Machine Learning for Networks

Prof. Dr. Ingo Scholtes

Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

ingo.scholtes@uni-wuerzburg.de

Lecture 03
Generative Models and Statistical Inference

May 11, 2022

Notes:

- **Lecture L03:** Generative Models and Statistical Inference 11.05.2022
- **Educational objective:** We introduce probabilistic generative models for graphs and show how they can be used for inference and machine learning tasks.
  - Generative Models for Random Graphs
  - Degree-preserving Generative Models
  - Statistical Inference in Networks

- **Exercise sheet 02:** Random Graphs and Inference due 18.05.2022

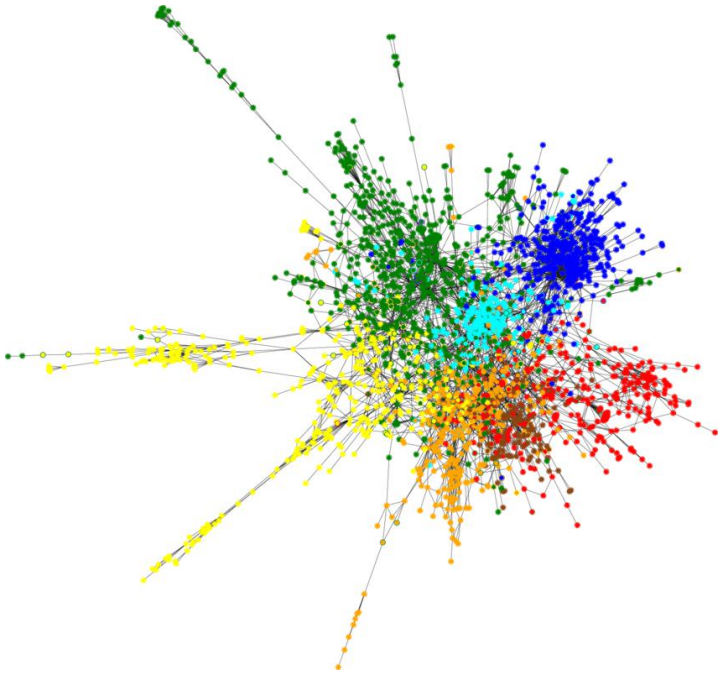
Motivation

- we introduced graph-theoretic methods to **detect communities in networks**

community detection techniques

- spectral clustering/minimal edge cuts → L02
- modularity maximization → SNA, L02

- cut-based approach is only **one perspective** on community detection in graphs
- challenge: we need to distinguish **community patterns** in a graph from **noise**
- requires **statistical methods** to infer communities in networks



scientific communities in CORA citation network

Notes:

- In the last lecture, we introduced graph-theoretic foundations and motivated the problem of community detection in networks. We considered a graph-theoretic, “cut perspective” on communities, which enabled us to translate community detection to the problem of finding minimal normalized edge cuts.
- We introduced the Laplacian matrix of undirected networks, and showed that we can use the eigenvector corresponding to the second-smallest eigenvalue to bisect the nodes in a network. This assignment provides us with an efficient, approximate solution of the normalized minimal edge cut.
- We further showed how we can use this theoretical result to implement a recursive spectral clustering algorithm for networks. This was the first algorithm of the course that can be used to address an unsupervised machine learning problem in complex networks. Those of you who attended the course *Statistical Network Analysis* will remember that we can also detect community structures by maximizing the modularity score of a partition. We do not repeat this here and instead refer to the script on WueCampus: → Lecture 02, Statistical Network Analysis
- The cut-based approach discussed in the previous lecture is only one perspective on community detection → cf. slide 13, Lecture 02 . In networks with complex patterns and highly heterogeneous degree distributions, a key challenge is to distinguish community patterns from noise, i.e. we not only need to identify groups of nodes that are “well-connected” to each other, but we also need to assess whether this connectivity is likely to result from chance. This requires statistical methods that help us assess the “statistical evidence” for community structures in networks.

Generative Models and Statistical Ensembles

probabilistic generative model

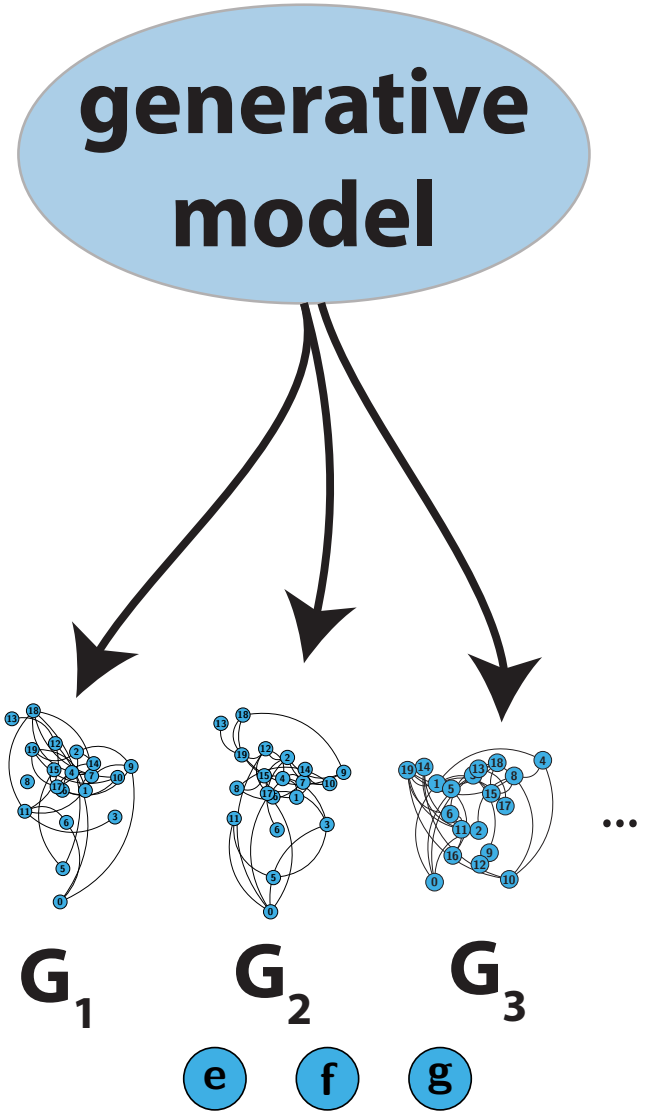
probabilistic generative model is a model that describes how a graph is generated. We can use it to generate samples of graphs (so-called **microstates**). Each graph is generated with a **microstate probability** that is given by the model.

statistical ensemble

probabilistic generative model with parameters Θ defines a **statistical ensemble** of networks, i.e. probability space with sample space $\Omega(\Theta)$ and probability measure P where

- $\Omega(\Omega)$ is the set of all graphs G consistent with Ω
- $P(G|\Theta) : G \to [0, 1]$ for each $G \in \Omega(X)$

Θ defines the **macrostate** of the ensemble. The graphs $G = (V, E) \in \Omega(\Theta)$ are called **microstates**. → cf. definition of statistical ensemble in L03, SNA



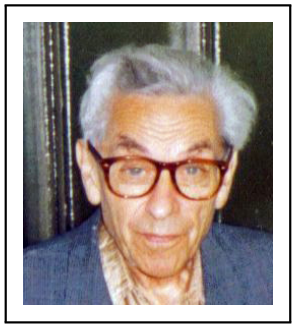
- probabilistic generative models are basis for **statistical inference in networks**

Notes:

- An important approach to statistical learning in networks is based on so-called **probabilistic generative models** of graphs. Loosely speaking, a generative model in machine learning is a statistical model that not only describes patterns in data, but that allows us to generate new data. In the context of complex networks, a probabilistic generative model allows us to generate samples of graphs, possibly based on a set of parameters that control the (probabilistic) graph generation process. In network science, such samples of graphs are often called **microstates**. An important aspect is that the model generates each of those microstates with a certain probability. We will see that we can use those microstate probabilities for statistical inference and learning tasks.
- Those of you, who attended our course *Statistical Network Analysis* remember that we extensively discussed **statistical ensembles of networks**, which we defined as a probability space where the sample space consists of the set of all graphs that are consistent with given macroscopic statistics (which we called the macrostate of the ensemble). We note that a probabilistic generative model defines a statistical ensemble, i.e. a probability space of microstates each generated with a certain probability, where the macrostate is given by the parameters Θ of the model.
- Complementing the perspective on complex networks discussed in our course *Statistical Network Analysis*, in this course we will show how statistical ensembles can be used for inference and learning tasks.

$G(n, m)$ random graph model

- consider maximally simple **generative model** for random graphs → P Erdős and A Rényi, 1959
- **model parameters**
  - number of nodes n
  - number of links m
- **generative procedure**
  - generate empty network with n nodes
  - add m links to pairs of nodes chosen uniformly at random
- **microstates**: all networks $G = (V, E)$ with $|V| = n$ and $|E| = m$
- for all microstates we have $\langle k \rangle = \frac{2m}{n}$
- probability $P(G)$ of a given microstate G?



Pál Erdős
1913 – 1996

Alfred Rényi
1921 – 1970

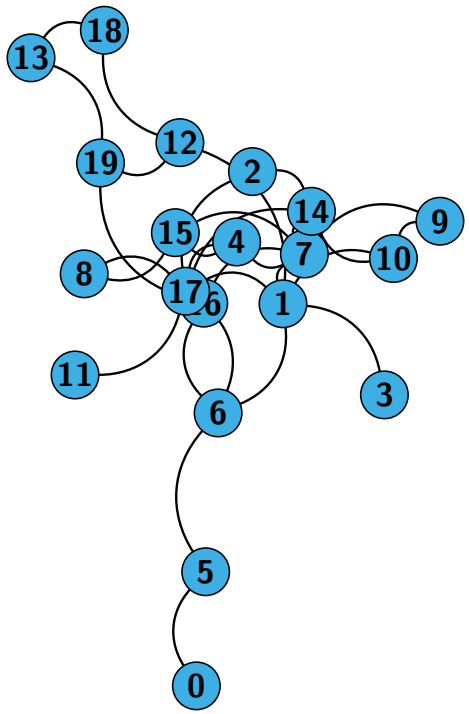


image credit: alchetron.com / fair use, user Kmhkmh, Wikimedia Commons, CC-BY-SA 2.0

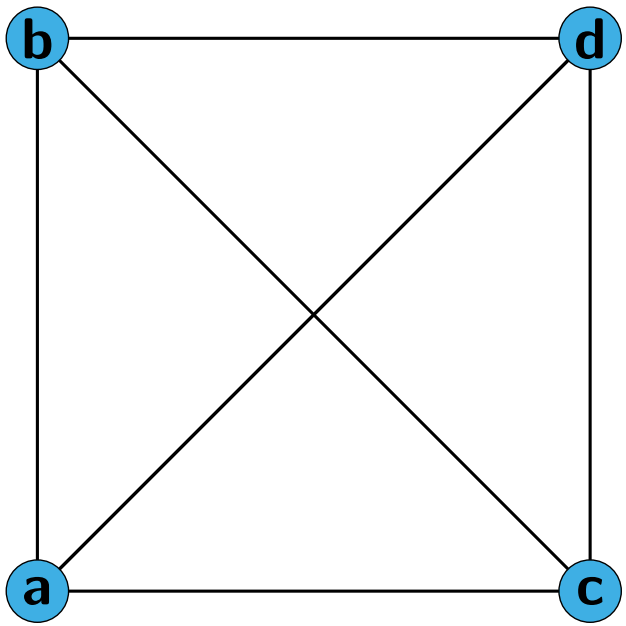
Notes:

- We start with one of the simplest possible generative models for random graphs, the so-called $G(n, m)$ **model**. The name refers to the two parameters n (number of nodes) and m (the number of edges) that control the network generation process. This model was introduced in 1959 by Pál Erdős and Alfréd Rényi, even though other researchers have studied random graph models before. The $G(n, m)$ assumes that the number of nodes and links of a network are fixed, while the topology of the network is randomly generated. The generative process works as follows:
- We start with an empty network with n nodes and distribute m links to pairs of nodes chosen uniformly at random.
- This is a standard urn problem in combinatorics, where the "urn" contains all possible links in a network with n nodes. From this urn, we draw m links uniformly at random (typically without replacement).
- To calculate the probability that a specific graph is generated, we need to know the size of the urn, i.e. how many links a network can maximally have. This depends on whether we study directed or undirected networks with or without self-loops. Hence, to make further statements about the microstate probabilities in this model we need a primer in (network) combinatorics.

Combinatorics primer: undirected networks

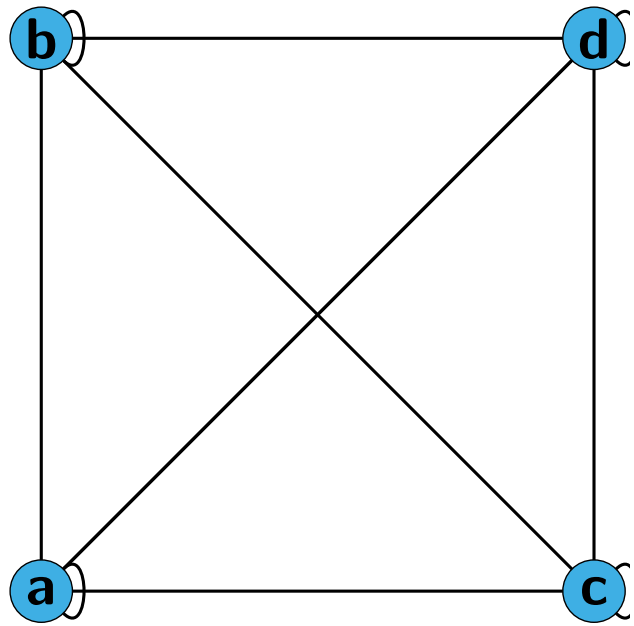
Maximum number of links in **undirected** network with n nodes?

without self-loops



$$\binom{n}{2} = \frac{n \cdot (n-1)}{2}$$

with self-loops



$$\binom{n+1}{2} = \frac{n \cdot (n+1)}{2}$$

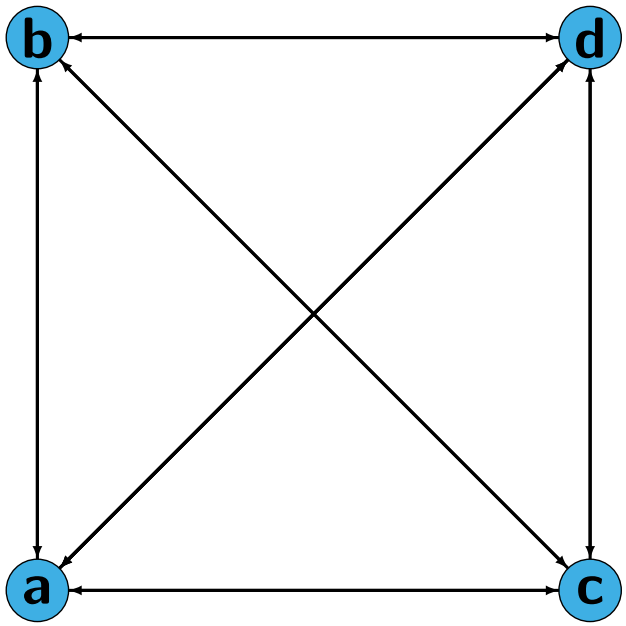
Notes:

- We first have to distinguish whether or not we allow self-loops, i.e. a link of a node to itself. For undirected networks drawing random links then corresponds to the following sampling processes:
- Without self-loops:
  - corresponds to a sampling of node pairs without ordering and without replacement
  - There are $\binom{n}{2}$ different combinations in which we can draw 2-element subsets from an urn with n elements
  - For an undirected network without self-loops and $n = 4$ we have: $\binom{n}{2} = \frac{12}{2} = 6$ possible links
- With self-loops:
  - corresponds to a sampling of node pairs without ordering but with replacement
  - For this process, the general formula is $\binom{n+k-1}{k}$, so for $k = 2$ we have the formula above
  - For an undirected network with self-loops and $n = 4$ we have: $\binom{n+1}{2} = \frac{20}{2} = 10$ possible links

Combinatorics primer: directed networks

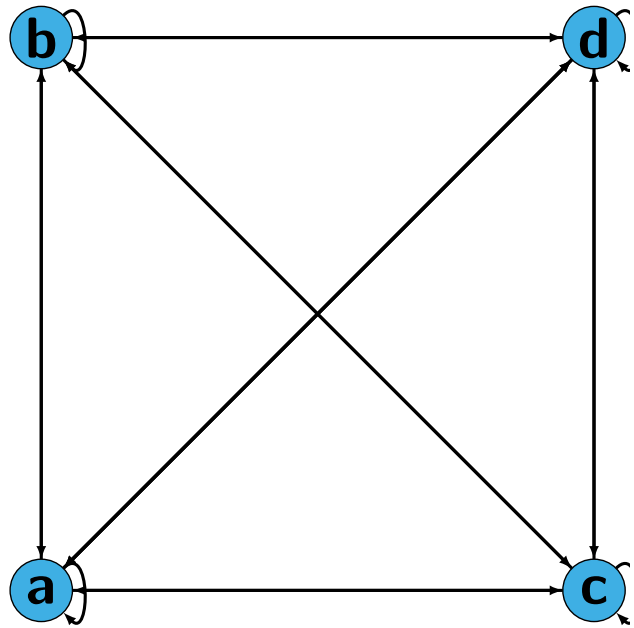
Maximum number of links in **directed** network with n nodes?

without self-loops



$$2 \cdot \binom{n}{2} = n \cdot (n-1)$$

with self-loops



$$n^2$$

Notes:

- For directed networks, we again distinguish whether we do or do not allow self-loops:
- Without self-loops
  - We consider a sampling process with ordering and without replacement
  - We can easily calculate this number based on the formula for undirected networks. Since each undirected link exists in two directions, we just multiply by this number by two
  - For a directed network without self-loops and $n = 4$ we have: $2 \cdot \binom{n}{2} = 12$ possible links
- With self-loops
  - Here we can simply consider all permutations (with repetitions) of length two, with elements chosen from a set of size n
  - In general, we have n^k so permutations of length k, so for pairs of n nodes we get n^2
  - For undirected networks with self-loops and $n = 4$ we thus have: $n^2 = 16$ possible links
- **Important:** in the following study of random graph models, we consider **undirected networks with self-loops**. However, using the discussion above it is easy to generalise this to other cases.

$G(n, m)$ model: microstate probability

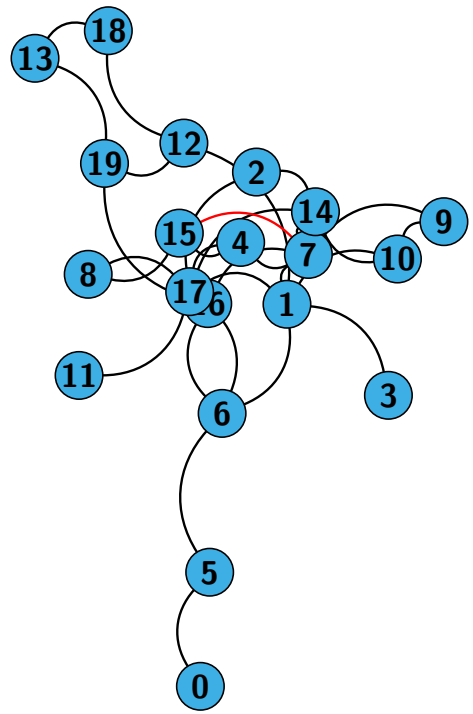
- consider $G(n, m)$ model for **undirected** networks **with** self-loops (no multi-edges)
- number of different microstates that are consistent with model parameters is

$$Z(n, m) := |\Omega(n, m)| = \binom{\binom{n+1}{2}}{m}$$

- in $G(n, m)$ model all **microstates are equiprobable**, i.e.

$$P(G) = \frac{1}{Z(n, m)} \text{ for all } G \in \Omega(n, m)$$

- what is the probability that a specific link (v, w) is generated?



random microstate generated using $G(n, m)$ model for $n = 20$ and $m = 30$

Notes:

- Knowing the number of possible links in a network with n nodes allows us to calculate the number of different microstates with n nodes and m links. For an undirected network with n nodes and self-loops, we have $\binom{n+1}{2}$ possible links. From this set of possible links, we chose m links at random without replacement (and we don't care about the order in which they are chosen). So the number of networks with n nodes and m links is $\binom{\binom{n+1}{2}}{m}$ (i.e. this is the size of our sample space $\Omega(n, m)$).
- Since we draw links uniformly at random, all microstates are generated with the same probability, so we have $P(G) = \frac{1}{Z(n,m)}$ where $Z(n, m) = \binom{\binom{n+1}{2}}{m}$ gives the size of the sample space $\Omega(n, m)$.
- In statistical mechanics, $Z(n, m)$ is called "partition function" or "Zustandssumme" (because it sums statistical weights over all possible states). The notation is borrowed from there, because statistical ensembles of nodes have been extensively studied by researchers in statistical physics.
- In other words: in the $G(n, m)$ model, the microstate is chosen uniformly at random from the space of all networks with n nodes and m edges. This assumes that, e.g. in a social network with n users and m friendships, each realization of the network is equally probable.
- There is one complication: consider the probability that a specific pair of nodes (v, w) is connected? Since we sample m links **without** replacement, links are actually not sampled independently, which makes it difficult to calculate this probability.

$G(n, p)$ random graph model

- consider different generative model for random graphs → E N Gilbert, 1959
  - create links **independently**
  - relax constraint of having **exactly** m links
- **model parameters**
  - number of nodes n
  - probability p to form links between pairs of nodes
- **microstates**: all networks with n nodes and **any number of links** (incl. self-loops)
- **expected number of links** $\langle m \rangle$ generated for a microstate G is

$$\langle m \rangle = p \cdot \binom{n+1}{2}$$

Edgar Nelson Gilbert

1923 – 2013

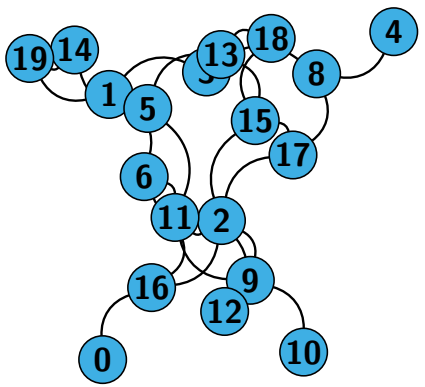


image credit: legacy.com, fair use

Notes:

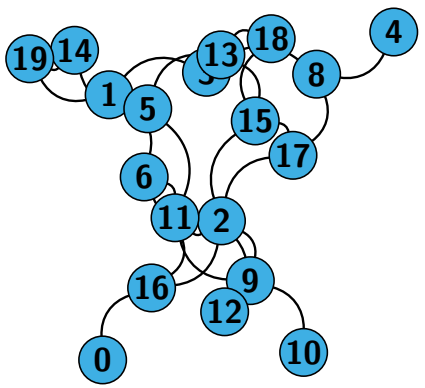
- The $G(n, p)$ model allows for a simpler mathematical treatment of the sampling process. The idea is to connect pairs of nodes independently from each other, by defining a single "link generation" probability p. In other words, for each pair of nodes, we perform a Bernoulli trial and generate a link upon the success of this trial (which happens with probability p). This simple generative model for random graphs was introduced in the 1959 paper "Random Graphs" by Edgar Nigel Gilbert.
- The sample space of the resulting statistical ensemble consists of all networks with exactly n nodes and any number of links. So, different from the $G(n, m)$ model, every network topology with n nodes is a microstate in the resulting sample space. Network realizations with different numbers of links have, in general, different probabilities. As an exercise, try to find a parameter p where this is not the case, i.e. where all microstates have the same probability (see self-study questions).
- If we allow self-loops, we have $\binom{n+1}{2}$ possible pairs of nodes. Since we make $\binom{n+1}{2}$ Bernoulli trials, where each trial yields a link with probability p, the expected number of links generated by the $G(n, p)$ model is $p \cdot \binom{n+1}{2}$.

$G(n, p)$ model: microstate probability

- **microstate probability** $P(G)$ depends on number of links in microstate
- for microstate G with n nodes and m_G links

$$P(G) = p^{m_G} \cdot (1-p)^{\binom{n+1}{2} - m_G}$$

- how does the microstate probability change if we change p?
- is there a value of p for which all microstates are equiprobable?



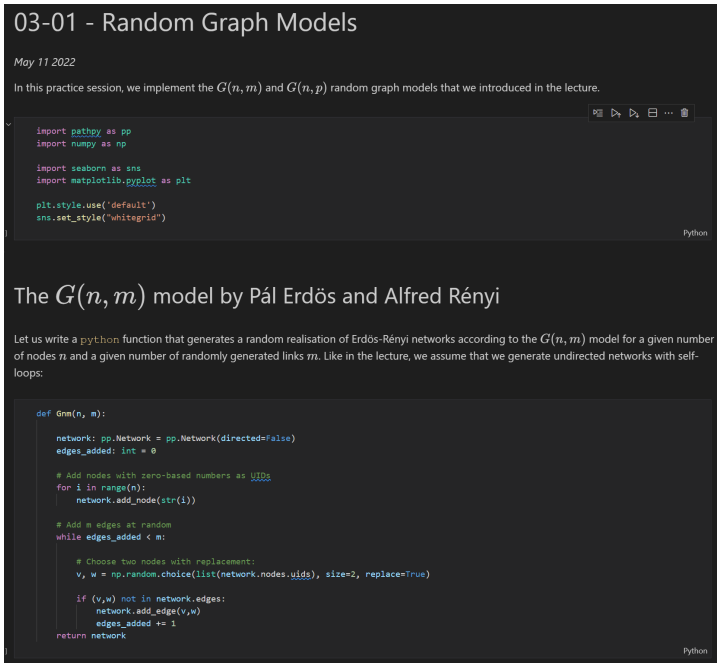
random microstate generated using $G(n, p)$ model with $n = 20$ and $p = 0.18$

Notes:

- In the $G(n, m)$ model, we could easily calculate the microstate probability based on (i) the known total number of microstates, and (ii) the fact that all microstates are equally likely. But what is the **probability of a microstate in the $G(n, p)$ model**?
- Here, not all microstates are equally likely because the probability to generate a particular microstate depends on the number of links of this microstate.
- What we can do is to calculate the probability of a microstate with exactly m links: this corresponds to having m successes in a sequence of $\binom{n+1}{2}$ Bernoulli trials, where each trial succeeds with probability p. $\binom{n+1}{2} - m$ of those Bernoulli trials are not successful (which happens with probability $1 - p$). This allows us to calculate the probability of a microstate with m links.
- Note that in the $G(n, p)$ model, the probabilities of all microstates that have the same number of links are the same (i.e. the model does not differentiate between different networks as long as they have the same number of links).

Practice session

- we implement **two probabilistic generative models** for random graphs
- we consider **empirical networks** and compare them to **random realizations**
- we inspect the **distribution of node degrees** of random graphs



03-01 - Random Graph Models

May 11 2022

In this practice session, we implement the $G(n,m)$ and $G(n,p)$ random graph models that we introduced in the lecture.

```
import pathpy as pp
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

plt.style.use('default')
sns.set_style("whitegrid")
```

Python

The $G(n,m)$ model by Pál Erdös and Alfred Rényi

Let us write a `python` function that generates a random realisation of Erdös-Rényi networks according to the $G(n,m)$ model for a given number of nodes n and a given number of randomly generated links m. Like in the lecture, we assume that we generate undirected networks with self-loops:

```
def Gnm(n, m):

    network: pp.Network = pp.Network(directed=False)
    edges_added: int = 0

    # Add nodes with zero-based numbers as UIDs
    for i in range(n):
        network.add_node(str(i))

    # Add m edges at random
    while edges_added < m:

        # Choose two nodes with replacement:
        v, w = np.random.choice(list(network.nodes.uids), size=2, replace=True)

        if (v,w) not in network.edges:
            network.add_edge(v,w)
            edges_added += 1
    return network
```

Python

practice session

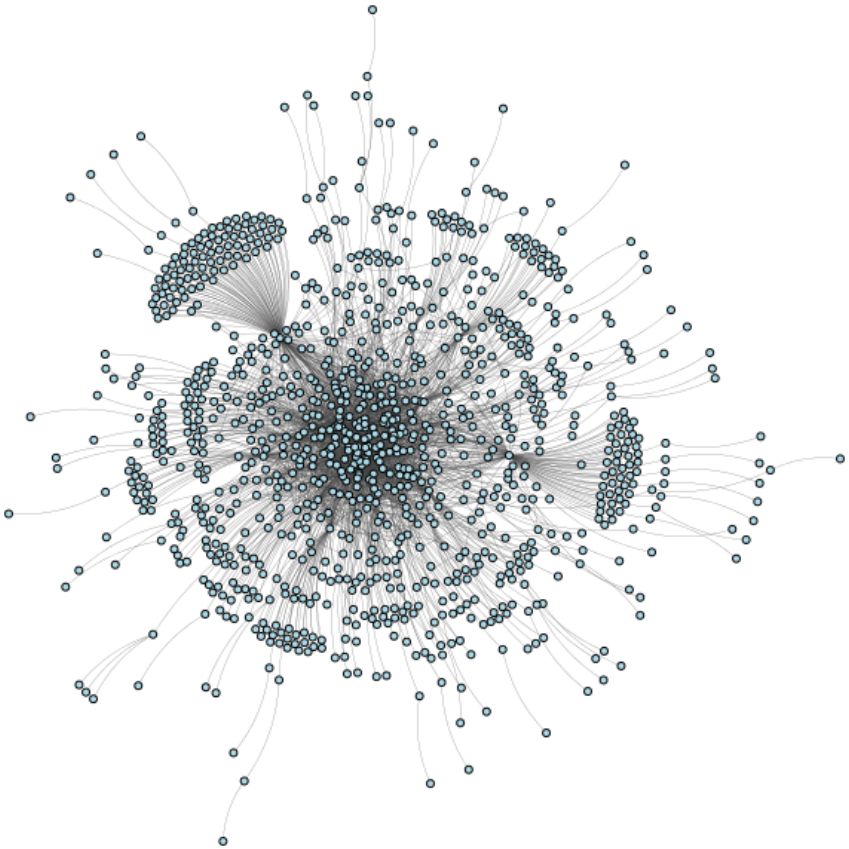
see notebooks 03-01 in `gitlab` repository at

→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2022_sose_ml4nets_notebooks`

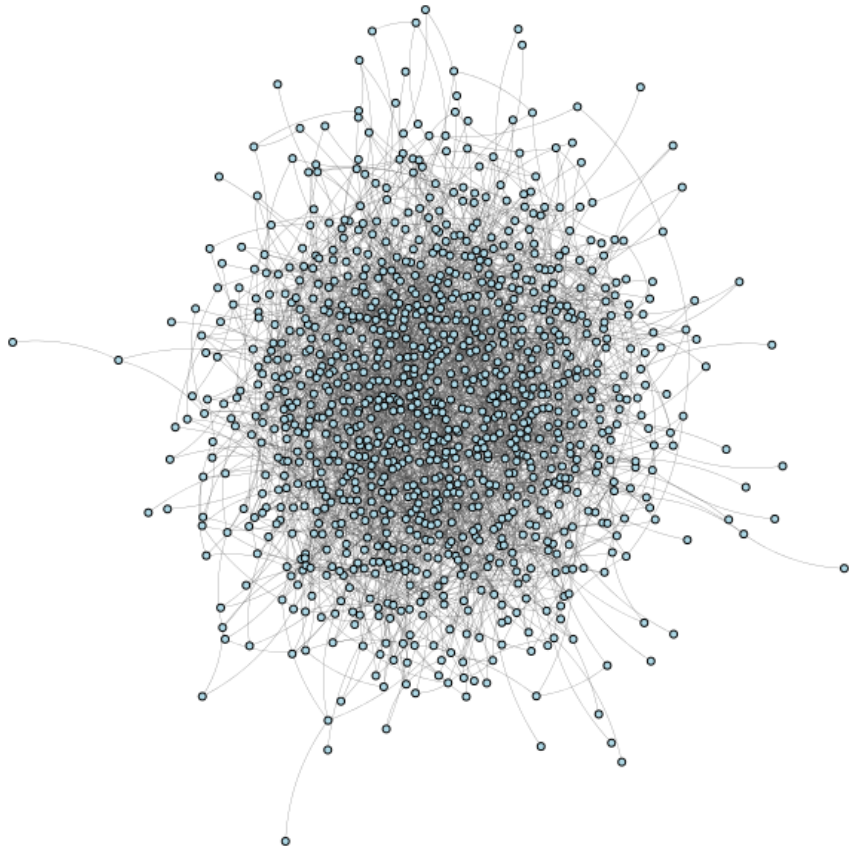
Notes:

- In the first practice session, we implement functions that generate random graphs according to the two probabilistic generative models introduced above.
- We will further use this model to generate “randomized” versions of empirical networks, and we compare the topologies of those randomized versions to the topologies of the empirical networks.

Random graphs vs. real-world networks



empirical collaboration network with n nodes and m links



random microstate generated by $G(n, m)$ model

can we generate random graphs that **preserve the degrees of nodes in an empirical network**?

Notes:

- In the practice session, we observe that the distribution of node degrees in empirical networks differs considerably from that of the random graphs generated by the two probabilistic generative models introduced above. This limits our ability to distinguish patterns from noise insofar as we cannot tell whether a pattern is due to the topology of links (i.e. to which other nodes a node is connected) or the distribution of node degrees (i.e. to how many other (random) nodes a node is connected).
- To address this issue, it is common to consider **degree-preserving** generative models, i.e. models that probabilistically generate graphs such that either the exact sequence or distribution of degrees of an empirical network is preserved.

Degree distribution

- **degree distribution** $P : \mathbb{N} \to [0, 1]$ of network $G = (V, E)$ is defined as

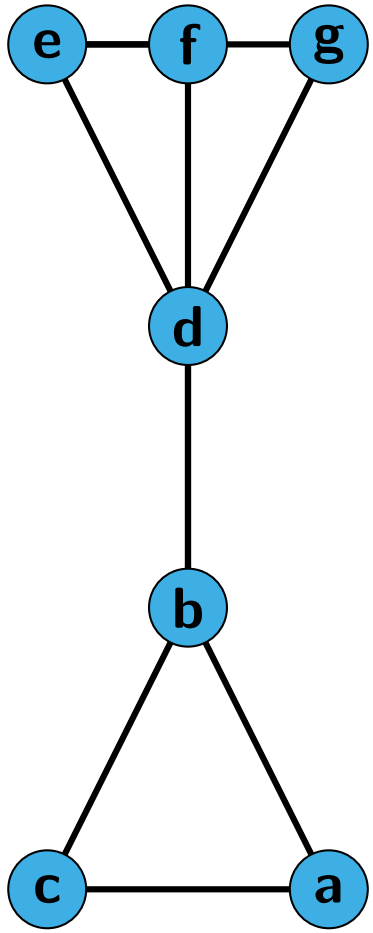
$$P(k) := \frac{N_k}{n}$$

$$\text{with } N_k := |\{i \in V : d_i = k\}|$$

$$\text{and } n := \sum_k N_k = |V|$$

- **mean degree** is defined as

$$\langle k \rangle := \sum_{k=0}^{\infty} k \cdot P(k) = \sum_{i \in V} \frac{d_i}{n} = \frac{2m}{n}$$



example

$$P(2) = \frac{4}{7}, P(3) = \frac{2}{7}, P(4) = \frac{1}{7}$$

$$\langle k \rangle \approx 2.57$$

Notes:

- Before we introduce a degree-preserving model for random graphs, we formally define the **degree distribution** of a network, which corresponds to the relative frequencies $P(k)$ of unique degrees node k. For each degree k we first calculate the number of nodes N_k with degree k and divide N_k by the number of nodes n in the network. Interpreted as a probability, $P(k)$ is the probability that a node chosen uniformly at random has a given degree k.
- We can use the degree distribution to calculate the **mean degree** in the network, which corresponds to the **expected degree** if we were to chose a node uniformly at random. The angle bracket notation $\langle k \rangle$ is again borrowed from statistical mechanics, where angle brackets commonly refer to average quantities of "statistical ensembles".
- In our course *Statistical Network Analysis*, we show that the degree distribution is a **simple yet powerful aggregate characterization of networks**. It can be used to make surprisingly strong analytical statements about expected network properties, such as diameter, average shortest path lengths, connectedness, component sizes, or robustness $\rightarrow$ cf. script Statistical Network Analysis

Degree distribution of random graphs

- consider **degree distribution** of random graph $G = (V, E)$ generated by $G(n, p)$ model
- for $v \in V$ chosen uniformly at random define **discrete random variable** $X : V \to \mathbb{N}^0$ with $X := d_v$
- to calculate probability mass function $P : \mathbb{N}^0 \to [0, 1]$ consider generative process
  - for node v we perform n **Bernoulli trials** with success probability p (i.e. incl. self-loops)
  - probability of exactly k successes given by **binomial distribution** $\text{Binom}(n, k)$
- **degree distribution** of random microstate is

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

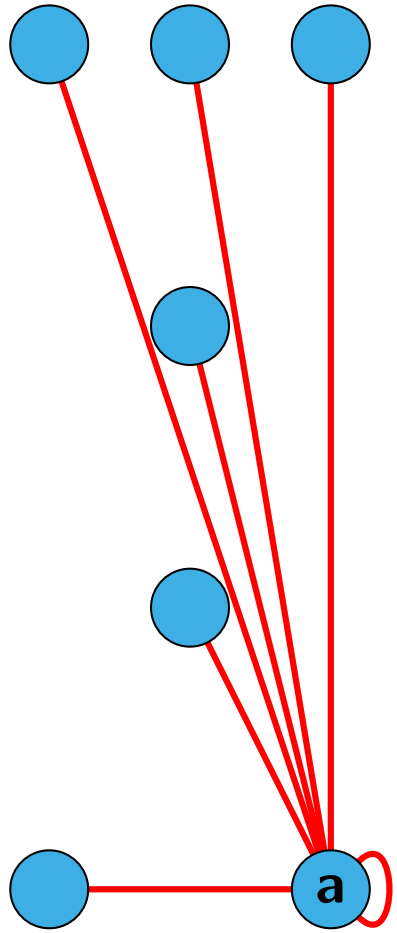


illustration of Bernoulli trials performed for node a in $G(n, p)$ model

Notes:

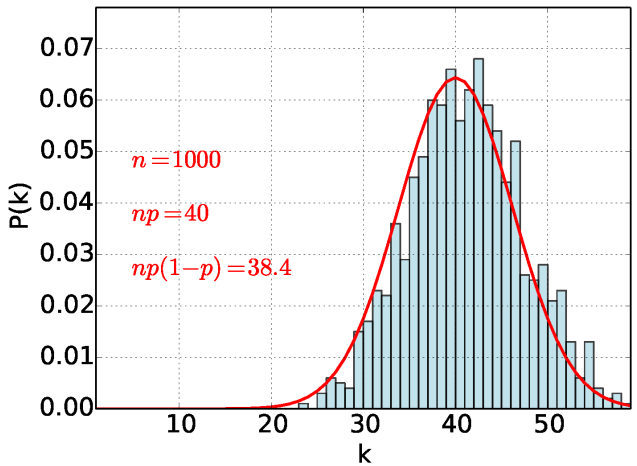
- What is the degree distribution in graphs generated by the generative models for random graphs? A simple combinatorial argument, and the interpretation of the $G(n, p)$ model as a repeated Bernoulli trial, allow us to calculate the **degree distribution of microstates generated by the $G(n, p)$ model**. Given parameters n and p we calculate the probability $P(k)$ that a random node in a randomly generated graph has degree k.
- For each of the n nodes, we perform n (with self-loops!) Bernoulli trials, where each trial yields a link with probability p. What is the probability to have exactly k successes in a sequence of n independent Bernoulli trials that succeed with probability p? This probability is given by the Binomial distribution, where the binomial coefficient captures the number of different ways how k successes can occur within the n trials, while the second and third factor capture the probability of k successes and $n - k$ "failures" respectively.
- Note that we obtain the Binomial distribution because we do not differentiate between different microstates, as long as we have k successes. I.e. we do not care to which nodes the k links are connected. This is why (different from the microstate probability above) we must include the binomial factor, which counts in how many different ways a node in a network with n can be connected to k nodes.

$G(n, p)$ model: limiting degree distribution

- for given number of n nodes, degree distribution of $G(n, p)$ microstate is

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

- what if n is large?



De Moivre-Laplace Theorem

for $n \to \infty$ the degree distribution of random $G(n, p)$ microstates converges to

$$P(k) \sim \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(np, np(1-p))$$

i.e. **normal distribution** with

$$P(k) = \frac{1}{\sqrt{np(1-p)}\sqrt{2\pi}} e^{-\frac{(k-np)^2}{2np(1-p)}}$$

central limit theorem for node degrees in random graphs

example

degree distribution of random microstate generated by $G(n, p)$ model with $n = 1000$ and $p = 0.04$

Notes:

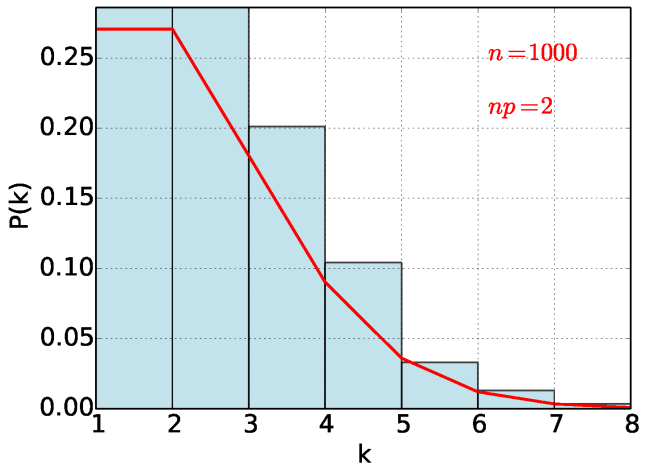
- What does the degree distribution of a network look like if the number of nodes is large? To answer this question it is helpful to study the **limiting degree distribution** as the number of nodes n goes to infinity. Let us consider this for the $G(n, p)$ model: For any network of finite size n, the degree distribution is a binomial distribution but can we approximate this for large n?
- For $n \to \infty$ we can use the *De Moive-Laplace theorem*, which follows as a special case from the *central limit theorem*. It states that for $n \to \infty$ the binomial distribution with success probability p converges to a normal distribution with mean np and variance $np(1-p)$. We confirm this in the example network shown on the right. We use the $G(n, p)$ model to generate a microstate with $n = 1000$ nodes and a link probability of $p = 0.04$ and plot the degree distribution of the microstate. The red curve shows a normal distribution with $np = \mu = 40$ and $np(1-p) = \sigma = 38.4$. Note that the normal distribution has a continuous domain, i.e. we have a probability distribution function rather than a probability mass function.
- The normal distribution closely approximates the degree distribution of a random network even for moderately large networks with a few hundred or a few thousand nodes. This tells us that we should not be surprised if we find a normal-shaped degree distribution in a large network. We expect this at random and all we need to explain it is a random link formation process. If we find a distribution that does not follow this expected shape, we need a different generative model if we want to preserve this property of an empirical network.

$G(n, p)$ model: sparse infinite networks

- degree distribution of $G(n, p)$ model

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

- what if n is large and $\langle k \rangle = np$ **is small**



Poisson limit theorem

for $n \to \infty$ and $p \to 0$ the degree distribution of random $G(n, p)$ microstates converges to

$$P(k) \sim \frac{\lambda^k e^{-\lambda}}{k!}$$

i.e. **Poisson distribution** with $\lambda = \langle k \rangle = np$

example

degree distribution of random microstate generated by $G(n, p)$ model with $n = 1000$ and $p = 0.002$

- $p \to 0 \Rightarrow np \equiv$ const, i.e. **sparse** network

Notes:

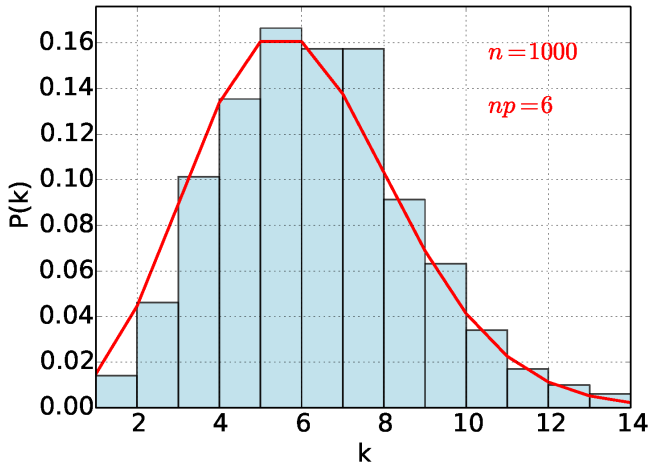
- The approximation of the degree distribution by a normal distribution assumes that the number of nodes n goes to infinity while the link probability p remains constant. If n goes to infinity, the expected degree np goes to infinity as well, i.e. the generated networks are becoming increasingly dense.
- We can instead consider **sparsely connected infinite random networks** with $n \to \infty$ but with a fixed expected degree $np \equiv const \ll n$. For $n \to \infty$ this implies $p \to 0$.
- Under these assumptions, the **Poisson limit theorem** states that the limiting degree distribution is a Poisson distribution with parameter $\lambda = np$. Hence, the degree distribution of sparse networks with a large number of nodes can be approximated by a Poisson distribution. We demonstrate this in the example shown on the right, which shows the degree distribution of a random microstate from the $G(n, p)$ model with $n = 1000$ and $np = 2$, i.e. $p = 0.002$. The red curve is the probability mass function of the limiting Poisson distribution. The limiting distribution closely approximates the empirical degree distribution, even for a moderately large network with one thousand nodes.
- Using Stirling's approximation $k! \approx \sqrt{2\pi k}\frac{k}{e}^k$ we easily see that the tail of a Poisson degree distribution decays faster than exponential, i.e. it is extremely unlikely to observe a node with degrees that are much larger than the expected degree np. This tells us that we do not expect to find "outliers" in terms of the degree in microstates that are generated using the $G(n, p)$ model.

$G(n, p)$ model: sparse infinite networks

- degree distribution of $G(n, p)$ model

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

- what if n is large and $\langle k \rangle = np$ **is small**



Poisson limit theorem

for $n \to \infty$ and $p \to 0$ the degree distribution of random $G(n, p)$ microstates converges to

$$P(k) \sim \frac{\lambda^k e^{-\lambda}}{k!}$$

i.e. **Poisson distribution** with $\lambda = \langle k \rangle = np$

example

degree distribution of random microstate generated by $G(n, p)$ model with $n = 1000$ and $p = 0.006$

- $p \to 0 \Rightarrow np \equiv$ const, i.e. **sparse** network

Notes:

- The example on the right shows the degree distribution of a $G(n, p)$ microstate with $n = 1000$ and $p = 0.006$, as well as the probability mass function of the Poisson distribution with $\lambda = np = 6$.
- This shows that, as we increase the expected degree np, for sufficiently large n the Poissonian degree distribution becomes increasingly similar to a Normal distribution.

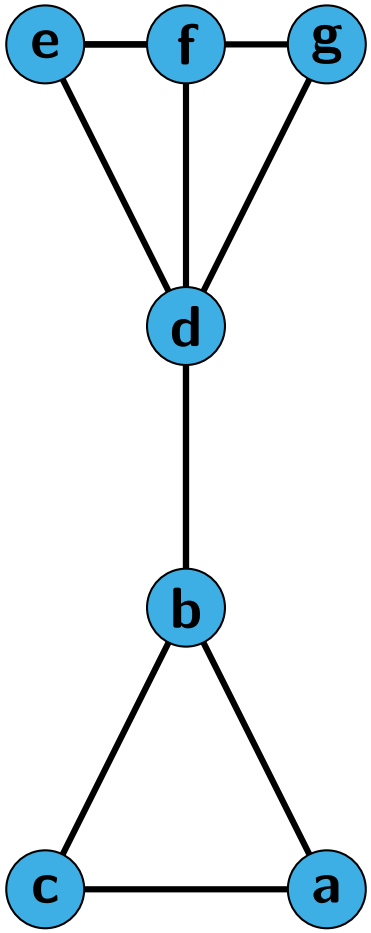
Degree sequence

- **degree sequence** of network $G = (V, E)$ is

$$S = (d_v)_{v \in V} \in \mathbb{N}^n$$

- which of the following are **degree sequences**?
  - $S = (4)$
  - $S = (1, 2, 2, 0, 2)$
  - $S = (1, 2, 2, 1, 2)$

- we call sequence $S \in \mathbb{N}^n$ **graphic** iff
  $\exists$ graph $G = (V, E) : S(G) = S$



example

$$S(G) = (2, 3, 2, 4, 2, 3, 2)$$

Notes:

- We have seen that – depending on the model parameters n and p – the degrees of nodes in a random graph generated by the $G(n, p)$ model follow a Poisson or Normal distribution. What if we want to generate random graphs with a different degree distribution? If we start with an empirical network, we can use a model that preserves the **degree sequence**, i.e. the (ordered) sequence of node degrees in a network. In the example above, we assume that nodes are in alphabetical order, i.e. the ordered sequence captures degrees of nodes (a, b, c, d, e, f, g).
- Although this degree sequence contains information on the network topology, its size for a network with n nodes is only n. You can view this as a **compression** of a network into a (smaller) number sequence. This is a lossy compression because there are – in general – many different network topologies that are consistent with the same degree sequence.
- Note that not every sequence of n integer numbers is also a valid degree sequence of a network with n nodes. In the three examples above, only the third is a valid degree sequence. We call number sequences for which at least one graph with that degree sequence exists **graphic**. From the examples above we immediately see that graphic sequences must have an even sum and the second sequence is thus not graphic. Also, the sum of the degrees in a graphic sequence is bounded by two times the maximum number of edges (which depends on the nodes). The first sequence is thus not graphic. As an exercise, try to construct sequence of numbers that are not graphic. Which properties must a sequence of n numbers minimally fulfil to be a graphic sequence?

Random graphs with given degrees

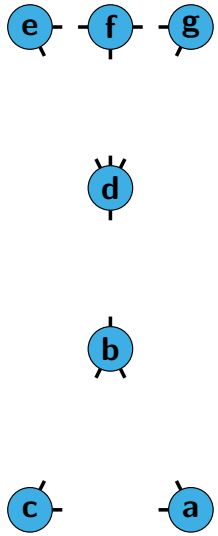
- **model parameters:** degree sequence $S = (d_i)_{i \in V}$
- **microstates:** all networks with degree sequence S
- **configuration model** generates equiprobable microstates G with

$$P(G) = \frac{1}{Z(S)}$$

where $Z(S)$ is the number of networks with degree sequence S

- we call this the **Molloy-Reed** model

→ M Molloy & B Reed, 1995



ensemble with **macrostate**
$S = (2, 3, 2, 4, 2, 3, 2)$

Michael Molloy

born ???

Bruce Reed

born 1962

image credit: left: University of Toronto Scarborough, fair use / right: David Eppstein, Wikimedia Commons, CC-BY-SA

Notes:

- Let us now consider a simple probabilistic generative model that generates an ensemble of all networks with the same degree sequence, i.e. the **macrostate** fixes the degree of each of the n nodes, which can be described by a vector S of n integer numbers. The microstates are all possible networks that correspond to the given degree sequence.
- We assume a **generative model that assigns equal probabilities to each of those microstates**, i.e. we have a discrete uniform probability mass function with a normalization constant $Z(S)$ that corresponds to the number of networks with a given degree sequence. Calculating the number of such networks is an interesting combinatorial problem, which has been addressed in → EA Bender and ER Canfield, 1978 . With this, we can again treat the network generation as an urn problem, i.e. we draw a network with a given degree sequence uniformly at random.
- An algorithmic formulation of this so-called **configuration model** for networks with a fixed degree sequence was presented in → M Molloy and B Reed, 1995 . For this reason, the configuration model is also often called the **Molloy-Reed model**. Just like for the $G(n, m)$ model, each microstate in this ensemble has the same probability, i.e. $P(G) = \frac{1}{Z(S)}$ where $Z(S)$ is the number of graphs with n nodes and degree sequence S.

Random graphs with given degrees

- we can relax constraint that generated graphs exhibit exact **degree sequence**
- **model parameters:** degree distribution $P(k)$ and number of nodes n
- **microstates:** networks with n nodes and **any** degree sequence
- all microstates with same degree sequence are equiprobable

configuration algorithm A

```
def configuration_A(S):
  """configuration model with degree sequence S"""
  stubs = []
  for i in range(len(S)):
    for j in range(S[i]):
      stubs.append(i)

  n := empty network with len(S) nodes
  while len(stubs)>0:
    sample v, w from stubs
    n.add_edge(v,w)
    remove v,w from stubs

  return n
```

configuration algorithm B

```
def configuration_B(n, P):
  """configuration model with degree dist.  P"""
  S := [1]
  while not is_graphic(S):
    S := sample n degrees from P

  return configuration_A(S)
```

Notes:

- Just like the $G(n, m)$ and the $G(n, p)$ models for random graphs, the configuration model comes in two variants. While the $G(n, m)$ model assumes that all networks have exactly the same number of links, the $G(n, p)$ model relaxes this constraint, instead fixing the expected number of links in the network. For degree-preserving models we use the same idea: rather than fixing the degree sequence (i.e. the specific degree of each node), we can **fix the degree distribution**, i.e. we fix the probability that a node with degree k is generated. The specific degree sequence can vary across different microstates.
- How can we implement both models (i.e. the model generating networks with fixed degree sequence and the one generated networks with fixed degree distribution)? For the model generating networks with fixed degree sequence we start from a network where we add link stubs to each node. We then add those stubs to a list from which we draw pairs uniformly at random without replacement. This resembles the construction procedure of the $G(n, m)$ model.
- We can implement the model for networks with fixed degree distribution as follows: we first draw a degree sequence of length n from the desired degree distribution. We then apply the previous algorithm to the sequence if the sequence is graphic. Like in the $G(n, p)$ model, the resulting ensemble includes all microstates with n nodes and any number of links (and any degree sequence). Depending on their degree sequence, some of these microstates are more likely than others. Microstates with the same degree sequence have the same microstate probability.

Practice session

- we explore necessary conditions for **graphic degree sequences**
- we use `pathpy` to implement the **Molloy-Reed model configuration model**
- we use the Molloy-Reed model to **generate random graphs that preserve degree sequences** of empirical networks

03-03 - Molloy-Reed Configuration Model

May 11 2022

In this notebook, we implement the configuration model after Molloy and Reed, which can be used to generate random networks with arbitrary degree distributions.

```python
import pathpy as pp
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

plt.style.use('default')
sns.set_style("whitegrid")
```

Python

```python
def MolloyReed(degrees):
    # assume that we are given a graphical degree sequence
    if not pp.generators.is_graphic_Erdos_Gallai(degrees):
        return None

    # create empty network with n nodes
    n = len(degrees)
    g = pp.Network(directed = False)

    # generate link stubs based on degree sequence
    stubs = []
    for i in range(n):
        for k in range(degrees[i]):
            stubs.append(str(i))

    # connect randomly chosen pairs of link stubs
    # note: if relax is True, we allow self-loops and multi-edges,
    # but remove them when we return the network.
    # This implies that the returned network may not have
    # exactly sum(degrees)/2 links.
    #print(stubs)
    while(len(stubs)>0):
        v, w = np.random.choice(stubs, 2, replace=False)
        if v!=w and ((v,w) not in g.edges):
            #print(v,w)
            stubs.remove(v)
            stubs.remove(w)
            g.add_edge(str(v),str(w))

    return g
```

Python

practice session

see notebooks 03-02 – 03-03 in `gitlab` repository at

→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2022_sose_ml4nets_notebooks`

Notes:

- In the second practice session of this week, we implement functions that generate random graphs according to the two variants of the degree-preserving Molly-Reed model. We use this model to generate random graphs that preserve the degree sequences of empirical networks.

Generative models and likelihood

- probabilistic generative models are basis for **statistical inference**

statistical inference

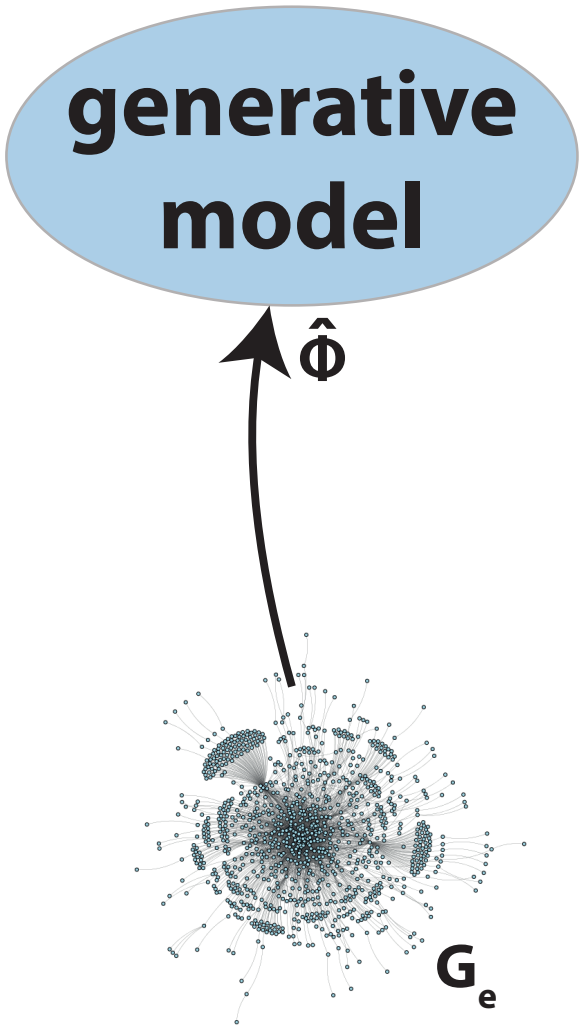
statistical inference refers to the process of **fitting and selecting a statistical model based on empirical data**. Statistical inference is one approach to address the training/learning step in machine learning.

model likelihood

for a given sample space Ω and outcome X we call

$$\mathcal{L}(\Theta|X) := P(X|\Theta)$$

the likelihood of model $X \sim P_\Theta$ with parameters Θ



- we interpret likelihood $\mathcal{L}(\Theta|X)$ as **plausability of model with parameters Θ** given observation X

Notes:

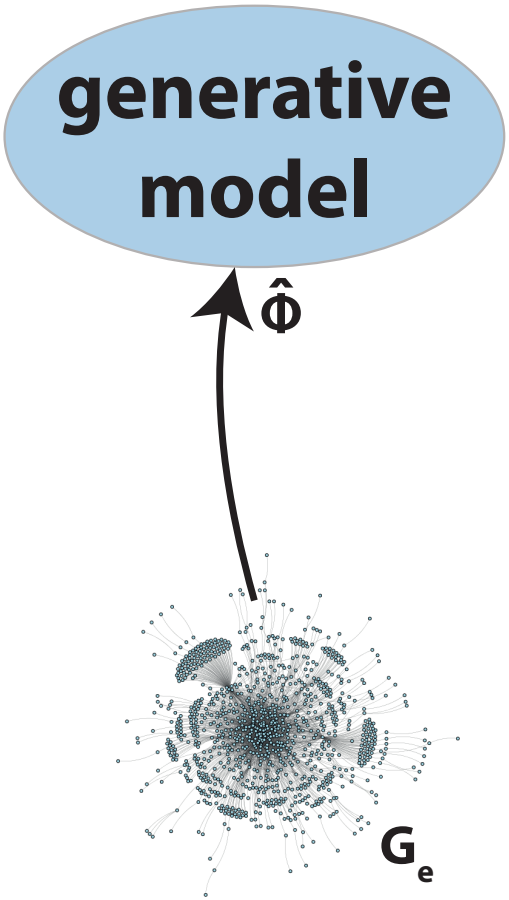
- So far, we used probabilistic generative models to generate random graphs, or to study the probability space of graphs generated by a given model. Different from the use of models and ensembles in our course *Statistical Network Analysis*, here we will take a statistical learning perspective. More precisely, we will use probabilistic generative models as basis for **statistical inference**. In general, statistical inference refers to the process of fitting (and selecting) statistical models based on empirical data. That is, for a given observation we are interested in the parameters of a model (or the specific model from a set of candidate models) that are "most plausible" given the observation. In other words, rather than using a generative model to reason about the (expected) properties of randomly generated microstates, we start from an empirical network and reason about the "most plausible" parameters of a given generative model.
- How can we assess the plausibility of a model for a given empirical network? We can use the fact that a probabilistic generative model generates graphs with different probabilities. Let $P(X|\Theta)$ denote the probability that a model generates an outcome X for parameters Θ. Somehow "inverting" the interpretation of this probability, we call this probability the **likelihood of the model with parameters** Θ given the observation X. This concept of likelihood was first studied by RA Fisher in the early 20th century, see → J Aldrich, 1997 .
- For a general introduction to likelihoods, statistical inference and statistical learning, we recommend to check the chapter "Foundations" in → KP Murphy: Probabilistic Machine Learning: An Introduction, 2022

Statistical inference in networks

- consider **generative model** with parameters Θ
- consider **empirical network** G_e
- we are interested in **most plausible model parameters** given G_e
- we can use **likelihood function**, i.e.

$$\hat{\Theta} := \arg\max_{\Theta} \mathcal{L}(\Theta)$$

- we perform **maximum likelihood estimation** of model parameters



Notes:

- The likelihood of a model with parameters Θ is the basis for the **frequentist perspective on statistical inference**. For this, let us consider a generative model with parameters Θ that generates networks with certain probabilities. As we shall see in the next lecture, different parameter values could encode different patterns, which we can use to reason about community structures. We further consider an empirical network G_e. We are now interested to find the "most plausible" value for the parameter Θ, which we assume to be the value that generates the observed network with maximum probability. We thus choose the parameter $\hat{\Theta}$ for which the likelihood function $\mathcal{L}(\Theta)$ is maximal.
- Note that this maximum is not necessarily unique, i.e. the arg max function may actually return a set of values $\hat{\Theta}$ from which we can choose.
- We obtain a function in the parameter Θ that we need to maximize. We can use established tools of mathematical optimization (analytical as well as computational techniques). For differentiable likelihood functions with a continuous parameter Θ we can use differential calculus to analytically calculate all maxima.
- Note: that this frequentist approach to inference is only one perspective. Using a **Bayesian approach**, we would consider the posterior probability of model parameters $P(\Theta|G_e)$ rather than the model likelihood. This posterior probability can be calculated using Bayes' theorem $P(\Theta|G_e) = \frac{P(G_e|\Theta)\cdot P(\Theta)}{P(G_e)}$, which relates it to the prior probability and the probability of the outcome G_e. More on this Bayesian perspective on inference in a later course.

Example: Inference in $G(n, p)$ model

- consider $G(n, p)$ **model for Erdös-Rényi random graphs**

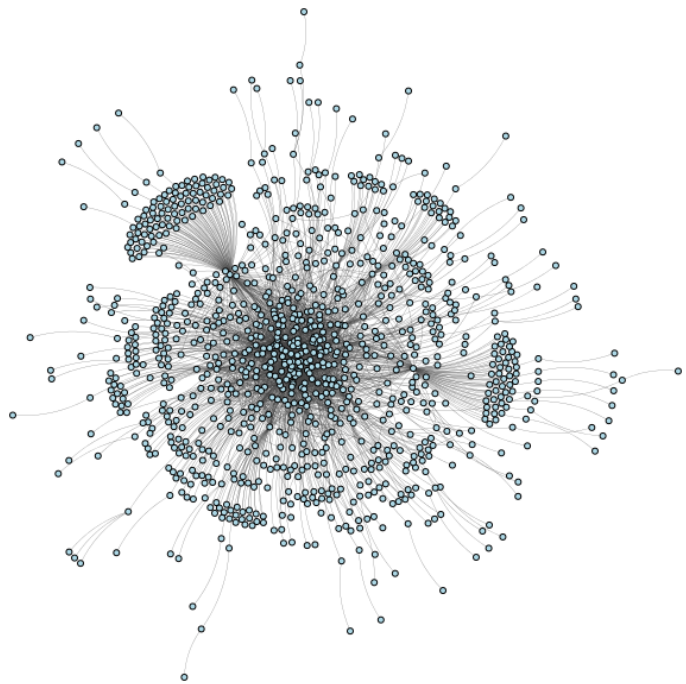
- for microstate G with n nodes and m links, microstate probability is given as

$$P(G) = p^m \cdot (1 - p)^{\binom{n}{2} - m}$$

- for given n and m, **likelihood function** of $G(n, p)$ model is thus

$$\mathcal{L}(p) = p^m \cdot (1 - p)^{\binom{n}{2} - m}$$

- which value p is most plausible given G_e?



example

GENTOO collaboration network with $n = 1071$ nodes and $m = 2949$ links

Notes:

- Before moving to more insightful applications of maximum likelihood estimation, let us illustrate statistical inference in an admittedly trivial example. We consider an empirical network such as the GENTOO collaboration network shown on the right. As a generative model, we consider the $G(n, p)$ model, where the parameter n corresponds to the number of nodes in the empirical network. We are left with a single model parameter $p \in [0, 1]$, whose "most plausible" value (given the observed network) we want to infer. We can rule out the values $p = 0$ and $p = 1$, which generate a single microstate, the empty and the full graph respectively.
- In this case, we can intuitively answer the question which parameter p is most plausible given the empirical network G_e. Based on the empirical mean degree $\langle k \rangle \approx 5.507 = np$, we can calculate $\hat{p} = \frac{np}{n} = \frac{5.507}{1071} \approx 0.00514$. That is, we choose the point in the parameter space at which we **expect** a random realization to have the same number of links as the observed network.
- For our example, this intuitively gives the most plausible parameter and we will see that this is equivalent to maximizing the likelihood (because in this simple case the likelihood has a single maximum precisely at the expected value). However, for more complex multi-modal distributions this is not necessarily the case.

Likelihood function of $G(n, p)$ model

- for empirical network G_e with n nodes and m links, **likelihood function of $G(n, p)$ model** is

$$\mathcal{L}(p) = p^m \cdot (1-p)^{\binom{n}{2}-m}$$

- for $n = 1071$ and $m = 2949$ we have

$$\mathcal{L}(p) = p^{2949} \cdot (1-p)^{570039}$$

- we derive the likelihood function by p

$$\mathcal{L}'(p) = 2949 \cdot p^{2948} \cdot (1-p)^{570039} - p^{2949} \cdot 570039 \cdot (1-p)^{570038}$$

- we solve equation

$$\mathcal{L}'(p) = 0$$

i.e.

$$2949 \cdot p^{2948} \cdot (1-p)^{570039} = p^{2949} \cdot 570039 \cdot (1-p)^{570038}$$

Notes:

- Let us see how we can find the same value for $\hat{p}$ based on the maximization of the likelihood function of the model. For a fixed n, the likelihood function of the $G(n, p)$ model is given by the microstate probability known from previous slides.
- We first replace the known number of n nodes and m links in our observation and obtain $\binom{n}{2} - m\binom{1071}{2} - 2949 = 570039$. This yields the likelihood function for the empirical network with a single free parameter p.
- In this simple case, the probability mass function of the microstates can only have a single maximum. Also, we have a continuous function on a closed interval, which implies that a local extremum is also a global extremum. Hence, to find the maximum point, we can check for which parameter p the first derivative is zero.
- We calculate the derivative of the likelihood function by using the product rule $(f \cdot g)' = f' \cdot g + f \cdot g'$
- We then need to solve the equation $\mathcal{L}'(p) = 0$ which, due to the exponents, is actually quite cumbersome.
- How can we simplify this task?

Log-likelihood function

- we often consider the **log-likelihood** function $\log \mathcal{L}(p)$
- logarithm is **monotonic transformation**, i.e.

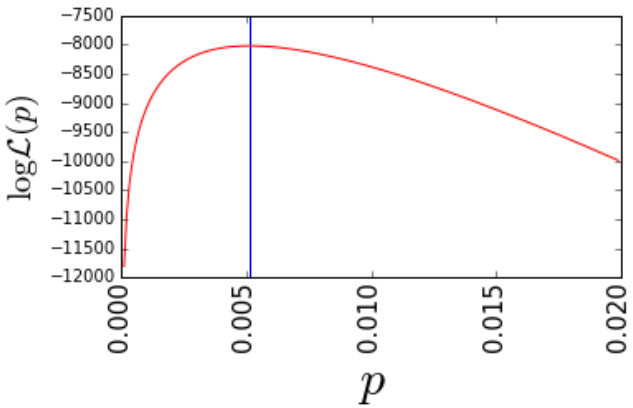
$$\arg\max_{\Theta} \mathcal{L}(\Theta) = \arg\max_{\Theta} \log \mathcal{L}(\Theta)$$

example

$$\log \mathcal{L}(p) = 2949 \log p + 570039 \log(1-p)$$

$$\frac{d \log \mathcal{L}(p)}{dp} = \frac{2949}{p} - \frac{570039}{1-p} = 0$$

$$\leftrightarrow \frac{2949}{p} = \frac{570039}{1-p}$$



log-likelihood function for $G(n, p)$ model of empirical collaboration network

- **maximum likelihood estimation** yields

$$\hat{p} \approx 0.00514 = \frac{\langle k \rangle}{n}$$

Notes:

- By taking the logarithm of the likelihood function we can turn the exponents into factors. Since the logarithm is a monotonic transformation this does not affect the maximum, i.e. we can take the derivative of the log-likelihood function instead.
- For the $G(n, p)$ model and $n = 1071$, $m = 2949$ we have:

$$\log \mathcal{L}(p) = 2949 \cdot \log p + 570039 \cdot \log(1 - p)$$

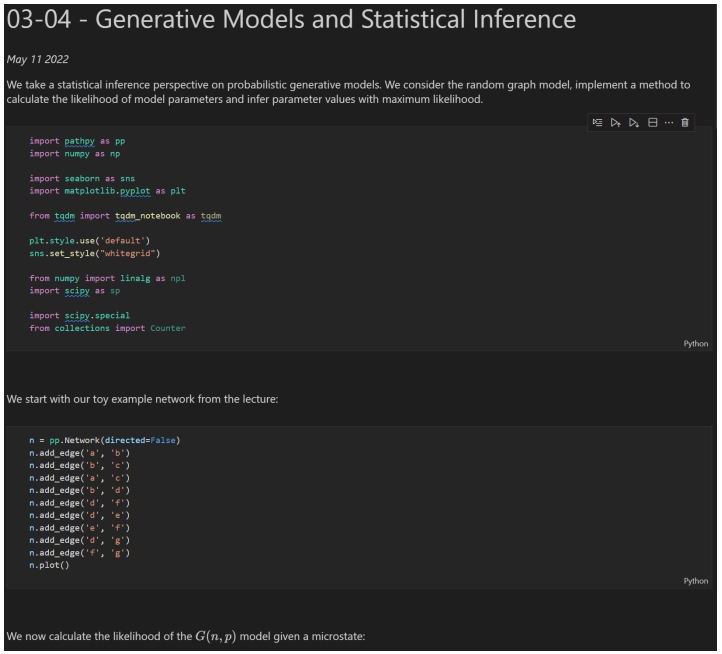
- The first derivative of this log-likelihood function is

$$\frac{d \log \mathcal{L}(p)}{dp} = \frac{2949}{p} - \frac{570039}{1 - p}$$

- By setting the derivative to zero, we get $\frac{2949}{p} = \frac{570039}{1-p}$
- As expected, this solution coincides with our estimate $\hat{p} = \frac{\langle k \rangle}{n}$. The plot on the right shows the values of the log-likelihood function close to the optimum value $\hat{p}$. Note the log-scale y-axis. The maximum likelihood value in the range of 10^{-8000} (we used the logarithm with base 10). This value is extremely small because it is the probability of generating one particular microstate with 1071 nodes and 2949 links (of which there are many different ones).

Practice session

- we implement functions to calculate the **likelihood of the $G(n, p)$ model** for a given empirical network
- we use the **log-likelihood function** to infer model parameters
- we compare microstate probabilities of different networks and calculate the **degree-based likelihood** of the $G(n, p)$ model



03-04 - Generative Models and Statistical Inference

May 11 2022

We take a statistical inference perspective on probabilistic generative models. We consider the random graph model, implement a method to calculate the likelihood of model parameters and infer parameter values with maximum likelihood.

```
import pathpy as pp
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

from tqdm import tqdm_notebook as tqdm

plt.style.use('default')
sns.set_style("whitegrid")

from numpy import linalg as npl
import scipy as sp

import scipy.special
from collections import Counter
```

Python

We start with our toy example network from the lecture:

```
n = pp.Network(directed=False)
n.add_edge('a', 'b')
n.add_edge('b', 'c')
n.add_edge('a', 'c')
n.add_edge('b', 'd')
n.add_edge('d', 'f')
n.add_edge('d', 'e')
n.add_edge('e', 'f')
n.add_edge('d', 'g')
n.add_edge('f', 'g')
n.plot()
```

Python

We now calculate the likelihood of the $G(n,p)$ model given a microstate:

practice session

see notebook 03-04 in `gitlab` repository at

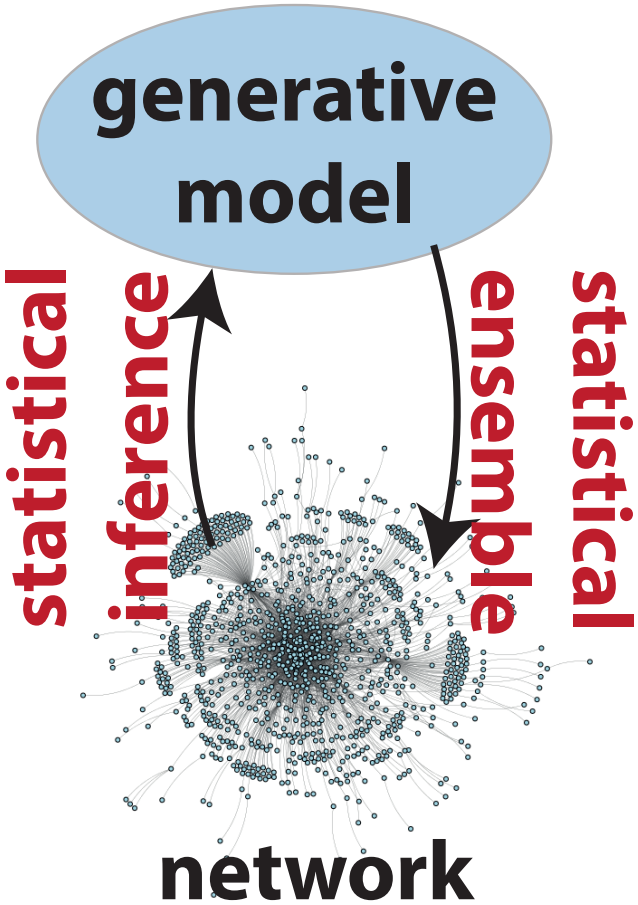
→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2022_sose_ml4nets_notebooks`

Notes:

- In the third practice session of this week, we implement a simple example for statistical inference in random graph models and demonstrate it in some example networks.

In summary ...

- we introduced basic **probabilistic generative models** for random graphs
- generative models for random graphs are foundation for **random graph theory, random graph theory, and statistical inference in graphs**
- we can use model likelihood to **infer most plausible model** given an empirical network
- suitably designed generative models help us to **detect community patterns in networks**

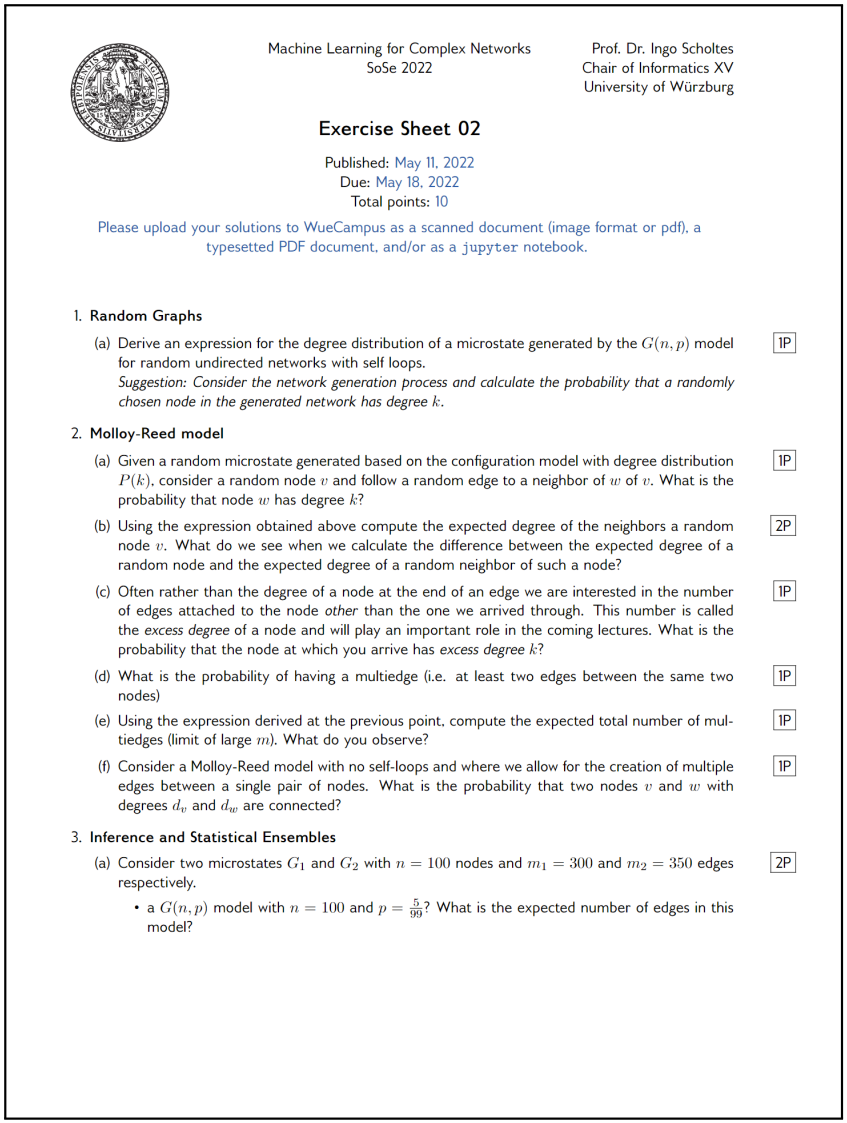


Notes:

- In summary, we introduced basic probabilistic generative models that are widely used in network analysis, random graph theory, and graph learning. Conceptually, such models can help us to distinguish which patterns are due to the specific topology of a graph (i.e. which nodes are connected to which other nodes) and which patterns are due to other aspects like, e.g. the number of nodes and links or the distribution of node degrees.
- As we shall see in the next lecture, generative models are an important basis for statistical learning techniques that help us to detect community structures in networks. For this, we will generalize the models introduced today such that different model parameters encode different community structures. With this, the detection of communities can be reinterpreted as a search for the model parameters (i.e. community structures) that are most plausible given the topology of the network.

Exercise sheet 02

- **second exercise sheet** will be released today
  - explore generative models
  - study statistical ensembles
  - take first steps in model inference
- solutions are due **May 18th** (via WueCampus)
- solutions will be discussed in week 5
- present your solution to earn bonus points



Machine Learning for Complex Networks
SoSe 2022

Prof. Dr. Ingo Scholtes
Chair of Informatics XV
University of Würzburg

Exercise Sheet 02

Published: May 11, 2022
Due: May 18, 2022
Total points: 10

Please upload your solutions to WueCampus as a scanned document (image format or pdf), a typesetted PDF document, and/or as a `jupyter` notebook.

1. **Random Graphs**
   (a) Derive an expression for the degree distribution of a microstate generated by the $G(n,p)$ model for random undirected networks with self loops. [1P]
   Suggestion: Consider the network generation process and calculate the probability that a randomly chosen node in the generated network has degree k.

2. **Molloy-Reed model**
   (a) Given a random microstate generated based on the configuration model with degree distribution $P(k)$, consider a random node v and follow a random edge to a neighbor of w of v. What is the probability that node w has degree k? [1P]
   (b) Using the expression obtained above compute the expected degree of the neighbors a random node v. What do we see when we calculate the difference between the expected degree of a random node and the expected degree of a random neighbor of such a node? [2P]
   (c) Often rather than the degree of a node at the end of an edge we are interested in the number of edges attached to the node *other* than the one we arrived through. This number is called the *excess degree* of a node and will play an important role in the coming lectures. What is the probability that the node at which you arrive has *excess degree* k? [1P]
   (d) What is the probability of having a multiedge (i.e. at least two edges between the same two nodes) [1P]
   (e) Using the expression derived at the previous point, compute the expected total number of multiedges (limit of large m). What do you observe? [1P]
   (f) Consider a Molloy-Reed model with no self-loops and where we allow for the creation of multiple edges between a single pair of nodes. What is the probability that two nodes v and w with degrees d_v and d_w are connected? [1P]

3. **Inference and Statistical Ensembles**
   (a) Consider two microstates G_1 and G_2 with $n = 100$ nodes and $m_1 = 300$ and $m_2 = 350$ edges respectively. [2P]
   - a $G(n,p)$ model with $n = 100$ and $p = \frac{5}{99}$? What is the expected number of edges in this model?

Notes:

Self-study questions

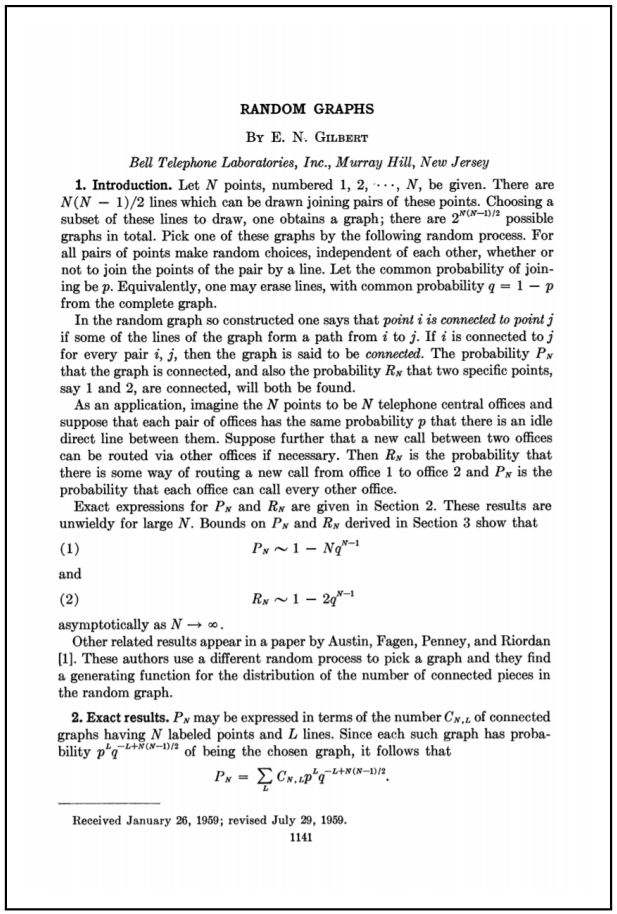
1. Write down an algorithmic formulation (in "pseudocode") of the network generation process underlying the $G(n, p)$ and the $G(n, m)$ model?
2. What is a statistical ensemble in the context of networks? What is a macrostate? What is a microstate?
3. What is the microstate probability of the empty network in the $G(n, m)$ model with $n = 5$ and $m = 10$.
4. Can you find a parameter of the $G(n, p)$ model such that all microstates in the resulting statistical ensemble have the same probability?
5. What is a probabilistic generative model and what is the likelihood function?
6. Explain how we can use probabilistic generative models of graphs for statistical inference.
7. Explain the difference between the likelihood function in frequentist inference and the posterior probability in Bayesian inference.
8. Why is the maximization of the likelihood function equivalent to the maximization of the log-likelihood function?

Notes:

References

reading list

- P Erdős, A Rényi: **On Random Graphs. I**, Publicationes Mathematicae, 1959
- EN Gilbert: **Random Graphs**, Annals of Mathematical Statistics, 1959
- EA Bender, ER Canfield: **The Asymptotic Number of Labeled Graphs with Given Degree Sequences**, Journal of Combinatorial Theory, 1978
- M Molloy, B Reed: **A critical point for random graphs with a given degree sequence**, Random Structures & Algorithms, 1995
- J Aldrich: **R. A. Fisher and the Making of Maximum Likelihood 1912 – 1922**, Statistical Science, 1997
- KP Murphy: **Probabilistic Machine Learning: An Introduction**, MIT Press, 2022 → Chapter I
- MEJ Newman, SH Strogatz, DJ Watts: **Random graphs with arbitrary degree distributions and their applications**, Physical Review E, 2001



RANDOM GRAPHS

By E. N. Gilbert

Bell Telephone Laboratories, Inc., Murray Hill, New Jersey

1. Introduction. Let N points, numbered 1, 2, $\cdots$, N, be given. There are $N(N-1)/2$ lines which can be drawn joining pairs of these points. Choosing a subset of these lines to draw, one obtains a graph; there are $2^{N(N-1)/2}$ possible graphs in total. Pick one of these graphs by the following random process. For all pairs of points make random choices, independent of each other, whether or not to join the points of the pair by a line. Let the common probability of joining be p. Equivalently, one may erase lines, with common probability $q = 1 - p$ from the complete graph.

In the random graph so constructed one says that *point i is connected to point j* if some of the lines of the graph form a path from i to j. If i is connected to j for every pair i, j, then the graph is said to be *connected*. The probability P_N that the graph is connected, and also the probability R_N that two specific points, say 1 and 2, are connected, will both be found.

As an application, imagine the N points to be N telephone central offices and suppose that each pair of offices has the same probability p that there is an idle direct line between them. Suppose further that a new call between two offices can be routed via other offices if necessary. Then R_N is the probability that there is some way of routing a new call from office 1 to office 2 and P_N is the probability that each office can call every other office.

Exact expressions for P_N and R_N are given in Section 2. These results are unwieldy for large N. Bounds on P_N and R_N derived in Section 3 show that

$$P_N \sim 1 - Nq^{N-1} \tag{1}$$

and

$$R_N \sim 1 - 2q^{N-1} \tag{2}$$

asymptotically as $N \to \infty$.

Other related results appear in a paper by Austin, Fagen, Penney, and Riordan [1]. These authors use a different random process to pick a graph and they find a generating function for the distribution of the number of connected pieces in the random graph.

2. Exact results. P_N may be expressed in terms of the number $C_{N,L}$ of connected graphs having N labeled points and L lines. Since each such graph has probability $p^L q^{-L+N(N-1)/2}$ of being the chosen graph, it follows that

$$P_N = \sum_L C_{N,L} p^L q^{-L+N(N-1)/2}.$$

Received January 26, 1959; revised July 29, 1959.

Notes: