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 05
Entropy and Description Length Minimization

May 29, 2024

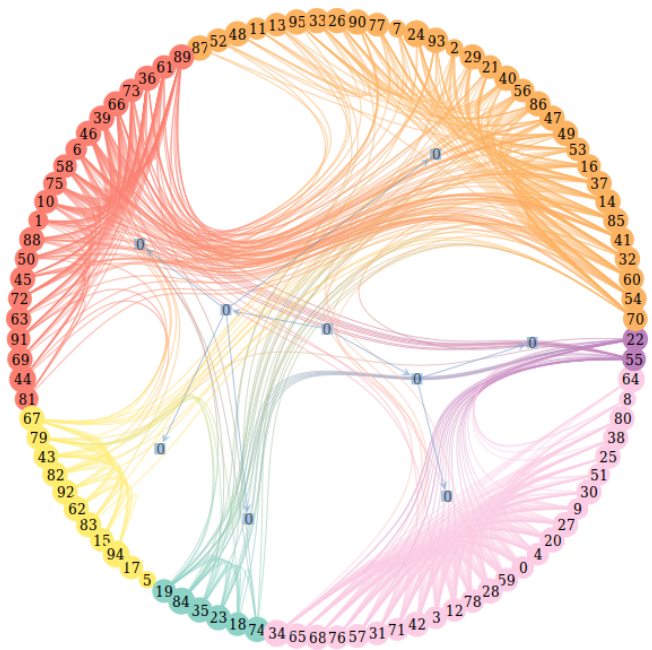
Notes:

- **Lecture L05:** Entropy and Description Length Minimization 29.05.2024
- **Educational objective:** We study the entropy of random graph ensembles. We cover Shannon's Source Coding Theorem, introduce the Minimum Description Length principle and show hot it helps us to avoid overfitting the stochastic block model.
  - Community Detection and Model Selection
  - Entropy and Information Compression
  - Entropy of Random Graph Ensembles
  - Minimum Description Length Principle

- **Exercise sheet 04:** Entropy and Kullback-Leibler Divergence 05.06.2024

Motivation

- we took a **statistical inference** perspective on generative models for random graphs



example: stochastic block model (SBM)

- for parameters $B, \vec{z}$, $\mathbf{M}$, SBM defines ensemble of **random graphs with latent community structure**
- microstate probability yields **likelihood function** of model parameters for given empirical network
- for given B, **optimal assignment of nodes to communities** can be inferred by finding $\vec{z}$ that **maximizes likelihood**

- how can we apply this to detect communities in real networks?

Notes:

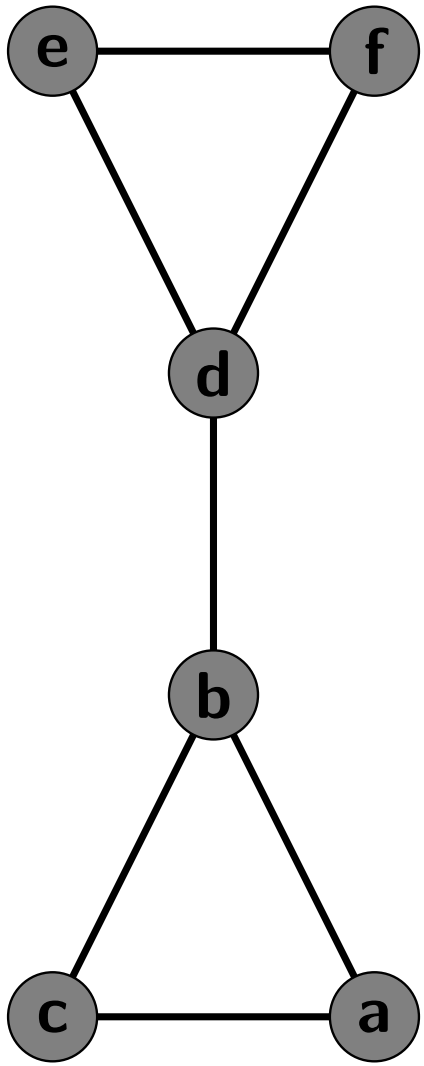
- In the previous two lectures, we took a statistical inference perspective on probabilistic generative models of graphs. Given an empirical network G_e, we formulated the goal to find the parameters of a given generative model that are "most plausible" given our observed network G_e. We exemplified this in an apparently trivial example, where we inferred the "most plausible" parameter of the $G(n, p)$ model. "Fitting" the parameter p to the number of edges in an observed network corresponds to the maximization of the likelihood of the parameter p.
- Last week, we introduced the stochastic block model, which can be used to detect community structures in networks. The stochastic block model is a probabilistic generative model that generates networks based on a given stochastic block matrix $\mathbf{M}$, a block number B, and a block assignment vector $\vec{z}$. Taking an inference perspective, we can detect communities by inferring the most "plausible" block assignment vector $\vec{z}$ for (i) the observed network topology, and (ii) a given number of blocks B.
- So far, we have exclusively assessed the "plausibility" or "explanatory power" of our models in terms of their likelihood. In particular, in our study of inference in the stochastic block model we assumed that we have a priori knowledge about the number of blocks B that we wish to infer. But what if we want to learn the number of communities from the data? This is a challenging issue, which naturally exemplifies the general problem of overfitting in machine learning.

Recap: Inference of block assignments

- consider **given empirical network** G_e and **given number of** B **blocks**
- for each block assignment vector $\vec{z}$ we can calculate maximum likelihood $\hat{\mathcal{L}}(\vec{z})$ of the stochastic block model across all stochastic block matrices $\mathbf{M}$
- for given G_e **optimal partitioning** into B communities is given by vector $\hat{\vec{z}}$ that **maximizes model likelihood**, i.e.

$$\hat{\vec{z}} := \underset{\vec{z}}{\operatorname{argmax}}\, \hat{\mathcal{L}}(\vec{z})$$

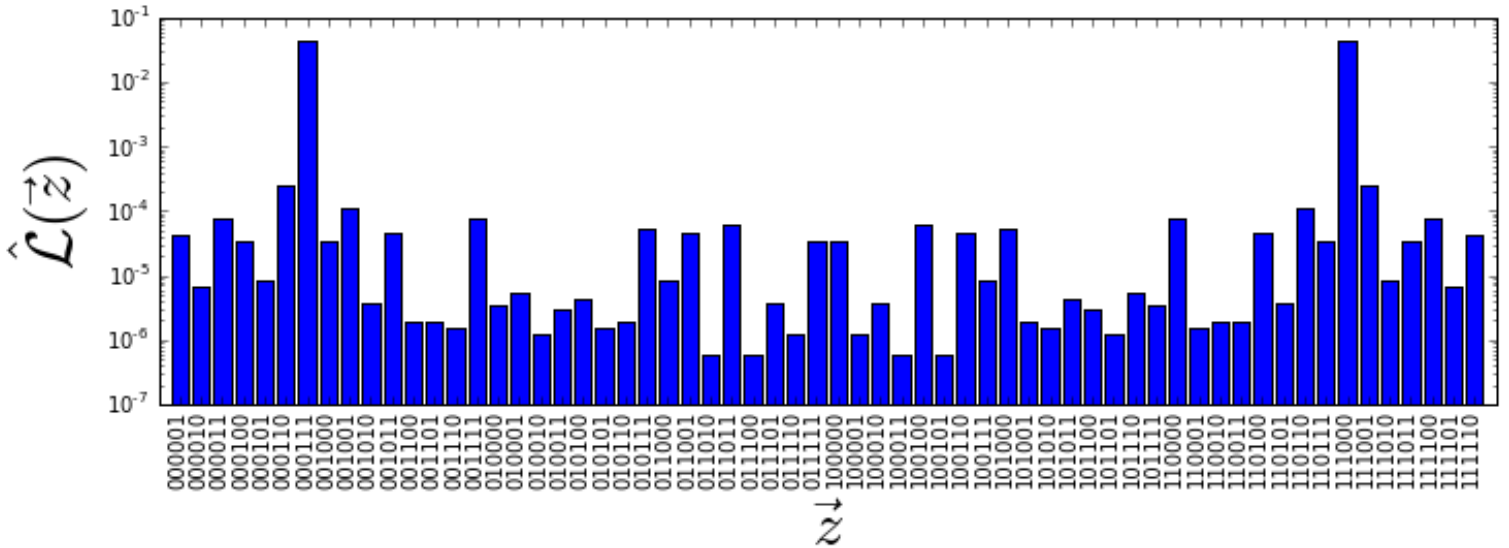
- no analytical solution but we can use **efficient heuristic optimization** algorithms



Notes:

- Before explaining overfitting in the stochastic block model, we briefly recap how we can use the model to infer community structures. For a given empirical network, we have (so far!) fixed the number of communities (or blocks) B that we want to infer. Using the derivations from L04, we can then calculate the maximum likelihood entries of the stochastic block matrix and the resulting maximum model likelihood $\mathcal{L}(\vec{z})$ for any given block assignment vector $\vec{z}$ (i.e. the maximum likelihood across all possible stochastic block matrices $\mathbf{M}$).
- To find the "most plausible" community structures given the observed network, we must now find the block assignment vector $\vec{z}$ which maximizes the likelihood of the observed network G_e.
- In the previous lecture (and exercise) we explored a heuristic optimization algorithm that can be used to quickly find an approximate solution for this optimization problem.

Maximizing likelihood



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

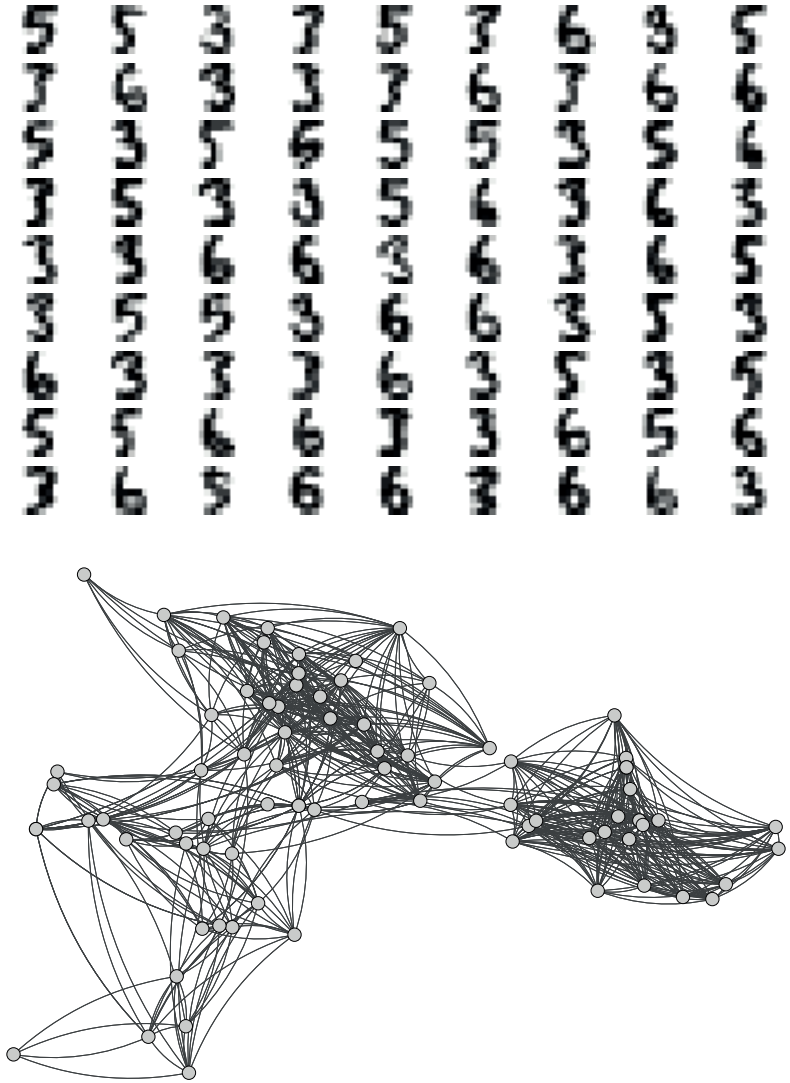
for known number of blocks $B = 2$ we find that $\hat{\mathcal{L}}(\vec{z})$ is maximal for $\vec{z} = (0, 0, 0, 1, 1, 1)$ and $\vec{z} = (1, 1, 1, 0, 0, 0)$

what if we **do not know the number of communities**, i.e. B is unknown?

