Random Graphs

ACMS 80770: Deep Learning with Graphs

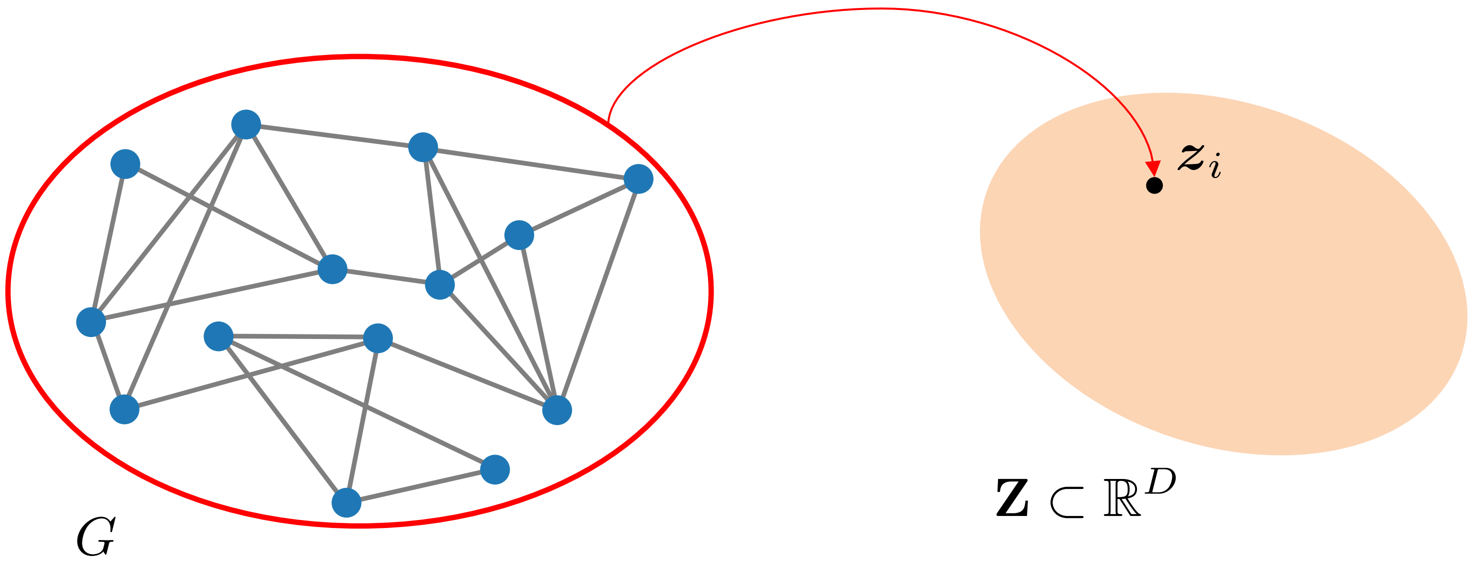
Instructor: Navid Shervani-Tabar

Department of Applied and Comp Math and Stats

UNIVERSITY OF NOTRE DAME

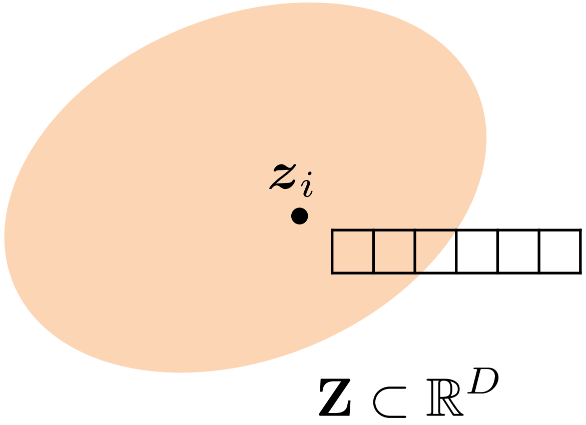
Intro

- So far in this course we have discussed **representation learning** from graphs.
- In this problem we construct **embedding** vectors from **graph** (and nodes and edges alike) to be used for inference in different problems.



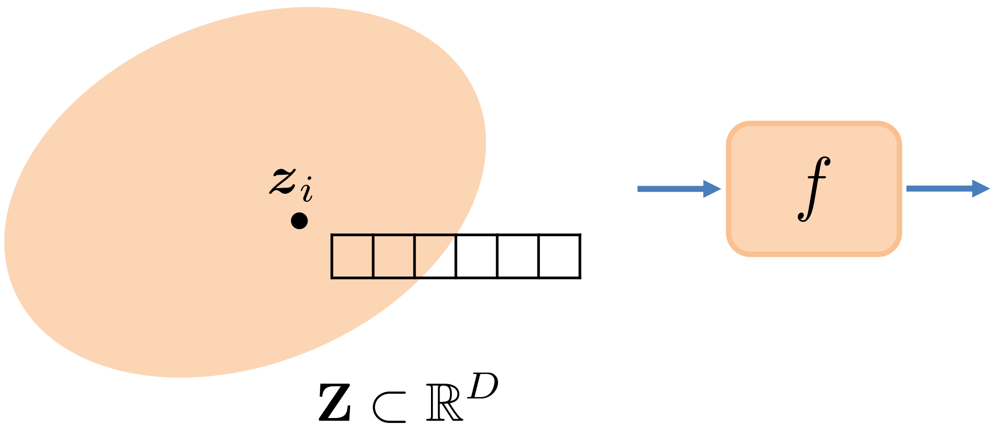
Graph Generation

- In this part of the course, we look at the **graph generation** task.
- In graph generation, a **deep generative model** takes an embedding vector as input and **returns a graph**.
- In principle this task is **inverse** of the graph representation learning.



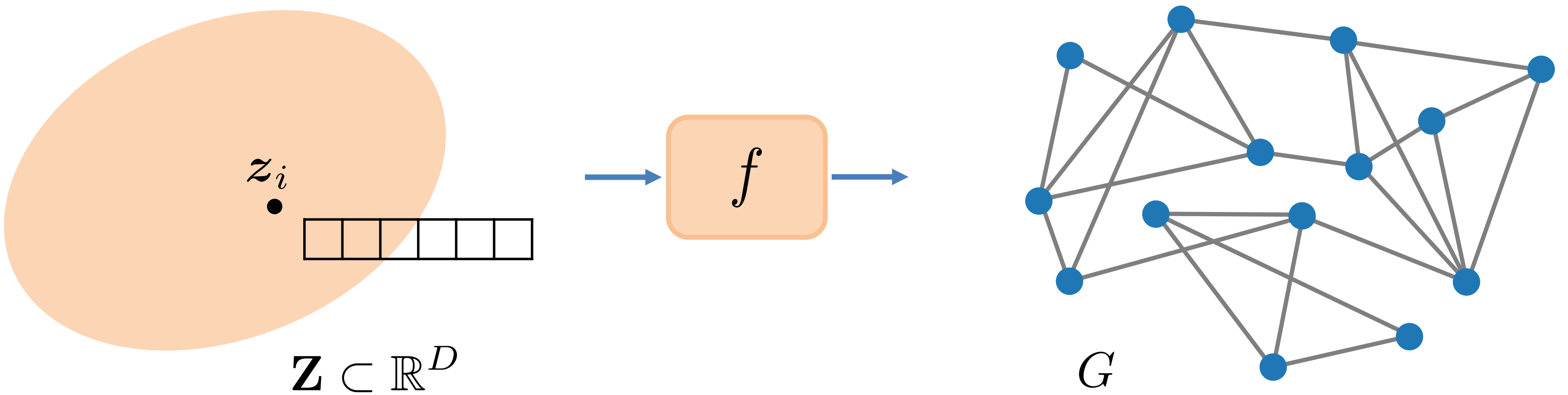
Graph Generation

- In this part of the course, we look at the **graph generation** task.
- In graph generation, a **deep generative model** takes an embedding vector as input and **returns a graph**.
- In principle this task is **inverse** of the graph representation learning.



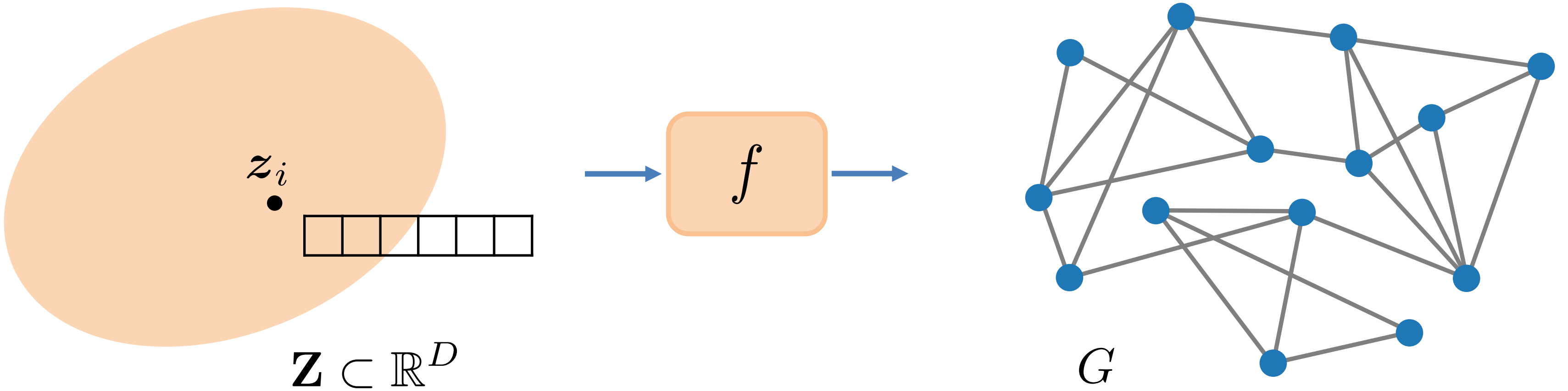
Graph Generation

- In this part of the course, we look at the **graph generation** task.
- In graph generation, a **deep generative model** takes an embedding vector as input and **returns a graph**.
- In principle this task is **inverse** of the graph representation learning.



Graph Generation

- In this part of the course, we look at the **graph generation** task.
- In graph generation, a **deep generative model** takes an embedding vector as input and **returns a graph**.
- In principle this task is **inverse** of the graph representation learning.



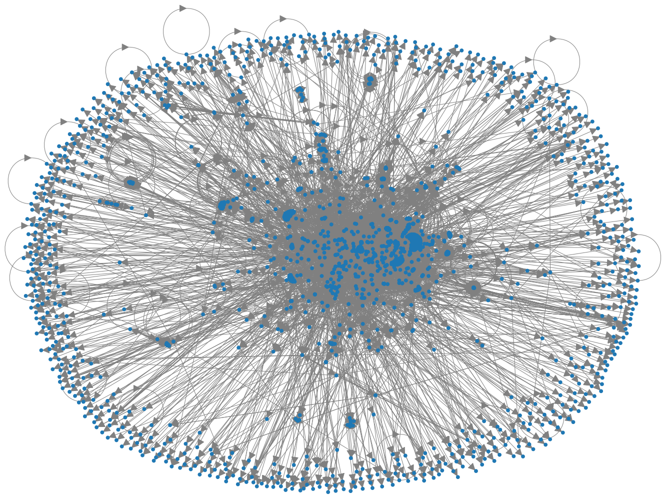
- However, before discussing deep graph generative models, we will look at the **characteristics** of the **real-world graphs**.

Graph Generation

- Graphs are constructed from seemingly simple components
  1. Nodes
  2. Edges
- But the main question is how to **place** these **edges** to recreate the **complexity** of the real-world graphs.

Graph Generation

- Graphs are constructed from seemingly simple components
  1. Nodes
  2. Edges
- But the main question is how to **place** these **edges** to recreate the **complexity** of the real-world graphs.
- Looking at the real-world graphs, the edges may **appear** to be **randomly connecting** pairs of nodes.
  - Internet,
  - social network.



Network Characteristics

- To verify this impression, we assume that edges are distributed **randomly** throughout the graph.
- Then, we inspect **if randomness can explain** these characteristics of the real-world graphs.

Network Characteristics

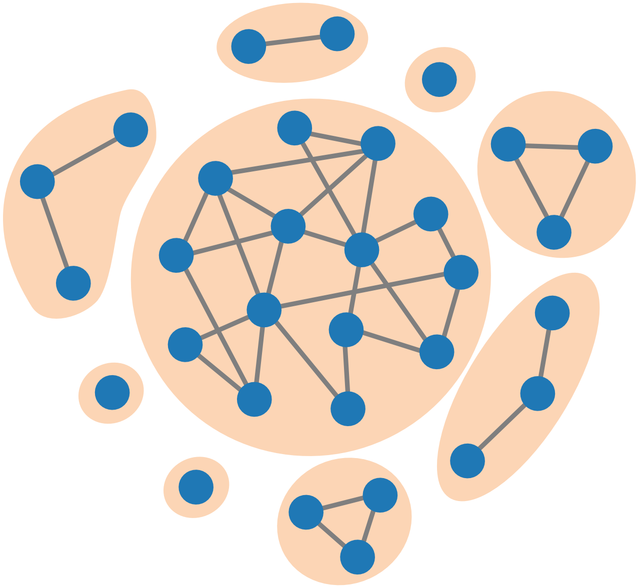
- To verify this impression, we assume that edges are distributed **randomly** throughout the graph.
- Then, we inspect **if randomness can explain** these characteristics of the real-world graphs.
- To address this, we first ask what properties **characterize** the **structure** of the real-world graphs.
- Important **characteristics** that affect the behavior of the graph include:
  - Connectedness
  - Path length
  - Degree distribution
  - Clustering coefficient

Connectedness

- Most real-world graphs are constructed of
  - A **giant component** that connects most of the nodes in the graph.
  - A few **small components**.

Connectedness

- Most real-world graphs are constructed of
  - A **giant component** that connects most of the nodes in the graph.
  - A few **small components**.



Connectedness

- Most real-world graphs are constructed of
  - A **giant component** that connects most of the nodes in the graph.
  - A few **small components**.
- **Actors' network** is a graph consisting of nodes that represent actors and edges connecting them based on the appearance of two actors in a film together.
  - Actors Network consists of **449913 nodes**.
  - The **largest component** of two actors Network consists of 440971 nodes.
  - This makes for **98%** of the graph!

Connectedness

- Some real-world graphs, however, only consists of **one component**.
- This may be dictated by the **nature** of the data or by the **measurement** approach used.

Connectedness

- Some real-world graphs, however, only consists of **one component**.
- This may be dictated by the **nature** of the data or by the **measurement** approach used.
  - **Internet:**
    - Internet is a communication Network whose underlying **nature** is to provide **connection** between all its entities
    - Therefore, **disconnected** components do **not have a use**.
    - Hence, the **largest** component of the internet is the graph **itself**.

Connectedness

- Some real-world graphs, however, only consists of **one component**.
- This may be dictated by the **nature** of the data or by the **measurement** approach used.
  - **Web:**
    - The structure of the web is mapped using web **crawlers**.
    - If we use a **single** crawler to map the web, it only visits the web pages that are **linked** by other web pages.
    - Therefore, the **measurement** approach dictates that the whole network has only one big giant component.

Small-world effect

- **Small-world effect** is a phenomenon that states for most real-world graphs the **typical distance** between the pairs of nodes is **short**.

Small-world effect

- **Small-world effect** is a phenomenon that states for most real-world graphs the **typical distance** between the pairs of nodes is **short**.
- Let d_{ij} represent the **distance** between nodes v_i and v_j.
- Then, the **mean distance** between v_i and any node $v_j \in \mathrm{V}$ is

$$\ell_i = \frac{1}{|V|} \sum_j d_{ij}$$

Small-world effect

- **Small-world effect** is a phenomenon that states for most real-world graphs the **typical distance** between the pairs of nodes is **short**.
- Let d_{ij} represent the **distance** between nodes v_i and v_j.
- Then, the **mean distance** between v_i and any node $v_j \in \mathrm{V}$ is

$$\ell_i = \frac{1}{|V|} \sum_j d_{ij}$$

- Thus, the mean distance between any two nodes $v_i, v_j \in \mathrm{V}$ in the graph is

$$\ell = \frac{1}{|V|} \sum_i \ell_i = \frac{1}{|V|^2} \sum_i \sum_j d_{ij}$$

Small-world effect

- However, d_{ij} is **not defined** for nodes v_i and v_j that do **not** belong to the **same component**.

Small-world effect

- However, d_{ij} is **not defined** for nodes v_i and v_j that do **not** belong to the **same component**.
- Hence, we reformulate ℓ as

$$\ell = \frac{\sum_m \sum_{ij \in V_m} \mathrm{d}_{ij}}{\sum_m |V_m|^2}$$

where $\mathcal{A}_m$ represents the component m in the graph with $|V_m|$ indicating the number of the nodes in that cluster.

Small-world effect

- However, d_{ij} is **not defined** for nodes v_i and v_j that do **not** belong to the **same component**.
- Hence, we reformulate ℓ as

$$\ell = \frac{\sum_m \sum_{ij \in V_m} \mathrm{d}_{ij}}{\sum_m |V_m|^2}$$

where $\mathcal{A}_m$ represents the component m in the graph with $|V_m|$ indicating the number of the nodes in that cluster.

- Looking at the **real-world graphs**, for graphs with sizes of order of millions of nodes, this measure is typically **less than 10**.

Diameter

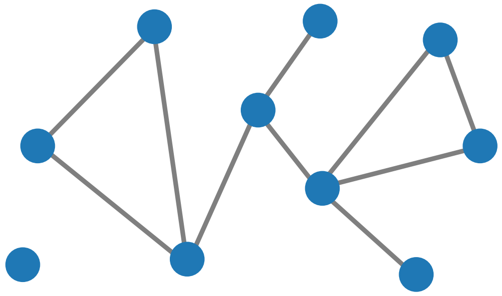
- **Diameter** of a graph is the longest, finite distance between any two nodes in the graph.
- One may suggest to inspect the diameter of the graph instead of looking at the average distance.

Diameter

- **Diameter** of a graph is the longest, finite distance between any two nodes in the graph.
- One may suggest to inspect the diameter of the graph instead of looking at the average distance.
- There are, however, two **downsides** to use of this measure to study real-world graphs:
  - It only measures the **extreme end** of the distribution of the distances in the graph.
  - This measure could substantially **change** by a single **modification** to the graph.

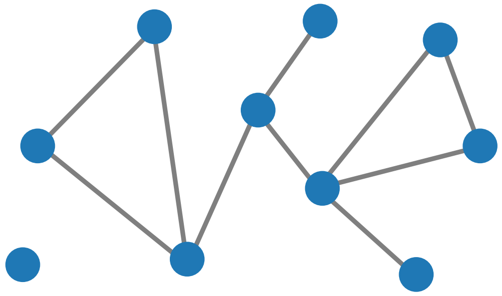
Degree Distribution

- **Degree distribution** is one of the most **fundamental** properties of the graph structure.
- Let p_k indicate the fraction of nodes in the graph with degree k.
  - Example:



Degree Distribution

- **Degree distribution** is one of the most **fundamental** properties of the graph structure.
- Let p_k indicate the fraction of nodes in the graph with degree k.
  - Example:

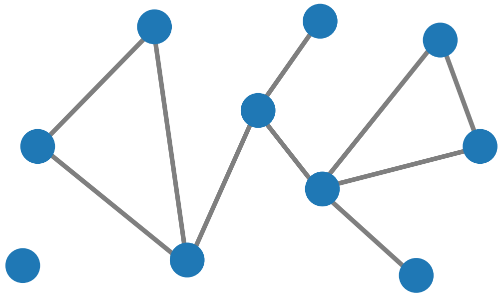


- For the graph above with $|V| = 10$, we can write

$$p_0 = \frac{1}{10}$$

Degree Distribution

- **Degree distribution** is one of the most **fundamental** properties of the graph structure.
- Let p_k indicate the fraction of nodes in the graph with degree k.
  - Example:

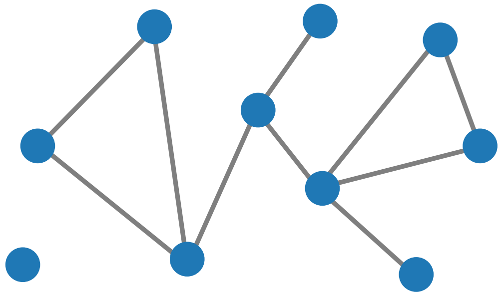


- For the graph above with $|V| = 10$, we can write

$$p_0 = \frac{1}{10}, \quad p_1 = \frac{2}{10}$$

Degree Distribution

- **Degree distribution** is one of the most **fundamental** properties of the graph structure.
- Let p_k indicate the fraction of nodes in the graph with degree k.
  - Example:

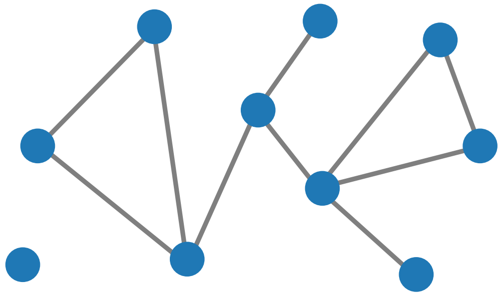


- For the graph above with $|V| = 10$, we can write

$$p_0 = \frac{1}{10}, \quad p_1 = \frac{2}{10}, \quad p_2 = \frac{4}{10}$$

Degree Distribution

- **Degree distribution** is one of the most **fundamental** properties of the graph structure.
- Let p_k indicate the fraction of nodes in the graph with degree k.
  - Example:

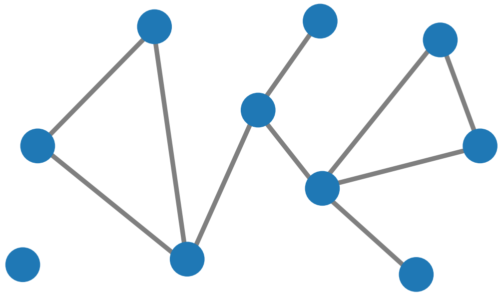


- For the graph above with $|V| = 10$, we can write

$$p_0 = \frac{1}{10}, \quad p_1 = \frac{2}{10}, \quad p_2 = \frac{4}{10}, \quad p_3 = \frac{2}{10}$$

Degree Distribution

- **Degree distribution** is one of the most **fundamental** properties of the graph structure.
- Let p_k indicate the fraction of nodes in the graph with degree k.
  - Example:

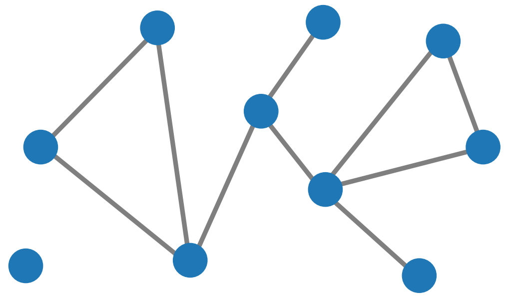


- For the graph above with $|V| = 10$, we can write

$$p_0 = \frac{1}{10}, \quad p_1 = \frac{2}{10}, \quad p_2 = \frac{4}{10}, \quad p_3 = \frac{2}{10}, \quad p_4 = \frac{1}{10}$$

Degree Distribution

- **Degree distribution** is one of the most **fundamental** properties of the graph structure.
- Let p_k indicate the fraction of nodes in the graph with degree k.
  - Example:

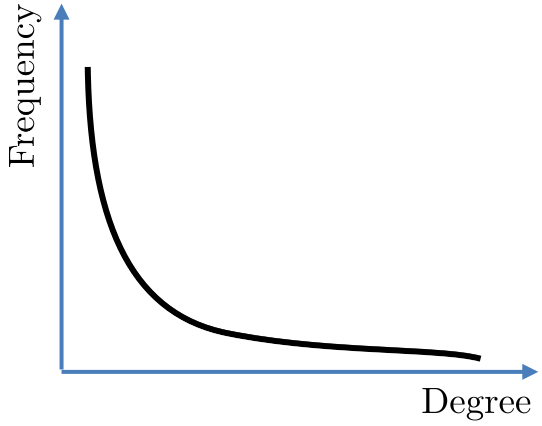


$$p_0 = \frac{1}{10}, \quad p_1 = \frac{2}{10}, \quad p_2 = \frac{4}{10}, \quad p_3 = \frac{2}{10}, \quad p_4 = \frac{1}{10}$$

- p_k represents the **degree distribution** of the network.
- In other words, p_k represents the probability of a randomly chosen node v_i having degree k.

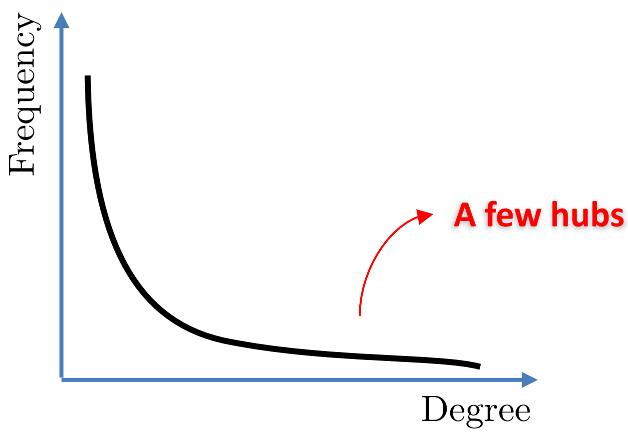
Power-law distribution

- The degree distribution for most real-world examples have a **heavy tail**.



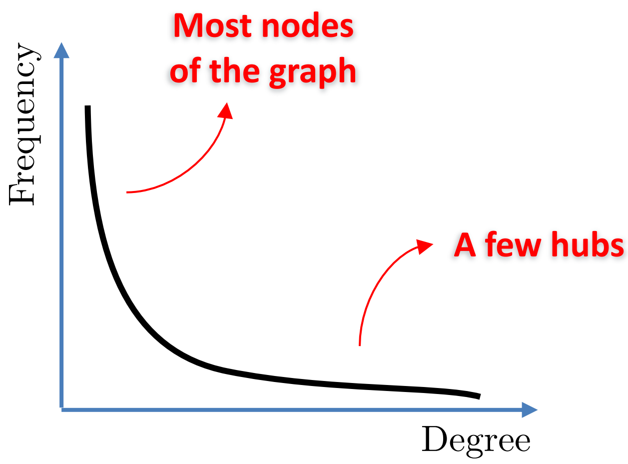
Power-law distribution

- The degree distribution for most real-world examples have a **heavy tail**.
- That is, the probability of having highly connected nodes is **non-zero**.
- These nodes with unusually high degree are referred to as **hubs**.



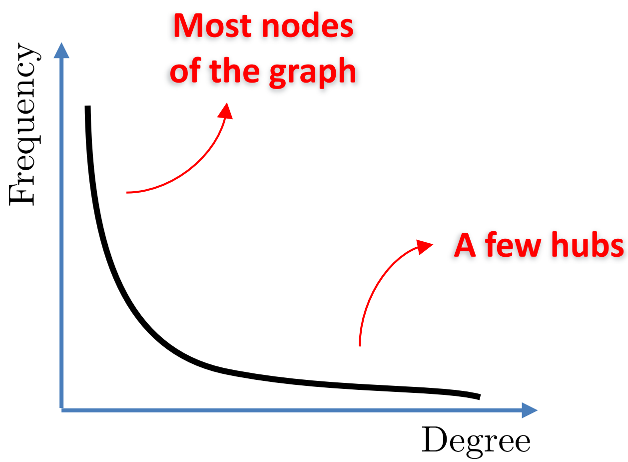
Power-law distribution

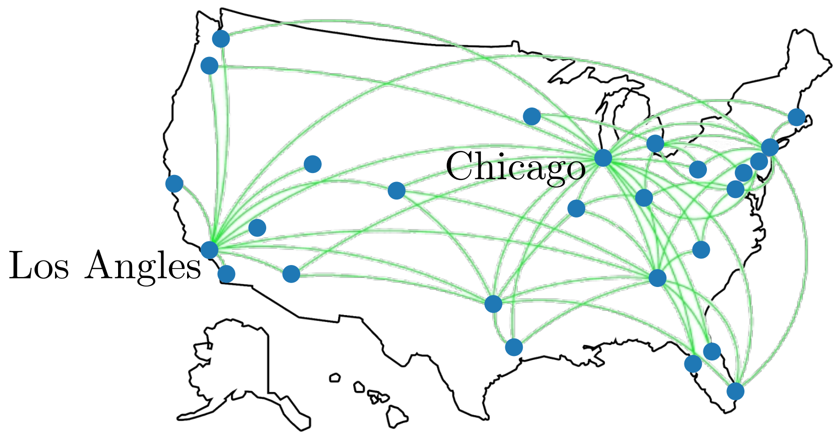
- The degree distribution for most real-world examples have a **heavy tail**.
- That is, the probability of having highly connected nodes is **non-zero**.
- These nodes with unusually high degree are referred to as **hubs**.
- Meanwhile, **most** nodes have a **very low degree**.



Power-law distribution

- The degree distribution for most real-world examples have a **heavy tail**.
- That is, the probability of having highly connected nodes is **non-zero**.
- These nodes with unusually high degree are referred to as **hubs**.
- Meanwhile, **most** nodes have a **very low degree**.





Power-law distribution

- When this degree distribution is plotted on a **log-scale**, it roughly follows a **straight line**.

Power-law distribution

- When this degree distribution is plotted on a **log-scale**, it roughly follows a **straight line**.
- We can **formulate** this interpretation as

$$\ln p_k = -\alpha \ln k + C$$

where α is the slope of the line.

Power-law distribution

- When this degree distribution is plotted on a **log-scale**, it roughly follows a **straight line**.
- We can **formulate** this interpretation as

$$\ln p_k = -\alpha \ln k + C$$

  where α is the slope of the line.
- Taking the **exponential** of both sides, we have

$$p_k \propto k^{-\alpha}$$

- This distribution is referred to as **power-law distribution**.

Scale-free Graphs

- Power-law distribution is characterized by its **heavy tail**.
- Power-law is also known as **pareto** distribution.
- Graphs that follow a power-law degree distribution are known as **scale-free networks**.
- These graphs consist of:
  - **A core**, that contains most of the nodes in the graph.
  - **Longer streams** that are attached to the core.

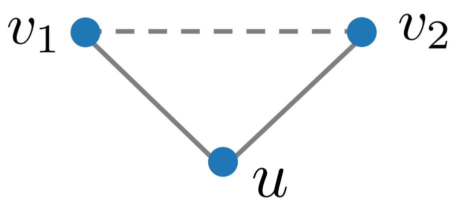
Clustering Coefficients

- Earlier in the course, we defined the clustering coefficient of a node as the **density of the triangles** in the vicinity of the node.

Connected pair of neighbors

$$c_u = \frac{|(v_1, v_2) \in \mathcal{E} : v_1, v_2 \in \mathcal{N}(u)|}{\frac{1}{2} d_u (d_u - 1)}$$

Total pair of neighbors



- Alternatively, one can interpret the clustering coefficient as the **average probability** of two neighbors of a node being **connected**.

Graph generation

- Now we look at the **graph generation** approaches.
- To motivate the **deep graph generative** models, we first look at the **traditional** graph generation algorithms.
- These methods try to **construct** non-trivial graphs that have the desirable **properties** of the **real-world** graphs.

Graph generation

- Now we look at the **graph generation** approaches.
- To motivate the **deep graph generative** models, we first look at the **traditional** graph generation algorithms.
- These methods try to **construct** non-trivial graphs that have the desirable **properties** of the **real-world** graphs.
- We refer to the construction algorithm as the **generative process.**
- Here we look at four different models:
  - Erdos-Renyi model
  - Configuration model
  - Stochastic Block model
  - Barabasi-Albert model

Erdos-Renyi Model

- **Erdos-Renyi model**, also known as the **random graph model** is the most well-known graph generation algorithm.
- In this model, edges are considered to **randomly connect** nodes in the graph.
- To that end, given the size of the graph, the model assumes that the **probability** of occurrence of an edge between any given nodes in the graph is **constant**.
- Mathematically put,

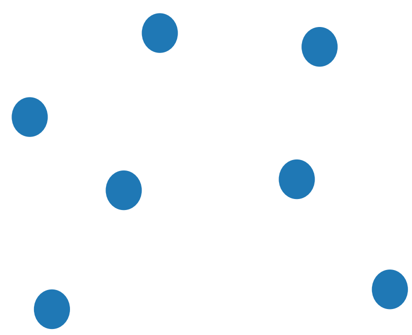
$$p\left(A_{ij}=1\right)=r, \quad \forall v_i, v_j \in V, \quad v_i \neq v_j$$

Erdos-Renyi Model

- The **generative process** for this model is as follows:
  - Set the total **number of the nodes** in the graph $|V|$.
  - For each pair of nodes v_i and v_j, sample a uniform random number $r' \sim \mathcal{U}[0,1]$.
    - If $r' > r$, $A_{ij} = 1$.
    - Otherwise, $A_{ij} = 0$.

Erdos-Renyi Model

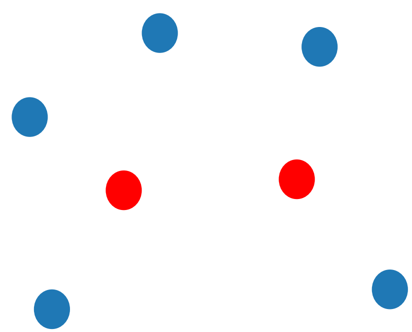
- The **generative process** for this model is as follows:
  - Set the total **number of the nodes** in the graph $|V|$.
  - For each pair of nodes v_i and v_j, sample a uniform random number $r' \sim \mathcal{U}[0,1]$.
    - If $r' > r$, $A_{ij} = 1$.
    - Otherwise, $A_{ij} = 0$.
- *Example*:



$r = 0.5$

Erdos-Renyi Model

- The **generative process** for this model is as follows:
  - Set the total **number of the nodes** in the graph $|V|$.
  - For each pair of nodes v_i and v_j, sample a uniform random number $r' \sim \mathcal{U}[0,1]$.
    - If $r' > r$, $A_{ij} = 1$.
    - Otherwise, $A_{ij} = 0$.
  - *Example*:

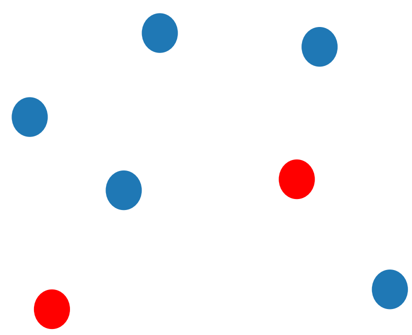


$r = 0.5$

$r' = 0.2$

Erdos-Renyi Model

- The **generative process** for this model is as follows:
  - Set the total **number of the nodes** in the graph $|V|$.
  - For each pair of nodes v_i and v_j, sample a uniform random number $r' \sim \mathcal{U}[0,1]$.
    - If $r' > r$, $A_{ij} = 1$.
    - Otherwise, $A_{ij} = 0$.
- *Example*:

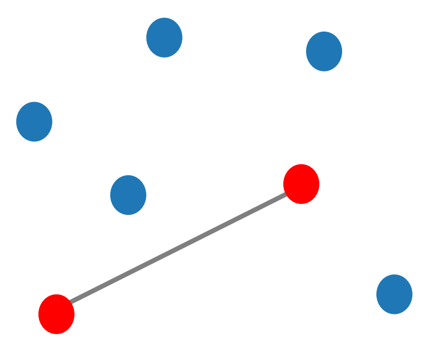


$r = 0.5$

$r' = 0.7$

Erdos-Renyi Model

- The **generative process** for this model is as follows:
  - Set the total **number of the nodes** in the graph $|V|$.
  - For each pair of nodes v_i and v_j, sample a uniform random number $r' \sim \mathcal{U}[0,1]$.
    - If $r' > r$, $A_{ij} = 1$.
    - Otherwise, $A_{ij} = 0$.
- *Example*:

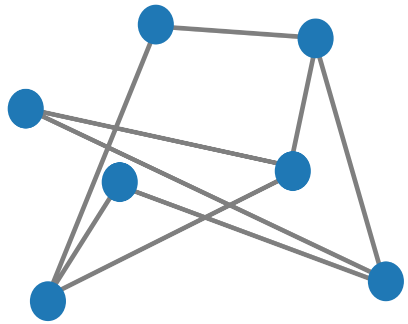


$r = 0.5$

$r' = 0.7$

Erdos-Renyi Model

- The **generative process** for this model is as follows:
  - Set the total **number of the nodes** in the graph $|V|$.
  - For each pair of nodes v_i and v_j, sample a uniform random number $r' \sim \mathcal{U}[0,1]$.
    - If $r' > r$, $A_{ij} = 1$.
    - Otherwise, $A_{ij} = 0$.
  - *Example*:



$r = 0.5$

Erdos-Renyi Model

- The Erdos-Renyi model can control the **density** of the graph through the **parameter** r.
- However, it is not able to capture other graph properties, such as:
  - Degree distribution,
  - Community structure
  - Clustering coefficient

Erdos-Renyi Model

- The Erdos-Renyi model can control the **density** of the graph through the **parameter** r.
- However, it is not able to capture other graph properties, such as:
  - Degree distribution,
  - Community structure
  - Clustering coefficient
- Therefore, Erdos-Renyi does **not capture** characteristics of the **real-world** graphs.
- Nevertheless, it is a **good indicator** if the observed characteristics can be explained by the randomness, or it is a result of a more complex property.

Configuration Model

- **Configuration model** builds graphs from a **sequence** of the node degrees

$$\{d_1, \dots, d_{|V|}\}$$

Configuration Model

- **Configuration model** builds graphs from a **sequence** of the node degrees

$$\{d_1, \dots, d_{|V|}\}$$

- Alternatively, one can instead **sample a sequence** of node degrees from a distribution p_k to construct graphs with desired degree distribution.

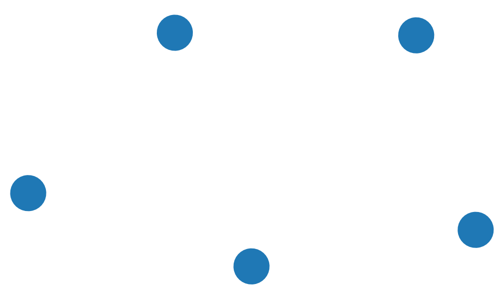
$$\{d_1, \dots, d_{|V|}\} \sim p_k$$

Configuration Model

- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \ldots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.

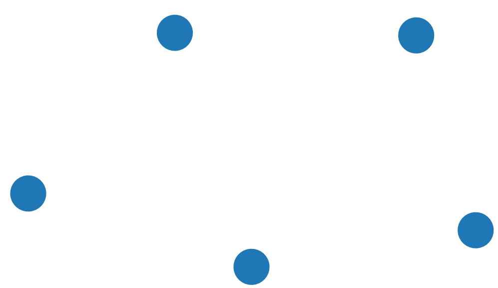
Configuration Model

- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \dots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



Configuration Model

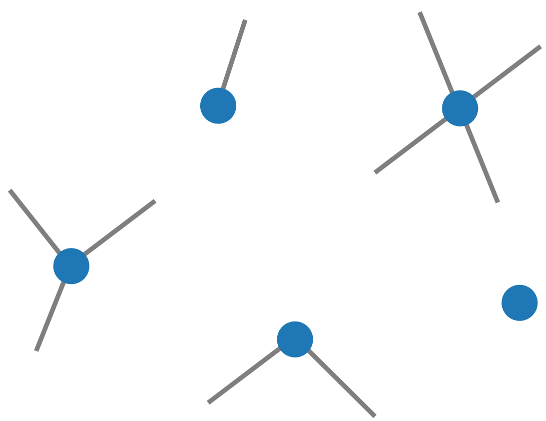
- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \ldots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



$$\{3, 1, 4, 2, 0\} \sim p_k$$

Configuration Model

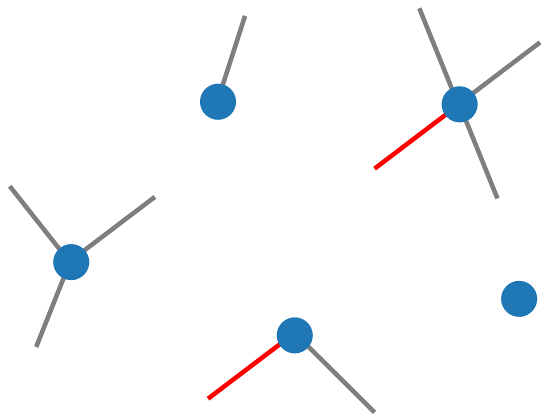
- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \ldots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



$$\{3, 1, 4, 2, 0\} \sim p_k$$

Configuration Model

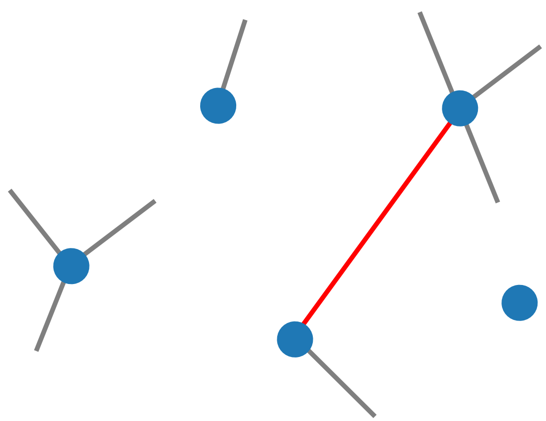
- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \dots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



$$\{3, 1, 4, 2, 0\} \sim p_k$$

Configuration Model

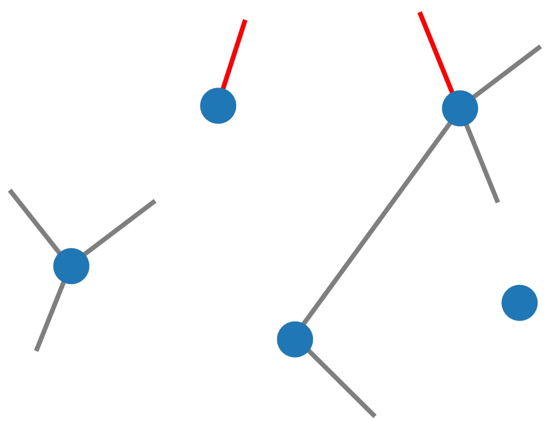
- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \ldots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



$$\{3, 1, 4, 2, 0\} \sim p_k$$

Configuration Model

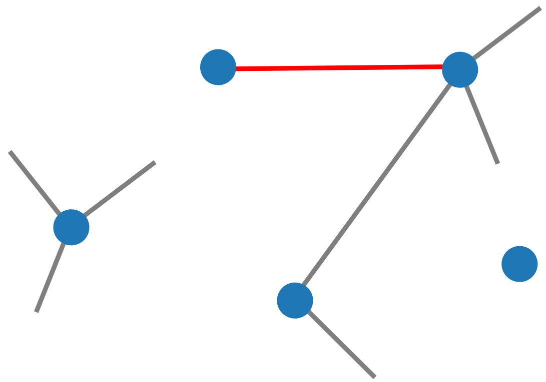
- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \ldots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



$$\{3, 1, 4, 2, 0\} \sim p_k$$

Configuration Model

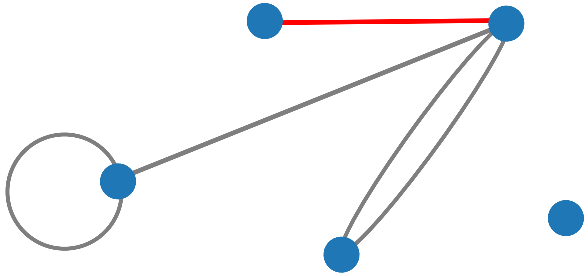
- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \dots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



$\{3, 1, 4, 2, 0\} \sim p_k$

Configuration Model

- The generative process of the configuration model is as follows:
  - **Set** the total number of the nodes in the graph $|V|$.
  - **Sample** a sequence of node degrees $\{d_1, \ldots, d_{|V|}\} \sim p_k$.
  - Given d_i, connect d_i **half-edges** to each node v_i.
  - Randomly **choose a pair** of half-edges and connect them.
- *Example:*



$$\{3, 1, 4, 2, 0\} \sim p_k$$

Configuration Model

- However, there are some **downsides** to this approach.
- First, for all half-edges to be paired, the total sum of the node degrees should be even.
- To remedy that one can reject sampled sequences with an odd sum $\sum_i^{|V|} d_i$.

Configuration Model

- However, there are some **downsides** to this approach.
- First, for all half-edges to be paired, the total sum of the node degrees should be even.
- To remedy that one can reject sampled sequences with an odd sum $\sum_i^{|V|} d_i$.
- Another issue is that the self-loops and multi-edges are often absent from real graphs.
- However, randomly connecting the half-edges may result in self-connected nodes and multiple edges between a pair of nodes.
- Nonetheless, as the size of the graph grows, the number of such cases becomes negligible.

Stochastic Block Model

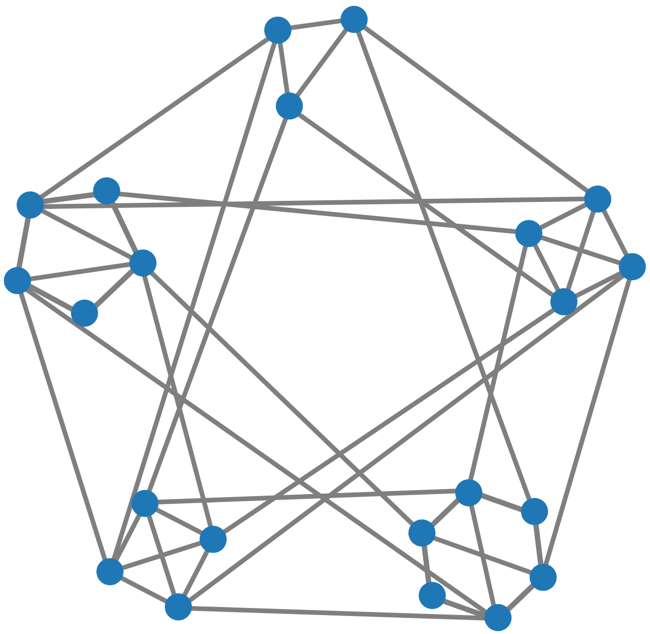
- **Stochastic block model** is designed to capture the community structure in the real-world graphs.
- To that end, SBM defines a number of node clusters $\mathcal{A}_1, \dots, \mathcal{A}_\gamma$ to represent different communities.
- Then, it assigns each node $v_i \in \mathrm{V}$ to one of these clusters by sampling from a categorical distribution $(p_1, \dots, p_\gamma)$ representing the probability of nodes belonging to each of these clusters.
- Finally, by setting different edge probabilities to inter-cluster and within-cluster pairs of nodes, SBM generates a graph.
- To that end, it defines a block-block probability matrix $\boldsymbol{C}$, where $\boldsymbol{C}_{ij}$ represents the probability of existence of an edge between nodes of two clusters $\mathcal{A}_i$ and $\mathcal{A}_j$.

Stochastic Block Model

- Thus, the generative process of this model can be summarized as follows:
  - Set the total number of nodes.
  - Assign each node $v_m \in \mathrm{V}$ to a block $\mathcal{A}_i$.
  - For each pair of nodes $v_m \in \mathrm{A}_i$ and $v_n \in \mathrm{A}_j$, sample edge with the probability

$$p\left(A_{ij}=1\right)=\boldsymbol{C}_{ij}, \quad \forall v_m, v_n \in V, \quad v_m \in \mathcal{A}_i, \quad v_n \in \mathcal{A}_j$$

Stochastic Block Model



Stochastic Block Model

- While the model discussed here shows the most basic SBM model, the shared property of all these models is generating graphs that show community structure.
- One application of the SBM model is to study the community detection algorithms.
- However, the downside of these models is that they do not capture the characteristics of the real-world networks.
- For example, setting the same edge probabilities for all the nodes in a block yields similar structural properties (e.g. clustering coefficient, degree) for all the nodes in the graph.
- Therefore, SBM fails to capture the degree distribution of the real-world graphs.

Barabasi-Albert model

- In the previous models, the goal was to recreate real-world graphs and study their structural features.
- In these methods, the parameters of the graph, such as nodes, or degree distribution was fixed in the beginning.
- Thus, the graph emerged at once.
- In the Barabasi-Albert model, however, the goal is to reenact the process of formation of the real-world graphs.
- Rather than constructing a graph with real-world characteristics, it investigates why such properties come to existence in the first place.

Barabasi-Albert model

- Barabasi-Albert model tries to construct graphs with real-world degree distribution.
- Many real-world graphs follow power law degree distribution.
- In other words, probability of a given node v_i having degree k

$$p\left(d_i = k\right) \propto k^{\alpha}$$

  Follows the power-law distribution (with $\alpha > 1$).
- Power-law distributions have heavy tail.
- Barabasi-Albert model constructs graphs that have a degree distribution that follows the power-law distribution.

Barabasi-Albert model

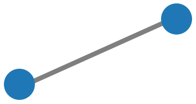
- The generative process of Barabasi-Albert model consists of:
  - Construct a complete graph with m_0 nodes.
  - Iteratively add a new node v_t to the graph.
  - Connect v_t to $m \leq m_0$ nodes that already exist in the graph $v_i \in V^{(t)}$ with the probability

$$P\left(A_{ti}=1\right)=\frac{d_i^{(t)}}{\sum_{v_j \in V^{(t)}} d_j^{(t)}}$$

- Based on BA, new nodes are connected to the nodes of the graph with a probability proportional to their degree.
- This follows the “rich gets richer” notion.

Barabasi-Albert model

- *Example:*

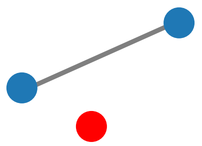


$$m = 2$$

$$m_0 = 2$$

Barabasi-Albert model

❖ *Example:*

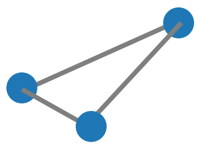


$$m = 2$$

$$m_0 = 2$$

Barabasi-Albert model

- *Example:*

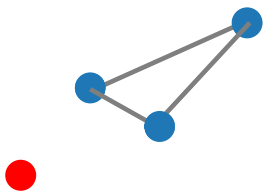


$$m = 2$$

$$m_0 = 2$$

Barabasi-Albert model

- *Example:*

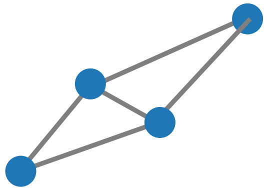


$$m = 2$$

$$m_0 = 2$$

Barabasi-Albert model

- *Example:*

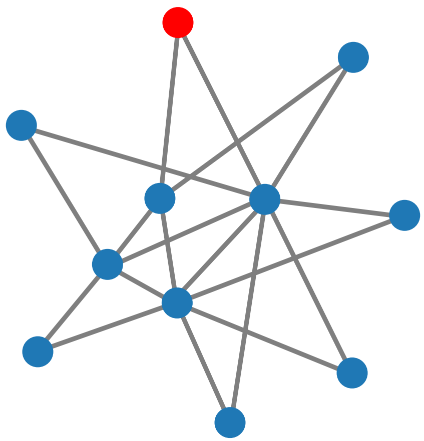


$m = 2$

$m_0 = 2$

Barabasi-Albert model

- *Example:*



$$m = 2$$

$$m_0 = 2$$

- The generative process of the Barabasi-Albert model is **autoregressive** and adds nodes to the graph one at a time.

Summary

- Graph generation
- Network Characteristics
  - Connectedness
  - Path length
  - Degree distribution
  - Clustering coefficient
- Graph generative models
  - Erdos-Renyi model
  - Configuration model
  - Stochastic Block model
  - Barabasi-Albert model