Organization of the course

Flipped Classroom & Team-Based Learning

This course will be taught inflipped-classroom style with elements of team-based learning. For the duration of the whole course you will be divided into teams of five students, see below for details.

Check thisvideo (in English) or thisinfographic (in Dutch) to learn more about flipped-classroom style. For you as students, the biggest difference to conventional lectures is that you have to**prepare yourself before attending classes**, as it is assumed during classes that you have studied the material already. The advantage of this model is that we can spend the face-to-face time during the work sessions to talk about the studied material together, to recapitulate the most difficult and important aspects, and to strengthen our understanding by actually working on the relevant problems in teams. The work sessions will be led by teacher and TAs, with an active involvement of the students.

Our week in Information Theory:

This is a 6 EC course given over 7 weeks, plus one week of exam. We are aiming to entertain you for approximately 20 hours per week with this course.

The content of every week is distributed via the following sources:

- 7 Canvas modules consisting of Pages and Quizzes, check out the overview and see the details under Modules
- 7 Weekly sets of (usually 6) practice problems. These problems will be solved during the work sessions, and presented during presentation sessions on Fridays, see below for details.
- 6 weekly homework sets, available from 16:00 on Wednesdays, to be handed in one week later on Wednesday at 12:00.

The weekly schedule looks as follows:

- Monday/Tuesday: individually study this week's Canvas modules, individually answer the reading questions (quizzes). There are two hours scheduled for "self-study" in your schedules on Tuesday 13-15. No actual session will take place during this time, and you can study the material whenever you like.
- Wednesday 12:00: deadline for Reading Questions (individual) and Homework (team).
- Wednesday 13:00-17:00: Work session, at 16:00 new homework becomes available.
- Friday 12:00: presentation schedule available, 13:00-15:00: Presentation session.

Every week, your actions will be the following:

1. go through the material of the week
2. read up on the material in other sources from the internet
3. (individually) answer the reading questions (due one hour **before** the work session)
4. during the work sessions, we start with an intro and team quiz as warmup (see below) and then work together in teams on practice problems. The goal is that every student is able to solve all of the practice problems. The work sessions are the chance to meet up and work with your team, so your attendance is expected. See below for details.
5. (in teams) carry out and hand in the homework problems (due 6 days after each work session).
6. (in teams) attend the presentation session, where teams present and moderate the practice problems.
7. if you have time and interest, check out the bonus material.

Work Session (Wed, 13:00-17:00)

First 3 weeks:
SP G0.10-0.12: Teams
SP G0.23-0.25: Teams
Eulerzaal: Teams

Bring your own device! **Bring along a charged laptop or smartphone on which you can solve a Canvas quiz in order to complete the Intro Quiz.** If you have trouble accessing Canvas with your laptop, please go to the laptop helpdesk (Tue and Thu 12-13 next to the library desk).

We expect regular work sessions to proceed as follows:

- 13:00; start
- 13:05-13:25; Intro Quiz (individual quiz)
- 13:25-13:45; Team Quiz (group quiz), Team Quiz is to be submitted by at least one team member. The whole team will receive the grades of the team member who submits first. Team Quiz has the same questions as the Intro Quiz.
- 13:45- ~14:00; discussion about the Intro/Team Quiz, additional explanations of the week's material, previous homework questions etc.
- until 16:00; work on the problem sets, in your teams. Remember, the goal is that every student is able to solve all practice problems. Each team will be told which (of the usually 6) problems will be the one they specialize in and will have to moderate on Friday. As a team, start with the problem you will be moderating, work out and write down a clean solution on paper or electronically. Think about how you want to moderate this question on Friday. Get the attention of one of the available teachers and get this written version approved by the teacher. Once it has been approved, start working on the other problems. Prepare yourself to a degree that your team is able to present any of the other problems.
- 16:00; new homework is made available
- 16:00-16:45; distribute and plan your further team work on the problem sets and homework

Presentation Session (Friday 13:00-15:00)

Presentations sessions are held in smaller groups of about 6 teams (so around 30 students):

NIKHEF ILLC F2.19, F0.20, F0.25: These rooms do not have enough chairs, we will have to place some temporary chairs in them in order to fit everybody. If your presentation session is in this room, please help the teachers to set up the room and chairs, and also to remove them again afterwards.

SP G0.23-0.25

A presentation schedule will be made available on Fridays at 12:00, it shows for each problem which team is moderating and which team is presenting it. For each problem, the presenting team is picked randomly among the teams who are not moderating the problem.

Each problem has an expert team who is responsible for moderating the problem. **The goal for moderators is that every student who is present at the session understands how to solve it.** However, as moderator, you cannot simply write down the perfect solution you already have prepared on paper during the work session, but you have to explain it "through" the presenter, i.e. by asking clarifying questions, helping whenever the presentation goes off path etc. The presentation of a single problem is supposed to take around 10 minutes with another 5 minutes for questions and discussion, resulting in the available 6*15=90 minutes. However, presentation lengths might vary considerably with difficulty and quality of the presenter and moderator. It is entirely possible that some problems will not be presented due to time constraints.

Check our collection of useful tips both for moderators and presenters

Team Dynamics & Dropping the Course

You are working in the same fixed teams during the first three weeks. At that point, your team performance will be evaluated by your fellow team members. As courtesy to your team, we ask you to follow this course for at least the first three weeks. That will be a natural point to drop the course if you don't see it fit. We will form new teams from those people who want to finish it. These teams will then be fixed for the rest of the course. Everything works the same, just with different people.

Final Exam

There will be a written final exam on Tuesday, 18 Dec, 9:00-12:00 on the other side of Amsterdam. The exam will consist of two parts: a first quick multiple-choice part about concept questions, and then some problems to work out. The first hour (9:00 - 10:00) will be closed-book, i.e. WITHOUT notes/internet/etc, and at 10:00 you will be asked to hand in the first part of the exam (of course, you can already start working on the second part before 10:00, as soon as you are done with the first part). From 10:00-12:00, the exam is open-book, meaning that you can consult any study material including these Canvas pages, our lecture notes, [CF], [CT], [MacKay] as well as any notes you have made. You are allowed to use a laptop (or iPad, e-reader etc.) to access the course pages and websites like wikipedia, wolfram alpha etc. The exam is not an exercise in googling solutions, therefore we ask you not to use any solution manuals for information theory exercises and similar material that can be found online.

Obviously, you are also not allowed to communicate with each other nor with anybody else on the internet. Therefore, your computer screen has to be large enough (no smartphones) and visible during all the time you are using it. We are trusting you that you play according to these rules.

Exams from previous years are available for practice: Exam 2014, Exam 2015 (in 2014/2015 it was open-book, but no electronic devices were allowed), Exam 2017 (same style as this year).

The grade obtained at the exam counts for 50% towards the final grade, you have to have at least a score of 50% at the final exam to pass the course!

Grades

For all team assignments such as homework and the Team Quiz every week, all members of a team receive the same grade.

Your final grades consist of three parts:

- 15% is determined by the average grade for the reading questions, intro and team quizzes. In case of active participation (i.e. participating in all activities marked with (p) such as filling in team-member evaluations, voting for problems to discuss, active participation in presentation session and forum discussions etc.), we drop the worst two of all these grades.
- 35% is by the average (team) homework grade
- 50% by the (individual) final exam, but you have to have at least a score of 50% at the final exam to pass the whole course.

Questions

If at any point you have a question, remarks, feedback etc. it's likely that you are not the only one wondering about this issue. Please raise it in the General Discussion forum, so other students or the teachers can answer them for everybody. If you want to get in touch with us directly, please use the Canvas-internal messaging function to send a **message to all teachers**. Please do not use regular email!

Course content (overview, now with video!)

(Download the **slides from the video** to "click on the links")

For a detailed overview of course content and learning objectives, see **Studiegids**.

The content of this course is organized into seven modules (one for each week the course runs, except the last week in which you will make a final exam):

01 Probability Theory

This is an introductory module that you will prepare before the start of the first lecture. It allows you to check your knowledge about logarithms, probability theory, and programming.

02 Entropy

This module introduces some of the core concepts of information theory: entropy, conditional entropy, and mutual information. You will learn how to work with these concepts, and how to relate them to each other.

03 Source Coding / Data Compression

In this module, you will learn how to compress sources of data that are not uniformly random (e.g., files of English text). You will see some specific algorithms (in pseudo-code) and learn why they perform optimally.

04 Typical Sets and Encryption

The first half of this module continues with the topic of data compression, and introduces the concept of typical sets to aid in efficient (but lossy) compression. In the second half, you will learn how to perform perfectly (information-theoretically) secure encryption, and what that necessarily costs.

05 Random Processes

In this module, you will study various random processes (such as Markov Chains), and learn how to measure their randomness using the concept of "entropy rate". We will also consider random walks on graphs in this module.

06 Error Correction and Zero-Error Transmission

This module considers the problem of trying to send a message over a noisy channel, and limiting the probability that the message (information) is lost while doing so. You will explore how much information you can send over different types of channels if you want to eliminate the probability of loss completely.

07 Noisy Channel Coding

This model continues with the topic of module 06, but takes a more fundamental approach. We will finish by proving Shannon's famous source-channel coding theorem, which determines how much information can be sent over a noisy channel if some (arbitrarily small) probability of losing the message is allowed.

Tips for Moderators and Presenters

The following tips are meant for moderators and presenters at the presentation sessions on Fridays. See course organization for details.

Goal for moderators: every student who is present at the session understands how to solve the problem. In other words, transfer your understanding and knowledge of the problem to the audience "through" the presenter.

Tips for moderators:

- Besides preparing a written “perfect solution” during the work session, also think about the following questions: What is “the point” of the exercise? Are there different solutions possible? What are the possible obstacles that (other) students might encounter? Discuss the answers to these questions with the teacher during the work session as well.
- Slow down presenters if they are moving too fast.
- Try to ask clarifying questions on the way.
- Keep an eye on time (e.g. by stopping students early enough if you see that the presentation is not going in the right way).
- If the presenters are stuck, ask them the right questions to help them on their way, or give them small hints. Do not take over the presentation by giving away the full solution immediately.

Tips for presenters:

- When preparing the exercise, think about which things you want to write down on the board, and which things you only want to talk about.
- It is a good idea to quickly recap what the problem is (possibly writing the essential parts on the board), but do not waste a lot of time by copying the whole exercise.
- Use the board, write down what you are doing.
- Try to write in a readable way. Always speak out what you write, it helps to decipher handwriting.

- Don’t go too fast, make sure everybody can follow and has understood the steps so far.
- Even as speaker, ask whether the audience has understood and could follow your explanations. How many have had the same solution?
- Don’t be afraid of making pauses and asking questions.

References and Shannon's original papers

Shannon's originals

Most of the contents of this course are based on (a modern understanding of) the contents of the two following papers by Claude Shannon:

A mathematical theory of communication - the original paper by Claude Shannon, published in 1948 in *The Bell System Technical Journal*. Check its **wikipedia entry**.

Communication Theory of Secrecy Systems is a paper published in 1949 by **Claude Shannon** discussing **cryptography** from the viewpoint of **information theory**. Check its **wikipedia entry**.

More recent references

For more contemporary readings of these papers, have a look at:

- Course lecture notes (work-in-progress) by Yfke Dulek and Christian Schaffner, available on**https://github.com/cschaffner/InformationTheory/raw/master/Script/InfTheory3.pdf**
- Good reference: [CT] Thomas M. Cover, Joy A. Thomas. 'Elements of information theory':**http://onlinelibrary.wiley.com/book/10.1002/0471200611**, 2nd Edition. New York: Wiley-Interscience, 2006. ISBN 0-471-24195-4.
- Good reference: [MacKay] David J. C. MacKay. 'Information Theory, Inference, and Learning Algorithms':**http://www.inference.phy.cam.ac.uk/mackay/itila/book.html**. Cambridge University Press, 2003. ISBN 0-521-64298-1

Copyright and Acknowledgements

The blended-learning material in this course was created from April to December 2018 by Yfke Dulek and Christian Schaffner, with the help of **Huub Rutjes** (interactive graphs) and **Krijn Boom** (videos) from the FNWI blended learning team. We were supported by a FNWI blended-learning grant. Thank you for the support!

Some of the material in this course is heavily inspired from:

- Jeremy Orloff, and Jonathan Bloom. *18.05 Introduction to Probability and Statistics, Spring 2014*. (Massachusetts Institute of Technology: MIT OpenCourseWare), **http://ocw.mit.edu** (Accessed). License: **Creative Commons BY-NC-SA**

Thanks a lot!

Introduction: Preliminaries

In this module, you will get an overview of the knowledge and skills we expect from you at the start of the course. Depending on your background, you might need to brush up on your probability theory, or you might want to grasp some basic programming skills.

Every piece of content ends with a small quiz. These quizzes allow you to assess your knowledge of that particular piece of content. You may choose to read the content first, or go directly to the quiz if you think you already master the content. **Important:** you need to finish these quizzes before the start of the first working session.

If you want to build a stronger foundation for the topics discussed in this module, you can have a look at these resources:

- The lecture notes for the **Basic Probability: Theory** course taught at UvA.
- Programming tutorials such as **Learn Python** or **Codecademy's Python tutorial**.
- any source you find yourself on the **world wide web**.

Some Important Distributions

We end the theoretic preliminaries with a (non-exhaustive) list of common probability distributions that you will come across throughout the course.

- The distribution of a biased coin with probability $P_X(1) = p$ to land heads, and a probability of $P_X(0) = 1 - p$ to land tails is called **Bernoulli(p) distribution**. The expected value is $\mathbb{E}[X] = p$ and the variance is $\mathrm{Var}[X] = p(1-p)$.
- When n coins $X_1, X_2, \ldots, X_n$ are flipped independently and every X_i is Bernoulli(p) distributed, let $S = \sum_{i=1}^{n} X_i$ be their sum, i.e., the number of heads in n throws of a biased coin. Then, S has the **binomial(n, p) distribution**:

$$P_S(k) = \binom{n}{k} p^k (1-p)^{n-k} \qquad \text{where } k = 0, 1, 2, \ldots, n\,.$$

  From simple properties of the expected value and variance, one can show that $\mathbb{E}[S] = np$ and $\mathrm{Var}[S] = np(1-p)$.
- The **geometric(p) distribution** of a random variable Y is defined as the number of times one has to flip a Bernoulli(p) coin before it lands heads:

$$P_Y(k) = (1-p)^{k-1} p \qquad \text{where } k = 1, 2, 3, \ldots\,.$$

  There is another variant of the geometric distribution used in the literature, where one excludes the final success event of landing heads in the counting:

$$P_Z(k) = (1-p)^k p \qquad \text{where } k = 0, 1, 2, 3, \ldots\,.$$

  While the expected values are slightly different, namely $\mathbb{E}[Y] = \frac{1}{p}$ and $\mathbb{E}[Z] = \frac{1-p}{p}$, their variances are the same: $\mathrm{Var}[Y] = \mathrm{Var}[Z] = \frac{1-p}{p^2}$.

What programming languages am I allowed to use?

We do not put any restrictions on the programming languages that you use for this course. We do *not* test your programming skills and we do *not* require you to hand in any code files or documentation. In this course, as is the case for many fields of research these days, programming is a supportive tool. It helps you to grasp certain definitions/concepts better, and it allows you to perform larger calculations that you would not be able to do by hand.

It does not matter if you use Java, C, Haskell, Python, Ruby, Excel, Mathematica, Rust, LaTeX (hey, **it's Turing complete**), bash, or **Piet**. As long as you are able to process input files in plaintext format, and get numerical/textual output. The output is all that matters.

If you have little or no programming experience, we advise you to write your programs in Python. We will provide you with python code stubs (starter files) that take care of the not-so-interesting parts for you: reading input files, storing their contents in arrays or other data structures, etc. That allows you to focus on the mathematically interesting parts of the program you are writing.

Jensen's Inequality

The following theorem will be very useful to derive basic properties of entropy.

Theorem: Jensen's inequality

Let $f : \mathcal{D} \to \mathbb{R}$ be a convex function, and let $n \in \mathbb{N}$. Then for any $p_1, \dots, p_n \in \mathbb{R}_{\geq 0}$ such that $\sum_{i=1}^{n} p_i = 1$ and for any $x_1, \dots, x_n \in \mathcal{D}$ it holds that

$$\sum_{i=1}^{n} p_i f(x_i) \geq f\left(\sum_{i=1}^{n} p_i x_i\right).$$

If f is strictly convex and $p_1, \dots, p_n > 0$, then equality holds if and only if $x_1 = \dots = x_n$. In particular, if X is a real random variable whose image $\mathcal{X}$ is contained in $\mathcal{D}$, then

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]),$$

and if f is strictly convex, equality holds if and only if there is a $c \in \mathcal{X}$ such that $X = c$ with probability 1.

Proof

The proof is by induction. The case $n = 1$ is trivial, and the case $n = 2$ is identical to the very definition of convexity. Suppose that we have already proved the claim up to $n - 1 \geq 2$. Assume, without loss of generality, that $p_n < 1$. Then:

$$\begin{aligned}
\sum_{i=1}^{n} p_i f(x_i) &= p_n f(x_n) + \sum_{i=1}^{n-1} p_i f(x_i) \\
&= p_n f(x_n) + (1 - p_n) \sum_{i=1}^{n-1} \frac{p_i}{1 - p_n} f(x_i) \\
&\geq p_n f(x_n) + (1 - p_n) f\left(\sum_{i=1}^{n-1} \frac{p_i}{1 - p_n} x_i\right) \text{(induction hypothesis)} \\
&\geq f\left(p_n x_n + (1 - p_n) \sum_{i=1}^{n-1} \frac{p_i}{1 - p_n} x_i\right) \text{(definition of convexity)} \\
&= f\left(p_n x_n + \sum_{i=1}^{n-1} p_i x_i\right) \\
&= f\left(\sum_{i=1}^{n} p_i x_i\right).
\end{aligned}$$

That proves the claim. As for the strictness claim, if $x_1, \ldots, x_n$ are not all identical, then either $x_1, \ldots, x_{n-1}$ are not all identical and the first inequality is strict by induction hypothesis, or $x_1 = \cdots = x_{n-1} \neq x_n$ so that the second inequality is strict by the definition of convexity.

Properties of Shannon Entropy

In this section, we list a few properties of the Shannon entropy, and show a trick to compute it more easily by hand.

Proposition: Positivity of entropy

Let X be a random variable with image $\mathcal{X}$. Then

$$0 \leq H(X).$$

Equality holds iff there exists $x \in \mathcal{X}$ with $P_X(x) = 1$ (and thus $P_X(x') = 0$ for all $x' \neq x$).

Proof

For all $x \in \{X\}$, we have $0 \leq P_X(x) \leq 1$, and hence $-P_X(x) \log P_X(x) \geq 0$. So $H(X)$, which is the sum of those terms, is always nonnegative. To characterize the condition for equality, note that by definition of Shannon entropy, $H(X) = 0$ when $P_X(x) = 1$ for some x. On the other hand, if $H(X) = 0$ then for any x with $P_X(x) > 0$ it must be that $\log(1/P_X(x)) = 0$ and hence $P_X(x) = 1$.

Proposition: Upper bound on entropy

Let X be a random variable with image $\mathcal{X}$. Then

$$H(X) \leq \log(|\mathcal{X}|).$$

Equality holds iff $P_X(x) = 1/|\mathcal{X}|$ for all $x \in \mathcal{X}$.

Proof hint

We encourage you to try to find the proof for this proposition yourself. As a first step, you may want to write out the definition of $H(X)$ apply Jensen's inequality.

Show full proof

The function $f : \mathbb{R}_{>0} \to \mathbb{R}$ defined by $y \mapsto \log y$ is strictly concave on $\mathbb{R}_{>0}$. Thus, by Jensen's inequality:

$$H(X) = \sum_{x \in \mathcal{X}} P_X(x) \cdot \log \frac{1}{P_X(x)} \leq \log\Big(\sum_{x \in \mathcal{X}} P_X(x) \cdot \frac{1}{P_X(x)}\Big) = \log\Big(\sum_{x \in \mathcal{X}} 1\Big) = \log(|\mathcal{X}|).$$

Furthermore, since we may restrict the sum to all x with $P_X(x) > 0$, equality holds if and only if $\log(1/P_X(x)) = \log(1/P_X(x'))$, and thus $P_X(x) = P_X(x')$, for all $x, x' \in \mathcal{X}$.

When working with explicit distributions, one can always compute the entropy of a random variable by filling in all the probabilities in the definition of entropy. However, in some cases there is some structure to the distribution. In those cases, the entropy can be computed in a smarter and faster way. This is especially useful when you are computing the entropy by hand, but can also help when analysing the entropy of a more complex distribution containing some unknown variables.

Video-2018-06-27-12-10-50_Smart ways to compute entropy.MP4

In general, the entropy of a random variable with probabilities $p_1, \ldots, p_n$ can be expressed as

$$H(p_1, \ldots, p_k, p_{k+1}, \ldots, p_n) = h\left(\sum_{i=1}^{k} p_i\right) \quad + \quad \left(\sum_{i=1}^{k} p_i\right) \cdot H\left(\frac{p_1}{\sum_{i=1}^{k} p_i}, \ldots, \frac{p_k}{\sum_{i=1}^{k} p_i}\right) \quad + \quad \left(\sum_{i=k+1}^{n} p_i\right) \cdot .$$

You can of course use this trick multiple times in a row to break down the entropies on the right-hand side of this equation even further.

Exercise

Consider a random variable X with $\mathcal{X} = a, b, c$ and $P_X(a) = \frac{1}{2}$, $P_X(b) = P_X(c) = \frac{1}{4}$. Compute the entropy of X using the techniques shown in the video.

Show solution

We can think of this distribution as the result of two fair coin tosses: if the first coin comes out heads, the outcome is a. If it comes out tails, we toss another fair coin to determine whether the outcome is b or c.

An appropriate underlying probability space (Ω, P) could be $\Omega = \mathsf{hh}, \mathsf{ht}, \mathsf{th}, \mathsf{tt}$ and $P(\omega) = \frac{1}{4}$ for all $\omega \in \Omega$. Then we define the function $X : \Omega \to \mathcal{X}$ as

$$X(\mathsf{hh}) = X(\mathsf{ht}) = a, \qquad X(\mathsf{th}) = b, \qquad X(\mathsf{tt}) = c.$$

This yields the correct distribution P_X.

The following computation now leads to the entropy of X:

$$H(X) = h\left(\frac{1}{2}\right) + \frac{1}{2}h(0) + \frac{1}{2}h\left(\frac{1}{2}\right) = \frac{3}{2}.$$

The first coin toss determines whether the outcome is a (on heads h) or something else (on tails t). On heads, the second coin toss does not give any more information, whereas on tails, the second coin toss still decides between outcome b and outcome c.

The Chain Rule

The chain rule expresses the relation between the conditional entropy and the joint/maringal entropies of the variables involved. We first state and prove the chain rule for two random variables, and then generalize it to n variables.

Proposition: Chain Rule

Let X and Y be random variables. Then

$$H(XY) = H(X) + H(Y|X)\,.$$

Proof hint

We encourage you to try to prove this for yourself. As a starting point, write out the definition of $H(XY)$, and rewrite the terms $P_{XY}(x,y)$ into a conditional form in order to relate it to $H(Y|X)$.

Show full proof

The chain rule is a matter of rewriting:

$$\begin{aligned}
H(XY) &= -\sum_{x,y} P_{XY}(x,y)\log P_{XY}(x,y) \\
&= -\sum_{x,y} P_{XY}(x,y)\log\big(P_X(x)P_{Y|X}(y|x)\big) \\
&= -\sum_{x,y} P_{XY}(x,y)\log P_X(x) - \sum_{x,y} P_{XY}(x,y)\log P_{Y|X}(y|x) \\
&= -\sum_{x} P_X(x)\log P_X(x) - \sum_{x,y} P_{XY}(x,y)\log P_{Y|X}(y|x) \\
&= -\sum_{x} P_X(x)\log P_X(x) - \sum_{x} P_X(x)\sum_{y} P_{Y|X}(y|x)\log P_{Y|X}(y|x) \\
&= H(X) + H(Y|X)\,.
\end{aligned}$$

This was to be shown.

The chain rule immediately results in the so-called 'independence bound':

Corollary: Subadditivity (independence bound)

$$H(XY) \leq H(X) + H(Y)\,.$$

Equality holds iff X and Y are independent.

Proof

$$H(XY) = H(X) + H(Y|X) \leq H(X) + H(Y),$$

where the equality is due to the chain rule and the inequality is due to **the upper bound on the conditional entropy** that $H(Y|X) \leq H(Y)$ (and equal iff X and Y are independent).

We can naturally generalize the definition of conditional entropy by applying it to the conditional distribution $P_{XY|\mathcal{A}}$; this results in $H(X|Y, \mathcal{A})$, the entropy of X given Y and conditioned on the event $\mathcal{A}$. Since the entropy is a function of the *distribution* of a random variable, the chain rule also holds when conditioning on an event $\mathcal{A}$. Furthermore, it holds that

$$H(X|YZ) = \sum_z P_Z(z) H(X|Y, Z{=}z)\,,$$

which is straightforward to verify. With this observation, we see that the chain rule generalizes as follows.

Corollary: generalized chain rule

Let X, Y and Z be random variables. Then

$$H(XY|Z) = H(X|Z) + H(Y|XZ)\,.$$

Inductively applying the (generalized) chain rule implies that for any sequence $X_1, \ldots, X_n$ of random variables:

$$H(X_1 \cdots X_n) = H(X_1) + H(X_2|X_1) + \cdots + H(X_n|X_{n-1} \cdots X_1)\,.$$

Combining this with the upper bound on the conditional entropy, we see that subadditivity generalizes to

$$H(X_1 \cdots X_n) \leq \sum_{i=1}^{n} H(X_i).$$

Definition: Mutual Information

Definition: Mutual information

Let X and Y be random variables. The mutual information $I(X;Y)$ of X and Y is defined as

$$I(X;Y) = H(X) - H(X|Y).$$

Thus, in a sense, mutual information reflects the reduction in uncertainty about X when we learn Y. Verify the following properties of the mutual information:

$$\begin{aligned}
&I(X;Y) = H(X) + H(Y) - H(XY) \\
&I(X;Y) = I(Y;X) && (\text{``symmetry''}) \\
&I(X;Y) \geq 0 && (\text{``positivity''}) \\
&I(X;Y) = 0 \text{ iff } X \text{ and } Y \text{ are independent} \\
&I(X;X) = H(X) && (\text{``self-information''})
\end{aligned}$$

Entropy Diagrams for Two Random Variables

The relations between entropy, joint entropy, conditional entropy, mutual information, and conditional mutual information can be summed up visually. Here, we show how to do this when only two random variables are involved: all information-theoretic measures can be nicely represented by means of a Venn-diagram-like **entropy diagram**. From the diagram, one can for instance easily read off the relations $H(X|Y) \leq H(X)$, $I(X;Y) = H(X) + H(Y) - H(XY)$, etc. The case of three random variables will be treated later in this course.

Use the tool below to play around with different distributions of X and Y, so you can see how the distribution affects the different entropic quantities. You can select one of the pre-defined distributions on the top, or use the arrows to move some of the probability weight around. Hover over the formulas on the bottom left to see which parts of the distribution were involved, and which part of the diagram corresponds to that quantity.

Direct link to interactive graph

Definition: Cross Entropy

Suppose you draw samples from a set $\mathcal{X}$, according do a distribution P, while *thinking* you are drawing those samples according to a distribution Q. How surprised do you expect to be? This surprisal value is expressed by the **cross entropy**, a quantity that is very closely related to the relative entropy.

Definition: Cross Entropy

The cross entropy of two probability distributions P and Q over the same $\mathcal{X}$ is defined by

$$H_C(P,Q) := -\sum_{x\in\mathcal{X},P(x)>0} P(x)\log Q(x).$$

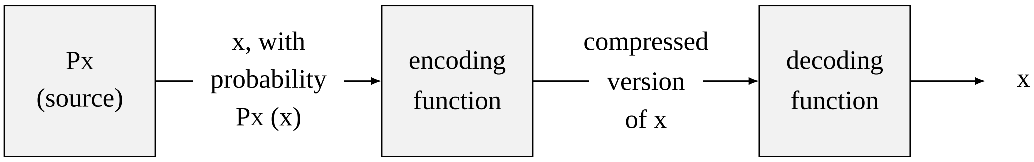
There are a few things to note about this definition:

- Note that if $Q(x) = 0$ for some x with $P(x) > 0$, then $H_C(P,Q) = \infty$.
- Also note that H_C is a function of the *distributions* P and Q. With regular Shannon entropy, we can be sloppy with the notation, and write $H(X)$ instead of the more correct $H(P)$. Here, that sloppy notation would lead to ambiguous expressions.
- In the literature, the notation $H(P,Q)$ is often used to denote the cross entropy. This notation can potentially be confused with the notation for joint entropy, so we use the subscript C to make the distinction explicit.

The cross entropy often pops up in topics related to machine learning: a system usually learns from a set of *training data*, from which it hypothesizes a certain distribution Q (on letter frequencies, cluster sizes, or whatever the system is learning about). When the system is released into the real world, it encounters a (possibly) different distribution P. The cross entropy $H_C(P,Q)$ quantifies how well the system performs in the real world when it was trained on the training set.

Introduction to Source Coding

Suppose we sample x from a distribution P_X with image $\mathcal{X}$. In the context of data compression, P_X is typically called a **source** that emits value $x \in \mathcal{X}$ with probability $P_X(x)$. We want to compress (or encode) symbols x sampled from P_X in such a way that we can later decompress (or decode) it reliably, without losing any information about the value x.



A counting argument shows that it is possible to encode the elements of $\mathcal{X}$ by bit strings of length n, where $n = \lceil \log(|\mathcal{X}|) \rceil$: we simply list all elements of $\mathcal{X}$, and use the (binary) index of x in the list as its encoding. Thus, to store or to transmit an element $x \in \mathcal{X}$, n bits of information always suffice. However, if not all $x \in \mathcal{X}$ are equally likely according to P_X, one should be able to exploit this to achieve codes with shorter *average* length. The idea is to use encodings of varying lengths, assigning shorter codewords to the elements in $\mathcal{X}$ that have higher probabilities, and vice versa.

In the video, Alice and Bob communicate by encoding their messages (dice rolls) from $\mathcal{X} = \{2, 3, \ldots, 12\}$ into a unitary alphabet $\{1\}$, where each 1 stands for a pluck of the wire. For example, the roll 8 is encoded as 111, or three plucks.

Exercise

At the end of the video, Bob gets a better idea. He notices that they can pluck the wire in two different ways that are easy to distinguish: long or short. Can you design a code using this binary alphabet of plucks? How long are your codewords on average?

Show solution

The code on the right is an example of a code that Alice and Bob may use: 0 stands for a short pluck, 1 stands for a long one. Each die roll has a different codeword, and short codewords are assigned to the most likely outcomes. The expected codeword length is

$$\frac{1}{36}\cdot 3+\frac{2}{36}\cdot 3+\frac{3}{36}\cdot 2+\ldots+\frac{1}{36}\cdot 3=\frac{35}{18}\approx 1.944.$$

So on average, Alice and Bob expect to pluck the wire a little less than two times per die roll they want to communicate. However, if they want to communicate a list of die roll outcomes, they run into a problem: if Alice receives 011, how can she tell whether Bob sent the list [7,4], or [5,6], or even [2]?

Die roll	Codeword
2	011
3	001
4	11
5	01
6	1
7	0
8	00
9	10
10	000
11	010
12	100

This problem is resolved in the code on the right: confusions do not arise even when variable-length lists of messages are sent. The average codeword length is longer, however: roughly 3.306 plucks on average. In this module, you will encounter several algorithms for constructing such codes yourself for any given probability distribution.

Die roll	Codeword
2	11110
3	0010
4	0011
5	100
6	000
7	010

Die roll	Codeword
8	011
9	101
10	110
11	1110
12	11111

The question we will answer in this module is: how short can codes be in general (on average over repeated samples x from P_X)? We explore both **lossless** codes (where we want to recover the original data with certainty) and **lossy** codes (where with small probability, the data is lost).

Definition: Prefix-freeness

One convenient way to guarantee that a code is unique decodable is to require it to be prefix-free:

Definition: Prefix-free code

A binary symbol code $C : \mathcal{X} \to \{0,1\}^*$ is prefix-free (or: **instantaneous**) if for all $x, x' \in \mathcal{X}$ with $x \neq x'$, $C(x)$ is *not* a prefix of $C(x')$.

With a prefix-free encoding, the elements $x_1, \ldots, x_m$ can be uniquely recovered from $C(x_1)| \cdots |C(x_m)$, simply by reading the encoding from left to right one bit at a time: by prefix-freeness it will remain unambiguous as reading continues when the current word terminates and the next begins. This is a loose argument for the following proposition:

Proposition

If a code $\mathcal{C}$ is prefix-free and $\mathcal{C} \neq \perp$ then $\mathcal{C}$ is uniquely decodable.

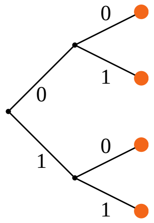
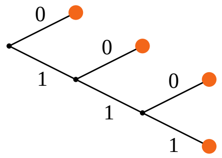
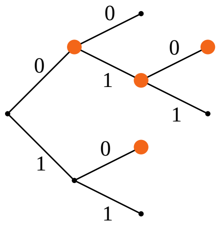
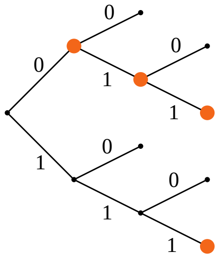
The other direction does not hold: uniquely decodable codes need not be prefix-free. A prefix-free code is appealing from an efficiency point of view, as it allows to decode "on the fly". For a general uniquely decodable code one may possibly have to inspect all bits in the entire string before being able to even recover the first word.

Example

The following are all codes for the source P_X, with $\mathcal{X} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$:

x	$P_X(x)$	$C_1(x)$	$C_2(x)$	$C_3(x)$	$C_4(x)$
a	0.5	00	0	0	0
b	0.25	01	10	010	01
c	0.125	10	110	01	011
d	0.125	11	111	10	111

These codes can be visualised a as binary trees, with marked codewords, as follows:

			
C_1	C_2	C_3	C_4

Which of these codes are uniquely decodable? Which are prefix-free?

Show solution

C_1 and C_2 are prefix-free, and therefore also uniquely decodable. C_3 is not uniquely decodable, as $C_3(ad) = C_3(b)$. C_4 is not prefix-free, but it is uniquely decodable, since it can be decoded from right to left (it is ``suffix-free"). Note that the binary trees for the prefix-free codes C_1, C_2 only have codewords at the leaves.

Theorem: Shannon's Source-Coding Theorem (Optimal Codes)

We now know that prefix-free codes can achieve the same minimal code lengths for a source P_X as the more general class of uniquely decodable codes. How small is this minimal code length in general? In this section we explore the following relation between the minimal code length and the entropy of the source:

Theorem: Shannon's source-coding theorem (for symbol codes)

For any source P_X, we have the following bounds:

$$H(X) \leq \ell_{\min}(P_X) \leq H(X) + 1.$$

$H(X) \leq \ell_{\min}(P_X)$

The proof relies on Kraft's inequality. Let C be a code, and write ℓ_x for $\ell(C(x))$ as a notational convenience. For the lower bound, we have that

$$\begin{aligned} H(X) - \ell_C(P_X) &= -\sum_{x\in\mathcal{X}} P_X(x)\log(P_X(x)) - \sum_{x\in X} P_X(x)\ell_x \\ &= \sum_{x\in\mathcal{X}} P_X(x)\left(-\log(P_X(x)) - \log\left(2^{\ell_x}\right)\right) \\ &= \sum_{x\in\mathcal{X}} P_X(x)\log\left(\frac{1}{P_X(x)\cdot 2^{\ell_x}}\right) \\ &\leq \log\left(\sum_{x\in\mathcal{X}}\frac{1}{2^{\ell_x}}\right) && \text{(by Jensen's inequality)} \\ &\leq \log(1) = 0 && \text{(by Kraft's inequality)} \end{aligned}$$

$\ell_{\min}(P_X) \leq H(X) + 1$

For the upper bound, let us denote by ℓ_x the surprisal value in bits rounded up to the next integer, i.e. for any $x \in \mathcal{X}$,

$$\ell_x := \left\lceil \log\frac{1}{P_X(x)} \right\rceil,$$

and note that

$$\sum_{x\in\mathcal{X}} 2^{-\ell_x} \leq \sum_{x\in\mathcal{X}} 2^{-\log\frac{1}{P_X(x)}} = \sum_{x\in\mathcal{X}} P_X(x) = 1.$$

Therefore, by Kraft's inequality, there exists a prefix-free code C such that $\ell(C(x)) = \ell_x$ for all $x \in \mathcal{X}$. This code satisfies

$$\begin{aligned}
\ell_C(P_X) &= \sum_{x\in\mathcal{X}} P_X(x)\ell_x \\
&\leq \sum_{x\in\mathcal{X}} P_X(x)\left(\log\frac{1}{P_X(x)} + 1\right) \\
&= -\sum_{x\in\mathcal{X}} P_X(x)\log P_X(x) + \sum_{x\in\mathcal{X}} P_X(x) \\
&= H(X) + 1.
\end{aligned}$$

We have thus constructed a code C with $\ell_C(P_X) \leq H(X) + 1$, so $\ell_{\min}(P_X) \leq H(X) + 1$.

Arithmetic Codes

Visualization using "language model" of controller presses (X,Y,A,B)

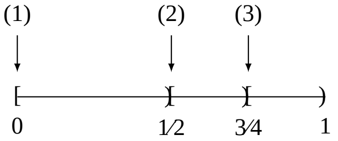
In this section, we study a different kind of code that can handle context-dependent distributions, unlike the Huffman code. Binary representations of numbers in the interval $[0,1)$ as studied in the previous exercise give rise to a very elegant code:

Definition: Arithmetic code

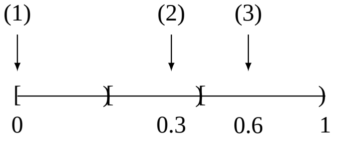
Given a source P_X with $\mathcal{X} = \{x_1, .., x_m\}$, construct the arithmetic code as follows. Divide the interval $[0,1)$ into disjoint subintervals $I_{x_j} = [a_j, a_{j+1})$, where $a_1, \dots, a_{m+1}$ are defined such that $a_{j+1} - a_j = P_X(x_j)$, and $a_1 = 0, a_{m+1} = 1$. The encoding $AC(x_j)$ of the element x_j is the (shortest possible) standard binary representation of some number in the interval I_{x_j}.

Example

Let X be a random variable with $\mathcal{X} = \{1,2,3\}$ and $P_X(1) = \frac{1}{2}$, $P_X(2) = P_X(3) = \frac{1}{4}$. The arithmetic code is constructed by first determining the intervals:



This image results in the arithmetic code $\mathcal{C}$ with $\mathcal{C}(1) = 0$ (the representation of 0), $\mathcal{C}(2) = 1$ (the representation of $\frac{1}{2}$), $\mathcal{C}(2) = 11$ (the representation of $\frac{3}{4}$). For P_X, the codewords happen to fall exactly *on* the boundaries of the intervals. This is not always the case, however. The same code would have resulted from this procedure if we started with the random variable Y with $\mathcal{Y} = \{1,2,3\}$ and $P_Y(1) = P_Y(2) = 0.3$ and $P_Y(3) = 0.4$:



Proposition

For any (X, P_X), the arithmetic code has average length $\ell_{AC}(P_X) \leq H(X) + 1$.

Proof

Let $x \in \mathcal{X}$, and define $\ell_x := \lceil \log(1/P_X(x)) \rceil$ to be the rounded surprisal value of x. Then

$$2^{-\ell_x} = 2^{-\lceil \log(1/P_X(x)) \rceil} \leq 2^{-\log(1/P_X(x))} = 2^{\log P_X(x)} = P_X(x).$$

Therefore, since the size of the interval I_x is $P_X(x)$, there must exist an integer $0 \leq s_x < 2^{\ell_x}$ such that $s_x \cdot 2^{-\ell_x}$ lies in the interval I_x. This number $s_x \cdot 2^{-\ell_x}$ has a binary representation of length $\ell_x \leq -\log P_X(x) + 1$. Repeating this argument for every x, we obtain

$$\ell_{AC}(P_X) = \mathbb{E}[\ell(AC(X))] = \sum_x P_X(x)\ell(AC(x)) \leq \sum_x P_X(x)(-\log P_X(x) + 1) = H(X) + 1.$$

As we can see from the example above, this construction for arithmetic codes does not necessarily yield prefix-free codes. However, at the expense of one extra bit of code (on average), the construction can be adapted into a prefix-free code. One example of this is the **Shannon-Fano-Elias**

code (see **Cover/Thomas**, Section 5.9 or **Wikipedia**), which provides a more sophisticated way of selecting a number within each interval than simply selecting the number with the shortest binary representation. This alternative selection procedure ensures prefix-freeness. Another option is to select *binary intervals* within each interval:

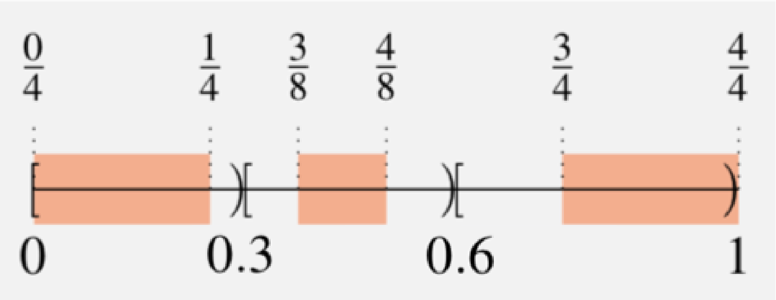
Definition: Arithmetic code (prefix-free version)

A prefix-free arithmetic code is identical to the definition above, except that the encoding $AC^{pf}(x_j)$ of the element x_j is now the name of a largest binary interval that fits entirely in I_{x_j}.

Similarly to the proposition above, it can be shown that for any source P_X, $\ell_{AC^{pf}}(P_X) \leq H(X) + 2$. Note that we get prefix-freeness only at the expense of an extra bit on average.

Example

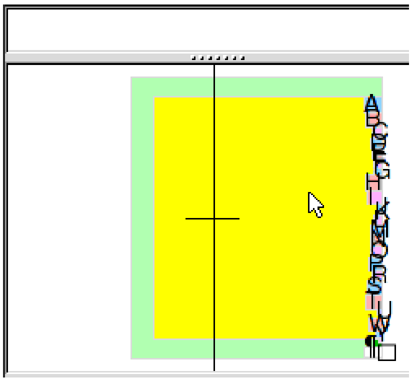
Let Y be the random variable as in the example above, that is, $P_Y(1) = P_Y(2) = 0.3$ and $P_Y(3) = 0.4$. The prefix-free code for Y is constructed as follows:



This results in the codewords $\mathcal{C}(1) = 00$, $\mathcal{C}(2) = 011$, and $\mathcal{C}(3) = 11$.

The arithmetic code is slightly less efficient than the Huffman code in terms of average codeword length. A big advantage is the way it is able to adapt to changing distributions, such as when we are encoding a stream of English text. Suppose we are given the (not necessarily i.i.d.) random variables $X_1, X_2, \ldots, X_n$, and we want to encode the source $P_{X_1 X_2 \cdots X_n}$. We start by dividing the interval $[0, 1)$ into subintervals according to P_{X_1}. If, for example, the event $X_1 = b$ happens, we zoom into the interval corresponding to b, and subdivide *that* interval according to $P_{X_2|X_1}$, so that the sizes of these intervals add up to $P_{X_1}(b)$.

The concept of arithmetic coding is exploited as an accessibility tool in the keyboard alternative **Dasher**, invented by the group of David MacKay at Cambridge University, UK. Try to download and play with it, it's great fun!



Source Coding using Typical Sets

The concept of typical sets is useful for designing codes for a source P_X. Instead of encoding the source symbol-by-symbol, we will encode the source symbols in blocks of n symbols at a time.

This strategy can be used to design either a lossy or a lossless code. For a lossy code, we notice that with overwhelming probability, a sequence of n iid samples from P_X is typical, so it suffices to assign binary labels of length (at most) $\lceil n(H(X) + \epsilon) \rceil$ to the elements of $A_\epsilon^{(n)}$, and assign some constant (dummy) codeword to all elements outside of the set. Decoding this dummy codeword will result in an error (data loss), but this error occurs with probability at most ϵ.

The above scheme can be extended to a lossless version by assigning longer labels to the elements outside of $A_\epsilon^{(n)}$, for example binary labels of length $\lceil \log |\mathcal{X}|^n \rceil = \lceil n \log |\mathcal{X}| \rceil$. An extra 'flag' bit is needed to indicate whether the element is inside or outside the typical set. For large enough n, this code is quite efficient:

Theorem

Let $X_1, \ldots, X_n$ be i.i.d. real random variables with respect to the set $\mathcal{X}$, and distributed according to P_X. Let $\epsilon > 0$. Then there exists a lossless code $\mathcal{X}^n \to \{0,1\}^*$ such that, for sufficiently large n, $\mathbb{E}[\frac{1}{n}\ell(X^n)] \leq H(X) + \epsilon$.

Proof

Consider the code described above: the code consist of a flag bit (indicating whether or not the element is inside the typical set), followed by either a short label (for elements in the typical set) or a longer one (for elements outside of it). Let $\epsilon' > 0$ (we will specify the value of ϵ' later). Let n be large enough such that $P[A_{\epsilon'}^{(n)}] > 1 - \epsilon'$ (see the second property of typical sets). Then

$$\begin{aligned}
\mathbb{E}[\ell(X^n)] &= \sum_{\vec{x}\in\mathcal{X}^n} P_{X^n}(\vec{x})\ell(\vec{x}) \\
&= \sum_{\vec{x}\in A_{\epsilon'}^{(n)}} P_{X^n}(\vec{x})\ell(\vec{x}) + \sum_{\vec{x}\notin A_{\epsilon'}^{(n)}} P_{X^n}(\vec{x})\ell(\vec{x}) \\
&\le P[A_{\epsilon'}^{(n)}]\cdot(\lceil n(H(X)+\epsilon')\rceil + 1) + P[\overline{A_{\epsilon'}^{(n)}}]\cdot(\lceil n\log|\mathcal{X}|\rceil + 1) \\
&\le P[A_{\epsilon'}^{(n)}]\cdot(n(H(X)+\epsilon') + 2) + P[\overline{A_{\epsilon'}^{(n)}}]\cdot(n\log|\mathcal{X}| + 2) \\
&\le n(H(X)+\epsilon') + \epsilon'\cdot n\log|\mathcal{X}| + 2 \\
&= n(H(X)+\epsilon),
\end{aligned}$$

where $\epsilon = \epsilon' + \epsilon'\log|\mathcal{X}| + \frac{2}{n}$ (note that ϵ can be made arbitrarily small by choosing ϵ' and n wisely). The +1 in the first inequality is a consequence of the `flag' bit.

For large enough blocks of symbols, the flag bit becomes irrelevant. Typical sets thus allow the construction of an efficient code without the 1 bit of overhead that symbol codes may necessarily have. However, this efficiency is only guaranteed for 'sufficiently large n', a rather theoretical condition that may not be achievable in practice.

We conclude this chapter by showing that the typical set is in a sense 'optimal', i.e. that picking a smaller set instead of the typical set does not allow for much shorter codewords on average in a lossy setting, not even if we allow rather large error probabilities by allowing about half of the elements to lie outside of the typical set.

Let $B_\delta^{(n)}$ denote the smallest subset of $\mathcal{X}^n$ such that $P[B_\delta^{(n)}] > 1-\delta$ (for some parameter $\delta > 0$). $B_\delta^{(n)}$ can be explicitly constructed by, for example, ordering $\mathcal{X}^n$ in order of decreasing probability, and adding elements to $B_\delta^{(n)}$ until the probability threshold of $1-\delta$ is reached. The following theorem states that even for large values of δ, we still need almost $nH(X)$ bits to denote an element from $B_\delta^{(n)}$.

Theorem

Let $X_1, \ldots, X_n$ be i.i.d. random variables distributed according to P_X. For any $\delta < \frac{1}{2}$, and any $\delta' > 0$, if $P[B_\delta^{(n)}] > 1-\delta$, then

$$\frac{1}{n}\log|B_\delta^{(n)}| > H(X) - \delta',$$

for sufficiently large n.

Proof

Let $\delta, \epsilon < \frac{1}{2}$, and consider some $B_\delta^{(n)}$ such that $P[B_\delta^{(n)}] > 1 - \delta$. By **the second property of typical sets**, $P[A_\epsilon^{(n)}] > 1 - \epsilon$, for large enough n. Thus, by the union bound,

$$\begin{aligned}
1 - \epsilon - \delta &< 1 - P[\overline{A_\epsilon^{(n)}}] - P[\overline{B_\delta^{(n)}}] \\
&\leq 1 - P[\overline{A_\epsilon^{(n)}} \cup \overline{B_\delta^{(n)}}] \\
&= P[A_\epsilon^{(n)} \cap B_\delta^{(n)}] \\
&= \sum_{\vec{x} \in A_\epsilon^{(n)} \cap B_\delta^{(n)}} P_{X^n}(\vec{x}) \\
&\leq \sum_{\vec{x} \in A_\epsilon^{(n)} \cap B_\delta^{(n)}} 2^{-n(H(X)-\epsilon)} \\
&= |A_\epsilon^{(n)} \cap B_\delta^{(n)}| \cdot 2^{-n(H(X)-\epsilon)} \\
&\leq |B_\delta^{(n)}| \cdot 2^{-n(H(X)-\epsilon)}.
\end{aligned}$$

Rearranging this expression and taking the logarithm, we get

$$H(X) - \epsilon + \frac{1}{n} \log(1 - \epsilon - \delta) < \frac{1}{n} \log |B_\delta^{(n)}|.$$

If we now set $\delta' := \epsilon - \frac{1}{n} \log(1 - \epsilon - \delta)$, then

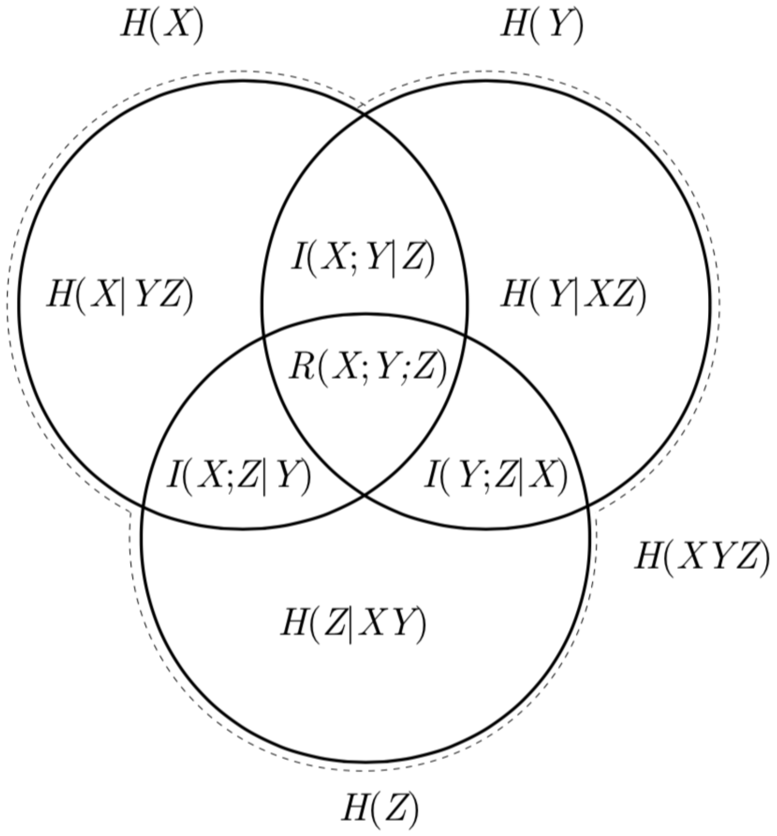
$$H(X) - \delta' < \frac{1}{n} \log |B_\delta^{(n)}|,$$

as desired. Observe that we can make the expression for δ' as small as desired by choosing a small enough $\epsilon > 0$ and a large enough n, even if δ is rather large.

Entropy Diagrams for Three Random Variables

Just like the **entropy diagrams for two random variables**, we can visualize the relations between entropy, conditional entropy, mutual information, and conditional mutual information for three random variables. From the diagram, one can easily read off all the relations between the information-theoretic measures, like for instance $H(X|YZ) = H(X) - I(X;Z) - I(X;Y|Z)$, which is a relation that is otherwise not immediately obvious.

One subtlety with the entropy diagram for three random variables is that the "area in the middle", $R(X;Y;Z) := I(X;Y) - I(X;Y|Z)$, may be *negative*. All other areas and quantities in the diagram are non-negative.



Minimum Key Length for Perfectly Secure Encryption

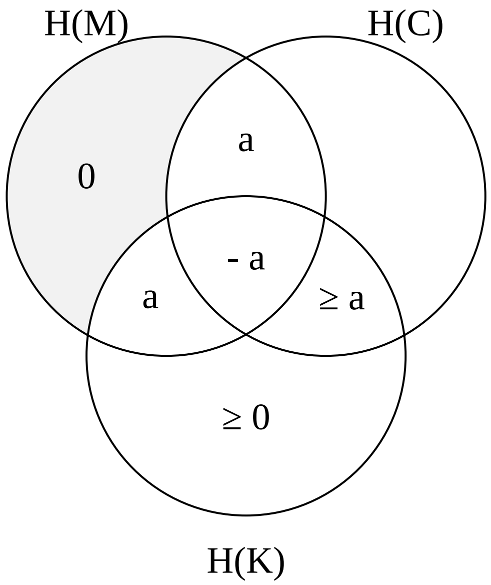
It turns out to be impossible to design an encryption scheme that provides both perfect security and short keys. So even though the one-time pad may seem inefficient, its key length is optimal for a perfectly secure scheme.

Theorem (Shannon, 1949)

For any perfectly secure encryption scheme, it holds that $H(K) \geq H(M)$.

Proof

Again, we turn to entropy diagrams. Write $a = I(M; C|K) \geq 0$. using the fact that $I(M; K) = 0$ (setup assumption) and $I(M; C) = 0$ (security), we can fill in the entropy diagram as follows:



Note that $I(K; C|M) \geq a$ follows from the fact that $I(C; K) \geq 0$ and $R(C; K; M) = -a$. From the diagram, we observe that $H(M) = a$, and that $H(K) \geq a$. Hence, $H(K) \geq H(M)$.

created: 2018-12-11

Definition: Longer Markov Chains

We can extend the definition of Markov chains to more than three variables:

Definition: Markov chain (of length n)

The random variables $X_1, X_2, \ldots, X_n$ form a Markov chain (notation: $X_1 \to X_2 \to \cdots \to X_n$) if for all $3 \leq i \leq n$,

$$P_{X_i|X_1 \cdots X_{i-1}} = P_{X_i|X_{i-1}} \, .$$

Markov chains of length n exhibit similar properties to the properties we have seen for Markov chains of length 3. In particular, the reverse chain is also a Markov chain ($X_n \to \cdots \to X_2 \to X_1$), and a more general form of the data-processing inequality holds in the sense that the further apart two variables are in the chain, the less correlated they are.

Proposition

If $X_1 \to X_2 \to X_3 \to X_4$ is a Markov chain, the following are Markov chains as well:

a. $X_1 \to X_2 \to X_3$
b. $X_2 \to X_3 \to X_4$
c. $X_1 \to X_2 X_3 \to X_4$
d. $X_4 \to X_3 \to X_2 \to X_1$

Proof

left as exercise

In the exercises, we will prove that if $X_1, X_2, \ldots, X_n$ forms a Markov chain, then for all $1 \leq i \leq j \leq k \leq n$: $I(X_i, X_j) \geq I(X_i, X_k)$. This is a generalized form of the data-processing inequality.

Markov Process: Irreducibility, Periodicity, Convergence

Every time-invariant Markov process has a stationary distribution, but it might not be unique, and the process might never reach the stationary distribution. For uniqueness and convergence, we need additional requirements on the Markov process.

Definition: Irreducible Markov process

A time-invariant Markov process is irreducible if every state is reachable from any other state in a finite number of steps.

The process in **Example 2** is irreducible: the state `a` is reachable from `b` and vice versa. The process in **Example 1** is not irreducible. For example, the state 0 is not reachable from the state 2.

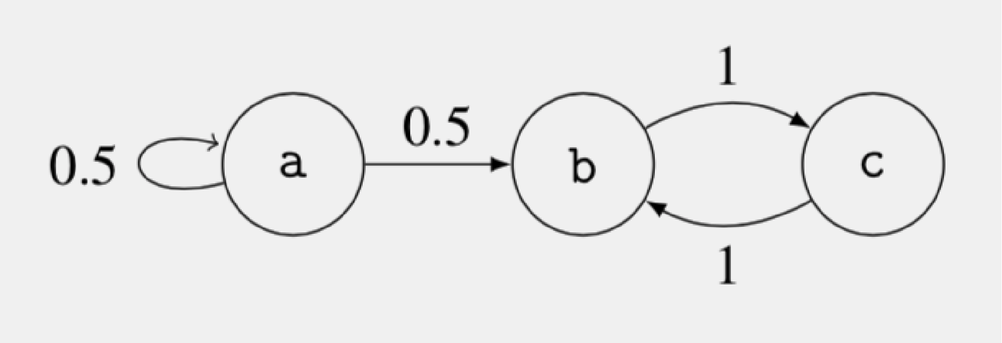
Definition: Aperiodic Markov process

A state in a time-invariant Markov process is aperiodic if the greatest common divisor of all path lengths from that state to itself is 1. The process itself is aperiodic if all states are aperiodic.

The processes in **Example 1** and **Example 2** are both aperiodic. In both examples, every state is reachable from itself with paths of any length, so the greatest common divisor of the path lengths is always 1. Below, we will present an example of a process that is periodic (i.e., not aperiodic).

Example 3: Periodic Markov process

Consider the process that starts in state `a` with probability 1, and is further described by the following state diagram:



The state `a` is aperiodic, but `b` and `c` are not. The paths from state `b` to itself are

all of even length, as are the paths from state c to itself. A typical run of this process might look like aaaaabcbcbcbc; the period 2 is clearly visible in this run.
decide whether this button is necessary, and whether the title of this block should be question instead of example

Proposition

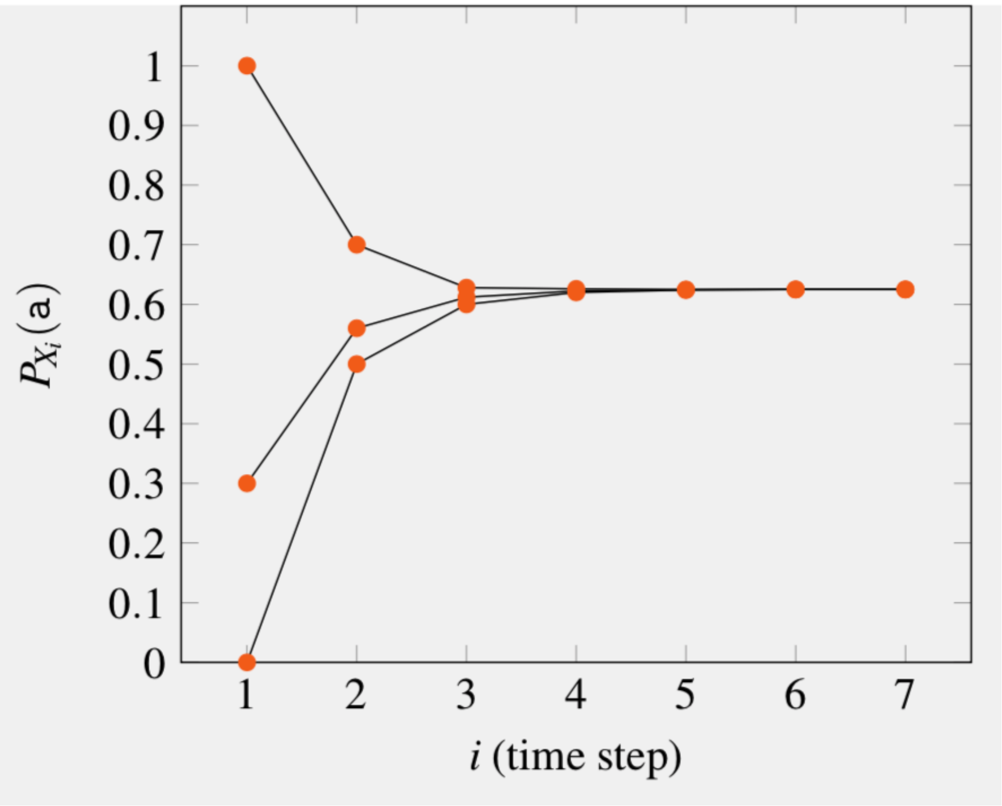
If a time-invariant Markov chain is finite-state, irreducible and aperiodic, then there exists a unique stationary distribution. Moreover, from any initial distribution, the distribution of X_n tends to the stationary distribution as $n \to \infty$.
Paste proof here

Example 2, continued

Consider again the process of Example 2. This process does have a stationary distribution, because it is finite-state, irreducible and aperiodic. Let $\mu_{\mathtt{a}}$ denote the probability of observing an a in the stationary distribution, and $\mu_{\mathtt{b}}$ the probability of observing a b. Then the stationary distribution must satisfy the following set of equations:

$$\begin{aligned} \mu_{\mathtt{a}} &= 0.7\mu_{\mathtt{a}} + 0.5\mu_{\mathtt{b}}, \\ \mu_{\mathtt{b}} &= 0.3\mu_{\mathtt{a}} + 0.5\mu_{\mathtt{b}}, \\ \mu_{\mathtt{a}} + \mu_{\mathtt{b}} &= 1. \end{aligned}$$

Solving this set of equations gives the stationary distribution $\mu_{\mathtt{a}} = 0.625$ and $\mu_{\mathtt{b}} = 0.375$. This outcome matches our observations in the figure below. The starting distribution is irrelevant, because in the limit, the stationary distribution is reached. The following plot exemplifies this for three different starting distributions ($P_{X_1}(\mathtt{a}) = 0$, $P_{X_1}(\mathtt{a}) = 0.3$, and $P_{X_1}(\mathtt{a}) = 1$).



created: 2018-12-11

AEP for Ergodic Stationary Processes

Under some natural assumptions, many statements and interpretations of the Shannon entropy we have seen in the context of the asymptotic equipartition property (AEP) can be generalized to the entropy rate $H(\{X_i\})$ of stochastic processes.

Definition: Ergodic Random Processes

A stochastic process $\{X_i\}$ is **ergodic** if its statistical properties can be deduced from a single, sufficiently long, random sample of the process.

There are many equivalent ways of giving precise mathematical definitions of this notion, but this goes beyond the scope of this course. Instead, we consider the following examples.

Example

Suppose we repeatedly pick a letter at random and print it three times: $\texttt{LLL EEE HHH QQQ MMM QQQ OOO TTT EEE YYY XXX GGG}\ldots$

This random process is ergodic (as we see all letters eventually) but not stationary.

On the other hand, if we pick a letter at random and print it forever: $\texttt{GGGGGGGGGGGGGGGGGG}\ldots$

This process is stationary, but not ergodic (from one sample of the process, we can only see a single letter).

Shannon–McMillan–Breiman theorem: AEP

If $H(\{X_i\})$ is the entropy rate of a finite-valued stationary ergodic process $\{X_i\}$, then

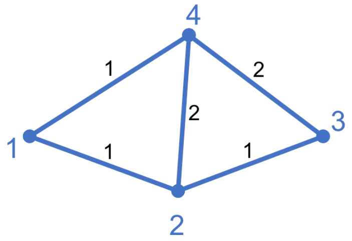
$$-\frac{1}{n}\log P_{X_1,\ldots X_n}(X_1,\ldots X_n) \to H(\{X_i\}) \qquad \text{with probability } 1.$$

In other words, the asymptotic equipartition property holds for such processes: we can define typical sets and all the results about data compression we have seen in the previous module not only hold for iid processes, but more generally for stationary ergodic processes. The entropy rate $H(\{X_i\})$ measures how many bits on average are needed to optimally compress the stationary ergodic process $\{X_i\}$.

Random Walks on Graphs

An important and widely applicable example of a time-invariant Markov process is a random walk on a connected graph G with strictly positive symmetric edge weights $W_{ij} = W_{ji}$. The random walk is defined as follows: at node i, walk to node j with probability $\frac{W_{ij}}{W_i}$ where $W_i := \sum_j W_{ij}$ is the sum of the weights of all edges involving node i, and $W := \frac{1}{2}\sum_i W_i$ is the total of all edge weights.

Example



For the following graph , we have that $W_1 = 2, W_2 = 4, W_3 = 3, W_4 = 5$ and $2 \cdot W = \sum_i W_i = 2 \cdot 7$.

The stationary distribution of this random walk is given by $\mu_i := \frac{W_i}{2W}$, because indeed, at every node i, we have that the sum of all incoming weight is

$$\sum_j \mu_j \frac{W_{ij}}{W_j} = \sum_j \frac{W_j}{2W} \frac{W_{ij}}{W_j} = \frac{W_i}{2W} = \mu_i \,.$$

We continue to compute the entropy rate of this random walk. Assuming we start in the stationary distribution, we can compute the entropy rate as follows.

$$\begin{aligned} H(\{X_i\}) = \sum_i \mu_i H(\dots \frac{W_{ij}}{W_i} \dots) &= -\sum_i \mu_i \sum_j \frac{W_{ij}}{W_i} \log \frac{W_{ij}}{W_i} \\ &= -\sum_{i,j} \frac{W_i}{2W} \cdot \frac{W_{ij}}{W_i} \log\left(\frac{W_{ij}}{2W} \cdot \frac{2W}{W_i}\right) \\ &= -\sum_{i,j} \frac{W_{ij}}{2W} \log\left(\frac{W_{ij}}{2W}\right) + \sum_{i,j} \frac{W_{ij}}{2W} \log\left(\frac{W_i}{2W}\right) \\ &= -\sum_{i,j} \frac{W_{ij}}{2W} \log\left(\frac{W_{ij}}{2W}\right) + \sum_i \frac{W_i}{2W} \log\left(\frac{W_i}{2W}\right) \\ &= H(\dots \frac{W_{ij}}{2W} \dots) - H(\dots \frac{W_i}{2W} \dots) \end{aligned}$$

which is the difference of the entropy of the edge distribution and the entropy of the stationary distribution.

Example, continued

In the example above, the edge distribution is $\frac{1}{14}(1,1,1,2,2,1,1,1,2,2)$ and the stationary distribution is $\frac{1}{14}(2,4,3,5)$, resulting in
$H(\{X_i\}) = H(\frac{1}{14}(1,1,1,2,2,1,1,1,2,2)) - H(\frac{1}{14}(2,4,3,5)) \approx 1.312$

Types of Discrete Channels

Definition: Deterministic channel

A channel is deterministic if $H(Y|X) = 0$. In other words,

$$\forall x \in \mathcal{X}\ \exists y \in \mathcal{Y} : P_{Y|X}(y|x) = 1.$$

Definition: Lossless channel

A channel is lossless (or **ideal**) if $H(X|Y) = 0$ for all input distributions P_X. In other words,

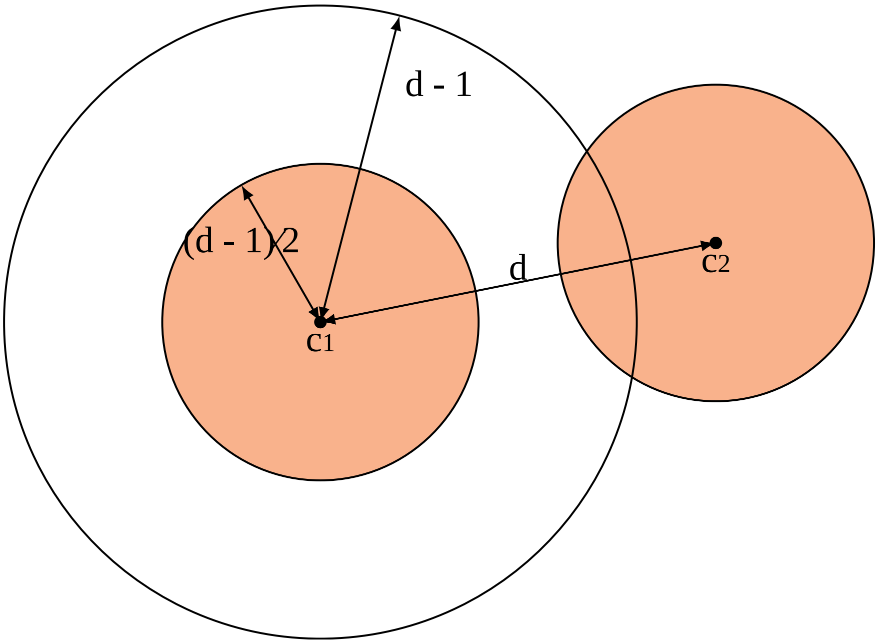
$$\forall y \in \mathcal{Y}\ \exists! x \in \mathcal{X} : P_{Y|X}(y|x) > 0.$$

(the notation $\exists! x$ means that there exists *exactly* one such x.)

In a deterministic channel, the output is completely determined by the input, whereas in a lossless channel, the input is completely determined by the output. A **noiseless** channel is a channel that is both deterministic and lossless.

Error Detection/Correction and the Hamming Bound

In general, a code with minimal distance d can ***detect*** up to $d - 1$ bit flip errors: in $d - 1$ bit flip 'steps', those bit flips can never result in another valid codeword. The code can accurately ***correct*** up to $\frac{d-1}{2}$ bit flip errors. Look at the diagram below to understand why: if two codewords c_1 and c_2 are guaranteed to be at least d bit flips apart, then the set of strings that result from $\frac{d-1}{2}$ bit flips on c_1 (the orange circle on the left) never overlaps with the set of strings that result from $\frac{d-1}{2}$ bit flips on c_2 (the orange circle on the right).



Because every codeword is guaranteed to have this neighborhood around it that it does not share with any other codeword, we can upper bound the total number of codewords in terms of the distance.

Proposition: Hamming bound

If C is a *binary* code of block length n and minimum distance 3, then $|C| \leq \frac{2^n}{n+1}$.

Proof

For each $c \in C$, define the neighborhood of c to be $N(c) := \{y \in \{0,1\}^n \mid d(x,y) \leq 1\}$. Every such neighborhood contains exactly $n+1$ elements. Since $d=3$, $N(c)\cap N(c')=\emptyset$ whenever $c\neq c'$. Hence,

$$2^n \geq |\bigcup_{c\in C} N(c)| = \sum_{c\in C} |N(c)| = |C|\cdot(n+1),$$

and the result follows.

The $[7,4]$ Hamming code is optimal in the sense that it achieves this Hamming bound: it is a code with block length 7 and minimal distance 3, so an upper bound to the codebook size is $\frac{2^7}{7+1}=2^4$. The Hamming code achieves exactly this codebook size.

Minimal Distance of Linear Codes

Apart from the trivial way to determine the minimal distance of a code (which is listing the entire codebook and comparing all the codeword pairs), there is a much faster way if the code is linear. It turns out that it already suffices to consider just the Hamming weights of the (nonzero) codewords:

Proposition

For a linear code C, the minimal distance is equal to the minimal weight of the nonzero codewords.

Proof

The following derivation proves the claim:

$$d_{\min} = \min_{\substack{x,y\in C\\ x\neq y}} d(x,y) = \min_{\substack{x,y\in C\\ x\neq y}} \sum_{i=1}^{n} |x_i - y_i| = \min_{\substack{x,y\in C\\ x\neq y}} d(x-y,0) = \min_{\substack{z\in C\\ z\neq 0}} d(z,0) = \min_{\substack{z\in C\\ z\neq 0}} |z|\,,$$

where |z| denotes the **Hamming weight** of a string z.

An equivalent way to determine the minimal distance of a linear code is possible if the parity check matrix is known.

Proposition

For a linear code C with parity check matrix H, the minimal distance $d_{\min}$ equals the minimum number of columns of H that are **linearly dependent**.

Proof

Left as an exercise.

Shannon Capacity of a Graph

In the previous section it became apparent that for some channels, two or more channel uses can provide a better rate than a single use of those channels. The maximum rate that can be achieved in this way is captured by the notion of Shannon capacity of the confusability graph of the channel.

Definition: Shannon capacity of a graph

The Shannon capacity of a graph is $$c(G) := \sup_{n \in \mathbb{N}} \frac{\log \alpha(G^{\boxtimes n})}{n}.$$

The Shannon capacity represents the *maximum number of bits per channeluse* that can be perfectly communicated over a channel with confusability graph G. Note that our definition in terms of message bits differs from the definition more commonly used in the literature (and on **wikipedia**) about zero-error communication where the capacity is defined without the log as $\sup_{n \in \mathbb{N}} \sqrt[n]{\alpha(G^{\boxtimes n}) }$, thus measuring the maximum number of actual messages (and not bits) per channel use that can be perfectly communicated.

Intuitively, allowing more channel uses can only increase the rate. This is reflected by the fact that we can replace the supremum with a limit, as captured by the following lemma.

Proposition

$$c(G) = \lim_{n \to \infty} \frac{\log \alpha(G^{\boxtimes n})}{n}.$$

Proof

First, note that $\alpha(G^{\boxtimes (k+\ell)}) \geq \alpha(G^{\boxtimes k})\alpha(G^{\boxtimes \ell})$ (you showed this inequality in the previous quiz). It then follows that $$\log \alpha(G^{\boxtimes(k+\ell)}) \geq \log \alpha(G^{\boxtimes k}) + \log \alpha(G^{\boxtimes \ell}),$$ and so the sequence $(\log \alpha(G^{\boxtimes n}))_{n \in \mathbb{N}}$ is superadditive. Then by **Fekete's lemma**, $$\lim_{n \to \infty} \frac{\log \alpha(G^{\boxtimes n})}{n}$$ exists and is equal to the supremum.

The exact computational complexity of $c(G)$ is unknown. It is not even known to be a decidable problem. We have seen that $c(C_5)$ is at least \(\frac{\log 5}

{2}\), a fact that has been known since Shannon showed it in 1956, but how can we decide whether the capacity is actually bigger than that number? This question remained open until Lovasz **showed** in **1979** that the Shannon capacity of \(C_5\) is exactly \(c(C_5) = \frac{\log 5}{2}\). For the relatively small circle of size 7, \(C_7\), the exact Shannon capacity remains unknown. From the fact that \(c(C_5) = \frac{\log 5}{2}\), we know that the 5-letter noisy typewriter does not benefit from more than two channel uses. In fact, all graphs for which the Shannon capacity is known attain that capacity with one, two or an infinite number of channel uses. It is not known if there is a deeper reason for this observed pattern.

It is quite remarkable that in this rather simple and natural problem of zero-error channel coding, we quickly reach the limit of our understanding of the underlying combinatorial problems.

Definition: Channel Capacity

We just discovered that for some noisy channels, zero-error communication is very hard, or even impossible. For example, if Alice and Bob have to communicate over a **binary symmetric channel (BSC)** that has non-zero bit-flip probability, they cannot hope to do any zero-error communication, because the Shannon capacity of the BSC's confusability graph is zero.

We also saw that error-correcting codes can help deal with such inherently noisy channels. Even though the communication error may not become zero, an error-correcting code can increase the probability of receiving the correct message. It does come at a cost, however, because the codewords are longer than the original messages, and so the amount of information that is transmitted *per channel use* does not necessarily increase.

In this final part of the module, we explore the limits of how much information can be sent over a channel if a small error is allowed. Central to our study will be the concept of channel capacity. It reflects the maximum amount of information that could *in principle* be communicated with a single use of a channel. In the next module, we will see how well that theoretical limit can be approached with actual error-correcting codes.

Definition: Channel capacity

The channel capacity C of a discrete, memoryless channel $(\mathcal{X}, P_{Y|X}, \mathcal{Y})$ is given by

$$C := \max_{P_X} I(X;Y).$$

Remember that using a certain input distribution P_X for a channel $P_{Y|X}$ yields a joint input-output distribution P_{XY} which determines the real quantity $I(X;Y)$ we can optimize over. One can **argue** that the maximum is attained and therefore the channel capacity is a well-defined quantity.

Important: the channel capacity is often called the Shannon capacity (of a channel). You should not confuse it with the **Shannon Capacity of a Graph**. Generally, the Shannon capacity of a channel is not equal to the Shannon capacity of its confusability graph.

Example: Capacity of a BSC

What is the capacity (in terms of f) of a binary symmetric channel with parameter $f \in [0, 1/2]$?

Show hint

Rewrite $I(X;Y)$ as $H(Y) - H(Y|X)$ and note that you can compute $H(Y|X)$ without fixing P_X. Then think about how to maximize $H(Y)$.

Show solution

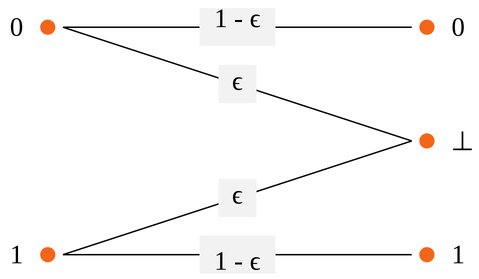
The channel capacity is

$$\begin{aligned}\max_{P_X} I(X;Y) &= \max_{P_X} (H(Y) - H(Y|X)) \\ &= \max_{P_X} \left(H(Y) - \sum_{x \in \mathcal{X}} P_X(x) \cdot H(Y|X=x) \right) \\ &= \max_{P_X} \left(H(Y) - \sum_{x \in \mathcal{X}} P_X(X) \cdot h(f) \right) \\ &= \max_{P_X} (H(Y) - h(f)) \\ &= 1 - h(f).\end{aligned}$$

The last step follows because $H(Y)$ is maximized if Y is uniform, which is achievable by choosing X to be uniform.

Example: Capacity of a BEC

Consider the binary erasure channel (BEC) with $\mathcal{X} = \{0,1\}$ and $\mathcal{Y} = \{0, 1, \perp\}$, where $\perp$ is the **erasure symbol**, and $\epsilon \in [0,1]$ is the **erasure probability**:



What is the channel capacity of the BEC, as a function of ϵ?

Show hint

Contrary to the previous example, break $I(X;Y)$ up as $H(X) - H(X|Y)$, using symmetry of the mutual information. Consider the three possible outputs separately: how much uncertainty is left if you receive output 0? What about output 1? And output $\perp$?

Show solution

Write p for $P_X(0)$.

$$\begin{aligned}\max_{P_X} I(X;Y) &= \max_{P_X}\left(H(X) - H(X|Y)\right)\\ &= \max_{p}\left(h(p) - \sum_{y\in\mathcal{Y}} P_Y(y)\cdot H(X|Y=y)\right)\\ &= \max_{p}\left(h(p) - P_Y(\perp)\cdot h(p)\right)\\ &= \max_{p}\left(h(p)(1-\epsilon)\right)\\ &= 1-\epsilon\,.\end{aligned}$$

Again, the last step follows because $H(X) = h(p)$ is maximized if X is uniform, hence $p = \frac{1}{2}$.

If a channel is memoryless, then using it more than once does not increase the capacity *per transmission*. Note that this is different from the zero-error setting, where multiple channel uses can in fact increase the efficiency of getting information across! This is formally captured in the following lemma, which we state without proof:

Lemma: Multiple Channel Uses

Let $X_1,\ldots,X_n =: X^n$ be n random variables. Let Y^n be the result of passing X^n through a discrete memoryless channel of capacity C. Then for any joint distribution P_{X^n},

$$I(X^n, Y^n) \leq n\cdot C.$$

Definition: Achievable Rate

As stated, the channel capacity reflects the maximum amount of information that could *in principle* be sent over a noisy channel per use of that channel. The question remains whether this capacity is **achievable** by an actual code in the following sense:

Definition: Achievable rate

For a given channel, a rate R is achievable if there exists a sequence of $(2^{n \cdot R}, n)$ codes (for $n = 1, 2, 3, \ldots$) such that $\lambda^{(n)} \xrightarrow{n \to \infty} 0$. Here, $\lambda^{(n)}$ is the **maximum error probability** as defined in the previous module.

Note that any $(2^{n \cdot R}, n)$ code has rate R, since $\frac{1}{n} \cdot \log 2^{n \cdot R} = R$. Thus, a rate R is achievable if there exists a sequence of rate R codes such that increasing the number of channel uses n (often called the block size) reduces the maximum error asymptotically to zero.

This final module will be concerned with proving **Shannon's noisy-channel coding theorem**, which states that any rate R that is strictly below the capacity C is achievable, and conversely, that any rate strictly above is not achievable.

Joint Asymptotic Equipartition Property (Joint AEP)

In the examples on the previous page, we have gathered some intuition about jointly typical sets. A (jointly) typical set is usually relatively small compared to the total sample space, but for large n, much of the probability lies on its elements.

We also see the effect of mutual information between X and Y. If X and Y are close to independent, the typical set has a rectangular shape: independently selecting a typical X sample and a typical Y sample will certainly yield a jointly typical element. On the other hand, if X and Y are highly dependent on each other, the typical set has a diagonal structure: there is a significant probability that sampling a typical X and a typical Y independently yields a pair that is not jointly typical. These observations lead us to a joint version of the **asymptotic equipartition property**, describing some properties of jointly typical sets:

Theorem: Joint Asymptotic Equipartition Property (Joint AEP)

Let X^nY^n be i.i.d. with respect to P_{XY}. Then:

1. $P_{X^nY^n}(A_\epsilon^{(n)}) \xrightarrow{n\to\infty} 1$.
2. $|A_\epsilon^{(n)}| \leq 2^{n(H(XY)+\epsilon)}$.
3. For random variables $\tilde{X}$ and $\tilde{Y}$, if $\tilde{X}^n\tilde{Y}^n$ is i.i.d. with respect to $P_X \cdot P_Y$, then
$$P[(\tilde{X}^n\tilde{Y}^n) \in A_\epsilon^{(n)}] \leq 2^{-n(I(X;Y)-3\epsilon)}.$$
4. For random variables $\tilde{X}$ and $\tilde{Y}$, if $\tilde{X}^n\tilde{Y}^n$ is i.i.d. with respect to $P_X \cdot P_Y$, then for large enough n,
$$P[(\tilde{X}^n\tilde{Y}^n) \in A_\epsilon^{(n)}] \geq (1-\epsilon)2^{-n(I(X;Y)+3\epsilon)}.$$

Proof

We show the statements separately, for an arbitrary X^nY^n that is i.i.d. with respect to P_{XY}.

1. The first statement follows from the weak law of large numbers. Similarly to the **proof of the asymptotic equipartition property**, we have that

$$\begin{aligned} -\frac{1}{n}\log P_{X^n}(X^n) &= -\frac{1}{n}\sum_{i=1}^{n}\log P_X(X) \\ &\overset{p}{\to} -\mathbb{E}[\log P_X(X)] \\ &= H(X). \end{aligned}$$

Writing out the definition of convergence in probability, we have that for all $\epsilon > 0$, there exists an $n_1 \in \mathbb{N}$ such that for all $n > n_1$,

$$\begin{aligned} P\left[\left|-\frac{1}{n}\log P_{X^n}(X^n) - H(X)\right| \geq \epsilon\right] &\leq P\left[\left|-\frac{1}{n}\log P_{X^n}(X^n) - H(X)\right| \geq \frac{\epsilon}{3}\right] \\ &< \frac{\epsilon}{3}. \end{aligned}$$

In the above, we can divide ϵ by 3 because the second inequality holds for *all* $\epsilon > 0$, so in particular also for $\frac{\epsilon}{3}$. We can derive statements about Y^n and X^nY^n that are similar to this bound above. Doing so, we find $n_2, n_3 \in \mathbb{N}$ for Y^n and X^nY^n, respectively. For all $\epsilon > 0$, we can now choose $n_0 := \max\{n_1, n_2, n_3\}$, and find that for all $n > n_0$,

$$\begin{aligned} P\left[(X^n, Y^n) \in A_\epsilon^{(n)}\right] &= 1 - P\left[(X^n, Y^n) \notin A_\epsilon^{(n)}\right] \\ &\geq 1 - \left(P\left[\left|-\frac{1}{n}\log P_{X^n}(X^n) - H(X)\right| \geq \epsilon\right] + \right. \\ &\qquad P\left[\left|-\frac{1}{n}\log P_{Y^n}(Y^n) - H(Y)\right| \geq \epsilon\right] + \\ &\qquad \left. P\left[\left|-\frac{1}{n}\log P_{X^nY^n}(X^nY^n) - H(XY)\right| \geq \epsilon\right]\right) \\ &\leq 1 - \left(\frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3}\right) \\ &= 1 - \epsilon. \end{aligned}$$

The first inequality is due to the union bound, and the second follows from the inequality in the proof of the previous point, and its analogues. By definition of convergence, the first statement is proven.

2. Observe that by rewriting the last line in the definition of $A_\epsilon^{(n)}$ in **the definition of joint typicality**, one can derive that for every element $(x^n, y^n) \in A_\epsilon^{(n)}$,

$$P_{X^nY^n}(x^ny^n) > 2^{-n(H(XY)+\epsilon)}.$$

The second statement follows by rearranging the following inequality:

$$\begin{aligned} 1 &= \sum_{(x^n,y^n)\in\mathcal{X}^n\times\mathcal{Y}^n} P_{X^nY^n}(x^n,y^n) \\ &\geq \sum_{(x^n,y^n)\in A_\epsilon^{(n)}} P_{X^nY^n}(x^n,y^n) \\ &\geq |A_\epsilon^{(n)}|\cdot 2^{-n(H(XY)+\epsilon)}. \end{aligned}$$

3. First, we write out the probability of sampling an element in the typical set, again by rearranging the inequalities in **the definition of joint typicality**.

   $$\begin{aligned} P[(\tilde{X}^n\tilde{Y}^n)\in A_\epsilon^{(n)}] &= \sum_{(x^n,y^n)\in A_\epsilon^{(n)}} P_{\tilde{X}^n\tilde{Y}^n}(x^n,y^n) \\ &= \sum_{(x^n,y^n)\in A_\epsilon^{(n)}} P_{X^n}(x^n)\cdot P_{Y^n}(y^n) \\ &\leq |A_\epsilon^{(n)}|\cdot 2^{-n(H(X)-\epsilon)}\cdot 2^{-n(H(Y)-\epsilon)}. \end{aligned}$$

   Combining this with part (2) of this theorem, we get

   $$\begin{aligned} P[(\tilde{X}^n\tilde{Y}^n)\in A_\epsilon^{(n)}] &\leq 2^{n(H(XY)+\epsilon)}\cdot 2^{-n(H(X)-\epsilon)}\cdot 2^{-n(H(Y)-\epsilon)} \\ &= 2^{-n(-H(XY)+H(X)+H(Y)-3\epsilon)} \\ &= 2^{-n(I(X;Y)-3\epsilon)}. \end{aligned}$$

   This completes the proof of the third part.
4. By part (1), we have that $P[A_\epsilon^{(n)}]\geq 1-\epsilon$ for large enough n. Thus,

   $$\begin{aligned} 1-\epsilon &\leq P[A_\epsilon^{(n)}] \\ &= \sum_{(x^n,y^n)\in A_\epsilon^{(n)}} P_{X^nY^n}(x^n,y^n) \\ &\leq |A_\epsilon^{(n)}| 2^{-n(H(XY)-\epsilon)}. \end{aligned}$$

   Rearranging this inequality and using the same type of derivation as in the proof of (3), it follows that

   $$\begin{aligned} P[(\tilde{X}^n\tilde{Y}^n)\in A_\epsilon^{(n)}] &= \sum_{(x^n,y^n)\in A_\epsilon^{(n)}} P_{X^n}(x^n)\cdot P_{Y^n}(y^n) \\ &\geq (1-\epsilon)\cdot 2^{n(H(XY)-\epsilon)}\cdot 2^{-n(H(X)+\epsilon)}\cdot 2^{-n(H(Y)+\epsilon)} \\ &= (1-\epsilon)\cdot 2^{-n(I(X;Y)+3\epsilon)}. \end{aligned}$$

Noisy-Channel Theorem: Converse

In the previous section, we showed that any rate strictly below the channel capacity is achievable. Here, we show that one cannot do better: rates strictly above the channel capacity are not achievable. Specifically, codes with such rates suffer from non-negligible error probabilities.

Theorem: Shannon's noisy-channel coding theorem (converse)

On a discrete memoryless channel with capacity C, any code with rate $R > C$ has average probability of error $p_e^{(n)} \geq 1 - \frac{C}{R} - \frac{1}{nR}$.

Proof

For a code with rate $R > C$, let W be uniformly distributed over all possible messages, let X^n describe the encoding of the message (and the input to the channel), let Y^n describe the output of the channel, and $\hat{W}$ the decoding of that output.

The average probability of error, $p_e^{(n)}$, is equal to $P[W \neq \hat{W}]$, the probability that the original message differs from the decoded message. Note that $W \to X^n \to Y^n \to \hat{W}$ forms a Markov chain.

As a first step, we show that the mutual information between the message W and the channel output Y^n is upper bounded by $n \cdot C$, that is, there is a limit to the amount of information that can get through the channel. To see this, first observe that

$$\begin{aligned}
H(Y^nW) &= H(Y^nW) + H(X^n \mid Y^nW) && \text{(since } W \text{ determin} \\
&= H(X^nY^nW) && \text{(chai} \\
&= H(X^{n-1}Y^{n-1}W) + H(Y_n \mid X^nY^{n-1}W) + H(X_n \mid X^{n-1}Y^{n-1}W) && \text{(chai} \\
&= H(X^{n-1}Y^{n-1}W) + H(Y_n \mid X^nY^{n-1}W) && \text{(since } W \text{ determin} \\
&= H(X^{n-1}Y^{n-1}W) + H(Y_n \mid X_n) && \text{(memo} \\
&= \ldots && \text{(r} \\
&= H(W) + \sum_{i=1}^{n} H(Y_i \mid X_i).
\end{aligned}$$

Therefore,

$$\begin{aligned}
I(W;Y^n) &= H(W) + H(Y^n) - H(Y^nW) && \text{(entropy diagram)} \\
&= H(Y^n) - \sum_{i=1}^{n} H(Y_i \mid X_i) && \text{(by the above derivation)} \\
&\leq \sum_{i=1}^{n} H(Y_i) - \sum_{i=1}^{n} H(Y_i \mid X_i) \\
&= \sum_{i=1}^{n} I(X_i;Y_i) \\
&\leq n \cdot C.
\end{aligned}$$

Now that we have established that $I(W;Y^n)$ is upper bounded by $n \cdot C$, we can show that the code with rate R induces a considerable error probability:

$$\begin{aligned}
R &= \frac{\log |\mathcal{W}|}{n} \\
&= \frac{1}{n} H(W) \\
&= \frac{1}{n}(H(W \mid Y^n) + I(W; Y^n)) \\
&\leq \frac{1}{n}(H(W \mid Y^n) + n \cdot C) \\
&\leq \frac{1}{n}\left(1 + P[W \neq \hat{W}] \cdot n \log |\mathcal{W}| + n \cdot C\right) \\
&= \frac{1}{n} + P[W \neq \hat{W}] \cdot R + C,
\end{aligned}$$

where the second inequality is an application of **Fano's inequality**. Dividing both sides by R and rearranging, we get the desired inequality:

$$p_e^{(n)} = P[W \neq \hat{W}] \geq 1 - \frac{C}{R} - \frac{1}{nR}.$$

This theorem shows that if Alice and Bob try to communicate using a code with a rate $R > C$, their probability of error will be bounded away from zero by a constant factor of $1 - \frac{C}{R}$ (for big n, the last term in the inequality becomes insignificant). This error probability worsens for a bigger difference between R and C.

Source-Channel Separation Theorem: Converse

Conversely, if a source V has entropy $H(V)$ that exceeds the channel capacity C, then it is impossible to transmit the source over the channel with arbitrarily small error.

Theorem: Source-channel separation theorem (converse)

Let $V_1, V_2, \ldots, V_n$ be i.i.d. random variables (the source) distributed according to some P_V. Let $(\mathcal{X}, P_{Y|X}, \mathcal{Y})$ be a discrete memoryless channel with capacity C. If $H(V) > C$, then any source-channel code has an error probability that is bounded away from zero.

Proof

Assume the error probability $p_e := P[\hat{V}^n \neq V^n]$ *does* converge to zero as n goes to infinity. We show that $H(V) \leq C$, in order to prove the theorem by contraposition.

$$\begin{aligned} H(V) &= \frac{H(V^n)}{n} && \text{(since the source is i.i.d.)} \\ &= \frac{1}{n}(H(V^n|\hat{V}^n) + I(V^n;\hat{V}^n)) && \text{(by entropy diagrams)} \\ &\leq \frac{1}{n}(1 + p_e \log(|\mathcal{V}|^n) + I(V^n;\hat{V}^n)) && \text{(by Fano's inequality)} \\ &= \frac{1}{n} + p_e \log(|\mathcal{V}|) + \frac{1}{n} I(V^n;\hat{V}^n) \\ &\leq \frac{1}{n} + p_e \log(|\mathcal{V}|) + \frac{1}{n} I(X^n;Y^n) && \text{(by the data-processing inequality)} \\ &\leq \frac{1}{n} + p_e \log(|\mathcal{V}|) + C && \text{(by the lemma on multiple channel uses)} \\ &\to 0 + 0 + C = C, \end{aligned}$$

as n goes to infinity.

We have shown the source-channel separation theorem (the forward direction and its converse) for discrete, memoryless channels, and i.i.d. sources. It also holds for channels with feedback, and any sources that satisfy the asymptotic equipartition property.