Notes:

- For the toy example in the lecture, we could simply apply a brute-force approach, i.e. here we can enumerate all possible block assignment vectors $\vec{z}$ and calculate their likelihood. This network has six nodes and each node can be assigned to one of two different blocks. Each block assignment can then be represented as a 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) along with the corresponding estimate of the stochastic block matrix.
- The plot is symmetric, because the left and right half of the assignment vectors represent the same block assignments with flipped binary block labels. We found two maxima for the (flipped) block assignments $(0, 0, 0, 1, 1, 1)$ and $(1, 1, 1, 0, 0, 0)$, which corresponds to the natural community structure of this network.
- So far, we made the unrealistic assumption that we know the number of blocks B from which we can choose entries of the block assignment vector. However, deciding how many communities or clusters exist in a network is a difficult task by itself. You may argue that we could again use a heuristic optimization algorithm to find the number of blocks $\hat{B}$ that maximizes the likelihood. Unfortunately, it is not that simple. Understanding why naturally lead us to problems of model selection, compression, information theory, and –in fact– crucial issues in epistemology and philosophy of science.

How many characters are there?

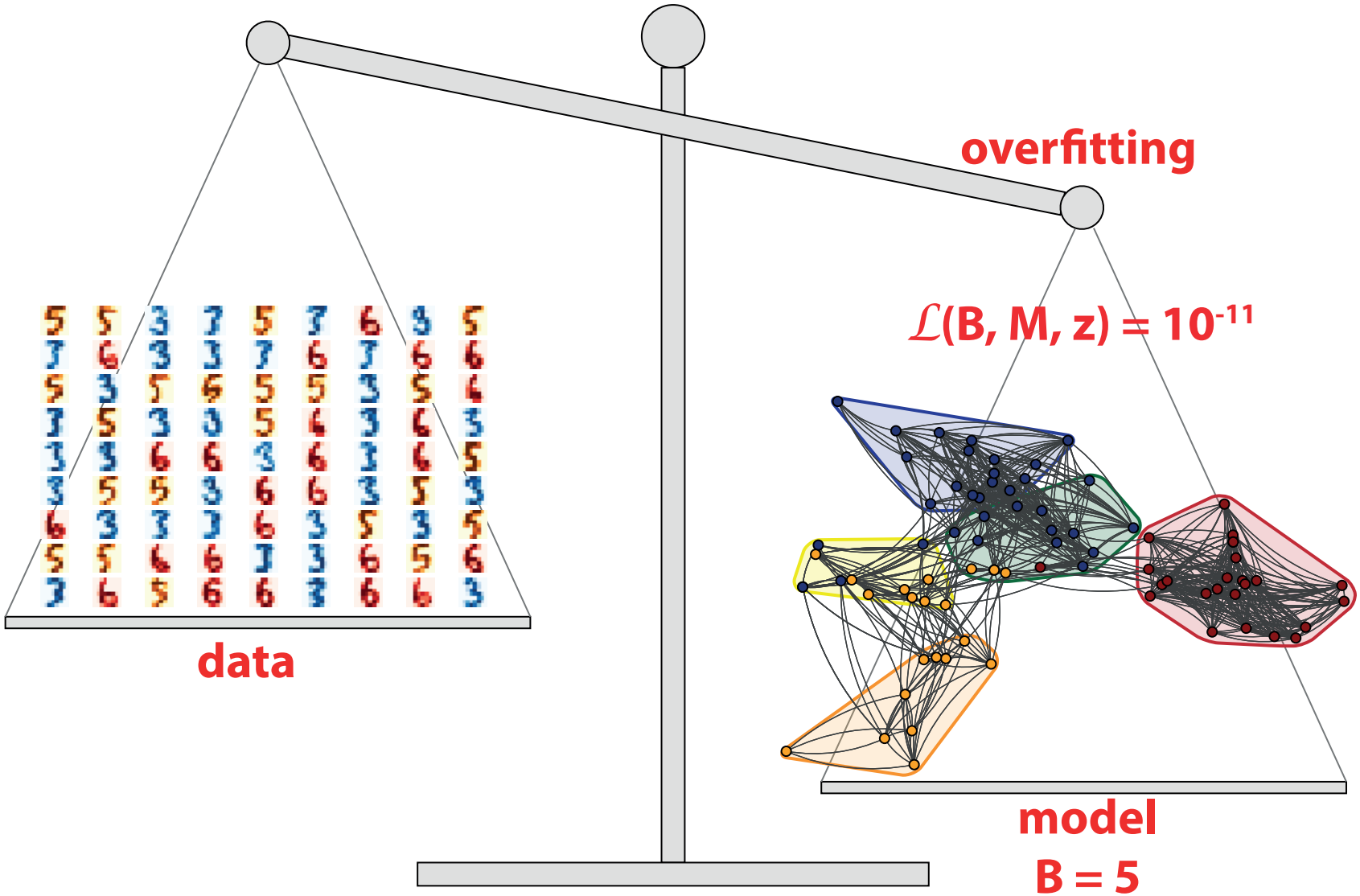


graph where nodes represent 64 **images of three hand-written characters**, links created based on Euclidean distance between images in two-dimensional feature space

Notes:

- This problem has a clear implication for an exemplary application in optical character recognition. Here we have a set of hand-written characters, which are represented by nodes in a graph. Using dimensionality reduction techniques like PCA (more on this in the next lecture), we can represent each image with a two-dimensional coordinate and create links between the corresponding nodes whenever the Euclidean distance is smaller than a given threshold. THe problem of finding groups of images that show the same digit can now be phrased as a community detection problem.
- If we are not able to automatically infer the correct number of communities, we are unable to infer the correct number of different characters that are shown in the collection of images. We are thus also not able to correctly group those images that show the same characters.
- In our example, overfitting the stochastic block model to the network structure corresponds to grouping images according to similarities that are not due to the different characters displayed (e.g. example characters written by different people, different ways to write characters, different pens, etc.).

What is the optimal model?



Notes:

- The problem of finding an optimal model that neither overfits nor underfits the data is a key challenge in community detection (and more general in clustering algorithms and machine learning).
- It is easy to see the implications of such an overfitting in our character recognition example. We could use **two communities** to explain the community structure in our networks, however the likelihood of our model (i.e. the explanatory power of our model for the data) is rather low. This is a sign that **we have underfitted our model**.
- **Increasing the number of clusters to three** massively increases explanatory power, but also increased the complexity of our model. Here, this additional complexity is justified by the actual pattern that we are interested in, i.e. we have found a better model for our data.
- The problem is that we can further increase the complexity of our model (i.e. setting $B = 10$). This trivially increases the likelihood of our model, since we can more easily adjust the parameters to the data. This comes with the **risk of overfitting our model**. In our example, we see that the clusters found by such a model do not correspond to actual digits anymore, i.e. the **model fails to extract knowledge from the data**.

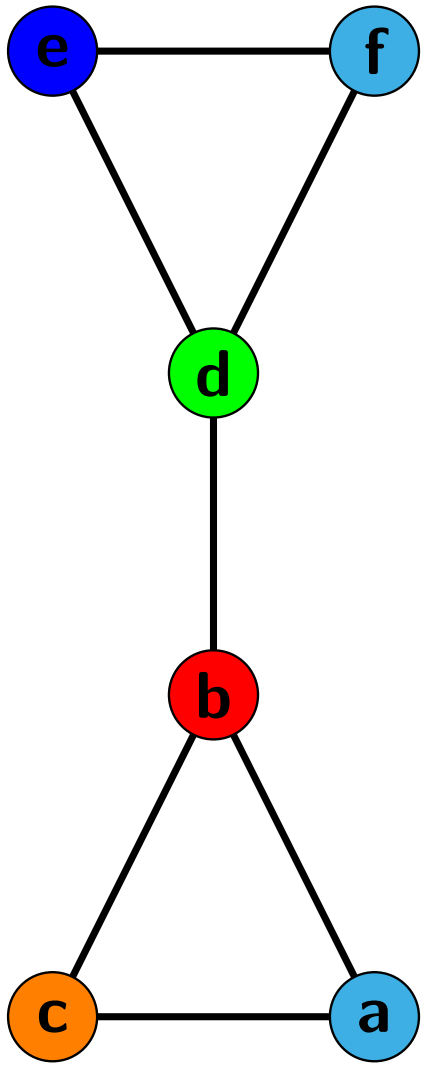
Maximizing likelihood?

- we can set $B = |V|$ and $\vec{z} = (0, 1, 2, 3, 4, 5)$
- with $\hat{M}_{kl} = \frac{E_{kl}}{N_{kl}}$ we obtain the following stochastic block matrix

$$\hat{\mathbf{M}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

with model likelihood $\mathcal{L}(\vec{z}) = 1$

- **inferred model exactly corresponds to data!**



Notes:

- Consider our example network and let $B = |V|$, i.e. we assign each node i its own block i. We now have $N_k = 1$ and thus $N_{kl} = 1$ for $k \neq l$ and $N_{kl} = 0$ for $k = l$. E_{kl} is either 1 if a link between the nodes k and l (corresponding to blocks) exists, or 0 if this link does not exist. By definition we have $\widehat{\mathbf{M}} = \mathbf{A}$, i.e. we obtain a model that generates a link iff a link is present in the empirical network. For this model, we trivially obtain a likelihood of 1 since the empirical network is the **only** network that the model can generate. Enumerating over all possible B and choosing the one that maximizes likelihood is thus not a solution to find reasonable community structures.
- This is a paradigmatic example for "overfitting", i.e. rather than describing those "relevant patterns" that we are interested in (communities) we have "learned" a model captures all "fluctuations" in the data (here, the individual links in the network).

Theory choice

- multiple **different** theories/models can be consistent with phenomenon

astronomical example

alternative explanations for **retrograde motion of planets**

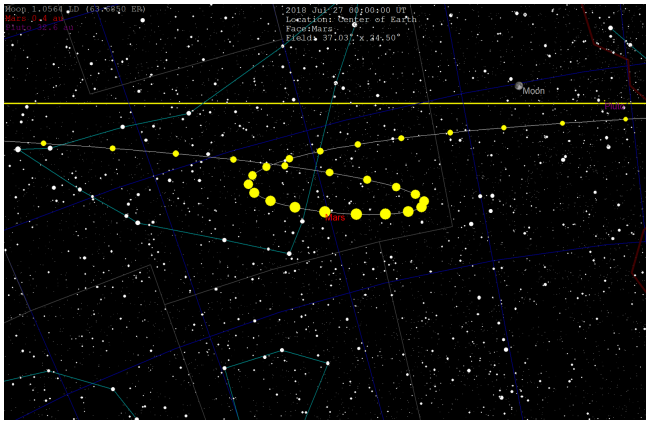
- heliocentric model

  $\longrightarrow$ Copernicus $\longrightarrow$ Kepler $\longrightarrow$ Newton

- geocentric model with epicycles

  $\longrightarrow$ ca. 300 BC – ca. 1680 AD

- which theory/model should we choose?
- theory that best matches the data?
- model with maximum likelihood?



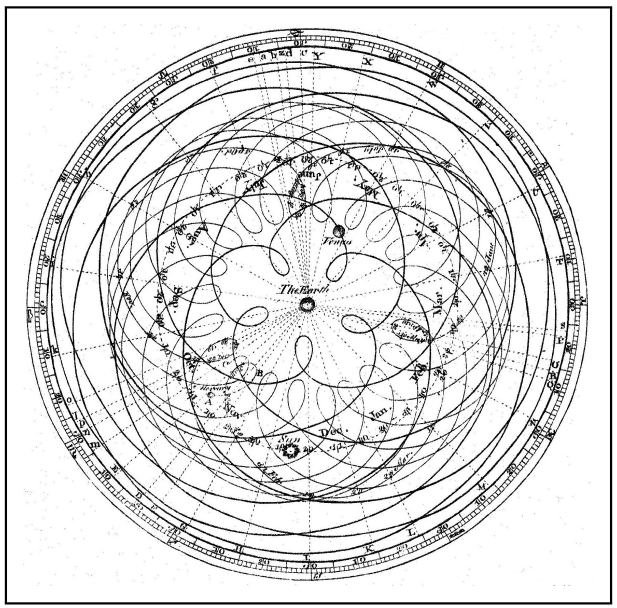
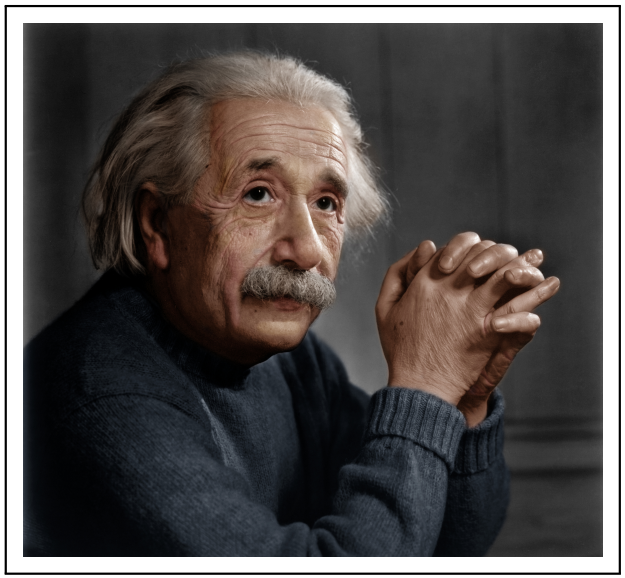


image credit: top: by Tomruen, CC-BY-SA, bottom: Encyclopaedia Britannica (1st Edition, 1771), public domain

Notes:

- This issue is linked to a fundamental and frequently misunderstood challenge in science (that has also important consequences for society): we often have multiple possible explanations (i.e. models or theories) for a given phenomenon and we need methods to chose between them. In astrophysics, a historically important example for this are early models to explained the *retrograde motion of outer planets* (which is due to fact that both earth and the outer planets orbit around the sun).
- The retrograde motion of outer planets can easily be understood based on a heliocentric model. However, in previous centuries some astronomers were reluctant to give up the geocentric view of the universe, which can be used to describe the motion of the inner but not the outer planets. They came up with alternative models in which planets and the sun move around earth on complicated epicycles. Compared to a heliocentric model, this epicycle model allows for very complex planetary motions that initially described the observed motion of planets more accurately than a heliocentric model (where planets move on idealized spherical orbits).
- So, which model should we choose? The one that best matches the observation? The problem of the epicycle model is that it “buys” the large degree to which explains reality with complexity, i.e. we need to make assumptions about many different parameters (i.e. the different epicycles of planets) while the heliocentric model uses much simpler assumptions (and fewer parameters). The degrees of freedom that we gain in terms of the choice of parameter values naturally allow us to better fit the model to reality. The model makes better predictions because we have encoded knowledge about the phenomenon that we seek to explain into the model.

Model selection



"Make things as simple as possible, but not simpler."

Albert Einstein

"Pluralitas non est ponenda sine necessitate."

William of Ockham

key challenge in machine learning

we need methods to balance **explanatory power** and **complexity** of inferred (or learned) models

Notes:

- A **key challenge in machine learning is to find an optimal balance between model complexity and explanatory power**. This is one of the main benefits of human intuition (although even humans sometimes have a hard time at this). Machine learning requires creative algorithmic solutions to this problem, e.g. using methods from statistics, information theory that incorporate and operationalize ideas from philosophy of science.

Model selection

- **principle of parsimony**
  - out of multiple competing theories consistent with the data, we prefer those that make the **least assumptions**
  - **Occam's razor**

> With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.
>
> → John von Neumann

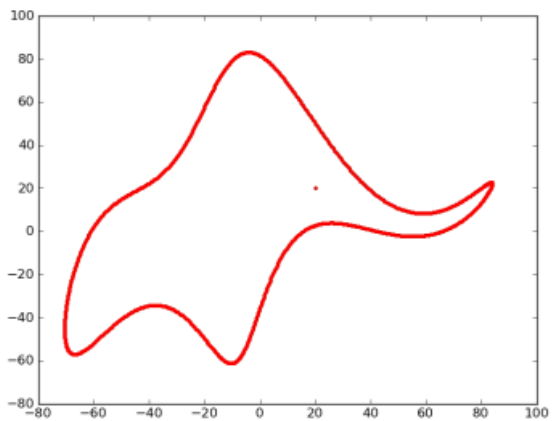
William of Ockham

1288 – 1347

> "Pluralitas non est ponenda sine necessitate."

- for stochastic block model **with larger number of blocks** B it is easier to find parameters with larger likelihood

- to assess how "plausible" a model or theory is **we must account for its complexity**



→ J Mayer et al, 2010

Notes:

- In science, we are generally interested in models that make as few assumptions as possible and still explain the phenomenon (possible with a little less accuracy). This principle is called Occam's razor. This is related to "Occam's razor" in philosophy of science. The Latin phrase translates to "plurality should never be posed without necessity" and it is attributed to the 13th century scholar William of Ockham (who was a role model for the character "William of Baskerville" in Umberto Eco's brilliant novel "The Name of the Rose").
- Balancing the complexity of a model (in terms of how much information we use in the modelling process) with its explanatory power (i.e. how accurately it predicts future observations) is – I would argue – the fundamental challenge in data science and machine learning (and, in fact, the Achilles heel of many techniques).
- This general issue finds its direct expression in the stochastic block model: For $B = |V|$ we have inferred a model (the stochastic block matrix) for our data that is just as complex as the data that we seek to explain (i.e. the adjacency matrix). Such a model is useless, as it does not provide a description of the phenomenon that is more compact than the phenomenon itself.
- We can address this problem by means of information theory, i.e. we are interested in a model that maximally "compresses" the network topology. But how can we quantify (and optimize) such a compression?

Entropy and Information

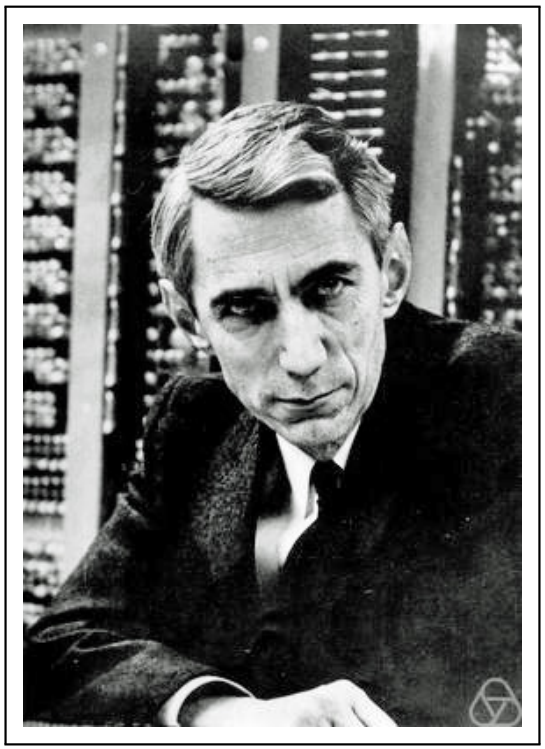
entropy

- Let X be a discrete random variable with pmf $P(X)$.
  Shannon entropy $H(X)$ (in bits) is defined as

$$H(X) := -\sum_i P(X = i) \log_2 P(X = i)$$

- Let X be a continuous random variable with pdf f.
  Differential entropy $H(X)$ (in nats) is defined as

$$H(X) := -\int_X f(x) \ln f(x) dx$$



Claude Eldwood Shannon

1916 - 2001

image credit: Konrad Jacobs, Oberwolfach Photo Collection, CC-BY-SA

- $-\log_2 P(X = i)$ = **information content** of i
- $H(X)$ = **expected information content**
- $H(X)$ = **lack of information about X**

Notes:

- To quantify information, we consider entropy, which is a central concept in probability theory, information theory, and —in fact— statistical physics. It also directly related to Bayesian approaches to statistical inference (more on this in a future course).
- We define the Shannon entropy of a (discrete) probability mass function as the probability weighted sum of the **information content** of all events, where the information content of an event is the negative logarithm of its probability. The idea behind the definition of information content is that it measures our level of surprisal if we see the event. For an event of which we know for sure that it will happen (i.e. $P(X) = 1$) there is no surprisal, so the entropy is zero. The smaller the probability, the larger our surprise if the event occurs. While we can use any logarithm base, for the discrete case we typically use the base two in which case the unit is **bits**.
- We can generalize this in a straight-forward way to differential entropy of a (continuous) probability density functions. Here, we use the natural logarithm (base e), which means the unit of differential entropy is **nats**. Note that this simple generalization of entropy to a continuous distribution introduces issues: First, since a pdf is not necessarily assuming values between 0 and 1, differential entropy can assume negative values. Second, the maximum differential entropy of a distribution can be infinite. And third, the differential entropy is not the limit of Shannon entropy for $n \to \infty$. Other notions of entropy for continuous probability distributions have been proposed and studying their properties is still an active area of research.

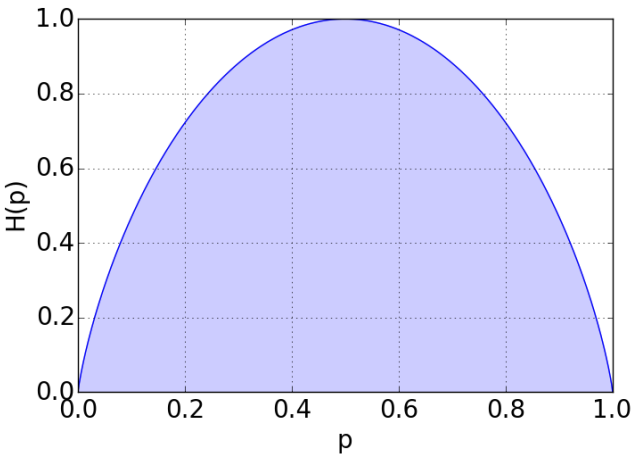
Information of a coin toss

example

- consider single toss of a coin, i.e. random variable X with $P(X = 1) = p$.
- **Shannon entropy of** X is

$$H(p) := H(X) = -p \log_2 p - (1 - p) \log_2 (1 - p)$$

- for single coin toss **maximum entropy of one bit** is reached for $p = \frac{1}{2}$
- **maximum entropy** is assumed if all possible states are equiprobable, i.e. we have no knowledge about the outcome



entropy of random variable X for different values of p

Second Law of Thermodynamics

- entropy of isolated systems never decreases over **time**
- thermodynamic equilibrium = maximum entropy

Notes:

- To illustrate entropy consider a simple coin tossing experiment, where we have two possible outcomes (heads and tails) and a probability p of one of the outcomes.
- The plot on the right shows the entropy of the resulting random variable as a function of the probability p. For a single coin toss, a maximum entropy of 1 is reached for $p = \frac{1}{2}$, i.e. when both events have the same probability. In general the maximum possible (differential) entropy in a given sample space is assumed by the uniform distribution where all outcomes have the same probability.
- An intuitive interpretation of the uniform distribution is that it maximizes our lack of knowledge (or minimizes our prior knowledge) about the outcome of an experiment. From a different point of view, this maximizes the expected information content of a single coin toss (which is 1 bit).

Entropy and Information Compression

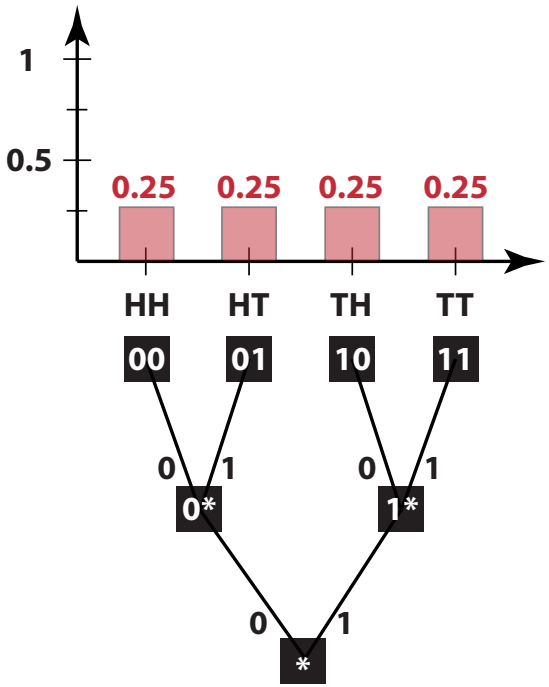
- how can we optimally code a sequence of random events?
- consider **toss of two fair coins**
- we require **2 bits to encode outcome**
- let L_n be the **number of bits needed on average** to store sequence of n outcomes

$$L_n = 2 \cdot n$$

- we have **no prior knowledge** about the outcome that could be used to **more efficiently encode sequence**

P(H) = 0.5

P(T) = 0.5



example for coding tree

$0.25 \cdot 2 + 0.25 \cdot 2 + 0.25 \cdot 2 + 0.25 \cdot 2 = 2$ bits

Notes:

- Entropy is intrinsically related to our ability to compress data, which –as we shall see in the remainder of this and the following lecture– can be used for model selection and community detection in networks.
- Entropy can be viewed as a measure for the number of bits that we expect to be minimally required to store information on a given phenomenon. If heads or tails are equally likely, we minimally require one bit to store each event in a sequence of heads and tails outcomes (i.e. no compression is possible). If heads is the only possible outcome (i.e. entropy zero), we require zero bits to store information about the outcomes (i.e. perfect compression). If either heads are tails are more likely, we can construct encodings that need less than 1 bit on average to store a sequence of coin tosses (due to the higher than random level of repeated entries in the sequence).
- To better understand this, consider a random trial where we toss two fair coins, i.e. we have four possible outcomes, each occurring with probability 0.25. For each outcome we need two bits and we can thus calculate the average number of bits needed to store the outcome as $4 \cdot 0.25 \cdot 2 = 2$ bits. If we chose an encoding where we assign one of the four outcomes a shorter code (i.e. only one bit), we might be more efficient to encode some sequences of coin tosses, but –given that all outcomes are equally likely– we would not be better on average. We can calculate the **expected code length to store a random sequence of events** by multiplying the probability of events with their respective code lengths. Here we find that, for a sequence of outcomes of this trial, we need two bits to encode the outcome for each toss of the two coins.

Huffman Code

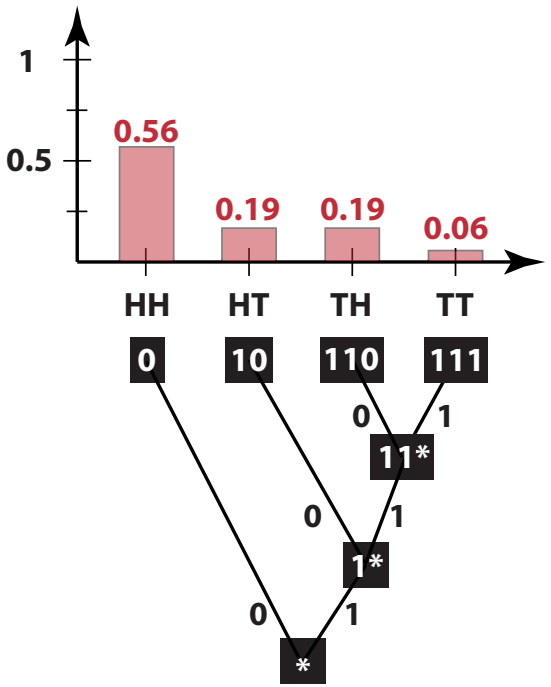
- consider **toss of two biased coins** with $P(H) = 0.75$
- idea: encode more frequent events by shorter bit sequence

P(H) = 0.75

P(T) = 0.25

Huffman coding

Huffman code is a **variable-length prefix-free coding scheme** for **lossless encoding** of sequences with minimal redundancy → DA Huffman, 1952



- with Huffman code we expect ≈ 1.7 **bits to encode outcome** of each toss

$$L_n \approx 1.7 \cdot n$$

- we **used prior knowledge** about outcome to compress the sequence

Huffman coding tree

$0.56 \cdot 1 + 0.19 \cdot 2 + 3 \cdot (0.19 + 0.06) \approx 1.7$ bits

Notes:

- Now consider the toss of two unfair coins where different faces show up with different probabilities. Here we can decrease the expected code length by using a variable-length code that assigns shorter codes to those events that are more frequent. This approach allows us to use prior knowledge about the outcome of the experiment to encode outcomes more efficiently. In the example above, if heads shows up with probability 0.75 we know that we are much more likely to observe the outcome of two heads, compared to two tails. We can accommodate for this knowledge by assigning a shorter encoding (one bit) to the most frequent event, while assigning a longer encoding (three bits) to the two least frequent events. This allows us find an encoding with an expected per-symbol code length of 1.7 bits, i.e. we have compressed the information.
- A particularly simple approach to generate such an encoding is the so-called Huffman coding scheme. Given the probabilities (i.e. relative frequencies) of symbols that occur independently in a sequence, it constructs an **optimal prefix-free coding tree** like the one shown on the right. Importantly, despite symbols having variable lengths, the code above does not require additional prefixed that indicate the start of a symbol. We can directly distinguish symbols without a delimiter, since each zero or triplet of ones terminates a symbol.
- Huffman coding is used in a number of lossless (and lossy) compression schemes, including JPEG or MP3. A description of the (relatively simple) algorithm to construct a prefix-free Huffman code based on the relative frequency of symbols can be found in many textbooks, as well as in → DA Huffman, 1952 . We will implement and demonstrate it in the first practice session.

Shannon's Source Coding Theorem

P(H) = 0.75

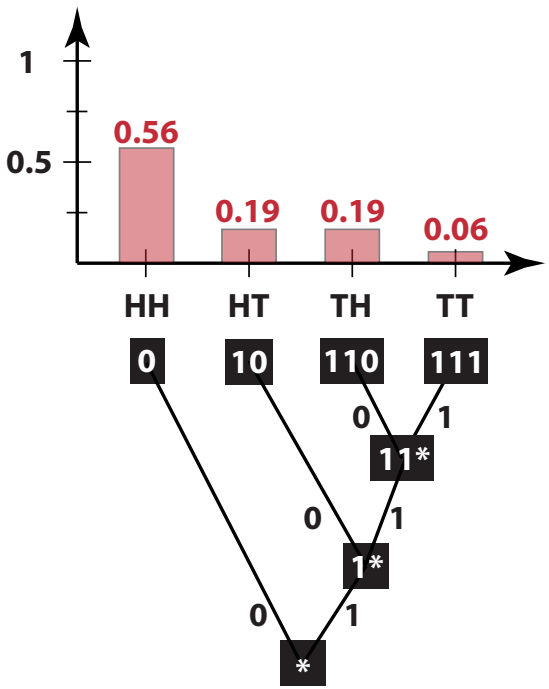
P(T) = 0.25

Shannon's Source Coding Theorem

Let X_i be a sequence of n i.i.d. random variables with entropy $H(X_i)$. For any $\epsilon > 0$ and sufficiently large n there exists a coding scheme that encodes a sequence of n realizations X_i in terms of $n \cdot H(X_i)$ bits such that the sequence can be recovered with probability larger than $1 - \epsilon$. → CE Shannon, 1948

- entropy = **lower bound for expected per-symbol code length** for sequence of n symbols ($n \to \infty$)
- Huffman code is **asymptotically optimal** for n i.i.d. symbols with known prob. mass function P, i.e.

$$L_n \to n \cdot H(P) \qquad \text{for } n \to \infty$$



Huffman coding tree

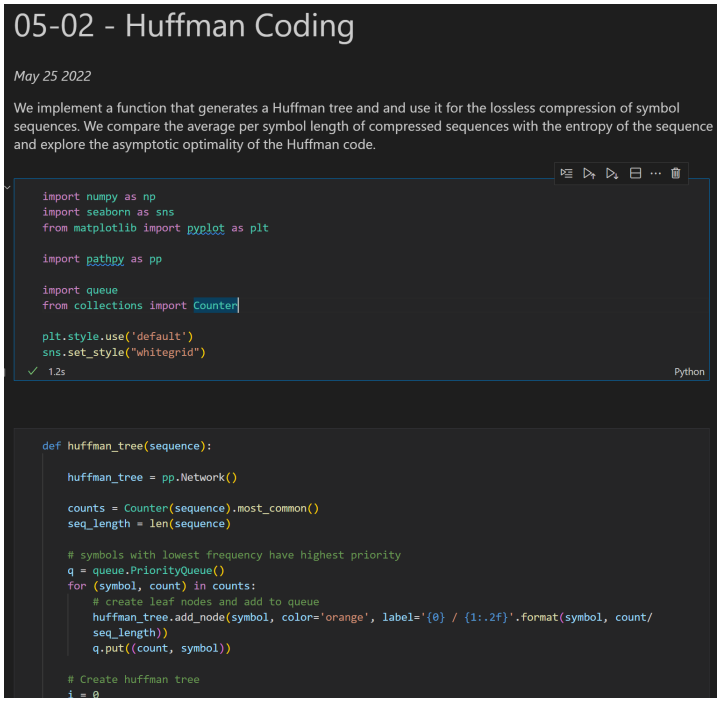
$$H(P) = -0.56 \log_2 0.56 - 2 \cdot (0.19 \log_2 0.19) - 0.06 \log_2 0.06 \approx 1.62 \text{ bits}$$
$$L_1 = 1.7 \text{ bits} > H(P)$$

Notes:

- The exact expected code length depends on the actual coding scheme. So a natural question that arises is how much we can compress information. Information theory teaches us that there is a lower bound for the minimal expected per-symbol code length beyond which we will necessarily lose information. That is, any lossless coding scheme, i.e. a coding scheme that allows us to fully recover the original information, can not be more efficient than this lower bound.
- This lower bound is derived in Shannon's source coding theorem and it corresponds to the Shannon entropy of the probability mass function of the symbols. For a fair coin, this tells us that no coding can be – on average – more efficient than a coding that uses one bit for each coin toss. For our tossing of two unfair coins where heads on each coin shows up with probability 0.75, we we obtain a minimal expected code length of 1.62 bits.
- For the Huffman code, one can show that the obtained coding is asymptotically optimal, i.e. for $n \to \infty$ the expected per-symbol code length in a sequence of n symbols converges to the minimum given by Shannon's source coding theorem. For a sequence of n tosses of our two coins, the Huffman code above has an expected length of 1.7 bits per toss and this expected code length converges to 1.62 bits per toss for $n \to \infty$, i.e. to the lower limit given by Shannon's source coding theorem.

Practice session

- we calculate the **entropy of random variables**
- we implement **Huffman coding** and use it for the lossless compression of text
- we explore **Shannon's Source Coding theorem** and test the optimality of Huffman coding
- we define the **Kullback-Leibler divergence** measure



05-02 - Huffman Coding

May 25 2022

We implement a function that generates a Huffman tree and and use it for the lossless compression of symbol sequences. We compare the average per symbol length of compressed sequences with the entropy of the sequence and explore the asymptotic optimality of the Huffman code.

```
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

import pathpy as pp

import queue
from collections import Counter

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

✓ 1.2s Python

```
def huffman_tree(sequence):

    huffman_tree = pp.Network()

    counts = Counter(sequence).most_common()
    seq_length = len(sequence)

    # symbols with lowest frequency have highest priority
    q = queue.PriorityQueue()
    for (symbol, count) in counts:
        # create leaf nodes and add to queue
        huffman_tree.add_node(symbol, color='orange', label='{0} / {1:.2f}'.format(symbol, count/
        seq_length))
        q.put((count, symbol))

    # Create huffman tree
    i = 0
```

practice session

see notebooks 05–01 and 05–02 in `gitlab` repository at

→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks`

Notes:

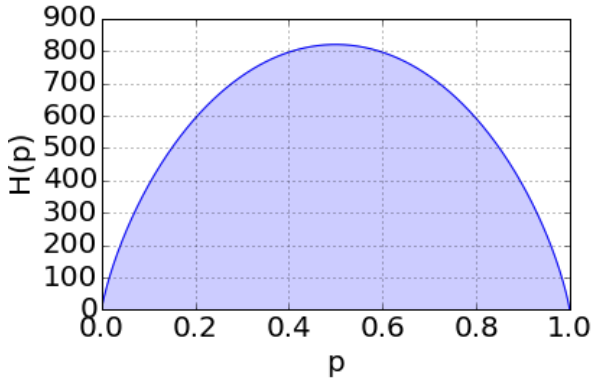
- In the first practice session, we explore the concept of entropy, Huffman coding, and information compression.

Entropy of random graphs: $G(n, p)$ model

- consider **entropy of $G(n, p)$ model**

$$H(n, p) = -\sum_{g \in \Omega(n,p)} P(G) \cdot \log P(G)$$

where $P(G)$ is the **microstate probability**



- summing over all numbers of m links, we can **calculate Shannon entropy** as

$$H(n, p) = -\sum_{m=0}^{\binom{n+1}{2}} \binom{\binom{n+1}{2}}{m} P(n, m) \cdot \log P(n, m)$$

with

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

- can you interpret $H(n, p)$ in terms of Shannon's source coding theorem?

example

entropy $H(n, p)$ (y-axis) of statistical ensemble defined by the $G(n, p)$ model for $n = 40$ and different values p (x-axis)

a **maximum value of** 820 **bits** is assumed for $p = \frac{1}{2}$

Notes:

- In next week's lecture, we will use Shannon's source coding theorem and optimal compression of sequences with Huffman coding to detect communities in networks. In the following, we first take an entropy perspective on random graphs, which yields an approach to address overfitting in the stochastic block model. We first note that we can calculate entropy for any discrete probability space, i.e. we can also calculate the entropy of ensembles of random graphs.
- For the $G(n, p)$ model, we can calculate entropy based on the microstate probability. We sum over all possible realizations and consider that all networks with m links are equally likely. The microstate probability for all networks with m links is $P(n, m) = p^m(1-p)^{\binom{n+1}{2}-m}$ (we allow for self-loops). We can sum over the number of possible links in a network with n nodes. The maximum number of links (including self-loops) is $\binom{n+1}{2}$ and for each value of m, there are exactly $\binom{\binom{n+1}{2}}{m}$ different network realizations, each having probability $P(n, m)$.
- This yields the entropy function above. For $p = \frac{1}{2}$ entropy is maximized since all possible microstates are equally likely. In this case our lack of knowledge about the generated networks is maximal. For $p = 0$ and $p = 1$ we have zero entropy. Since the resulting network (empty or fully connected) is determined by p, our lack of information is zero. In other words, $H(n, p)$ gives the bits of information that we lose about a network if we only know model parameters n and p.
- For $n = 40$ we obtain a maximum of 820 bits of entropy for $p = 0.5$. The adjacency matrix has 1600 entries (storing one bit each). Since the network is undirected, we only store the lower half of the matrix, leaving 1600/2 + 40/2 = 820 binary entries!

Entropy of random networks: $G(n, m)$ model

- in the **micro-canonical** $G(n, m)$ model all microstates G are equally likely → L03, i.e.

$$P(G) = \frac{1}{Z(n,m)} = \binom{\binom{n+1}{2}}{m}^{-1}$$

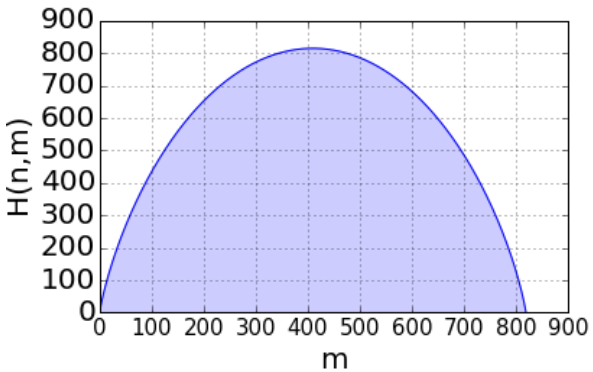
- we then have

$$H(n,m) = -\sum_{g \in \Omega(n,m)} \binom{\binom{n+1}{2}}{m}^{-1} \log \binom{\binom{n+1}{2}}{m}^{-1}$$

and thus

$$H(n,m) = \log \binom{\binom{n+1}{2}}{m} = \log Z(n,m)$$

for sample space Ω with **equiprobable outcomes**, entropy of probability mass function is equal to **log of number of possible outcomes**, i.e. $H(X) = \log |\Omega|$



example

entropy $H(n, m)$ (y-axis) of statistical ensemble defined by the $G(n, m)$ model for $n = 40$ and different values m (x-axis)

a **maximum value of** 820 **bits** is assumed for $m = 410$

Notes:

- We now consider the entropy of the $G(n, m)$ model, for which –as argued in lecture L03– all microstates are equiprobable.
- This is a so-called **micro-canonical ensemble**, where the entropy of the ensemble is the logarithm of the partition function Z, i.e. the logarithm of the number of possible microstates. Micro-canonical ensembles play an important role in the definition of entropy in statistical physics, which links random graph theory to fundamental laws of thermodynamics. For the $G(n, m)$ model, the entropy function becomes:

$$H(n, m) = -\sum_{g \in \Omega(n,m)} \frac{1}{Z(n, m)} \log \frac{1}{Z(n, m)}$$

- With $|\Omega(n, m)| = Z(n, m)$ we have:

$$H(n, m) = -Z(n, m) \cdot \frac{1}{Z(n, m)} \cdot \log \frac{1}{Z(n, m)} = -\log \frac{1}{Z(n, m)} = \log Z(n, m)$$

- Let us compare this to the entropy of the $G(n, p)$ model. Here we have zero entropy for $m = 0$ and $m = \binom{n+1}{2} = 820$, because in both cases there is only one possible outcome (empty or fully connected network respectively). Iff exactly half of the possible links exist ($m = 410$), the entropy is maximal and we recover the entropy of 820 bits previously calculated for the $G(n, p)$ model with $p = 0.5$.

Entropy of stochastic block model

- consider a **micro-canonical stochastic block model** where block matrix entries M_{kl} fix **number of links** between block k and l
- like in the $G(n, m)$ model we assume that all microstates are equiprobable:

$$P(G) = \frac{1}{Z(\mathbf{M}, \vec{z})}$$

- for each pair of blocks k, l we have $\binom{N_{kl}}{M_{kl}}$ microstates, i.e.

$$Z(\mathbf{M}, \vec{z}) = \prod_{0 \leq k \leq l \leq B-1} \binom{N_{kl}}{M_{kl}} \text{ and } H(\mathbf{M}, \vec{z}) = \sum_{0 \leq k \leq l \leq B-1} \log \binom{N_{kl}}{M_{kl}}$$

- for empirical network G_e, block assignment vector $\vec{z}$ and $\hat{M}_{kl} := E_{kl}$ entropy is given as

$$H(\vec{z}) := H(\hat{\mathbf{M}}, \vec{z}) = \sum_{0 \leq k \leq l \leq B-1} \log \binom{N_{kl}}{E_{kl}}$$

Notes:

- Let us now consider the entropy of the stochastic block model. To simplify matters, in analogy to the $G(n, m)$ model, we can consider a so-called **micro-canonical version of the stochastic block model**, where all microstates have the same probability. With this we can simply calculate entropy as the logarithm of the number of realizations.
- The idea is very simple: rather than fixing the **probability** that two nodes in block k and l are connected, we fix the **number of links M_{kl} randomly generated between block k and l**. We then assume that all possible configurations of these M_{kl} links are generated with equal probability.
- Since the probability of all microstates is the same, the entropy only depends on the total number of different realizations, which can be calculated as follows:
  - For each pair of blocks k and l connected by M_{kl} links, we can select one of $\binom{N_{kl}}{M_{kl}}$ possible ways to distribute M_{kl} links among N_{kl} pairs of nodes, with N_{kl} as defined in L04, i.e. the number of possible links between block k and l.
  - Each realization for such a pair of blocks can be combined with each possible realization for any of the other pairs of blocks, so we have to multiply all numbers to get the number of possible microstates $Z(\mathbf{M})$.
  - The entropy is the logarithm of this product (which turns the product into a sum).
- This allows us to calculate the entropy of the stochastic block model ensemble for a given vector $\vec{z}$ and a block matrix $\hat{\mathbf{M}}$, where we simply set the entries to the observed number of links between blocks.

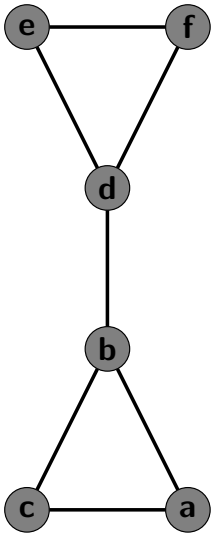
Community detection: entropy minimization

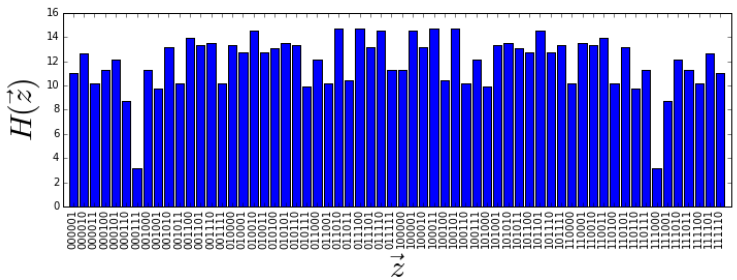
- for micro-canonical ensembles with equiprobable microstates we have **max. likelihood = min. entropy**

naive community detection algorithm

for given number of B blocks we can exhaustively enumerate block assignment vectors $\vec{z}$

1. for each $\vec{z}$ calculate entropy $H(\vec{z})$
2. choose $\hat{\vec{z}} := \text{argmin}_{\vec{z}} H(\vec{z})$





- $H(\vec{z})$ is **lower bound for number of bits required to encode microstate** generated by SBM for given B and $\vec{z}$
- what is the entropy $H(\vec{z})$ for $B = n$ and $\vec{z} = (0, 1, 2, \ldots, n-1)$?

example

$H(\vec{z})$ of stochastic block model for different block assignment vectors $\vec{z}$ and $B = 2$.
$H(\vec{z})$ is minimal for $\vec{z} = (0, 0, 0, 1, 1, 1)$ and $\vec{z} = (1, 1, 1, 0, 0, 0)$.

Notes:

- How can we use the entropy for model inference? Since we have equiprobable microstates, the likelihood of our model parameters is simply the inverse of the number of realizations. The smaller that number, the larger the probabilities of equiprobable microstates (and thus the larger the likelihood) and the smaller the entropy. This means that in the micro-canonical ensemble, the parameters with **maximum likelihood** corresponds to those parameters for which the ensemble has **minimal entropy**.
- This has an intuitive interpretation: we choose the ensemble that minimizes our lack of knowledge about the observed networks, i.e. we choose the model that best explains our observation.
- For or toy example, among all stochastic block models with $B = 2$, those with $\vec{z} = (0, 0, 0, 1, 1, 1)$ and $\vec{z} = (1, 1, 1, 0, 0, 0)$ minimize entropy of the ensemble (and maximize the likelihood of the observation).
- But what about the overfitting example? For $B = 6$, the SBM with $\vec{z} = (0, 1, 2, 3, 4, 5)$ and $\mathbf{M} = \mathbf{A}$ produces only a single network with probability one. We have zero entropy, i.e. our knowledge about the outcome is perfect (just as if we toss a coin that always shows only one side).
- By itself, minimizing entropy does not help us to solve the overfitting issue, but it is the basis to quantify our lack of information about the outcome of the stochastic block model in terms of bits of information, a unit that we can also use to quantify the complexity of our model.

Minimum description length principle

- for micro-canonical stochastic block model with B blocks we define **description length** $\lambda(\vec{z})$ as

$$\lambda(\vec{z}) := H(\vec{z}) + \Delta(\vec{z})$$

where $\Delta(\vec{z})$ are **bits needed to describe the model**

- we can approximate $\Delta(\vec{z})$ as → T Peixoto 2012

$$\Delta(\vec{z}) = m \cdot h\left(\frac{B(B+1)}{2m}\right) + n \cdot \log(B)$$

where $h(x) = (1+x) \cdot \log(1+x) - x \cdot \log(x)$

- applying Occam's razor, we can find the "best model" by **minimizing description length** $\lambda(\vec{z})$

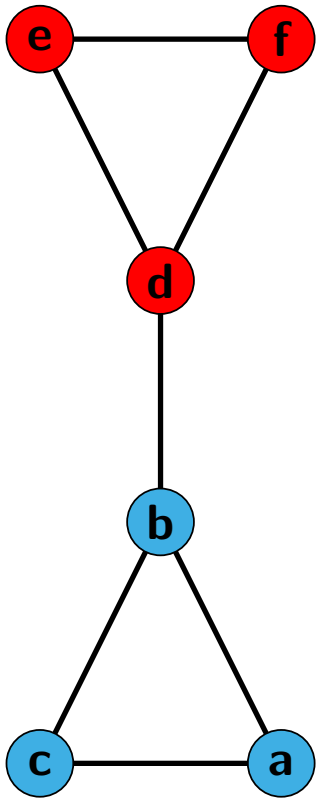
Jorma Rissanen
1932 – 2020

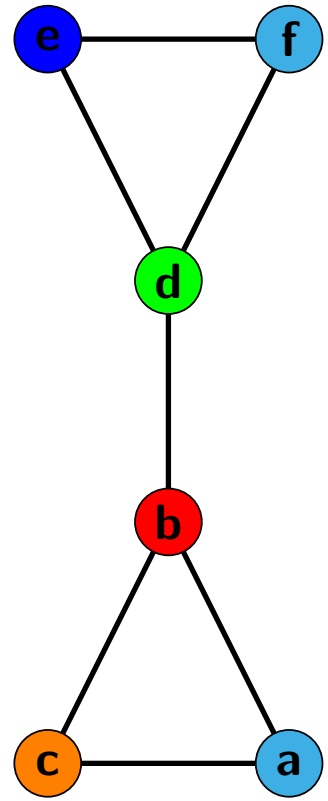
image credit: itsoc.org, fair use

Notes:

- We want a model that minimizes the lack of knowledge about the observed network, but at the same time the **complexity of the model itself should be minimal**. The measure of entropy is the key to achieve this, because we can not only use it to quantify our lack of knowledge about the outcome, but we can also measure the amount of information that we have encoded in the model.
- For $B = 6$ we have simply encoded the desired outcome into the model, i.e. our model (the stochastic block matrix) literally encodes the desired empirical network (the adjacency matrix). We can quantify the amount of information that we have encoded in our model in terms of entropy as well. This captures both the explanatory power of our model (in terms of the lack of knowledge about the outcome) and the complexity of the model (in terms of the number of bits needed to store the model) in the same units (bits). The idea to minimize the description length of data, which consists of the uncertainty about the outcome as well as the description length of the model, goes back to the seminal work → Jorma Rissanen, 1978 . This tells us that pattern recognition can be seen as a kind of data compression.
- But how can we calculate the number of bits needed to encode the model? This depends on the number of blocks B, the number of nodes and the number of links, where larger B correspond to more complex models. The technical details of the derivation of the description length of the stochastic block model are actually quite involved, but an approximation can be derived in terms of the simple expression above. For full details see → T Peixoto, 2013

Overfitting: minimizing description length





example 1

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

$$H(z) = 3.17, \Delta(z) = 10.27$$

$$\lambda(\vec{z}) = H(\vec{z}) + \Delta(\vec{z}) = \mathbf{13.44}$$

example 2

$$B = 6, \vec{z} = (0, 1, 2, 3, 4, 5)$$

$$H(\vec{z}) = 0, \Delta(\vec{z}) = 26.50$$

$$\lambda(\vec{z}) = H(\vec{z}) + \Delta(\vec{z}) = \mathbf{26.50}$$

Notes:

- Using the minimum description length principle, we can **minimize the sum of model entropy and model complexity**, which provides us with an **optimal "compression" of the observed network**. That is we can find the most compact model that at the same minimizes our lack of information about the outcome of the model. A perfect model would be a maximally simple one that allows us to perfectly describe our observed network. In the stochastic block model this corresponds to a description of a perfectly clustered network in terms of a stochastic block matrix.
- In our example, the entropy of the (overfitted) block model (right) is zero while it is 3.17 for the (correct) model on the left. However, encoding the overfitted model as a block matrix actually requires 26.5 bits of information, while the model on the left can be encoded using only 10.27 bits of information.
- In other words, the model on the right requires more than 16 additional bits for a mere 3.17 additional bits of information that we gain about our observed network.
- Following the idea of Ockham's razor, we thus conclude that the model on the left is a better explanation for the observed network, and we thus reject the overfitted model in favor of the "correct" simpler model for the community structures in the network.

Heuristic Description Length Minimization

- how can we efficiently find parameters $B, \vec{z}$ that minimize description length?
- we can use **Monte Carlo algorithm** to find (near-)optimal solution based on **search heuristic**

heuristic optimization algorithm

- let v be a node in block r and let t be block of random neighbor w of v
- move v from block r to block s with probability

$$p(r \to s|t) = \frac{e_{ts} + \epsilon}{e_t + \epsilon B}$$

where e_{st} is number of edges between blocks s and t, e_t is sum of node degrees in block t and $\epsilon > 0$

→ T Peixoto, 2014

Monte Carlo Casino

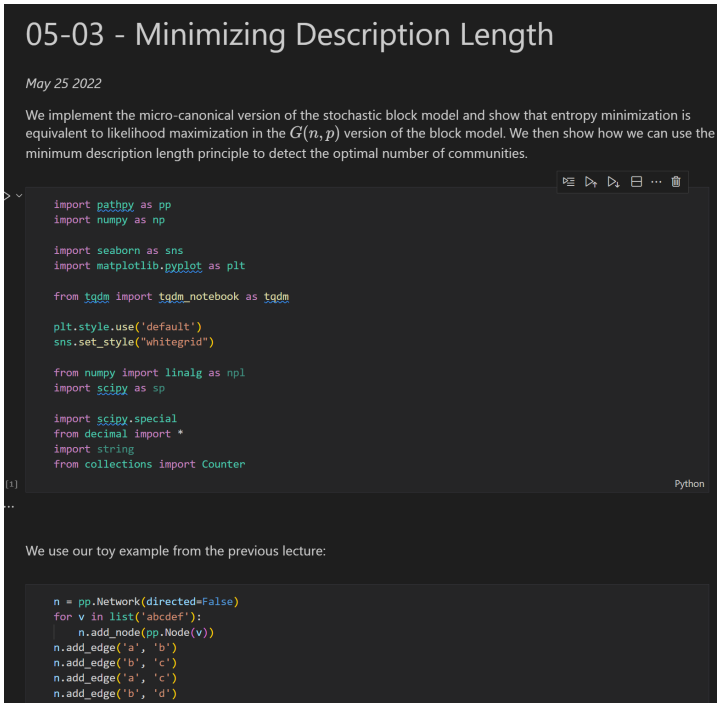
image credit: Wikimedia Commons, User Fruitpunchline, CC BY-SA 4.0

Notes:

- How can we use description length minimization to detect community structures in real-world networks. We note that, compared to the problem of finding an optimal block assignment vector for a given number of blocks B, we have further complicated our optimization problem as we now need to also infer the optimal number of blocks.
- To efficiently find an (approximately) optimal solution, we can use heuristic optimization techniques like Monte Carlo sampling. The idea is to randomly "move" nodes between blocks such that probability of such a move depends on the entropy difference between the two associated block assignments. If we suitably chose the probabilities of those moves, we can efficiently find good community partitions even in very large networks. The algorithm described in → T Peixoto, 2014 is able to find good solutions in $O(n(\log n)^2)$ where n is the number of nodes in the network.

Practice session

- we calculate the **entropy of ensembles of random graphs**
- we implement the **micro-canonical stochastic block model**
- we calculate the **entropy of the microcanonical stochastic block model**
- we use the **minimum description length principle** to detect parsimonious community structures in networks



05-03 - Minimizing Description Length

May 25 2022

We implement the micro-canonical version of the stochastic block model and show that entropy minimization is equivalent to likelihood maximization in the $G(n,p)$ version of the block model. We then show how we can use the minimum description length principle to detect the optimal number of communities.

```
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
```

[1] Python

We use our toy example from the previous lecture:

```
n = pp.Network(directed=False)
for v in list('abcdef'):
    n.add_node(pp.Node(v))
n.add_edge('a', 'b')
n.add_edge('b', 'c')
n.add_edge('a', 'c')
n.add_edge('b', 'd')
```

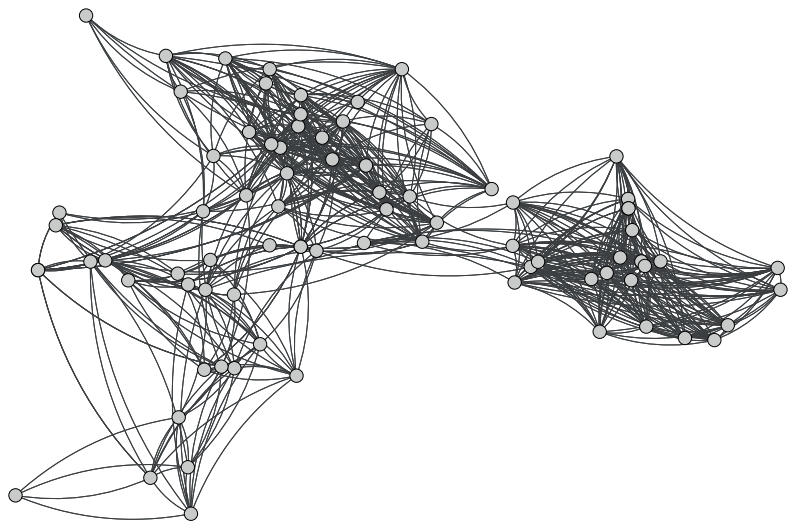
practice session

see notebooks 05–03 and 05–04 in `gitlab` repository at

→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks`

Notes:

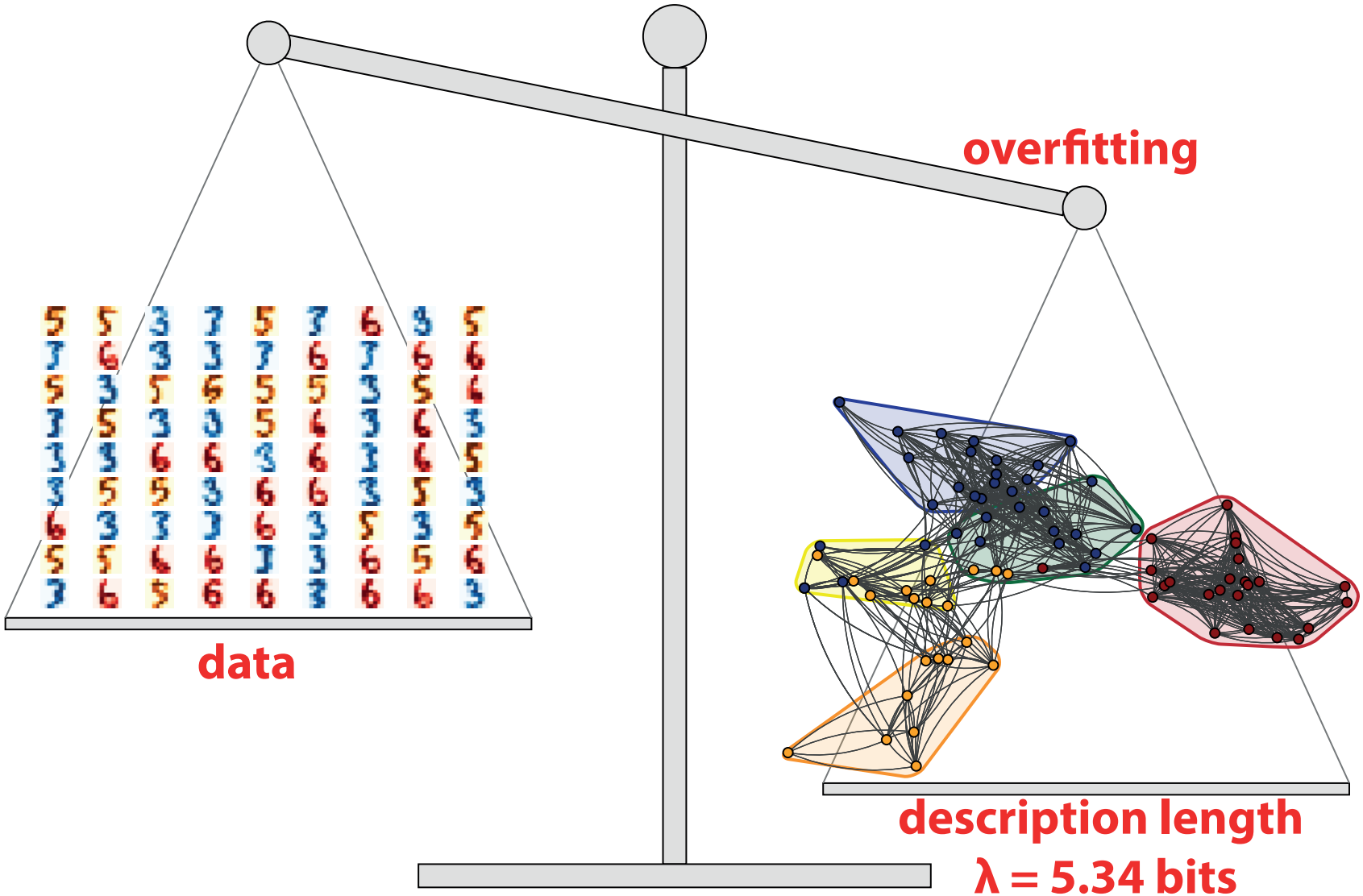
Application: Optical Character Recognition



Notes:

- This overfitting problem has a clear implication for our example of optical character recognition.
- if we are not able to automatically infer the correct number of communities, we are unable to infer the correct number of different characters that are shown in the collection of images. We are thus also not able to correctly group those images that show the same characters.
- In our example, overfitting the stochastic block model to the network structure corresponds to grouping images according to similarities that are not due to the different characters displayed (e.g. example characters written by different people, different ways to write characters, different pens, etc.).

Decription length Minimization



Notes:

- The problem of finding an optimal model that neither overfits nor underfits the data is a key challenge in community detection (and more general in clustering algorithms and machine learning).
- It is easy to see the implications of such an overfitting in our character recognition example. We could use **two communities** to explain the community structure in our networks, however the likelihood of our model (i.e. the explanatory power of our model for the data) is rather low. This is a sign that **we have underfitted our model**.
- **Increasing the number of clusters to three** massively increases explanatory power, but also increased the complexity of our model. Here, this additional complexity is justified by the actual pattern that we are interested in, i.e. we have found a better model for our data.
- The problem is that we can further increase the complexity of our model (i.e. setting $B = 10$). This trivially increases the likelihood of our model, since we can more easily adjust the parameters to the data. This comes with the **risk of overfitting our model**. In our example, we see that the clusters found by such a model do not correspond to actual digits anymore, i.e. the **model fails to extract knowledge from the data**.

In summary

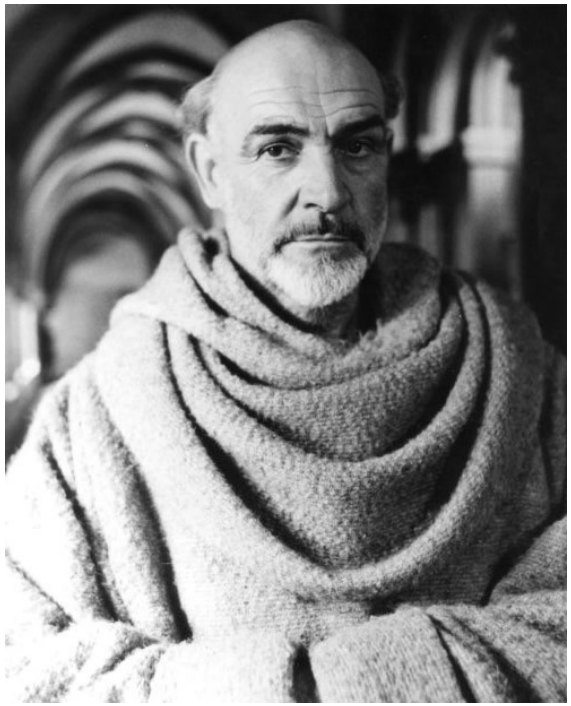
- to detect optimal communities we need **methods to select the best model**

challenge

model selection techniques must account for

1. **explanatory power**
   i.e. how well the model describes the data
2. **model complexity**
   i.e. how many parameters we need to fit

- we can use **information theory** to capture both dimensions of a model in terms of **description length**
- highlights fundamental relation between **pattern recognition and data compression**



Sean Connery as **William of Baskerville** in Jean-Jacques Annaud's movie adaptation of Umberto Eco's "The name of the rose"

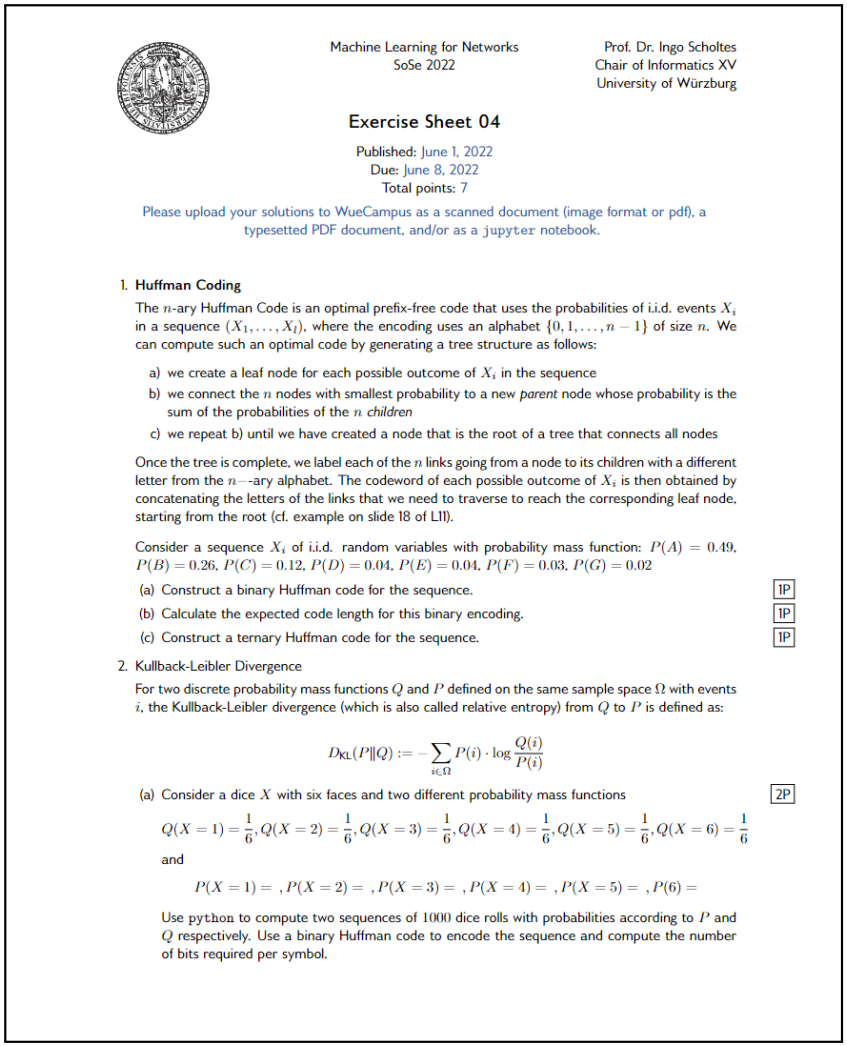
image credit: 20th Century Studios, fair use

Notes:

- In summary, the detection of optimal community structures in networks is a complex machine learning problem that requires techniques to select the best out of a set of candidate models. These model selection techniques must account for the explanatory power of a model, i.e. how well it explains the observed phenomenon (here: the community structure of the observed network). But it must also account for the complexity of the model, i.e. how many parameters we fit to the data.
- In today's lecture, we have shown that we can use information-theoretic methods to address this challenge, by selecting the model that "most efficiently describes" the data in terms of the overall description length. This highlights the natural relation between data compression and machine learning that we will further explore next week. It also provides an elegant approach to incorporate Occam's razor into statistical learning techniques, with applications beyond community detection.
- A remark about the picture above: Sean Connery played the character "William of Baskerville" in the movie adaption of Umberto Eco's brilliant novel "The name of the rose" (I highly recommend you to read it). The name is a combination of "William of Ockham" and "Baskerville" from Sir Arthur Conan Doyle's Sherlock Holmes story: "The Hound of the Baskervilles". It is used in appreciation of the logical reasoning used both by William of Ockham and Sherlock Holmes, which often included the application of Ockham's razor to reason about a criminal case.

Exercise sheet 4

- **fourth exercise sheet** will be released today
  - entropy of graph ensembles
  - Shannon's source coding theorem
  - Kullback-Leibler divergence
- solutions are due **June 5** (via Moodle)
- present your solution to earn bonus points



Machine Learning for Networks
SoSe 2022

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

Exercise Sheet 04

Published: June 1, 2022
Due: June 8, 2022
Total points: 7

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. **Huffman Coding**

The n-ary Huffman Code is an optimal prefix-free code that uses the probabilities of i.i.d. events X_i in a sequence $(X_1, \ldots, X_l)$, where the encoding uses an alphabet $\{0, 1, \ldots, n-1\}$ of size n. We can compute such an optimal code by generating a tree structure as follows:

a) we create a leaf node for each possible outcome of X_i in the sequence
b) we connect the n nodes with smallest probability to a new *parent* node whose probability is the sum of the probabilities of the n *children*
c) we repeat b) until we have created a node that is the root of a tree that connects all nodes

Once the tree is complete, we label each of the n links going from a node to its children with a different letter from the n−-ary alphabet. The codeword of each possible outcome of X_i is then obtained by concatenating the letters of the links that we need to traverse to reach the corresponding leaf node, starting from the root (cf. example on slide 18 of L11).

Consider a sequence X_i of i.i.d. random variables with probability mass function: $P(A) = 0.49$, $P(B) = 0.26$, $P(C) = 0.12$, $P(D) = 0.04$, $P(E) = 0.04$, $P(F) = 0.03$, $P(G) = 0.02$

(a) Construct a binary Huffman code for the sequence. 1P
(b) Calculate the expected code length for this binary encoding. 1P
(c) Construct a ternary Huffman code for the sequence. 1P

2. Kullback-Leibler Divergence

For two discrete probability mass functions Q and P defined on the same sample space Ω with events i, the Kullback-Leibler divergence (which is also called relative entropy) from Q to P is defined as:

$$D_{\mathrm{KL}}(P \| Q) := -\sum_{i \in \Omega} P(i) \cdot \log \frac{Q(i)}{P(i)}$$

(a) Consider a dice X with six faces and two different probability mass functions 2P

$$Q(X=1) = \frac{1}{6}, Q(X=2) = \frac{1}{6}, Q(X=3) = \frac{1}{6}, Q(X=4) = \frac{1}{6}, Q(X=5) = \frac{1}{6}, Q(X=6) = \frac{1}{6}$$

and

$$P(X=1) = \ , P(X=2) = \ , P(X=3) = \ , P(X=4) = \ , P(X=5) = \ , P(6) =$$

Use `python` to compute two sequences of 1000 dice rolls with probabilities according to P and Q respectively. Use a binary Huffman code to encode the sequence and compute the number of bits required per symbol.

Notes:

Questions

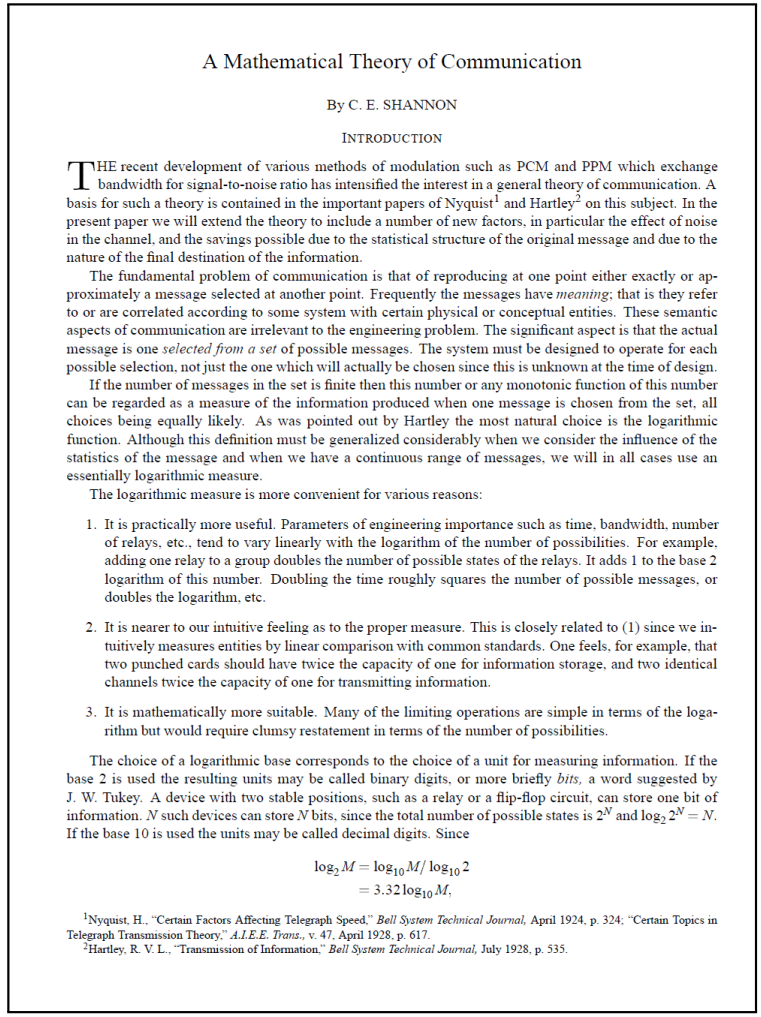
1. How is the entropy of a probability mass function defined?
2. Calculate the entropy of the $G(n, p)$ model and explain for which value p it is maximized.
3. Investigate the definition of the Kullback-Leibler divergence and explain its interpretation in terms of entropy.
4. Explain under which conditions the maximization of likelihood corresponds to a minimization of entropy.
5. Explain how we can construct a Huffman code for a sequence of random variables.
6. What statement does Shannon's source coding theorem make?
7. Use Shannon's source coding theorem to calculate the optimal compression for a sequence of biased coin tosses with $p \neq 0.5$.
8. How can entropy be used for community detection based on the stochastic block model?
9. How is the description length of the stochastic block model defined?
10. Explain how the detection of the optimal number of communities in a network is related to Occam's razor.

Notes:

References

reading list

- CE Shannon: **A Mathematical Theory of Communication**, Bell System Technical Journal, 1948
- S Kullback, RA Leibler: **On information and sufficiency**, Annals of Mathematical Statistics, 1951
- DA Huffman: **A Method for the Construction of Minimum-Redundancy Codes**, Proceedings of the IRE, 1952
- J Rissanen: **Modeling by shortest data description**. Automatica, 1978
- PW Holland, KB Laskey, S Leinhardt: **Stochastic blockmodels: First steps**, Social Networks, 1983
- G Bianconi: **Entropy of network ensembles**, Physical Review E, 2009
- J Mayer, K Khairy, J Howard: **Drawing an elephant with four complex parameters**, Am. J. Phys., 2010
- B Karrer and MEJ Newman: **Stochastic blockmodels and community structure in networks**, Phys. Rev. E. 83, 2011
- TP Peixoto: **Entropy of stochastic block model ensembles**, Phys. Rev. E. 85, 2012
- TP Peixoto: **Parsimonious module inference in large networks**, Phys. Rev. Lett. 110, 2013
- TP Peixoto: **Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models**, Phys. Rev. E. 89, 2014



A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

The logarithmic measure is more convenient for various reasons:

1. It is practically more useful. Parameters of engineering importance such as time, bandwidth, number of relays, etc., tend to vary linearly with the logarithm of the number of possibilities. For example, adding one relay to a group doubles the number of possible states of the relays. It adds 1 to the base 2 logarithm of this number. Doubling the time roughly squares the number of possible messages, or doubles the logarithm, etc.

2. It is nearer to our intuitive feeling as to the proper measure. This is closely related to (1) since we intuitively measures entities by linear comparison with common standards. One feels, for example, that two punched cards should have twice the capacity of one for information storage, and two identical channels twice the capacity of one for transmitting information.

3. It is mathematically more suitable. Many of the limiting operations are simple in terms of the logarithm but would require clumsy restatement in terms of the number of possibilities.

The choice of a logarithmic base corresponds to the choice of a unit for measuring information. If the base 2 is used the resulting units may be called binary digits, or more briefly *bits*, a word suggested by J. W. Tukey. A device with two stable positions, such as a relay or a flip-flop circuit, can store one bit of information. N such devices can store N bits, since the total number of possible states is 2^N and $\log_2 2^N = N$. If the base 10 is used the units may be called decimal digits. Since

$$\log_2 M = \log_{10} M / \log_{10} 2 = 3.32 \log_{10} M,$$

[1]Nyquist, H., "Certain Factors Affecting Telegraph Speed," *Bell System Technical Journal*, April 1924, p. 324; "Certain Topics in Telegraph Transmission Theory," *A.I.E.E. Trans.*, v. 47, April 1928, p. 617.
[2]Hartley, R. V. L., "Transmission of Information," *Bell System Technical Journal*, July 1928, p. 535.

Notes: