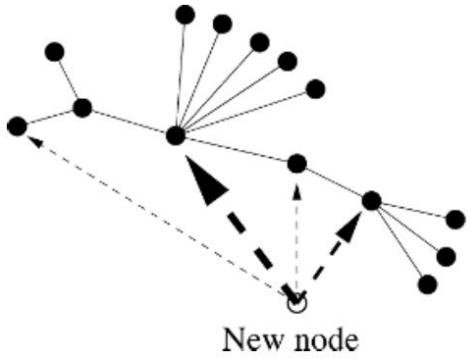
Preferential Attachment Models

- Prominent method: Initial Attractiveness (IA)
  - Extension of well-known BA model (Barabasi and Albert)
  - Allows to generate graphs following a power law degree distribution
  - Can realize a power law exponent γ in the range [2,infty)
    - BA model was stuck to exponent of γ=3



Initial Attractiveness [DMS2000]: Algorithm

1. Start with m_0 many nodes
2. Add a new node w
3. Simultaneously insert m directed edges (u, v)
   - **probability that the endpoint of an edge $(\mathbf{u}, \mathbf{v})$ corresponds to $\mathbf{v}$ is proportional to $\mathbf{A_v = A + indeg(v)}$**
     - A is the initial attractiveness of a node (same for all nodes)
     - $indeg(v)$ is the number of **currently** (!) incoming edges (**increases over time**)
   - Note: Not important where the edges (u, v) start
     - Example 1: all new edges start from w (like in the BA model)
     - Example 2: randomly select existing nodes
4. Goto step 2 until required number of nodes is obtained



[DMS2000] S. N. Dorogovtsev, J. F. F. Mendes and A. N. Samukhin, Structure of Growing Networks with Preferential Linking, Physical Review Letters

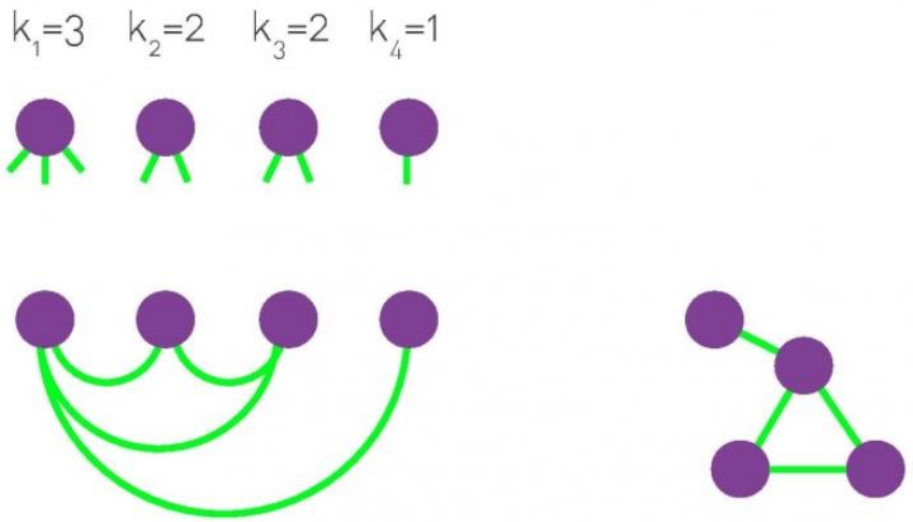
Initial Attractiveness: Properties

- Parameters: m (new edges per step) and A (initial attractiveness)
- **Degree distribution:**
  - The degree distribution follows a power law with exponent $\gamma = 2 + \frac{A}{m}$
  - Matches many real world data ☺
- **Diameter**:
  - For $m \geq 2$ the diameter grows as $O(\frac{\log N}{\log \log N})$
  - Matches the small world effect (diameter much smaller than number of nodes)
  - But: still slightly increasing in contrast to real world data 😐
- **Average degree:**
  - Remains constant over time
  - But: Increases for real world data; densification law ☹

Configuration Model

- Generating networks with arbitrary **specified** degree distribution
  1. Assign a degree to each node, represented as stubs or half-links
  2. Randomly select a stub pair and connect them
  - Depending on the order and way in which the stubs were chosen, we obtain different networks

- Preserves degree structure



Further Classical Graph Generators

- Many graph generators have been introduced
  - Overview presented in [CF2006]
  - Some further prominent methods:
    - Edge copying methods: realize community structure
    - Forest Fire Model: densification and shrinking diameter

[CF2006] Deepayan Chakrabarti, Christos Faloutsos: Graph mining: Laws, generators, and algorithms. ACM Comput. Surv. (CSUR) 38(1) (2006)

Roadmap

- **Chapter: Graphs**
  1. Graphs & Networks
  2. **Generative Models**
     - Models assuming (conditional) independent edges
     - Preferential Attachment Models
     - **Deep Generative Models**
  3. Clustering
  4. Node Embeddings
  5. Ranking
  6. Semi-Supervised Learning
  7. Limitations of GNNs

Motivation

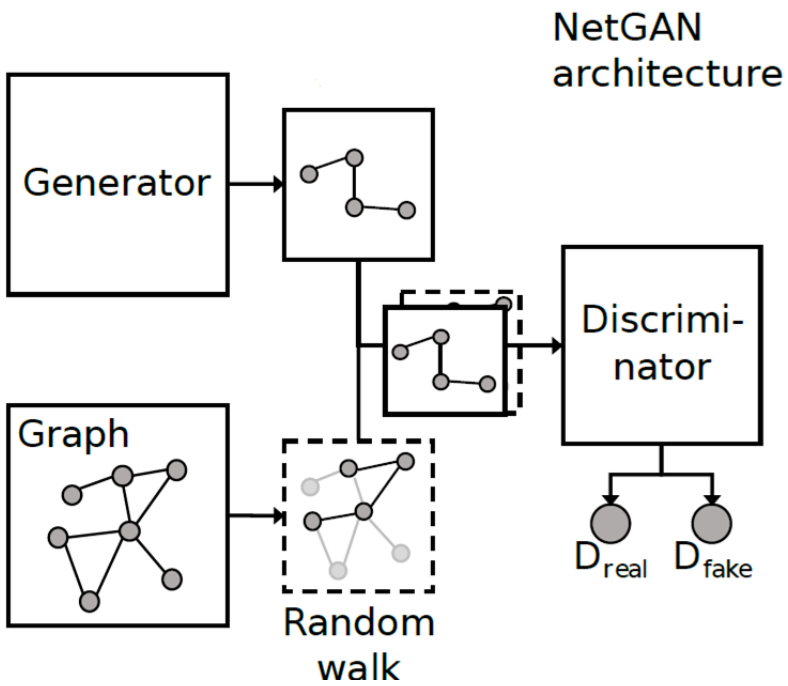
- All previously introduced generative models are “hand-crafted”
  - Observe properties (power law, triangles, communities, etc.) of real graphs → Build a generative model that generates them
  - Difficult to discover all relevant properties of real graphs
  - Difficult to “hand-craft” a single model capturing all properties simultaneously

- How can we find a model that captures all the complex (potentially even unknown) properties of real graphs?

- Let us use the concept of deep generative models
  - i.e. *flexible* models that can be *learned* based on given data

NetGAN: First GAN for Networks/Graphs

- How to adopt the GAN idea to graphs?
- Naive solution: Learning a distribution over adjacency matrices directly (treat them as images)
  - Computationally infeasible: N^2 possible edges for a graph with N nodes
  - Not permutation invariant over nodes
- **Solution**: learn a distribution over random walks instead
  - Recall: real graphs are sparse
  - Only consider non-zero entries



Bojchevski et.al. NetGAN: Generating Graphs via Random Walks. ICML 2018

Random Walk Definition

- Given graph $G = (V, E)$, walk length t
- Define as $\mathcal{W}_{v_i} = (w_j : 0 \leq j < t; w_0 = v_i)$ a random walk with starting node v_i.
- w_j is the j-th node in the random walk, and the nodes are sampled from

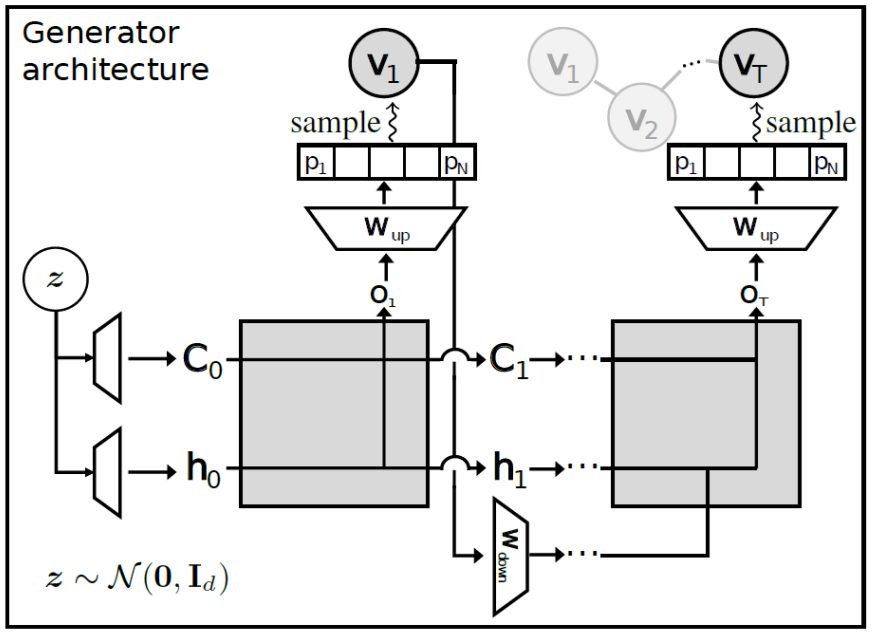
$$\Pr(w_{j+1} = v_u \mid \mathrm{w_j}) = \begin{cases} \frac{1}{\mathrm{d}_{w_j}}, if\ v_u \in \mathcal{N}(w_j) \\ 0, else \end{cases} \quad \forall j:\ 0 < j < t$$

where $\mathcal{N}(v_k)$ is the set of neighbors of node v_k and d_{v_k} its degree

NetGAN: Generator

- Based on Long Short Term Memory (LSTM) cells
- Generative process as follows:
  1. Sample latent noise from a normal distribution $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$
  2. Pass $\boldsymbol{z}$ through a neural network to obtain starting states C_0, h_0 for the LSTM
  3. Run the LSTM from this state to generate a sequence of nodes (random walk)

$$
\begin{aligned}
& & \boldsymbol{z} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d) \\
& & \boldsymbol{m}_0 &= g_{\theta'}(\boldsymbol{z}) \\
\boldsymbol{v}_1 &\sim \mathrm{Cat}(\sigma(\boldsymbol{p}_1)), & (\boldsymbol{p}_1, \boldsymbol{m}_1) &= f_\theta(\boldsymbol{m}_0, \mathbf{0}) \\
\boldsymbol{v}_2 &\sim \mathrm{Cat}(\sigma(\boldsymbol{p}_2)), & (\boldsymbol{p}_2, \boldsymbol{m}_2) &= f_\theta(\boldsymbol{m}_1, \boldsymbol{v}_1) \\
& \vdots & & \vdots \\
\boldsymbol{v}_T &\sim \mathrm{Cat}(\sigma(\boldsymbol{p}_T)), & (\boldsymbol{p}_T, \boldsymbol{m}_T) &= f_\theta(\boldsymbol{m}_{T-1}, \boldsymbol{v}_{T-1})
\end{aligned}
$$

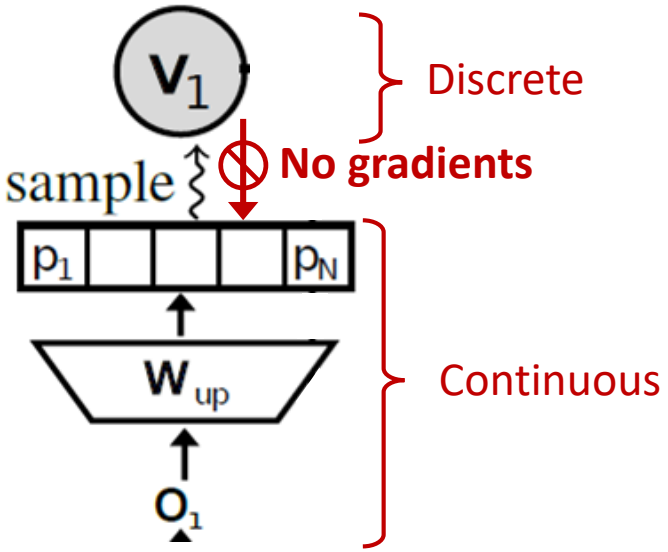


NetGAN: Generator

- **Problem**: sampling from a categorical distribution is **not differentiable** with respect to parameters π
  - Same problem as in variational inference

- But we need:
  a) Discrete samples to feed into the discriminator
  b) Gradients to train the generator

- Solution: Categorical Reparameterization Trick



Categorical Reparameterization Trick

- The random variable g is said to have a standard Gumbel distribution if

$$g = -\log(-\log(u)) \quad \text{with} \quad u \sim Uniform[0,1]$$

- Let v be a discrete random variable $P(v = k) \propto \boldsymbol{\pi}_k$ and let $\{g_k\}_{1 \le k \le K}$ be an i.i.d. sequence of standard Gumbel random variables. Then:

$$v = \arg\max_k [g_k + \log \boldsymbol{\pi}_k]$$

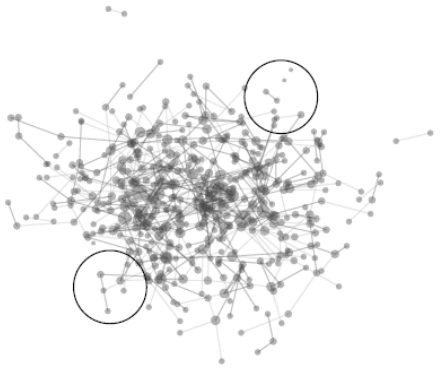
- Recipe for (reparametrized) sampling from a categorical distribution:
  1. Draw Gumbel noise by transforming uniform samples
  2. Add it to $\log \boldsymbol{\pi}_k$ which only has to be known up to a normalizing constant
  3. Take the value k that produces the maximum

Categorical Reparametrization Trick

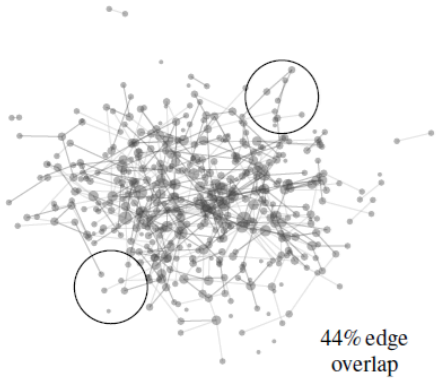
- Problem: The argmax in $v = \arg\max_k [g_k + \log \boldsymbol{\pi}_k]$ is still not differentiable
- Solution: **Approximate** it with softmax + temperature parameter τ
- All together**:** Approximate v via $v^* = \sigma\left(\frac{\log \boldsymbol{\pi} + \boldsymbol{g}}{\tau}\right)$ where σ is the softmax
  - As the temperature $\tau \to 0$, v^* converges to v
- We need to use the approximation with softmax only for the backward pass. The forward pass can be exact: $\arg\max_i v^*$
- Remark: The approximation comes only from the softmax. The reparametrization using Gumbel + argmax is exact

NetGAN: What can we do with it?

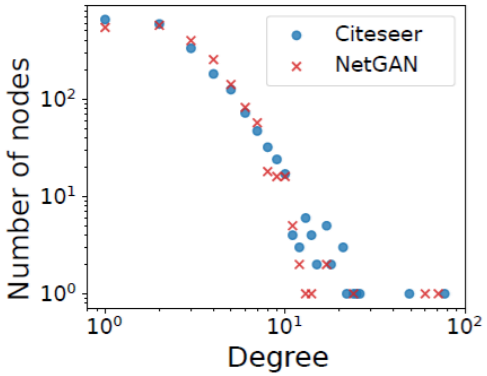
- NetGAN learns properties of real graphs without manually specifying them
- Generate graphs that have the same structure but are not replicas



(a) Original graph

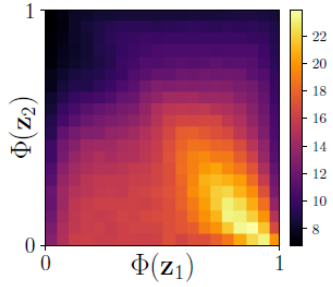


(b) Graph generated by NetGAN

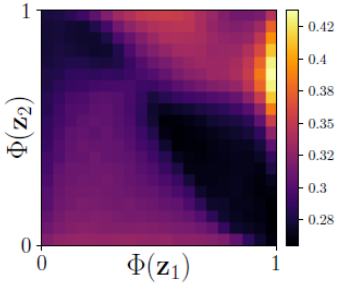


(c) Degree distribution comparison

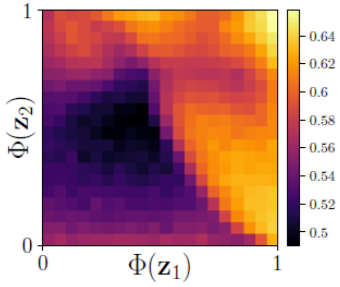
- Latent space interpolations produces smoothly changing graphs



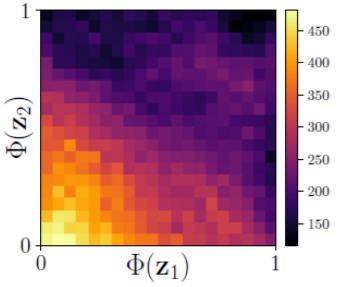
(a) Avg. degree of start node



(b) Avg. share of nodes in start community



(c) Gini coefficient (input graph: 0.48)



(d) Max. degree (input graph: 240)

Single vs. Multi-Graph Scenario

- NetGAN is designed for the setting of “learning from a single, large graph”
  - similar to classic network generators like SBM
  - Examples: social network, citation network, etc.
    - From an application point of view, the “actual entities/objects” one is usually interested in are the individual nodes of the graph (e.g. a person in a social network); different entities are interlinked with each other, thus, forming a graph

- Other scenario: learning from multiple, usually small, graphs
  - Example: molecules
    - From an application point of view, the “actual entities/objects” one is usually interested in are the graphs as a whole, i.e. the entity itself is a graph.

Questions

- Can the Erdös-Renyi model generate all graphs that the Stochastic Block Model can generate?
- Is the Initial Attractiveness model equal to the Erdös-Renyi model as $A \to \infty$?
- Why does π not need to be normalized for the Gumbel trick to work?

Summary

- Classic generative models for graphs are
  - (relatively) easy to analyze
  - but do not capture all important properties of real graphs

- Deep generative models learn to generate graphs that automatically capture the underlying laws and characteristics (e.g. power law, small world) without manually specifying them
  - though, theoretically analyzing such models is tricky
  - evaluation of generative models is hard in general, even harder for graphs
  - still a highly unexplored research field

- Reparameterization allows us to differentiate through random/stochastic discrete nodes in the computation graph

Machine Learning for Graphs and Sequential Data

Graphs – Generative Models

Lecturer: Prof. Dr. Stephan Günnemann

www.daml.in.tum.de

Summer Term 2020

Data Analytics and
Machine Learning