Universität Zürich[UZH]

Blockchain & Distributed Ledger Technologies

UZH Blockchain Center

Statistical Inference of Networks

Network Science - Lecture 7

Carlo Campajola

Blockchain & Distributed Ledger Technologies Group

Claudio J. Tessone

Lecture Objectives

1. Learn how statistical inference is used in Network Science
2. Understand the Maximum Entropy Principle to generate optimal models
3. Learn how to infer parametric network models from data
4. Learn non-parametric methods of network inference

Inference? Entropy?
Parametric?
Am I in the right classroom?

DON'T PANIC
Everything will make sense

Inference

*It is the process of moving from **premises** to **conclusions***

...ok maybe that's a bit vague

***Statistical inference** is the process of using **data analysis** to characterise the probability distribution that generated observations*

Key goals:

+ try to go beyond simple descriptions like sample averages
+ find and characterise generative mechanisms
+ make predictions/forecasts/generalisations

Inference

Statistical inference *is the process of using* ***data analysis*** *to characterise the probability distribution that generated observations*

In other words:

Get some data
↓
Statistical inference
↓
Make predictions
↓
Draw conclusions

Statistical inference is a data-driven methodology to draw conclusions from empirical observations

When you hear about Machine Learning, AI, etc...
Those are inferential methods

Non-parametric inference

Sometimes you don't have a good model for your data

However, you can still draw conclusions without making modeling assumptions

Examples: Spearman's ρ and Kendall's τ (remember Assignment 3?)

$$\rho(X, Y) = r(\text{rank}(X), \text{rank}(Y))$$

$$\tau(X, Y) = \frac{1}{\binom{N}{2}} \sum_{i<j} \text{sign}(X_i - X_j)\text{sign}(Y_i - Y_j)$$

Parametric inference

We already talked about how to generate a random network

A **random network model** M is fully characterised by

$$\mathbb{P}_M(\mathcal{G}|\Theta)$$

i.e. the probability function of a given network $\mathcal{G}$ under model M with parameters Θ

In general then

Define $M \rightarrow$ assign values to $\Theta \rightarrow$ sample $\mathcal{G}$

Parametric inference

We already talked about how to generate a random network

A **random network model** M is fully characterised by

$$\mathbb{P}_M(\mathcal{G}|\Theta)$$

i.e. the probability function of a given network $\mathcal{G}$ under model M with parameters Θ

What if we change the order?

Observe $\mathcal{G} \rightarrow$ define $M \stackrel{?}{\rightarrow}$ assign values to Θ

Parametric inference

We already talked about how to generate a random network

A **random network model** M is fully characterised by

$$\mathbb{P}_M(\mathcal{G}|\Theta)$$

i.e. the probability function of a given network $\mathcal{G}$ under model M with parameters Θ

What if we change the order?

Observe $\mathcal{G} \rightarrow$ define $M \stackrel{?}{\rightarrow}$ assign values to Θ

*This is what you call **parametric inference***

Maximum Likelihood Estimation

Formalising parametric inference

Let's say we are doing parametric inference at this point. We have:

- a model M
- a set of parameters Θ
- some observed network $\mathcal{G}$

all tied together by some probability function $\mathbb{P}_M(\mathcal{G}|\Theta)$

Formalising parametric inference

Let's say we are doing parametric inference at this point. We have:

- a model M
- a set of parameters Θ
- some observed network $\mathcal{G}$

all tied together by some probability function $\mathbb{P}_M(\mathcal{G}|\Theta)$

Example: the Erdos-Renyi random network model $G(N,p)$

- $M = G(N,p)$
- $\Theta = (N,p)$; N nodes and probability p of link
- $\mathcal{G} = (V,E)$; $N = |V|$ is a fixed parameter, E is random
- $\mathbb{P}_M(\mathcal{G}|\Theta) = p^{|E|}(1-p)^{\frac{N(N-1)}{2}-|E|}$

Formalising parametric inference

Let's say we are doing parametric inference at this point. We have:

- a model M
- a set of parameters Θ
- some observed network $\mathcal{G}$

all tied together by some probability function $\mathbb{P}_M(\mathcal{G}|\Theta)$

Our goal is to find the most probable values of Θ from $\mathcal{G}$

Why? Because they are our **best guess** driven by the data

Formalising parametric inference

Let's say we are doing parametric inference at this point. We have:

+ a model M
+ a set of parameters Θ
+ some observed network $\mathcal{G}$

all tied together by some probability function $\mathbb{P}_M(\mathcal{G}|\Theta)$

Our goal is to find the most probable values of Θ from $\mathcal{G}$

What does it mean **most probable**?

$$\hat{\Theta} = \text{argmax}_{\Theta} \mathbb{P}_M(\Theta|\mathcal{G})$$

Bayes rule

$$\mathbb{P}_M(\Theta|\mathcal{G}) = \frac{\mathbb{P}_M(\mathcal{G}|\Theta)\mathbb{P}(\Theta)}{\mathbb{P}(\mathcal{G})}$$

- $\mathbb{P}_M(\Theta|\mathcal{G})$: the **posterior**, probability that Θ were the parameters generating $\mathcal{G}$ under model M
- $\mathbb{P}_M(\mathcal{G}|\Theta)$: the **likelihood**, probability that under model M with parameters Θ you observe $\mathcal{G}$
- $\mathbb{P}(\Theta)$: the **prior**, probability that parameters Θ are realistic (does **not** depend on data!)
- $\mathbb{P}(\mathcal{G})$ normalises the expression (it's a probability so it sums to 1)

Bayes rule

$$\mathbb{P}_M(\Theta|\mathcal{G}) = \frac{\mathbb{P}_M(\mathcal{G}|\Theta)\mathbb{P}(\Theta)}{\mathbb{P}(\mathcal{G})}$$

- $\mathbb{P}_M(\Theta|\mathcal{G})$: the **posterior**, probability that Θ were the parameters generating $\mathcal{G}$ under model M
- $\mathbb{P}_M(\mathcal{G}|\Theta)$: the **likelihood**, probability that under model M with parameters Θ you observe $\mathcal{G}$
- $\mathbb{P}(\Theta)$: the **prior**, probability that parameters Θ are realistic (does **not** depend on data!)
- $\mathbb{P}(\mathcal{G})$ normalises the expression (it's a probability so it sums to 1)

Bayes rule

$$\mathbb{P}_M(\Theta|\mathcal{G}) = \frac{\mathbb{P}_M(\mathcal{G}|\Theta)\mathbb{P}(\Theta)}{\mathbb{P}(\mathcal{G})}$$

- $\mathbb{P}_M(\Theta|\mathcal{G})$: the **posterior**, probability that Θ were the parameters generating $\mathcal{G}$ under model M
- $\mathbb{P}_M(\mathcal{G}|\Theta)$: the **likelihood**, probability that under model M with parameters Θ you observe $\mathcal{G}$
- $\mathbb{P}(\Theta)$: the **prior**, probability that parameters Θ are realistic (does **not** depend on data!)
- $\mathbb{P}(\mathcal{G})$ normalises the expression (it's a probability so it sums to 1)

Bayes rule

$$\mathbb{P}_M(\Theta|\mathcal{G}) = \frac{\mathbb{P}_M(\mathcal{G}|\Theta)\mathbb{P}(\Theta)}{\mathbb{P}(\mathcal{G})}$$

- $\mathbb{P}_M(\Theta|\mathcal{G})$: the **posterior**, probability that Θ were the parameters generating $\mathcal{G}$ under model M
- $\mathbb{P}_M(\mathcal{G}|\Theta)$: the **likelihood**, probability that under model M with parameters Θ you observe $\mathcal{G}$
- $\mathbb{P}(\Theta)$: the **prior**, probability that parameters Θ are realistic (does **not** depend on data!)
- $\mathbb{P}(\mathcal{G})$ normalises the expression (it's a probability so it sums to 1)

Bayes rule

$$\mathbb{P}_M(\Theta|\mathcal{G}) = \frac{\mathbb{P}_M(\mathcal{G}|\Theta)\mathbb{P}(\Theta)}{\mathbb{P}(\mathcal{G})}$$

- $\mathbb{P}_M(\Theta|\mathcal{G})$: the **posterior**, probability that Θ were the parameters generating $\mathcal{G}$ under model M
- $\mathbb{P}_M(\mathcal{G}|\Theta)$: the **likelihood**, probability that under model M with parameters Θ you observe $\mathcal{G}$
- $\mathbb{P}(\Theta)$: the **prior**, probability that parameters Θ are realistic (does **not** depend on data!)
- $\mathbb{P}(\mathcal{G})$ normalises the expression (it's a probability so it sums to 1)

Maximum Likelihood

$$\mathbb{P}_M(\Theta|\mathcal{G}) = \frac{\mathbb{P}_M(\mathcal{G}|\Theta)\mathbb{P}(\Theta)}{\mathbb{P}(\mathcal{G})}$$

If the problem is finding $\hat{\Theta} = \text{argmax}_\Theta \mathbb{P}_M(\Theta|\mathcal{G})$ it's called **Maximum A Posteriori Estimation** (MAP)

If however there is no prior information (i.e. **uniform prior**) the MAP becomes

$$\hat{\Theta} = \text{argmax}_\Theta \mathbb{P}_M(\Theta|\mathcal{G}) \propto \text{argmax}_\Theta \mathbb{P}_M(\mathcal{G}|\Theta)$$

and becomes **Maximum Likelihood Estimation** (MLE)

Maximum Likelihood: a primer

When you want to find the maximum of a function $f(x)$

- solve $\frac{df}{dx} = 0$ analytically, if possible
- find the maximum numerically

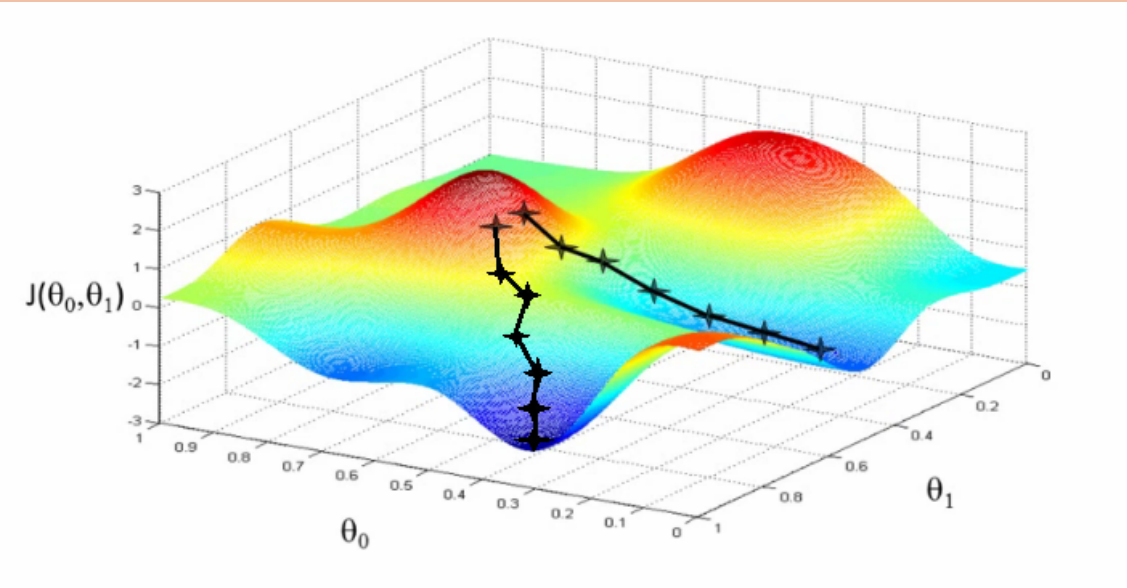
When analytical solutions are not possible, the typical numerical method is **gradient ascent**

Maximum Likelihood: a primer

When you want to find the maximum of a function $f(x)$

- solve $\frac{df}{dx} = 0$ analytically, if possible
- find the maximum numerically

When analytical solutions are not possible, the typical numerical method is **gradient ascent**



Model selection

However we still have one problem:

Sometimes choosing a model M is not easy

We have mentioned several models so far

- Erdos-Renyi
- Configuration Model
- Hidden Parameter Model
- Watts-Strogatz
- Barabasi-Albert
- ...

But these are far from being the only options

Model selection

However we still have one problem:

Sometimes choosing a model M is not easy

We have mentioned several models so far

- Erdos-Renyi
- Configuration Model
- Hidden Parameter Model
- Watts-Strogatz
- Barabasi-Albert
- ...

But these are far from being the only options

We can define our own models for our needs!

The Maximum Entropy Principle

Entropy

Entropy is a measure of uncertainty

It comes from **physics**, where it measures unpredictability in dynamical systems

In **information theory** it was defined in 1948 by Shannon

$$H(X) = -\sum_{i=1}^{n} \mathbb{P}(x_i) \log \mathbb{P}(x_i)$$

and it measures the **average amount of information** carried by realisations of X

Entropy

Entropy is a measure of uncertainty

$$H(X) = -\sum_{i=1}^{n} \mathbb{P}(x_i) \log \mathbb{P}(x_i)$$

The (brilliant) idea is the following:

The least likely an observation, the more I am surprised by it ⇒ ***the more information it carries***

or, in other words,

$$\text{information}(x_i) = -\log(\mathbb{P}(X = x_i))$$

Entropy

Entropy is a measure of uncertainty

this means that, if $X \in \{x_i\}_{i=1,\dots,n}$

$$H(X) = -\sum_{i=1}^{n} \mathbb{P}(x_i) \log \mathbb{P}(x_i)$$

is the average information content of observing one realisation of X

High entropy ↔ high information ↔ ***low prior knowledge about X***

Why entropy

Entropy quantifies our lack of prior knowledge about X before we observe its realisation

So we can use it to find a **model** that makes **minimal assumptions** about the random variable X

Remember: H depends on $\mathbb{P}(X)$. If you change model, you also change $\mathbb{P}$ and then H!

In other words, H is a **functional**: a "function of functions"

Entropy: an example

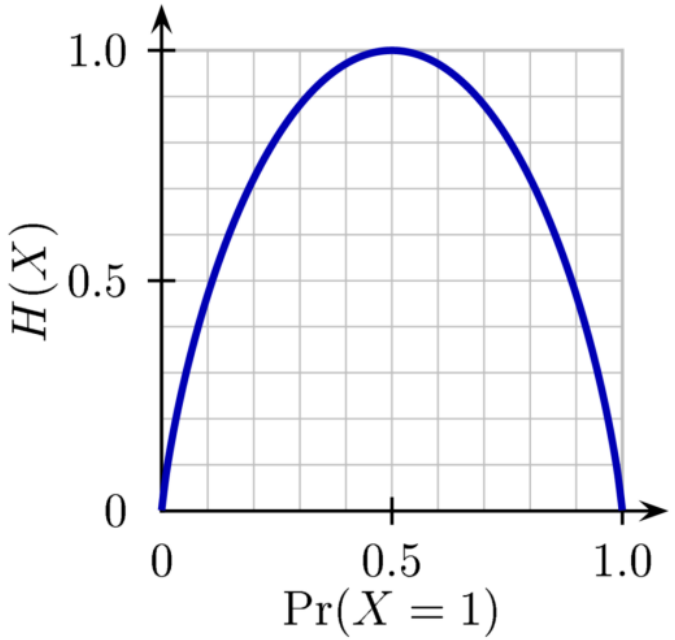
Take X to be a Bernoulli r.v. with success probability p. Then

$$H(X) = -\sum_{x=0}^{1} p(x) \log p(x) = -p \log p - (1-p) \log(1-p)$$

Entropy: an example

Take X to be a Bernoulli r.v. with success probability p. Then

$$H(X) = -\sum_{x=0}^{1} p(x) \log p(x) = -p \log p - (1-p) \log(1-p)$$



The Maximum Entropy Principle

If H is a measure about the information that we do **not** put a priori on X...

*the **best** model is the one that maximises H given what we know about X!*

If we call $\mathcal{C}(M)$ a set of **constraints**, i.e. of conditions we want to hold true for our model M, then

$$\hat{M}_{|\mathcal{C}} = \text{argmax}_M H(X|\mathcal{C}) =$$
$$= \text{argmax}_M \left[-\sum_i \mathbb{P}_M(x_i) \log \mathbb{P}_M(x_i) \right] \quad s.t.\ \mathcal{C}(M) \text{ hold}$$

What does this mean in practice?

A simple example

- You observe a network $\overline{\mathcal{G}}$ with N nodes and $|E(\overline{\mathcal{G}})| = \overline{L}$ edges
- You want to find a **random network model** M such that the average number of links of a sampled network is exactly $\overline{L}$
- This can be written as $\sum_{\mathcal{G}} |E(\mathcal{G})| \mathbb{P}_M(\mathcal{G}) = \overline{L}$
- You want M to be the Maximum Entropy model

So we want to maximise the entropy...

$$\hat{M} = \text{argmax}_M \left[-\sum_{\mathcal{G}} \mathbb{P}_M(\mathcal{G}) \log \mathbb{P}_M(\mathcal{G}) + \right.$$

A simple example

- You observe a network $\overline{\mathcal{G}}$ with N nodes and $|E(\overline{\mathcal{G}})| = \overline{L}$ edges
- You want to find a **random network model** M such that the average number of links of a sampled network is exactly $\overline{L}$
- This can be written as $\sum_{\mathcal{G}} |E(\mathcal{G})| \mathbb{P}_M(\mathcal{G}) = \overline{L}$
- You want M to be the Maximum Entropy model

...we want $\mathbb{P}_M$ to be normalised (it's a probability!)...

$$\hat{M} = \text{argmax}_M \left[-\sum_{\mathcal{G}} \mathbb{P}_M(\mathcal{G}) \log \mathbb{P}_M(\mathcal{G}) + \right.$$

$$+\lambda_0 \left(\sum_{\mathcal{G}} \mathbb{P}_M(\mathcal{G}) - 1 \right) +$$

A simple example

- You observe a network $\overline{\mathcal{G}}$ with N nodes and $|E(\overline{\mathcal{G}})| = \overline{L}$ edges
- You want to find a **random network model** M such that the average number of links of a sampled network is exactly $\overline{L}$
- This can be written as $\sum_{\mathcal{G}} |E(\mathcal{G})| \mathbb{P}_M(\mathcal{G}) = \overline{L}$
- You want M to be the Maximum Entropy model

...and we want the average density to be fixed

$$\hat{M} = \text{argmax}_M \left[-\sum_{\mathcal{G}} \mathbb{P}_M(\mathcal{G}) \log \mathbb{P}_M(\mathcal{G}) + \right.$$
$$\left. + \lambda_0 \left(\sum_{\mathcal{G}} \mathbb{P}_M(\mathcal{G}) - 1 \right) + \lambda_1 \left(\sum_{\mathcal{G}} |E(\mathcal{G})| \mathbb{P}_M(\mathcal{G}) - \overline{L} \right) \right]$$

A simple example

$$\mathcal{F}[\mathbb{P}_M(\mathcal{G})] = -\sum_{\mathcal{G}} \mathbb{P}_M(\mathcal{G}) \log \mathbb{P}_M(\mathcal{G}) +$$
$$- \lambda_0 \left(\sum_{\mathcal{G}} \mathbb{P}_M(\mathcal{G}) - 1 \right) - \lambda_1 \left(\sum_{\mathcal{G}} |E(\mathcal{G})| \mathbb{P}_M(\mathcal{G}) - \overline{L} \right)$$

λ_0 and λ_1 are called **Lagrange Multipliers** and are introduced to solve this **constrained optimization problem** on the new functional $\mathcal{F}[\mathbb{P}_M(\mathcal{G})]$ (called the **Lagrangian**)

How do we solve it?

Just solve $\frac{d\mathcal{F}}{d\mathbb{P}_M} = 0$ *for* $\mathbb{P}_M$*!*

A simple example: solution

It turns out that the solution to this problem is

$$\mathbb{P}_M(\mathcal{G}) = \frac{e^{\lambda_1 |E(\mathcal{G})|}}{\left(1+e^{\lambda_1}\right)^{\binom{N}{2}}} = \left(\frac{e^{\lambda_1}}{1+e^{\lambda_1}}\right)^{|E(\mathcal{G})|} \left(\frac{1}{1+e^{\lambda_1}}\right)^{\binom{N}{2}-|E(\mathcal{G})|} =$$

$$= p^{|E(\mathcal{G})|}(1-p)^{\binom{N}{2}-|E(\mathcal{G})|}$$

Looks familiar?

It's Erdos-Renyi G(N,p) model!!!

Exponential Random Graph Models

A whole new class of models

Turns out Maximum Entropy network models can all be written with the same type of distribution:

$$\mathbb{P}(\mathcal{G}|\Theta) = \frac{1}{Z(\Theta)} \exp\left\{-\sum_i \theta_i x_i\right\}$$

where $\Theta = \{\theta_i\}$ are Lagrange Multipliers (or combinations of them) and x_i are constrained average quantities, such as number of links, degree distribution, ...

*These are called **Exponential Random Graph Models** (ERGMs)*

ERGMs: so easy?

$$\mathbb{P}(\mathcal{G}|\Theta) = \frac{1}{Z(\Theta)} \exp\left\{-\sum_i \theta_i x_i\right\}$$

This looks very good, no? We can define a model and then using **maximum likelihood** we can find the best Θ given our observations $\hat{\mathcal{G}}$

*In principle, **YES***

However, $Z(\Theta)$ is often too hard to compute

Estimating ERGMs

However, $Z(\Theta)$ is often too hard to compute

We have 2 solutions to this problem:

- Restricting ourselves to **solvable** models (where Z is easy to calculate)
- Using numerical methods to approximate it

Solvable models

An ERGM is considered **solvable** if its partition function $Z(\Theta)$ can be computed in **polynomial time**.

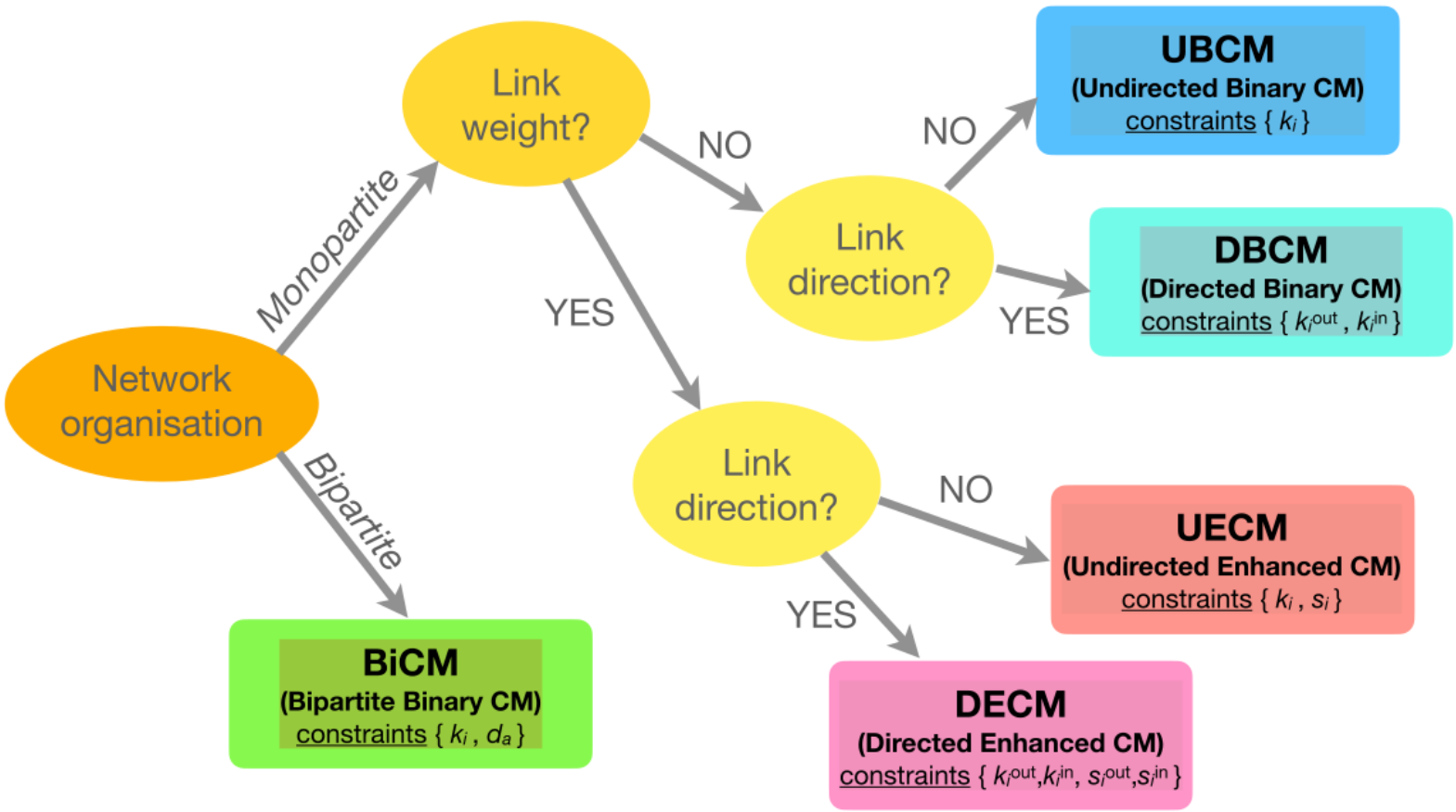
This means that it can be written in a way that does not require $O(e^N)$ operations but only $O(N^\alpha)$ with α independent from N.

Examples:

- $Z(\Theta) = (1 + e^{\lambda_1})^{\binom{N}{2}}$ in the $G(N, p)$ model
- $Z(\Theta) = \prod_{i<j} \left[1 + e^{-\theta_i - \theta_j}\right]$ in the Configuration Model
- ...

Solvable ERGMs: Configuration Models

Several extensions to the CM (see lecture 5!) are actually solvable ERGMs



Undirected Binary Configuration Model

The Undirected Binary Configuration Model (UBCM) is what we already saw in lecture 5. Here the degrees k_i are constrained, leading to a likelihood that reads

$$\mathbb{P}_{UBCM}(\mathrm{A}|\Theta) = \prod_{i<j} p_{ij}^{a_{ij}} (1 - p_{ij})^{1-a_{ij}}$$

where $\mathrm{A} = (a_{ij})$ is the network's adjacency matrix and

$$p_{ij} = \frac{e^{-\theta_i - \theta_j}}{1 + e^{-\theta_i - \theta_j}}$$

Notice: here the degree distribution is fixed, not the exact sequence

Directed Binary Configuration Model

The Directed Binary Configuration Model (DBCM) is the directed network version of the UBCM, with

$$\mathbb{P}_{DBCM}(\mathrm{A}|\Theta) = \prod_{i \neq j} p_{ij}^{a_{ij}} (1 - p_{ij})^{1 - a_{ij}}$$

where $\mathrm{A} = (a_{ij})$ is the network's adjacency matrix and

$$p_{ij} = \frac{e^{-\theta_i - \theta_j}}{1 + e^{-\theta_i - \theta_j}}$$

Notice that here the product runs over $i \neq j$ rather than $i < j$ as in UBCM

Enhanced Configuration Models

The Undirected (Directed) Enhanced Configuration Model (U/DECM) is the **weighted** version of the UBCM (DBCM). A new set of quantities is constrained that is the node **strengths** s_i, i.e. the total weights of i's links
Here $\mathbb{P}_{UECM}(\mathrm{W}|\Theta) = \prod_{i<j} q_{ij}(w_{ij})$ ($i \neq j$ for DECM), where

$$q_{ij}(w_{ij}) = \begin{cases} 1 - p_{ij} \text{ , if } w = 0 \\ p_{ij} e^{-(\beta_i+\beta_j)(w_{ij}-1)} (1 - e^{-\beta_i-\beta_j}) \text{ , if } w > 0 \end{cases}$$

and

$$p_{ij} = \frac{e^{-\alpha_i-\alpha_j-\beta_i-\beta_j}}{1 - e^{-\beta_i-\beta_j} + e^{-\alpha_i-\alpha_j-\beta_i-\beta_j}}$$

Non-solvable models

UBCM, DBCM, UECM and DECM are not the only solvable models. There are (potentially infinitely!) many others.

However not all of ERGMs allow you to compute their $Z(\Theta)$ in polynomial time. What to do then?

Markov Chain Monte Carlo simulations

Monte Carlo simulations

Monte Carlo takes the name from the world-renowned Casino of the Principality of Monaco. This may suggest that it's a gambling technique...
...more or less it is.

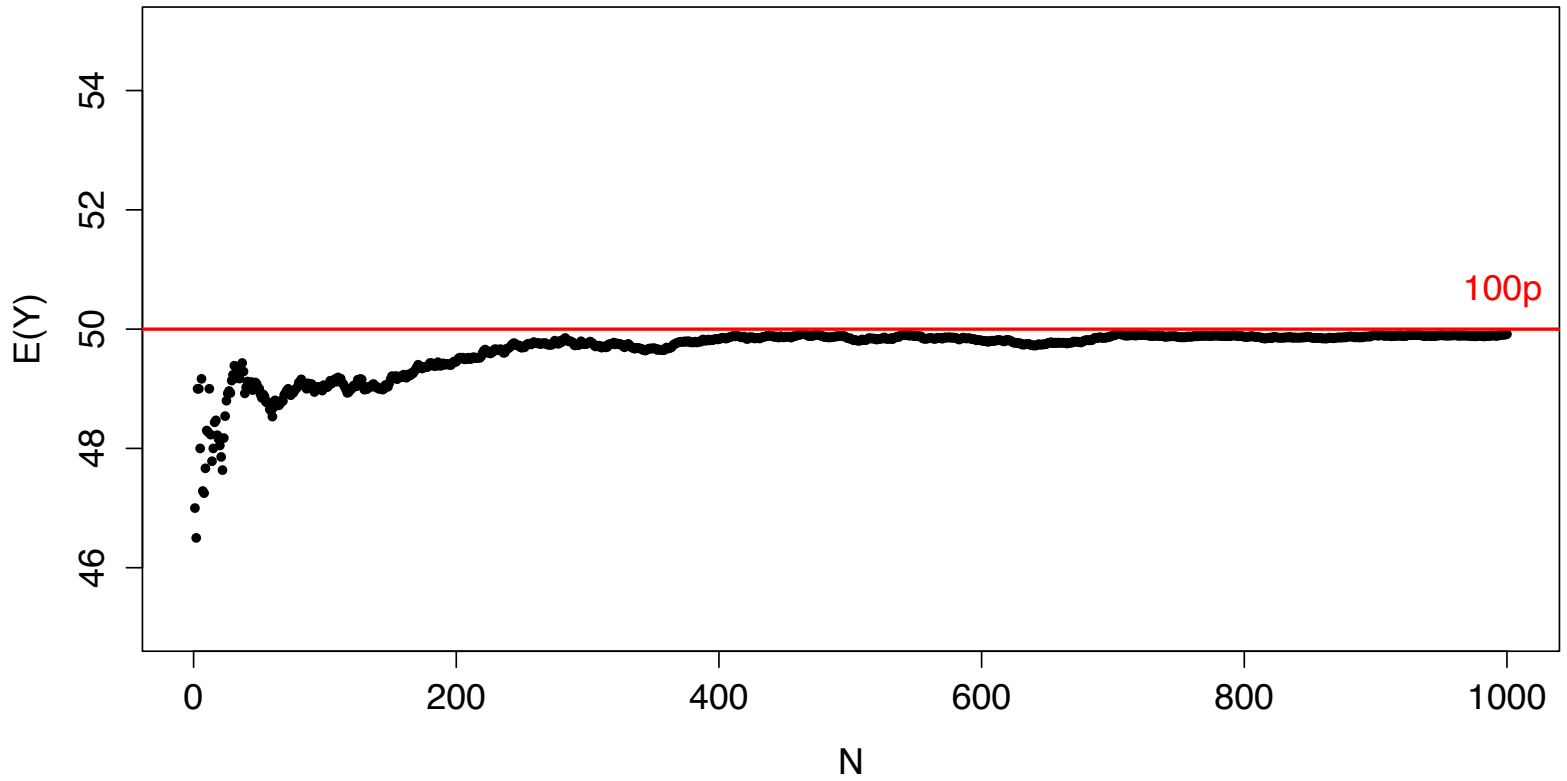
Monte Carlo simulations are about sampling from a given probability distribution to estimate a value

Monte Carlo simulations: example

- Say you have a Bernoulli random variable X, with success probability p
- You want to estimate the average value of a sum of 100 samples of X
- You sample $Y_i = \sum_{i=1}^{100} X_i$ for N times
- Then take $\mathbb{E}(Y_i)$: by the Law of Large Numbers, for $N \to \infty$ this will converge to the correct value

This is a MC simulation. It's quite a useless one, but there are contexts where you have no alternatives

Monte Carlo simulations: example



Markov Chain Monte Carlo

When sampling is not as straightforward as it may be for a series of Bernoulli random variables, the strategy you use for sampling can affect the results.

For ERGMs the fastest and more reliable way is using Markov Chain Monte Carlo (MCMC)

Metropolis-Hastings algorithm:

- Start from a network $\mathcal{G}_0$
- Propose a change: add/remove link (i, j) to generate $\mathcal{G}_1$
- Compute $\mathbb{P}(\mathcal{G}_0)$ and $\mathbb{P}(\mathcal{G}_1)$
- Accept the change with probability $Q = \min\left(1, \frac{\mathbb{P}(\mathcal{G}_1)}{\mathbb{P}(\mathcal{G}_0)}\right)$

MCMC and estimation

How do we use MCMC to estimate an ERGM?

Remember: we need to estimate Θ from a known network $\overline{G}$ maximising $\mathbb{P}(\overline{G}|\Theta) = \frac{1}{Z(\Theta)} \exp\left\{\sum_k \theta_k x_k(\overline{G})\right\}$

- Start from an initial value Θ_0
- Compute the likelihood of Θ_0 using MCMC
- Propose a new value Θ_1
- Repeat until you maximise the likelihood

Statistically Validated Networks

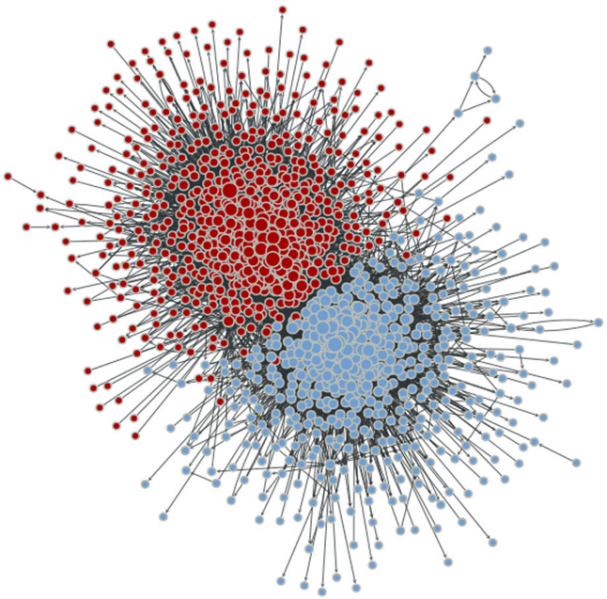
What if I don't have $\overline{G}$

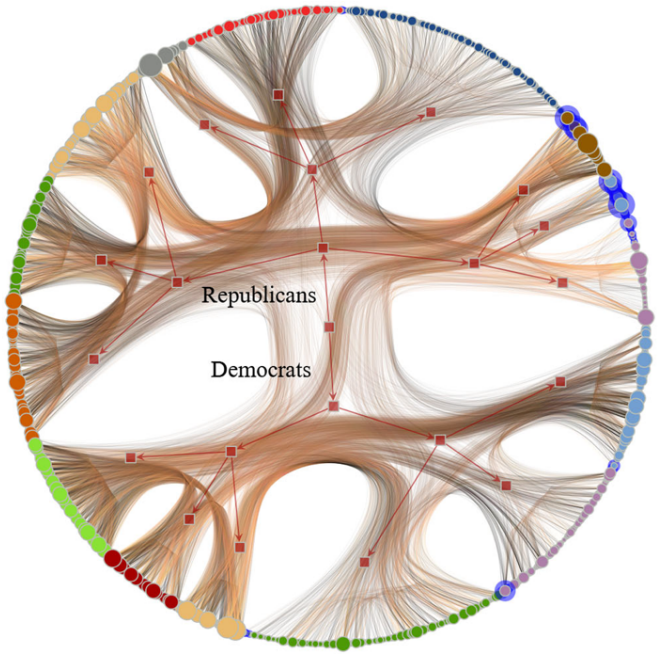
Sometimes you have noisy observations about networks. For example, you may be observing the outcome of a process (e.g. spreading of fake news on a social network) or you may be unsure whether a link is there or not.

You can perform multiple statistical tests to validate your network

What if I don't have $\overline{G}$

You can perform multiple statistical tests to validate your network





An example

Let's consider a **weighted network** of N_A actors

- links (i, j) represent coappearance of actors i and j in movies
- link weight $w_{ij} \in \mathbb{N}$ is the number of movies i and j have co-starred
- let's also call N_M the total number of movies we indexed and N_i the number of movies i starred in

*Question: do some actor pairs co-star movies more often than you would expect? How do we **validate** links?*

An example

Let's do some statistical **hypothesis testing** (lecture 5!)

- Null hypothesis: actors have heterogeneous popularity, but no preferential partners
- Consequence: weights w_{ij} should be distributed proportionally to i and j popularity, i.e. N_i and N_j
- in other words, actors should collaborate just by randomly starring in movies

An example

Let's do some statistical **hypothesis testing** (lecture 5!)

- Null hypothesis: actors have heterogeneous popularity, but no preferential partners
- Consequence: weights w_{ij} should be distributed proportionally to i and j popularity, i.e. N_i and N_j
- in other words, actors should collaborate just by randomly starring in movies

An example

Let's do some statistical **hypothesis testing** (lecture 5!)

- Null hypothesis: actors have heterogeneous popularity, but no preferential partners
- Consequence: weights w_{ij} should be distributed proportionally to i and j popularity, i.e. N_i and N_j
- in other words, actors should collaborate just by randomly starring in movies

If the null is true, the probability that i and j share $w_{ij} = X$ movies is well approximated by a **hypergeometric distribution**

$$\mathbb{P}(X|N_M, N_i, N_j) = \frac{\binom{N_i}{X}\binom{N_M - N_i}{N_j - X}}{\binom{N_M}{N_j}}$$

An example

$$\mathbb{P}(X|N_M, N_i, N_j) = \frac{\binom{N_i}{X}\binom{N_M - N_i}{N_j - X}}{\binom{N_M}{N_j}}$$

Given this, we can get a **p-value** for w_{ij}

$$p(w_{ij}) = 1 - \sum_{X=0}^{w_{ij}-1} \mathbb{P}(X|N_M, N_i, N_j)$$

Based on this p-value then we can decide whether link (i, j) is validated (p < α) or not ($p > \alpha$), and the resulting links form a **Statistically Validated Network**

The problem with p-values

When we do multiple tests (in this case up to N^2) we are subject to **family-wise error rate** (FWER)

FWER is the probability of making at least one Type I error in a family of tests

- Type I error: a False Positive, i.e. rejecting the null when it was actually true
- Family of tests: a set of tests that is made within the same experiment

In our case, we make a test for each actor pair and we are interested in the combined result of these tests (the validated network), so we're subject to FWER!

Understanding FWER

By definition, $FWER = 1 - \mathbb{P}(FP = 0)$.

Let's assume all our N tests yield the same $p = 0.04$. Then for each test we have 4% probability of doing a type I error. This means that

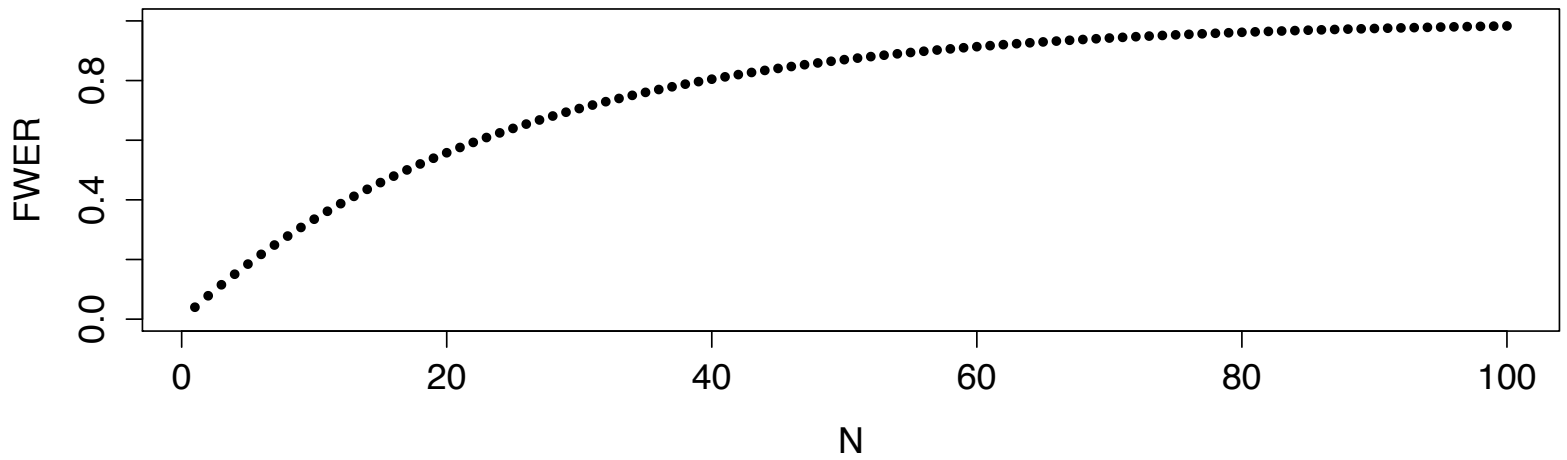
$$FWER(N) = 1 - \mathbb{P}(FP = 0) = 1 - (1 - p)^N = 1 - 0.96^N$$

Understanding FWER

By definition, $FWER = 1 - \mathbb{P}(FP = 0)$.

Let's assume all our N tests yield the same $p = 0.04$. Then for each test we have 4% probability of doing a type I error. This means that

$$FWER(N) = 1 - \mathbb{P}(FP = 0) = 1 - (1 - p)^N = 1 - 0.96^N$$



Correcting for FWER

Statisticians have developed many methods to correct p-values to account for FWER. The goal is to reduce FWER to a fixed threshold α. The most common are

- Bonferroni:
  - rescale all p-values by the number of tests, $p_B = Np$
  - reject nulls that have $p_B < \alpha$
- False Discovery Rate:
  - order p-values from smallest to largest $\{p_{(k)}\}$
  - find the largest k s.t. $p_k \leq \frac{k}{N}\alpha$
  - reject the first k nulls in this ordered sequence

Wrapping up on SVNs

Statistically Validated Networks can be used to create networks out of datasets where the network is not clearly defined

- It is a non-parametric method
- Using FWER corrections it can be very selective or very powerful
- It has been used to model financial networks, phone calls, criminal suspects

References I

- T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 2006.
- D. R. Hunter *Curved Exponential Family Models for Social Networks*, Social Networks 29(2):216:230 2007.
- S. Micciché and R. N. Mantegna, *A Primer on Statistically Validated Networks*, arXiv:1902.07074, 2019.