Machine Learning for Complex 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 04
Stochastic Block Model

May 17, 2023

Notes:

- **Lecture L04:** Stochastic Block Model 17.05.2023
- **Educational objective:** We introduce the stochastic block model and show how it can be used to infer community patterns in complex networks.
  - The Stochastic Block Model
  - Inference in Stochastic Block Model
  - Heuristic Community Detection
  - Degree-corrected Stochastic Block Model

- **Exercise sheet 03:** Stochastic Block Model Inference due 24.05.2023
- **Handout version:** *2023/05/02 15:21:30*

Motivation

- we covered cut-based community detection using **Fiedler vector of graph Laplacian**
- can we use **probabilistic generative models to infer communities** in networks?

statistical inference of communities

1. define **probabilistic generative model**, where model parameters encode different **community structures**
2. **infer most plausible parameters** for a given empirical network

The many facets of community detection in complex networks

Michael T. Schaub,[1,2,3,*] Jean-Charles Delvenne,[2,4] Martin Rosvall,[5,†] and Renaud Lambiotte[3,‡]
[1] Institute for Data, Systems, and Society, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2] ICTEAM, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium
[3] naXys and Department of Mathematics, University of Namur, B-5000 Namur, Belgium
[4] CORE, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium[§]
[5] Integrated Science Lab, Department of Physics, Umeå University, SE-901 87 Umeå, Sweden

Community detection, the decomposition of a graph into essential building blocks, has been a core research topic in network science over the past years. Since a precise notion of what constitutes a community has remained evasive, community detection algorithms have often been compared on benchmark graphs with a particular form of assortative community structure and classified based on the mathematical techniques they employ. However, this comparison can be misleading because apparent similarities in their mathematical machinery can disguise different goals and reasons for why we want to employ community detection in the first place. Here we provide a focused review of these different motivations that underpin community detection. This problem-driven classification is useful in applied network science, where it is important to select an appropriate algorithm for the given purpose. Moreover, highlighting the different facets of community detection also delineates the many lines of research and points out open directions and avenues for future research.

Keywords: community detection, graph partitioning, Modularity; block models

I. INTRODUCTION

Sparked by the work of Newman and Girvan [1, 2] on Modularity in Complex Systems, the area of community detection has become one of the main pillars of network science research. The promise that we can gain a deeper understanding of a system by discerning important structural patterns within a network has spurred a huge number of studies in this area. However, as has become abundantly clear by now, this problem has no canonical solution. In fact, even a general definition of what constitutes a community is still lacking. The reasons for this are not only grounded in the computational difficulties of tackling community detection. Furthermore, various research areas view community detection from different perspectives, which the lack of a consistent terminology illustrates: 'network clustering', 'graph partitioning', 'community', 'block' or 'module detection' all carry slightly different connotations. This jargon barrier creates confusion as soon as readers and authors have different preconceptions and intuitive notions are not made explicit.

We argue that community detection should not be considered as a well-defined problem, but rather as an umbrella term with many facets. These facets emerge from different goals and motivations of what it is about the network that we want to understand or achieve, which lead to different perspectives on how to formulate the problem of community detection. Therefore, it is critical to be aware of these underlying motivations when selecting and comparing community detection methods. Thus, rather than an in-depth discussion of the technical details of different algorithmic implementations [3–10], here we focus on the conceptual differences between different perspectives on community detection.

By providing a problem-driven classification, however, we do *not* argue that the different perspectives are unrelated. In fact, in some situations, different mathematical problem formulations can lead to similar algorithms and methods, and the different perspectives can offer valuable insights. For example, for undirected networks, optimizing the objective function Modularity [1], initially proposed from a clustering perspective, can be interpreted both as optimizing a particular stochastic block model [11] and a particular diffusion process on the networks [12], In other situations, however, such relations are not apparent.

Neither do we argue that there is a particular perspective that is *a priori* better suited for any given network. In fact, no method can consistently perform best on all kinds of networks [13]. Community detection is an unsupervised learning task and we cannot know what are the quantities of interest for the analysis. Instead, to understand how useful a particular method is, we must take into account the context of why the researcher is interested in the communities [14].

In the following, we unfold different aims underpinning community detection and discuss how the resulting problem perspectives relate to various applications. We focus on four broad perspectives that have served as motivation for community detection in the literature: (i) community detection as minimization of some form of constraint violation; (ii) community detection framed as a discretised analogue of data clustering, in which densely knit groups of nodes are to be found; (iii) community detection aiming to identify structurally equivalent nodes in a network, leading to notions such as stochastic block models; and (iv) community detection looking for simplified descrip-

* mschaub@mit.edu; corresponding author
† martin.rosvall@umu.se
‡ renaud.lambiotte@unamur.be
§ jean-charles.delvenne@uclouvain.be

→ M Schaub, JC Delvenne, M Rosvall, R Lambiotte, 2017

Notes:

- Last week, we introduced the very basics of statistical inference in complex networks. We have used probabilistic generative models that, based on a given set of parameters, generate microstates with certain probabilities. We obtain a statistical ensemble of random graphs, where the probability mass function is given by our probabilistic model. While we can use such a model to generate a random network based on known parameter values, we can also do the opposite, i.e. knowing a specific network realization we can ask which parameters are most likely, i.e. for which parameter values the microstate probability of the observed network is maximal.
- Let us now use this approach to address community detection, which we previously addressed using the Fiedler vector of the graph Laplacian (see L02).
- We now introduce a different approach that is closely related to other statistical learning algorithms. It highlights general problems in model inference, model selection, and machine learning that are relevant in many other contexts.
- This approach will be based on a simple generalization of the random graph models that we have already discussed. We are interested in a generalization where the model parameters capture different community structures, which we can then infer.

The stochastic block model

- idea: **generalize $G(n,p)$ model** to incorporate **latent community labels**

stochastic block model

1. generates networks with n nodes and B **blocks/communities** $\{0,\ldots,B-1\}$
2. entries of **block assignment vector** $\vec{z} \in \{0,\ldots,B-1\}^n$ map node i to block z_i
3. **stochastic block matrix** $\mathbf{M} \in \mathbb{R}^{B\times B}$ gives link probabilities M_{kl} between pairs of nodes v and w with $z_v = k$ and $z_w = l$

Kathryn B Laskey

Paul W Holland
born 1940

Sam Leinhardt

- within blocks k, links are generated as in $G(n,p)$ model with $p = M_{kk}$ and $n = |\{v \in V : z_v = k\}|$
- for $B = 1$ and $\mathbf{M} = (p)$ we recover the $G(n,p)$ model

image credit: George Mason University (left), National Academic of Education (middle), Crunchbase (right), fair use

Notes:

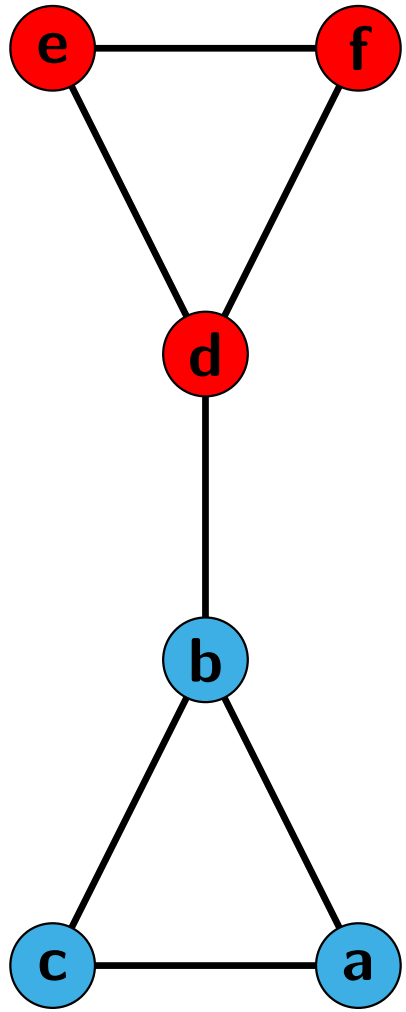
- We consider a generalization of the Erdős-Rényi model which incorporates so-called **latent features** of nodes that are used in the random link formation process (the word "latent" in Latin means "being hidden").
- The model introduces "blocks" (read: communities or clusters) and **assigns nodes to blocks** by means of a block assignment vector $\vec{z}$.
- Compared to the simple $G(n, p)$ model, we do not randomly generate links with a single probability p but rather consider **different link probabilities, depending on the assignment of nodes to blocks**.
- The different probabilities of links between pairs of nodes in different blocks are captured in a **stochastic block matrix** M.
- The stochastic block model was first introduced in 1983, see → PW Holland, KB Laskey, S Leinhardt, 1983 . It allows us to generate arbitrary network topologies based on a given community structure. In fact, the model is even more general: based on the correlation between block assignments and the entries in the stochastic block matrix, we can also generate networks in which nodes are **less** likely to connect to nodes in the same block, i.e. networks that exhibit the opposite of community structures. For those who attended our course Statistical Network Analysis, this would be networks with a negative community assortativity coefficient, i.e. nodes exhibit a preference to connect to nodes in different communities.

Toy example

- network with $n = 6$ nodes and $B = 2$ blocks denoted as 0 (blue) and 1 (red)
- $\vec{z} = (0, 0, 0, 1, 1, 1)$ assigns nodes $\{a, b, c, d, e, f\}$ to blocks
- stochastic block matrix

$$\mathbf{M} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix} \end{array}$$

- we obtain **statistical ensemble of random graphs with given community structure**



example

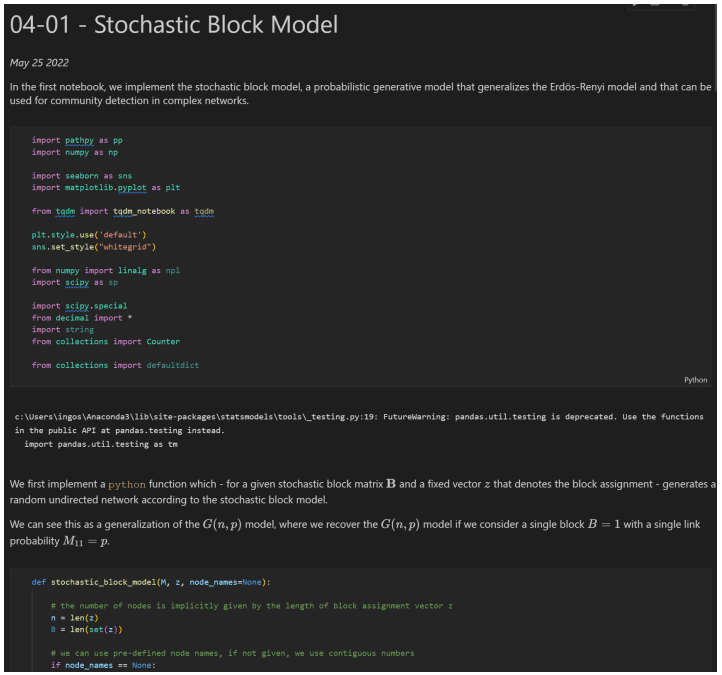
random microstate generated by stochastic block model

Notes:

- We consider a simple toy example for a stochastic block model with six nodes, two blocks and an assignment vector that assigns three nodes a, b, c to one block (red), and three other nodes d, e, f to another block (blue).
- The stochastic block matrix above sets the probability for nodes within the same block to be connected to each other to one, while the probability for connections between nodes in different blocks is 0.1. We obtain a statistical ensemble of networks and the network on the right is one particular realiation in this ensemble.
- In other words, we have defined a probabilistic generative model that generates networks with probabilities that depend on the number of blocks B, the block assignment vector z and the stochastic block matrix $\mathbf{M}$.
- If we know the parameters of the stochastic block model (i.e. the number of nodes, the number of blocks, the block assignment vector and the stochastic block model), we can calculate the probability $P(G)$ of every possible microstate G. This implies that for each given empirical network G_e we can calculate the likelihood of the parameters B, $\vec{z}$ and $\mathbf{M}$ of the stochastic block model.

Practice session

- we implement the **stochastic block model**
- we use is to generate **randomized versions of an empirical network with given community structure**



04-01 - Stochastic Block Model

May 25 2022

In the first notebook, we implement the stochastic block model, a probabilistic generative model that generalizes the Erdös-Renyi model and that can be used for community detection in complex networks.

```
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 decimal import *
import string
from collections import Counter

from collections import defaultdict
```

Python

```
c:\Users\ingos\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions
in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

We first implement a python function which - for a given stochastic block matrix $\mathbf{B}$ and a fixed vector z that denotes the block assignment - generates a random undirected network according to the stochastic block model.

We can see this as a generalization of the $G(n,p)$ model, where we recover the $G(n,p)$ model if we consider a single block $B = 1$ with a single link probability $M_{11} = p$.

```
def stochastic_block_model(M, z, node_names=None):

    # the number of nodes is implicitly given by the length of block assignment vector z
    n = len(z)
    B = len(set(z))

    # we can use pre-defined node names, if not given, we use contiguous numbers
    if node_names == None:
```

practice session

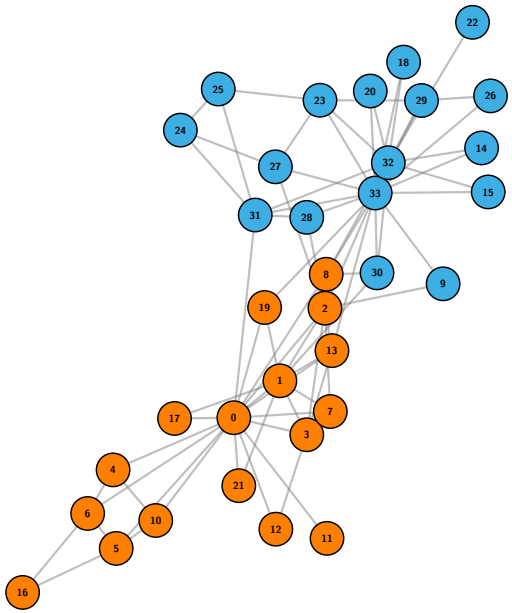
see notebook 04–01 in `gitlab` repository at

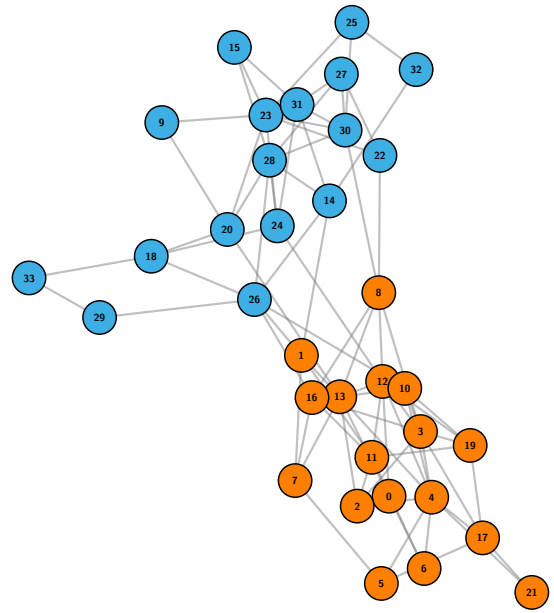
→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2023_sose_ml4nets_notebooks`

Notes:

- In the first practice session, we implement the stochastic block model in python. We further use ot to generate randomized versions of empirical networks with a known community structure, using the estimation of block matrix entries mentioned on the previous slide.

Empirical example: Karate Club network





empirical network

social network with two ground truth communities given by two "factions" within a Karate club

$\longrightarrow$ WW Zachary, 1977

random network

random network generated by stochastic block model with $B = 2$, $\vec{z}$ matching factions, and $\mathbf{M}$ matching observed links within/across communities

challenge

how can we use stochastic block model to **infer community structures** in networks **without** ground truth community labels?

Notes:

- An interesting feature of generative models is that we can use them to generate new (randomized) versions of our data. For this, we need to somehow "fit" the parameters of our model to an observed network, such that the model generates networks that are similar to our observation. In the practice sessions of the previous lecture, we have done the same using the $G(n, p)$ model, where we adjusted the parameter p such that it generated networks whose expected number of links corresponds to those in an empirical example. This allowed us to generate randomized versions of a network with the same density of links.
- For the stochastic block model, we must fit the number of nodes n (that is easy), the number of blocks B and the block assignment vector $\vec{z}$ (both is easy if we have a network with known community labels) and entries of the stochastic block matrix $\mathbf{M}$.
- Let us consider this for a popular small empirical network with ground truth information on group structure, the so-called **Karate Club network**. This network is based on a sociological study from 1977, which observed the relationships between members of a Karate Club that split during the observation. The club eventually separated into two factions centered around two influential individuals, which provides us with ground truth on communities.
- Using the known community structure, we can count the number of nodes in each block. This allows us to calculate the number of possible links for each pair of blocks. We can further calculate the number of actual links connecting each pair of blocks, which can be used to calculate the block matrix entries in analogy to slide 24 in L03.

Likelihood function

- consider empirical network $G_e = (V, E)$
- for given number of blocks B, block assignment vector $\vec{z}$ and stochastic block matrix $\mathbf{M}$, the **likelihood function** is given as

$$\mathcal{L}(\mathbf{M}, \vec{z}) = \prod_{(i,j)\in E} M_{z_i z_j} \cdot \prod_{(i,j)\notin E} (1 - M_{z_i z_j})$$

- can be rewritten as product over pairs of blocks k, l

$$\mathcal{L}(\mathbf{M}, \vec{z}) = \prod_{0\leq k\leq l\leq B-1} M_{kl}^{E_{kl}} \cdot (1 - M_{kl})^{N_{kl} - E_{kl}}$$

where E_{kl} is number of links between blocks k and l and N_{kl} is number of **possible** links between blocks k and l

Notes:

- For the network shown on the previous slide we simply calculated the entries of the stochastic block matrix based on the size of blocks and the number of links connecting each pair of blocks. As we shall see in the following slides, with this we have effectively performed a maximum likelihood estimation of the block matrix.
- Thanks to the simplicity of the stochastic block model, we can directly calculate the likelihood of a particular network realization based on a known stochastic block matrix and block assignment vector (see above).
- In this expression, we simply multiply the **probabilities of links** between pairs of connected nodes, as well as the **probabilities of not having links** between those nodes that are not connected. using the block assignment vector $\vec{z}$, these two probabilities are given by the corresponding block matrix entries (and one minus these entries respectively).
- We can reinterpret this expression in analogy to flipping a biased coin for each pair of blocks. For each pair of blocks k, l we include a factor M_{kl} as many times as we have links between blocks k and l. Similarly, we include a factor $1 - M_{kl}$ as many times as we have missing links between blocks k and l.
- We can rewrite the likelihood function above in terms of a product over pairs of blocks k, l. For undirected networks, it is sufficient to consider each pair k, l only in one direction (thus the subscript $0 \leq k \leq l \leq B - 1$).
- In other words: for each pair of blocks k, l we have the probability of having E_{kl} successes in a trial that succeeds with probability M_{kl}, while we have $N_{kl} - E_{kl}$ failures (which happens with probability $1 - M_{kl}$).

Likelihood function

- consider empirical network $G_e = (V, E)$ with given number of blocks B and block assignment vector $\vec{z}$
- **number of links between blocks k and l** defined as

$$E_{kl} := |\{(i,j) \in E : z_i = k \text{ and } z_j = l\}|$$

- **number of nodes in block k** defined as

$$C_k := |\{i \in V : z_i = k|$$

- **number of possible links between block k and l** is given as

$$N_{kl} := \begin{cases} \binom{C_k}{2} & \text{if } k = l \\ C_k \cdot C_l & \text{if } k \neq l \end{cases}$$

Notes:

- To calculate the likelihood function from the previous slide for a given empirical network G_e, we must count the **number of existing links** E_{kl} between blocks k and l, as well as the **number of possible links** N_{kl} between those blocks.
- If we consider an **undirected network with no self-loops**, the number of **possible links within a block** is $\binom{N_k}{2}$ (cf. the combinatorial primer in L03). For the maximum number of **possible links across blocks** (i.e. $k \neq l$) we can connect any of the N_k nodes in block k with any of the N_l nodes in block l. This implies that we have to consider two different cases for the calculation of N_{kl}.
- It is not difficult to adjust the stochastic block model in such a way that it also allows to study directed networks (or networks with multi-edges or self-loops), but here we limit our analysis to the simplest possible case.

Maximum likelihood estimate of M

- for pair of blocks k, l the **probability to generate microstate with exactly E_{kl} links** yields the following likelihood for block entry M_{kl}

$$\mathcal{L}(M_{kl}, \vec{z}) = M_{kl}^{E_{kl}} \cdot (1 - M_{kl})^{N_{kl} - E_{kl}}$$

- consider the **log-likelihood function**:

$$\log \mathcal{L}(M_{kl}, \vec{z}) = E_{kl} \cdot \log(M_{kl}) + (N_{kl} - E_{kl}) \cdot \log(1 - M_{kl})$$

- derivation by M_{kl} yields:

$$\frac{d}{dM_{kl}} \log \mathcal{L}(M_{kl}, \vec{z}) = \frac{E_{kl}}{M_{kl}} - \frac{N_{kl} - E_{kl}}{1 - M_{kl}}$$

- solving $\frac{d}{dM_{kl}} \log \mathcal{L}(M_{kl}, \vec{z}) = 0$ yields **maximum likelihood estimate for block matrix entries**

$$\hat{M}_{kl} = \frac{E_{kl}}{N_{kl}}$$

Notes:

- We have calculated the number of observed links E_{kl} and the number of possible links N_{kl} within and across any pair of blocks k and l. We also know the probability M_{kl} to generate a link between nodes in block k and l.
- This allows us to calculate the probability of generating exactly the number of links between each pair of blocks k and l individually, based on the entries M_{kl} in the stochastic block matrix. The resulting expression is very similar to the likelihood function of the $G(n, p)$ model (see slide 24 in L03).
- Just like for the $G(n, p)$ model, we use the log-likelihood function. We can directly compute the maximum likelihood entries $\hat{M}_{kl}$ by means of a simple derivation (note that we again have a function with a single local extremum, which is the global maximum).
- For a given block assignment vector $\vec{z}$ and a fixed number of blocks B this allows us to compute the maximum likelihood value $\hat{M}_{kl}$ for each entry of the stochastic block matrix $\mathbf{M}$ separately.
- We find that the maximum likelihood estimate of the stochastic block matrix entries is simply given by the ratio between the number of observed links and the number of possible links, which is the formal justification of our estimation from before.

Maximum likelihood of stochastic block model

1. consider likelihood function of stochastic block model

$$\mathcal{L}(\mathbf{M}, \vec{z}) = \prod_{0 \leq k \leq l \leq B-1} M_{kl}^{E_{kl}} \cdot (1 - M_{kl})^{N_{kl} - E_{kl}}$$

2. for given empirical network G_e and block assignment vector $\vec{z}$, we can **substitute M_{kl} by maximum likelihood estimate $\hat{M}_{kl} = \frac{E_{kl}}{N_{kl}}$**

3. we obtain a **maximum likelihood function across all block matrices $\mathbf{M}$**

$$\hat{\mathcal{L}}(\vec{z}) := \mathcal{L}(\hat{\mathbf{M}}, \vec{z}) = \prod_{0 \leq k \leq l \leq B-1} \frac{E_{kl}}{N_{kl}}^{E_{kl}} \cdot \left(1 - \frac{E_{kl}}{N_{kl}}\right)^{N_{kl} - E_{kl}}$$

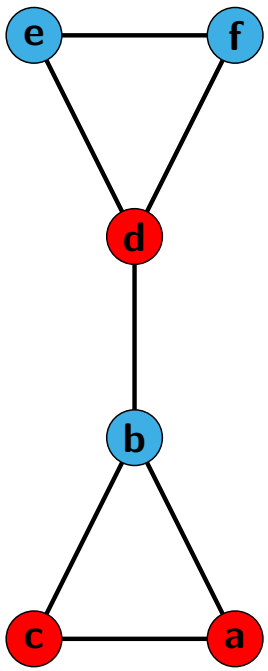
Notes:

- Since links between different pairs of blocks are generated independently, we can substitute the entries M_{kl} by the maximum likelihood estimation $\hat{M}_{kl}$ to maximize the likelihood function of the stochastic block model
- We obtain a maximum likelihood function which only depends on the numbers N_{kl} and E_{kl} calculated for our observed network as well as our choice of $\vec{z}$. This means that for any empirical network G_e and any block assignment vector $\vec{z}$ we can directly compute the block matrix $\mathbf{M}$ that maximizes the likelihood of our model given the observation G_e.
- For a given empirical network G_e and a given number of blocks B, we can directly compare the maximum likelihoods for different block assignment vectors $\vec{z}$ (since $\hat{\mathbf{M}}$ is determined by the choice of $\vec{z}$).

Examples

example 1

$$B = 2$$
$$\vec{z} = (0, 1, 0, 0, 1, 1)$$

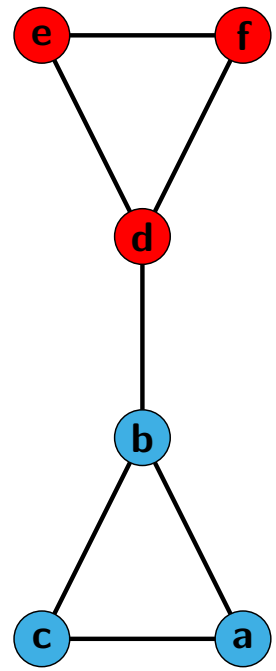


$$\hat{M}_{kl} = \frac{E_{kl}}{N_{kl}}, \qquad \widehat{\mathbf{M}} = \begin{matrix} \\ 0 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 \end{matrix} \\ \begin{bmatrix} \frac{1}{3} & \frac{5}{9} \\ \frac{5}{9} & \frac{1}{3} \end{bmatrix} \end{matrix}$$

$$\mathcal{L}(\vec{z}) = 0.0000453$$

example 2

$$B = 2$$
$$\vec{z} = (0, 0, 0, 1, 1, 1)$$



$$\hat{M}_{kl} = \frac{E_{kl}}{N_{kl}}, \qquad \widehat{\mathbf{M}} = \begin{matrix} \\ 0 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 \end{matrix} \\ \begin{bmatrix} 1 & \frac{1}{9} \\ \frac{1}{9} & 1 \end{bmatrix} \end{matrix}$$

$$\mathcal{L}(\vec{z}) = 0.043$$

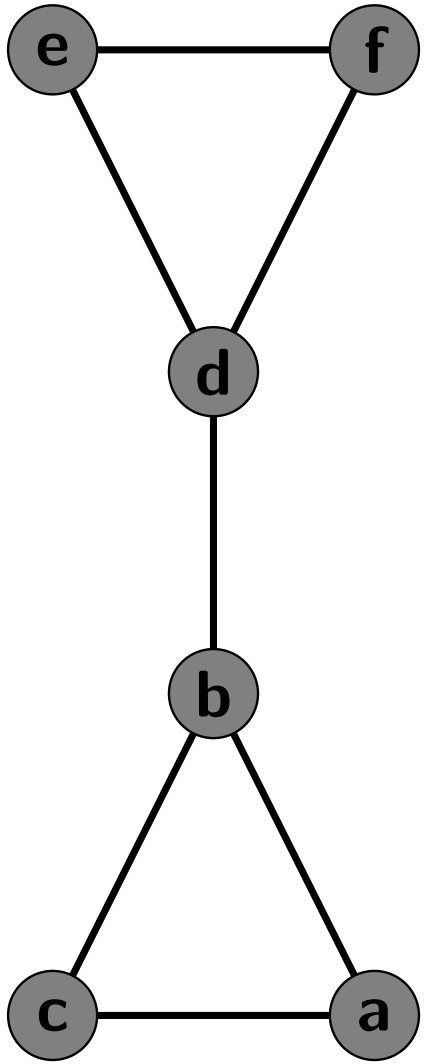
Notes:

- We apply this to our toy network (where we assume that self-loops are impossible). Above we assume a fixed number of two blocks, which are represented by entries 0 and 1 in the given block assignment vector $\vec{z}$.
- For example 1, we calculate $N_{00} = N_{11} = 3$ and $N_{01} = N_{10} = 9$. We further have $E_{00} = 1$ and $E_{11} = 1$ and $E_{01} = E_{10} = 5$. This yields the following maximum likelihood estimators for the entries in the stochastic block matrix $\hat{\mathbf{M}}$
  - $\hat{M}_{00} = \frac{E_{00}}{N_{00}} = \frac{1}{3}$, $\hat{M}_{01} = \frac{E_{01}}{N_{01}} = \frac{5}{9}$, $\hat{M}_{10} = \frac{E_{10}}{N_{10}} = \frac{5}{9}$, $\hat{M}_{11} = \frac{E_{11}}{N_{11}} = \frac{1}{3}$
- Based on this maximum likelihood estimation of $\mathbf{M}$, we can directly calculate the maximum likelihood for the block assignment vector $\vec{z}$ above. We can compare this to the maximum likelihood that we get for a block assignment vector that encodes "more plausible" communities (i.e. those block assignments that were actually used to generate the toy example, see example 2). Here we have $N_{00} = N_{11} = 3$ and $N_{01} = N_{10} = 9$. We further find $E_{00} = E_{11} = 3$ and $E_{01} = E_{10} = 1$ We thus have the following maximum likelihood estimators for the entries in the stochastic block matrix $\hat{\mathbf{M}}$
  - $\hat{M}_{00} = \frac{E_{00}}{N_{00}} = 1$, $\hat{M}_{01} = \frac{E_{01}}{N_{01}} = \frac{1}{9}$, $\hat{M}_{10} = \frac{E_{10}}{N_{10}} = \frac{1}{9}$, $\hat{M}_{11} = \frac{E_{11}}{N_{11}} = 1$
- We obtain a maximum likelihood that is about three orders of magnitude larger than for the "wrong" block assignment vector. Assuming the best possible choice of the stochastic block matrix, for the "correct" block assignment vector $\vec{z} = (0, 0, 0, 1, 1, 1)$ the model is about 1000 times more likely to generate our toy example than for the "wrong" assignment vector $\vec{z} = (0, 1, 0, 0, 1, 1)$.

Community detection with SBM

- consider **empirical network** G_e and **given number of** B **blocks**
- for each block assignment vector $\vec{z}$ we can calculate maximum likelihood $\hat{\mathcal{L}}(\vec{z})$ over all block matrices
- **optimal partition** in B communities is given by block assignment vector $\hat{\vec{z}}$ which maximizes likelihood of G_e, i.e.

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



Notes:

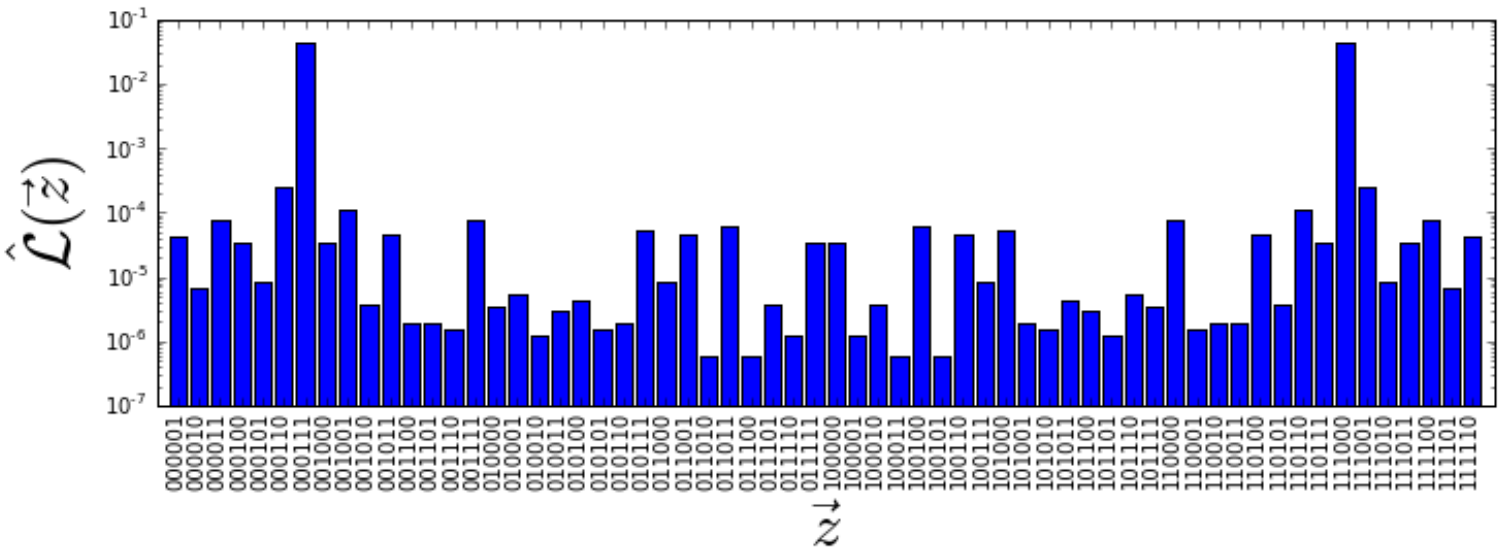
- This provides us with an interesting approach to detect communities. For a given empirical network G_e, we first fix the number of communities (or blocks) B that we want to detect. We note that, in practice it is difficult to choose this number but we will address this in the next lecture.
- For a given value B we can calculate the maximum likelihood $\mathcal{L}(\vec{z})$ for each block assignment vector $\vec{z}$, across all possible stochastic block matrices.
- To find what is the "most plausible" block assignment, we must identify the block assignment vector $\vec{z}$ which maximizes $\mathcal{L}(\vec{z})$.
- Unfortunately, since the space of block assignment vectors is discrete the resulting likelihood is not a "smooth" function. Finding the optimal block assignment vector is thus not as simple as taking a derivative, which complicates the application of this approach. We will see that we can fix this using heuristic optimization techniques, however this introduces downstream complications as well as additional "hyperparameters", which makes the method less elegant in practice than it appears at first glace (more on this in a moment).

Example: community detection

naive approach

we can exhaustively enumerate $\vec{z}$ for fixed $B = 2$

1. generate all possible block assignment vectors $\vec{z}$
2. for each $\vec{z}$ calculate maximum likelihood $\hat{\mathcal{L}}(\vec{z})$
3. choose $\hat{\vec{z}} := \arg\max_{\vec{z}} \hat{\mathcal{L}}(\vec{z})$



maximum likelihood of SBM for different vectors $\vec{z}$ and example network from previous slide

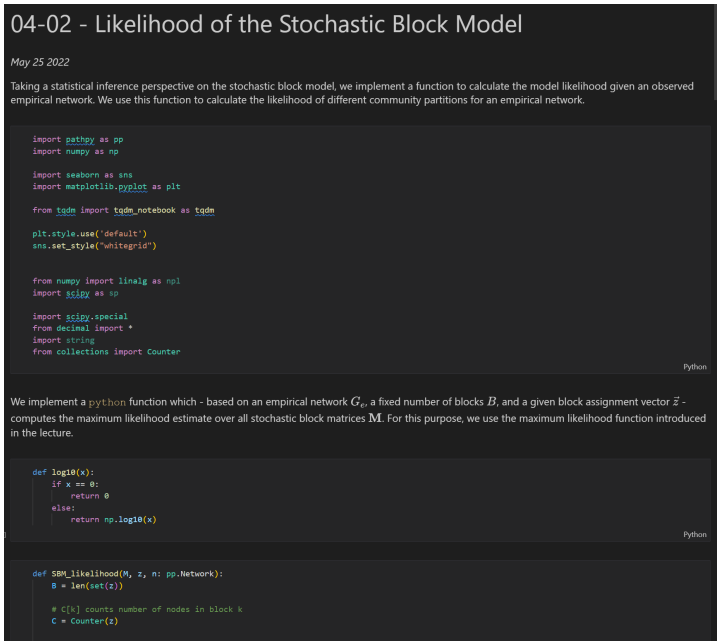
$\hat{\mathcal{L}}(\vec{z})$ maximal for $\vec{z} = (0,0,0,1,1,1)$ and $\vec{z} = (1,1,1,0,0,0)$

Notes:

- For our small toy example we can use a naive “brute force” strategy that enumerates all possible block assignment vectors $\vec{z}$. We have six nodes, each of which can be assigned to one of two blocks. This means that each block assignment vector can be represented as 6-digit binary number (0 or 1 corresponding to the two blocks) (see x-axis of plot). For each of these $2^6 = 64$ assignments, we can calculate the maximum likelihood $\mathcal{L}(\vec{z})$ (see y-axis of plot).
- This plot is symmetric, because the left and the right half of the assignment vectors just represent the same community assignments with flipped community labels. We find two maxima that correspond to the (flipped) partition $(0, 0, 0, 1, 1, 1)$ and $(1, 1, 1, 0, 0, 0)$.
- Obviously, this naive strategy of enumerating all possible block assignment vectors for a given number of blocks B is not efficient as the number of possible assignments grows as B^n. In practice we thus use heuristic methods to find the “optimum” block assignment vector $\vec{z}$ (see following slides).

Practice session

- we calculate the **likelihood of block assignment vectors**
- we use maximum likelihood estimation to **infer community structures**



04-02 - Likelihood of the Stochastic Block Model

May 25 2022

Taking a statistical inference perspective on the stochastic block model, we implement a function to calculate the model likelihood given an observed empirical network. We use this function to calculate the likelihood of different community partitions for an empirical network.

```
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 decimal import *
import string
from collections import Counter
```

Python

We implement a `python` function which - based on an empirical network G_e, a fixed number of blocks B, and a given block assignment vector $\vec{z}$ - computes the maximum likelihood estimate over all stochastic block matrices $\mathbf{M}$. For this purpose, we use the maximum likelihood function introduced in the lecture.

```
def log10(x):
    if x == 0:
        return 0
    else:
        return np.log10(x)
```

Python

```
def SBM_likelihood(M, z, n: pp.Network):
    B = len(set(z))

    # c[k] counts number of nodes in block k
    c = Counter(z)
```

practice session

see notebook 04–02 in `gitlab` repository at

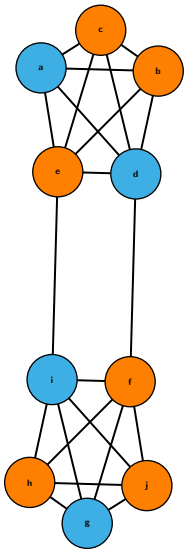
→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2023_sose_ml4nets_notebooks`

Notes:

- In the second practice session, we implement a function that calculates the maximum likelihood of the stochastic block model for a given number of blocks B and a block assignment vector $\vec{z}$.
- We then use it to detect communities in our small toy example, where we can exhaustively enumerate all possible community assignments.

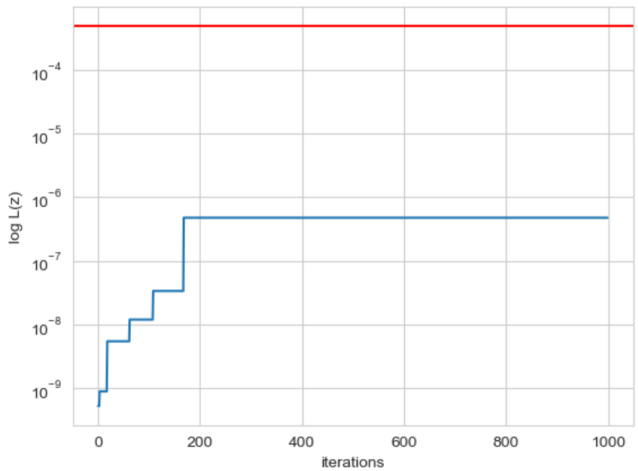
Greedy Maximization of Likelihood?

- enumeration of block assignments not feasible in real networks
- need **heustic optimization algorithm** to efficiently find (near-)optimal solution
- naive idea: use **likelihood $\mathcal{L}(\vec{z})$ as search heuristic** in simple greedy algorithm



naive greedy algorithm

```
for i in range(iterations):
  old_z = z[v]
  old_L = L(z)
  z[v] = random.randint(B)
  if L(z)<old_L:
    z[v] = old_z
```



evolution of likelihood for 1000 iterations of greedy algorithm (max. likelihood = red line)

- problem: naive greedy algorithm gets stuck in **suboptimal solutions**

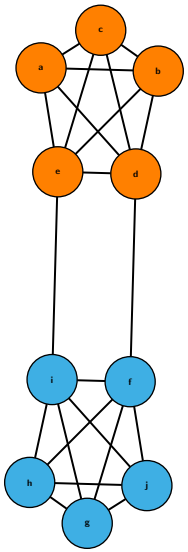
Notes:

- Clearly, we can not use an exhaustive enumeration of all possible community structures to a real network with a large number of nodes (and communities). Since there is no analytical solution for the maximum likelihood estimate of the block assignment vector, we must use heuristic optimization algorithms. Note that the term “heuristic” originates from Greek “heurísko” (I find).
- A naive idea is to apply a greedy heuristic using the likelihood function. Starting with a random assignment of nodes to communities, we can just randomly change the block label of a single node (i.e. we randomly change one entry in $\vec{z}$). If this change increases the likelihood, we accept it and continue. If not, we revert the change.
- While this is a very simple approach to optimize the likelihood of our block assignment vector, we face the problem that the likelihood landscape exhibits local maxima. That is, we can reach block assignment vectors for which (i) the likelihood does not correspond to the global likelihood maximum, and (ii) any change of an entry decreases the likelihood. If we reach such a local optimum, the algorithm is stuck and we will not find the optimal solution.
- Unfortunately, the solution landscape of our problem is complex in the sense that we are very likely to reach such local maxima even for very simple examples like the one shown above. The figure shows the optimal block assignment reaches by the greedy algorithm. In the plot, we show the evolution of the likelihood (which increases monotonically) and the maximum likelihood iof the optimal solution (shown as red line).

Optimization by Simulated Annealing

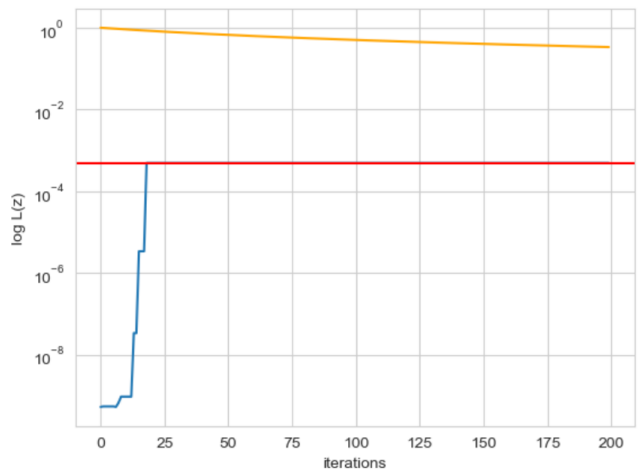
- we can use simulated annealing to **escape local likelihood maxima**
- accept solution with likelihood l' smaller than current value l with prob.

$$p(t) = e^{-\frac{l-l'}{t}} (t > 0)$$



simulated annealing algorithm → S Kirkpatrick et al., 1983

```
for i in range(iterations):
  t = (1+c)/(i+1+c) # cooling param.  c>0
  old_z = z[v]
  old_L = L(z)
  z[v] = random.randint(B)
  p = np.exp(-(old_L-L(z))/t)
  if L(z)<old_L or rand()>p:
    z[v] = old_z
```



- $t \to \infty \Rightarrow p(t) \to 1$
- $t \to 0 \Rightarrow p(t) \to 0$

Notes:

- A common approach to address this problem is by means of **simulated annealing** [**?**]. The name of this approach is due to an analogy to the annealing of metals, which is used by blacksmiths. The problem here is that due to the hammering of the metal, the atomic structure can get stuck in a sub-optimal configuration that weakens the material. A simple method to allow atoms to rearrange themselves into a stronger crystal structure is to heat the metal beyond its recrystallization point. This recrystallization is driven by temperature-dependent fluctuations.
- We can adopt a similar approach to avoid the local optima of our likelihood landscape. We introduce a temperature-dependent fluctuation by allowing our algorithm to accept solutions that are worse (i.e. have smaller likelihood) than the current one. The probability to accept such a solution depends on the likelihood and a temperature parameter t. For t very large the probability to accept a block assignment is close to one, independent of its likelihood. For t very small, we only accept it if the likelihood increases.
- To find an optimal solution, it is common to start with a high value for the temperature, which allows us to explore the solution landscape and to escape local optima. We then step by step reduce the temperature to allow the algorithm to settle to a solution that is –ideally– close to the global optimum.

Evaluating Community Detection

- to **quantify quality of community assignments** we need measure for correspondence between block assignment vectors $\vec{x}$ and $\vec{y}$

- **mutual information** of two random variables X and Y is defined as

$$I(X;Y) := \sum_{x,y} p(x,y) \cdot \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

- division by product of **entropy $H(X)$ and $H(Y)$**
  → L05 yields **normalized mutual information**

$$NMI(X;Y) := 2 \cdot \frac{I(X;Y)}{H(X) \cdot H(Y)} \in [0,1]$$

example 1

$$\vec{x} = [0,0,1,1]$$
$$\vec{y} = [0,1,0,1]$$
$$I(\vec{x};\vec{y}) = 0$$

example 2

$$\vec{x} = [0,0,1,1]$$
$$\vec{y} = [1,1,0,0]$$
$$I(\vec{x};\vec{y}) = 1$$

example 3

$$\vec{x} = [1,1,0,2]$$
$$\vec{y} = [0,0,2,1,]$$
$$MI(\vec{x};\vec{y}) = 1.5$$
$$NMI(\vec{x};\vec{y}) = 1$$

example 4

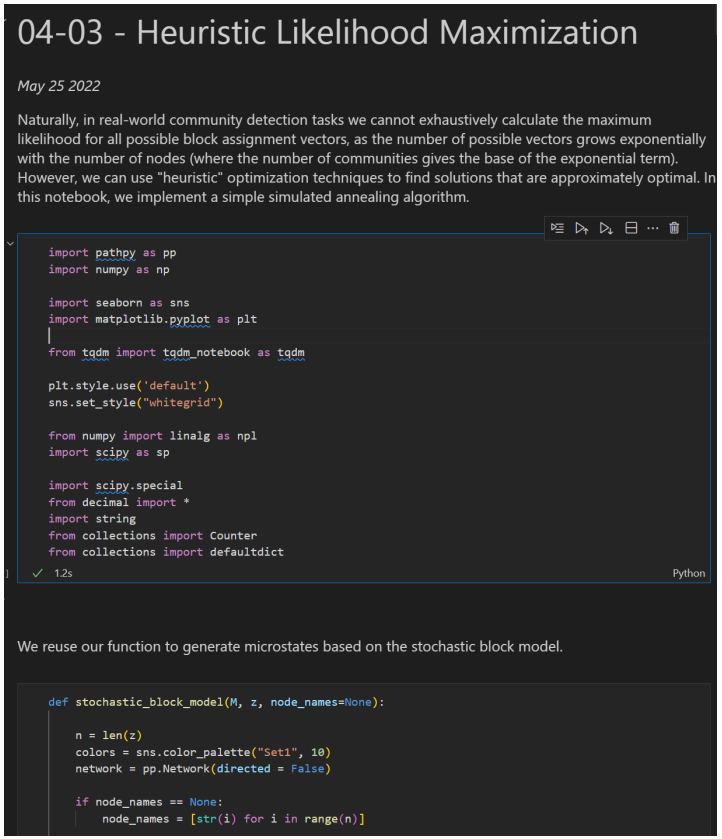
$$\vec{x} = [0,0,0,1]$$
$$\vec{y} = [1,1,1,0]$$
$$MI(\vec{x};\vec{y}) \approx 0.81$$
$$NMI(\vec{x};\vec{y}) = 1$$

Notes:

- Before we try this, we introduce a measure to evaluate the quality of detected communities if we have ground truth. We need to evaluate the correspondence between two block assignment vectors $\vec{x}$ and $\vec{y}$, independent of a relabeling of blocks.
- A common measure for this purpose is based on the concept of entropy and information, which we will introduce in more depth in the next lecture. For this, we consider the corresponding entries x_i and y_i of our two vectors as realizations of two random variables X and Y. We can now calculate the probability mass functions $p(x)$ and $p(y)$ of the entries, as well as the joint probability $p(x, y)$ of pairs of entries. We can use this to define the mutual information of $\vec{x}$ and $\vec{y}$, which tells us by how many bits of information our uncertainty about the entries of one vector is reduced if we know the entries of the other.
- In example 1, knowing that x_i is one or zero does not give us any information about y_i, which is why the mutual information is zero. In the second example, knowing one vector gives us perfect information about the other vector. Since in this case, the information content of the vector is one bit, our uncertainty is reduced by this one bit.
- Mutual information is not normalized, as it depends on the entropy of variables. In example 3, the mutual information is more than one bit because we have more than two communities (wich require more than one bit to encode). In example 4, the mutual information is less than one bit even though we have a perfect correspondence between the two vectors. This is due to the fact that one community label occurs more frequently, which reduces the entropy. We can normalize it by dividing by the product of entropies.

Practice session

- we use **simulated annealing to efficiently detect community structures**
- we use **normalized mutual information** to evaluate detected communities against ground truth
- we test the stochastic block model in a popular **empirical network**



04-03 - Heuristic Likelihood Maximization

May 25 2022

Naturally, in real-world community detection tasks we cannot exhaustively calculate the maximum likelihood for all possible block assignment vectors, as the number of possible vectors grows exponentially with the number of nodes (where the number of communities gives the base of the exponential term). However, we can use "heuristic" optimization techniques to find solutions that are approximately optimal. In this notebook, we implement a simple simulated annealing algorithm.

```
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 decimal import *
import string
from collections import Counter
from collections import defaultdict
```

✓ 1.2s Python

We reuse our function to generate microstates based on the stochastic block model.

```
def stochastic_block_model(M, z, node_names=None):

    n = len(z)
    colors = sns.color_palette("Set1", 10)
    network = pp.Network(directed = False)

    if node_names == None:
        node_names = [str(i) for i in range(n)]
```

practice session

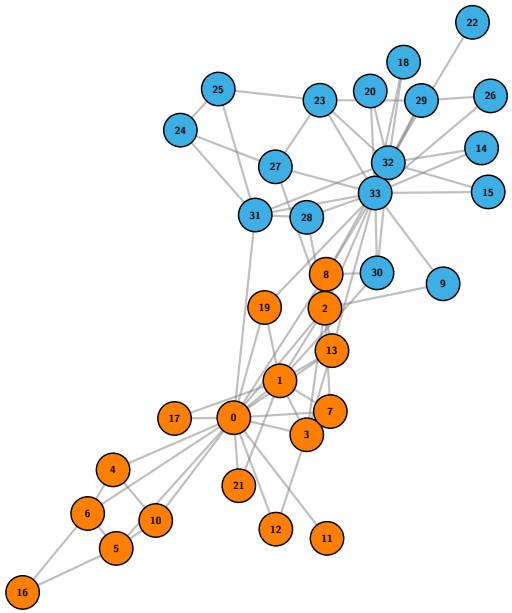
see notebook 04–03 in `gitlab` repository at

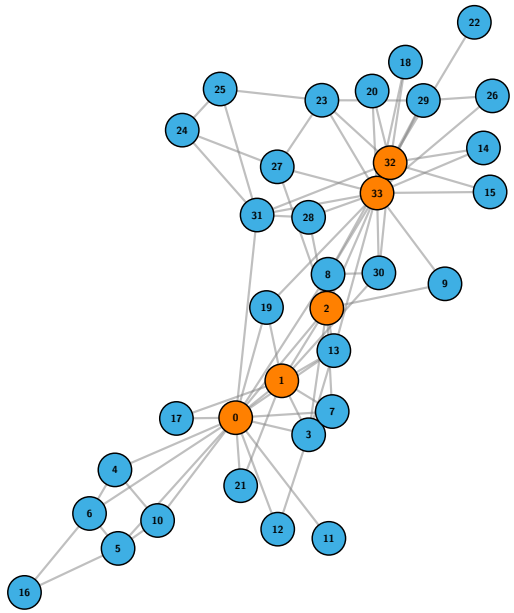
→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2023_sose_ml4nets_notebooks`

Notes:

- In the third practice session, we implement simulated annealing to find block assignment vectors with maximum likelihood. We further implement normalized mutual information and use it evaluate our algorithm in a data set where we have ground truth community labels.
- We finally test our algorithm in an empirical network.

Community detection in Karate club network





empirical network

social network with **two ground truth communities** given by two "factions" within a Karate club

→ WW Zachary, 1977

detected communities

communities detected by maximum likelihood estimation of stochastic block model with $B = 2$ (using simulated annealing)

$$NMI \approx 0.005$$

observation

detected communities correspond to **groups of high and low-degree nodes** rather than "natural" communities

Notes:

- As empirical example, we again consider the Karate Club network, where we can use the factions of the club to evaluate communities detected by the stochastic block model. Unfortunately, even though we used the simulated annealing approach to escape local maxima, we are not able to detect meaningful community structures.
- While you may suspect that this is due to an issue of our heuristic optimization algorithm, the problem is more complicated. In the practice session, we confirmed that the block assignment vector inferred by our algorithm, has an associated likelihood that is even larger than that of the ground truth communities!, i.e. something appears to be wrong with the notion of communities used by the stochastic block model, i.e. groups of nodes with higher link probabilities.
- Upon closer inspection of the community structures detected above (right image) we note that our algorithm has found a block assignment that places exactly those nodes with highest degree in one community and all other nodes in another community!

Node degrees and Stochastic Block Model

- consider link probability (i, j) in **Molloy-Reed configuration model** → L03

$$p(i,j) \propto \frac{d_i d_j}{2m}$$

- based on degree sequence, we expect higher link probability to be higher within groups of high-degree nodes
- problem: stochastic block model generates networks with **Poisson degree distribution**
- need **degree-corrected version** of stochastic block model that reproduces **expected degree sequence** rather than expected number of links

→ B Karrer, M Newman, 2011

Brian Karrer

image credit:
https://research.facebook.com/people/karrer-brian/

Notes:

- We can understand this issue if we reconsider the Molloy-Reed configuration model for random networks with arbitrary degree sequence. In lecture L03 we have seen that the probability of links in such a network is not uniform, as it depends on the degrees of the involved nodes. This tells us that, just based on the degree sequence of a network, we can expect higher link probabilities within groups of high-degree nodes. This is, in fact, what the stochastic block model detected.
- The root of this problem is that we need a model that is able to distinguish patterns from noise. Given the degree sequence of a network, we expect high-degree nodes to form groups with high link probability at random, i.e. even without the presence of any community structure. However, the stochastic block model dos not consider heterogeneous degree sequences as it generates networks with a Poisson degree distribution. We need a version of the model that corrects for heterogeneous degrees, i.e. a model that reproduces the (expected) degree sequence of the network in which we wish to infer communities.

Degree-corrected SBM

- let $\vec{d}$ be vector where d_i is **expected degree** of node i
- **expected value of entry** A_{ij} in adjacency matrix of generated network is

$$d_i d_j M_{z_i z_j}$$

- for network with adjacency matrix $\mathbf{A}$ **likelihood function of degree-corrected stochastic block model** is given as

$$\mathcal{L}(\mathbf{M}, \vec{z}, \vec{d}) = \prod_{i<j} \frac{(d_i d_j M_{z_i z_j})^{A_{ij}}}{A_{ij}!} e^{-d_i d_j M_{z_i z_j}} \cdot \prod_i \frac{(\frac{1}{2} d_i^2 M_{z_i z_j})^{\frac{A_{ii}}{2}}}{(A_{ii}/2)!} e^{-\frac{1}{2} d_i^2 M_{z_i z_j}}$$

- for **maximum log-likelihood function** over all block matrices $\mathbf{M}$ we have

$$\hat{\mathcal{L}}(\vec{z}) \propto \sum_{kl} E_{kl} \log \frac{E_{kl}}{\kappa_k \kappa_l}$$

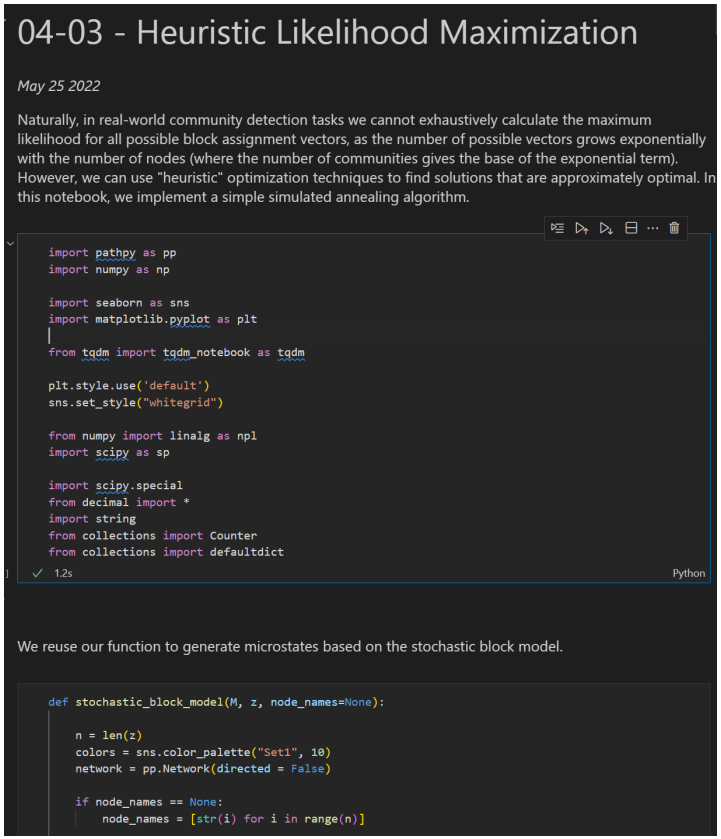
where κ_l is sum of degrees of nodes in block l → B Karrer, M Newman, 2011

Notes:

- Such a degree-corrected version of the stochastic block model has been proposed by Karrer and Newman in 2011. It essentially generalizes the Molloy-Reed model insofar as it is able to generate networks with a given (expected) degree sequence and a given latent community structure. Following the arguments from the original paper, we can write down a likelihood function of this model, see above.
- Using the same argument as before, we can directly calculate a maximum log-likelihood function over all stochastic block matrices that is even simpler than the expression before. We just replace the number pf possible links within and across communities with an expression that is given by the product of the degree sums in those blocks. Note that the simplified form of the maximum likelihood function above does not give the properly normalized likelihoods, but a value that is –up to some constants– proportional to the likelihood (and which we can thus use for optimization based on the simulated annealing algorithm).

Practice session

- we use the **degree-corrected stochastic block model** to infer communities



04-03 - Heuristic Likelihood Maximization

May 25 2022

Naturally, in real-world community detection tasks we cannot exhaustively calculate the maximum likelihood for all possible block assignment vectors, as the number of possible vectors grows exponentially with the number of nodes (where the number of communities gives the base of the exponential term). However, we can use "heuristic" optimization techniques to find solutions that are approximately optimal. In this notebook, we implement a simple simulated annealing algorithm.

```
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 decimal import *
import string
from collections import Counter
from collections import defaultdict
```

✓ 1.2s Python

We reuse our function to generate microstates based on the stochastic block model.

```
def stochastic_block_model(M, z, node_names=None):

    n = len(z)
    colors = sns.color_palette("Set1", 10)
    network = pp.Network(directed = False)

    if node_names == None:
        node_names = [str(i) for i in range(n)]
```

practice session

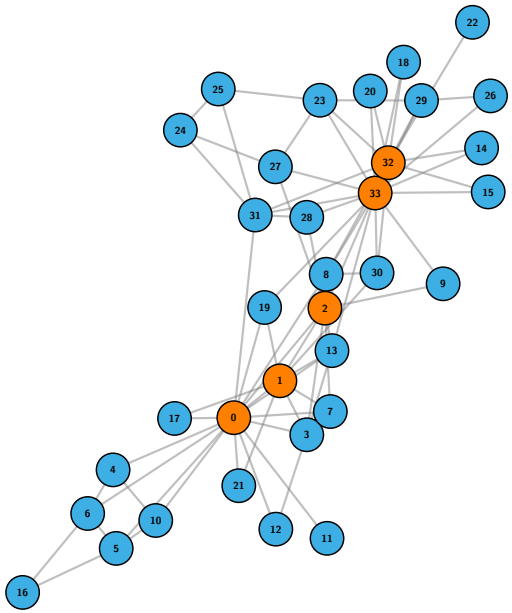
see notebook 04–04 in `gitlab` repository at

→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2023_sose_ml4nets_notebooks`

Notes:

- In the fourth and final practice session, we implement the likelihood function of the degree-corrected stochastic block model and apply it to the Karate club network.

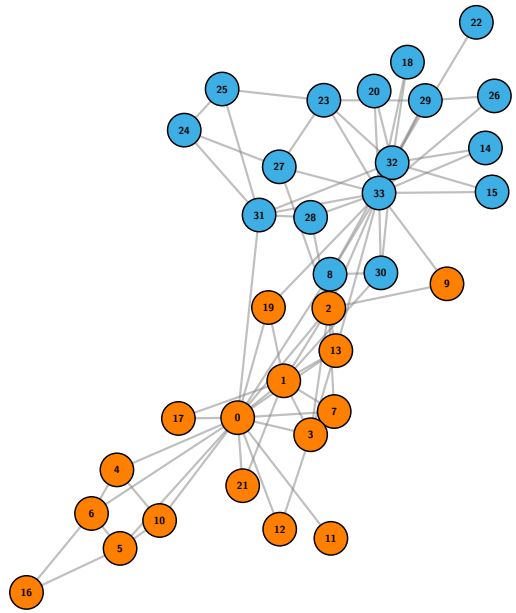
Application to example



detected communities

optimal community assignment inferred using **stochastic block model** with $B = 2$ using simulated annealing

$NMI \approx 0.005$



detected communities

optimal community assignment inferred using **degree-corrected stochastic block model** with $B = 2$ using simulated annealing

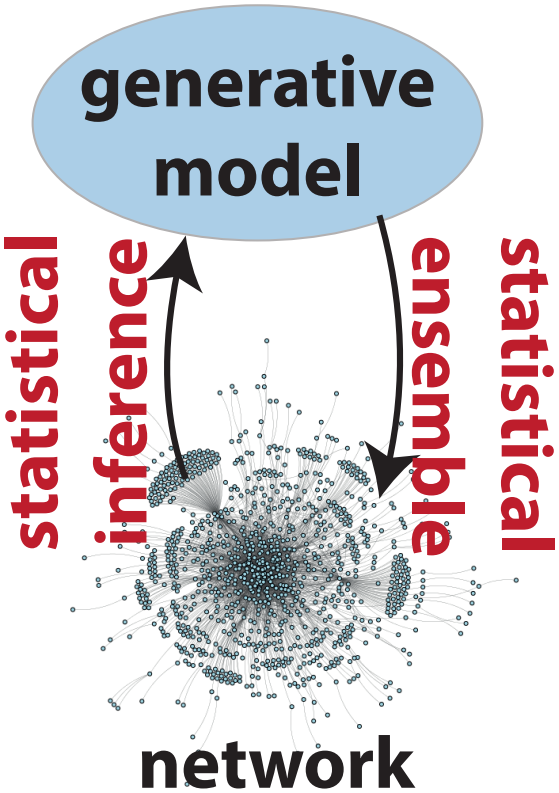
$NMI \approx 0.82$

Notes:

- We can now plug this new likelihood function, which is based on a Molloy-Reed version of the stochastic block model, into our simulated annealing algorithm and apply it to the Karate club network. We now find that –thanks to the fact that we consider the heterogeneous node degrees– we are now able to detect much more reasonable community structures. The normalized mutual information of the detected communities (compared to the two ground truth factions) increases from close to zero to a value of 0.68.

In summary ...

- we can **infer the most plausible model** that could have generated an empirical network
- suitably chosen models allow us to **discover knowledge hidden in relational data**
- **stochastic block model** can be used to detect community structures
- networks with heterogeneous degrees require **degree-corrected stochastic block model**
- need suitable probabilistic model to **dinstinguish patterns from noise**

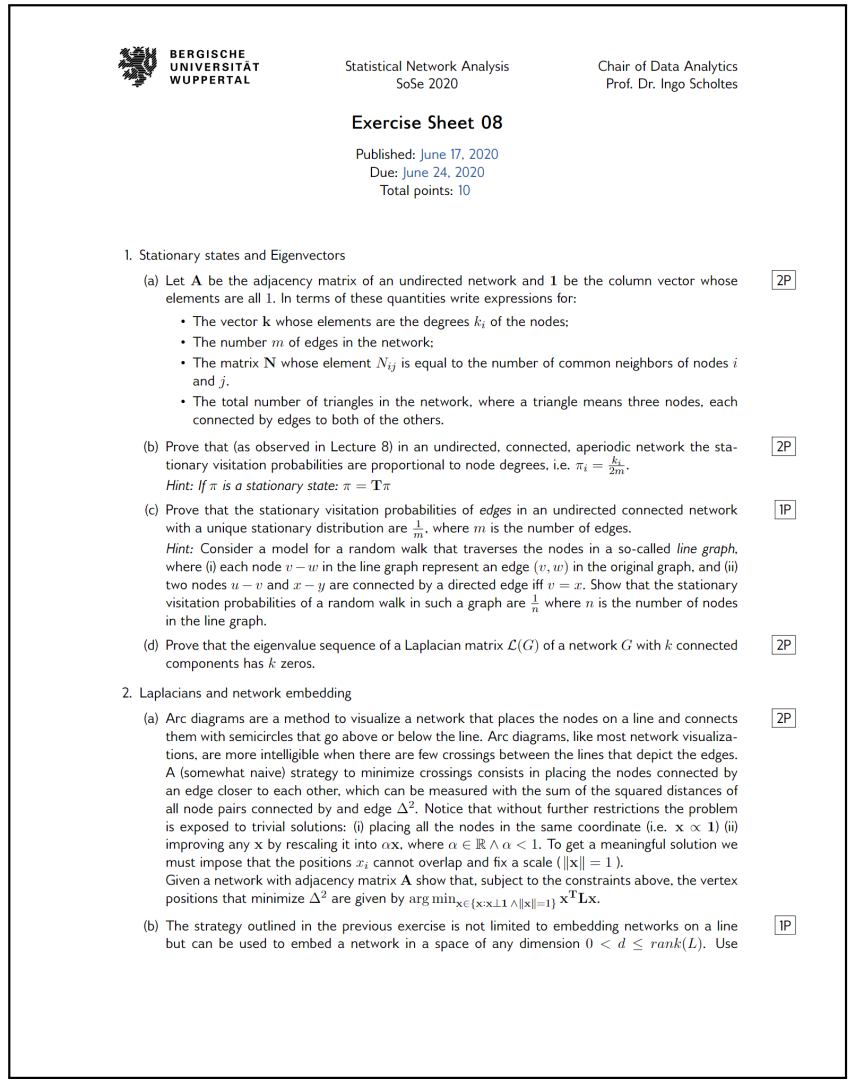


Notes:

- In summary, we have shown how we can use the stochastic block model to infer meaningful community structures, both in networks with homogeneous and (using the degree-corrected version) heterogeneous degree sequences.
- With this, we have seen how we can use inference to learn latent structures in a network, and how a suitably chosen probabilistic generative model helps us to distinguish topological patterns from noise.
- We will continue this topic next week, when we address the important question how we can learn the optimal number of communities that we wish to detect.

Exercise sheet 3

- **third exercise sheet** will be released today
  - explore the stochastic block model
  - study community detection
- solutions are due **May 24th** (via WueCampus)
- present your solution to earn bonus points



BERGISCHE UNIVERSITÄT WUPPERTAL

Statistical Network Analysis
SoSe 2020

Chair of Data Analytics
Prof. Dr. Ingo Scholtes

Exercise Sheet 08

Published: June 17, 2020
Due: June 24, 2020
Total points: 10

1. Stationary states and Eigenvectors

(a) Let $\mathbf{A}$ be the adjacency matrix of an undirected network and $\mathbf{1}$ be the column vector whose elements are all 1. In terms of these quantities write expressions for: [2P]

- The vector $\mathbf{k}$ whose elements are the degrees k_i of the nodes;
- The number m of edges in the network;
- The matrix $\mathbf{N}$ whose element N_{ij} is equal to the number of common neighbors of nodes i and j.
- The total number of triangles in the network, where a triangle means three nodes, each connected by edges to both of the others.

(b) Prove that (as observed in Lecture 8) in an undirected, connected, aperiodic network the stationary visitation probabilities are proportional to node degrees, i.e. $\pi_i = \frac{k_i}{2m}$. [2P]
Hint: If π is a stationary state: $\pi = \mathbf{T}\pi$

(c) Prove that the stationary visitation probabilities of *edges* in an undirected connected network with a unique stationary distribution are $\frac{1}{m}$, where m is the number of edges. [1P]
Hint: Consider a model for a random walk that traverses the nodes in a so-called *line graph*, where (i) each node $v-w$ in the line graph represent an edge (v,w) in the original graph, and (ii) two nodes $u-v$ and $x-y$ are connected by a directed edge iff $v=x$. Show that the stationary visitation probabilities of a random walk in such a graph are $\frac{1}{n}$ where n is the number of nodes in the line graph.

(d) Prove that the eigenvalue sequence of a Laplacian matrix $\mathcal{L}(G)$ of a network G with k connected components has k zeros. [2P]

2. Laplacians and network embedding

(a) Arc diagrams are a method to visualize a network that places the nodes on a line and connects them with semicircles that go above or below the line. Arc diagrams, like most network visualizations, are more intelligible when there are few crossings between the lines that depict the edges. A (somewhat naive) strategy to minimize crossings consists in placing the nodes connected by an edge closer to each other, which can be measured with the sum of the squared distances of all node pairs connected by and edge Δ^2. Notice that without further restrictions the problem is exposed to trivial solutions: (i) placing all the nodes in the same coordinate (i.e. $\mathbf{x} \propto \mathbf{1}$) (ii) improving any $\mathbf{x}$ by rescaling it into $\alpha\mathbf{x}$, where $\alpha \in \mathbb{R} \wedge \alpha < 1$. To get a meaningful solution we must impose that the positions x_i cannot overlap and fix a scale ($\|\mathbf{x}\| = 1$). [2P]
Given a network with adjacency matrix $\mathbf{A}$ show that, subject to the constraints above, the vertex positions that minimize Δ^2 are given by $\arg\min_{\mathbf{x}\in\{\mathbf{x}:\mathbf{x}\perp\mathbf{1}\wedge\|\mathbf{x}\|=1\}} \mathbf{x}^{\mathbf{T}}\mathbf{L}\mathbf{x}$.

(b) The strategy outlined in the previous exercise is not limited to embedding networks on a line but can be used to embed a network in a space of any dimension $0 < d \leq rank(L)$. Use [1P]

Notes:

Questions

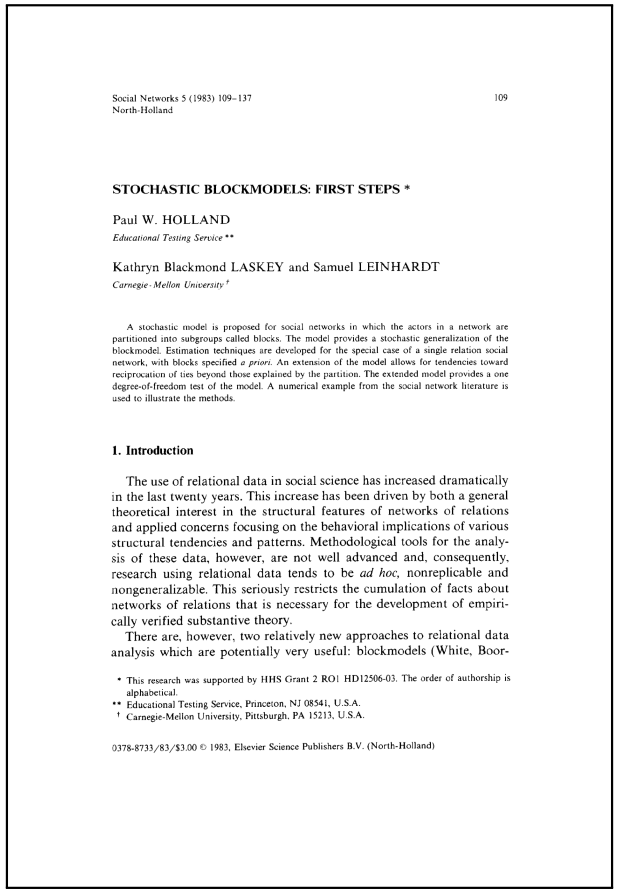
1. How can we calculate the maximum likelihood of a block assignment vector $\vec{z}$ with B blocks for a given network.
2. Explain how we can use the stochastic block model to infer community structures in networks.
3. Give an example for block assignment vectors, where mutual information is smaller than one while normalized mutual information is one.
4. Write down a pseudo-code formulation of the simulated annealing algorithm to detect communities based on the stochastic block model.
5. Explain when and why we need the degree-corrected stochastic block model.
6. How can we estimate block matrix entries in the stochastic block model?
7. Consider an example network with five nodes and ten links. Give parameters B and $\vec{z}$ for the stochastic block model such that the maximum likelihood is one.
8. Consider a version of the simulated annealing algorithm where the number of blocks can increase over time. Test this algorithm and explain which community structures it detects.

Notes:

References

reading list

- N Metropolis, AW Rosenbluth; MN Rosenbluth, AH Teller, E Teller: **Equation of State Calculations by Fast Computing Machines**, The Journal of Chemical Physics, 1953
- WW Zachary: **An Information Flow Model for Conflict and Fission in Small Groups**, Journal of Anthropological Research, Vol. 33, 1977
- PW Holland, KB Laskey, S Leinhardt: **Stochastic blockmodels: First steps**, Social Networks, 1983
- S Kirkpatrick, CD Gelatt Jr, MP Vecchi: **Optimization by Simulated Annealing**, Science, 1983
- A Decelle, F Krzakala, C Moore, L Zdeborová: **Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications**, Phys. Rev. E, 2011
- B Karrer, MEJ Newman: **Stochastic blockmodels and community structure in networks**, Phys. Rev. E, 2011
- TP Peixoto: **Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models**, Phys. Rev. E, 2014



Social Networks 5 (1983) 109–137
North-Holland

STOCHASTIC BLOCKMODELS: FIRST STEPS *

Paul W. HOLLAND
Educational Testing Service **

Kathryn Blackmond LASKEY and Samuel LEINHARDT
Carnegie-Mellon University †

A stochastic model is proposed for social networks in which the actors in a network are partitioned into subgroups called blocks. The model provides a stochastic generalization of the blockmodel. Estimation techniques are developed for the special case of a single relation social network, with blocks specified *a priori*. An extension of the model allows for tendencies toward reciprocation of ties beyond those explained by the partition. The extended model provides a one degree-of-freedom test of the model. A numerical example from the social network literature is used to illustrate the methods.

1. Introduction

The use of relational data in social science has increased dramatically in the last twenty years. This increase has been driven by both a general theoretical interest in the structural features of networks of relations and applied concerns focusing on the behavioral implications of various structural tendencies and patterns. Methodological tools for the analysis of these data, however, are not well advanced and, consequently, research using relational data tends to be *ad hoc*, nonreplicable and nongeneralizable. This seriously restricts the cumulation of facts about networks of relations that is necessary for the development of empirically verified substantive theory.

There are, however, two relatively new approaches to relational data analysis which are potentially very useful: blockmodels (White, Boor-

* This research was supported by HHS Grant 2 RO1 HD12506-03. The order of authorship is alphabetical.

** Educational Testing Service, Princeton, NJ 08541, U.S.A.
† Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.

Notes: